NASA Technical Memorandum 86227

Documentation of the Goddard Laboratory for Atmospheres Fourth-Order Two-Layer Shallow Water Model

FEBRUARY 1986

NASA

NASA Technical Memorandum 86227

Documentation of the Goddard Laboratory for Atmospheres Fourth-Order Two-Layer Shallow Water Model

Compiled by
Lawrence L. Takacs
Sigma Data Services Corporation
for Goddard Laboratory for Atmospheres
Goddard Space Flight Center
Greenbelt, Maryland

NASA
National Aeronautics
and Space Administration
Scientific and Technical
Information Branch
1986

TABLE OF CONTENTS

I. Introduction

The purpose of this documentation is to familiarize the user with the theory and numerical treatment used in the 2-level GLA global fourth-order shallow water model. This model was designed to emulate the horizontal finite-differences used by the GLA Fourth-Order General Circulation Model (see Kalnay *et al.*, 1983) in addition to its grid structure, form of high-latitude and global filtering, and time-integration schemes. Although shallow water models have been previously used to compare with GCMs, they have the disadvantage that, being barotropic, they cannot reproduce one of the most fundamental processes in the atmosphere. For this reason we have decided to develop a 2-layer shallow water model, which can also be used as a conventional 1-layer model.

It has been found that this model provides a very useful tool in assessing the impact of numerical techniques under consideration for use in the GLA GCM. Experiments on the high-latitude and global filter (Takacs and Balgovind, 1983a,b), linear and non-linear normal mode initialization (Navon *et al.*, 1985, Takacs, *et al.* 1985), aerosol tracer advection, and multi-variate data assimilation have been performed using this model with satisfying results.

Section II develops the equations used for the two-layer shallow water system. Section III describes the grid structure and the horizontal finite-difference equations used by the model, while Section IV considers the addition of friction and external forcing to the model. The time-integration schemes used by the model are presented in Section V. In Section VI an examination is made of the types of filters needed by the model, i.e., high-latitude filtering to control linear instability due to fast-moving gravity waves near the poles, and global filtering to control non-linear instability due to aliasing. Section VII presents the normal mode initialization process and its implementation into the model, while Section VIII discusses the addition of a passive tracer. In Section IX a user's guide is provided instructing the user on how to create intitial conditions, execute the shallow water model,

and post-process the data history. Section X contains the list of references used, while Section XI contains sample outputs.

II. Model Equations

a. Away from pole

The governing equations for the shallow water model are based on a two-layer inviscid fluid system using the quasi-static approximation. Referring to Fig. 2.1, the momentum and hydrostatic equations for each layer are given by

$$\frac{D}{Dt}\mathbb{V} = -\frac{1}{\rho}\nabla P - f\,\hat{k}\times\mathbb{V}, \tag{2.1}$$

and

$$\frac{\partial p}{\partial z} = -\rho g, \tag{2.2}$$

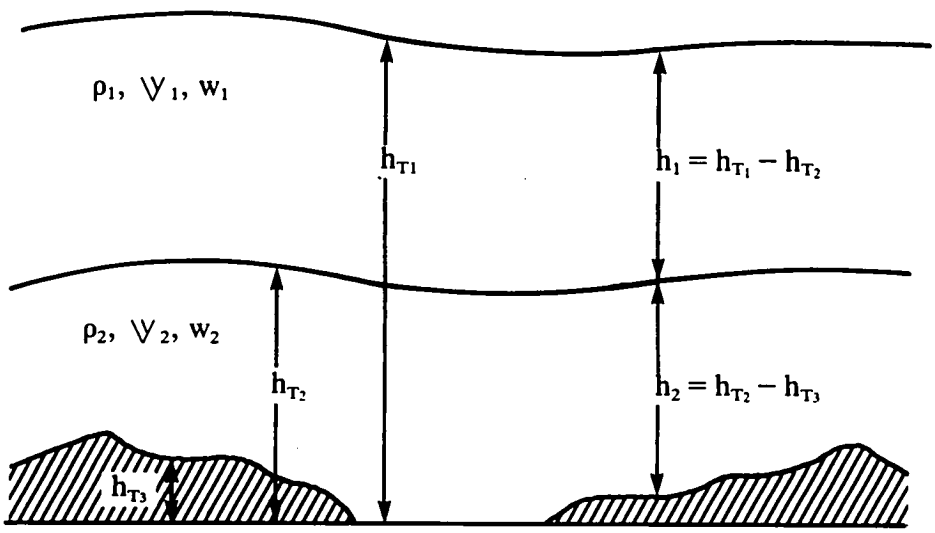


Fig. 2.1

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbb{V}\cdot\nabla$.

Integrating (2.2) from any given height z within the layer to the top of the model, we have

Layer 1:

$$\int_z^{h_{T_1}} \frac{\partial p_1}{\partial z}\, dz = -\rho_1 g\,(h_{T_1} - z),\ z \geq h_{T_2} \tag{2.3}$$

Layer 2:

$$\int_z^{h_{T_2}} \frac{\partial p_2}{\partial z}\, dz + \int_{h_{T_2}}^{h_{T_1}} \frac{\partial p_1}{\partial z}\, dz = -\rho_2 g(h_{T_2} - z) - \rho_1 g(h_{T_1} - h_{T_2}),\ z \leq h_{T_2} \tag{2.4}$$

From (2.3) and (2.4) we see that

$$P_1(z) = \rho_1 g(h_{T_1} - z),\ z > h_{T_2} \tag{2.5}$$

and

$$P_2(z) = \rho_2 g(h_{T_2} - z) + \rho_1 g(h_{T_1} - h_{T_2}),\ z < h_{T_2} \tag{2.6}$$

Here we used $P_1(h_{T_1}) = 0$ and $P_1(h_{T_2}) = P_2(h_{T_2})$. Taking the horizontal gradient of (2.5) and (2.6) yields

$$\frac{1}{\rho_1} \nabla P_1 = g \nabla h_{T_1} \tag{2.7}$$

and

$$\frac{1}{\rho_2} \nabla P_2 = g \nabla h_{T_2} + \frac{\rho_1}{\rho_2} g \nabla (h_{T_1} - h_{T_2}). \tag{2.8}$$

Using (2.7) and (2.8), the momentum equations become

$$\frac{D}{Dt} \mathbb{V}_1 = -f\, \hat{k} \times \mathbb{V}_1 - g \nabla h_{T_1} \tag{2.9}$$

and

$$\frac{D}{Dt} \mathbb{V}_2 = -f\, \hat{k} \times \mathbb{V}_2 - g \nabla h_{T_2} - \frac{\rho_1}{\rho_2} g \nabla (h_{T_1} - h_{T_2}). \tag{2.10}$$

or, in general for layer k,

$$\frac{D}{Dt} \mathbb{V}_k = -f\, \hat{k} \times \mathbb{V}_k - g \nabla [\alpha_k h_{T_1} + (1 - \alpha_k) h_{T_2}] \tag{2.11}$$

where $\alpha_1 = 1$, $\alpha_2 = \rho_1/\rho_2$, and

$$\frac{D}{Dt_k} = \frac{\partial}{\partial t} + \mathbb{V}_k \cdot \nabla.$$

The continuity equation for layer k is given by

$$\nabla \cdot \mathbb{V}_k + \frac{\partial w_k}{\partial z} = 0. \tag{2.12}$$

Integrating (2.12) from h_{Tk+1} to h_{Tk} yields

$$\frac{\partial h_k}{\partial t} = -\nabla \cdot (h_k \mathbb{V}_k) \tag{2.13}$$

In summary, the momentum and continuity equations for layer k are given by

$$\frac{\partial}{\partial t} \mathbb{V}_k = -(\mathbb{V}_k \cdot \nabla)\mathbb{V}_k - f\,\hat{k} \times \mathbb{V}_k - g\nabla[\alpha_k h_{T_1} + (1-\alpha_k) h_{T_2}] \tag{2.14}$$

$$\frac{\partial h_k}{\partial t} = -\nabla \cdot (h_k \mathbb{V}_k). \tag{2.15}$$

The flux form of the shallow water equations may be obtained by multiplying the momentum equation by h_k and adding it to the continuity equation multiplied by V_k. Doing so we find that the shallow water equations in component form on a sphere are given by

$$\frac{\partial(hu)}{\partial t} = -\frac{1}{a\cos\phi}\left[\frac{\partial}{\partial\lambda}(hu)u + \frac{\partial}{\partial\phi}(hv\cos\phi)u\right] + \left(f + \frac{u\tan\phi}{a}\right)hv$$

$$-\frac{gh}{a\cos\phi}\frac{\partial}{\partial\lambda}[\alpha h_{T_1} + (1-\alpha)h_{T_2}] \tag{2.16}$$

$$\frac{\partial(hv)}{\partial t} = -\frac{1}{a\cos\phi}\left[\frac{\partial}{\partial\lambda}(hu)v + \frac{\partial}{\partial\phi}(hv\cos\phi)v\right] - \left(f + \frac{u\tan\phi}{a}\right)hu$$

$$-\frac{gh}{a}\frac{\partial}{\partial\phi}[\alpha h_{T_1} + (1-\alpha)h_{T_2}] \tag{2.17}$$

and

$$\frac{\partial h}{\partial t} = -\frac{1}{a\cos\phi}\left[\frac{\partial(hu)}{\partial\lambda} + \frac{\partial(hv\cos\phi)}{\partial\phi}\right]. \tag{2.18}$$

In equations (2.16)-(2.18), we have dropped the subscript k for simplicity.

b. At the pole

The equations used at the pole are obtained, following Kalnay *et al.* (1983), by first transforming the equations using a stereographic projection, and then area-averaging the equations over a polar cap.

The stereographic projection of u and v result in a unique U_p and V_p, independent of longitude λ. These are given by

$$U_{pm} = -u \sin(\lambda) - (-1)^m v \cos(\lambda) \tag{2.19}$$

$$V_{pm} = (-1)^m u \cos(\lambda) - v \sin(\lambda) \tag{2.20}$$

where m=1 for the south pole, and m=2 for the north pole. Multiplying (2.16) and (2.17) by the appropriate sin (λ) or cos (λ) and adding, we obtain

$$\frac{\partial}{\partial t} hp_m = -\frac{1}{a \cos\phi}\left[\frac{\partial}{\partial\lambda}(hu) + \frac{\partial}{\partial\phi}(hv \cos\phi)\right] \tag{2.21}$$

$$\frac{\partial}{\partial t}(hUp_m) = \frac{-1}{a \cos\phi}\left[\frac{\partial}{\partial\lambda}(hu)Up_m + \frac{\partial}{\partial\phi}(hv \cos\phi)Vp_m\right] + f\,hVp_m$$

$$+\frac{gh}{a}\left[\frac{\sin\lambda}{\cos\phi}\frac{\partial}{\partial\lambda} + (-1)^m \cos\lambda \frac{\partial}{\partial\phi}\right](\alpha h_{T_1} + (1-\alpha)h_{T_2}) \tag{2.22}$$

$$\frac{\partial}{\partial t}(hVp_m) = \frac{-1}{a \cos\phi}\left[\frac{\partial}{\partial\lambda}(hu)Vp_m + \frac{\partial}{\partial\phi}(hv \cos\phi)Vp_m\right] - f\,hUp_m$$

$$-\frac{gh}{a}\left[(-1)^m \frac{\cos\lambda}{\cos\phi}\frac{\partial}{\partial\lambda} - \sin\lambda \frac{\partial}{\partial\phi}\right](\alpha h_{T_1} + (1-\alpha)h_{T_2}) \tag{2.23}$$

Equations (2.21)-(2.23) are then area-averaged over a small polar cap. The results of this integration will be shown in Section III.

III. Horizontal finite differences

a. Away from pole

Following Kalnay *et al.* (1983), the GLA shallow water model uses a non-staggered A-grid (Arakawa, 1972), i.e. the height and wind components are defined at every grid-point:

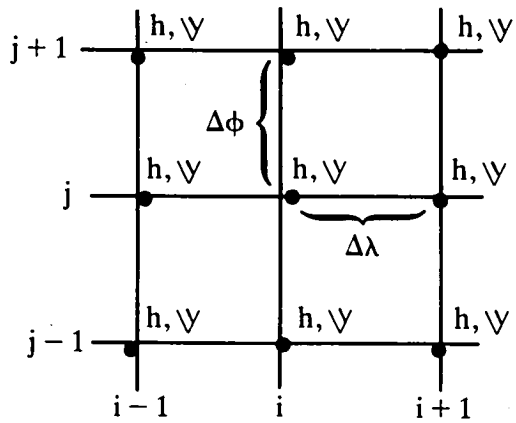


Fig. 3.1

Defining IM as the number of grid-points in the zonal direction and JM as the number of grid-points in the meridional direction, then the grid increments λ_i and ϕ_j are given by

$$\lambda_i = -\pi + (i-1)\Delta\lambda, \qquad \Delta\lambda = 2\pi/IM \tag{3.1}$$

$$\phi_j = -\pi/2 + (j-1)\Delta\phi, \qquad \Delta\phi = \pi/(JM-1) \tag{3.2}$$

As an aid in notation, let us define the following:

$$\delta_x(\bar{q}^x)_j = \frac{\bar{q}^x_{j+1/2} - \bar{q}^x_{j-1/2}}{\Delta x} \tag{3.3}$$

$$\delta_{2x}(\bar{q}^{2x})_j = \frac{\bar{q}^{2x}_{j+1} - \bar{q}^{2x}_{j-1}}{2\Delta x} \tag{3.4}$$

$$\bar{q}^x_{j+1/2} = \frac{q_{j+1} + q_j}{2} \tag{3.5}$$

$$\bar{q}^{2x}_{j+1} = \frac{q_{j+2} + q_j}{2}. \tag{3.6}$$

Then $\frac{\partial q}{\partial x}$ may be approximated to fourth-order accuracy by

$$\left(\frac{\partial q}{\partial x}\right)_j = \frac{4}{3}\,\delta_x(\bar{q}^x)_j - \frac{1}{3}\,\delta_{2x}(\bar{q}^{2x})_j + o(\Delta x^4). \tag{3.7}$$

When δ operates on a single quantity as in the above example, (3.7) simplifies to

$$\left(\frac{\partial q}{\partial x}\right)_j = \frac{4}{3}\left(\frac{q_{j+1} - q_{j-1}}{2\Delta x}\right) - \frac{1}{3}\left(\frac{q_{j+2} - q_{j-2}}{4\Delta x}\right) + o(\Delta x^4). \tag{3.8}$$

Before writing the finite-difference version of the momentum and continuity equations, let us define the mass variables U and V as

$$U = hu \tag{3.9}$$

$$V = hv \cos \phi \tag{3.10}$$

and the generalized height h_T as

$$h_T = \alpha h_{T_1} + (1 - \alpha) h_{T_2}. \tag{3.11}$$

Then, using the notation of (3.3)–(3.6) together with the definitions (3.9)–(3.11), the model equations (2.16)–(2.18) may be written in finite-difference form as

$$\frac{\partial}{\partial t}(hu)_{i,j} = \frac{-1}{a \cos \phi_j}\left[\frac{4}{3}\,\delta_\lambda(\bar{U}^\lambda \bar{u}^\lambda) - \frac{1}{3}\,\delta_{2\lambda}(\bar{U}^{2\lambda} \bar{u}^{2\lambda}) + \frac{4}{3}\,\delta_\phi(\bar{V}^\phi \bar{u}^\phi) - \frac{1}{3}\,\delta_{2\phi}(\bar{V}^{2\phi} \bar{u}^{2\phi})\right]$$

$$+ \left(f + \frac{u \tan \phi}{a}\right)_{i,j} (hv)_{i,j} - \frac{g h_{i,j}}{a \cos \phi_j}\left[\frac{4}{3}\,\delta_\lambda(\bar{h}_T^\lambda) - \frac{1}{3}\,\delta_{2\lambda}(\bar{h}_T^{2\lambda})\right] \tag{3.12}$$

$$\frac{\partial}{\partial t}(hv)_{i,j} = \frac{-1}{a \cos \phi_j}\left[\frac{4}{3}\,\delta_\lambda(\bar{U}^\lambda \bar{v}^\lambda) - \frac{1}{3}\,\delta_{2\lambda}(\bar{U}^{2\lambda} \bar{v}^{2\lambda}) + \frac{4}{3}\,\delta_\phi(\bar{V}^\phi \bar{v}^\phi) - \frac{1}{3}\,\delta_{2\phi}(\bar{V}^{2\phi} \bar{v}^{2\phi})\right]$$

$$- \left(f + \frac{u \tan \phi}{a}\right)_{i,j} (hu)_{i,j} - \frac{g h_{i,j}}{a}\left[\frac{4}{3}\,\delta_\phi(\bar{h}_T^\phi) - \frac{1}{3}\,\delta_{2\phi}(\bar{h}_T^{2\phi})\right] \tag{3.13}$$

and

$$\frac{\partial}{\partial t} h_{i,j} = \frac{-1}{a \cos \phi_j}\left[\frac{4}{3}\,\delta_\lambda(\bar{U}^\lambda) - \frac{1}{3}\,\delta_{2\lambda}(\bar{U}^{2\lambda}) + \frac{4}{3}\,\delta_\phi(\bar{V}^\phi) - \frac{1}{3}\,\delta_{2\phi}(\bar{V}^{2\phi})\right]. \tag{3.14}$$

It can be shown that (3.12)-(3.14) conserves total energy in finite-difference form. Noting that the time rate-of-change of kinetic and potential energy is given by

$$\frac{\partial}{\partial t} h\frac{1}{2}(u^2 + v^2 + gh_T + gh_s) = u\frac{\partial(hu)}{\partial t} + v\frac{\partial(hv)}{\partial t} - \frac{1}{2}(u^2 + v^2)\frac{\partial h}{\partial t} + gh_T\frac{\partial h}{\partial t} \tag{3.15}$$

where h_s is the height of the surface topography, then after some manipulation we find

$$\frac{\partial}{\partial t}\sum_{i,j}\left[h_{i,j}\frac{1}{2}(u^2 + v^2 + gh_T + gh_s)_{i,j}\, a^2 \cos\phi_j\, \Delta\lambda\, \Delta\phi\right] = 0 \tag{3.16}$$

which is the finite-difference analog of

$$\frac{\partial}{\partial t}\int_0^{2\pi}\int_{-\pi/2}^{\pi/2} h\frac{1}{2}(u^2 + v^2 + gh_T + gh_s)\, a^2 \cos\phi\, d\lambda\, d\phi = 0. \tag{3.17}$$

b. At the pole

As previously mentioned in Section 2, the equations used at the poles are obtained by first transforming the governing model equations using a stereographic projection, and then integrating or area-averaging these equations over a small polar cap.

Let us consider integrating some function $F(\lambda, \phi)$ over a north polar cap:

$$F_{NP} = \int_0^{2\pi}\int_{\pi/2-\Delta\phi}^{\pi/2} F(\lambda, \phi)\, a^2 \cos\phi\, d\lambda\, d\phi. \tag{3.18}$$

Similarly, for the south pole we have

$$F_{SP} = \int_0^{2\pi}\int_{-\pi/2}^{-\pi/2+\Delta\phi} F(\lambda, \phi)\, a^2 \cos\phi\, d\lambda\, d\phi. \tag{3.19}$$

Letting $\phi^1 = -\phi$, (3.19) becomes

$$F_{SP} = \int_0^{2\pi}\int_{\pi/2-\Delta\phi}^{\pi/2} F(\lambda, -\phi^1)\, a^2 \cos\phi^1\, d\lambda\, d\phi^1. \tag{3.20}$$

Since ϕ and ϕ^1 are dummy variables in (3.19) and (3.20), we see that the only difference between integrating $F(\lambda, \phi)$ over the north pole and over the south pole is the substitution of $-\phi$ for ϕ in the function $F(\lambda, \phi)$.

We shall now consider integrating the transformed equations (2.21)-(2.23). Let us define A_c as the area of the polar cap, given by

$$A_c = \int_0^{2\pi} \int_{\pi/2-\Delta\phi}^{\pi/2} a^2 \cos\phi \, d\lambda \, d\phi = 2\pi a^2 (1 - \cos\Delta\phi). \tag{3.21}$$

Then, integrating the continuity equation, we have

$$\overline{\frac{\partial h_p}{\partial t}}^{\Delta\phi} = -\frac{1}{A_c} \int_0^{2\pi} \int_{\pi/2-\Delta\phi}^{\pi/2} \left[\frac{\partial (hu)}{\partial \lambda} + (-1)^m \frac{\partial}{\partial \phi} (hv \cos\phi) \right] a \, d\lambda \, d\phi \tag{3.22}$$

where m = 1 for the south pole and m = 2 for the north pole. The first term on the RHS of (3.22) vanishes since it is being integrated around a complete latitude circle. The second term, however, becomes

$$-\int_0^{2\pi} \int_{\pi/2-\Delta\phi}^{\pi/2} (-1)^m \frac{\partial}{\partial \phi} (hv \cos\phi) \, a \, d\lambda \, d\phi = (-1)^m a \int_0^{2\pi} (hv \cos\phi) \Big/ d\lambda \Big|_{\pi/2-\Delta\phi}. \tag{3.23}$$

Equation (3.23) is approximated in finite-difference form as

$$(-1)^m a \int_0^{2\pi} (hv \cos\phi) \Big/ d\lambda \Big|_{\pi/2-\Delta\phi} \approx (-1)^m \frac{2\pi \, a \sin\Delta\phi}{IM} \sum_{i=1}^{IM} (hv)_{i,j \text{ pole} \pm 1}. \tag{3.24}$$

Substituting (3.24) into (3.22) we have

$$\overline{\frac{\partial h}{\partial t} p}^{\Delta\phi} = (-1)^m \frac{\sin\Delta\phi}{a \, IM \, (1 - \cos\Delta\phi)} \sum_{i=1}^{IM} (hv)_{i,j \text{ pole} \pm 1}. \tag{3.25}$$

In practice, in order to achieve fourth-order accuracy, we repeat the process over a polar cap of $2\Delta\phi$, yielding

$$\overline{\frac{\partial h}{\partial t} p}^{2\Delta\phi} = (-1)^m \frac{\sin 2\Delta\phi}{a \, IM \, (1 - \cos 2\Delta\phi)} \sum_{i=1}^{IM} (hv)_{i,j \text{ pole} \pm 2}. \tag{3.26}$$

Combining (3.25) and (3.26) we obtain

$$\frac{\partial h}{\partial t} p = \frac{4}{3} \overline{\frac{\partial h}{\partial t} p}^{\Delta\phi} - \frac{1}{3} \overline{\frac{\partial h}{\partial t} p}^{2\Delta\phi} \tag{3.27}$$

This entire process is also applied to the transformed momentum equations (2.22) and (2.23), yielding

$$\frac{\partial}{\partial t}(hUp) = \frac{4}{3}\frac{\partial}{\partial t}\overline{(hUp)}^{\Delta\phi} - \frac{1}{3}\frac{\partial}{\partial t}\overline{(hUp)}^{2\Delta\phi} \tag{3.28}$$

and

$$\frac{\partial}{\partial t}(hVp) = \frac{4}{3}\frac{\partial}{\partial t}\overline{(hVp)}^{\Delta\phi} - \frac{1}{3}\frac{\partial}{\partial t}\overline{(hVp)}^{2\Delta\phi} \tag{3.29}$$

where

$$\frac{\partial}{\partial t}\overline{(hUp)}^{n\Delta\phi} = C(n\Delta\phi)\left[(-1)^m \sum_{i=1}^{IM} (hvUp_m)_{i,j\ pole\ \pm n} - gh_{pole} \sum_{i=1}^{IM} h_{T_{i,j\ pole\ \pm n}} \cos\lambda_i\right] + f_{pole}\, h_{pole}\, Vp_m \tag{3.30}$$

$$\frac{\partial}{\partial t}\overline{(hVp)}^{n\Delta\phi} = C(n\Delta\phi)\left[(-1)^m \sum_{i=1}^{IM} (hvVp_m)_{i,j\ pole\ \pm n} - (-1)^m gh_{pole} \sum_{i=1}^{IM} h_{T_{i,j\ pole\ \pm n}} \sin\lambda_i\right] - f_{pole}\, h_{pole}\, Up_m \tag{3.31}$$

and

$$C(n\Delta\phi) = \frac{\sin(n\Delta\phi)}{a\ IM\ (1-\cos(n\Delta\phi))},$$

$$h_T = \alpha h_{T_1} + (1-\alpha)\, h_{T_2}\ .$$

IV. Friction and forcing

Frictional damping and zonal forcing can be added as source and sink terms to the momentum equations as input through the NAMELIST parameters. To examine this, let us consider the tendency of the zonal wind component, neglecting all other processes, as

$$\frac{\partial u}{\partial t} = \frac{(\bar{u} - u)}{\tau} \tag{4.1}$$

where τ is a time constant, and $\bar{u}$ is the mean zonal wind as a function of latitude. Integrating (4.1) in time from t to $t + \Delta t$, we find

$$u(t + \Delta t) = \bar{u}(1 - e^{-\Delta t/\tau}) + u(t)e^{-\Delta t/\tau}. \tag{4.2}$$

Thus, at $\Delta t = 0$ we have

$$u = u(t), \tag{4.3}$$

while for $\Delta t \to \infty$, we obtain

$$u = \bar{u}. \tag{4.4}$$

Letting $\alpha = \Delta t/\tau$, we may expand the exponential as

$$e^{-\alpha} = 1 - \alpha + \alpha^2 - \alpha^3 + \cdots \tag{4.5}$$

Using (4.5) in (4.4) we find

$$u(t + \Delta t) = u(t) + (\bar{u} - u(t))(\alpha - \alpha^2 + \alpha^3 - \cdots). \tag{4.6}$$

In practice, (4.6) is truncated after the quadratic term, giving

$$u(t + \Delta t) = u(t) + \alpha(1 - \alpha)(\bar{u} - u(t)). \tag{4.7}$$

The frictional damping and zonal forcing terms are added after all other hydrodynamic processes are completed. During the model integration, both the u and v wind components are updated, although the zonal mean v wind is assumed to be zero. The mean zonal u wind is computed from the initial condition. The dissipative time constant τ is a NAMELIST parameter.

V. Time integration schemes

Following Kalnay *et al.* (1983), the model has the option of using either the Matsuno time differencing scheme, the leapfrog time scheme, or a combination of both. The choice of which scheme is used is determined by the SWE NAMELIST parameter SCHEME which is set to either 'MATS' or 'LEAP'. The NAMELIST parameter NLEAP is the number of leapfrog time-steps to perform before restarting with a Matsuno time-step. This is done in order to control the separation of solutions arising from the computational mode of the leapfrog scheme.

The Matsuno time scheme is implemented using a predictor-corrector sequence. Letting q^n represent a typical variable which is to be updated to time level n+1, and $D(q^n)$ represent the non-linear space differences evalutated at time n, we have

$$q^* = q^n + \Delta t\, D(q^n) \qquad \text{(Predictor step)}$$

$$q^{n+1} = q^n + \Delta t\, D(q^*) \qquad \text{(Corrector step)}$$

For the leapfrog time differencing scheme, we begin the forecast with the Matsuno scheme thus producing the n+1 time level, and then proceed with the leapfrog scheme using

$$q^{n+1} = q^{n-1} + 2\,\Delta t\, D(q^n) \qquad \text{(Leapfrog)}$$

VI. Filters

a) High-latitude

The process of filtering in global grid-point models is fairly common in meteorological studies. High-latitude filtering is done in order to avoid the use of prohibitively short time-steps required due to fast-moving inertia-gravity waves near the poles. The specific filtering response needed to ensure stability is defined by the space and time differencing schemes used, and by the type of grid structure in the model (Arakawa and Lamb, 1977).

Takacs and Balgovind (1983a) have examined in detail various filtering techniques for use in this model. This work and further studies have shown that, using realistic initial conditions and topography, filtering the time-tendencies of the prognostic fields yields the closest simulation to a control run using a small time-step and no high-latitude filter. Other filtering techniques examined were the prognostic field filter, the energy-conserving filter, and the P.G. filter (see Takacs and Balgovind, 1983a).

In all techniques used, the filtered quantities were fast-Fourier transformed into their wave constituents whose amplitudes were then altered by a wave-dependent damping function. In order to obtain the functional form of the damping function for the time-tendency filter, we shall consider the

linearized zonal momentum and continuity equations with a mean zonal wind U and a mean depth H. Using a non-staggered A grid (Arakawa, 1972) and fourth-order spatial differences we have

$$\frac{\partial}{\partial t} u_{i,j} = \left(\frac{-U}{a \cos \phi_j} \delta(u)_{i,j} - \frac{g}{a \cos \phi_j} \delta(h)_{i,j} \right)^{F_j} \tag{6.1}$$

$$\frac{\partial}{\partial t} h_{i,j} = \left(\frac{-U}{a \cos \phi_j} \delta(h)_{i,j} - \frac{H}{a \cos \phi_j} \delta(u)_{i,j} \right)^{F_j} \tag{6.2}$$

where

$$\delta(q)_{i,j} = \frac{4}{3}\left(\frac{q_{i+i,j} - q_{i-i,j}}{2\Delta\lambda} \right) - \frac{1}{3}\left(\frac{q_{i+2,j} - q_{i-2,j}}{4\Delta\lambda} \right) \tag{6.3}$$

and $(\ )^{Fj}$ denotes that the quantity is filtered with filter function F_j.

Assuming solutions of the form

$$(u, h)_{i,j} = \mathrm{Re}\ \{\hat{u}_j, \hat{h}_j \exp(\bar{i}[i\theta - \nu t])\} \tag{6.4}$$

where $\theta = k\Delta\lambda$ and $\bar{i} = \sqrt{-1}$, and that $(\ )^{Fj}$ implies multiplying the Fourier amplitudes by F_j, then (6.1) and (6.2) become

$$\nu \hat{u}_j = F_j \left(\frac{U}{a \cos \phi_j} \frac{\omega}{\Delta\lambda} \hat{u}_j + \frac{g}{a \cos \phi_j} \frac{\omega}{\Delta\lambda} \hat{h}_j \right) \tag{6.5}$$

$$\nu \hat{h}_j = F_j \left(\frac{U}{a \cos \phi_j} \frac{\omega}{\Delta\lambda} \hat{h}_j + \frac{H}{a \cos \phi_j} \frac{\omega}{\Delta\lambda} \hat{u}_j \right) \tag{6.6}$$

where $\omega = 4/3 \sin\theta - 1/6 \sin 2\theta$. Simplifying (6.5) and (6.6) we obtain

$$(\nu - U\omega_j)\hat{u}_j - g\omega_j \hat{h}_j = 0 \tag{6.7}$$

$$-H\omega_j \hat{u}_j + (\nu - U\omega_j)\hat{h}_j = 0 \tag{6.8}$$

where

$$\omega_j = \frac{\omega F_j}{a \cos \phi_j \Delta\lambda}.$$

For non-trivial solutions, it must be true that

$$\nu = \omega_j (U + Cg) \tag{6.9}$$

where

$$C_g = \sqrt{gH}$$

For the leapfrog or Matsuno time integration schemes, linear stability requires

$$\nu\Delta t \leq 1 . \tag{6.10}$$

Using (6.10) in (6.9) we obtain

$$F_j \leq \frac{a\Delta\lambda}{(U + Cg)\Delta t} \frac{\cos \phi_j}{\omega} \tag{6.11}$$

In practice, we let

$$F_j = \min\left(1, \frac{a\Delta\lambda}{(U + Cg)\Delta t} \frac{\cos \phi_j}{\omega}\right) \tag{6.12}$$

with U = 50 m/sec and $C_g = \sqrt{gH}$. The mean height H is calculated using the initial conditions for a given experiment. Fig. 6.1 shows the damping coefficients as a function of wavenumber for the five latitudes nearest the poles (i.e. ϕ = 86, 82, 78, 74, and 70 degrees) which are filtered in the 4 deg. lat. × 5 deg. long. resolution model. Here we are using H = 10,000 m and Δt = 450 sec.

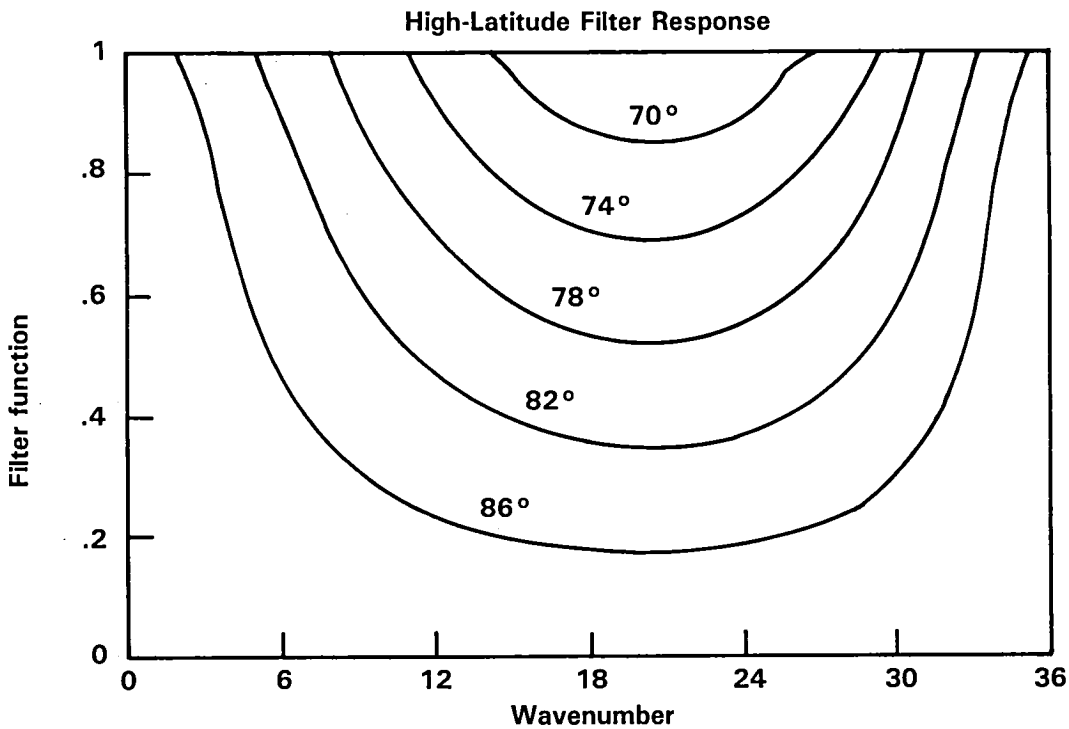


Fig. 6.1

We see that the longest and the shortest scales are not affected by the filter. The filter is the strongest near the 4 grid-length wave, which is characteristic of the fourth-order differences on the A grid.

In summary, the time-tendencies of the prognostic fields, $\left(\text{i.e.}, \frac{\partial h}{\partial t}, \frac{\partial (hu)}{\partial t}, \frac{\partial (hv)}{\partial t}\right)$ are Fourier transformed into their wave consituents whose amplitudes are then multiplied by the damping function F_j given by (6.12). These adjusted fields are then back-transformed and used to update the original fields.

b. Global

Spatial difference schemes which are not enstrophy conserving nor implicitly damping require global filtering of short waves to eliminate the build-up of energy in the shortest wavelengths due to aliasing. Global filtering as applied in the GLA GCM (see Kalnay *et al.*, 1983) is performed by using a 16th-order Shapiro (1979) filter on the sea-level pressure, sea-level temperature, potential temperature and winds. In the GLA shallow water model, the 16th-order Shapiro filter is applied to the wind and height fields at some specified frequency (normally every two simulated hours).

The form of the two-dimensional 16th-order Shapiro filter is simply two passes of the one-dimensional filter applied in succession. To obtain the high order of the filter, eight passes of the 3-point second-order operator are used as given by

$$\bar{q}_{i,j} = [1 - (F^2_\phi)^8][1 - (F^2_\lambda)^8]\, q_{i,j} \tag{6.13}$$

where

$$F^2_\lambda(q_{i,j}) = (q_{i+1,j} - 2q_{i,j} + q_{i-1,j})/4 \tag{6.14}$$

$$F^2_\phi(q_{i,j}) = (q_{i,j+1} - 2q_{i,j} + q_{i,j-1})/4 \tag{6.15}$$

Here, q is any quantity being filtered.

To understand the effect of using the one-dimensional Shapiro filter on a given quantity, let us assume that

$$q_j = \mathrm{Re}\{\hat{q} e^{ikj\Delta\lambda}\} \tag{6.16}$$

Operating on q_j using (6.14) yields

$$F^2(q_j) = (q_{j+1} - 2q_j + q_{j-1})/4 \tag{6.17}$$

$$= \frac{(e^{i\theta} - 2 + e^{-i\theta})}{4} q_j \tag{6.18}$$

where $\theta = k\Delta\lambda$

or

$$F^2(q_j) = -\left(\sin^2 \frac{\theta}{2}\right) q_j \tag{6.19}$$

Thus, the 16th-order Shapiro filter has a frequency response given by

$$\bar{q}_j = \left(1 - \sin^{16} \frac{\theta}{2}\right) \hat{q} e^{ij\theta}, \tag{6.20}$$

where each wavenumber amplitude coefficient $\hat{q}$ is modified by an amount $(1 - \sin^{16} \theta/2)$. Fig. 6.2 shows the damping characteristics of the 16th-order Shapiro filter as a function of wavenumber, in addition to the 2nd-, 4th- and 8th-order filters for comparison. We clearly see that the 2 grid-interval wave is damped completely after one application. Scales longer than 4 grid-lengths are almost untouched for the 16th-order filter.

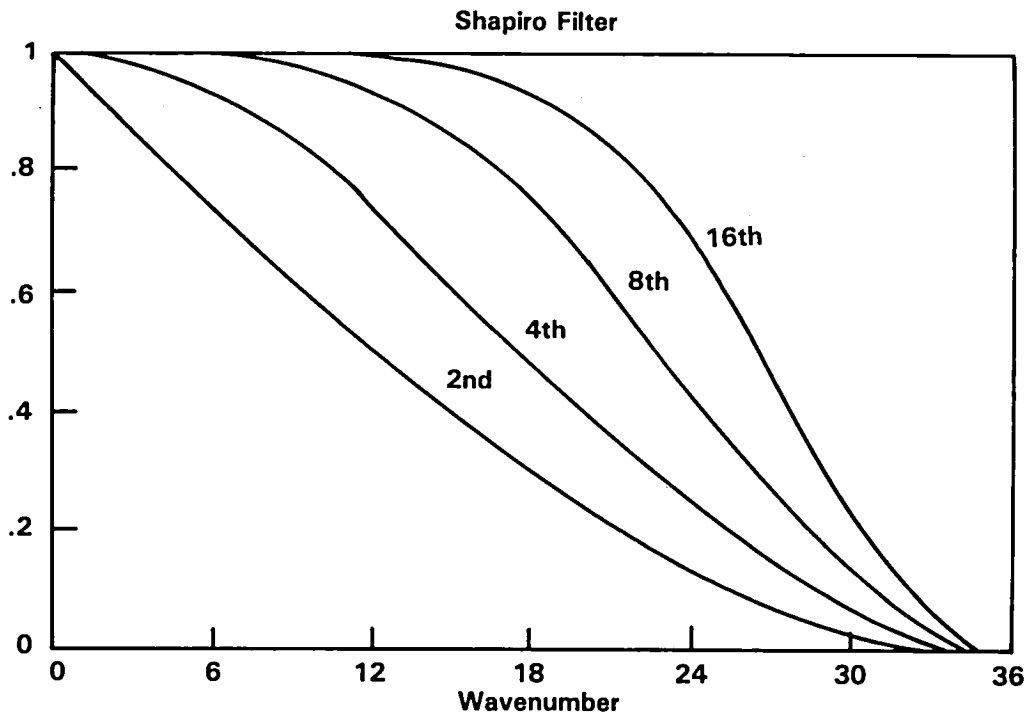


Fig. 6.2

Takacs and Balgovind (1983b) have shown that filtering of the winds in the meridional direction creates spurious sources of vorticity and divergence near the poles which ultimately affect the solution at all latitudes. Therefore, the 16th-order Shapiro filter is applied in practice to the height field in both directions and to the wind field in the zonal direction only.

VII. Normal modes

A number of experiments incorporating the normal modes of the GLA 4th-order shallow water model have been performed (see Navon *et al.* (1985), Takacs *et al.* (1985)). These works are based on the work of Bloom (1983) in which the normal modes of the GLA 4th-order General Circulation Model were computed.

The normal modes of the shallow water model have been incorporated into the shallow water system in two ways, a) Initialization, and b) High-Latitude Filtering.

a. Initialization

Following Daley (1980), based on the work of Machenhauer (1977), we can write the time-change of the normal mode projection coefficients C as

$$\frac{dC}{dt} = i\omega C + r, \tag{7.1}$$

where ω is the normal mode eigenfrequency. Here r represents then non-linear and forcing terms. Assuming r to be independent of time, integrating (7.1) yields.

$$C(t) = \left(C(t_o) + \frac{r}{i\omega}\right) e^{i\omega t} - \frac{r}{i\omega}. \tag{7.2}$$

The first term in (7.2) represents the oscillation term which we want to suppress for normal mode initialization. Clearly, if

$$C_o = -\frac{r}{i\omega}, \tag{7.3}$$

the oscillation term would vanish. In practice, the numerical method of iteration is used to calculate C_o. Letting

$$C_o' = -r/i\omega, \tag{7.4}$$

we see, after substituting into (7.1),

$$\frac{dC}{dt} = i\omega C - i\omega C_o' . \tag{7.5}$$

Solving for C_o' we find

$$C_o' = C_o - \frac{C_1 - C_o}{i\omega\Delta t} , \tag{7.6}$$

where a simple forward time-difference was used for dC/dt. Thus, the new normal mode coefficients at time t=0 are calculated by taking the time-change of the coefficients of the fast modes after one model integration step, and dividing by the normal mode eigenfrequency. This procedure is usually repeated until a critical number of iterations (4-10) have been performed, or some convergence criterion has been reached.

It has been experimentally found that a few particular modes and wavenumbers fail to converge when the Machenhauer technique is applied, particularly mode 43 for wavenumber 1. However, a more robust method of initialization, based on the work of Rasch (1984), is also available in the shallow water system. In this method, the assumption of the non-linear terms in (7.1) being constant in time is no longer required. Defining

$$F(C) \equiv \frac{dC}{dt} , \tag{7.7}$$

then F may be expanded in a Taylor series about t=0,

$$F(C(t)) \approx F(C_o) + (C - C_o) \left.\frac{dF}{dC}\right/_o , \tag{7.8}$$

or

$$C_o = C - \frac{F(C) - F(C_o)}{\left.\frac{dF}{dC}\right/_o} . \tag{7.9}$$

By definition, however, we want F(Co) = 0, i.e. the initial normal mode coefficients whose time tendency for the fast modes are zero. Thus, (7.9) may be expressed as

$$C_o = C_n - \frac{(C_{n+1} - C_n)/\Delta t}{(dF/dC)/_o} \tag{7.10}$$

where we have approximated F(C) by

$$F(C) \simeq \frac{C_{n+1} - C_n}{\Delta t} \tag{7.11}$$

Notice that from (7.1), $\frac{dF}{dC} = i\omega$ if r is independent of C(t). Thus, under this assumption (7.10) reduces to the Machenhauer initialization previously discussed. In practice, however, we let

$$\frac{dF}{dC}\Big/_o = \frac{F_n - F_o}{C_n - C_o} \tag{7.12}$$

where

$$F_n = \frac{C_{n+1} - C_n}{\Delta t} \tag{7.13}$$

and

$$F_o = \frac{C_n - C_o}{\Delta t} \tag{7.14}$$

This procedure is again repeated until a critical number of iterations or a convergence criterion has been reached.

The non-linear normal mode initialization techniques are summarized in Fig. 7.1. Here a plot is made of the time history of the LOG of the balance parameter BAL. BAL is defined to be the square of the time-tendencies of the normal mode projection coefficients, summed over all gravity modes and all wavenumbers. We clearly see that both the Machenhauer and Rasch techniques tend toward a balanced state during the first 5 iterations of the NMI scheme. After 5 iterations, however, the Machenhauer technique tends to diverge while the Rasch technique continues to reduce the balance parameter. It has been found that the failure by the Machenhauer technique is, as previously mentioned, primarily due to mode 43 for wavenumber 1, which results in a large divergent component in the polar wind field.

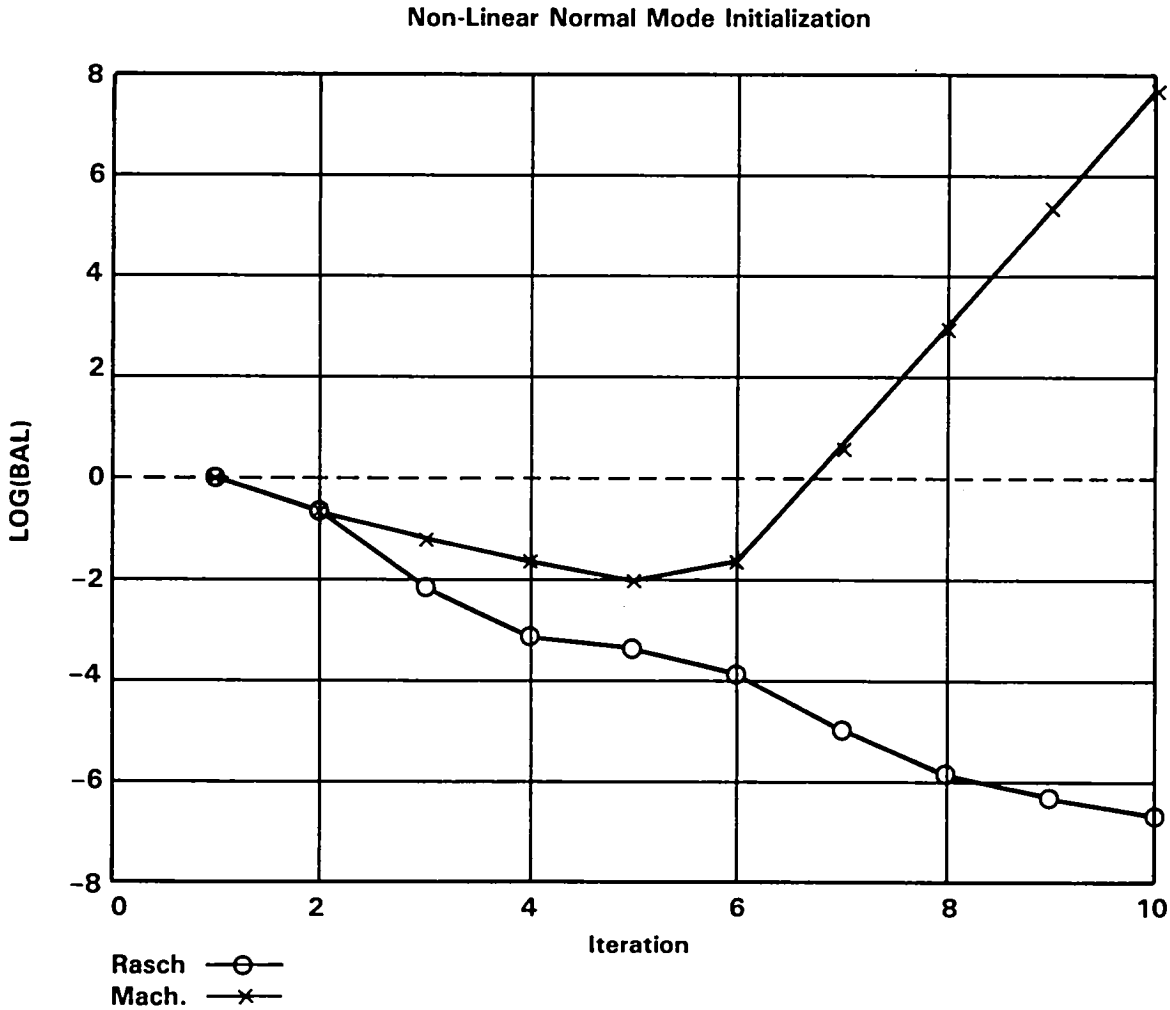


Fig. 7.1

b. Filter

Following the work of Navon *et al.* (1985) and Takacs *et al.* (1985), the model normal modes have been applied toward the problem associated with high-latitude filtering to control linear stability.

In an explicit leapfrog or Matsuno time-integration scheme the maximum allowable time-step is governed by the frequency of the fastest propogating mode. For a given gravity mode with frequency ωg, the maximum permissible time-step is given by

$$\Delta t_{max} = 1/|\omega g| \; . \qquad (7.15)$$

Thus, by ordering the normal modes in decreasing eigenfrequency, we may easily find all modes for which

$$|\omega g| > 1/\Delta t \tag{7.16}$$

holds true. Here, Δt is the time-step used in the model integration. These modes, which consist of a relatively small number of gravity modes, are denoted as 'fast modes', while the remaining gravity modes and all of the Rossby modes are denoted as 'slow modes'. Thus in order to prevent linear instability, at each time-step we simply forward transform the dependent variables $(u,v,h)^n$ given to us by the model thereby producing a normal mode coefficient set. To obtain the projection onto the fast modes, we reconstruct the fields by back-transforming only those mode coefficients whose eigenfrequency is greater than the maximum allowed. This produces Δu, Δv, and Δh which are then subtracted from the current fields to produce 'filtered' new values.

VIII. Tracer advection

Through the use of the logical NAMELIST parameter LTRACE, it is possible to perform experiments to observe the time evolution of passive tracer advection. The numerical advection scheme used in these studies is presented in Takacs (1985), and has the property of advecting localized disturbances (or profiles) with minimized dissipation and dispersion errors. The actual version of the scheme implimented here is a slight variation from Takacs (1985) in the sense that both zonal and meridional advections are calculated simultaneously.

The tracer advection scheme is derived from the conservation of the tracer having no source or sink terms. Letting q be defined as the mixing ratio of the tracer quantity, then

$$\frac{d}{dt}\int_{Vol(t)} \rho q dV = \int_{Vol(t)} \rho \frac{dq}{dt}\, dV = \int_{Vol(t)} \rho S dV - \int_s F \cdot \hat{n} ds, \tag{8.1}$$

where S is a source or sink of q per unit time, and F is the flux of q across the surface of the volume element. Equation (8.1) may be written as

$$\rho \frac{dq}{dt} = \rho S - \nabla \cdot F \tag{8.2}$$

and for no sources or sinks and no net flux across the boundary, we have

$$\frac{dq}{dt} = \frac{\partial q}{\partial t} + \mathbb{V} \cdot \nabla q + w\frac{\partial q}{\partial z} = 0 \tag{8.3}$$

For our shallow water system, we assume q = q(x,y,t), thus (8.3) becomes

$$\frac{\partial q}{\partial t} + \nabla \cdot (\mathbb{V} q) = q\nabla \cdot \mathbb{V}. \tag{8.4}$$

Using the shallow water continuity equation given by

$$\frac{\partial h}{\partial t} + \mathbb{V} \cdot \nabla h + h\nabla \cdot \mathbb{V} = 0, \tag{8.5}$$

we see that

$$\nabla \cdot \mathbb{V} = -\frac{1}{h}\left(\frac{\partial h}{\partial t} + \mathbb{V} \cdot \nabla h\right). \tag{8.6}$$

Using (8.6) in (8.4) and re-arranging, we have

$$\frac{\partial (hq)}{\partial t} + \nabla \cdot (h\mathbb{V} q) = 0. \tag{8.7}$$

Equation (8.7) may be written in component form on the sphere, given by

$$\frac{\partial (hq)}{\partial t} + \frac{1}{a\cos\phi}\left[\frac{\partial (hu)q}{\partial \lambda} + \frac{\partial}{\partial \phi}(hv\cos\phi)q\right] = 0. \tag{8.8}$$

Defining

$$U = hu$$

$$V = hv\cos(\phi)$$

$$\delta_x = a\cos(\phi)\,\delta\lambda$$

$$\delta y = a\delta\phi$$

equation (8.8) may be written as

$$\frac{\partial (hq)}{\partial t} + \frac{\partial (Uq)}{\partial x} + \frac{1}{\cos\phi}\cdot\frac{\partial (Vq)}{\partial y} = 0. \tag{8.9}$$

Following Takacs (1985), equation (8.9) is approximated in a predictor-corrector sequence as follows:

Defining

$$\delta_x(F) = F_{i+1/2,j} - F_{i-1/2,j}$$

$$\delta_y(G) = G_{i,j+1/2} - G_{i,j-1/2}$$

$$u_{i+1/2,j} = (U_{i+1,j} + U_{i,j})\Delta t/\Delta x$$

$$\bar{v}_{i,j+1/2} = (V_{i,j+1} + V_{i,j})\,\Delta t/\Delta y$$

Then:

PREDICTOR:

$$(hq)^*_{i,j} = (hq)^n_{i,j} - \left[\delta_x(F) + \frac{1}{\cos\phi_j}\delta_y(G)\right]$$

$$F_{i+1/2,j} = u^+_{i+1/2,j}\,q^n_{i,j} + u^-_{i+1/2,j}\,q^n_{i+1,j}$$

$$G_{i,j+1/2} = v^+_{i,j+1/2}\,q^n_{i,j} + v^-_{i,j+1/2}\,q^n_{i,j+1}$$

$$u^{\pm}_{i+1/2,j} = \left(\frac{\bar{u} \pm |\bar{u}|}{2}\right)_{i+1/2,j}$$

$$v^{\pm}_{i,j+1/2} = \left(\frac{\bar{v} \pm |\bar{v}|}{2}\right)_{i,j+1/2}$$

CORRECTOR:

$$q^{n+1}_{i,j} = q^n_{i,j} - \tfrac{1}{2}\left[\delta_x(P) + \frac{1}{\cos\phi_j}\delta_y(R)\right] + \left[\delta_x(\alpha Q) + \frac{1}{\cos\phi_j}\delta_y(\beta S)\right]$$

$$P_{i+1/2,j} = u^+_{i+1/2,j}(q^*_{i+1,j} + q^n_{i,j}) + u^-_{i+1/2,j}(q^*_{i,j} + q^n_{i+1,j})$$

$$R_{i,j+1/2} = v^+_{i,j+1/2}(q^*_{i,j+1} + q^n_{i,j}) + v^-_{i,j+1/2}(q^*_{i,j} + q^n_{i,j+1})$$

$$Q_{i+1/2,j} = u^+_{i+1/2,j}(q^*_{i+1,j} - q^n_{i,j} - q^*_{i,j} + q^n_{i-1,j}) + u^-_{i+1/2,j}(q^n_{i+2,j} - q^*_{i+1,j} - q^n_{i+1,j} + q^*_{i,j})$$

$$S_{i,j+1/2} = v^+_{i,j+1/2}(q^*_{i,j+1} - q^n_{i,j} - q^*_{i,j} + q^n_{i,j-1}) + v^-_{i,j+1/2}(q^n_{i,j+2} - q^*_{i,j+1} - q^n_{i,j+1} + q^*_{i,j})$$

$$\alpha_{i+1/2,j} = \left(\frac{1+|u|}{6}\right)_{i+1/2,j}$$

$$\beta_{i,j+1/2} = \left(\frac{1+|v|}{6}\right)_{i,j+1/2}$$

IX. User's guide

The GLA Fourth-Order Shallow Water Model is run through use of a series of CMS EXEC's kept on the F400M disk 200 machine. In order for the user to run the shallow water model, the user must be linked to a set of particular disks which contain model code, text and execs. Thus, the user must execute the following:

```
LINKTO F400M 200 591 M
LINKTO F400M 192 592 C
LINKTO F400M 207 593 F
590 3
```

As previously mentioned, the F400M 200 disk contains the model code and CMS EXEC's which run the model. The F400M 192 and 207 disks are library disks for the GLA modelling group which contain many useful execs and processing routines which are called upon from the shallow water utilities. The command 590 3 invokes 3 cylinders of TDISK at address 590 with filemode I. This is done so that all processing and output can be temporarily stored on TDISK and thereby not interfere with the user's own A-disk. Should the user find that more than 3 cylinders are needed, this number can be increased.

The shallow water exec system contains the following execs: SWEIC, SWERUN, SWEPOST, SWEPLOT, SWEBATCH, SWEMODE, SWEDIFF, SWEMVS. To obtain information on any of the execs, the user need only type the exec name followed by a question mark, eg. SWERUN ?A brief description of each exec and the namelist data set associated with each program is now presented.

SWEIC

This program creates initial conditions for the GLA Fourth-Order 2-Level Shallow Water Model. The number of levels and the type of Initial Conditions desired are NAMELIST parameters. Output from this EXEC is the initial condition file SWEIC OUTPUT I.

Hit return to continue, or X to quit.

```
* ********************************************************************
* *  M / A - C O M  S I G M A  D A T A  I N C .  N A S A  -  G S F C  *
* ********************************************************************
*
*      THIS PROGRAM CREATES INITIAL CONDITIONS
*           FOR THE SHALLOW WATER MODEL
*
* ********************************************************************
*
*  INPUT :
*
*  KMAX   :  Number of levels (1 OR 2) in model
*  SETUP  :  Choice of Initial Conditions
*  TPG    :  FLAG (1 OR 0) for realistic topography
*  R      :  Wavenumber to be used for ROSSBY-HAURWITZ wave
*
*  LEVEL  :  Manditory pressure level(s) to be used
*            for realistic GCM Initial Conditions:
*
*            Level =  1    50 mb        Level =  7   300 mb
*            Level =  2    70 mb        Level =  8   400 mb
*            Level =  3   100 mb        Level =  9   500 mb
*            Level =  4   150 mb        Level = 10   700 mb
*            Level =  5   200 mb        Level = 11   850 mb
*            Level =  6   250 mb        Level = 12  1000 mb
*
* NOTE:  SETUP = 'RSBY' for ROSSBY-HAURWITZ wave initial conditions
*              = 'GMCD' for UNINITIALIZED GCM DATA
*              = 'WINT' for JAN   9, 1979 UNINITIALIZED DATA
*              = 'WNMI' for JAN   9, 1979   INITIALIZED DATA
*              = 'SUMM' for JUNE 11, 1979 UNINITIALIZED DATA
*              = 'SNMI' for JUNE 11, 1979   INITIALIZED DATA
*

 &LEVELS
 KMAX=1
 &END

 &ICON
 SETUP='WINT', LEVEL=7,9
 &END

 &WAVENO
 R=3, TPG=0
 &END
```

SWERUN

This program runs the GLA Fourth-Order 2-Level Shallow Water Model. Experiment parameters such as the timestep, length of integration, frequency of output, etc. are NAMELIST parameters. The model reads initial conditions from the file SWEIC OUTPUT I. Output from the model is stored in the file VARS OUTPUT I.

Hit return to continue, or X to quit.

```
* ************************************************************************
* *  M / A - C O M  S I G M A  D A T A  I N C .  N A S A  -  G S F C  *
* ************************************************************************
*
*               THIS PROGRAM RUNS THE GLA FOURTH-ORDER
*                    2-LEVEL SHALLOW WATER MODEL
*
* ************************************************************************
*
* INPUT :
*
* SCHEME:  Time-scheme used in model,  'MATS' for MATSUNO
*                                      'LEAP' for LEAPFROG
* NLEAP :  Number of consecutive LEAPFROGS to do
*          before restarting with a MATSUNO step
* TSTART:  Hour from INITIAL CONDITION
*          at which to start integration
* TMAX  :  Hour at which to stop integration
* DT    :  Time-step in seconds
* TPRT  :  Frequency (IN HOURS) at which to print output
* KAPPA :  Ratio of densities between UPPER and LOWER level
* KMAX  :  Number of levels used in model (1 or 2)
*
* GFILT :  Global Shapiro filter flag                  (T or F)
* HFILT :  High-latitude filter flag                   (T or F)
* LNMF  :  High-latitude normal mode filter flag       (T or F)
* LNMI  :  Non-linear normal mode initialization flag (T or F)
* LRASCH:  True for RASCH'S NMI iteration, False for MACHENHAUR
* NMIMAX:  Number of iterations for normal mode initialization
* TSHAP :  Frequency (IN HOURS) at which to apply global shapiro filter
*
* LTRACE:  Logical for advection of tracer
* LPRIME:  Logical for experiments to remove initial time tendencies
* LFORCE:  Logical for experiments to call FORCING terms
* ALPHA :  Parameter to enhance or reduce FORCING ( alpha X FORCING )
*
* LFRICT:  Logical for frictional and zonal forcing
* TAO   :  Dissipation time scale (DAYS) for friction and zonal forcing
*
* XLAB  :  Label associated with current experiment
```

```
&TSCHEM
SCHEME = 'MATS', NLEAP=20
&END

&INPUT
TSTART = 000.000, TMAX=000.50, DT=0450.0, TPRT=000.60
&END

&LEVELS
KAPPA=1.00, KMAX=1
&END

&FILT
GFILT=.T., TSHAP=2.,
HFILT=.T., LNMF =.F.,
LNMI =.T., LRASCH=.T., NMIMAX=10,
&END

&STRESS
LFRICT=.F., TAO=7.,
&END

&FORCE
LPRIME=.F., LFORCE=.F., ALPHA=50.0,
LTRACE=.F.,
&END

&LABEL
XLAB = 'RASCH TE','CHNIQUE ','WITH NEW',' BALANCE',
       ' PARAMET','ER      ','        ','        ',
&END
```

SWEPOST

This program post-processes data from an SWERUN or SWEBATCH run. During the execution of this EXEC, the input data is plotted and/or displayed in various user-defined formats. Diagnostic quantities such as vorticity, divergence, streamfunction, as well as zonal and eddy kinetic energies, are also available. All options are NAMELIST parameters. The input data set defaults to VARS OUTPUT I, but may be any stored output data set of the VARS OUTPUT format.

Hit return to continue, or X to quit.

```
* ************************************************************************
* *  M / A - C O M  S I G M A  D A T A  I N C .   N A S A  -  G S F C  *
* ************************************************************************
*
*               THIS PROGRAM POST-PROCESSES DATA FROM THE
*                     2-LEVEL SHALLOW WATER MODEL
*
* ************************************************************************
*
* INPUT :
*
*  ALL:     Logical to read ALL data on history file
*  TIMIN:   Beginning time for post-processing if ALL = False
*  TIMOUT:  Ending    time for post-processing if ALL = False
*
*  KU:      FLAG for u wind component
*  KV:      FLAG for v wind component
*  KH:      FLAG for height field
*  KQ:      FLAG for tracer
*  KVOR:    FLAG for vorticity
*  KDIV:    FLAG for mass divergence
*  KSTR:    FLAG for streamfunction
*  KCHI:    FLAG for velocity potential
*  KMDIV:   FLAG for mass divergence
*  KPVOR:   FLAG for potential vorticity
*
*  GRF:   FLAG for GRAFICS ROUTINE on CMS
*  DSP:   FLAG for NUMERICAL DISPLAY of array
*  PHY:   FLAG for ENERGY and MOMENTUM analysis
*  ENS:   FLAG for TIME SERIES of TOTAL POTENTIAL ENSTROPHY
*                              and TOTAL MASS
*
*  KFFT:  Fourier decomposition of specified fields
*
*    KPRS:   FLAG to plot history record of sea-level pressure (heights)
*   NGRID:   The total number of points used for KPRS
*   IGRID:   The i-index of points used for KPRS
*   JGRID:   The j-index of points used for KPRS
*  SPLINE:   FLAG for using SPLINE for KPRS plot
*
```

```
*
*   SHAP:  FLAG to Shapiro filter data before plotting
*  NSHAP:  1/2 the Order of the Shapiro filter (2,4,6, or 8)
*  DSHAP:  The number of dimensions to filter, 1: zonal only, 2: both
*
*  NUM:  Offset in GRAFICS ROUTINE for UPPER and LOWER LIMITS (NUM=0,8)
*  NX:   Number of interpolated points in x-direction for GRAFICS
*  NY:   Number of interpolated points in y-direction for GRAFICS
*        Note:  Usually  NX=3, NY=2  for COARSE GRID
*                        NX=2, NY=2  for FINE   GRID
*
*  NSCALE:  FREQUENCY of scaling (NSCALE TIMES)
*
 &PARMS

  ALL    = F,
  TIMIN  = 000.,
  TIMOUT = 012.,

  GRF    = F,
  DSP    = F,
  PHY    = F,
  ENS    = F,
  KFFT   = T,

  KQ     = F,
  KU     = F,
  KV     = F,
  KH     = T,
  KVOR   = F,
  KDIV   = F,
  KSTR   = T,
  KCHI   = T,
  KMDIV  = F,
  KPVOR  = F,

  SHAP = F, NSHAP = 2, DSHAP = 2,

  NX=2, NY=2, NUM=0, NSCALE=3

 &END
 &GRIDS
  KPRS  = F, SPLINE = T,
  NGRID = 8,
  IGRID = 01, 10, 20, 30, 40, 50, 60, 72,
  JGRID = 01, 05, 10, 15, 20, 30, 40, 46,
 &END
```

SWEPLOT

This program post-processes data from an SWERUN or SWEBATCH run. During the execution of this EXEC, the input data is plotted using the WOLFPLOT plotting package. With this package, wind vectors and streamlines as well as diagnostic quantities such as vorticity, divergence, etc. are available. All options are NAMELIST parameters. This plots may be run inter-actively or under the BATCH facility. A NON-LABEL tape is required for plots using BATCH.

Hit return to continue, or X to quit.

Type B for BATCH job or

C for CMS

Type in the VMID and the NODES (max=3) of the MVS data set.

eg.)W3LLT NODE1 NODE2 NODE3

Enter the NONLABEL SCRATCH TAPE #

Enter TIME in minutes. Default is 5 Minutes.

Enter FILENAME for batch namelist. Default is SWEBATCH.

```
*************************************************************************
*   M / A - C O M   S I G M A   D A T A   I N C .   N A S A  -  G S F C  *
*************************************************************************
*
*                  THIS PROGRAM PLOTS DATA FROM THE
*                     2-LEVEL SHALLOW WATER MODEL
*
* ***********************************************************************
*
* INPUT :
*
*  LTPG:    FLAG for topography                DTPG:  Contour interval
*
*  LU:      FLAG for u wind component          DU:    Contour interval
*  LV:      FLAG for v wind component          DV:    Contour interval
*  LH:      FLAG for height field              DH:    Contour interval
*  LQ:      FLAG for tracer
*  LVOR:    FLAG for vorticity                 DVOR:  Contour interval
*  LDIV:    FLAG for mass divergence           DDIV:  Contour interval
*  LSTR:    FLAG for streamfunction            DSTR:  Contour interval
*  LCHI:    FLAG for velocity potential        DCHI:  Contour interval
*  LHDIV:   FLAG for mass divergence           DHDIV: Contour interval
*  LPVOR:   FLAG for potential vorticity       DPVOR: Contour interval
*
*  Note:    Contour interval = 0 implies computer generated interval
*
*  LSTRLN: FLAG for streamlines
*  LVEC:   FLAG for wind vectors
*  VREF:   Wind velocity whose magnitude is represented by 1 grid length
*
*  ALL:     FLAG to read and process ALL data in history file
*  TIMIN :  Time (in hours) to begin processing data if ALL = .F.
*  TIMOUT:  Time (in hours) to end   processing data if ALL = .F.
*  NSTART:  YYMMDD at which to begin for labeling purposes
*
*  LAT1 :   Southern latitudinal  boundary
*  LAT2 :   Northern latitudinal  boundary
*  LONG1:   Western  longitudinal boundary
*  LONG2:   Eastern  longitudinal boundary
*
*  LPRIME:  FLAG to plot differences between history file
*           and the initial condition
*
```

```
 &INPUT

  LPRIME = F,

  LTPG   = F, DTPG  = 200.,

  LVEC   = T, VREF  = 30.0,
  LSTRLN = F,
  LU     = F, DU    =   0.0,
  LV     = F, DV    =   0.0,
  LH     = T, DH    = 060.0,
  LQ     = F, DQ    =   5.0,
  LVOR   = F, DVOR  =   0.0,
  LDIV   = F, DDIV  =   0.0,
  LSTR   = F, DSTR  =   0.0,
  LCHI   = F, DCHI  =   0.0,
  LHDIV  = F, DHDIV =   0.0,
  LPVOR  = F, DPVOR =   0.0,

    ALL  = T,
  TIMIN  = 000.,
  TIMOUT = 000.,
  NSTART = 0,

  LAT1  = -090, LAT2  =  90,
  LONG1 = -180, LONG2 = 180,

 &END

  CreatingSWEPLOT BATCH I

/JOB W3LLT SWEPLOT
/QUERY PRINT=*
/IDENT SWEPLT
/ERROR PRINT=*
/SET RATE   =  BASIC
/SET DUMP   = NO
/SET NOTIFY = YES
/SET TIME   = 10
/SET CARDS  = 100000
/SET LINES  = 100000
/SET SIZE   = 2
/SET TAPE   = 1
// SETUP TAPE = 123456 NL RI 181
LINKTO F400M  200 254 M
EXEC GETTAP 123456 NL RI 181
EXEC SWEPLOTS 123456 PLOT1 TSOUT2 W3LLT TRACER DATA
/*


  Do you wish to SUBMIT?
```

SWEBATCH

This program runs the GLA Fourth-Order 2-Level Shallow Water Model under a BATCH environment. Input NAMELISTS for SWEIC and SWERUN are created and stored as SWEIC BATCH# and SWE BATCH# files. The BATCH # is USER defined, with a default of BATCH. The data history file, VARS OUTPUT, will be sent back to the USER's reader upon completion.

Hit return to continue, or X to quit.

```
Enter JOB #.  Default is BLANK.
Enter STORAGE SIZE #.  Default is 2 Meg.
Enter TIME in minutes.  Default is 2 Minutes.
Do you wish to compile a new version on BATCH?

Creating SWEBATCH JOB I
```

```
/JOB W3LLT SWEJOB
/QUERY PRINT=*
/IDENT SWE
/ERROR PRINT=*
/SET RATE   =  PRIORITY
/SET DUMP   = NO
/SET NOTIFY = YES
/SET TIME   = 2
/SET CARDS  = 100000
/SET SIZE   = 2
CP LINK W3LLT 191 237 RR
ACC 237 B/A
EXEC SWEBM BATCH
/*
```

```
  Do you wish toSUBMIT?
```

SWEMODE

This program analyzes data from an SWERUN or SWEBATCH run. During the execution of this EXEC, the input data is projected onto specific user-defined wavenumbers and normal modes, and then reconstructs the u, v and h field using only those wavenumbers and modes specified.The default input data set is VARS OUTPUT I, although the user can input any data set of the VARS OUTPUT format currently stored. The output data set is called MODE DATA I. Also, the normal mode amplitudes of the output data set as a function of wavenumber is produced in the file AMPL OUTPUT I.

Hit return to continue, or X to quit.

```
* ************************************************************************
* *  M / A - C O M  S I G M A  D A T A  I N C .  N A S A  -  G S F C  *
* ************************************************************************
*                                                                        *
*           This program PROJECTS data U,V and H onto the                *
*           Shallow Water Model NORMAL MODES, and then                   *
*           reconstructs the data using ONLY user-defined                *
*           WAVENUMBERS (00-36) and MODES (01-66,67,68).                 *
*                                                                        *
*           Note:  For ALL wavenumbers and/or ALL modes,                 *
*           set logicals ALLWAV and/or ALLMOD to True.                   *
*           Otherwise, set to False and PICK individual                  *
*           wavenumbers and modes.                                       *
*                                                                        *
* ************************************************************************
*
 &INPUT
  ALLWAV = .F.,
  ALLMOD = .F.,
  NWAVE  = 1,2,3,4,
  NMODE  = 45,46,47,52,
  HEADER = 'NO FILTE','R - TEND','ENCY FIL','TER     ',
           '       ','  ','      ','  ','      ','  '
 &END

  Type in FILENAME FILETYPE FILEMODE of the data to be processed.
  Default is VARS OUTPUT I, or ...
  Type 7 if the data is on the Amdahl V7.
```

SWEDIFF

This program differences two stored data sets, and produces the differenced u, v and h fields in the file VARS OUTPUT I. The input data sets may by stored data on CMS or MVS. In addition to the VARS OUTPUT file, the STATS between the two input fields (i.e., RMS, correlation coefficient, etc.) are created in the file STATS OUTPUT I for the globe and various regional areas. One region is USER defined, given as a NAMELIST input data file called SWEDIFF DATA A.

Hit return to continue, or X to quit.

```
Type inFILENAME FILETYPE FILEMODE of the first data set to read,
or ...
Type7
followed by theVMIDand theNODES(max=3) of theMVSdata set
if the data is on the Amdahl V7.
eg.)7 W3ABC NODE1 NODE2 NODE3


Type inFILENAME FILETYPE FILEMODE of the second data set to read,
or ...
Type7
followed by theVMIDand theNODES(max=3) of theMVSdata set
if the data is on the Amdahl V7.
eg.)7 W3ABC NODE1 NODE2 NODE3
```

SWEMVS

This program stores data from an SWERUN or SWEBATCH run onto Mass Storage. During the execution of this EXEC, the data is temporarily converted to a formatted LRECL 80 data set called VARS1 OUTPUT I, and is then sent to Mass Storage via MVS. The default name of the input data set is VARS OUTPUT I, although the user can input any data set of the VARS OUTPUT format currently stored. The name of the stored data set on MVS is of the form:

```
VMID node1 node2 node3
eg) W3ABC SWE TEST DATA
```

Hit return to continue, or X to quit.

X. References

Arakawa, A., 1972: Design of the UCLA general circulation model. Tech. Rept. No. 7, Dept. of Meteorology, UCLA, 116 pp. *Numerical Simulation of Weather and Climate.*

Daley, R., 1980: The development of efficient time integration schemes uing model normal modes. *Mon. Wea. Rev., 108, 100-110.*

Kalnay, E., R.C. Balgovind, W. Chao, J. Edelmann, J. Pfaendtner, L. Takacs, and K. Takano, 1983: Documentation of the GLAS Fourth-Order General Circulation Model. NASA Technical Memorandum 86064, Greenbelt, MD 20771.

Machenhauer, B., 1977: On the dynamics of gravity oscillations in a shallow water model, with applications to normal mode initialization. *Contrib. Atmos. Phys., 50, 253-271.*

Navon, I.M., S. Bloom, and L.L. Takacs, 1985: Computational modes and the Machenhauer N.L.N.M.I. of the GLAS 4th Order Model. Seventh Conference on Numerical Weather Prediction, American Meteorological Society.

Rasch, P.J., 1984: Extending normal mode initialization: A new method. Ph.D. Thesis. Cooperative Thesis No. 81. Florida State University and NCAR, NCAR/CT-81, PB84-196435, 139 pp.

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.

Takacs, L.L., 1985: A two-step scheme for the advection equation with minimized dissipation and dispersion errors. *Mon. Wea. Rev., 113,* 1050-1065.

Takacs, L.L., and R.C. Balgovind, 1983a: High latitude filtering in global grid point models. *Mon. Wea. Rev., 111,* 2005-2015.

Takacs, L.L., and R.C. Balgovind, 1983b: On the effect of using the Shapiro filter to smooth winds on a sphere. NASA Technical Memorandum 86053, pp. 105–113, Greenbelt, MD 20771.

Takacs, L.L., Kalnay, E., and I.M. Navon, 1985: High latitude filtering in global grid-point models using model normal modes. Seventh Conference on Numerical Weather Prediction, American Meteorological Society.

XI. Appendix A

a. Sample output

The following is a brief sample of the type of initial conditions available, the realistic topography data set used, and a few sample runs.

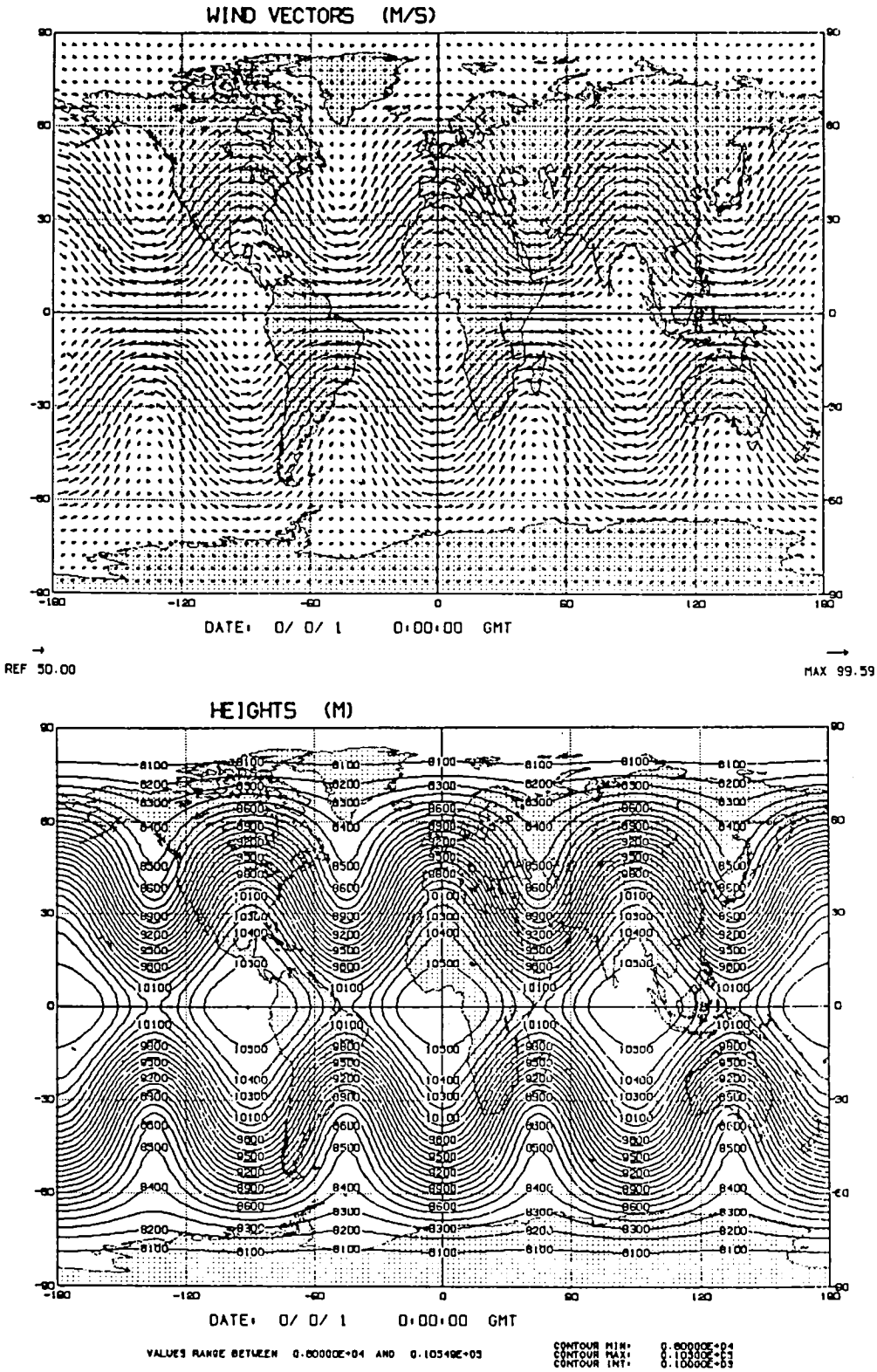


Fig. 1. Rossby-Haurwitz wave. Background map for perspective only.

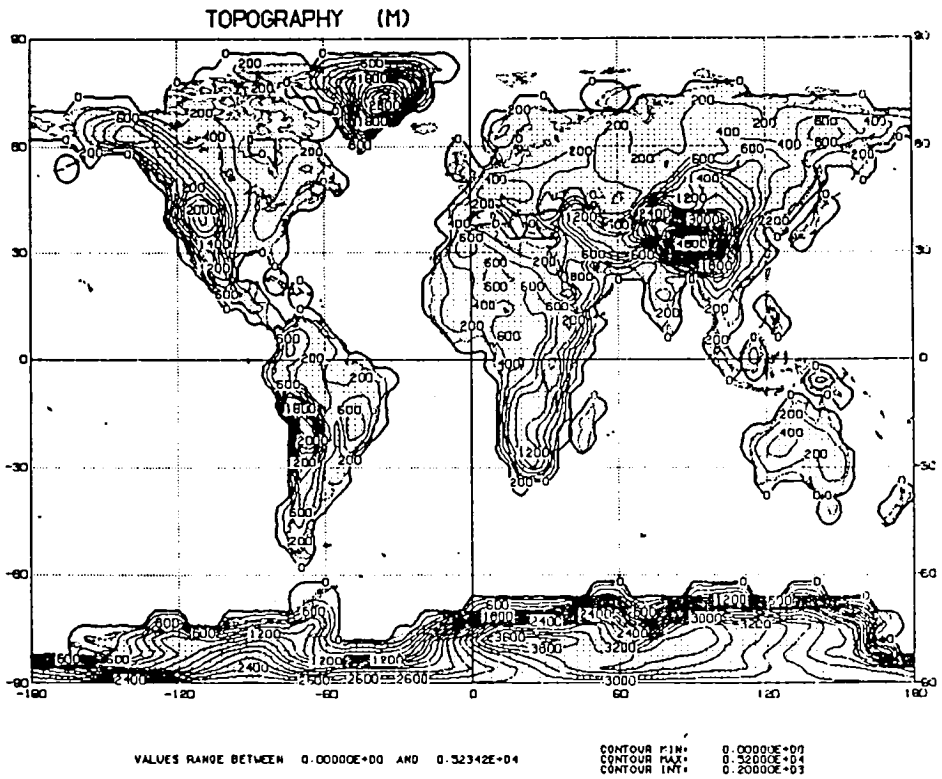


Figure 2. Topography used in conjunction with realistic initial conditions

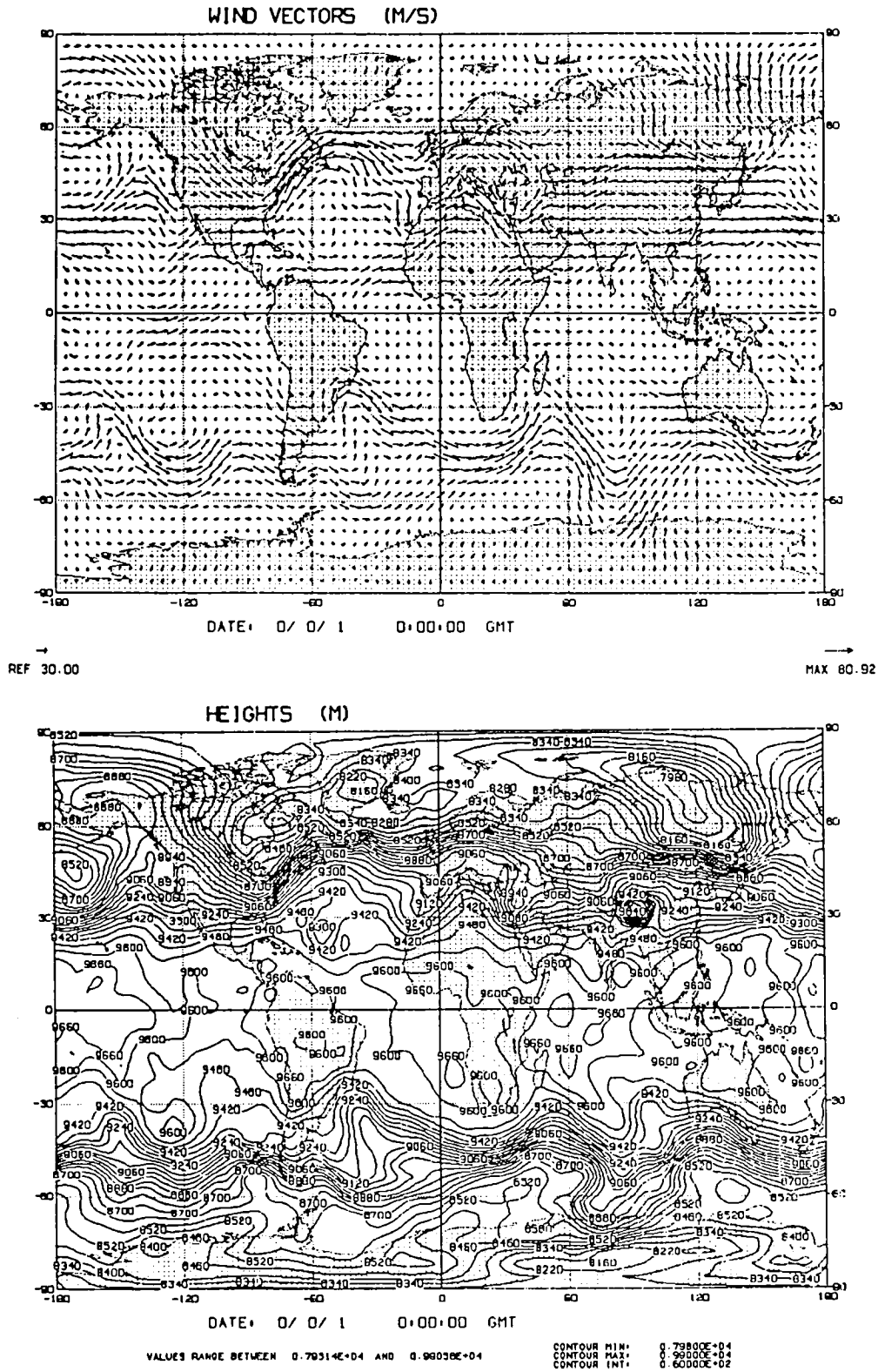


Fig. 3. Uninitialized 300 mb Northern Hemisphere winter conditions

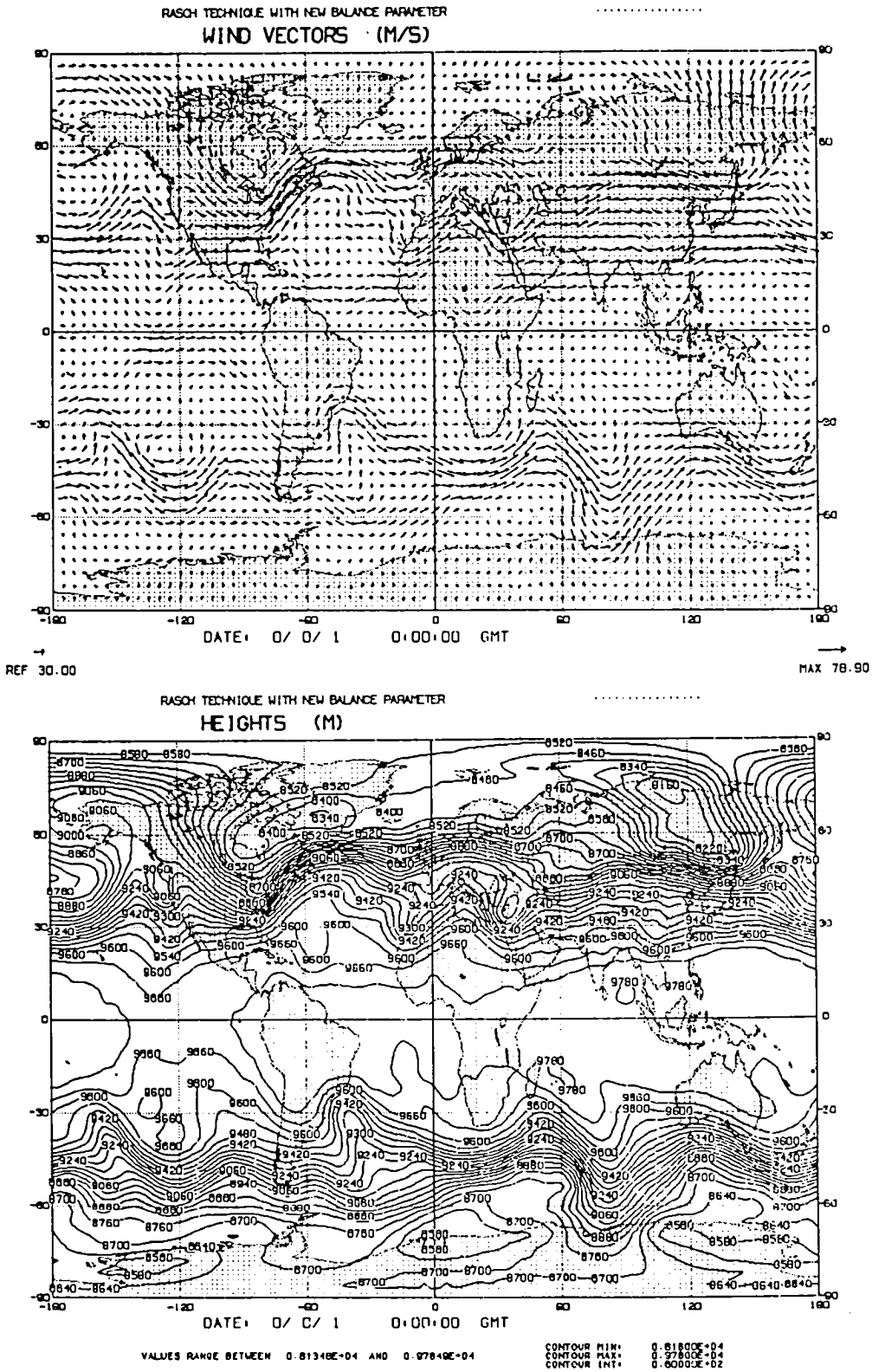


Fig. 4. Normal mode initialized 300 mb Northern Hemisphere winter conditions

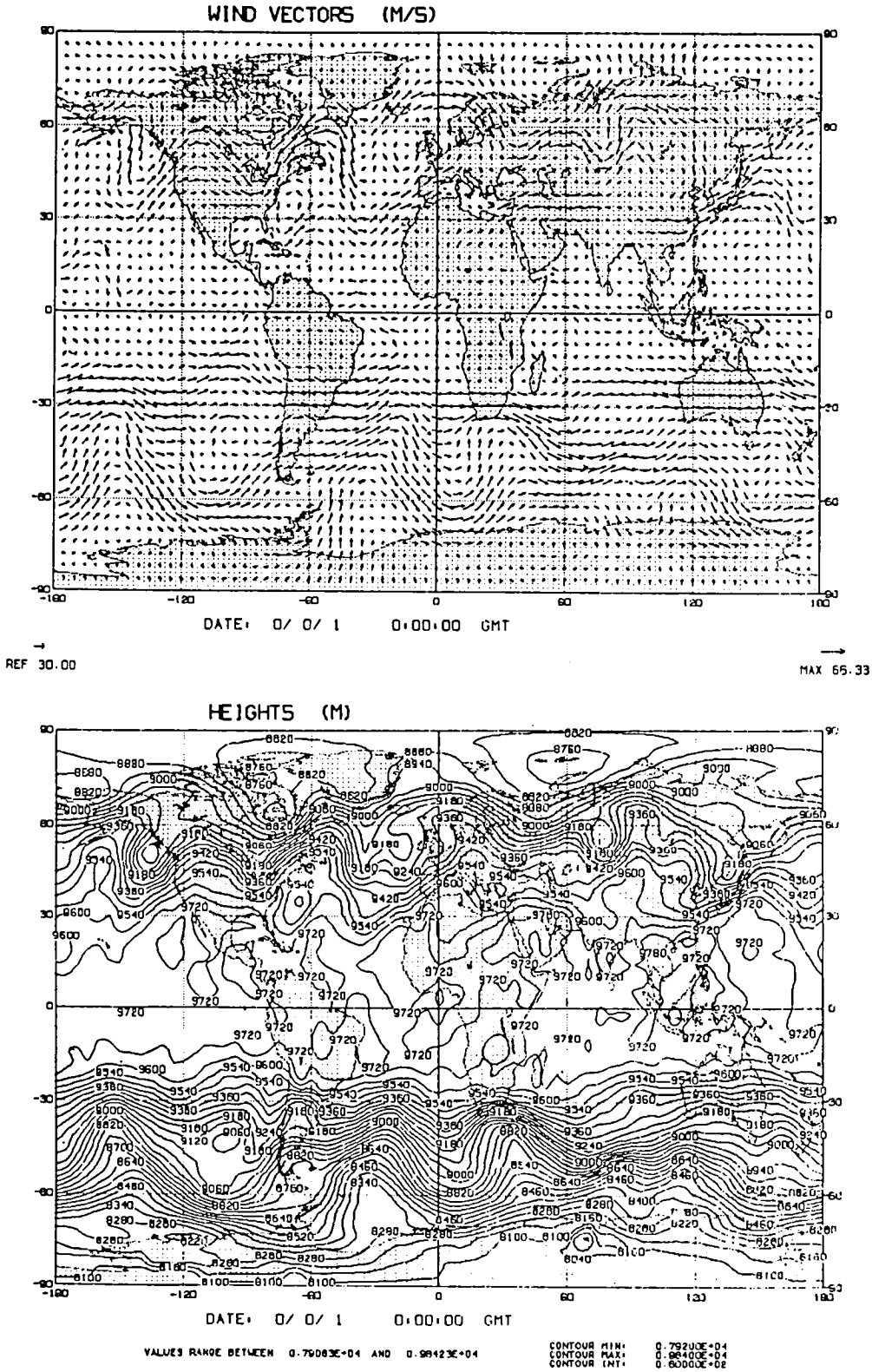


Fig. 5. Uninitialized 300 mb Northen Hemisphere summer conditions

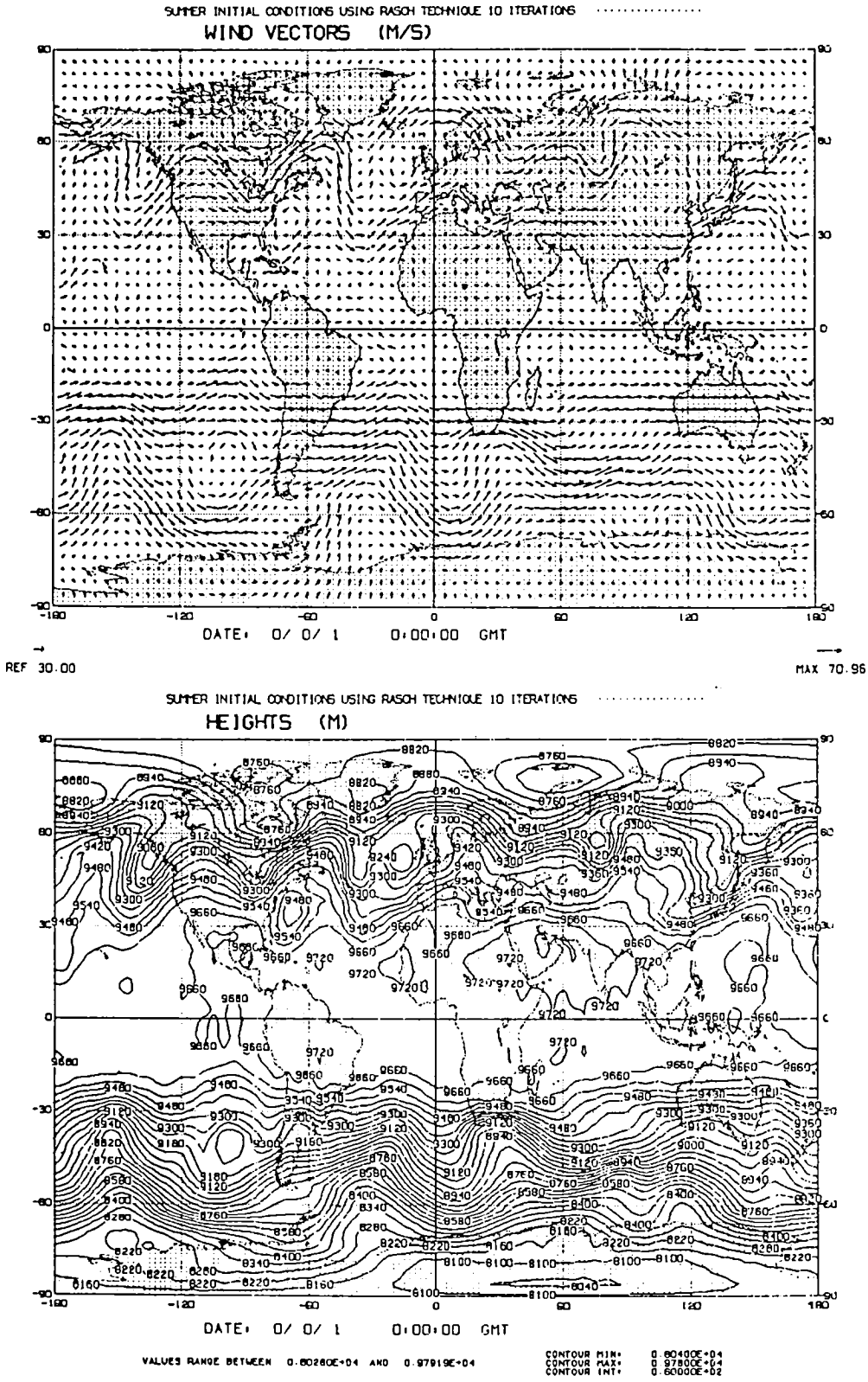


Fig. 6. Normal mode initialized 300 mb Northern Hemisphere summer conditions

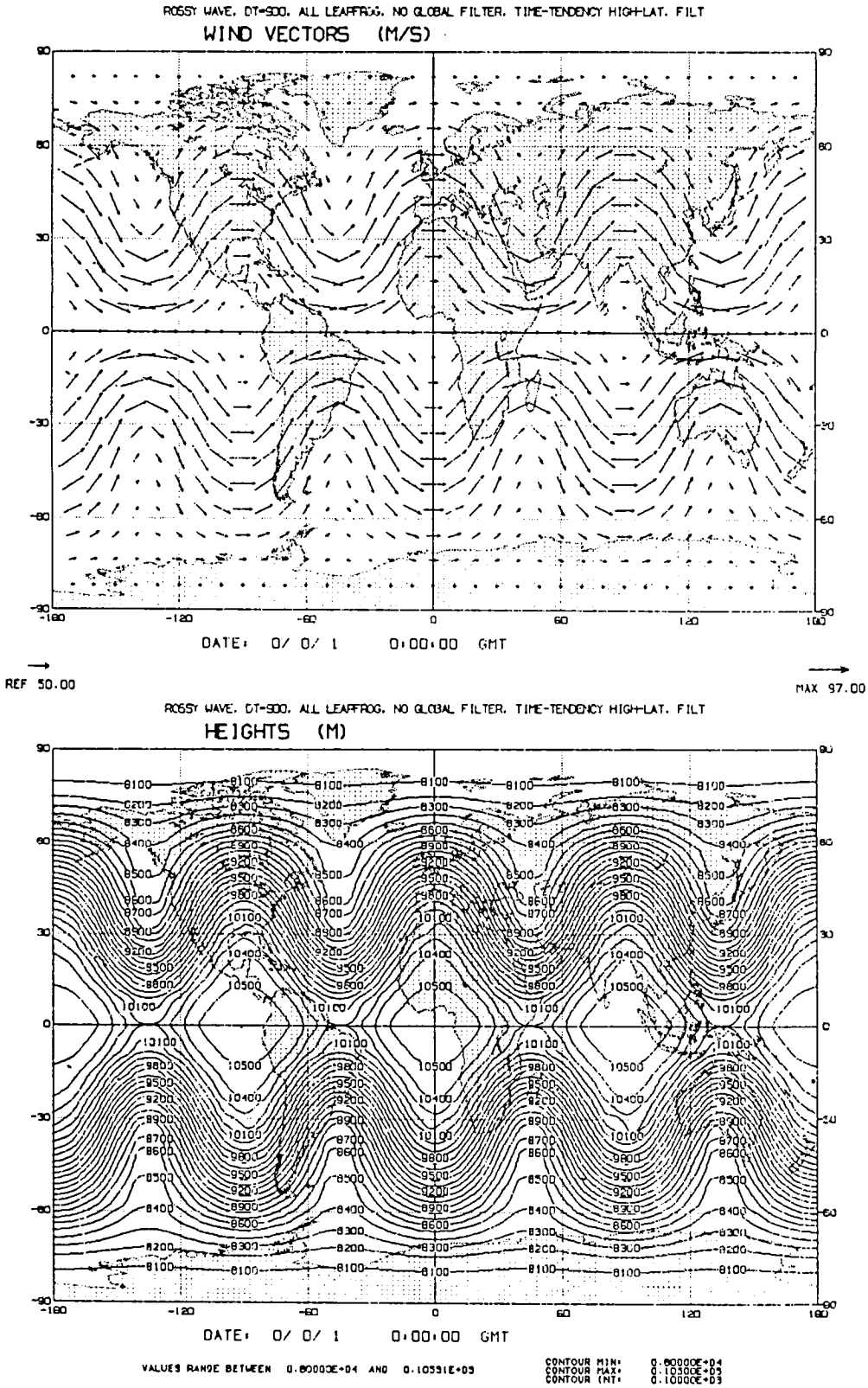


Fig. 7. Rossby-Haurwitz wave initial condition using 36×23 resolution

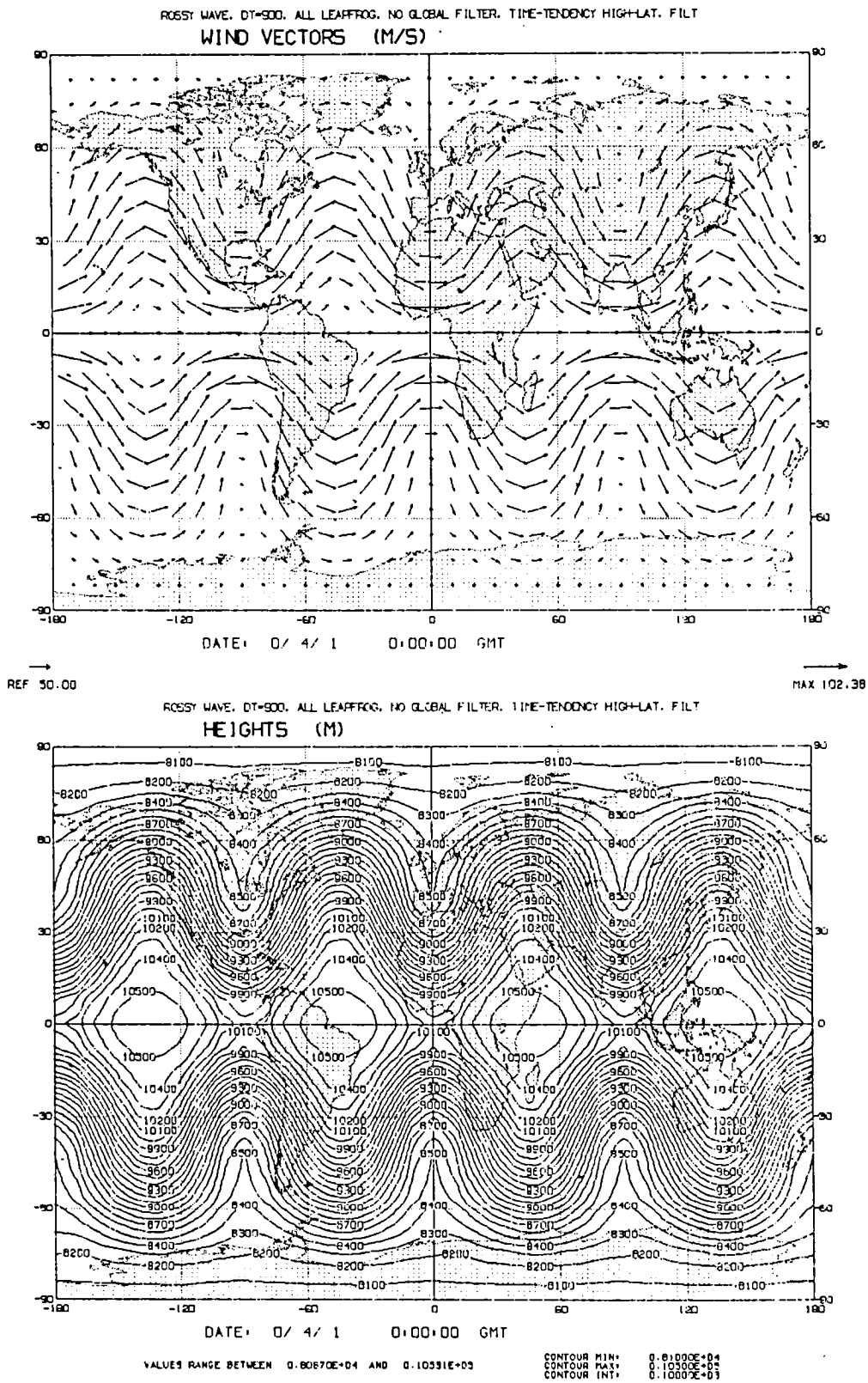


Fig. 8. Rossby-Haurwitz wave after 4 days, using dt=900 seconds

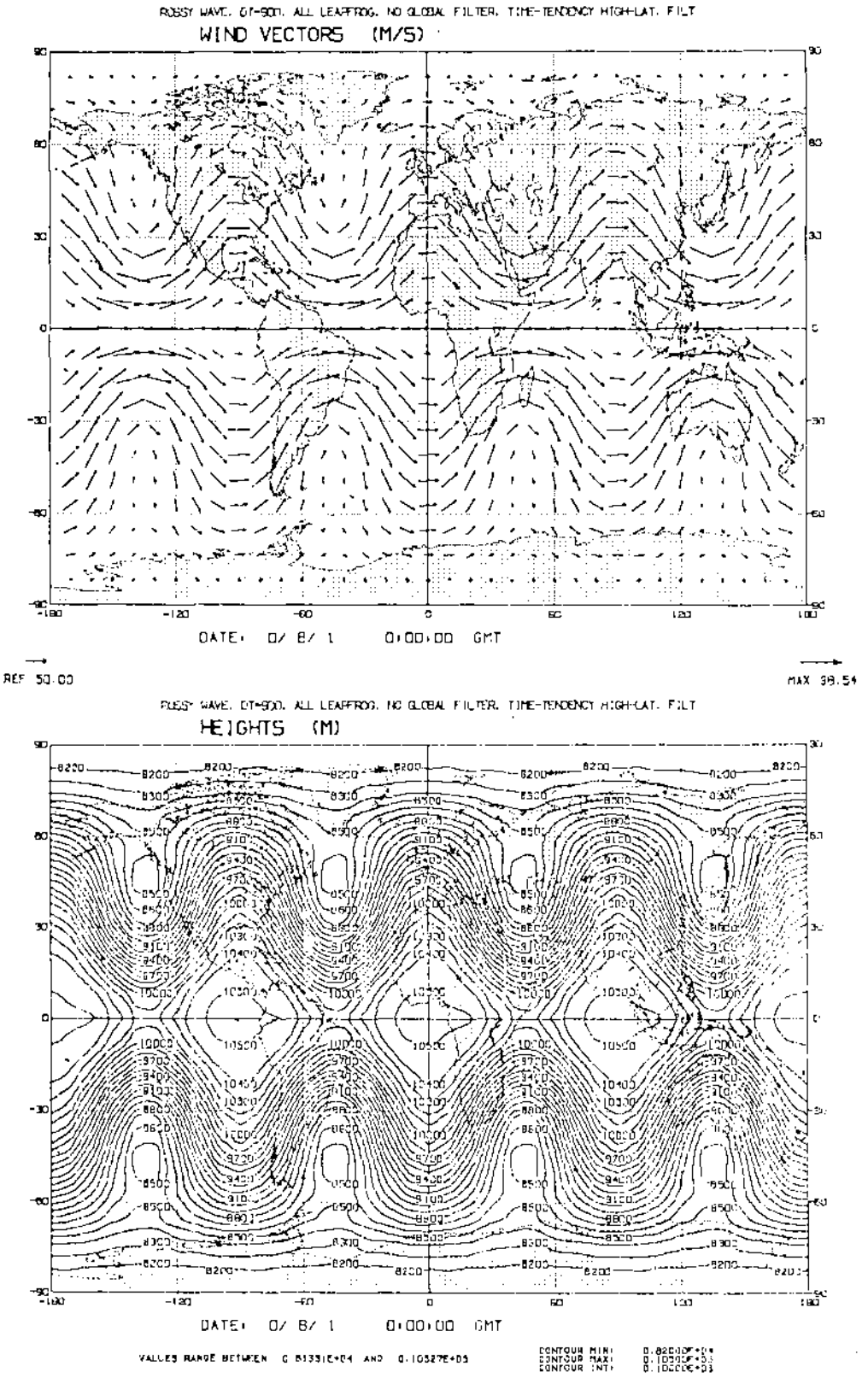


Fig. 9. Rossby-Haurwitz wave after 8 days

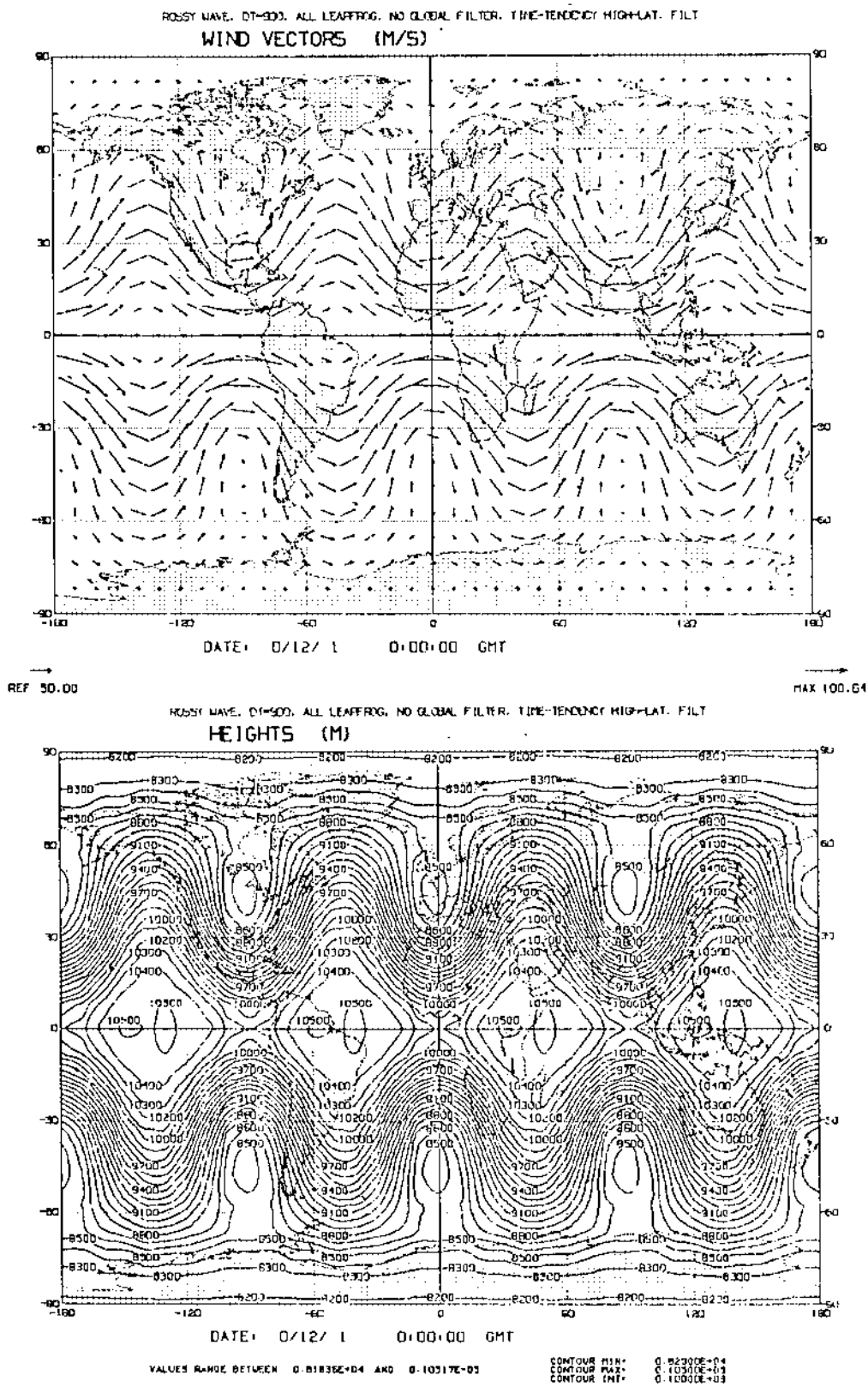


Fig. 10. Rossby-Haurwitz wave after 12 days

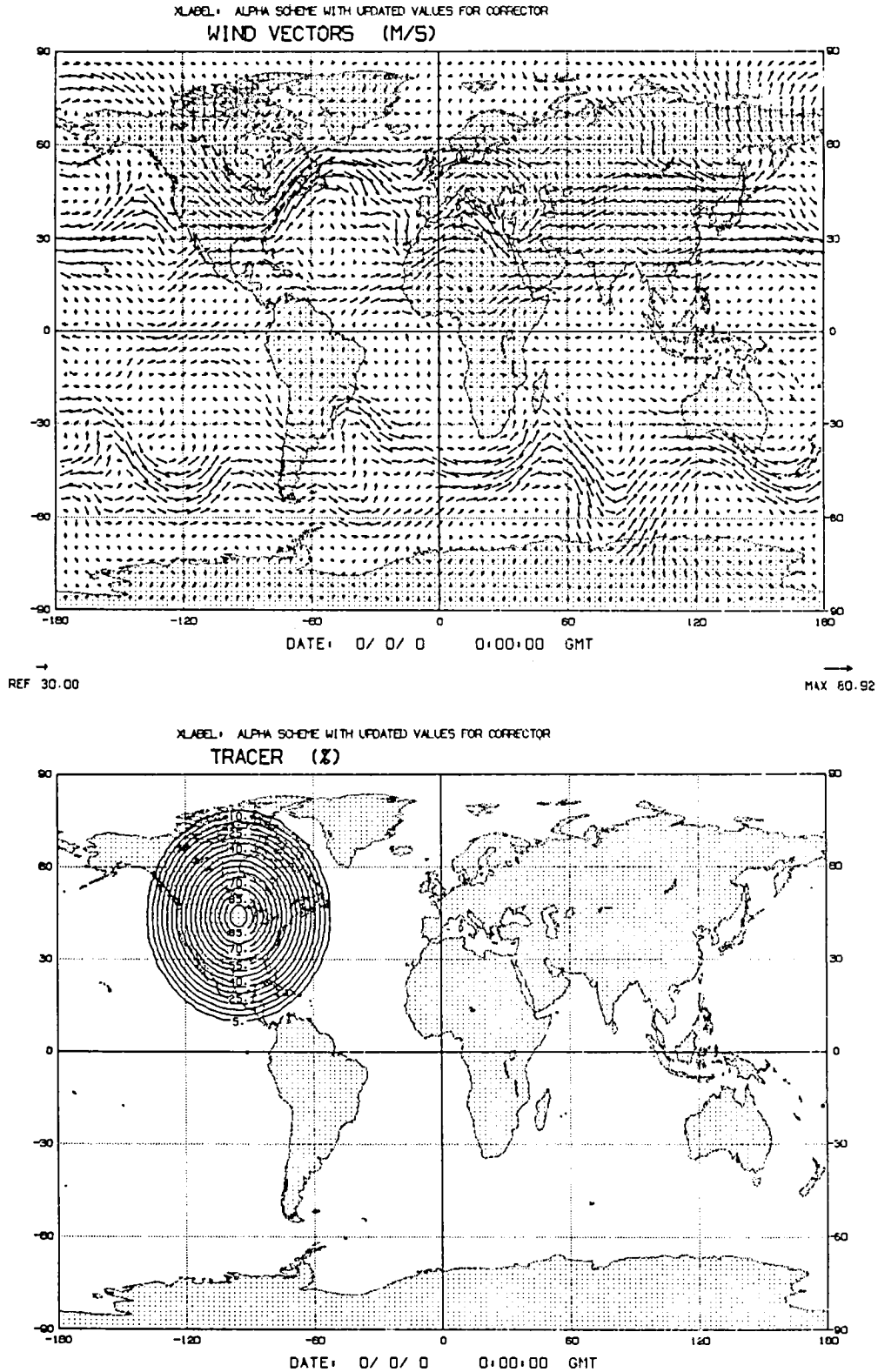


Fig. 11. Tracer initial conditions, using 72×46 resolution

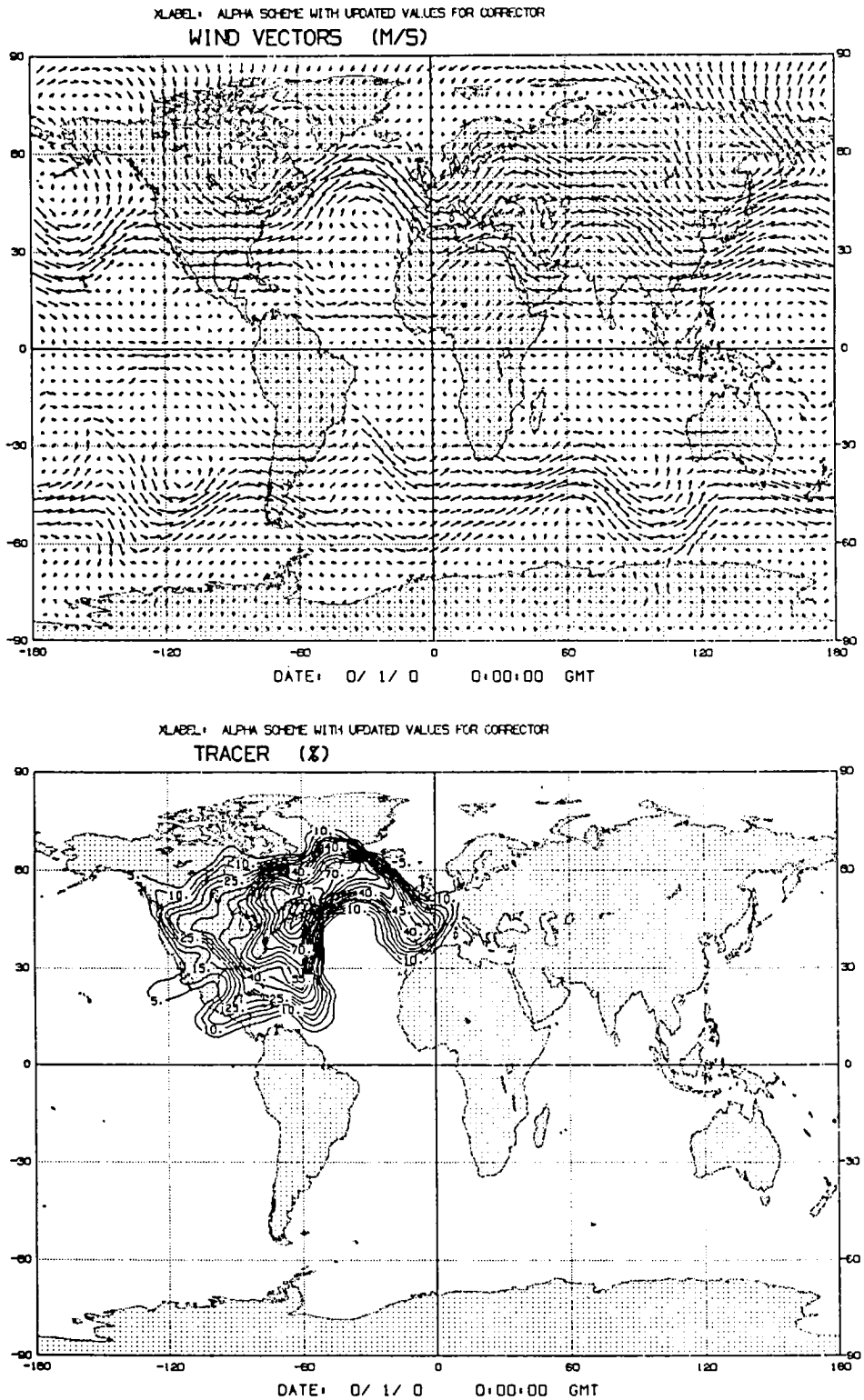
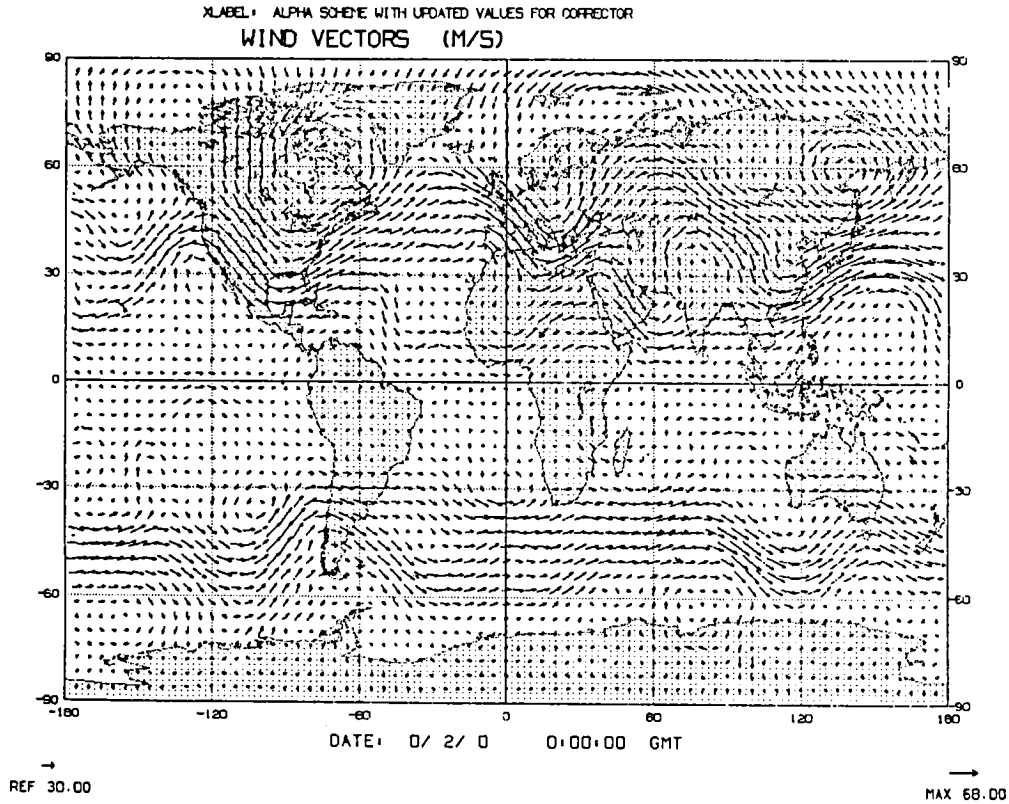


Fig. 12. Tracer experiment after 1 day



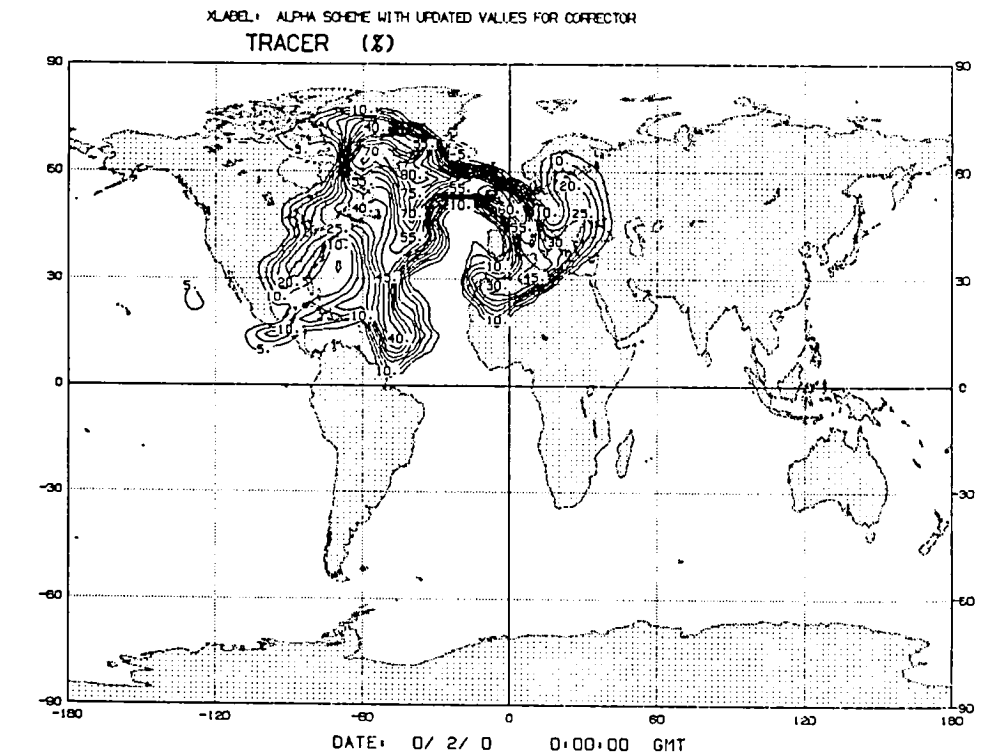


Fig. 13. Tracer experiment after 2 days

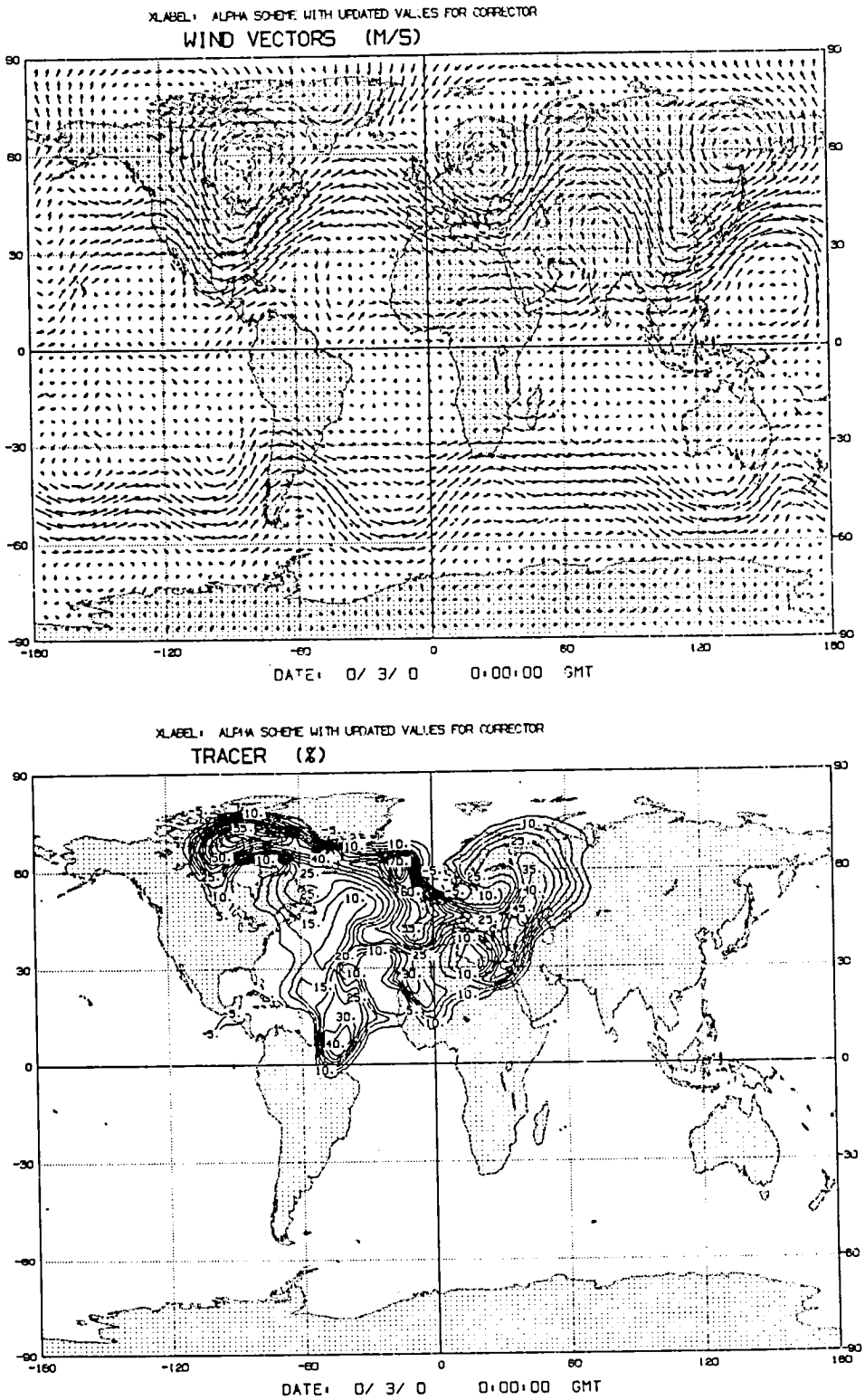


Fig. 14. Tracer experiment after 3 days

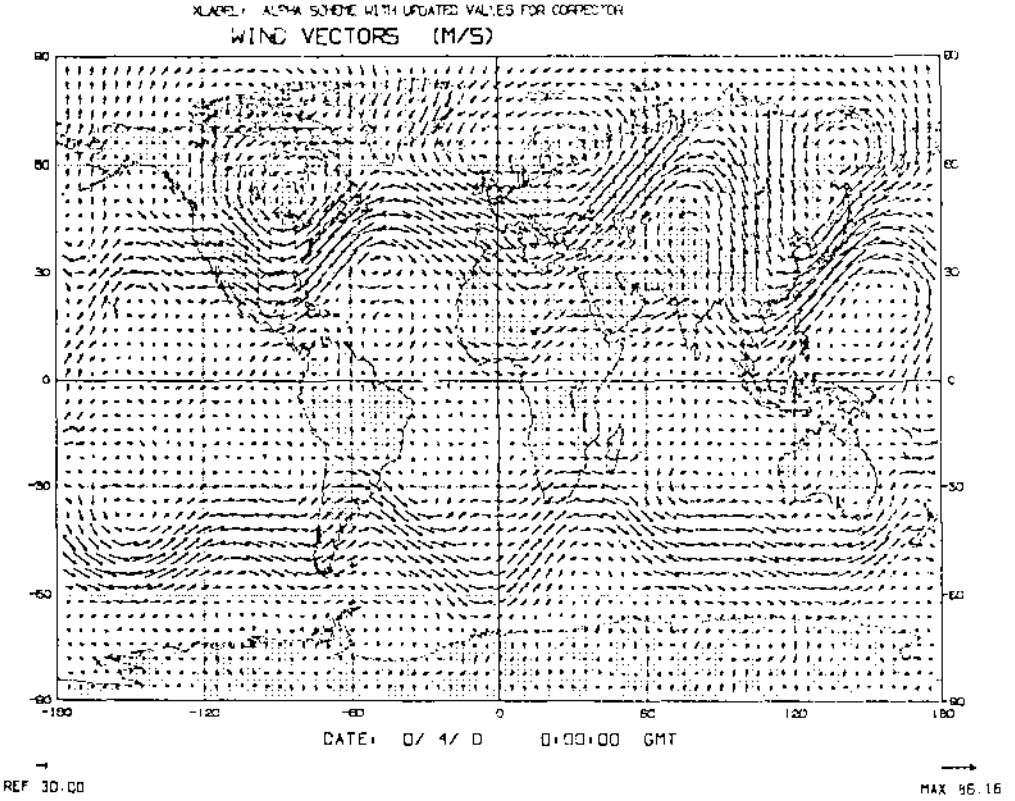


Fig. 15. Tracer experiment after 4 days

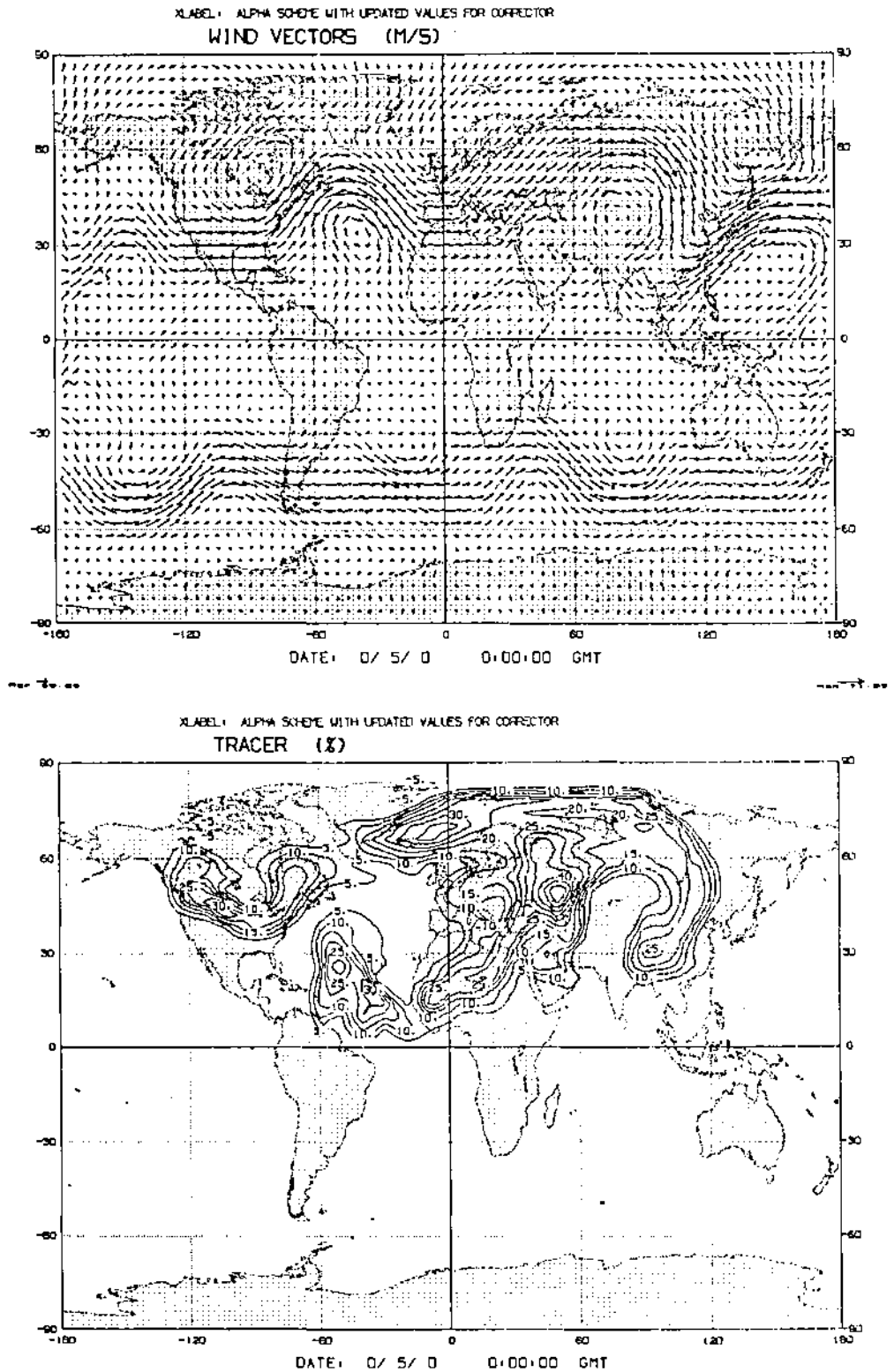


Fig. 16. Tracer experiment after 5 days

b. SWE source code

The following is the FORTRAN source code for the 2-layer shallow water model.

```
C ***************************************************************
C *  M / A - C O M  S I G M A  D A T A  I N C .  N A S A - G S F C  *
C ***************************************************************
C
C     PROGRAM SWE      (INPUT,  OUTPUT,
C    $                  UNIT5  = INPUT,
C    $                  UNIT6  = OUTPUT,
C    $                  UNIT45 = VARS,
C    $                  UNIT46 = ICSET )
C
C ***********************************************************
C ****                                                   ****
C ****  THIS PROGRAM SOLVES A TWO-LAYER SHALLOW WATER    ****
C ****  MODEL WITH TOPOGRAPHY ON A SPHERE.  THE NUMBER   ****
C ****  OF LAYERS USED IS A NAMELIST PARAMETER.          ****
C ****                                                   ****
C ****  SPATIAL DERIVATIVES ARE APPROXIMATED BY          ****
C ****  ENERGY-CONSERVING FOURTH-ORDER DIFFERENCES.      ****
C ****  THE MATSUNO OR LEAPFROG TIME-SCHEME MAY BE       ****
C ****  USED AND IS A NAMELIST PARAMETER.                ****
C ****                                                   ****
C ****  THE POLES ARE TREATED AS SINGULAR POINTS AND     ****
C ****  A SPECIAL SOLUTION IS USED THERE.                ****
C ****                                                   ****
C ****  A SIXTEENTH-ORDER SHAPIRO FILTER IS PERIOD-       ****
C ****  ICALLY APPLIED TO THE HEIGHTS IN BOTH            ****
C ****  DIRECTIONS AND TO THE WINDS IN THE ZONAL         ****
C ****  DIRECTION TO PREVENT NON-LINEAR INSTABILITY.     ****
C ****  HIGH-LATITUDE FOURIER FILTERING IS DONE TO       ****
C ****  PREVENT LINEAR INSTABILITY.                      ****
C ****                                                   ****
C ***********************************************************
C
C
      NAMELIST /TSCHEM/ SCHEME, NLEAP
      NAMELIST /INPUT / TSTART, TMAX, DT, TPRT
      NAMELIST /LEVELS/ KMAX  , KAPPA
      NAMELIST /FILT  / GFILT , HFILT, LNMF, LNMI, LRASCH, NMIMAX, TSHAP
      NAMELIST /STRESS/ LFRICT, TAO
      NAMELIST /FORCE / LPRIME, LFORCE, LTRACE, ALPHA
C
      LOGICAL  LPRIME, LFORCE, LFRICT, LTRACE
      LOGICAL  HFILT,  GFILT,  LNMF, LNMI, LRASCH
C
C
C
      REAL   KAPPA
      REAL*4 ZMAT
      REAL*4 ZLEP
      REAL*4 SCHEME
```

```
C
      DATA ZMAT /'MATS'/
      DATA ZLEP /'LEAP'/
C
      DIMENSION TP(500),TPSTAR(500),IMAXD2(128),RHO(2)
      DIMENSION STOFFT(735)
C
      LOGICAL FALSE
      INTEGER CORREC
      DATA    M1 /-1/, FALSE/.F./
C
C
      COMMON/PRIME/  HPRIME(72,46,2)
      COMMON/PRIME/  UPRIME(72,46,2)
      COMMON/PRIME/  VPRIME(72,46,2)
C
C
      COMMON/UVH/      U(72,46,2)
      COMMON/UVH/      V(72,46,2)
      COMMON/UVH/     HT(72,46,3)
      COMMON/UVH/      Q(72,46,2)
      COMMON/UVH/ HFORCE(72,46,2)
      COMMON/UVH/     HH(72,46)
C
      COMMON/UVH/  UBAR(46,2)
      COMMON/UVH/  VBAR(46,2)
      COMMON/UVH/ HTBAR(2)
C
C
      COMMON/UVHP/      UP(2,2)
      COMMON/UVHP/      VP(2,2)
      COMMON/UVHP/     HTP(2,3)
C
C
      COMMON/DOT/      H(72,46,2,2)
      COMMON/DOT/     HU(72,46,2,2)
      COMMON/DOT/     HV(72,46,2,2)
      COMMON/DOT/     HQ(72,46,2)
      COMMON/DOT/   HDOT(72,46)
      COMMON/DOT/  HUDOT(72,46)
      COMMON/DOT/  HVDOT(72,46)
C
C
      COMMON/DOTP/      HP(2,2,2)
      COMMON/DOTP/     HUP(2,2,2)
      COMMON/DOTP/     HVP(2,2,2)
      COMMON/DOTP/   HPDOT(2)
      COMMON/DOTP/  HUPDOT(2)
      COMMON/DOTP/  HVPDOT(2)
C
C
      COMMON /WORK/ SPACE(72,46,16)
```

```
C
      COMMON/PARM/    IMAX,JMAX,DL,DP,A,DT,K,KMAX,TITLE(20),
     $                CON( 46),
     $               COSP( 46),
     $               SINP( 46),
     $               COSL( 72),
     $               SINL( 72),
     $                  F( 46)
      COMMON /DAMP/  DAMPG(46,36)
      COMMON /DAMP/  DAMPA(46,36)
C
C
C
      DIMENSION   HT1X(72,1),  HT2X(72,1)
      DIMENSION   HT1Y(72,1),  HT2Y(72,1)
      DIMENSION   HU1X(72,1),  HU2X(72,1)
      DIMENSION   HV1Y(72,1),  HV2Y(72,1)
      DIMENSION  HUU1X(72,1), HUU2X(72,1)
      DIMENSION  HUV1X(72,1), HUV2X(72,1)
      DIMENSION  HVU1Y(72,1), HVU2Y(72,1)
      DIMENSION  HVV1Y(72,1), HVV2Y(72,1)
C
C
      EQUIVALENCE (  HT1X(1,1),SPACE(1,1,01) )
      EQUIVALENCE (  HT2X(1,1),SPACE(1,1,02) )
      EQUIVALENCE (  HT1Y(1,1),SPACE(1,1,03) )
      EQUIVALENCE (  HT2Y(1,1),SPACE(1,1,04) )
      EQUIVALENCE (  HU1X(1,1),SPACE(1,1,05) )
      EQUIVALENCE (  HU2X(1,1),SPACE(1,1,06) )
      EQUIVALENCE (  HV1Y(1,1),SPACE(1,1,07) )
      EQUIVALENCE (  HV2Y(1,1),SPACE(1,1,08) )
      EQUIVALENCE ( HUU1X(1,1),SPACE(1,1,09) )
      EQUIVALENCE ( HUU2X(1,1),SPACE(1,1,10) )
      EQUIVALENCE ( HUV1X(1,1),SPACE(1,1,11) )
      EQUIVALENCE ( HUV2X(1,1),SPACE(1,1,12) )
      EQUIVALENCE ( HVU1Y(1,1),SPACE(1,1,13) )
      EQUIVALENCE ( HVU2Y(1,1),SPACE(1,1,14) )
      EQUIVALENCE ( HVV1Y(1,1),SPACE(1,1,15) )
      EQUIVALENCE ( HVV2Y(1,1),SPACE(1,1,16) )
C
C
      IMAX = 72
      JMAX = 46
      NMAX = IMAX/2
C
C
      CALL CLOCK
C
      READ(5,TSCHEM)
      READ(5,INPUT)
      READ(5,LEVELS)
      READ(5,FILT)
      READ(5,STRESS)
      READ(5,FORCE)
```

```
C
C ***********************************************************
C ****              PHYSICAL PARAMETERS                  ****
C ***********************************************************
C
      PI    = 4.0*ATAN(1.0E0)
      G     = 9.81
      OMEGA = 0.7272205E-4
      A     = 6.372E06
C
C ***********************************************************
C ****                TIME PARAMETERS                    ****
C ****               DT IS IN SECONDS                    ****
C ****          TMAX AND TPRT ARE IN HOURS               ****
C ***********************************************************
C
      TSHAP = TSHAP*3600.
C
      IF(SCHEME.EQ.ZMAT) MATS = 1
      IF(SCHEME.EQ.ZLEP) MATS = 0
C
C     CHANGE TIME UNITS TO SECS.
      TM    = TMAX
      TO    = TSTART
      TMAX = TMAX * 3600.0
      DO 40 I=1,500
      TPSTAR(I) = I*TPRT
   40     TP(I) = (TPSTAR(I)+TO) * 3600.0
C
C ***********************************************************
C ****              DEFINE DOMAIN LIMITS                 ****
C ***********************************************************
C
      JSP = 1
      JNP = JMAX
C
      DP = PI/(JNP-JSP)
      DL = 2.0*PI/IMAX
      DPSTAR = DP*180/PI
C
C ***********************************************************
C ****           BEGIN MAIN INTEGRATION LOOP             ****
C ***********************************************************
C
   15 CONTINUE
C
C ***********************************************************
C ****    CHOOSE PREDICTOR OR CORRECTOR FOR MATSUNO     ****
C ***********************************************************
C
      IF(MATS.EQ.1) CORREC = CORREC * M1
C
      DO 2 K=1,KMAX
C
C ***********************************************************
C ****              SET CURRENT VALUES                   ****
C ***********************************************************
```

```
C
      DO 101 J=JSP,JNP
      DO 101 I=1,IMAX
      HH(I,J)      = RHO(K)*( HT(I,J,1)-HT(I,J,2) ) + HT(I,J,2)
       H(I,J,K,N) = HT(I,J,K)    - HT(I,J,K+1)
      HU(I,J,K,N) =  H(I,J,K,N) *  U(I,J,K)
      HV(I,J,K,N) =  H(I,J,K,N) *  V(I,J,K)
  101 CONTINUE
C
      DO 102 M=1,2
       HP(M,K,N) = HTP(M,K) - HTP(M,K+1)
      HUP(M,K,N) =  HP(M,K,N)  * UP(M,K)
      HVP(M,K,N) =  HP(M,K,N)  * VP(M,K)
  102 CONTINUE
C
C ***********************************************************
C ****              CALL TRACER ROUTINE                  ****
C ***********************************************************
C
      IF(LTRACE) CALL TRACER ( H(1,1,K,N),  U(1,1,K), V(1,1,K),
     $                         Q(1,1,K),   HQ(1,1,K), IMAX,JMAX )
C
C ***********************************************************
C ****         CALCULATE AVERAGE QUANTITIES              ****
C ***********************************************************
C
      DO 110 J=JSP,JNPM2
      JP1=J+1
      JP2=J+2
C
      I=IMAX-1
      IP1=IMAX
      DO 120 IP2=1,IMAX
C
        U1X         =   U(IP1,J,K) +  U(I,J,K)
        V1X         =   V(IP1,J,K) +  V(I,J,K)
       HU1X(I,J)    =  HU(IP1,J,K,N) + HU(I,J,K,N)
      HUU1X(I,J)    =  HU1X(I,J) * U1X
      HUV1X(I,J)    =  HU1X(I,J) * V1X
       HT1X(I,J)    =  HH(IP1,J) + HH(I,J)
C
        U2X         =   U(IP2,J,K) +  U(I,J,K)
        V2X         =   V(IP2,J,K) +  V(I,J,K)
       HU2X(IP1,J)  =  HU(IP2,J,K,N) + HU(I,J,K,N)
      HUU2X(IP1,J)  =  HU2X(IP1,J) * U2X
      HUV2X(IP1,J)  =  HU2X(IP1,J) * V2X
       HT2X(IP1,J)  =  HH(IP2,J)   + HH(I,J)
C
        U1Y         =   U(I,JP1,K) + U(I,J,K)
        V1Y         =   V(I,JP1,K) + V(I,J,K)
       HV1Y(I,J)    =  HV(I,JP1,K,N)*COSP(JP1) + HV(I,J,K,N)*COSP(J)
      HVU1Y(I,J)    =  HV1Y(I,J) * U1Y
      HVV1Y(I,J)    =  HV1Y(I,J) * V1Y
       HT1Y(I,J)    =  HH(I,JP1) + HH(I,J)
```

```
C
        U2Y          =   U(I,JP2,K) + U(I,J,K)
        V2Y          =   V(I,JP2,K) + V(I,J,K)
       HV2Y(I,JP1)   =  HV(I,JP2,K,N)*COSP(JP2) + HV(I,J,K,N)*COSP(J)
      HVU2Y(I,JP1)   =  HV2Y(I,JP1) * U2Y
      HVV2Y(I,JP1)   =  HV2Y(I,JP1) * V2Y
       HT2Y(I,JP1)   =  HH(I,JP2)   + HH(I,J)
C
      I=IP1
      IP1=IP2
  120 CONTINUE
  110 CONTINUE
C
C ***********************************************************
C ****     CALCULATE AVERAGE QUANTITIES NEAR POLE       ****
C ***********************************************************
C
      J  =JNPM1
      JP1=J+1
C
      I=IMAX-1
      IP1=IMAX
      DO 130 IP2=1,IMAX
C
               U1X =   U(IP1,J,K) +  U(I,J,K)
               V1X =   V(IP1,J,K) +  V(I,J,K)
       HU1X(I,J)   =  HU(IP1,J,K,N) + HU(I,J,K,N)
      HUU1X(I,J)   =  HU1X(I,J) * U1X
      HUV1X(I,J)   =  HU1X(I,J) * V1X
       HT1X(I,J)   =  HH(IP1,J) + HH(I,J)
C
               U2X =   U(IP2,J,K) +  U(I,J,K)
               V2X =   V(IP2,J,K) +  V(I,J,K)
       HU2X(IP1,J) =  HU(IP2,J,K,N) + HU(I,J,K,N)
      HUU2X(IP1,J) =  HU2X(IP1,J) * U2X
      HUV2X(IP1,J) =  HU2X(IP1,J) * V2X
       HT2X(IP1,J) =  HH(IP2,J)   + HH(I,J)
C
               U1Y =   U(I,JP1,K) + U(I,J,K)
               V1Y =   V(I,JP1,K) + V(I,J,K)
       HV1Y(I,J)   =  HV(I,JP1,K,N)*COSP(JP1) + HV(I,J,K,N)*COSP(J)
      HVU1Y(I,J)   =  HV1Y(I,J) * U1Y
      HVV1Y(I,J)   =  HV1Y(I,J) * V1Y
       HT1Y(I,J)   =  HH(I,JP1) + HH(I,J)
C
       HV2Y(I,JSP) =   0.0
      HVU2Y(I,JSP) =   0.0
      HVV2Y(I,JSP) =   0.0
       HT2Y(I,JSP) =   HH(I+IMAXD2(I),JSPP1) + HH(I,JSPP1)
C
       HV2Y(I,JNP) =   0.0
      HVU2Y(I,JNP) =   0.0
      HVV2Y(I,JNP) =   0.0
       HT2Y(I,JNP) =   HH(I+IMAXD2(I),JNPM1) + HH(I,JNPM1)
```

```
C
      I=IP1
      IP1=IP2
  130 CONTINUE
C
C ***********************************************************
C ****          CALCULATE TIME TENDENCIES                ****
C ***********************************************************
C
      DO 150 J=JSPP1,JNPM1
      JP1=J+1
      JM1=J-1
C
      I  = IMAX
      IM1= IMAX-1
      DO 160 IP1=1,IMAX
C
      FSTAR =   F(J) + U(I,J,K)*SINP(J) * CON(J)
       PHIZ =    G   * H(I,J,K,N)
C
      HUCOR =   FSTAR *  HV(I,J,K,N)
      HVCOR =   FSTAR *  HU(I,J,K,N)
      DHTDL =  ( HT1X(I,J)  -HT1X(IM1,J) )*DL24
     $              - ( HT2X(IP1,J)-HT2X(IM1,J) )*DL4
      DHTDP =  ( HT1Y(I,J)  -HT1Y(I,JM1) )*DP24
     $              - ( HT2Y(I,JP1)-HT2Y(I,JM1) )*DP4
      HUPRES =   PHIZ * CON(J) * DHTDL
      HVPRES =   PHIZ * AINV   * DHTDP
C
C
      HDOT(I,J)   = -CON(J) * (
     $                ( HU1X( I , J ) - HU1X(IM1, J )  )*DL24
     $              - ( HU2X(IP1, J ) - HU2X(IM1, J )  )*DL4
     $           +    ( HV1Y( I , J ) - HV1Y( I ,JM1)  )*DP24
     $              - ( HV2Y( I ,JP1) - HV2Y( I ,JM1)  )*DP4 )
C
C
      HUDOT(I,J)   = -CON(J) * (
     $                ( HUU1X( I , J ) - HUU1X(IM1, J ) )*DL44
     $              - ( HUU2X(IP1, J ) - HUU2X(IM1, J ) )*DL8
     $           +    ( HVU1Y( I , J ) - HVU1Y( I ,JM1) )*DP44
     $              - ( HVU2Y( I ,JP1) - HVU2Y( I ,JM1) )*DP8 )
     $           + HUCOR - HUPRES
C
C
      HVDOT(I,J)   = -CON(J) * (
     $                ( HUV1X( I , J ) - HUV1X(IM1, J ) )*DL44
     $              - ( HUV2X(IP1, J ) - HUV2X(IM1, J ) )*DL8
     $           +    ( HVV1Y( I , J ) - HVV1Y( I ,JM1) )*DP44
     $              - ( HVV2Y( I ,JP1) - HVV2Y( I ,JM1) )*DP8 )
     $           - HVCOR - HVPRES
C
      IM1=I
      I  =IP1
  160 CONTINUE
  150 CONTINUE
```

```
C
C ***********************************************************
C ****       CALCULATE TIME TENDENCIES AT POLES          ****
C ***********************************************************
C
      DO 170 M=1,2
C
      MSGN = (-1)**M
C
      JSTAR1 = JSP+1 + (M-1)*( JNP-1 - (JSP+1) )
      JSTAR2 = JSP+2 + (M-1)*( JNP-2 - (JSP+2) )
      JPOLE  = JSP   + (M-1)*( JNP - JSP )
C
      SUM11 = 0.0
      SUM12 = 0.0
      SUM21 = 0.0
      SUM22 = 0.0
      SUM41 = 0.0
      SUM42 = 0.0
      SUM31 = 0.0
      SUM32 = 0.0
      SUM51 = 0.0
      SUM52 = 0.0
      DO 180 I=1,IMAX
C
      SUM11 = SUM11 +  HV(I,JSTAR1,K,N)
      SUM12 = SUM12 +  HV(I,JSTAR2,K,N)
      SUM21 = SUM21 +  HV(I,JSTAR1,K,N) * (-U(I,JSTAR1,K)*SINL(I)
     $      - MSGN * V(I,JSTAR1,K)*COSL(I) )
      SUM22 = SUM22 +  HV(I,JSTAR2,K,N) * (-U(I,JSTAR2,K)*SINL(I)
     $      - MSGN * V(I,JSTAR2,K)*COSL(I) )
      SUM41 = SUM41 +  HV(I,JSTAR1,K,N)*( MSGN * U(I,JSTAR1,K)*COSL(I)
     $      - V(I,JSTAR1,K)*SINL(I) )
      SUM42 = SUM42 +  HV(I,JSTAR2,K,N)*( MSGN * U(I,JSTAR2,K)*COSL(I)
     $      - V(I,JSTAR2,K)*SINL(I) )
C
      SUM31 = SUM31 + COSL(I) * HH(I,JSTAR1)
      SUM32 = SUM32 + COSL(I) * HH(I,JSTAR2)
      SUM51 = SUM51 + SINL(I) * HH(I,JSTAR1)
      SUM52 = SUM52 + SINL(I) * HH(I,JSTAR2)
C
  180 CONTINUE
C
       HPDOT(M) =   MSGN * ( C11*SUM11 - C12*SUM12 )
C
      HUPDOT(M) = ( MSGN * ( C11*SUM21 - C12*SUM22 )
     $     - G* HP(M,K,N)* ( C11*SUM31 - C12*SUM32 ) )
     $     + F(JPOLE)* HVP(M,K,N)
C
      HVPDOT(M) = MSGN * ( ( C11*SUM41 - C12*SUM42 )
     $   - G* HP(M,K,N)*   ( C11*SUM51 - C12*SUM52 ) )
     $   - F(JPOLE)* HUP(M,K,N)
C
  170 CONTINUE
```

```
C
C ********************************************************
C ****     APPLY FOURIER FILTER TO TIME TENDENCIES     ****
C ********************************************************
C
      IF(.NOT.HFILT) GOTO 325
C
      CALL FILTER ( HDOT,IMAX,JMAX,STOFFT)
      CALL FILTER (HUDOT,IMAX,JMAX,STOFFT)
      CALL FILTER (HVDOT,IMAX,JMAX,STOFFT)
C
  325 CONTINUE
C
C ********************************************************
C ****              ADD FORCING TERMS                 ****
C ********************************************************
C
      IF(.NOT.LFORCE) GOTO 260
C
      DO 265 J=JSPP1,JNPM1
      DO 265 I=1,IMAX
       HDOT(I,J) =  HDOT(I,J) + ALPHA*HFORCE(I,J,K)
  265 CONTINUE
C
  260 CONTINUE
C
C ********************************************************
C **** SUBTRACT INITIAL TENDENCY FOR PERTURBATION EXP. ****
C ********************************************************
C
      IF(MATS.NE.1)    GOTO 285
      IF(.NOT.LPRIME)  GOTO 285
      IF(CORREC.EQ.-1) GOTO 285
C
      DO 275 J=JSPP1,JNPM1
      DO 275 I=1,IMAX
       HDOT(I,J) =  HDOT(I,J) - HPRIME(I,J,K)
      HUDOT(I,J) = HUDOT(I,J) - UPRIME(I,J,K)
      HVDOT(I,J) = HVDOT(I,J) - VPRIME(I,J,K)
  275 CONTINUE
C
  285 CONTINUE
C
C ********************************************************
C ****   USE LEAP FROG OR MATSUNO TO ADVANCE SOLUTION  ****
C ********************************************************
C
                        Z = 2.0
      IF(NCALL.EQ.1)    Z = 1.0
                      ZDT = Z*DT
```

```
C
      DO 270 J=JSPP1,JNPM1
      DO 270 I=1,IMAX
       H(I,J,K,NP1) =  H(I,J,K,NM1) + ZDT* HDOT(I,J)
      HU(I,J,K,NP1) = HU(I,J,K,NM1) + ZDT*HUDOT(I,J)
      HV(I,J,K,NP1) = HV(I,J,K,NM1) + ZDT*HVDOT(I,J)
  270 CONTINUE
C
      DO 280 M=1,2
       HP(M,K,NP1) =  HP(M,K,NM1) + ZDT* HPDOT(M)
      HUP(M,K,NP1) = HUP(M,K,NM1) + ZDT*HUPDOT(M)
      HVP(M,K,NP1) = HVP(M,K,NM1) + ZDT*HVPDOT(M)
  280 CONTINUE
C
C ************************************************************
C ****                    END K-LOOP                      ****
C ************************************************************
C
    2 CONTINUE
C
C ************************************************************
C ****      CALCULATE NEW VELOCITIES AND HEIGHTS          ****
C ************************************************************
C
      DO 295 KK=1,KMAX
      K = KMAX - KK + 1
C
      DO 290 J=JSPP1,JNPM1
      DO 290 I=1,IMAX
       U(I,J,K) = 0.0
       V(I,J,K) = 0.0
      HT(I,J,K) = HT(I,J,K+1)  + H(I,J,K,NP1)
      IF(H(I,J,K,NP1).LE.0.0) GOTO 290
       U(I,J,K) = HU(I,J,K,NP1) / H(I,J,K,NP1)
       V(I,J,K) = HV(I,J,K,NP1) / H(I,J,K,NP1)
       Q(I,J,K) = HQ(I,J,K)     / H(I,J,K,NP1)
  290 CONTINUE
C
      DO 300 M=1,2
       UP(M,K) = 0.0
       VP(M,K) = 0.0
      HTP(M,K) = HTP(M,K+1)  + HP(M,K,NP1)
      IF(HP(M,K,NP1).LE.0.0) GOTO 300
       UP(M,K) = HUP(M,K,NP1) / HP(M,K,NP1)
       VP(M,K) = HVP(M,K,NP1) / HP(M,K,NP1)
  300 CONTINUE
C
C ************************************************************
C ****        APPLY BOUNDARY CONDITIONS AT POLES          ****
C ************************************************************
C
      DO 310 M=1,2
C
      MSGN  = (-1)**M
      JPOLE = JSP + (M-1)*( JNP - JSP )
```

```
C
      DO 320 I=1,IMAX
       U(I,JPOLE,K) = -UP(M,K)*SINL(I) + MSGN * VP(M,K)*COSL(I)
       V(I,JPOLE,K) = -MSGN * UP(M,K)*COSL(I) - VP(M,K)*SINL(I)
      HT(I,JPOLE,K) =  HTP(M,K)
       Q(I,JPOLE,K) =  HQ(I,JPOLE,K)/HTP(M,K)
  320 CONTINUE
C
  310 CONTINUE
  295 CONTINUE
C
C ***********************************************************
C ****          DO BACKWARD STEP FOR MATSUNO              ****
C ***********************************************************
C
      IF( NCALL.NE.1 )  GOTO 412
C
         N   = NP1
      NSTEP = NSTEP + 1
      IF( NSTEP.EQ.1 )  GOTO 15
      NSTEP = 0
C
  412 CONTINUE
C
C ***********************************************************
C ****         ADD FRICTION AND ZONAL FORCING             ****
C ***********************************************************
C
      IF(.NOT.LFRICT) GOTO 221
      DO 224 K=1,KMAX
      CALL FRICT (U(1,1,K),V(1,1,K),UBAR(1,K),VBAR(1,K),
     $             IMAX,JMAX,DT,TAO)
  224 CONTINUE
C
  221 CONTINUE
C
C ***********************************************************
C ****                   UPDATE TIME                       ****
C ***********************************************************
C
      T = T+DT
C
C ***********************************************************
C ****         APPLY NORMAL MODE FILTER                  ****
C ***********************************************************
C
      IF(.NOT.LNMF)             GOTO 321
C
      DO 324 K=1,KMAX
      CALL NORMAL ( U(1,1,K),V(1,1,K),HT(1,1,K),
     $              HTBAR(K),IMAX,JMAX,DT,LNMI,LNMF,LRASCH )
  324 CONTINUE
C
  321 CONTINUE
C
```

```
C ********************************************************
C ****              APPLY SHAPIRO FILTER                ****
C ********************************************************
C
      IF(.NOT.GFILT)                          GOTO 330
      IF((T-TSTART).LT.NEXT*TSHAP)            GOTO 330
C
      DO 600 K=1,KMAX
      CALL SHAP ( U(1,1,K),IMAX,JMAX,1)
      CALL SHAP ( V(1,1,K),IMAX,JMAX,1)
      CALL SHAP (HT(1,1,K),IMAX,JMAX,2)
      DO 610 M=1,2
      JPOLE = JSP + (M-1)*(JNP-JSP)
      S1 = 0.
      DO 620 I=1,IMAX
      S1 = S1 + HT(I,JPOLE,K)
  620 CONTINUE
      HTSTAR = S1/IMAX
      DO 630 I=1,IMAX
      HT(I,JPOLE,K) = HTSTAR
  630 CONTINUE
  610 CONTINUE
  600 CONTINUE
C
      TSTAR = T/3600.
      WRITE (6,3322) TSTAR,NCALL,NTOT
 3322 FORMAT(1X,'TIME :',1X,F8.3,1X,'HRS.',6X,'NCALL :',I3,4X,
     $ 'NTOT :',I4,4X,'SHAPIRO FILTER APPLIED')
C
      IF(SCHEME.EQ.ZMAT)   NEXT = NEXT + 1
      IF(SCHEME.EQ.ZMAT)   GOTO 330
C
      ICOUNT = ICOUNT+1
      IF(ICOUNT.EQ.1)      GOTO 330
      ICOUNT = 0
      NEXT   = NEXT + 1
  330 CONTINUE
C
C ********************************************************
C ****       TEST FOR INSTABILITY ON HEIGHT FIELD     ****
C ********************************************************
C
      DO 7 K=1,KMAX
C
      HBAR = 0.1*HTBAR(K)
      DO 450 J=JSP,JNP
      DO 450 I=1,IMAX
      IF(HT(I,J,K) .LE. HBAR) FLAG = 1.0
  450 CONTINUE
```

```
C
      IF(FLAG.EQ.0.0) GOTO 7
      TSTAR = T/3600.0
      WRITE(6,480) TSTAR
      WRITE(18,4117) TITLE,K,TSTAR
      CALL HARM (HT(1,1,K),IMAX,JMAX,18)
 4117 FORMAT(/1X,20A4,/1X,'LEVEL ',I2,' BECAME UNSTABLE AT T = ',
     $        F7.3,' HOURS',//)
      GOTO 490
C
    7 CONTINUE
C
C ***********************************************************
C ****       PERFORM NORMAL MODE INITIALIZATION          ****
C ***********************************************************
C
      IF(.NOT.LNMI) GOTO 457
C
      DO 458 K=1,KMAX
      CALL NORMAL ( U(1,1,K),V(1,1,K),HT(1,1,K),
     $              HTBAR(K),IMAX,JMAX,DT,LNMI,LNMF,LRASCH )
  458 CONTINUE
      CALL POLE
           T = T-DT
        NMICNT = NMICNT + 1
      IF(NMICNT.LT.NMIMAX) GOTO 14
      LNMI = FALSE
      WRITE(45)   TITLE,IMAX,JMAX
      WRITE(45) ((HT(I,J,KTPG),I=1,IMAX),J=1,JMAX)
      DO 459 K=1,KMAX
      WRITE(45)    K,T
      WRITE(45) (( Q(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( U(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( V(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) ((HT(I,J,K),I=1,IMAX),J=1,JMAX)
  459 CONTINUE
      CALL CLOCK
      GOTO 14
C
  457 CONTINUE
C
C ***********************************************************
C ****                  PRINT OUTPUT                     ****
C ***********************************************************
C
      IF(T-TP(IPN).LT. 0.0) GOTO 500
C
  490 CONTINUE
      DO 8 K=1,KMAX
      TSTAR=T/3600.0
C
      WRITE(45)    K,TSTAR
      WRITE(45) (( Q(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( U(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( V(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) ((HT(I,J,K),I=1,IMAX),J=1,JMAX)
```

```
C
      WRITE (6,749) TSTAR,NCALL,NTOT,K
C
    8 CONTINUE
C
C ************************************************************
C ****            CHECK FOR END OF RUN                    ****
C ************************************************************
C
      IF(FLAG .EQ. 1.0) STOP
      IPN=IPN+1
C
  500 CONTINUE
C
      TDAY = 86400.0 * AINT(T/86400.0)
C
      IF(T.EQ.TDAY)      CALL CLOCK
      IF(T.GE.TMAX)      CALL CLOCK
      IF(T.GE.TMAX)      STOP
C
C ************************************************************
C ****           SWITCH NM1, N, NP1 ARRAYS                ****
C ************************************************************
C
      IF(SCHEME.EQ.ZLEP) NCALL = NCALL+1
      GOTO(431,432,433), NCALL
C
C
      COMMON/PARM/    IMAX,JMAX,DL,DP,A,DT,K,KMAX,TITLE(20),
     $                 CON( 46),
     $                COSP( 46),
     $                SINP( 46),
     $                COSL( 72),
     $                SINE( 46),
C
                           KTP = KMAX+1
      READ(45,END=100)    TITLE,IMAX,JMAX
      READ(45,END=100) ((HT(I,J,KTP),I=1,IMAX),J=1,JMAX)
C
      DO 10 K=1,KMAX
      DO 20 J=1,JMAX
      DO 30 I=1,IMAX
      HPRIME(I,J,K) = 0.0
      UPRIME(I,J,K) = 0.0
      VPRIME(I,J,K) = 0.0
   30 CONTINUE
   20 CONTINUE
   10 CONTINUE
C
  200 CONTINUE
      DO 40 L=1,KMAX
      READ(45,END=100)     K,TSTAR
      READ(45,END=100) (( U(I,J,K),I=1,IMAX),J=1,JMAX)
      READ(45,END=100) (( V(I,J,K),I=1,IMAX),J=1,JMAX)
      READ(45,END=100) ((HT(I,J,K),I=1,IMAX),J=1,JMAX)
   40 CONTINUE
```

```
C
      DO 50 KK=1,KMAX
      K = KTP-KK
      DO 60 J=1,JMAX
      DO 70 I=1,IMAX
      H(I,J,K,1) = HT(I,J,K)-HT(I,J,K+1)
   70 CONTINUE
   60 CONTINUE
C
      DO 80 J=1,JMAX
      DO 90 I=1,IMAX
      HPRIME(I,J,K) =          H(I,J,K,1) - HPRIME(I,J,K)
      UPRIME(I,J,K) = U(I,J,K)*H(I,J,K,1) - UPRIME(I,J,K)
      VPRIME(I,J,K) = V(I,J,K)*H(I,J,K,1) - VPRIME(I,J,K)
   90 CONTINUE
   80 CONTINUE
   50 CONTINUE
C
      GOTO 200
  100 CONTINUE
C
      DO 110 K=1,KMAX
      DO 120 J=1,JMAX
      DO 130 I=1,IMAX
      HPRIME(I,J,K) = HPRIME(I,J,K)/DT
      UPRIME(I,J,K) = UPRIME(I,J,K)/DT
      VPRIME(I,J,K) = VPRIME(I,J,K)/DT
  130 CONTINUE
  120 CONTINUE
  110 CONTINUE
C
      REWIND 45
      RETURN
      END
      SUBROUTINE UVHR (TSTART,LFORCE,LNMI)
      DIMENSION GEOP(46,72),TOPOG(72,46)
      REAL*8     XLAB(10)
  433 NM1 = N
      N   = NP1
      NP1 = NM1
C
      IF(NCALL-1.LE.NLEAP) GOTO 439
      NSA = N
      N   = NP1
      NP1 = NSA
      NM1 = N
      NCALL = 1
      GOTO 439
C
  431 NSA = NM1
      NM1 = NP1
      NP1 = NSA
      N   = NM1
      GOTO 439
C
  432 NP1 = NM1
C
```

```
C
  439 CONTINUE
C
      NTOT =NTOT +1
      GO TO 15
C
C
  749 FORMAT(1X,'TIME :',1X,F8.3,1X,'HRS.',6X,'NCALL :',I3,4X,
     $ 'NTOT :',I4,4X,'LEVEL :',I2)
  480 FORMAT(//,5X,'HEIGHT FIELD BECAME UNSTABLE AT T=',G12.4,' HRS.')
  105 FORMAT(1X,'NO. OF GRID POINTS IN LONGITUDE:  ',I3,/,1X,
     $'NO. OF GRID POINTS IN LATITUDE:   ',I3,/,1X,
     $'GRID SIZE:  ',F5.2,' DEG  X',F5.2,' DEG',/,1X,
     $'TIME STEP:  ',F6.1,' SECS.',/,1X,
     $'PRINTING EVERY ',F6.2,1X,'HR(S) UP TO ',F7.2,1X,'HR(S). STARTI',
     $'NG AT ',F6.1,1X,'HOURS')
  108 FORMAT(1X,'DENSITY RATIO:  ',F5.2)
  109 FORMAT(1X,'NO. OF LEVELS:  ',I3)
C
      END
      SUBROUTINE PRIMES
C
      COMMON/PRIME/  HPRIME(72,46,2)
      COMMON/PRIME/  UPRIME(72,46,2)
      COMMON/PRIME/  VPRIME(72,46,2)
C
C
      COMMON/UVH/       U(72,46,2)
      COMMON/UVH/       V(72,46,2)
      COMMON/UVH/      HT(72,46,3)
      COMMON/UVH/       Q(72,46,2)
      COMMON/UVH/ HFORCE(72,46,2)
      COMMON/UVH/      HH(72,46)
C
      COMMON/UVH/  UBAR(46,2)
      COMMON/UVH/  VBAR(46,2)
      COMMON/UVH/ HTBAR(2)
C
C
      COMMON/DOT/       H(72,46,2,2)
      COMMON/DOT/      HU(72,46,2,2)
      COMMON/DOT/      HV(72,46,2,2)
      COMMON/DOT/      HQ(72,46,2)
      COMMON/DOT/   HDOT(72,46)
      COMMON/DOT/  HUDOT(72,46)
      COMMON/DOT/  HVDOT(72,46)
      EQUIVALENCE ( XLAB(1),TITLE(1) )
      LOGICAL LFORCE, LNMI
      NAMELIST /LABEL/ XLAB
C
      COMMON/UVH/       U(72,46,2)
      COMMON/UVH/       V(72,46,2)
      COMMON/UVH/      HT(72,46,3)
      COMMON/UVH/       Q(72,46,2)
      COMMON/UVH/ HFORCE(72,46,2)
      COMMON/UVH/      HH(72,46)
```

```
C
      COMMON/UVH/  UBAR(46,2)
      COMMON/UVH/  VBAR(46,2)
      COMMON/UVH/ HTBAR(2)
C
      COMMON/PARM/   IMAX,JMAX,DL,DP,A,DT,K,KMAX,TITLE(20),
     $               CON( 46),
     $              COSP( 46),
     $              SINP( 46),
     $              COSL( 72),
     $              SINL( 72),
     $                 F( 46)
C
      OMEGA = 0.7272205E-4
      G = 9.8
C
      KTP = KMAX+1
C
C ***********************************************************
C ****       READ IN INITIAL U,V,H AND TOPOGRAPHY       ****
C ***********************************************************
C
C
      READ(46,END=100)   TITLE,IMAX,JMAX
      READ(46,END=100) ((HT(I,J,KTP),I=1,IMAX),J=1,JMAX)
      READ(5,LABEL)
  200 CONTINUE
      DO 500 L=1,KMAX
      READ(46,END=100)    K,TSTAR
      READ(46,END=100) (( Q(I,J,K),I=1,IMAX),J=1,JMAX)
      READ(46,END=100) (( U(I,J,K),I=1,IMAX),J=1,JMAX)
      READ(46,END=100) (( V(I,J,K),I=1,IMAX),J=1,JMAX)
      READ(46,END=100) ((HT(I,J,K),I=1,IMAX),J=1,JMAX)
  500 CONTINUE
C
      IF(TSTAR.GE.TSTART) GOTO 100
      GOTO 200
  100 CONTINUE
C
C ***********************************************************
C ****              READ IN FORCING TERMS                ****
C ***********************************************************
C
      IF(.NOT.LFORCE) GOTO 400
      DO 300 K=1,KMAX
      READ(49)  ((HFORCE(I,J,K),I=1,IMAX),J=1,JMAX)
  300 CONTINUE
  400 CONTINUE
C
C ***********************************************************
C ****        WRITE INITIAL U,V,H AND TOPOGRAPHY       ****
C ***********************************************************
C
      CALL POLE
      IF(LNMI)    RETURN
      TSTART =    TSTAR*3600.0
```

```
      WRITE(45)   TITLE,IMAX,JMAX
      WRITE(45) ((HT(I,J,KTP),I=1,IMAX),J=1,JMAX)
      DO 600 K=1,KMAX
      WRITE(45)     K,TSTAR
      WRITE(45) (( Q(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( U(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) (( V(I,J,K),I=1,IMAX),J=1,JMAX)
      WRITE(45) ((HT(I,J,K),I=1,IMAX),J=1,JMAX)
  600 CONTINUE
C
      RETURN
      END
      SUBROUTINE FILTER (Q,IM,JM,STO)
      DIMENSION  Q(IM,JM), QTEMP(360), STO(735)
      COMMON /DAMP/  DAMPG(46,36)
      COMMON /DAMP/  DAMPA(46,36)
C
C
      NMAX = IM/2
      JMM1 = JM-1
      ZIM  = 1./FLOAT(IM)
C
      DO 10 J=2,JMM1
C
      DMIN = 1.0
      DO 15 N=1,NMAX
      DMIN = AMIN1(DAMPG(J,N),DMIN)
   15 CONTINUE
      IF(DMIN.GT.0.99) GOTO 10
C
      DO 20 I=1,IM
      QTEMP(I) = Q(I,J)
   20 CONTINUE
C
      CALL RFFTF (IM,QTEMP,STO)
      DO 30 NWAVE=1,NMAX
      N2 = NWAVE*2
         QTEMP(N2)    = QTEMP(N2)*  DAMPG(J,NWAVE)
      IF(NWAVE .EQ. NMAX)  GO TO 30
         QTEMP(N2+1) = QTEMP(N2+1)*DAMPG(J,NWAVE)
   30 CONTINUE
      CALL RFFTB (IM,QTEMP,STO)
C
      DO 40 I=1,IM
       Q(I,J) = QTEMP(I)*ZIM
   40 CONTINUE
C
   10 CONTINUE
```

```
C
      RETURN
      END
      SUBROUTINE COEFF (HTBAR)
      DIMENSION HTBAR(2)
      COMMON/PARM/   IMAX,JMAX,DL,DP,A,DT,K,KMAX,TITLE(20),
     $               CON( 46),
     $              COSP( 46),
     $              SINP( 46),
     $              COSL( 72),
     $              SINL( 72),
     $                 F( 46)
      COMMON /DAMP/  DAMPG(46,36)
      COMMON /DAMP/  DAMPA(46,36)
C
C
C ***********************************************************
C ****   SET DEFAULT VALUES FOR DAMPING COEFFICIENTS   ****
C ***********************************************************
C
       NMAX = IMAX/2
         GH = 9.81*HTBAR(1)
         CG = SQRT(GH)
       UBAR = 50.0
      DNORM = (DL*A) / (DT*(UBAR+CG))
C
      DO 10 J = 1,JMAX
         DMIN = 1.0
      DO 20 N = 1,NMAX
      THETA  = N*DL
      THETA2 = THETA + THETA
           W = DNORM * 6.0 / (8.*SIN(THETA) - SIN(THETA2))
           D = COSP(J)*W
       DAMPG(J,N) = AMIN1(1.0,D)
            DMIN = AMIN1(DAMPG(J,N),DMIN)
   20 CONTINUE
      IF (DMIN .LT. 0.99) GOTO 10
      DO 30 N = 1,NMAX
      DAMPG(J,N) = 1.0
   30 CONTINUE
   10 CONTINUE
C
      RETURN
      END
      SUBROUTINE POLE
C
      COMMON/UVH/      U(72,46,2)
      COMMON/UVH/      V(72,46,2)
      COMMON/UVH/     HT(72,46,3)
      COMMON/UVH/      Q(72,46,2)
      COMMON/UVH/ HFORCE(72,46,2)
      COMMON/UVH/     HH(72,46)
C
      COMMON/UVH/  UBAR(46,2)
      COMMON/UVH/  VBAR(46,2)
      COMMON/UVH/ HTBAR(2)
```

```
C
C
      COMMON/UVHP/       UP(2,2)
      COMMON/UVHP/       VP(2,2)
      COMMON/UVHP/      HTP(2,3)
C
      COMMON/PARM/    IMAX,JMAX,DL,DP,A,DT,K,KMAX,TITLE(20),
     $                 CON( 46),
     $                COSP( 46),
     $                SINP( 46),
     $                COSL( 72),
     $                SINL( 72),
     $                   F( 46)
C
      OMEGA = 0.7272205E-4
      G = 9.8
C
      DO 50 M=1,2
             JPOLE = 1 + (M-1)*(JMAX-1)
      HTP(M,KMAX+1) = HT(1,JPOLE,KMAX+1)
      DO 70 I=1,IMAX
      HT(I,JPOLE,KMAX+1) = HTP(M,KMAX+1)
   70 CONTINUE
   50 CONTINUE
C
      DO 80 K=1,KMAX
      DO 90  M=1,2
      JPOLE = 1 + (M-1)*(JMAX-1)
      MSGN  = (-1)**M
C
      UPOLE = 0.0
      VPOLE = 0.0
      DO 100 I=1,IMAX
      SL = SIN((I-1)*DL)
      CL = COS((I-1)*DL)
      UPOLE = UPOLE - U(I,JPOLE,K)*SL - MSGN * V(I,JPOLE,K)*CL
      VPOLE = VPOLE + MSGN * U(I,JPOLE,K)*CL - V(I,JPOLE,K)*SL
  100 CONTINUE
C
      HTP(M,K) = HT(1,JPOLE,K)
       UP(M,K) =  UPOLE/IMAX
       VP(M,K) =  VPOLE/IMAX
C
      DO 101 I=1,IMAX
      SL = SIN((I-1)*DL)
      CL = COS((I-1)*DL)
       U(I,JPOLE,K)     = -UP(M,K)*SL + MSGN * VP(M,K)*CL
       V(I,JPOLE,K)     = -MSGN * UP(M,K)*CL - VP(M,K)*SL
      HT(I,JPOLE,K)     = HTP(M,K)
  101 CONTINUE
C
   90 CONTINUE
C
   80 CONTINUE
```

```
C
      RETURN
      END
      SUBROUTINE AVERAG (Q,IM,JM,QBAR,LNMI,LNMF)
      DIMENSION  Q(IM,JM), COSP(180)
      LOGICAL LNMI, LNMF
      DATA WR /9358.2996/
C
      PI = 4.*ATAN(1.)
      DL = 2.*PI/IM
      DP = PI/(JM-1)
      DO 20 J=1,JM
      PHI = (J-1)*DP - PI/2.
      COSP(J) = COS(PHI)
   20 CONTINUE
C
         QSMJ = 0.0
      DO 110 J=1,JM
         QSMI = 0.0
      DO 120 I=1,IM
         QSMI = QSMI + Q(I,J)
  120 CONTINUE
         QSMJ = QSMJ + QSMI*COSP(J)
  110 CONTINUE
         QBAR = QSMJ * DL*DP/(4.*PI)
C
      IF(.NOT.LNMI .AND. .NOT.LNMF) RETURN
      DO 130 J=1,JM
      DO 140 I=1,IM
      Q(I,J) = Q(I,J) - QBAR + WR
  140 CONTINUE
  130 CONTINUE
C
         QSMJ = 0.0
      DO 150 J=1,JM
         QSMI = 0.0
      DO 160 I=1,IM
         QSMI = QSMI + Q(I,J)
  160 CONTINUE
         QSMJ = QSMJ + QSMI*COSP(J)
      DLSTAR = DL*180/PI
      KTPG = KMAX + 1
C
      JNPM1 = JNP-1
      JNPM2 = JNP-2
      JSPP2 = JSP+2
      JSPP1 = JSP+1
C
C ************************************************************
C ****                MODEL CONSTANTS                     ****
C ************************************************************
```

```
C
      AINV = 1./A
      DL2  = 1./(2.0*DL*3.0)
      DP2  = 1./(2.0*DP*3.0)
      DL4  = 1./(4.0*DL*3.0)
      DP4  = 1./(4.0*DP*3.0)
      DL8  = 1./(8.0*DL*3.0)
      DP8  = 1./(8.0*DP*3.0)
      DL24 = 4.*DL2
      DP24 = 4.*DP2
      DL44 = 4.*DL4
      DP44 = 4.*DP4
      IDIM = IMAX
      JDIM = JMAX
      IMAXH = IMAX/2
      C11 =  4.* SIN( DP  )/(3.*A*IMAX*(1.0-COS(  DP )))
      C12 =      SIN(2.*DP)/(3.*A*IMAX*(1.0-COS(2.*DP)))
                    CORREC = -1
      IF(MATS.EQ.1) CORREC =  1
      RHO(1) = 1.0
      RHO(2) = KAPPA
      DO 60 J=JSP,JNP
      PHI = (J-JSP)*DP - PI/2.0
      COSP(J) = COS(PHI)
      SINP(J) = SIN(PHI)
      IF(J.NE.JSP .AND. J.NE.JNP)  CON(J)  = 1.0/(A*COSP(J))
      F(J)    = 2.0*OMEGA*SINP(J)
   60 CONTINUE
      DO 70 I=1,IMAX
      IMAXD2(I) = IMAXH - IMAX*( I/(IMAXH +1) )
      COSL(I)   = COS((I-1)*DL)
      SINL(I)   = SIN((I-1)*DL)
   70 CONTINUE
C
C ***********************************************************
C ****            READ INITIAL DATA                      ****
C ***********************************************************
C
      IF(LPRIME) CALL PRIMES
                 CALL UVHR (TSTART,LFORCE,LNMI)
      DO 55 K=1,KMAX
      CALL AVERAG (HT(1,1,K),IMAX,JMAX,HTBAR(K),LNMI,LNMF)
      CALL ZONAL  ( U(1,1,K),IMAX,JMAX,UBAR(1,K) )
   55 CONTINUE
C
C ***********************************************************
C ****              WRITE HEADER                         ****
C ***********************************************************
C
                           WRITE(6,50) TITLE
      IF(SCHEME.EQ.ZLEP)   WRITE(6,54)
      IF(SCHEME.EQ.ZMAT)   WRITE(6,53)
      IF(GFILT)            WRITE(6,51)
      IF(HFILT)            WRITE(6,52)
      IF(LNMF)             WRITE(6.56)
```

```
      IF(LNMI)              WRITE(6,59)  NMIMAX
                            WRITE(6,109) KMAX
                            WRITE(6,108) KAPPA
                            WRITE(6,105) IMAX,JMAX,DLSTAR,
     $                                   DPSTAR,DT,TPRT,TM,TO
C
   50 FORMAT(1X,//,1X,'FOURTH ORDER 2-LEVEL SHALLOW WATER MODEL'
     $          ,/ ,1X,20A4,/)
   51 FORMAT(1X,'GLOBAL SHAPIRO FILTER ')
   52 FORMAT(1X,'HIGH-LATITUDE FOURIER FILTER')
   53 FORMAT(1X,'MATSUNO TIME SCHEME')
   54 FORMAT(1X,'LEAPFROG TIME SCHEME')
   56 FORMAT(1X,'HIGH-LATITUDE NORMAL MODE FILTER')
   77 FORMAT(/1X,'LEVEL ',I2,' SCALE HEIGHT =',F13.4/)
   59 FORMAT(1X,'NON-LINEAR NORMAL MODE INITIALIZATION:  ',I2,1X,
     $          'ITERATIONS')
C
      DO 76 K=1,KMAX
      WRITE(6,77) K,HTBAR(K)
   76 CONTINUE
C
C ***********************************************************
C ****       PERFORM NORMAL MODE INITIALIZATION          ****
C ***********************************************************
C
      IF(.NOT.LNMI) GOTO 57
C
      DO 58 K=1,KMAX
      CALL NORMAL ( U(1,1,K),V(1,1,K),HT(1,1,K),
     $              HTBAR(K),IMAX,JMAX,DT,LNMI,LNMF,LRASCH )
   58 CONTINUE
C
   57 CONTINUE
C
C ***********************************************************
C ****             TIME-SETP INITIALIZATION              ****
C ***********************************************************
C
      TAVE  = 0.0
      T     = TSTART
      NCALL = 1
      NTOT  = 1
      IPN   = 1
      NEXT  = 1
C
      NSTEP  = 0
      NMICNT = 0
C
      CALL RFFTI  (IMAX,STOFFT)
C
C ***********************************************************
C ****       CALCULATE DAMPING COEFFICIENTS              ****
C ****           FOR FOURIER FILTERING                   ****
C ***********************************************************
```

```
C
      CALL COEFF (HTBAR)
      CALL CLOCK
C
C ***********************************************************
C ****            SET INITIAL TIME INDEX                 ****
C ***********************************************************
C
   14 CONTINUE
      NM1 = 1
      N   = 1
      NP1 = 2
  150 CONTINUE
         QBAR = QSMJ * DL*DP/(4.*PI)
C
      RETURN
      END

R; T=2.04/4.45 19:39:35
```

BIBLIOGRAPHIC DATA SHEET

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86227		

4. Title and Subtitle	5. Report Date
Documentation of the Goddard Laboratory for Atmospheres Fourth-Order Two-Layer Shallow Water Model	FEBRUARY 1986
	6. Performing Organization Code 611
7. Author(s) Lawrence L. Takacs, Compiler	8. Performing Organization Report No. 85B0505
9. Performing Organization Name and Address Goddard Modelling and Simulation Branch Goddard Space Flight Center Greenbelt, Maryland 20771	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

Lawrence L. Takacs: Sigma Data Services Corporation.

16. Abstract

This documentation describes the theory and numerical treatment used in the 2-level GLA fourth-order shallow water model. This model was designed to emulate the horizontal finite-differences used by the GLA Fourth-Order General Circulation Model (Kalnay et al., 1983) in addition to its grid structure, form of high-latitude and global filtering, and time-integration schemes. A user's guide is also provided instructing the user on how to create initial conditions, execute the model, and post-process the data history.

17. Key Words (Selected by Author(s))	18. Distribution Statement
shallow water model, global filtering, 2-layer, documentation, time-integration schemes, user's guide, high-latitude filtering, horizontal finite-differences	unclassified - unlimited subject category 47

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
unclassified	unclassified	84	A05

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1986