Computers & Geosciences Vol. 17, No. 9, pp. 1311–1343, 1991
Printed in Great Britain

0098-3004/91 $3.00 + 0.00
Pergamon Press plc

EXSHALL: A TURKEL–ZWAS EXPLICIT LARGE TIME-STEP FORTRAN PROGRAM FOR SOLVING THE SHALLOW-WATER EQUATIONS IN SPHERICAL COORDINATES*

I. M. NAVON[1,2] and JIAN YU[1]
[1]Department of Mathematics, and [2]Supercomputer Computations Research Institute, Florida State University, Tallahassee, FL 32306, U.S.A.

(Received 3 July 1990; revised 15 August 1991)

Abstract—A FORTRAN computer program is presented and documented applying the Turkel–Zwas explicit large time-step scheme to a hemispheric barotropic model with constraint restoration of integral invariants of the shallow-water equations. We then proceed to detail the algorithms embodied in the code EXSHALL in this paper, particularly algorithms related to the efficiency and stability of T–Z scheme and the quadratic constraint restoration method which is based on a variational approach. In particular we provide details about the high-latitude filtering, Shapiro filtering, and Robert filtering algorithms used in the code. We explain in detail the various subroutines in the EXSHALL code with emphasis on algorithms implemented in the code and present the flowcharts of some major subroutines. Finally, we provide a visual example illustrating a 4-day run using real initial data, along with a sample printout and graphic isoline contours of the height field and velocity fields.

Key Words: Shallow-water equations, Spherical coordinates, Explicit finite-differences, Constraint restoration, Filtering techniques.

INTRODUCTION

Recently, a considerable amount of work has been dedicated and aimed at efficient integration of shallow-water equations in view of using these methods in numerical weather prediction models. In order to achieve computational accuracy and efficiency, most methods are concerned with the different time-scale of the advection and the gravity-inertia terms in the shallow-water equations model separately. Semi-implicit schemes (Robert, 1979; Burridge, 1975) and split-explicit schemes (Magazenkov, Shvets, and Shneyerov, 1971; Gadd, 1978a, 1978b) are examples of those methods. In the split-explicit schemes, a substantial computational economy is achieved when compared to usual explicit time integration schemes.

Turkel and Zwas (1979) proposed a space-splitting rather than a time-splitting algorithm for the explicit integration of the shallow-water equations. Their method is based on the fact that the fast gravity-inertia waves contain only a small fraction of the total available energy and therefore these waves can be calculated with a lower accuracy than the slow Rossby waves, that is on a coarser mesh. An application of the T–Z space split-explicit integration schemes with real initial data is presented and discussed by Navon and de Villiers (1987) and its properties are discussed further in Neta and Navon (1989). A linear transfer function analysis of the shallow-water equations in spherical coordinates for the Turkel–Zwas explicit large time-step scheme was carried out by Neta, Navon, and Yu (1990).

The purpose of this paper is to present a practical FORTRAN code, EXSHALL, which implements the T–Z explicit large time-step scheme for the shallow-water equations in spherical coordinates along with constraint restoration methods for enforcing a posteriori conservation of the integral invariants of the shallow-water equations. The computer program is explained in detail in connection with the various algorithms implemented in the code EXSHALL. This paper can be used as a user's guide to the program EXSHALL both in providing a brief description of the theory as well as detailed programming implementation.

We present here the T–Z scheme for the shallow-water equations in spherical coordinates and its related algorithmic background. Various filters used with T–Z scheme and which impact on its stability also are presented; a detailed description of the various subroutines in the code EXSHALL is presented; and a typical example of a 4-day run with the program EXSHALL is presented along with graphical output. Finally, in the Appendix the commented and documented FORTRAN source listing the code of the program EXSHALL is attached.

*Geophysical Fluid Dynamic Institute Contribution No. 329.

TURKEL–ZWAS EXLICIT LARGE TIME-STEP SCHEME

Shallow-water Equations in Spherical Coordinates

The shallow-water equations in spherical coordinates are given by

$$\frac{\partial u}{\partial t}+\frac{1}{a\cos\theta}\left[u\frac{\partial u}{\partial\lambda}+v\cos\theta\frac{\partial u}{\partial\theta}\right]-\left(f+\frac{u}{a}\tan\theta\right)v+\frac{g}{a\cos\theta}\frac{\partial h}{\partial\lambda}=0 \quad (1)$$

$$\frac{\partial v}{\partial t}+\frac{1}{a\cos\theta}\left[u\frac{\partial v}{\partial\lambda}+v\cos\theta\frac{\partial v}{\partial\theta}\right]+\left(f+\frac{u}{a}\tan\theta\right)u+\frac{g}{a}\frac{\partial h}{\partial\theta}=0 \quad (2)$$

$$\frac{\partial h}{\partial t}+\frac{1}{a\cos\theta}\left[\frac{\partial}{\partial\lambda}(hu)+\frac{\partial}{\partial\theta}(hv\cos\theta)\right]=0. \quad (3)$$

Here f is the Coriolis parameter given by

$$f=2\Omega\sin\theta. \quad (4)$$

Ω is the angular speed of the rotation of the Earth, h is the height of the homogeneous atmosphere, u and v are the zonal and meridional wind components respectively, whereas the latitudinal and longitudinal directions are given by θ and λ respectively, g is the acceleration of gravity, and a is the radius of the Earth.

Turkel–Zwas Explicit Large Time-step Scheme

In the Cartesian situation, the stability condition for the leap-frog explicit finite difference time integration of the shallow-water equations requires

$$\frac{\Delta t}{\Delta x}\leqslant\frac{1}{|u|+|v|+\sqrt{2gh}}. \quad (5)$$

For typical meteorological conditions one has

$$\sqrt{gh}\gg(|u|+|v|) \quad (6)$$

wheareas most of the energy is carried out at the convective speed O $(|u|+|v|)$. Here Δt and Δx are the time and space mesh sizes, respectively.

Turkel and Zwas (1979) calculated the terms related to the gravity-waves, namely (gh_x+fv) and (gh_y+fu) in the continuity equation on a coarser mesh than the advective terms and applied a Padé compact fourth-order accurate finite-difference approximation for their calculation, to compensate for the higher truncation error on the coarser grid.

It has been shown (Navon and de Villiers, 1987) that the stability condition for T–Z scheme with leap-frog explicit time differencing is

$$\frac{\Delta t}{\Delta x}\leqslant\frac{1}{|u|+|v|+\sqrt{2gh/p}} \quad (7)$$

(see Turkel and Zwas, 1979; Navon and de Villiers, 1987).

In the spherical situation, the stability condition for the shallow-water equations model assumes the form

$$\Delta t_{\max}=\frac{a\cos\Delta\lambda}{\sqrt{gH}\omega_{\max}} \quad (8)$$

where $\omega=\sin(k\Delta\lambda)$, k is the zonal wave number. This stability condition is obtained by using an analysis similar to that of Arakawa and Lamb (1977) for the linearized shallow-water equations in spherical coordinates (see also Takacs and Balgovind, 1983).

The Turkel–Zwas scheme for the nonlinear shallow-water equations in spherical coordinates takes the following form:

$$\begin{aligned}u^{n+1}_{i,j}=u^n_{i,j}&-\sigma\left[\frac{u^n_{i,j}}{\cos\theta_j}(u^n_{i+1,j}-u^n_{i-1,j})+v^n_{i,j}(u^n_{i,j+1}-u^n_{i,j-1})+\frac{g}{p\cos\theta_j}(h^n_{i+p,j}-h^n_{i-p,j})\right]\\&+2\Delta t\left[(1-\alpha)\left(2\Omega\sin\theta_j+\frac{u^n_{i,j}}{a}\tan\theta_j\right)v^n_{i,j}+\frac{\alpha}{2}\left(2\Omega\sin\theta_j+\frac{u^n_{i+p,j}}{a}\tan\theta_j\right)v^n_{i+p,j}\right.\\&\left.+\frac{\alpha}{2}\left(2\Omega\sin\theta_j+\frac{u^n_{i-p,j}}{a}\tan\theta_j\right)v^n_{i-p,j}\right] \quad (9)\end{aligned}$$

$$\begin{aligned}v^{n+1}_{i,j}=v^n_{i,j}&-\sigma\left[\frac{u^n_{i,j}}{\cos\theta_j}(v^n_{i+1,j}-v^n_{i-1,j})+v^n_{i,j}(v^n_{i,j+1}-v^n_{i,j-1})+\frac{g}{q}(h^n_{i,j+q}-h^n_{i,j-q})\right]\\&-2\Delta t\left[(1-\alpha)\left(2\Omega\sin\theta_j+\frac{u^n_{i,j}}{a}\tan\theta_j\right)u^n_{i,j}+\frac{\alpha}{2}\left(2\Omega\sin\theta_{j+q}+\frac{u^n_{i,j+q}}{a}\tan\theta_{j+q}\right)u^n_{i,j+q}\right.\\&\left.+\frac{\alpha}{2}\left(2\Omega\sin\theta_{j-q}+\frac{u^n_{i,j-q}}{a}\tan\theta_{j-q}\right)u^n_{i,j-q}\right] \quad (10)\end{aligned}$$

$$\begin{aligned}h^{n+1}_{i,j}=h^n_{i,j}&-\alpha\left\{\frac{u^n_{i,j}}{\cos\theta_j}(h^n_{i+1,j}-h^n_{i-1,j})+v^n_{i,j}(h^n_{i,j+1}-h^n_{i,j-1})\right.\\&+\frac{h^n_{i,j}}{\cos\theta_j}\left[(1-\alpha)(u^n_{i+p,j}-u^n_{i-p,j})+\frac{\alpha}{2}(u^n_{i+p,q}-u^n_{i-p,j+q}+u^n_{i+p,j+q}-u^n_{i-p,j-q})\right]\frac{1}{p}\\&+\left[(1-\alpha)(v^n_{i,j+q}\cos\theta_{j+q}-v^n_{i,j-q}\cos\theta_{j-q})+\frac{\alpha}{2}(v^n_{i+p,j-q}\cos\theta_{j+q}-v^n_{i+p,j-q}\cos\theta_{j-q}\right.\\&\left.\left.+v^n_{i-p,j+q}\cos\theta_{j+q}-v^n_{i-p,j-q}\times\cos\theta_{j-q})\right]\frac{1}{q}\right\} \quad (11)\end{aligned}$$

where

$$\sigma = \frac{\Delta t}{a\Delta\lambda} = \frac{\Delta t}{a\Delta\theta}. \qquad (12)$$

For $\alpha = 1/3$ the geostrophic balance and the incompressibility condition are satisfied to a higher order in the Cartesian coordinate situation, this being exactly the fourth-order Padé finite-difference approximation (see Turkel and Zwas, 1979; Navon and de Villiers, 1987). The stability condition of the T–Z scheme (9)–(12) for the Padé weighting parameter, α, is $\alpha < 1/2$ (see Neta, Navon, and Yu, 1990). p and q are parameters of the coarse mesh ratio to the fine mesh.

Filters Used in the Code EXSHALL

Fourier filtering for polar regions

Near the poles the longitudinal distances between neighboring points $\Delta x = a \cos\theta\Delta\lambda$ decrease as one approaches the poles for a fixed $\Delta\lambda$.

Owing to these short distances and to the presence of fast moving inertia-gravity waves near the poles, prohibitively short time-steps are required to ensure computational stability. Different Fourier filtering or high-latitude filtering methods have been put forward for latitude–longitude global gridpoint models to allow the use of large time-steps for explicit time integration. For a comprehensive survey of this topic see Takacs and Balgovind (1983).

In the code EXSHALL, we have used the Arakawa and Lamb (1977) Fourier high-latitude filtering method in which the zonal pressure gradient and zonal mass flux terms are Fourier filtered. For the Turkel–Zwas scheme differencing gravity-wave related terms on a coarse grid over points p meshes away in the longitudinal direction results in a typical stability condition of the form

$$\frac{(gH)^{1/2}}{a\cos\theta}\sin\frac{(pk\Delta\lambda)}{p\Delta\lambda}\Delta t \leqslant 1. \qquad (13)$$

The increase in the maximum allowable time-step is the result of the differencing of gravity-wave related terms on a coarse mesh has several implications for high-latitude filtering. On one hand, on the coarse mesh, the frequency of the fastest resolved propagating gravity mode, w_g decreases, but on the other hand, we can use a larger time-step.

As has been shown by Daley (1980) and Takacs, Kalnay, and Navon (1985), the condition for an explicit time-differencing scheme for the shallow-water equations on the sphere to be stable linearly is

$$|w_g| \leqslant \frac{1}{\Delta t}. \qquad (14)$$

The set of all gravity modes whose eigenfrequencies satisfy

$$|w_g| > \frac{1}{\Delta t} \qquad (15)$$

constitutes the set of "fast" modes which will have to be Fourier filtered to maintain stability near the poles.

The coarse mesh differencing reduces the size of the fast modes set, but the use of a larger time-step again increases the number of modes in the "fast" set, as defined by Daley (1980), meaning that as far as high-latitude filtering is concerned the same amount of effort will be required to maintain computational stability.

In all techniques used, the filtered quantities were fast-Fourier transformed into their wave components whose amplitudes then were altered by a wave-dependent damping function. In the code EXSHALL, the damping function assumes the form:

$$F_j = \min\left(1, \frac{\cos\theta_j}{\sin(k\Delta\lambda)}\right). \qquad (16)$$

Shapiro low-pass filtering

Spatial finite difference schemes which are not enstrophy conserving nor implicitly damping require global filtering of short waves ($2\Delta x$ to $4\Delta x$) to eliminate the build-up of energy in the shortest wavelengths resulting from nonlinear aliasing. Global filtering is applied in the code EXSHALL, using a 17-point Shapiro filter (Shapiro, 1979) on the geopotential height field as well as on the velocity field components u and v.

The form of the two-dimensional 17-point Shapiro filter consists of two passes of a one-dimensional filter applied in succession (Takacs, 1986, p. 15). To obtain the high order of the filter, eight passes of the 3-point second-order operator are used as given by

$$q_{ij}^{sh} = [(1 - (F_\theta^2)^8][1 - (F_\lambda^2)^8]q_{ij} \qquad (17)$$

where

$$F_\lambda^2(q_{i,j}) = (q_{i+1,j} - 2q_{i,j} + q_{i-1,j})/4 \qquad (18)$$

$$F_\theta^2(q_{i,j}) = (q_{i,j+1} - 2q_{i,j} + q_{i,j-1})/4. \qquad (19)$$

Here, q is any quantity being filtered. In our situation, q would consist of the fields of either h, u, or v.

Robert filtering for leapfrog time integration

The Robert filter is used in connection with the leapfrog time integration scheme (Robert, 1966; see also Haltiner and Williams, 1980, p. 145) in order to suppress the computational component inherent in this three-level explicit time-differencing scheme, as follows. First one integrates in time using the leapfrog scheme,

$$F^{n+1} = F^{*n-1} + 2\Delta t(\partial F/\partial t)^n \qquad (20)$$

then one averages for three terms to obtain F^{*n},

$$F^{*n} = F^n + \gamma(F^{n+1} - 2F^n + F^{*n-1}) \qquad (21)$$

where F^{*n} is the Robert-filtered field.

The Constraint Restoration Method

The idea of enforcing "a posteriori" integral invariants conservation has been pursued by Sasaki (1976, 1977), as well as Isaacson (1977). These ideas have been tested by Navon (1987). Miele and Heideman (1968) and Miele, Heideman, and Damoulakis (1969) proposed a constraint restoration method based on a least-square change in the coordinates of the state-vector. A new approach based on an augmented Lagrangian combined penalty-multiplier method related to equality-constrained optimization has been proposed by Navon and de Villiers (1983, 1987).

Some algorithmic differences as well as the equivalence between the Bayliss–Isaacson algorithm and the constraint restoration method are discussed by Navon (1987). The method implemented in the code EXSHALL is the constraint restoration method (CRM) which will be presented briefly in the following content.

The idea guiding the constraint restoration method is to make a modification of the smallest norm to the value predicted by the finite difference scheme at a given time step, such that certain integral invariants of the solution remain almost unchanged during the time integration. In our situation, to be specific, we let

$$x = (u^n_{11}, \ldots, u^n_{N_xN_y}, v^n_{11}, \ldots, v^n_{N_xN_y}, h^n_{11}, \ldots, h^n_{N_xN_y}) \qquad (22)$$

which is the approximate solution at time $t = n\Delta t$ by leapfrog time differencing integration of the Turkel–Zwas scheme. We want to determine an optimal point x^* such that

$$\Phi(x^*) = \begin{bmatrix} \phi_1(x^*) \\ \phi_2(x^*) \\ \phi_3(x^*) \end{bmatrix} = \begin{bmatrix} H(x^*) - H^0 \\ Z(x^*) - Z^0 \\ E(x^*) - E^0 \end{bmatrix} = 0, \qquad (23)$$

where $H(x^*), Z(x^*), E(x^*)$ are the total mass, enstrophy, and energy respectively at the point x^* which satisfies the equality constraints. H^0, Z^0, E^0 are the initial total mass, enstrophy, and total energy integral invariants respectively. In the spherical situation, our definitions of $H(x)$, $Z(x)$, and $E(x)$ are, respectively

$$H(x) = \frac{d \sin(d/2)}{\pi} \Sigma_{ij} h_{ij} \cos\theta_j \qquad (24)$$

$$Z(x) = \frac{ad^2}{2} \Sigma_{ij} \frac{\cos\theta_j}{h_{ij}} \times \left[\frac{v_{i+1,j} - v_{i-1,j}}{2ad\cos\theta_j} - \frac{u_{i,j+1} - u_{i,j-1}}{2ad} + f_j \right]^2 \qquad (25)$$

$$E(x) = \frac{(ad)^2}{2} \Sigma_{ij} (u^2_{ij} + v^2_{ij} + gh_{ij}) h_{ij} \cos\theta_j \qquad (26)$$

where

$$d = \Delta\theta = \Delta\lambda. \qquad (27)$$

The term $d\sin(d/2)$ in H comes from a spherical area average where a is the radius of the Earth. Let us term x the normal point not consistent with the constraint (23), and let x^v denote a varied point. The presented idea can be summarized as a minimization problem,

$$\min(\tfrac{1}{2}\|x^v - x\|^2_{L_2}). \qquad (28)$$

Subject to

$$\Phi(x^v) = 0. \qquad (29)$$

Quasilinearizing the constraint, we obtain:

$$\Phi(x) + A^T(x)\delta x = 0 \qquad (30)$$

where $\delta x = x^v - x$ and $A(x)$ is the Jacobian matrix of the function $\Phi(x)$.

$$A(x) = \begin{bmatrix} \partial\phi_1/\partial x_1 & \partial\phi_2/\partial x_1 & \partial\phi_3/\partial x_1 \\ \partial\phi_1/\partial x_2 & \partial\phi_2/\partial x_2 & \partial\phi_3/\partial x_2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \partial x_1/\partial x_3 & \partial\phi_2/\partial x_3 & \partial\phi_3/\partial x_3 \end{bmatrix} \qquad (31)$$

In our situation we have

$$\frac{\partial\phi_1}{\partial h_{ij}} = \frac{\partial H}{\partial h_{ij}} = \frac{d\sin(d/2)}{\pi}\cos\theta_j \qquad (32)$$

$$\frac{\partial\phi_1}{\partial u_{ij}} = \frac{\partial\phi_1}{\partial v_{ij}} = 0 \qquad (33)$$

$$\frac{\partial\phi_2}{\partial u_{ij}} = \frac{ad}{2}\left[\frac{\cos\theta_{j+1}}{h_{i,j+1}} \left(\frac{v_{i+1,j+1} - v_{i-1,j+1}}{2ad\cos\theta_{j+1}} + \frac{u_{i,j+2} - u_{i,j}}{2ad} + f_{j+1} \right) - \frac{\cos\theta_{j-1}}{h_{i,j-1}} \left(\frac{v_{i+1,j-1} - v_{i-1,j-1}}{2ad\cos\theta_{j-1}} - \frac{u_{i,j} - u_{i,j-2}}{2ad} + f_{j-1} \right) \right] \qquad (34)$$

$$\frac{\partial\phi_2}{\partial v_{ij}} = \frac{ad}{2}\left[\frac{\cos\theta_{j+1}}{h_{i-1,j}} \left(\frac{v_{i,j} - v_{i-2,j}}{2ad\cos\theta_j} - \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2ad} + f_j \right) - \frac{\cos\theta_j}{h_{i+1,j}} \left(\frac{v_{i+2,j} - v_{i,j}}{2ad\cos\theta_j} - \frac{u_{i+1,j+1} - u_{i+1,j-1}}{2ad} + f_j \right) \right] \qquad (35)$$

$$\frac{\partial \phi_2}{\partial h_{i,j}} = -\frac{a^2 d^2 \cos\theta_j}{2\,h_{i,j}}\left[\frac{v_{i+1,j}-v_{i-1,j}}{2ad\cos\theta_j} - \frac{u_{i,j+1}-u_{i,j-1}}{2ad} + f_j\right]^2 \tag{36}$$

$$\frac{\partial E}{\partial u_{i,j}} = \frac{\partial \phi_3}{\partial u_{i,j}} = a^2 d^2 u_{i,j} h_{i,j}\cos\theta_j \tag{37}$$

$$\frac{\partial E}{\partial v_{i,j}} = \frac{\partial \phi_3}{\partial v_{i,j}} = a^2 d^2 v_{i,j} h_{i,j}\cos\theta_j \tag{38}$$

$$\frac{\partial E}{\partial h_{i,j}} = \frac{\partial \phi_3}{\partial h_{i,j}} = \frac{a^2 d^2}{2}(u_{i,j}^2 + v_{i,j}^2 + 2gh_{i,j})\cos\theta_j. \tag{39}$$

As shown in Miele and Heideman (1968), by using standard methods of the theory of maxima and minima, the functional to be minimized is

$$F = \tfrac{1}{2}\delta x^T \delta x + \lambda^T[\alpha\Phi(x) + \mathbf{A}^T(x)\,\delta x] \tag{40}$$

where α is a prescribed restoration step-size factor in the range $0 \leqslant \alpha \leqslant 1$, which prevents the perturbation δx from becoming too large.

The following algorithm of the constraint restoration method is used in the code EXSHALL (Navon and de Villiers, 1987),

(1) Assume a nominal point x.

(2) At the nominal point compute the constraint vector $\Phi(x)$, the matrix $\mathbf{A}(x)$ with Equation (31) and matrix $\mathbf{B} = \mathbf{A}^T(x)\mathbf{A}(x)$ with Equation (31), as well as the performance index P which is given by

$$P = \Phi^T(x)\Phi(x). \tag{41}$$

(3) Assume the restoration of step-size $\alpha = 1$ and determine δx, using the equation

$$\delta x = \alpha\mathbf{A}(x)\mathbf{B}^{-1}(x)\Phi(x). \tag{42}$$

(4) Compute the varied point x^v by

$$x^v = x + \delta x. \tag{43}$$

(5) At the varied point compute the performance index P^v. If $P^v < P$, the first iteration is completed and the restoration step-size $\alpha = 1$ is acceptable. If this inequality is violated, that is $P^v > P$, instead of conducting a step-size search, Miele, Heidemann, and Damoulakis (1969a, 1969b) and Miele, Levy, and Cragg (1971) propose a bisection process, that is α is bisected several times until the condition

$$P^v < P \tag{44}$$

is met first. That this happens is guaranteed by the descent property.

(6) After a value of α in the range $0 \leqslant \alpha \leqslant 1$ has been determined such that $P^v < P$, the first iteration is completed. The point $x^v = x + \delta x$ is employed as the new nominal point x for the second iteration; this procedure then is repeated until a desired degree of accuracy is obtained, namely until the performance index satisfies the inequality

$$P \leqslant \varepsilon \tag{45}$$

where ε is a small number.

In our situation $\varepsilon = 10^{-10}$, that is we continued iterations until

$$P \leqslant 10^{-10}. \tag{46}$$

PROGRAM EXSHALL

Initial Conditions and Test Problem

The integration domain in the code EXSHALL consists of the northern hemisphere. Our hemispheric mesh is a mesh of (128 × 32) grid points, corresponding to a spacing of 2.8135° in $\Delta\lambda$ and $\Delta\theta$. The initial velocity fields are generated geostrophically using the initial geopotential height field $h_{i,j}^0$. The initial height used in the sample run is taken from a realistic data set of 1 January 1979, 00Z (taken from the FGGE data set). A bicubic interpolation is used to interpolate the realistic data set from the original FGGE grid to our hemispheric grid resolutions, which is the initial height field $h_{i,j}^r$ saved in a file.

Let $h_{i,j}^r$ be the initial height read from an auxiliary storage, the initial horizontal velocity components $u_{i,j}^0, v_{i,j}^0$ are generated geostrophically using

$$u_{i,j}^0 = \frac{-g}{af}\,\frac{h_{i+1,j+1}^r + h_{i,j+1}^r - h_{i+1,j}^r - h_{i,j}^r}{2\Delta\theta} \tag{47}$$

$$v_{i,j}^0 = \frac{g}{a\cos\theta_{j+1/2} f}\,\frac{h_{i+1,j+1}^r - h_{i,j+1}^r + h_{i+1,j}^r - h_{i,j}^r}{2\Delta\lambda}. \tag{48}$$

Then a 10-point interpolation is used to interpolate $h_{i,j}^0$ on the same points as $u_{i,j}^0, v_{i,j}^0$ are evaluated. Two passes of a one-dimensional interpolation are applied, first interpolating along longitude lines, then along latitude lines.

The first pass of the interpolation assumes the form:

$$h_{i,j}^* = \sum_{i=1}^{10} P_i h_{i,j}^r. \tag{49}$$

The second pass is

$$h_{i,j}^0 = \sum_{i=1}^{10} P_j h_{i,j}^*. \tag{50}$$

Here

$$P_i = \prod_{j=1, j\neq i}^{10}\left(\frac{j-5.5}{j-i}\right), \quad 1 \leqslant i \leqslant 10. \tag{51}$$

Computer Implementation

Structure of the program EXSHALL

Input specifications. The input to the program consists of a single data card of format (8F16.11). It reads the initial geopotential height from a file named "hfield". All the constants used in the code are initialized in the beginning of the program.

Preprocessing. Before starting the integration of the T–Z scheme, we require initial conditions for the velocity field. The initial velocity components $u_{i,j}^{0}$, $v_{i,j}^{0}$ are generated geostrophically using the initial height field $h_{i,j}^{0}$. The algorithm is explained in detail in the subroutine GEOWND. The subroutine SCALE scales down the initial conditions and other constants used, as the result of the vastly different physical units, the integral constraints would assume mainly different orders of magnitude and slow down the minimization process because of ill-conditioning (see Navon and de Villiers, 1983).

Postprocessing. There are two subroutines, MAPPA and PLOT, which plot the approximate solutions every 24 h. MAPPA produces a printer-plot contour of the height field (see Fig. 1). Subroutine PLOT produces graphic contours of the height field and graphic vector velocity fields using the NCAR graphics package (see Figs. 2–10). This subroutine can be modified easily to use any other graphic package available. If a user desires to observe the behavior of the integral invariants evolution, he can look at the file named "rate" which saves the relative integral invariants at each time step (see Fig. 11).

The main program EXSHALL. The main program initializes the constants used in the code, reads the initial geopotential height, then carries out the time integration of the Turkel–Zwas large time-step explicit scheme by leapfrogging. It then carries out a postprocessing phase by calling the subroutine DAYEND every 24 h. For the detailed relationship between the main program and the various subroutines, see Figure 12.

The common blocks used in main program are explained in the following context. The resolution implemented in the code is 128 grid points in the longitudinal direction and 32 grid points in the latitudinal direction. Because of the restriction on the number of the grid points required by the subroutine FFT which implements the Fast Fourier Transform, the number of grid points taken in both longitudinal and latitudinal directions must an integer power of 2^m where m is a positive integer.

The arrays XE, XO are of dimension $N_x \times N_y$ where N_x, N_y are the number of grid points in longitudinal and latitudinal directions, respectively. In the code, $N_x = 2^7, N_y = 2^5$. XA is an array of dimension $N_x \times (N_y + 1)$. The arrays XE, XO, XA are used for storing field variables, namely, $u_{i,j}$, $v_{i,j}$, and $h_{i,j}$, respectively (Fig. 13). The storage layout of the arrays XE, XO, XA is displayed in Figure 14.

The vectors XO and XE store the integrated solutions at even and odd time steps, respectively. The array XA is used for the solution vector after performing the Robert filtering.

The common block C0 stores the initial integral invariants, namely initial total mass CH0, initial total enstrophy CZ0 and initial total energy CE0. The common block CA stores some constants used in the code. D is the $2\Delta\theta$, G is the Earth's gravity acceleration, A is the radius a of the Earth. DS is equal to $4\Delta\theta a$, IP, IQ are the mesh sizes in the latitudinal and longitudinal directions respectively. ALF is the weighting of the Padé compact finite-difference approximation, and DT is the scaled time-step size.

The common block CC just as C0 stores integral invariants of the shallow-water equations model at the most recent integration time-step. The common block CG stores two counters. NG counts the integral invariants saved while IG counts data points saved for the graphics package. The common block CT stores while NRP, the number of points where constraint restoration of the integral invariants was carried out, TIME is the scaled accumulated time of integration. The common block CW is a working storage used throughout the entire code by several subroutines for memory saving.

The array IDT stores different time-step sizes in seconds for the automatic time-step selection according to the mesh size IP in latitude.

Subroutine SMTHFLD (A, G). This subroutine performs Fourier filtering for high latitude, that is for latitudes of 55° and higher. On entry, array A contains the field to be smoothed out using Fourier filtering. A could be the geopotential height field, or the velocity field components, u or v. Array G is just a working storage of dimension N_x. The subroutine smooths out a given field row by row (or longitude by longitude) using the Fast Fourier Transform algorithm. For details, see Figure 15.

Subroutine MINMAX (H). This subroutine locates and prints out the minimum and maximum of the height field for the users reference. On entry, array H contains the height field. On exit, H is unchanged.

Subroutine GEOWND (X, FS). This subroutine computes velocity components u, v geostrophically at intermediate points using Equations (47)–(51). On entry, the array XA in high position contains the height field. Array FS contains the Coriolis parameters. On exit, FS is unchanged, while the array XA in low position contains the velocity field components u, v.

Subroutine MAPPA (H, C, NX, NY). This subroutine produces a printer-plot contours (see Fig. 1) of the height field on the line printer. On entry, the array H contains the height field. The constant C controls the contour intervals required by the user. The integers NX, NY are N_x, N_y, respectively, that is the number of grid points in the longitudinal and latitudinal directions, respectively.

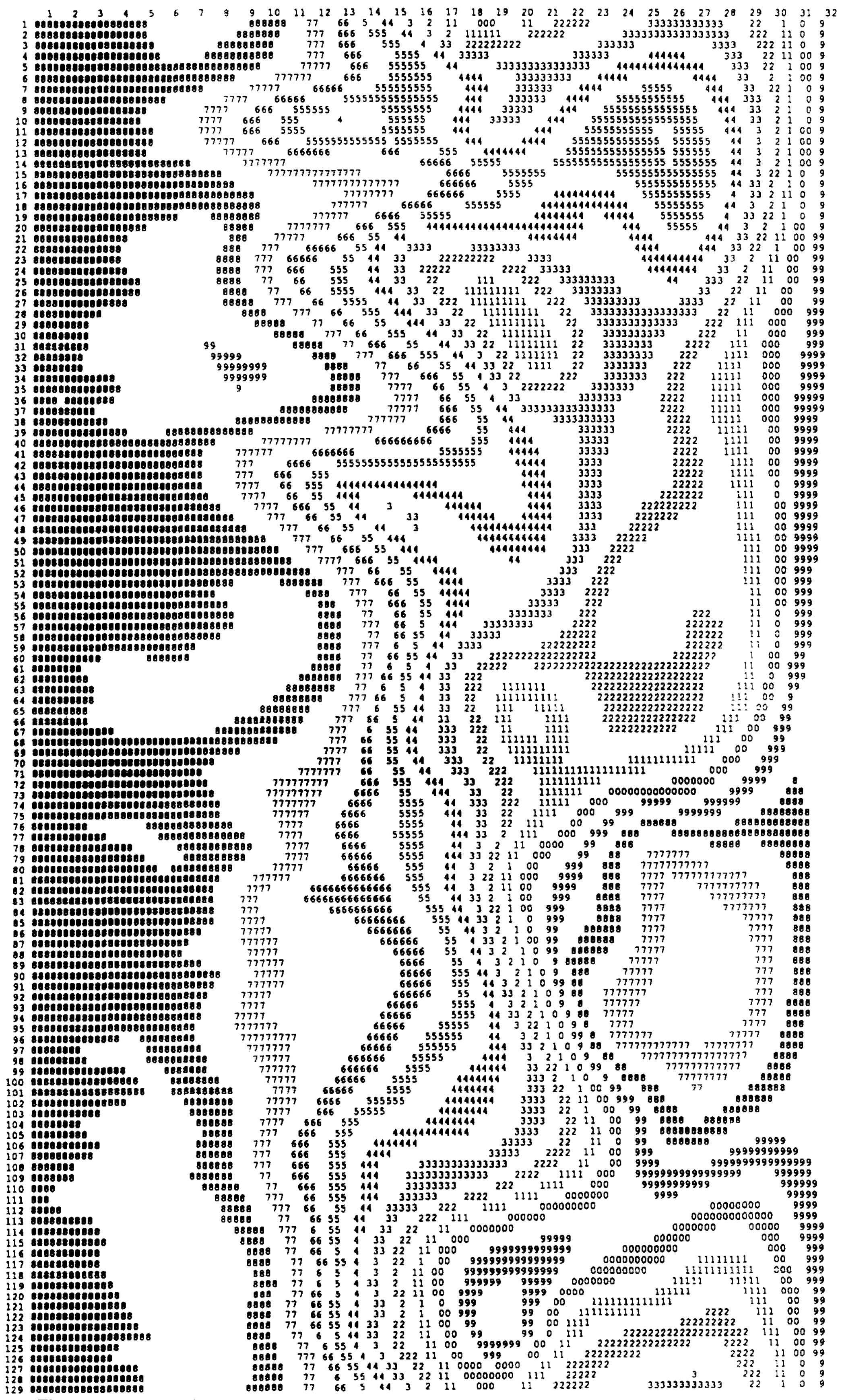

Figure 1A. Printer-plot contours of initial geopotential height field, contoured in intervals of 100 m, from 4800 to 5600 m.

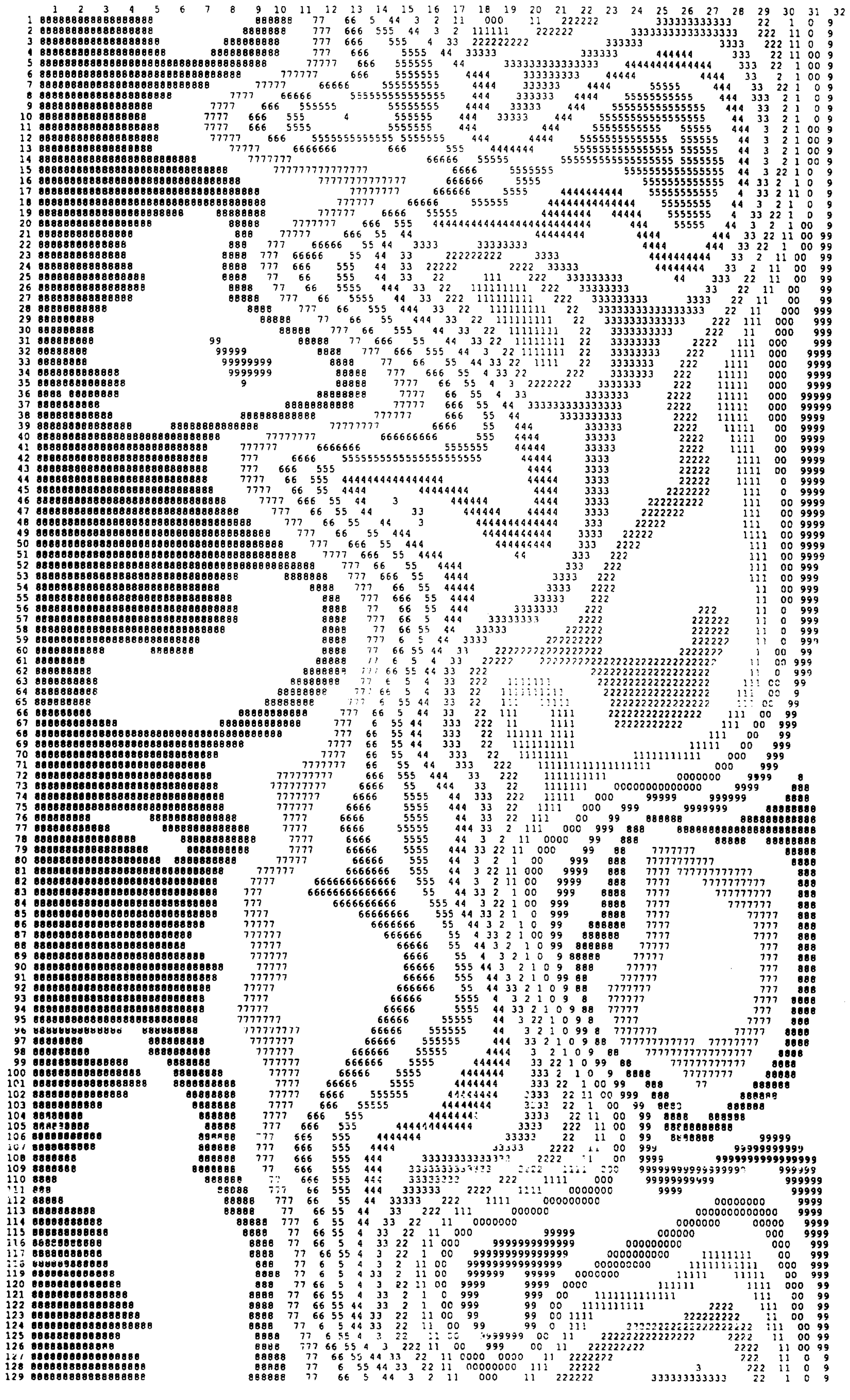

Figure 1B. Initial geopotential height field.

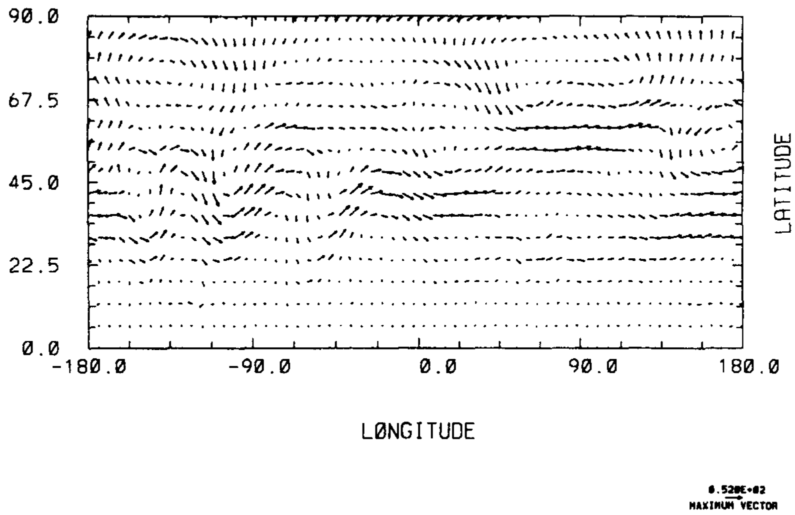

Figure 2. Vector velocity field at given geopotential height after 1 day of integration T–Z scheme for $p = 2, q = 1$ with $\alpha = 1/3(\Delta t = 300\,\mathrm{s})$ on mesh with resolution 128 × 32 points.

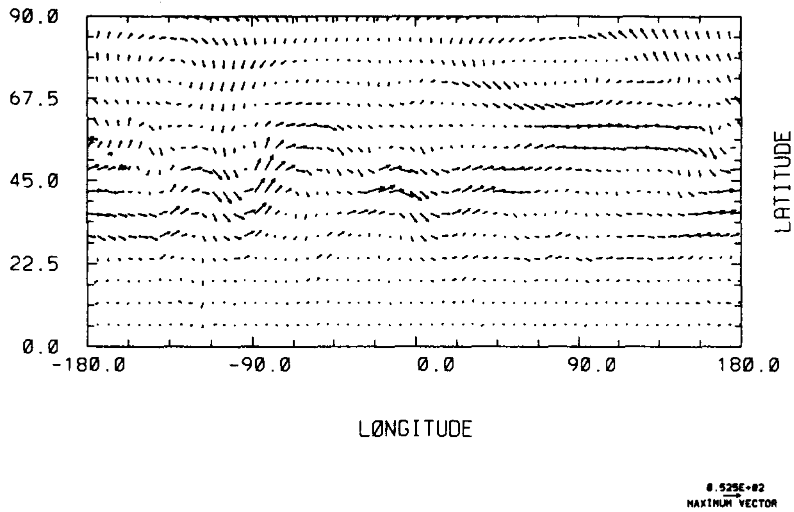

Figure 3. Same as Figure 2 but after 2 days.

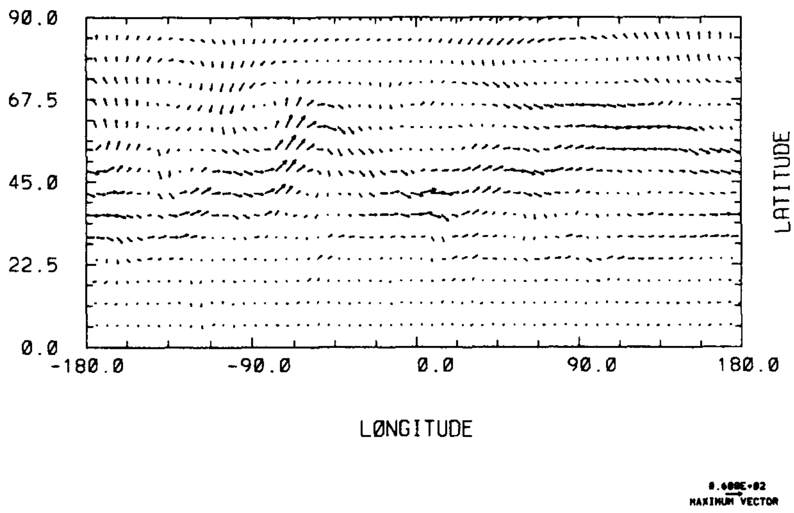

Figure 4. Same as Figure 2 but after 3 days.

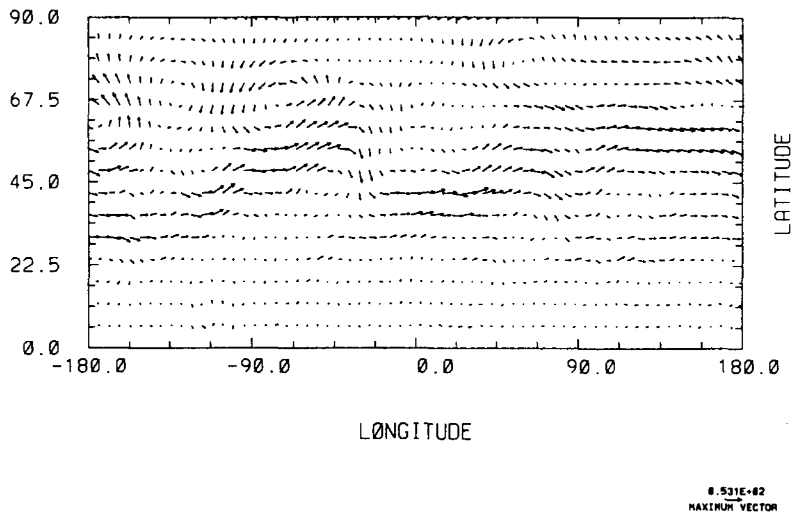

Figure 5. Same as Figure 2 but after 4 days.

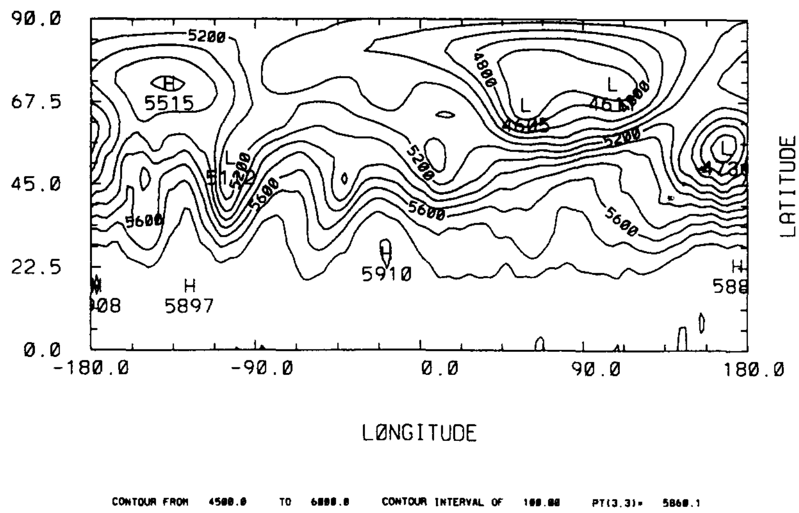

Figure 6. Geopotential height field after 1 day of integration of T–Z scheme for $p = 2, q = 1$ with $\alpha = 1/3(\Delta t = 300\,\mathrm{s})$ on mesh with resolution 128×32 points.

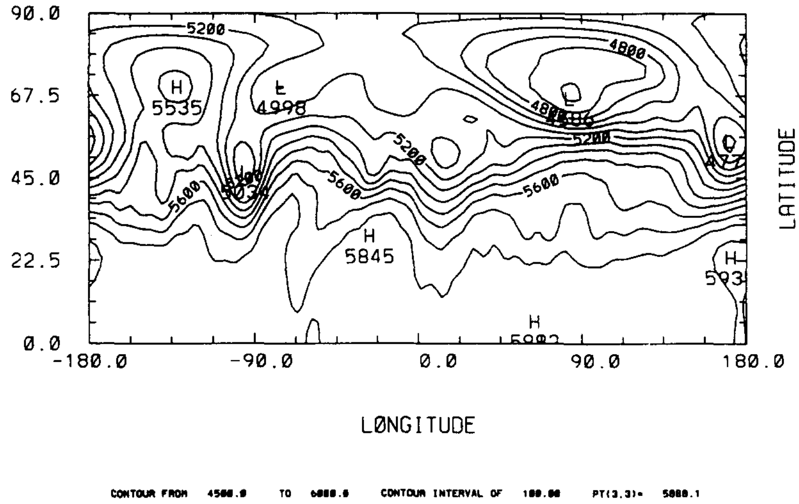

Figure 7. Same as Figure 6 but after 2 days.

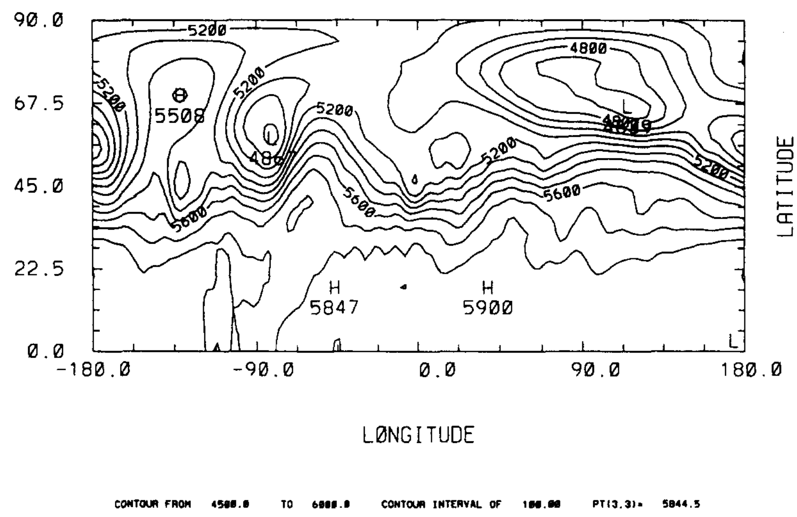

Figure 8. Same as Figure 6 but after 3 days.

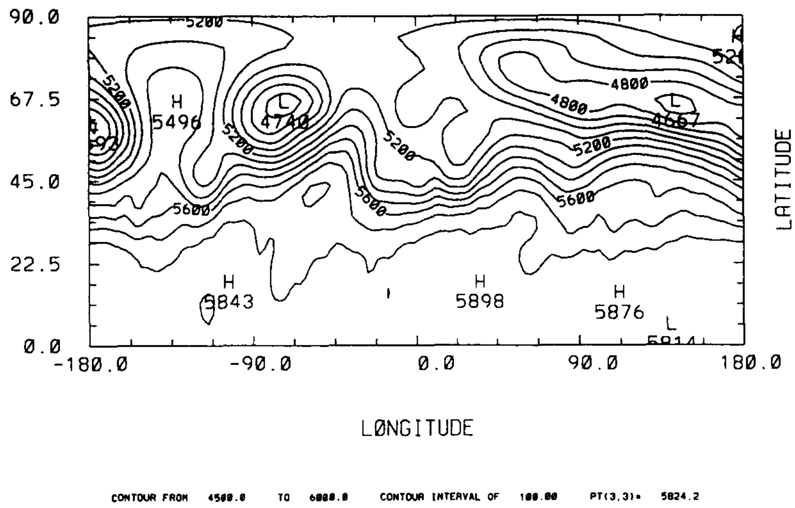

Figure 9. Same as Figure 6 but after 4 days.

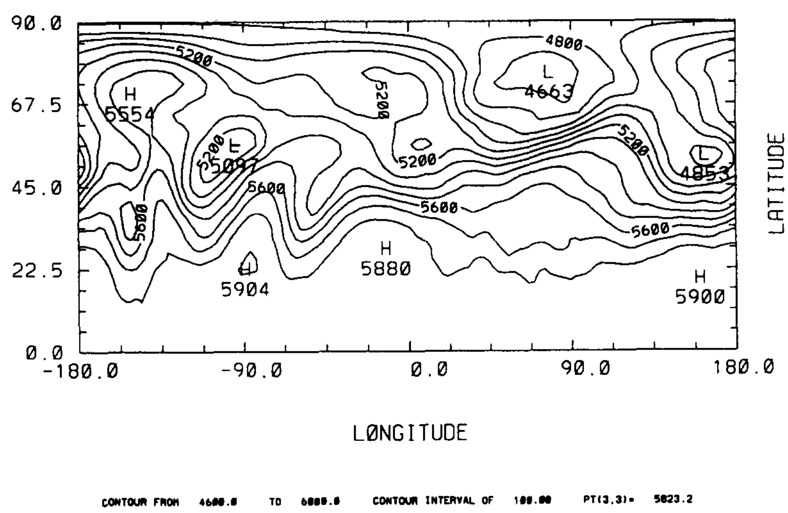

Figure 10. Initial geopotential height field contours for hemispheric T–Z scheme taken from 500 mb, FGGE data set, 1 January 1979, 00Z.

```
FILE rate saves relative integral invariants
                   Hn/Ho              Zn/Zo        En/Eo
NG=            1  0.100000E+01  0.100000E+01  0.100000E+01
NG=            2  0.100000E+01  0.100007E+01  0.100000E+01
NG=            3  0.999999E+00  0.999999E+00  0.999997E+00
NG=            4  0.999998E+00  0.998781E+00  0.999992E+00
NG=            5  0.999997E+00  0.998432E+00  0.999981E+00
NG=            6  0.999996E+00  0.997087E+00  0.999944E+00
NG=            6  0.100000E+01  0.100000E+01  0.100000E+01
NG=            7  0.999995E+00  0.996977E+00  0.999941E+00
NG=            7  0.100000E+01  0.100000E+01  0.100000E+01
NG=            8  0.999998E+00  0.998666E+00  0.999977E+00
NG=            9  0.999998E+00  0.998790E+00  0.999981E+00
NG=           10  0.999996E+00  0.997956E+00  0.999966E+00
NG=           11  0.999996E+00  0.997966E+00  0.999967E+00
NG=           12  0.999995E+00  0.997421E+00  0.999958E+00
NG=           12  0.100000E+01  0.100000E+01  0.100000E+01
NG=           13  0.999995E+00  0.997676E+00  0.999962E+00
NG=           14  0.999998E+00  0.999137E+00  0.999986E+00
NG=           15  0.999994E+00  0.997672E+00  0.999962E+00
NG=           16  0.999997E+00  0.998591E+00  0.999977E+00
NG=           17  0.999994E+00  0.997669E+00  0.999961E+00
NG=           18  0.999996E+00  0.998257E+00  0.999971E+00
NG=           19  0.999994E+00  0.997677E+00  0.999961E+00
NG=           20  0.999995E+00  0.998057E+00  0.999968E+00
NG=           21  0.999994E+00  0.997693E+00  0.999960E+00
NG=           22  0.999995E+00  0.997939E+00  0.999965E+00
NG=           23  0.999994E+00  0.997713E+00  0.999960E+00
NG=           24  0.999994E+00  0.997876E+00  0.999962E+00
NG=           25  0.999993E+00  0.997742E+00  0.999958E+00
NG=           26  0.999993E+00  0.997856E+00  0.999959E+00
NG=           27  0.999992E+00  0.997786E+00  0.999956E+00
NG=           28  0.999992E+00  0.997876E+00  0.999956E+00
NG=           29  0.999991E+00  0.997854E+00  0.999953E+00
NG=           30  0.999990E+00  0.997935E+00  0.999952E+00
NG=           31  0.999989E+00  0.997949E+00  0.999949E+00
```

Figure 11. Printout list of file 'rate'.

Subroutine SMTH (B, Q), FFT (A, M, S), RLFFT (LL, K, D, SN). These subroutines are called by the Fourier filtering subroutine SMTHFLD. When S, SN are equal to 1, FFT and RLFFT decompose the discrete real data stored in an array into its constitutes of different waves, then the amplitudes of different wave numbers are smoothed by the filter function defined in Equation (16), the smoothing part being carried out in the subroutine SMTH. Right after the smoothing, an inverse Fast Fourier Transform is performed to get the filtered data again from the Fourier domain to the physical domain using RLFFT, FFT with $S = SN = -1$.

Subroutine AV1 (A, B, C, G) and subroutine AV2 (A, B, C, D, G). These two subroutines perform the Robert filtering [Eq. (21)] after each leapfrog time-step integration. AV1 is used for the second timestep, whereas AV2 performs averaging after the third and subsequent timesteps. On entry, B, C, D contains the solution $(u, v, h)^n_{ij}, (u, v, h)^{n-1}_{ij}$, $(u, v, h)^{*n-1}_{ij}$, respectively. On exit, A contains the averaged solution $(u, v, h)^{*n}_{ij}$ of B, C, D with a weight given by G.

Subroutine LOOK (X, F, IPRNT). This subroutine calculates the integral invariants [Eqs. (24)–(27)] of the shallow-water equation model. On entry, array X contains the geopotential height and the velocity components u, v. Array F contains the Coriolis parameters. The logical parameter IPRNT tells whether to print out the integral invariants at a given time-step. If IPRNT is .TRUE., it prints out the integral invariants. Some constants in common block CA are used in this subroutine. On exit, common block CC contains the integral invariants of the shallow-water equations, that is SH, SZ, SE which are the integral invariants of the most recent time level of the numerical integration.

Subroutine SCALE (X, F). This subroutine scales the height h, the velocity components u, v in order to achieve comparable magnitudes for the total mass, enstrophy, and total energy integral invariants so as to facilitate the task of the constrained optimization computational procedure. On entry, the array X contains the unscaled fields h, u and v, whereas F contains the unscaled Coriolis parameters. On exit, X, F contain scaled values of h, u, v, and f, respectively. The scale used in the code are 10^2 for the time dimension and 10^5 for the length dimension. The constants G, DS, A in the common block CA also are scaled. This subroutine is called only once in the preprocessing stage before starting the time integration.

Subroutine FORCE (X, F). This subroutine tests whether the integral invariant constraints are satisfied within some prescribed limits. If they are not conserved, it calls the subroutine RESTO to restore the integral invariants by carrying out a small modification of the current approximation according to a least-square principle guiding the constraint restoration approach. On entry, X contains the

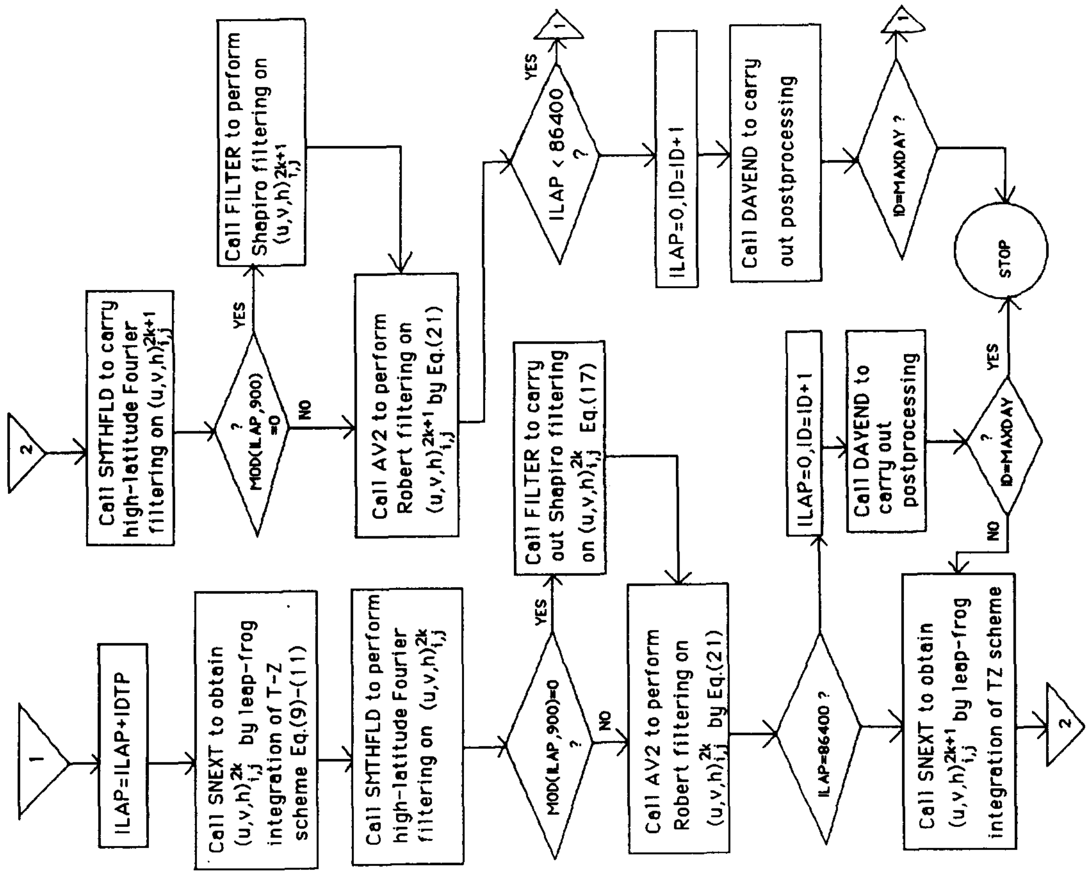
1
ILAP=ILAP+IDTP
Call SNEXT to obtain $(u,v,h)^{2k}_{i,j}$ by leap-frog integration of T-Z scheme Eq.(9)-(11)
Call SMTHFLD to perform high-latitude Fourier filtering on $(u,v,h)^{2k}_{i,j}$
MOD(ILAP,900)=0 ?
YES
NO
Call FILTER to carry out Shapiro filtering on $(u,v,h)^{2k}_{i,j}$ Eq.(17)
Call AV2 to perform Robert filtering on $(u,v,h)^{2k}_{i,j}$ by Eq.(21)
ILAP=86400 ?
ILAP=0,ID=ID+1
Call DAYEND to carry out postprocessing
ID=MAXDAY ?
YES
NO
STOP
Call SNEXT to obtain $(u,v,h)^{2k+1}_{i,j}$ by leap-frog integration of TZ scheme
2
2
Call SMTHFLD to carry high-latitude Fourier filtering on $(u,v,h)^{2k+1}_{i,j}$
MOD(ILAP,900) =0 ?
YES
NO
Call FILTER to perform Shapiro filtering on $(u,v,h)^{2k+1}_{i,j}$
Call AV2 to perform Robert filtering on $(u,v,h)^{2k+1}_{i,j}$ by Eq.(21)
ILAP < 86400 ?
YES
1
ILAP=0,ID=ID+1
Call DAYEND to carry out postprocessing
ID=MAXDAY ?
1

Figure 12B. Flowchart of main program EXSHALL (Part 2).

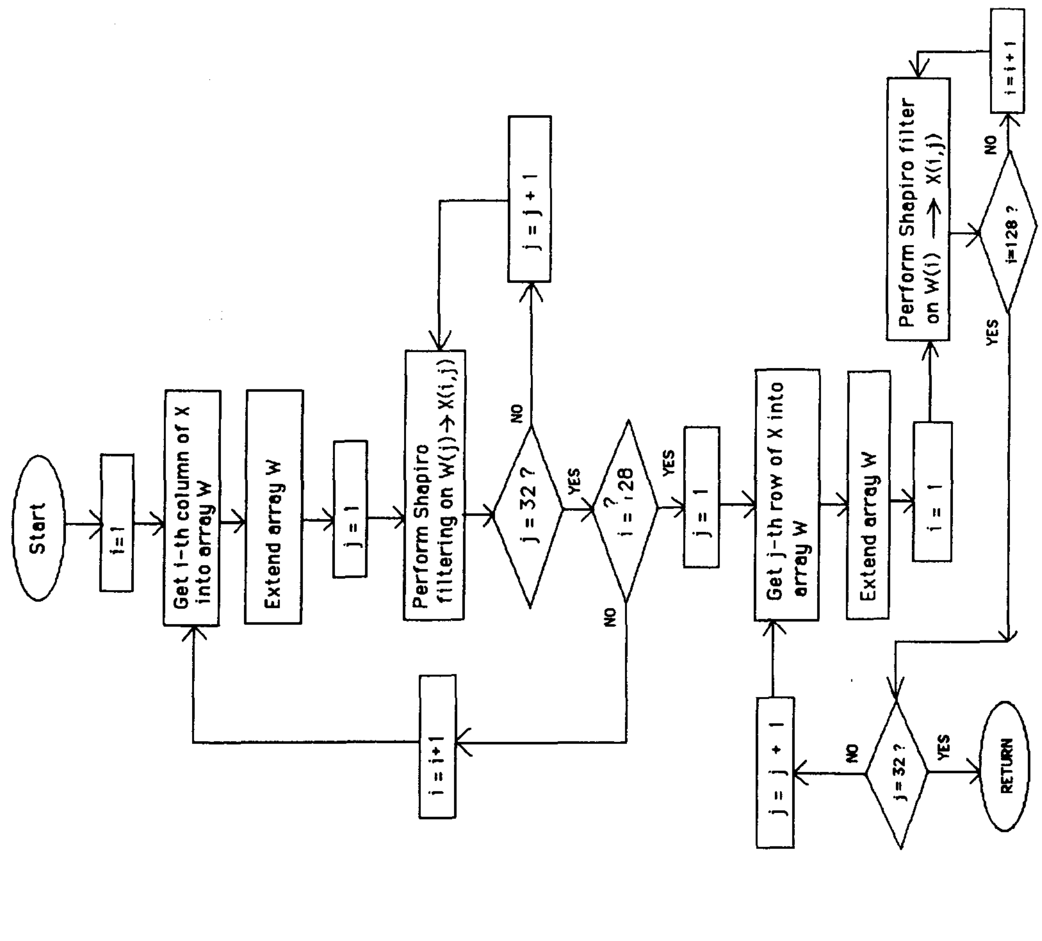
Start
i=1
Get i-th column of X into array W
Extend array W
j = 1
Perform Shapiro filtering on W(j) → X(i,j)
j = 32 ?
NO
j = j + 1
YES
i = 128 ?
NO
i = i+1
YES
j = 1
Get j-th row of X into array W
Extend array W
i = 1
Perform Shapiro filter on W(i) → X(i,j)
i=128 ?
NO
i = i + 1
YES
j = 32 ?
NO
j = j + 1
YES
RETURN

Figure 12A. Main program EXSHALL (Part 1).

```
 File EXS.OUT saves monitoring message
OMIN. H = 4662.9630
 MAX. H = 5904.1140
 MASS, ENSTROPHY, ENERGY :    0.559852E-01    0.191205E+02    0.395393E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =   300 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559852E-01    0.191219E+02    0.395394E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =   600 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559851E-01    0.191205E+02    0.395392E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =   900 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559851E-01    0.190972E+02    0.395390E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  1200 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559850E-01    0.190905E+02    0.395386E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  1500 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559850E-01    0.190648E+02    0.395371E+02
 CONSTRAINTS AFTER RESTO-ALGORITHM     0.559852E-01    0.191205E+02    0.395393E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  1800 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190627E+02    0.395370E+02
 CONSTRAINTS AFTER RESTO-ALGORITHM     0.559852E-01    0.191205E+02    0.395393E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  2100 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559850E-01    0.190950E+02    0.395384E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  2400 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559850E-01    0.190974E+02    0.395386E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  2700 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190815E+02    0.395380E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  3000 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190816E+02    0.395381E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  3300 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190712E+02    0.395377E+02
 CONSTRAINTS AFTER RESTO-ALGORITHM     0.559852E-01    0.191205E+02    0.395393E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  3600 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190761E+02    0.395378E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  3900 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559851E-01    0.191040E+02    0.395388E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  4200 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190760E+02    0.395378E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  4500 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559850E-01    0.190936E+02    0.395384E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  4800 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190760E+02    0.395378E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  5100 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190872E+02    0.395382E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  5400 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190761E+02    0.395378E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  5700 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559849E-01    0.190834E+02    0.395381E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  6000 SECONDS, 0 DAY(S).
 MASS, ENSTROPHY, ENERGY :    0.559848E-01    0.190764E+02    0.395378E+02
OALPHA = 0.33; P = 2; Q = 1; TIME =  6300 SECONDS, 0 DAY(S).
```

Figure 13. Printout list of file 'EXS.OUT'.

approximate solution at the current time-level of the integration whereas F contains the Coriolis parameters. On exit, X contains the restored h, u, and v fields with the integral invariants conservation constraint being satisfied within the prescribed limits. See Figure 17.

Subroutine RESTO (N, X, F). This subroutine carries out the integral invariant restoration algorithm [see Eqs. (41)–(46)]. On entry, array X contains the nominal solution X defined in (22) not consistent with the integral invariant constraints conservation (23) whereas the array F contains the Coriolis parameters.

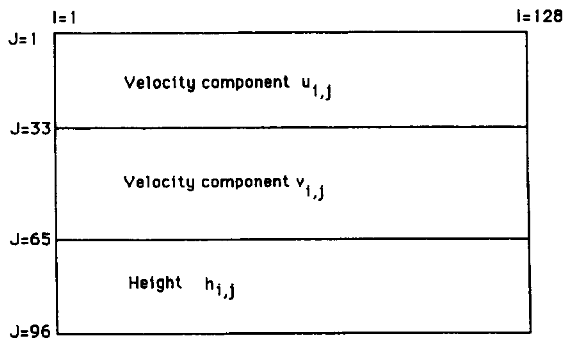

Figure 14. Layout of array EX, OX, and AX.

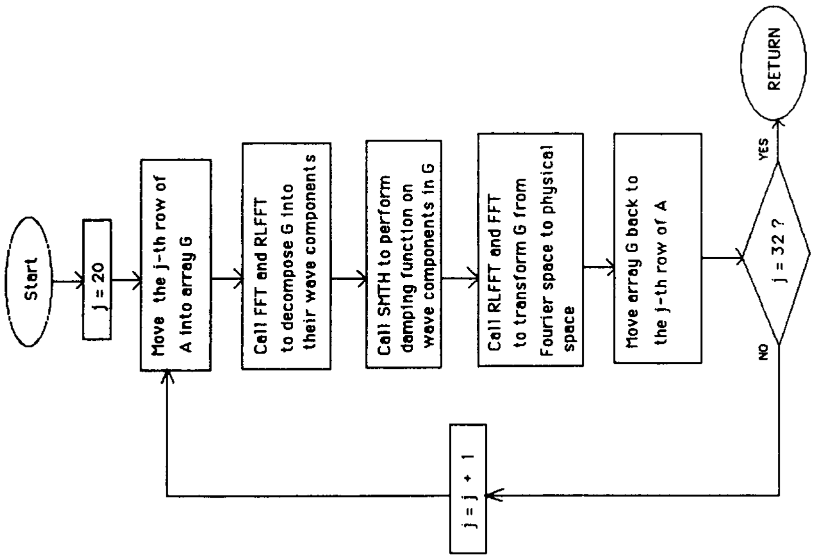
Start
j = 20
Move the j-th row of A into array G
Call FFT and RLFFT to decompose G into their wave components
Call SMTH to perform damping function on wave components in G
Call RLFFT and FFT to transform G from Fourier space to physical space
Move array G back to the j-th row of A
j = 32 ?
NO
YES
RETURN
j = j + 1

Figure 16. Subroutine SMTHFLD.

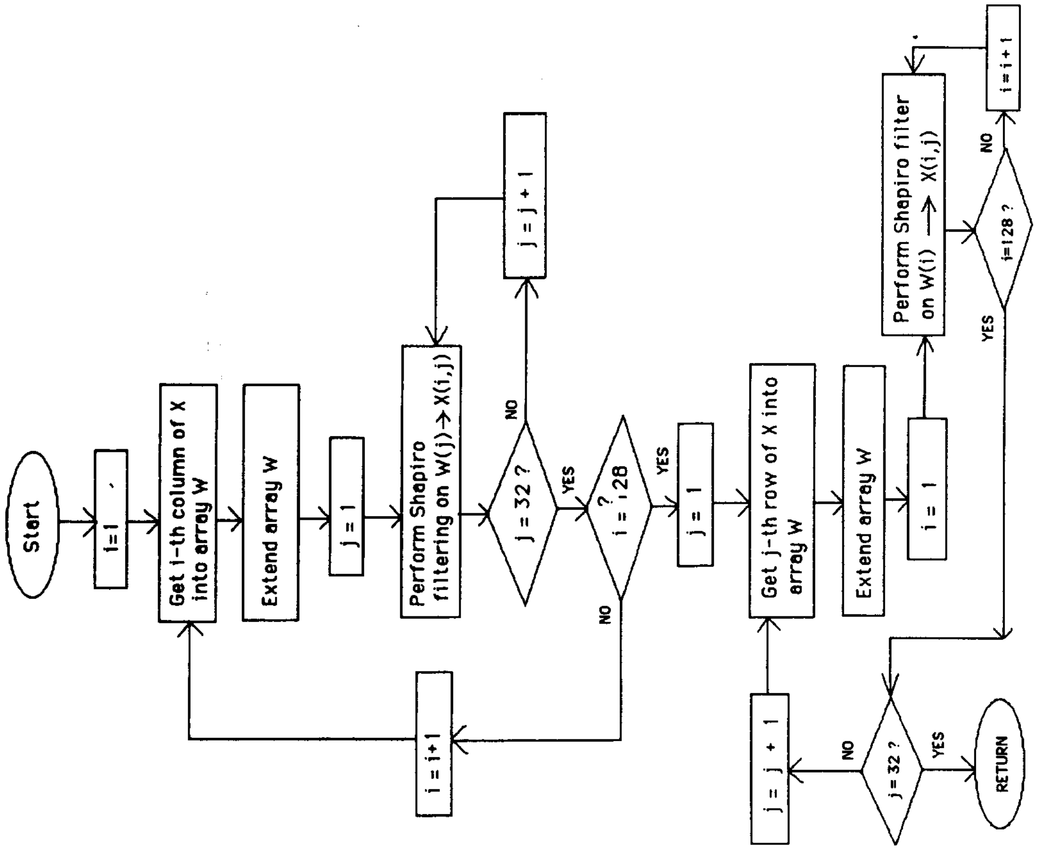
Start
i=1
Get i-th column of X into array W
Extend array W
j = 1
Perform Shapiro filtering on W(j)→X(i,j)
j = 32 ?
NO
j = j + 1
YES
i = 128 ?
NO
i = i+1
YES
j = 1
Get j-th row of X into array W
Extend array W
i = 1
Perform Shapiro filter on W(i) → X(i,j)
i=128 ?
NO
i = i+1
YES
j = 32 ?
NO
j = j + 1
YES
RETURN

Figure 15. Subroutine FILTER.

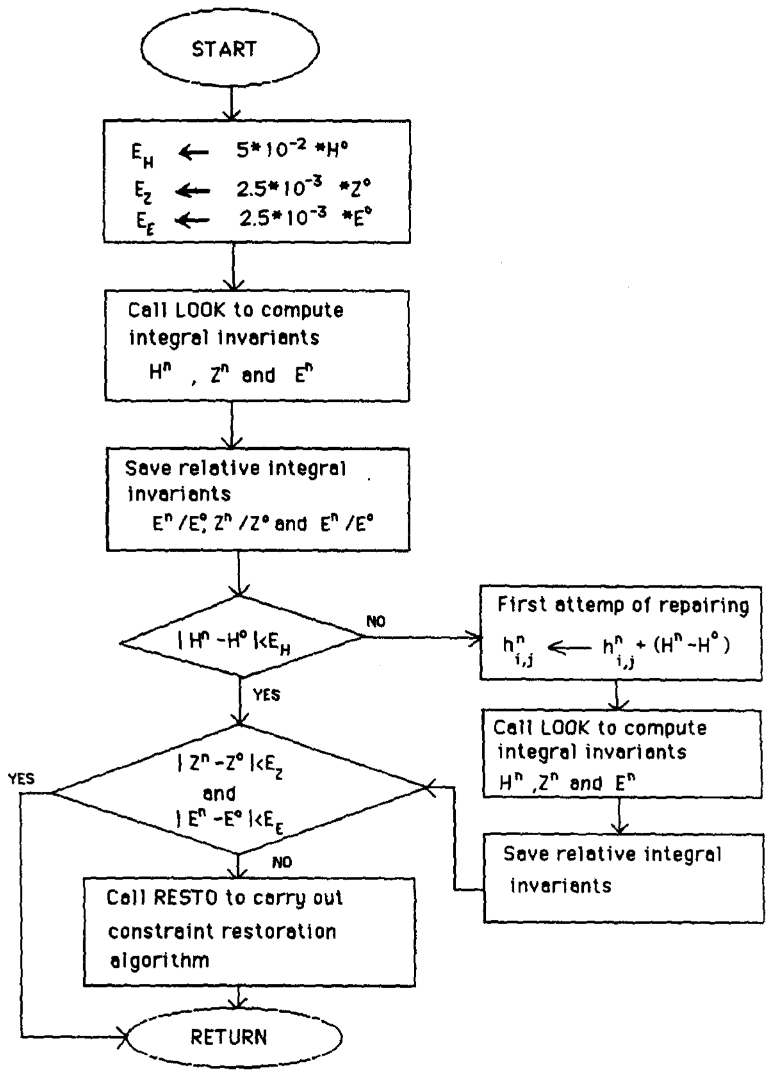
START
$E_H \leftarrow 5*10^{-2} *H^0$
$E_Z \leftarrow 2.5*10^{-3} *Z^0$
$E_E \leftarrow 2.5*10^{-3} *E^0$
Call LOOK to compute integral invariants H^n, Z^n and E^n
Save relative integral invariants E^n/E^0, Z^n/Z^0 and E^n/E^0
$|H^n - H^0| < E_H$
NO
YES
First attemp of repairing $h^n_{i,j} \leftarrow h^n_{i,j} + (H^n - H^0)$
Call LOOK to compute integral invariants H^n, Z^n and E^n
Save relative integral invariants
$|Z^n - Z^0| < E_Z$ and $|E^n - E^0| < E_E$
YES
NO
Call RESTO to carry out constraint restoration algorithm
RETURN

Figure 17. Subroutine FORCE.

The integer N is the dimension of the vector X defined in (22), which is $3 \times N_x \times N_y$. On exit, array X contains the restored point X at which the integral invariant constraints are satisfied within the prescribed limits (23). See Figure 18.

Subroutine FUNCT (X, F, PHZ, PHE, PHIHH, PHIHZ, PHIHE, PHIZZ, PHIZE, PHIEE). This subroutine calculates the gradients required by the constraint restoration algorithm. On entry, the array X contains an approximate solution whereas F contains the Coriolis parameters. On exit, array PHZ contains the second column of the matrix A(x), that is $\partial\phi_2/\partial x_i$ in Equations (34)–(36), PHE contains the third column of the matrix A(x), that is $\partial\phi_3/\partial x_i$ in Equations (37)–(39). PHIHH, PHIHZ, PHIHE, PHIZZ, PHIZE, PHIEE form the entries of the symmetric matrix $B(x)$.

Subroutine DAYEND (X, F, ID). This subroutine carries out the postprocessing stage every 24 h. It calls subroutine MAPPA to produce printer-plot contours of the height field, saves the height field and velocity field onto an auxiliary storage for later graphics or for other processing purposes. It also calls subroutine PLOT to produce a graphical display for the height field and velocity field using a graphic software package, for example the NCAR graphics package. On entry, X contains the field of h, u, and v at the ID-th day whereas F contains the Coriolis parameters.

Subroutine FILTER (X, IUVH). This subroutine performs a 17-point low-pass Shapiro filtering on the approximate solution contained in array X. Integer IUVH tells which one of the fields of h, u, v is contained in array X. On exit X contains the low-pass filtered field. See Figure 16.

Subroutine STEP1 (X1, X0, F). This subroutine carries out the first step of the time integration procedure by leapfrogging. On entry, X0 contains the initial conditions, namely, $h^0_{i,j}, u^0_{i,j}, v^0_{i,j}$ whereas F contains the Coriolis parameters. On exit, F is unchanged, while the array X1 contains now the approximation at $t = \Delta t$, namely, $u^1_{i,j}, v^1_{i,j}, h^1_{i,j}$ using

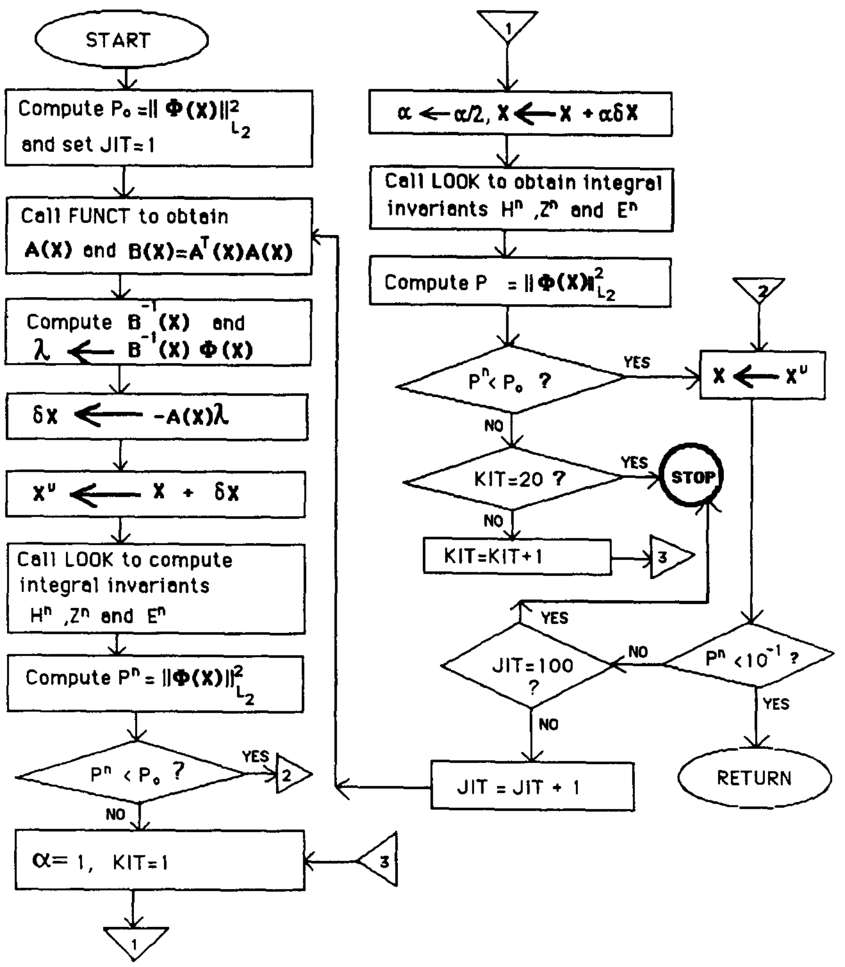
START
Compute P₀ = ‖Φ(X)‖²L2 and set JIT=1
Call FUNCT to obtain A(X) and B(X)=Aᵀ(X)A(X)
Compute B⁻¹(X) and λ ← B⁻¹(X) Φ(X)
δX ← -A(X)λ
Xᵘ ← X + δX
Call LOOK to compute integral invariants Hⁿ, Zⁿ and Eⁿ
Compute Pⁿ = ‖Φ(X)‖²L2
Pⁿ < P₀ ?
YES
NO
2
α= 1, KIT=1
3
1
α ← α/2, X ← X + αδX
Call LOOK to obtain integral invariants Hⁿ, Zⁿ and Eⁿ
Compute P = ‖Φ(X)‖²L2
Pⁿ < P₀ ?
X ← Xᵘ
KIT=20 ?
STOP
KIT=KIT+1
JIT=100 ?
Pⁿ < 10⁻¹ ?
JIT = JIT + 1
RETURN

Figure 18. Subroutine RESTO.

Equations (5)–(7) where $(u, v, h)^{-1}$ is defined now as $(u, v, h)^0$.

Subroutine SNEXT (X1, X0, X, F). This subroutine advances the time integration by leapfrogging using the T–Z algorithm for the second and subsequent time-steps. On entry, X0 and X contain approximate solutions at time levels $t = n\Delta t$ and $(n - 1)\Delta t$ respectively, whereas F contains the Coriolis parameters. On exit, X1 contains the approximate solution at time-level $t = (n + 1)\Delta t$, namely, $(u, v, h)_{i,j}^{n+1}$ integrated using Equations (5)–(7).

Acknowledgments—The first author was supported partially by the Supercomputer Computations Research Institute which is funded partially by the U.S. Department of Energy through contract No. DE.FC05-85ER250000. The second author was supported partially by the Geophysical Fluid Dynamics Institute through AFOSR Grant nos. 89-0462 and 90-0009.

REFERENCES

Arakawa, A., and Lamb, V. R., 1977, Computational design of the basic dynamical processes of the U.C.L.A. general circulation model, *in* Methods in computational physics: Academic Press, New York, v. 17, p. 174–265.

Burridge, D. M., 1975, A split semi-implicit reformulation of the Bushy–Thompson 10 level model: Quart. Jour. Roy. Meteor. Soc., v. 101, no. 430, p. 777–792.

Daley, R., 1980, The development of efficient time integration schemes using model normal modes: Mon. Wea. Rev., v. 108, no. 1, p. 100–110.

Gadd, A., 1978a, A split-explicit integration scheme for numerical weather prediction: Quart. Jour. Roy. Meteor. Soc., v. 104, no. 441, p. 569–582.

Gadd, A., 1978b, A numerical advection scheme with small phase speed errors: Quart. Jour. Roy. Meteor. Soc., v. 104, no. 441, p. 583–594.

Haltiner, G. J., and Williams, R. T., 1980, Numerical prediction and dynamic meteorology (2nd ed.): John Wiley & Sons, New York, 477 p.

Isaacson, E., 1977, Integration schemes for long-term calculations in advances in computer methods for partial differential equations II, *in* Vichnevetsky, R., ed., Publ. IMACS (AICA), v. 2, p. 251–255.

Magazenkov, L. M., Shvets, M. Ye., and Shneyerov, B. Ye., 1971, The distributed analysis approach to integration of equations of dynamics of a barotropic fluid over a long time interval: Isvestya Academy of Science, U.S.S.R. Atmos. Oceanic Phys., English Edition, v. 7, p. 560–564 (Published by the American Meteorological Union and American Meteorological Society, 2000 Florida Ave., N.W., Washington, DC 20009).

Miele, A., and Heideman, J. C., 1968, The restoration of constraints in holonomic problems: Aero Astronautics Rept. No. 39, Rice University, 14 p.

Miele, A., Heideman, J. C., and Damoulakis, J. N., 1969a, The restoration of constraints in holonomic and nonholonomic problems: Jour. Optimiz. Theory Appl., v. 3, no. 5, p. 361–381.

Miele, A., Huang, H. Y., and Heideman, J. C., 1969b, Sequential gradient restoration algorithm for the minimization of constraint functions. Ordinary and conjugate-gradient version: Jour. Optimiz. Theory Appl., v. 4, no. 4, p. 213–243.

Miele, A., Levy, V., and Cragg, E. E., 1971, Modifications and extensions of the conjugate-gradient-restoration algorithm for mathematical programming problems: Jour. Optimiz. Theory and Appl., v. 7, no. 6, p. 450–472.

Navon, I. M., 1987, The Bayliss–Isaacson Algorithm and the constraint restoration method are equivalent: Meteorol. Atmos. Phys., v. 37, no. 1, p. 143–152.

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

Navon, I. M., and de Villiers, R., 1987, The application of the Turkel–Zwas explicit large time-step scheme to a Hemispheric Barotropic Model with Constraint Restoration: Mon. Wea. Rev., v. 115, no. 5, p. 1036–1051.

Neta, B., and Navon, I. M., 1989, Analysis of the Turkel–Zwas Scheme for the shallow-water equations: Jour. Comput. Phys., v. 81, no. 2, p. 277–299.

Neta, B., Navon, I. M., and Yu, J., 1990, Analysis of the Turkel–Zwas scheme for the two-dimensional shallow water equations in spherical coordinates: Rept. No. FSU-SCRI-90-91, Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida, 17 p.

Robert, A., 1979, The semi-implicit method, *in* Numerical Methods Used in Atmospheric Models: GARP Publ. Ser. v. 17, Part II, Ch. 8, 499 p. (Available from World Meteorological Organization, Case Postale no. 5, CH-1211, Geneva 20, Switzerland).

Robert, A. J., 1966, The integration of a low order spectral form of the primitive meteorological equations: Jour. Meteor. Soc. Japan, Ser. v. 44 no. 5, p. 237–245.

Sasaki, Y., 1975, Variational design of finite-difference scheme for initial value problem of conservative system: Dept. of Meteorology, Univ. Oklahoma, 31 p.

Sasaki, Y., 1976, Variational design of finite difference schemes for initial value problems with an integral in variant: Jour. Comput. Phys., v. 21, no. 3, p. 270–278.

Sasaki, Y., 1977, Variational design of finite-difference scheme for initial value problems with a global divergent barotropic model: Contrib. Atmos. Phys., v. 50, no. 1–2, p. 284–289.

Shapiro, R., 1979, Linear filtering on the surface of a sphere: AFGL-TR-79-0263, Environ. Res. Paper No. 683, Air Force Geophysics Laboratory, Hanscom AFB, MA 01731, 23 p.

Takacs, L. L., 1986, Documentation of the Goddard Laboratory for Atmospheres Fourth-Order Two-Layer Shallow Water Model: NASA Tech. Mem. 86227, 80 p.

Takacs, L. L., and Balgovind, 1983, High-latitude filtering in Global Grid Point Models: Mon. Wea. Rev., v. 111, no. 10, p. 2005–2015.

Takacs, L. L., Kalnay, E., and Navon, I. M., 1985, High-latitude filtering in a global grid point Model using model normal modes: Proc. Seventh Conf. on Numerical Weather Prediction, Montreal, Preprint Volume, Am. Meteor. Soc. p. 277–283.

Turkel, E., and Zwas, G., 1979, Explicit large-time-step scheme for the shallow water equations: Proc. Third Int. Symp. in Advances in Computer Methods for Partial Differential Equations, *in* Vichnevtsky, R., and Stepleman, R. S., eds., Publ. IMACS, Lehigh Univ., p. 65–69, 442 p.

APPENDIX

Program EXSHALL

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      IMPLEMENTS THE TURKEL-ZWAS EXPLICIT LARGE-TIMESTEP SCHEME
C      FOR SOLVING THE SHALLOW WATER EQUATIONS ON A HEMISPHERE
C      IN SPHERICAL COORDINATES.
C
C        VERSION 1.0 EXSHALL WRITTEN IN 1986 (ON MESH 128X32)
C        VERSION 2.0 EXSHALL 1990 (FOR MESH 256X64)
C        VERSION 1.1 EXSHALL DOCUMENTED VER.1.0 ,1990,JUNE.
C
C      INITIAL VALUES FOR THE HEIGHT FIELD ARE READ FROM TAPE1
C      VALUES OF THE INTEGRAL INVARIANTS H, Z & E ARE WRITTEN TO TAPE7
C      AT EACH TIMESTEP
C      THE HEIGHT FIELD IS WRITTEN TO TAPE10 EVERY 24 HOURS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DIMENSION XE(128,96),XO(128,96),XA(128,97),IDT(6),F(32)
      COMMON/CO/CH0,CZ0,CE0
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      COMMON/CC/CH,CZ,CE
      COMMON/CG/NG,IG
      COMMON/CT/NRP,TIME
      COMMON/CW/GD(128),WS(49024)
C
C   XE,XO,XA (I,J) FOR I=1 TO 128,J=1 TO 32 STOTRES VEL. COMPONENT U
C   XE,XO,XA (I,J) FOR I=1 TO 128,J=33 TO 64 STORES VEL. VOMP. V
C   XE,XO,XA (I,J) FOR I=1 TO 128,J=65 TO 96 STORES HEIGHT H.
C
C   CH0,CZ0,CE0 ARE INITIAL TOTAL MASS,ENSTROPHY AND ENERGY RESPECTIVELY.
C
C   D--MESH SIZE;    G--GRAVITY;       A--EARTH RADIUS;   IP--LAT. MESH
C   IQ-LON. MESH;    ALF-PADE WEIGH.; DT-TIME STEP SIZE;
C
```

```
C  CH--TOTAL MASS;   CZ--TOTAL ENSTROPHY;   CE--TOTAL ENERGY;
C
C  NG,IG ARE COUNTERS;   OP----GRAPHICS OPTION
C
C  TIME--TIME ACCUMULATOR; GD,WS ARE WORKING STORAGE;
C
C  JU,JV,JH ARE INDEX POINTERS FOR H,U AND V IN ARRAY XO,XE AND XA
C
C        ISHAPI--------TIME STEP SIZE OF SHAPIRO FILTERING
C
C        INITIALIZATION
C
      DATA IDT/180,150,480,600,720,900/
      DATA GAM/0.1/,G/9.80616/,A/6370000./,NRP/0/,TIME/0./
      DATA IP/2/,IQ/1/,IG/1/,ID/0/,NG/1/,GH/1./,GZ/1./,GE/1./
      DATA NX/128/,NY/32/,OP/1/,ONEDAY/86400/,MAXDAY/2/
      DATA JU/1/,JV/33/,JH/65/,ISHAPI/900/
      IDTP=IDT(IP)
      ALF=1./3.
      PI=4.*ATAN(1.)
      D=PI/64.
      DS=D*A*2
      DT=IDTP/100.
      ILAP=IDTP
C
C        INITIALIZATION OF NCAR GRAPHICS
C
       CALL OPNGKS
C
C        FILE 'EXS.OUT' SAVES MESSAGES OF INTEGRATION FOR MONITORING
C        FILE 'hfield' INPUT FILE OF INITIAL HEIGHT FIELD


C        FILE 'rate' FOR RELATIVE INTEGRAL INVARIANTS AT EACH STEP
C        FILE 'save' SAVES HEIGHT,VELOCITY FIELD FOR LATER USE.
C
         OPEN(6,FILE='EXS.OUT')
         OPEN(1,FILE='hfield')
         OPEN(7,FILE='rate')
         OPEN(10,FILE='save')
         WRITE(6,890)
         WRITE(7,891)
         WRITE(10,892)
C
C        GET INITIAL GEOPOTENTIAL HEIGHT H
C
      READ(1,999) ((XA(I,J),I=1,NX),J=JH,JH+NY)
C
C        FIND MIN. AND MAX. OF INITIAL HEIGHT H
C
      CALL MINMAX(XA(1,JH))
C
C        FIND INITIAL CONDITIONS GEOSTROPHICALLY
C
      CALL GEOWND(XA,F)
C
C        SCALE U,V,H AND OTHER CONSTANTS CONSISTENTLY FOR
c        CONSTRAINT RESTORATION.
C
      CALL SCALE(XA,F)
C
C        PERFORM HIGH LATITUDE FOURIER FILTERING
C        XA(1,JH)---HEIGHT,  XA(1,JU)---VEL. U,  XA(1,JV)---VEL. V
C
      CALL SMTHFLD(XA(1,JU),GD)
      CALL SMTHFLD(XA(1,JV),GD)
      CALL SMTHFLD(XA(1,JH),GD)
C
C        PRODUCE PRINTER-PLOT CONTOURS OF THE HEIGHT FIELD
C
      CALL MAPPA(XA(1,JH),2000.,NX,NY)
C
C        PLOT HEIGHT CONTOURS AND VELOCITY FIELD
C
      CALL PLOT(XA(1,JH),XA(1,JU),XA(1,JV),NX,NY)
C
C        COMPUTE INITIAL INTEGRAL INVARIANTS
C
      CALL LOOK(XA,F,.TRUE.)
      CHO=CH
      CZO=CZ
      CEO=CE
      WRITE(7,991) NG,GH,GZ,GE
      WRITE(6,992) ALF,IP,IQ,ILAP,ID
```

```
C
C      FIRST TIMESTEP (ODD)
C
       CALL STEP1(XO,XA,F)
C
C        PERFORM HIGH LATITUDE FOURIER FILTERING
C        XO(1,JU)----VEL. U,  XO(1,JV)----VEL. V, XO(1,JH)----HEIGHT
C
       CALL SMTHFLD(XO(1,JU),GD)
       CALL SMTHFLD(XO(1,JV),GD)
       CALL SMTHFLD(XO(1,JH),GD)
       ILAP=ILAP+IDTP
C
C      SECOND TIMESTEP (EVEN)
C


      WRITE(6,992) ALF,IP,IQ,ILAP,ID
      CALL SNEXT(XE,XA,XO,F)
C
C       PERFORM HIGH LATITUDE FOURIER FILTERING
C       XO(1,JU)----VEL. U,  XO(1,JV)----VEL. V, XO(1,JH)----HEIGHT
C
      CALL SMTHFLD(XE(1,JU),GD)
      CALL SMTHFLD(XE(1,JV),GD)
      CALL SMTHFLD(XE(1,JH),GD)
C
C       PERFORM ROBERT FILTERING
C
      CALL AV1(XA,XO,XE,GAM)
C
C     SUBSEQUENT STEPS
C     SUBROUTINE SNEXT ADVANCES THE INTEGRATION
C     SUBROUTINE SMTHFLD PERFORMS HIGH-LATITUDE FOURIER FILTERING
C     AT EACH TIMESTEP
C     SUBROUTINE FILTER PERFORMS SHAPIRO FILTERING EVERY 900 SECONDS
C
    2 ILAP=ILAP+IDTP
C
C     ODD TIMESTEP
C
      WRITE(6,992) ALF,IP,IQ,ILAP,ID
C
C       TURKEL-ZWAS SCHEME WITH LEAP-FROG TIME INTEGRATION
C
      CALL SNEXT(XO,XA,XE,F)
C
C       PERFORM HIGH LATITUDE FOURIER FILTERING
C       XO(1,JU)----VEL. U,  XO(1,JV)----VEL. V,    XO(1,JH)----HEIGHT
C
      CALL SMTHFLD(XO(1,JU),GD)
      CALL SMTHFLD(XO(1,JV),GD)
      CALL SMTHFLD(XO(1,JH),GD)
      IF(MOD(ILAP,ISHAPI).NE.0) GO TO 7
C
C       PERFORM SHAPIRO 17-POINT FILTERING
C       XO(1,JU)----VEL. U,  XO(1,JV)----VEL. V,    XO(1,JH)----HEIGHT
C
      CALL FILTER(XO(1,JU),1)
      CALL FILTER(XO(1,JV),2)
      CALL FILTER(XO(1,JH),3)
C
C       PERFORM ROBERT FILTERING
C
    7 CALL AV2(XA,XO,XE,XA,GAM)
      IF(ILAP.LT.ONEDAY) GO TO 3
      ID=ID+1
C
C       POSTPROCESSING
C
      CALL DAYEND(XO,F,ID,OP)
      IF(ID.EQ.MAXDAY) GO TO 4
      ILAP=0
C
C     EVEN TIMESTEP
C
    3 ILAP=ILAP+IDTP
      WRITE(6,992) ALF,IP,IQ,ILAP,ID
C
C       TURKEL-ZWAS EXPLICIT LARGE-TIME STEP
C          SCHEME WITH LEAP-FROG TIME INTEGRATION
C
      CALL SNEXT(XE,XA,XO,F)
```

```
C
C       PERFORM HIGH LATITUDE FOURIER FILTERING
C       XE(1,JU)----VEL. U,   XE(1,JV)----VEL. V,     XE(1,JH)----HEIGHT
C
      CALL SMTHFLD(XE(1,JU),GD)
      CALL SMTHFLD(XE(1,JV),GD)
      CALL SMTHFLD(XE(1,JH),GD)
      IF(MOD(ILAP,ISHAPI).NE.0) GO TO 8
C
C       PERFORM SHAPIRO FILTERING
C       XE(1,JU)----VEL. U,   XE(1,JV)----VEL. V,     XE(1,JH)----HEIGHT
C
      CALL FILTER(XE(1,JU),1)
      CALL FILTER(XE(1,JV),2)
      CALL FILTER(XE(1,JH),3)
    8 CALL AV2(XA,XE,XO,XA,GAM)
50     IF(ILAP.LT.ONEDAY) GO TO 2
      ID=ID+1
C
C       POSTPROCESSING
C
      CALL DAYEND(XE,F,ID,OP)
      IF(ID.EQ.MAXDAY) GO TO 4
      ILAP=0
      GO TO 2
C
C       CLOSE NCAR GRAPGICS
C
       CALL CLSGKS
C
    4 WRITE(6,995) IG
      WRITE(6,996) NRP,TIME
C
C
 890  FORMAT(1X,'File EXS.OUT saves monitoring message')
 891  FORMAT(1X,'FILE rate saves relative integral invariants',
     +        /,13X,'Hn/Ho',14X,'Zn/Zo',8X,'En/Eo')
 892  FORMAT(1X,'FILE <SAVE> SAVES H,U AND V FOR EVERY 24HRS',
     +      2X,'NX=',I4,5X,'NY=',I4,
     +      /,2X,'FORMAT(8E16.10)((X(I,J),I=1,128),J=1,96)',
     +      /,2X,'LAYOUT OF X,H(i,j)=X(i,j+64),U(i,j)=X(i,j),
     +      V(i,j)=X(i,j+32)')
 991  FORMAT(1X,'NG=',I9,2X,3(2X,E12.6))
 992  FORMAT('0ALPHA =',F5.2,'; P =',I2,'; Q =',I2,'; TIME =',I6,
     +' SECONDS,',I2,' DAY(S).')
 995  FORMAT('0NUMBER OF DATA POINTS FOR GRAPH =',I5)
 996  FORMAT('0NO. OF REPAIR POINTS =',I3,' ,CPU TIME TAKEN =',F5.2)
 999  FORMAT(8F16.11)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     FOURIER FILTERING CALLING ROUTINE
C     ON EXIT : A IS SMOOTHED BY HIGH LATITUDE FOURIER FILTERING
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SMTHFLD(A,G)
      DIMENSION A(128,32),G(128)
      DO 30 J=20,32
C
C     HIGH-LATITUDE FOURIER FILTERING STARTS
C          AT APPROX. 55 DEGREES LATITUDE
C
         D=ATAN(1.)/16.
C
C       GET ONE ROW INTO G FOR FOURIER FILTERING
C
         DO 10 I=1,128
   10    G(I)=A(I,J)
         Q=COS((J-.5)*D)
C
C       DECOMPOSE THE DATA IN G INTO ITS WAVE COMPONENTS
C
         CALL FFT(G,6,1.)
         CALL RLFFT(128,6,G,1.)
C
C       PERFORMING DAMPING FUNCTION ON AMPLITUTDES OF DIFF. WAVE NOS.
C
         CALL SMTH(G,Q)
C
C       TRANSFORMING BACK FROM FOURIER SPACE TO PHYSICAL SPACE
C
         CALL RLFFT(128,6,G,-1.)
         CALL FFT(G,6,-1.)
```

```
C
C       REPLACE THE UNSMOOTHED SOLUTION
C
        DO 20 I=1,128
   20   A(I,J)=G(I)/64.
   30 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     MINIMUM AND MAXIMUM OF THE HEIGHT FIELD
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE MINMAX(H)
      DIMENSION H(4224)
      HMIN=H(1)
      HMAX=H(1)
      DO 10 L=2,4224
         IF(H(L).GE.HMIN) GO TO 5
         HMIN=H(L)
         GO TO 10
    5    IF(H(L).LE.HMAX) GO TO 10
         HMAX=H(L)
   10 CONTINUE
      WRITE(6,15) HMIN,HMAX
   15 FORMAT('0MIN. H =',F10.4/' MAX. H =',F10.4)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS ROUTINE TAKES THE GIVEN H VALUES
C     (ON A HEMISPHERICAL GRID WHICH INCLUDES THE EQUATOR AND POLE),
C     CALCULATES U AND V VELOCITY COMPONENTS GEOSTROPHICALLY AT
C     INTERMEDIATE POINTS, THEN USES A 10-POINT INTERPOLATION TO
C     OBTAIN H AT THE SAME POINTS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GEOWND(X,FS)
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      COMMON/CW/W(137)
      DIMENSION X(128,97),FS(32)
      DO 20 J=1,32
         JH=J+64
         JP=JH+1
         FS(J)=1.45842E-4*SIN((J-.5)*D)
         F=AMAX1(SIN((J-.5)*D),.5)*1.45842E-4
         Q=COS((J-.5)*D)
         DO 20 I=1,128

            IP=1
            IF(I.LT.128) IP=I+1
C
C       COMPUTE V-COMPONENT OF VELOCITY GEOSTROPHICALLY
C
            X(I,J+32)=G*(X(IP,JP)-X(I,JP)+X(IP,JH)-X(I,JH))/DS/F/Q
C
C       COMPUTE U-COMPONENT OF VELOCITY GEOSTROPHICALLY
C
            X(I,J)=-G*(X(IP,JP)+X(I,JP)-X(IP,JH)-X(I,JH))/DS/F
   20 CONTINUE
C
C INTERPOLATE H IN THE LATITUDINAL DIRECTION
C
        DO 100 I=1,128
        I64=I+64
        IF(I64.GT.128) I64=I64-128
C
C TAKE ONE ROW ALONG THE LATITUDINAL DIRECTION
C
        DO 30 J=65,97
   30   W(J)=X(I,J)
        DO 31 J=65,68
   31   W(129-J)=X(I,J)
        DO 32 J=94,97
   32   W(195-J)=X(I64,J)
        DO 40 J=65,96
C
C INTERPOLATES H USING 10-POINT
C
   40   X(I,J)=.00053406*W(J-4)-.00617981*W(J-3)+.03460693*W(J-2)
     +  -.13458252*W(J-1)+.60562134*W(J )+.60562134*W(J+1) -.13458252*
     +  W(J+2)+.03460693*W(J+3)-.00617981*W(J+4) +.00053406*W(J+5)
  100 CONTINUE
C
```

```
C INTERPOLATES H IN THE LONGITUDINAL DIRECTION
C
      DO 200 J=65,96
         DO 130 I=1,128
C
C TAKE ONE ROW ALONG THE LONGITUDINAL DIRECTION
C
  130    W(I+4)=X(I,J)
         DO 131 I=125,128
  131    W(I-124)=X(I,J)
         DO 132 I=1,5
  132    W(132+I)=X(I,J)
         DO 140 I=1,128
C
C INTERPOLATES H USING 10-POINT
C
  140    X(I,J)=.00053406*W(I )-.00617981*W(I+1)+.03460693*W(I+2)
     +   -.13458252*W(I+3)+.60562134*W(I+4)+.60562134*W(I+5) -.13458252*
     +   W(I+6)+.03460693*W(I+7)-.00617981*W(I+8) +.00053406*W(I+9)
  200 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C    PRODUCES HEIGHT FIELD CONTOURS AND VELOCITY VECTOR FIELD USING
C    GRAPHICS SOFTWARE PACKAGE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUbROUTINE PLOT(FLD,NX,NY)
        DIMENSION H(NX,NY),U(NX,NY),V(NX,NY)


C       PLOT HEIGHT FIELD
C
        CALL PWRITY(0.5,0.9,'HEIGHT FIELD',12,2,0,0)
        CALL PWRITY(0.5,0.1,'LONGITUDE',9,2,0,0)
        CALL PWRITY(0.05,0.5,'LATITUDE',8,2,90,0)
        CALL SET(0.2,0.8,0.2,0.8,-180.,180.0,0.,90.,1)
        CALL LABMOD('(f6.1)','(f5.1)',6,6,2,2,0,0,0)
        CALL PERIML(2,2,2,2)
        CALL CONREC(H,NX,NX,NY,4500.0,6000.0,100.0,-1,0,0)
        CALL FRAME
C
C       PLOT VECTOR VELOCITY FIELD
C
        CALL PWRITY(0.5,0.9,'VECTOR VEL. FILED',17,2,0,0)
        call pwrity(0.5,0.1,'LONGITUDE',9,2,0,0)
        CALL PWRITY(0.95,0.4,'LATITUDE',8,2,90,0)
        CALL WTSTR(0.5,0.75,TITLE,15,0,0)
        call set(0.1,0.9,0.2,0.6,-180.,180.0,0.,90.,1)
        call labmod('(f6.1)','(f5.1)',6,6,2,2,0,0,0)
        call periml(4,4,4,4)
        call velvct(U,NX,V,NX,NX,NY,0.,0.,-1,0,0,0.)
        call frame
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PRODUCES HEIGHT FIELD CONTOURS ON THE PRINTER
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE MAPPA(H,C,NX,NY)
      COMMON/CW/Z(125),IZ(125)
      DIMENSION H(NX,NY),NUM(10)
C
C       SET UP CHARACTERS USED IN THE PRNTER-PLOT
C
      DATA NUM/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H0/,BL/1H /
      DATA N/4/,FN/4/,I/0/
      NYM=NY-1
C
C       PRODUCE FIRST LINE ON THE PRINTER-PLOT
C
      WRITE(6,1) (J,J=1,NY)
    1 FORMAT(' ',3X,32I4)
C
C       LOOP OVER EACH ROW OF DATA IN ARRAY H
C
   10 I=I+1
      IP1=I+1
      IF(IP1.GT.NX) IP1=IP1-NX
C
C       GET ONE COLUMN OF DATA INTO Z
```

```
      DO 15 J=1,NY
         JX=1+N*(J-1)
   15 Z(JX)=H(I,J)
C
C        GET MORE POINTS IN EACH COLUMN BY LINEAR INTERPOLATION
C
   18 DO 20 J=1,NYM
         JX=1+N*(J-1)
         YDIF=(Z(JX+N)-Z(JX))/FN
         M1=JX+1
         M2=JX+N-1
         DO 20 M=M1,M2
   20 Z(M)=Z(M-1)+YDIF
C
      MEND=1+N*NYM
C
C        TRANSFORM NUMERICAL DATA INTO CHARACTERS FOR PRINTER-PLOT
C
      DO 40 M=1,MEND
         IF(Z(M).GE.0.) GO TO 30
         AANS=-Z(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
         IZ(M)=NUM(KANS)
         GO TO 40
   30    KANS=C*Z(M)
         KKANS=2*(KANS/2)
         IF(KANS.EQ.KKANS) GO TO 25
   35    IZ(M)=BL
   40 CONTINUE
   50 CONTINUE
C
C        PRINT OUT ONE COLUMN OF TRANSFORMED DATA ON PRINTER
C
      WRITE(6,2) I,(IZ(M),M=1,MEND)
    2 FORMAT(' ',I3,1X,125A1)
      IF(I-NX) 10,55,65
   55 I=I+1
      DO 60 J=1,NY
         JX=1+N*(J-1)
   60 Z(JX)=H(1,J)
      GO TO 18
   65 RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CALLED BY THE FOURIER FILTERING ROUTINE
C     THIS SUBROUTINE PERFORMS THE DAMPING OF THE AMPLITUDES OF
C       DIFFERENT WAVE NUMBERS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SMTH(B,Q)
      DIMENSION B(128)
      D = ATAN(1.0)/32.
      DO 10 L=3,128
         M=(L-1)/2
         S=Q/SIN(M*D)
         IF(S.GT.1.) S=1.
         B(L)=S*B(L)
   10 CONTINUE
      S=Q/SIN(64*D)
      IF(S.GT.1.) S=1.
      B(2)=S*B(2)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CALLED BY THE FOURIER FILTERING ROUTINE
C
C       THIS SUBROUTINE TOGETHER WITH THE SUBROUTINE RLFFT TRANSFORMS
C       THE DISCRETE DATA IN ARRAY A INTO ITS WAVE COMPONENTS IF S=1.
C       IF S=-1,IT TRANSFORMSTHE DISCRETE DATA FROM FOURIER SPACE
C       BACK TO THE PHYSICAL SPACE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FFT(A,M,S)

      COMPLEX A(1),U,W,T
      N=2**M
      NV2=N/2
      NM1=N-1
      J=1
      PI=4.*ATAN(1.)
      DO 7 I=1,NM1
         IF(I.GE.J) GO TO 5
         T=A(J)
         A(J)=A(I)
         A(I)=T
    5    K=NV2
    6    IF(K.GE.J) GO TO 7
         J=J-K
         K=K/2
         GO TO 6
    7 J=J+K
      DO 20 L=1,M
         LE=2**L
         LE1=LE/2
         U=(1.,0.)
         ANG=PI/LE1
         W=CMPLX(COS(ANG),S*SIN(ANG))
         DO 20 J=1,LE1
            DO 10 I=J,N,LE
               IP=I+LE1
               T=A(IP)*U
               A(IP)=A(I)-T
   10       A(I)=A(I)+T
   20 U=U*W
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CALLED BY THE FOURIER FILTERING ROUTINE
C
C       THIS SUBROUTINE TOGETHER WITH THE SUBROUTINE FFT TRANSFORMS
C       THE DISCRETE DATA IN ARRAY A INTO ITS WAVE COMPONENTS IF S=1.
C       IF S=-1,IT TRANSFORMSTHE DISCRETE DATA FROM FOURIER SPACE
C       BACK TO THE PHYSICAL SPACE.
C
C      LL IS DIMENSION OF REAL ARRAY D
C      L GIVEN BY LL=2**L
C      K=L-1
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RLFFT(LL,K,D,SN)
      DIMENSION D(LL)
      PI=4.*ATAN(1.)
      M=2**K
      N=M+M
      MM1=M-1
      DO 10 J1=3,MM1,2
         J2=J1+1
         NJ1=N-J1+2
         NJ2=NJ1+1
         T=.5*(D(J1)+D(NJ1))
         D(NJ1)=.5*(D(NJ1)-D(J1))
         D(J1)=T
         T=.5*(D(J2)-D(NJ2))
         D(NJ2)=.5*(D(J2)+D(NJ2))
         D(J2)=T
   10 CONTINUE
      T=D(1)
      D(1)=T+D(2)
      D(2)=T-D(2)

      ST=SN*PI/FLOAT(M)
      TH=0.
      DO 20 J1=3,MM1,2
         TH=TH+ST
         C=COS(TH)
         S=SIN(TH)
         J2=J1+1
         NJ1=N-J1+2
         NJ2=NJ1+1
         T1=D(NJ2)*C-D(NJ1)*S
         T2=D(NJ1)*C+D(NJ2)*S
         D(NJ1)=D(J1)-T1
         D(NJ2)=-D(J2)+T2
         D(J1)=D(J1)+T1
         D(J2)=D(J2)+T2
   20 CONTINUE
      D(M+2)=SN*D(M+2)
      IF(SN.GT.0.) RETURN
      MD2=M/2
      DO 30 I=2,MD2
         L=2*I-1
```

```
            LM=(M+2-I)*2-1
            C=D(L)
            S=-D(L+1)
            D(L)=D(LM)
            D(L+1)=-D(LM+1)
            D(LM)=C
            D(LM+1)=S
   30    CONTINUE
         DO 40 I=1,2
            D(I)=D(I)/2.
   40    CONTINUE
         MD2=(MD2+1)*2
         D(MD2)=-D(MD2)
         RETURN
         END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROBERT FILTER OF THE LEAP-FROG EXPLICIT TIME-INTEGRATION SCHEME
C     PERFORMS AVERAGING AFTER THE SECOND TIMESTEP
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE AV1(A,B,C,G)
      DIMENSION A(128,96),B(128,96),C(128,96)
      H=1.-G
      DO 1 J=1,96
         DO 1 I=1,128
            A(I,J)=H*B(I,J)+G*C(I,J)
    1 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROBERT FILTER OF THE LEAP-FROG EXPLICIT TIME-INTEGRATION SCHEME
C     PERFORMS AVERAGING AFTER THIRD AND SUBSEQUENT TIMESTEPS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE AV2(A,B,C,D,G)
      DIMENSION A(128,96),B(128,96),C(128,96),D(128,96)
      H=1.-G-G
      DO 1 J=1,96
         DO 1 I=1,128
            A(I,J)=G*B(I,J)+H*C(I,J)+G*D(I,J)
    1 CONTINUE
      RETURN
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CALCULATES INTEGRAL INVARIANTS OF THE SHALLOW-WATER EQUATION
C     MODEL , NAMELY
C     H - TOTAL MASS
C     Z - TOTAL POTENTIAL ENSTROPHY
C     E - TOTAL ENERGY
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LOOK(X,F,IPRNT)
      DIMENSION X(128,96),F(32)
      LOGICAL IPRNT
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      COMMON/CC/SH,SZ,SE
      C2=DS
      CH=SIN(.5*D)/64.
      CZ=(A*D)**2/2.
      SH=0.
      SZ=0.
      SE=0.
      DO 10 I=1,128
         IP1=I+1
         IF(IP1.GT.128) IP1=IP1-128
         IM1=I-1
         IF(IM1.LT.1) IM1=IM1+128
         I64=I+64
         IF(I64.GT.128) I64=I64-128
         DO 10 J=1,32
            PMJP1=1.
            IJP1=I
            JP1=J+1
            IF(JP1.LE.32) GO TO 1
            PMJP1=-1.
            IJP1=I64
            JP1=32
    1       JM1=J-1
            IF(JM1.GE.1) GO TO 2
            JM1=1
    2       CJ=COS((J-.5)*D)
            C1=C2*CJ
C
C       ACCUMULATES THE MASS
C
            SH=SH+X(I,J+64)*CJ
C
C       ACCUMULATES THE ENSTROPHY
C
            SZ=SZ+((X(IP1,J+32)-X(IM1,J+32))/C1- (PMJP1*X(IJP1,JP1)-X(I,
     +      JM1))/C2+F(J))**2*CJ/X(I,J+64)
C
C       ACCUMULATES THE ENERGY
C
            SE=SE+(X(I,J)**2+X(I,J+32)**2+G*X(I,J+64))*X(I,J+64)*CJ
   10 CONTINUE
      SH=SH*CH
      SZ=SZ*CZ
      SE=SE*CZ
      IF(IPRNT) WRITE(6,11) SH,SZ,SE
   11 FORMAT(' MASS, ENSTROPHY, ENERGY :',3E16.6)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE ATTEMPTS TO REDUCE INTEGRAL INVARIANTS H, Z AND E
C     TO COMPARABLE MAGNITUDES USING CONSISTENT SCALING ,FOR THE SAKE OF
C     EFFICIENT MINIMIZATION.
C
C       THE SCALE USED IS :
C         LENGTH SCALE IS 1.E5 METERS
C         TIME SCALE IS 1.E2 SECONDS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SCALE(X,F)
      DIMENSION X(128,96),F(32)
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      DO 10 J=1,64
         DO 10 I=1,128
   10 X(I,J)=X(I,J)/1.E3
      DO 11 J=65,96
         DO 11 I=1,128
   11 X(I,J)=X(I,J)/1.E5
      DO 12 J=1,32
   12 F(J)=F(J)*1.E2
      G=G/10.
      DS=DS/1.E5
      A=A/1.E5
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE FORCE
C     TESTS WHETHER THE CONSTRAINTS ARE SATISFIED WITHIN SPECIFIED LIMITS
C     IF NOT, CALLS THE RESTORATION ROUTINE
C
C       ON ENTRY: X CONTAINS H,U AND V, F CONTAINS THE CORIOLIS PARAM.
C       ON EXIT: X SATISFIES THE INTEGRAL INVARIANTS CONSERVATION
C                CONDITION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FORCE(X,F)
      DIMENSION X(128,96),F(32)
      COMMON/C0/CH0,CZ0,CE0
      COMMON/CC/CH,CZ,CE
      COMMON/CG/NG,IG
      COMMON/CT/NRP,TIME
      EPSH=CH0*5.E-2
      EPSZ=CZ0*2.5E-3
      EPSE=CE0*2.5E-3
C
C       COMPUTES THE INTEGRAL INVARIANTS
C
      CALL LOOK(X,F,.TRUE.)
C
      IG=IG+1
      NG=NG+1
      GH=CH/CH0
      GZ=CZ/CZ0
      GE=CE/CE0
C
C       SAVE RELATIVE INTEGRAL INVARIANTS
C
      WRITE(7,220) NG,GH,GZ,GE
```

```
220   FORMAT(1X,'NG=',I9,3(2X,E12.6))
C
C       CHECKING CONSERVATION OF THE TOTAL MASS
C
      IF(ABS(CH-CH0).LT.EPSH) GO TO 10
C
C      IF NOT SATISFIED,FIRST ATTEMPT OF REPARING THE TOTAL MASS
C             INTEGRAL INVARIANTS.
C
      DO 20 J=65,96
         DO 20 I=1,128
   20 X(I,J)=X(I,J)+CH0-CH
C
C       COMPUTES THE INTEGRAL INVARIANTS AGAIN AFTER FIRST ATTEMPT
C
      CALL LOOK(X,F,.TRUE.)
      IG=IG+1
      GH=CH/CH0
      GZ=CZ/CZ0
      GE=CE/CE0
      WRITE(7,220) NG,GH,GZ,GE
C
C       CHECKING CONSERVATION OF THE TOTAL ENSTROPHY AND TOTAL ENERGY
C
   10 IF(ABS(CZ-CZ0).LT.EPSZ.AND.ABS(CE-CE0).LT.EPSE) RETURN
C
C       IF NOT SATISFIED, PERFORM THE CONSTRAINT RESTORATION ALGORITHM
C
      TICK = SECOND()
      CALL RESTO(12288,X,F)
      TOCK = SECOND()
      TIME=TIME+TOCK-TICK
C
      NRP=NRP+1
   80 IG=IG+1
      GH=CH/CH0
      GZ=CZ/CZ0
      GE=CE/CE0
      WRITE(7,220) NG,GH,GZ,GE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     APPLIES THE RESTORATION ALGORITHM
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RESTO(N,X,F)
      DIMENSION X(N),F(32)
      COMMON/CA/D
      COMMON/C0/CSH0,CSZ0,CSE0
      COMMON/CC/CH,CZ,CE
      COMMON/CW/PHZ(12288),PHE(12288),SMLP(12288),XNEW(12288)
      HC=SIN(.5*D)/64.
      BIGP0=(CH-CSH0)**2+(CZ-CSZ0)**2+(CE-CSE0)**2
      DO 200 JIT=1,100
C
C COMPUTES MATRIX A(X) AND B(X)
C
         CALL FUNCT(X,F,PHZ,PHE,PHIHH,PHIHZ,PHIHE,PHIZZ,PHIZE,PHIEE)
C
C     COMPUTESTHE INVERSE OF THE MATRIX B(X)
C
         CHH=PHIZZ*PHIEE-PHIZE**2
         CHZ=-PHIHZ*PHIEE+PHIZE*PHIHE
         CHE=PHIHZ*PHIZE-PHIZZ*PHIHE
         CZZ=PHIHH*PHIEE-PHIHE**2
         CZE=-PHIHH*PHIZE+PHIHZ*PHIHE
         CEE=PHIHH*PHIZZ-PHIHZ**2
         FACT=PHIHH*CHH+PHIHZ*CHZ+PHIHE*CHE
C
C       COMPUTES THE LAGRANGE MULTIPLIERS LAMBDA VECTOR
C
         HLAM=(CHH*(CH-CSH0)+CHZ*(CZ-CSZ0)+CHE*(CE-CSE0))/FACT
         ZLAM=(CHZ*(CH-CSH0)+CZZ*(CZ-CSZ0)+CZE*(CE-CSE0))/FACT
         ELAM=(CHE*(CH-CSH0)+CZE*(CZ-CSZ0)+CEE*(CE-CSE0))/FACT
         NXY=N/3

         I2=NXY+NXY
C
C       COMPUTES A PERTURBATION OF THE NOMINAL POINT X
C
         DO 10 I=1,I2
            SMLP(I)=-PHZ(I)*ZLAM-PHE(I)*ELAM
   10     XNEW(I)=X(I)+SMLP(I)
          I=I2
          DO 11 KJ=1,32
             CJ=COS((KJ-.5)*D)
             DO 11 KI=1,128
                I=I+1
                SMLP(I)=-HLAM*CJ*HC-PHZ(I)*ZLAM-PHE(I)*ELAM
   11     XNEW(I)=X(I)+SMLP(I)
C
C        COMPUTES THE INTEGRAL INVARIANTS AT THE POINT XNEW
C
          CALL LOOK(XNEW,F,.FALSE.)
          BIGP=(CH-CSH0)**2+(CZ-CSZ0)**2+(CE-CSE0)**2
          IF(BIGP.LT.BIGP0) GO TO 100
          ALPH=1.
          DO 30 KIT=1,20
             ALPH=ALPH/2.
             DO 20 I=1,N
   20        XNEW(I)=X(I)+ALPH*SMLP(I)
             CALL LOOK(XNEW,F,.FALSE.)
             BIGP=(CH-CSH0)**2+(CZ-CSZ0)**2+(CE-CSE0)**2
             IF(BIGP.LT.BIGP0) GO TO 100
   30     CONTINUE
          WRITE(6,40)
   40 FORMAT(' NO LUCK - 20 BISECTIONS PERFORMED')
          STOP
  100     DO 110 I=1,N
  110     X(I)=XNEW(I)
          IF(BIGP.LE.1.E-10) GO TO 220
  200 CONTINUE
      WRITE(6,210)
  210 FORMAT(' NO LUCK - 100 DIRECTIONS TRIED')
      STOP
  220 WRITE(6,230) CH,CZ,CE
  230 FORMAT(' CONSTRAINTS AFTER RESTO-ALGORITHM ',3E16.6)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CALCULATES THE GRADIENTS REQUIRED BY THE
C       CONSTRAINT RESTORATION ALGORITHM
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FUNCT(X,F,PHZ,PHE,PHIHH,PHIHZ,PHIHE,PHIZZ,PHIZE,PHIEE
     +)
      DIMENSION X(128,32,3),Q(128,32),PHZ(128,32,3),PHE(128,32,3),F(32)
      COMMON/C0/CSH0,CSZ0,CSE0
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      COMMON/CC/CH,CZ,CE
      HC=SIN(.5*D)/64.
      C2=DS
      DO 10 I=1,128
         IP1=I+1
         IF(IP1.GT.128) IP1=IP1-128
         IM1=I-1
         IF(IM1.LT.1) IM1=IM1+128
         I64=I+64
         IF(I64.GT.128) I64=I64-128
         DO 10 J=1,32
            PMJP1=1.


            IJP1=I
            JP1=J+1
            IF(JP1.LE.32) GO TO 1
            PMJP1=-1.
            IJP1=I64
            JP1=32
    1       JM1=J-1
            IF(JM1.GE.1) GO TO 2
            JM1=1
    2       CJ=COS((J-.5)*D)
            C1=C2*CJ
            Q(I,J)=((X(IP1,J,2)-X(IM1,J,2))/C1-(PMJP1*X(IJP1,JP1,1)-
     +      X(I,JM1,1))/C2+F(J))/X(I,J,3)
   10 CONTINUE
      ADH=A*D/2.
      ADSQ=(A*D)**2
      ADSH=ADSQ/2.
      PHIHH=0.
      PHIHZ=0.
      PHIHE=0.
      PHIZZ=0.
      PHIEE=0.
      PHIZE=0.
      DO 40 I=1,128
         IP1=I+1
         IF(IP1.GT.128) IP1=IP1-128
         IM1=I-1
         IF(IM1.LT.1) IM1=IM1+128
         DO 30 J=2,31
            JP1=J+1
            JM1=J-1
            CJP1=COS((JP1-.5)*D)
            CJM1=COS((JM1-.5)*D)
            PHZ(I,J,1)=(CJP1*Q(I,JP1)-CJM1*Q(I,JM1))*ADH
   30    CONTINUE
         PHZ(I,1,1)=COS(1.5*D)*Q(I,2)*ADH
         PHZ(I,32,1)=-COS(30.5*D)*Q(I,31)*ADH
         DO 40 J=1,32
            CJ=COS((J-.5)*D)
C
C        COMPUTES COMPONENTS OF THE MATRIX A(X)
C
            PHZ(I,J,2)=(Q(IM1,J)-Q(IP1,J))*ADH
            PHZ(I,J,3)=-(Q(I,J)**2)*ADSH*CJ
            PHE(I,J,1)=X(I,J,1)*X(I,J,3)*ADSQ*CJ
            PHE(I,J,2)=X(I,J,2)*X(I,J,3)*ADSQ*CJ
            PHE(I,J,3)=(X(I,J,1)**2+X(I,J,2)**2+2.*G*X(I,J,3))*ADSH*CJ
C
C        COMPUTE THE COMPONENTS OF THE MATRIX B(X)
C
            PHIHH=PHIHH+(CJ*HC)**2
            PHIHZ=PHIHZ+PHZ(I,J,3)*CJ*HC
            PHIHE=PHIHE+PHE(I,J,3)*CJ*HC
            PHIZZ=PHIZZ+PHZ(I,J,1)**2+PHZ(I,J,2)**2+PHZ(I,J,3)**2
            PHIEE=PHIEE+PHE(I,J,1)**2+PHE(I,J,2)**2+PHE(I,J,3)**2
            PHIZE=PHIZE+PHZ(I,J,1)*PHE(I,J,1)+PHZ(I,J,2)*PHE(I,J,2)+
     +      PHZ(I,J,3)*PHE(I,J,3)
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     OUTPUTS INFORMATION EVERY 24 HOURS
C
C       IF OP IS 1,   PRODUCE PRINTER-PLOT OF HEIGHT FIELD
C       IF OP IS 2,   PRODUCE GARAPHICS CONTOURS AND VELOCITY FIELD
C       IF OP IS 3,   SAVE HEIGHT,U AND V IN A FILE FOR LATER USE.
C       IF OP IS 4,   DO 1 AND 3
C       IF OP IS 5,   DO 2 AND 3
C       ELSE          PRODUCE NO GRAPHICS,SAME AS OP 3
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DAYEND(X,F,ID,OP)
      INTEGER OP
      DIMENSION X(128,96),F(32)
      WRITE(6,1) ID
    1 FORMAT('1VALUES AFTER',I2,' DAY(S)')
      IF((OP.EQ.1).OR.(OP.EQ.4))CALL MAPPA(X(1,65),2000.,128,32)
      IF((OP.EQ.2).OR.(OP.EQ.5))
     +      CALL PLOT(X(1,65),X(1,1),X(1,32),128,32)
      IF((OP.GE.3).AND.(OP.LE.5))
     +      WRITE(10,88) ((X(I,J),I=1,128),J=1,96)
88      FORMAT(1X,8E16.10)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE FILTER IMPLEMENTS THE
C     17-POINT SHAPIRO FILTER
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FILTER(X,IUVH)
      DIMENSION X(128,32)
      COMMON/CW/W(144)
      PMEQ=1.
      PMPOLE=1.
      IF(IUVH.EQ.2) PMEQ=-1.
      IF(IUVH.LE.2) PMPOLE=-1.
C
C     SMOOTH ALONG EACH LATITUDE DIRECTION
C
      DO 100 I=1,128
         I64=I+64
         IF(I64.GT.128) I64=I64-128
         DO 10 J=1,32
   10    W(J+8)=X(I,J)
         DO 20 J=1,8
   20     W(9-J)=PMEQ*X(I,J)
          DO 30 J=25,32
   30     W(73-J)=PMPOLE*X(I64,J)
          DO 40 J=1,32
   40     X(I,J)=(-W(J)+16*W(J+1)-120*W(J+2)+560*W(J+3)-1820*W(J+4)
     +    +4368*W(J+5)-8008*W(J+6)+11440*W(J+7)+52666*W(J+8) +11440*W(J+
     +    9)-8008*W(J+10)+4368*W(J+11)-1820*W(J+12) +560*W(J+13)-120*W(J+
     +    14)+16*W(J+15)-W(J+16))/65536.
  100 CONTINUE
C
C        SMOOTH ALONG EACH LONGITUDE DIRECTION
C
      DO 200 J=1,32
          DO 110 I=1,128
  110     W(I+8)=X(I,J)
          DO 120 I=121,128
  120     W(I-120)=X(I,J)
          DO 130 I=1,8
  130     W(136+I)=X(I,J)
          DO 140 I=1,128
  140     X(I,J)=(-W(I)+16*W(I+1)-120*W(I+2)+560*W(I+3)-1820*W(I+4)
     +    +4368*W(I+5)-8008*W(I+6)+11440*W(I+7)+52666*W(I+8) +11440*W(I+
     +    9)-8008*W(I+10)+4368*W(I+11)-1820*W(I+12) +560*W(I+13)-120*W(I+
     +    14)+16*W(I+15)-W(I+16))/65536.
  200 CONTINUE

      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C        SUBROUTINE STEP1
C      ADVANCES INTEGRATION ONE STEP BY LEAPFROGGING
C      FIRST STEP
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE STEP1(X1,X0,F)
      DIMENSION X1(128,96),X0(128,96),F(32)
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      C1=DT/DS
      AHLF=ALF/2.
      DO 10 I=1,128
         IP1=I+1
C
C DEALSWITH PERIODIC BOUNDARY CONDITIONS IN LONGITUDE DIRECTION
C
         IF(IP1.GT.128) IP1=IP1-128
         IM1=I-1
         IF(IM1.LT.1) IM1=IM1+128
         IPP=I+IP
         IF(IPP.GT.128) IPP=IPP-128
         IMP=I-IP
         IF(IMP.LT.1) IMP=IMP+128
         I64=I+64
         IPP64=IPP+64
         IMP64=IMP+64
         IF(I64.GT.128) I64=I64-128
         IF(IPP64.GT.128) IPP64=IPP64-128
         IF(IMP64.GT.128) IMP64=IMP64-128
         DO 10 J=1,32
C
C HANDLES BOUNDARY CONDITIONS AT NORTH POLE POINT
C
            PMJP1=1.
            IJP1=I
            JP1=J+1
            IF(JP1.LE.32) GO TO 1
            PMJP1=-1.
            IJP1=I64
            JP1=32
    1       PMJPQ=1.
            IJPQ=I
            IPPJPQ=IPP
            IMPJPQ=IMP
            JPQ=J+IQ
            IF(JPQ.LE.32) GO TO 2
            PMJPQ=-1.
            IJPQ=I64
            IPPJPQ=IPP64
            IMPJPQ=IMP64
            JPQ=65-J-IQ
    2       PMJM1=1.
            JM1=J-1
            IF(JM1.GE.1) GO TO 3
            PMJM1=-1.
            JM1=1
    3        PMJMQ=1.
             JMQ=J-IQ
             IF(JMQ.GE.1) GO TO 4
             PMJMQ=-1.
             JMQ=1-J+IQ
    4        CJ=COS((J-.5)*D)
             CJPQ=COS((J+IQ-.5)*D)
             CJMQ=COS((J-IQ-.5)*D)
             TJ=TAN((J-.5)*D)
             TJPQ=TAN((J+IQ-.5)*D)
             TJMQ=TAN((J-IQ-.5)*D)
             C2=G/CJ/IP
             C3=G/IQ
C
C        INTEGRATES MOMENTUM EQUATION RELATED TO THE
C           VELOCITY COMPONENT U USING LEAPFROGGING
C
             X1(I,J)=X0(I,J )-C1*(X0(I,J)/CJ*(X0(IP1,J )-X0(IM1,J
     +       ))+ X0(I,J+32)*(PMJP1*X0(IJP1,JP1 ) -X0(I,JM1 ))+C2*(
     +       X0(IPP,J+64)-X0(IMP,J+64)))+DT*((1.-ALF)*(F(J)+X0(I,J)*TJ/A)
     +       * X0(I,J+32)+AHLF*(F(J)+X0(IPP,J)*TJ/A)*X0(IPP,J+32)+AHLF*(
     +       F(J)+ X0(IMP,J)*TJ/A)*X0(IMP,J+32))
C
C        INTEGRATES MOMENTUM EQUATION RELATED TO THE
C           VELOCITY COMPONENT V USING LEAPFROGGING
C
             X1(I,J+32)=X0(I,J+32)-C1*(X0(I,J)/CJ*(X0(IP1,J+32)-X0(IM1,J+
     +       32))+ X0(I,J+32)*(PMJP1*X0(IJP1,JP1+32)-PMJM1*X0(I,JM1+32))+
     +       C3*( X0(IJPQ,JPQ+64)-X0(I,JMQ+64)))-DT*((1.-ALF)*(F(J)+X0(I,
     +       J)*TJ/A)* X0(I,J)+AHLF*((F(JPQ)+PMJPQ*X0(IJPQ,JPQ)*TJPQ/A)*
     +       PMJPQ *X0(IJPQ,JPQ)+(PMJMQ*F(JMQ)+X0(I,JMQ)*TJMQ/A)*X0(I,
     +       JMQ)))
C
C        INTEGRATES CONTINUITY EQUATION USING LEAGFROGGING
C
             X1(I,J+64)=X0(I,J+64)-C1*(X0(I,J)/CJ*(X0(IP1,J+64)-X0(IM1,J+
     +       64))+ X0(I,J+32)*(X0(IJP1,JP1+64)-X0(I,JM1+64))+X0(I,J+64)/
     +       CJ*(((1.-ALF)*(X0(IPP,J)-X0(IMP,J))+AHLF*(PMJPQ*X0(IPPJPQ,
     +       JPQ)-PMJPQ* X0(IMPJPQ,JPQ)+X0(IPP,JMQ)-X0(IMP,JMQ)))/IP+((1.
     +       -ALF)*(PMJPQ* X0(IJPQ,JPQ+32)*CJPQ-PMJMQ*X0(I,JMQ+32)*CJMQ)+
     +       AHLF*(PMJPQ* X0(IPPJPQ,JPQ+32)*CJPQ-PMJMQ*X0(IPP,JMQ+32)*
     +       CJMQ+PMJPQ* X0(IMPJPQ,JPQ+32)*CJPQ-PMJMQ*X0(IMP,JMQ+32)*
     +       CJMQ))/IQ))
   10 CONTINUE
C
C        IMPOSE 'A POSTERIORI' INTEGRAL INVARIANTS RESTORATION
C
      CALL FORCE(X1,F)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C        SUBROUTINE SNEXT
C     ADVANCES NUMERICAL INTEGRATION ONE STEP BY LEAPFROGGING
C     THE SECOND AND SUBSEQUENT TIME STEPS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SNEXT(X1,X0,X,F)
      DIMENSION X1(128,96),X0(128,96),X(128,96),F(32)
      COMMON/CA/D,G,DS,A,IP,IQ,ALF,DT
      C1=DT/(D*A)
      TDT=DT+DT
      AHLF=ALF/2.
      DO 10 I=1,128
C
C DEALS WITH PERIODIC BOUNDARY CONDITIONS IN LONGITUDE DIRECTION
C
         IP1=I+1
         IF(IP1.GT.128) IP1=IP1-128
         IM1=I-1
         IF(IM1.LT.1) IM1=IM1+128
         IPP=I+IP
         IF(IPP.GT.128) IPP=IPP-128
         IMP=I-IP
         IF(IMP.LT.1) IMP=IMP+128
         I64=I+64
         IPP64=IPP+64
         IMP64=IMP+64
         IF(I64.GT.128) I64=I64-128
         IF(IPP64.GT.128) IPP64=IPP64-128
         IF(IMP64.GT.128) IMP64=IMP64-128
         DO 10 J=1,32
```

```
C
C HANDLES BOUNDARY CONDITIONS AT NORTH POLE POINT
C
          PMJP1=1.
          IJP1=I
          JP1=J+1
          IF(JP1.LE.32) GO TO 1
          PMJP1=-1.
          IJP1=I64
          JP1=32
    1     PMJPQ=1.
          IJPQ=I
          IPPJPQ=IPP
          IMPJPQ=IMP
          JPQ=J+IQ
          IF(JPQ.LE.32) GO TO 2
          PMJPQ=-1.
          IJPQ=I64
          IPPJPQ=IPP64
          IMPJPQ=IMP64
          JPQ=65-J-IQ
    2     PMJM1=1.
          JM1=J-1
          IF(JM1.GE.1) GO TO 3
          PMJM1=-1.
          JM1=1
    3     PMJMQ=1.
          JMQ=J-IQ
          IF(JMQ.GE.1) GO TO 4
          PMJMQ=-1.
          JMQ=1-J+IQ
    4     CJ=COS((J-.5)*D)
          CJPQ=COS((J+IQ-.5)*D)
          CJMQ=COS((J-IQ-.5)*D)
          TJ=TAN((J-.5)*D)
          TJPQ=TAN((J+IQ-.5)*D)
          TJMQ=TAN((J-IQ-.5)*D)
          C2=G/CJ/IP
          C3=G/IQ
C
C       INTEGRATES MOMENTUM EQUATION RELATED TO THE
C          VELOCITY COMPONENT U USING LEAPFROGGING
C
          X1(I,J)=X0(I,J )-C1*(X(I,J)/CJ*(X(IP1,J )-X(IM1,J ))+
     +    X(I,J+32)*(PMJP1*X(IJP1,JP1 )- X(I,JM1 ))+C2*( X(IPP,J+64)-
     +    X(IMP,J+64)))+TDT*((1.-ALF)*(F(J)+X(I,J)*TJ/A)* X(I,J+32)+
     +    AHLF*(F(J)+X(IPP,J)*TJ/A)*X(IPP,J+32)+AHLF*(F(J)+ X(IMP,J)*
     +    TJ/A)*X(IMP,J+32))
C
C       INTEGRATES MOMENTUM EQUATION RELATED TO THE
C          VELOCITY COMPONENT V USING LEAPFROGGING
C
          X1(I,J+32)=X0(I,J+32)-C1*(X(I,J)/CJ*(X(IP1,J+32)-X(IM1,J+32)
     +    )+ X(I,J+32)*(PMJP1*X(IJP1,JP1+32)-PMJM1*X(I,JM1+32))+C3*(
     +    X(IJPQ,JPQ+64)-X(I,JMQ+64)))-TDT*((1.-ALF)*(F(J)+X(I,J)*TJ/
     +    A)* X(I,J)+AHLF*((F(JPQ)+PMJPQ*X(IJPQ,JPQ)*TJPQ/A)*PMJPQ
     +    *X(IJPQ,JPQ)+(PMJMQ*F(JMQ)+X(I,JMQ)*TJMQ/A)*X(I,JMQ)))


C
C       INTEGRATES CONTINUITY EQUATION USING LEAGFROGGING
C
          X1(I,J+64)=X0(I,J+64)-C1*(X(I,J)/CJ*(X(IP1,J+64)-X(IM1,J+64)
     +    )+ X(I,J+32)*(X(IJP1,JP1+64)-X(I,JM1+64))+X(I,J+64)/CJ*(((1.
     +    -ALF) *(X(IPP,J)-X(IMP,J))+AHLF*(PMJPQ*X(IPPJPQ,JPQ)-PMJPQ*
     +    X(IMPJPQ,JPQ)+X(IPP,JMQ)-X(IMP,JMQ)))/IP+((1.-ALF)*(PMJPQ*
     +    X(IJPQ,JPQ+32)*CJPQ-PMJMQ*X(I,JMQ+32)*CJMQ)+AHLF*(PMJPQ*
     +    X(IPPJPQ,JPQ+32)*CJPQ-PMJMQ*X(IPP,JMQ+32)*CJMQ+PMJPQ*
     +    X(IMPJPQ,JPQ+32)*CJPQ-PMJMQ*X(IMP,JMQ+32)*CJMQ))/IQ))
   10 CONTINUE
C
C       IMPOSES'A POSTERIORI' INTEGRAL INVARIANTS RESTORATION
C
      CALL FORCE(X1,F)
      RETURN
      END
```