NASA Technical Memorandum 104606, Vol. 2

Technical Report Series on Global Modeling and Data Assimilation

Max J. Suarez, Editor
Goddard Space Flight Center
Greenbelt, Maryland

Volume 2

Direct Solution of the Implicit Formulation of Fourth Order Horizontal Diffusion for Gridpoint Models on the Sphere

Yong Li
General Sciences Corporation
Laurel, Maryland

S. Moorthi
National Meteorological Center
Camp Springs, Maryland

J. Ray Bates
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland
1994

Abstract

High order horizontal diffusion of the form $K\nabla^{2m}$ is widely used in spectral models as a means of preventing energy accumulation at the shortest resolved scales. In the spectral context, an implicit formulation of such diffusion is trivial to implement. The present note describes an efficient method of implementing implicit high order diffusion in global finite difference models. The method expresses the high order diffusion equation as a sequence of equations involving ∇^2. The solution is obtained by combining fast Fourier transforms in longitude with a finite difference solver for the second order ordinary differential equation in latitude.

The implicit diffusion routine is suitable for use in any finite difference global model that uses a regular latitude/longitude grid. The absence of a restriction on the timestep makes it particularly suitable for use in semi-Lagrangian models. The scale selectivity of the high order diffusion gives it an advantage over the uncentering method that has been used to control computational noise in two-time-level semi-Lagrangian models.

Contents

List of Figures

1 Introduction

Diffusion and filtering have been common features of numerical models since the beginning of numerical weather prediction (Shuman, 1957). The reason for this is two-fold. First, diffusion and filtering can serve as crude parameterizations of subgrid scale processes that are not represented by the model. In the real atmospheric flow, there is a cascade of energy to smaller and smaller scales due to nonlinear interactions, until the motion is dissipated by friction. But in a model, even with the largest computer available, only scales far larger than the inertial subrange of turbulence can be resolved. In other words, what needs to be parameterized is much more than molecular diffusion. Secondly, models are usually subject to undesirable high wavenumber noise due, for example, to numerical discretization procedures. Diffusion and filtering can serve to reduce or eliminate this kind of noise.

Explicit formulations of diffusion equations are only conditionally stable (Mesinger and Arakawa, 1976). As a consequence, the coefficient of diffusion cannot be increased beyond a critical value that depends on gridlength and timestep. This imposes a stringent restriction in application to atmospheric models, especially those such as semi-Lagrangian models that allow the use of large timesteps: the allowed diffusion coefficient is too small to eliminate the high wavenumber noise. Such a restriction is particularly marked in global models using a uniform latitude/longitude grid where the convergence of the meridians leads to a grid size much smaller at high than at low latitudes. A diffusion coefficient that is barely large enough to ensure smooth solutions at low latitudes can far exceed the critical value for computational stability at high latitudes. This restriction becomes even more marked if a diffusion equation of higher order is used.

An alternative to diffusion are spatial filters; see Raymond and Garder (1991) for a review. The most commonly used in meteorological models is the Shapiro filter (Shapiro, 1971), which has the advantages of being very simple in formulation and of being easily extendible to higher orders. The filter is designed in a manner analogous to that used in solving the explicit formulation of a diffusion equation of equivalent order with the computational stability criterion satisfied. When large timesteps are used, the implied diffusion coefficient is so small that high wavenumber noise cannot be effectively reduced. In addition, the preferred spherical formulation of the filter, which involves simply averaging among the surrounding points, does not preserve area-weighted mean values. This implies violation of global conservation of the quantity being filtered. All attempts at using the filter in the context of spherical geometry have been shown to lead to degraded performance at high latitudes (Shapiro, 1979).

In some semi-Lagrangian models, an uncentered discretization has been employed to damp computational noise (Tanguay, *et al.*, 1992; Bates, *et al.*, 1993; Gravel, *et al.*, 1993). This, however, can reduce the accuracy of those terms discretized semi-implicitly (the second order accuracy is decreased to first order). Also, being insufficiently scale selective, the uncentering can have a negative impact on the large scale features in both amplitude and

phase, as has been recognized in our experience and also by Semazzi and Dekker (1994).

Implicit high order horizontal diffusion is trivial to implement in spectral models since the model variables are expressed in spectral harmonics which are eigenfunctions of the spherical Laplace operator. In this context, implicit diffusion is widely used (e.g., Williamson and Olson, 1994; Ritchie, *et al.*, 1994). The need for an implicit formulation of horizontal diffusion in gridpoint models has also been recognized, but the solution is much less straightforward. Jakimow *et al.* (1992) presented an implicit formulation for second order horizontal diffusion, and showed that its application to a semi-Lagrangian hemispheric model gave positive results. It was found that the optimum value for the diffusion coefficient exceeds the value allowed by the stability criterion of an explicit formulation. McDonald (1994) has developed an implicit solver for an approximated form of ∇^{2m} diffusion for limited area finite difference models, and has found the diffusion to be effective in reducing noise in the semi-Lagrangian context. Our experience with the NASA/GLA Semi-Lagrangian GCM (Bates *et al.*, 1993; Moorthi *et al.*, 1994) indicates that at moderate spatial resolution the intrinsic smoothing associated with the interpolations, along with a small uncentering, can prevent the growth of high wavenumber noise. But in high resolution integrations (e.g., $1^0 \times 1.25^0$ in the horizontal with 46 vertical layers), computational noise is present even with increased uncentering, especially at high latitudes and near orography (Moorthi, 1994). To alleviate this problem, the present implicit formulation of fourth order horizontal diffusion for a regular latitude/longitude grid on the sphere has been developed. Some justification for the use of fourth order rather than second order diffusion is also presented.

A detailed description of the solver is presented in Section 2. The justification for using fourth order diffusion is discussed in Section 3. Results of applying the solver to a global shallow water semi-Lagrangian model are given in Section 4. Concluding remarks are presented in Section 5. The Fortran program of the solver is attached to the note as an Appendix.

2 Description of The Solver

2.1 Equations

The fourth order horizontal diffusion equation takes the form

$$\frac{\partial \phi}{\partial t} = -K\nabla^4 \phi \tag{1}$$

and can be discretized as

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = -K\nabla^4 \left((1-\xi)\,\phi^{n+1} + \xi \phi^n \right) \tag{2}$$

where ξ is a parameter whose value is determined by the type of temporal discretization used; it is semi-implicit if ξ is $\frac{1}{2}$ and implicit if ξ is 0. (This terminology is adopted for consistency with the meteorological literature; the terms Crank-Nicholson and Euler implicit are corresponding descriptions used elsewhere.)

Integrating (2) involves the solution of the following equation, in a manner similar to that used by Lindzen and Kuo (1969):

$$\nabla^4\phi^{n+1} + \frac{1}{K\Delta t\,(1-\xi)}\phi^{n+1} = \frac{1}{K\Delta t\,(1-\xi)}\phi^n - \frac{\xi}{(1-\xi)}\nabla^4\phi^n \tag{3}$$

i. e.,

$$\nabla^4\phi + r\phi = f \tag{4}$$

where r is a prescribed parameter, f is the forcing term and the superscript $(n+1)$ has been omitted. Numerical solution of (4) on the sphere requires a conservative discretization of the ∇^4 operator, with a consistent treatment of the polar boundary conditions. Difficulties associated with these as well as with the cross derivative terms can be avoided by writing the equation in the form of two second order equations:

$$\nabla^2 Z + r\phi = f \tag{5}$$

$$\nabla^2\phi - Z = 0 \tag{6}$$

where

$$\nabla^2 = \frac{1}{a^2\cos^2\theta}\frac{\partial^2}{\partial\lambda^2} + \frac{1}{a^2\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial}{\partial\theta}\right)$$

2.2 Longitudinal and Latitudinal Discretization

Since the coefficients in the above equations are independent of longitude, we can perform Fourier transforms using the Fast Fourier Transform (FFT) algorithm in the $\lambda-$direction:

$$\begin{pmatrix} Z \\ \phi \\ f \end{pmatrix} = \sum \begin{pmatrix} \hat{Z} \\ \hat{\phi} \\ \hat{f} \end{pmatrix} e^{ik\lambda}. \tag{7}$$

Then the partial differential equations (5) and (6) lead to the ordinary differential equations (ODE) in vector form:

$$L\vec{X} + Q\vec{X} = \vec{F} \tag{8}$$

where

$$\vec{X} = \begin{pmatrix} \hat{z} \\ \hat{\phi} \end{pmatrix} \quad \vec{F} = \begin{pmatrix} \hat{f} \\ 0 \end{pmatrix} \quad Q = \begin{pmatrix} -q & r \\ -1 & -q \end{pmatrix};$$

$$L = \frac{1}{a^2\cos\theta}\frac{d}{d\theta}\left(\cos\theta\frac{d}{d\theta}\right)$$

$$q = \frac{k^2}{a^2 \cos^2 \theta}.$$

The discretized forms of L and q at the interior points are:

$$(Ly)_j = u_j y_{j+1} + v_j y_j + w_j y_{j-1}$$

$$q_j = \frac{\sin^2\left(\frac{k\Delta\lambda}{2}\right)}{\left(\frac{\Delta\lambda}{2}\right)^2 a^2 \cos^2 \theta_j}.$$

Here, y is any variable operated on by L and

$$u_j = \frac{\cos\theta_{j+\frac{1}{2}}}{a^2 \cos\theta_j \left(\Delta\theta\right)^2} \quad v_j = -\frac{\cos\theta_{j+\frac{1}{2}} + \cos\theta_{j-\frac{1}{2}}}{a^2 \cos\theta_j \left(\Delta\theta\right)^2} \quad w_j = \frac{\cos\theta_{j-\frac{1}{2}}}{a^2 \cos\theta_j \left(\Delta\theta\right)^2} \tag{9}$$

for $j = 1, 2, 3, \ldots, N-1$ ($j = 0$ and $j = N$ represent the south and the north poles, respectively). We denote

$$A_j = w_j I \qquad B_j = v_j I + Q_j \qquad C_j = u_j I \tag{10}$$

where I is the unit diagonal matrix. Thus, the discretized equation for the interior points, $j = 1, 2, 3, \ldots, N-1$, can be written as:

$$A_j \vec{X}_{j-1} + B_j \vec{X}_j + C_j \vec{X}_{j+1} = \vec{F}_j. \tag{11}$$

2.3 Boundary Conditions at The Poles

The boundary conditions at the south and the north poles are defined as:

$$A_0 \vec{X}_0 + B_0 \vec{X}_1 = \vec{F}_0 \tag{12}$$

$$A_N \vec{X}_{N-1} + B_N \vec{X}_N = \vec{F}_N. \tag{13}$$

At each pole, there should be a single unique value for the solution Z and ϕ. For all longitudinal asymmetric components ($k > 0$), the boundary conditions for the Fourier components $\vec{X}$ at the poles simply reduce to:

$$\vec{X}_0 = 0$$

$$\vec{X}_N = 0.$$

The Fourier component ($k = 0$) is the zonal mean, whose solution is derived from the ODE for it. The boundary conditions at the poles can be determined simply by discretizing (5) and (6) at the poles. We arrive at

$$\begin{pmatrix} -\left(\frac{2}{a\Delta\theta}\right)^2 & r \\ -1 & -\left(\frac{2}{a\Delta\theta}\right)^2 \end{pmatrix} \begin{pmatrix} \bar{z}(0) \\ \bar{\phi}(0) \end{pmatrix} + \begin{pmatrix} \left(\frac{2}{a\Delta\theta}\right)^2 & 0 \\ 0 & \left(\frac{2}{a\Delta\theta}\right)^2 \end{pmatrix} \begin{pmatrix} \bar{z}(1) \\ \bar{\phi}(1) \end{pmatrix} = \begin{pmatrix} \bar{f}(0) \\ 0 \end{pmatrix} \quad (14)$$

for the south pole, and similarly

$$\begin{pmatrix} -\left(\frac{2}{a\Delta\theta}\right)^2 & r \\ -1 & -\left(\frac{2}{a\Delta\theta}\right)^2 \end{pmatrix} \begin{pmatrix} \bar{z}(N) \\ \bar{\phi}(N) \end{pmatrix} + \begin{pmatrix} \left(\frac{2}{a\Delta\theta}\right)^2 & 0 \\ 0 & \left(\frac{2}{a\Delta\theta}\right)^2 \end{pmatrix} \begin{pmatrix} \bar{z}(N-1) \\ \bar{\phi}(N-1) \end{pmatrix} = \begin{pmatrix} \bar{f}(N) \\ 0 \end{pmatrix} \quad (15)$$

for the north pole. The overbar denotes zonal average. These correspond to the boundary conditions of the form in (12) and (13) with the following coefficient matrices:

$$A_0 = \begin{pmatrix} -\left(\frac{2}{\Delta\theta}\right)^2 & r \\ -1 & -\left(\frac{2}{\Delta\theta}\right)^2 \end{pmatrix}; \quad (16)$$

$$B_0 = \begin{pmatrix} \left(\frac{2}{\Delta\theta}\right)^2 & 0 \\ 0 & \left(\frac{2}{\Delta\theta}\right)^2 \end{pmatrix}; \quad (17)$$

$$\vec{F}_0 = \begin{pmatrix} \bar{f}(0) \\ 0 \end{pmatrix}; \quad (18)$$

$$A_N = \begin{pmatrix} \left(\frac{2}{\Delta\theta}\right)^2 & 0 \\ 0 & \left(\frac{2}{\Delta\theta}\right)^2 \end{pmatrix}; \quad (19)$$

$$B_N = \begin{pmatrix} -\left(\frac{2}{\Delta\theta}\right)^2 & r \\ -1 & -\left(\frac{2}{\Delta\theta}\right)^2 \end{pmatrix}; \quad (20)$$

$$\vec{F}_N = \begin{pmatrix} \bar{f}(N) \\ 0 \end{pmatrix}. \quad (21)$$

2.4 Numerical Solution

We again adopt the method of Lindzen and Kuo (1969) to solve the system (11) coupled with (12) and (13). Assume that the solution of (11) takes the following form:

$$\vec{X}_j = \alpha_j \vec{X}_{j+1} + \beta_j \quad (22)$$

where α_j is a 2×2 matrix and β_j is a vector. We know from (12) that

$$\alpha_0 = -{A_0}^{-1} B_0 \tag{23}$$

$$\beta_0 = {A_0}^{-1} \vec{F}_0. \tag{24}$$

If we substitute

$$\vec{X}_{j-1} = \alpha_{j-1} \vec{X}_j + \beta_{j-1}$$

into (11), we have

$$\vec{X}_j = -\left(A_j \alpha_{j-1} + B_j\right)^{-1} C_j \vec{X}_{j+1} + \left(A_j \alpha_{j-1} + B_j\right)^{-1} \left(\vec{F}_j - A_j \beta_{j-1}\right). \tag{25}$$

Comparing (25) and (22) leads to:

$$\alpha_j = -\left(A_j \alpha_{j-1} + B_j\right)^{-1} C_j \tag{26}$$

$$\beta_j = \left(A_j \alpha_{j-1} + B_j\right)^{-1} \left(\vec{F}_j - A_j \beta_{j-1}\right) \tag{27}$$

for $j = 1, 2, 3, \ldots, N-1$. With the following two equations:

$$A_N \vec{X}_{N-1} + B_N \vec{X}_N = \vec{F}_N \tag{28}$$

$$\vec{X}_{N-1} = \alpha_{N-1} \vec{X}_N + \beta_{N-1} \tag{29}$$

we can obtain the solution for the north pole as:

$$\vec{X}_N = \left(A_N \alpha_{N-1} + B_N\right)^{-1} \left(\vec{F}_N - A_N \beta_{N-1}\right). \tag{30}$$

Now we recursively use (22) to obtain the complete set of Fourier components for each $j = N-1, N-2, \ldots, 2, 1, 0$. Finally, we use inverse FFT to obtain the solution at grid points.

2.5 Coding Strategy

The coding strategy used here is similar to that of Moorthi and Higgins (1992, 1993). The FFT has been made very efficient by using the CRAY subroutine RFFTMLT to transform (both forward and backward) all latitudes simultaneously. In addition, the code for solving the second order O.D.E's for the FFT components and the zonal mean is vectorized over the wave numbers. After vectorization, a speed-up by a factor of 40 was observed on the CRAY C-90 computer. The Fortran program with comprehensive documentation is supplied in the Appendix.

2.6 Extension to Higher Orders

Extending the solver presented above to higher order is straightforward, both in formulation and in coding. Suppose we are to solve a m^{th} order equation (m being an even number). In the place of (2.5) and (2.6), we now have a set of $m/2$ second order equations. The FFT, latitudinal discretization and numerical procedures are almost the same. The matrix size of 2x2 then becomes $m/2 \times m/2$. As long as m is not too large, say, not larger than 8, the computational cost is not expected to increase dramatically.

3 Justification for Using Fourth Order Diffusion

It is well known that higher order diffusion can effectively damp the high wavenumber noise but leave the low wavenumbers very little affected. Here we present a discussion of the damping properties of the second and fourth order diffusion equations, in the context of both implicit and semi-implicit formulations. Though these properties have been discussed in earlier works (e.g., Mesinger and Arakawa, 1976), we present these here for the convenience of reference. We use the 1-D diffusion equation for illustration and then discuss the implication for spherical applications.

The second order diffusion equation in its discretized form can be written as:

$$\frac{\phi_I^{n+1} - \phi_I^n}{\Delta t} = \mu \left((1-\xi) \left(\delta_{xx} \phi^{n+1} \right)_I + \xi \left(\delta_{xx} \phi^n \right)_I \right) \tag{31}$$

where $\xi = 0$ corresponds to the implicit scheme and $\xi = \frac{1}{2}$ the semi-implicit scheme. The notation δ_{xx} is the finite difference approximation of the second order derivative, which is:

$$\delta_{xx} A = \frac{A_{I+1} + A_{I-1} - 2.0 A_I}{(\Delta x)^2}.$$

Assume the solution takes the following form:

$$\phi_I^n = \Phi_0 \Lambda^n e^{ikI\Delta x}.$$

With second order centered difference for the spatial discretization, the damping rate, Λ_2, becomes:

$$\Lambda_2 = \frac{1 - 4\xi r_2 \sin^2\left(\frac{\pi}{m}\right)}{1 + 4\left(1-\xi\right) r_2 \sin^2\left(\frac{\pi}{m}\right)}. \tag{32}$$

where $r_2 = \frac{\mu \Delta t}{(\Delta x)^2}$ and m $(= \frac{2\pi}{k\Delta x})$ is the wave number measured by grid intervals.

The fourth order diffusion equation in its discretized form is

$$\frac{\phi_I^{n+1} - \phi_I^n}{\Delta t} = -K \left((1-\xi) \left(\delta_{xxxx} \phi^{n+1} \right)_I + \xi \left(\delta_{xxxx} \phi^n \right)_I \right) \tag{33}$$

where

$$\delta_{xxxx}A = \frac{(\delta_{xx}A)_{I+1} + (\delta_{xx}A)_{I-1} - 2.0\,(\delta_{xx}A)_I}{(\Delta x)^2}$$

is the finite difference approximation of the fourth order operator. The damping rate, Λ_4, is

$$\Lambda_4 = \frac{1 - 16\xi\, r_4 \sin^4\left(\frac{\pi}{m}\right)}{1 + 16\,(1-\xi)\, r_4 \sin^4\left(\frac{\pi}{m}\right)}. \tag{34}$$

where $r_4 = \frac{K\Delta t}{(\Delta x)^4}$.

From (32) and (34) it is clear that for both second order and fourth order diffusion, the fully implicit treatment ($\xi = 0$) is unconditionally stable. When $0 < \xi < 1$, there is a critical value for K above which there is a reversal of the sign of the damping rate. When $\xi > \frac{1}{2}$, there is also a critical value of K above which the scheme becomes unstable, this critical value being the smallest when the scheme is fully explicit (i.e., $\xi = 1$). When ξ is nonzero, it is possible to choose a value of K such that $2\Delta x$ waves are completely eliminated.

Figure 1 shows the damping rates as a function of wave number for the second and the fourth order diffusion equations in both implicit and semi-implicit formulations. The parameters r_2 and r_4 are so chosen that the $2\Delta x$ waves are completely eliminated with the semi-implicit formulations. Three features are observed: 1) In all cases, $|\Lambda| < 1$, which indicates unconditional stability; 2) The fourth-order equation is much more scale selective than the second order one; while the second order diffusion damps the $20\Delta x$ waves by about 5%, the fourth order leaves it almost unchanged; 3) The semi-implicit scheme can completely eliminate the $2\Delta x$ waves but the implicit scheme damps it only by a factor.

The above results might mislead one to believe that the semi-implicit scheme is better than the implicit one since the former has the potential of eliminating the $2\Delta x$ waves completely with an appropriate diffusion coefficient. But for our case on the sphere with a uniform latitude/longitude grid, it may well be the situation that K is too large near the poles but too small near the equator. When K is very large, the $2\Delta x$ waves just change sign instead of being smoothed (see (34)).

Figure 2a is the geopotential field of the initial conditon described in McDonald and Bates (1989), consisting of spherical harmonic wave number one. Figure 2b is the same field with $2\Delta x$ waves embedded. The result of applying the semi-implicit fourth order diffusion equation, with r_4 so chosen that the $2\Delta x$ waves at the equator are eliminated, is shown in Figure 2c. The noise away from the equator remains. Careful examination shows that sign changes occur. Figure 2d represents the result after applying the implicit diffusion with the same value for the diffusion coefficient. Now the noise near the poles is damped, but some noise near the equator remains. Figure 2e is obtained by applying the semi-implicit formulation followed by the implicit formulation. It is almost identical to the original field in Figure 2a. Therefore, the use of the semi-implicit formulation of the diffusion equation alone should be avoided. The implicit formulation is preferred. If some noise near the

equator needs to be damped, the semi-implicit formulation, in addition to the implicit one, can be used if the additional computational cost is not a concern.

4 Application to A Semi-Lagrangian Shallow Water Model

In this section, we present some results of applying the fourth order diffusion to the shallow water model based on the momentum formulation, termed the (u, v) model as in Bates *et al.* (1994). This is the 2-D version of the NASA/GLA semi-Lagrangian GCM (Bates *et al.*, 1993).

4.1 Initial Condition of McDonald and Bates (1989)

For a resolution of 512x257 on a uniform latitude/longitude grid and a timestep of one hour, a minimum value of 0.03 for the uncentering parameter is needed to suppress noise in the case of the geostrophically balanced initial condition of McDonald and Bates (1989). The initial geopotential is defined as:

$$\phi(\lambda, \theta) = \overline{\phi} + 2\Omega a v_0 \sin^3\theta \cos\theta \sin\lambda$$

where Ω and a are, respectively, the earth's rotation and radius. The values $\overline{\phi} = 5.7879 \times 10^4$ m^2/s and $v_0 = 20$ m/s are used. The geopotential at 50 days of the integration is presented in Figure 3a.

Now we replace the uncentering by the fourth order diffusion. In order to remove the residual noise near the equator, we apply the semi-implicit formulation in addition to the implicit one as discussed in the previous section. The diffusion coefficient for the semi-implicit formulation takes a value of 1.32×10^{15} m^4/s, which is just large enough to completely eliminate the $2\Delta x$ waves at the equator. The value needed for the implicit formulation is found to be dependent on the way the diffusion is applied. If the diffusion is applied to the geopotential only, a minimum value of 9.25×10^{15} m^4/s is needed. The geopotential at 50 days of the integration thus obtained is shown in Figure 3b. If the diffusion is applied only to the divergence, the minimum value needed is 11.825×10^{15} m^4/s. The geopotential at 50 days of the corresponding integration is shown in Figure 3c. Remember that the divergence is not a basic prognostic model variable, rather it is an intermediate variable. This may be the reason why a relatively larger minimum value is needed. If we apply the diffusion to both the divergence and the geopotential, the minimum value for the diffusion coefficient becomes 5.28×10^{15} m^4/s, which is smaller than the value used in either of the previous two cases. The geopotential at 50 days of the integration is now presented in Figure 3d.

The results shown in Figures 3b-d, which were obtained by applying the diffusion in various ways, demonstrate less damping of the long waves than is the case when using uncentering

(Figure 3a). Of the three cases (diffusion applied to geopotential alone, to divergence alone, and to both geopotential and divergence), it seems that the last two cases have the least undesirable damping effects. In addition, the results in Figures 3b-d all demonstrate the same phase speed, but the predicted geopotential of the uncentering case shows faster westward propagation of the long wave. It has been observed in our numerical results (not shown) that the westward phase speed monotonically increases as the uncentering parameter increases. This clearly demonstrates the negative impact of the uncentering on the long waves in both amplitude and phase. Improvement is obtained by replacing it by the fourth order diffusion.

Examination of the evolution of the global mean geopotential indicates that the fourth order diffusion, regardless of the magnitude of the diffusion coefficient, has almost no impact on global mass conservation.

4.2 Real Data Case

The ECMWF analysis of 0000 UTC, 1 December 1992, is chosen. Again, the horizontal resolution is 512x257 points on a uniform latitude/longitude grid, but the timestep is chosen to be 30 minutes. The data are initialized using a digital filter (Lynch and Huang, 1992), with a cutoff period of 6 hours. The initialized geopotential is shown in Figure 4a.

Without any diffusion, a value of 0.05 is needed for the uncentering parameter. The geopotential at day 3 of the integration is shown in Figure 4b. When we apply the fourth order diffusion to both geopotential and divergence, the model can be integrated stably without uncentering. The diffusion coefficient for the semi-implicit formulation is $1.32 \times 10^{15}\ m^4/s$, and that for the implicit formulation is $3.96 \times 10^{15}\ m^4/s$. However, there is still some residual noise near the equator, which can be eliminated most effectively by using a small value of uncentering parameter (0.015 in this case). Figure 4c is the geopotential at day 3 of the integration thus obtained. The improvement of the representation of the resolved features in Figure 4c over Figure 4b is recognizable (The high in the upper left part of the model domain is stronger in Figure 4c). The advantages of using the fourth order diffusion should become more obvious when orography is present, in which case a much larger value for the uncentering parameter (undesirable) is otherwise needed to control the high wavenumber noise in mountainous regions.

5 Application to a 3-D Semi-Lagrangian GCM

We have incorporated the fourth-order horizontal diffusion into the NASA/GLA semi-Lagrangian semi-implicit (SLSI) general circulation model(GCM) with full physics. The adiabatic part of this GCM is described in Bates *et al.* (1993) and the version with full

physics is described in Moorthi *et al.* (1994). Here, we present some results to illustrate the use of the fourth-order diffusion.

The version of the model used here has twenty sigma layers in the vertical and a horizontal resolution of 2 degree latitude by 2.5 degree longitude. The initial state for the experiments is taken from the NASA/GLA Data Assimilation Office analyses for November 1, 1992, performed during the 1992 fall SPADE mission, using the GLA Eulerian GCM. The fourth order diffusion in its implicit formulation is applied to both temperature and moisture. The diffusion coefficient is taken as $K = min\left(\frac{0.25\times10^{15}}{\sigma_l}, 5.0 \times 10^{15}\right)$. Here σ_l is the value of the vertical coordinate sigma at the middle of the l^{th} layer. This choice is made particularly due to the fact that there exists stronger undesired noise in the model integration above the 100 hPa level than at lower levels. The associated e-folding time for the shortest resolvable waves at the equator is approximately 2 days near the bottom and 0.1 day near the top.

We have performed four ten-day runs, two without diffusion but with values of 0.05 and 0.07 for the uncentering parameter, and two with diffusion and with a value of 0.05 for the uncentering parameter. Both runs without diffusion show significant noise at the upper level of the model. An example is shown in Figure 5, where the temperature field at 50 hPa level is shown for the forecasts at days 3, 5, 7, and 10 respectively for the run with the uncentering parameter value of 0.05. It can be seen that the temperature field is noisy and the noise progressively increases near the equator as the integration proceeds. The corresponding field for the forecast with diffusion is shown in Figure 6. It can be seen that the inclusion of diffusion has effectively reduced the high wavenumber noise. Of course, one can obtain even better smoothing by increasing the value of the diffusion coefficient. The results for the uncentering parameter value 0.07 are similar to those shown above. Because the noise in the upper level equatorial region (which may be the result of the gravity waves generated by convective processes), cannot be efficiently eliminated through judicial choice of the uncentering parameter, inclusion of a higher order diffusion becomes very important.

6 Concluding Remarks

A direct solver for implicit fourth order horizontal diffusion on the sphere has been developed. It has been made efficient by vectorizing over the FFT wave spectrum. Extension of the formulation to the solution of even higher order diffusion is straightforward, and the additional coding effort needed is trivial.

Applying the solver to a global shallow water semi-Lagrangian model leads to positive results. It gives a more accurate representation of larger scales in both amplitude and phase than does the use of the uncentering method of damping. It is found that horizontal diffusion has almost no impact on the global mass conservation. The solver is also applied to the 3-D NASA/GLA semi-Lagrangian semi-implicit GCM and found to be helpful in

controlling the high wavenumber noise. The solver takes approximately 4% of the total CPU time on a single processor of the CRAY C98 computer when applied to the temperature field in the adiabatic version of the 3-D GCM.

Acknowledgements. This research was supported by the NASA Global Atmospheric Modeling and Analysis Program under Grant 578-41-16-20

APPENDIX A

Fortran Program of The Solver

```
      SUBROUTINE DIRSOL4(IMA,IM,JM,AE,INIT1,INIT2,DIFF,FI,FO,XK,DT,IMPL)
C*****************************************************************************
C                          Subroutine DIRSOL4
C*****************************************************************************
C WRITTEN BY:  Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE:  This routine solves the fourth-order diffusion equation on the
C sphere implicitly or semi-implicitly (Crank-Nicolson scheme), that is,
C to solve the following equation:
C
C         (\partial phi over \partial t) = - K \delta^4 (phi)
C
C where K is the diffusion coefficient.
C
C Implicit Soluton: (phi^(n+1)-phi^(n))/DT = - K \del^4 (phi^(n+1))
C
C Semi-Implicit:    (phi^(n+1)-phi^(n))/DT =
C                                - K (1/2) \del^4 (phi^(n+1)+phi^(n))
C
C     Each of the above discrete equation converts to the solution
C of a linear fourth-order partial differential equation on the shpere
C of the following form:
C
C               \delta^4 (phi)^(n+1) - r*phi^(n+1) = f
C
C The strategy of solution is similar to that in Lindzen and Kuo (1969,
C Mon. Wea. Rev., p732-734), which is summarized as:
C
C     1). Convert the \del^4-equation into a set of two \del^2 equations.
C         Now the model variables become a vector of
C         (\del^2(phi)^(n+1), (phi)^(n+1));
C
C     2). Since the coefficients are only latitude-dependent, FFT along the
C         latitudes are performed to transform the second-order vector PDE
C         to a series of second-order vector ODE's for the FFT coefficients;
C
C     3). Solve the set of second-order ODE's;
C
C     4). Perform inverse FFT to obtain the solution (phi)^(n+1) on gridpoints.
C
C
C CALLED FROM: Any routine which applies this smoother.
C
C SYSTEM ROUTINES USED:
C          FFTFAX: Cray routine which initializes the FFT routines;
C          RFFTMLT: Cray FFT routine (Cray Manual: SR-2081 8.0)
C
C SUBROUTINES CALLED:
C     CAL_R: calculate the coefficient r from K and and DT;
C     FORCE: calculate the forcing term f for the \del^4 equation;
C     S2ODE: solve the set of the second-order ODE's for FFT
C            coefficients k > 0;
C     SOMEAN:solve for the zonal mean;
C     WRAPD: wrap ghost points along the latitudes.
C
C VARIABLES:
C
C     INPUT:
C         AE-radius of earth (meters);
C         DIFF-logical variable meaning discrete FFT when it is .TRUE.;
C         DT-time step (seconds)
C         FI-input field before the smoothing (phi)^(n);
C         IM-number of grid points along latitudes;
C         IMA-dimension in X-direction (=IM + # of ghost points);
C         IMPL-fully implicit scheme (=1) and semi-implicit scheme (=0);
C         INIT1-logical variable (need to be TRUE);
C         INIT2-logical variable (need to be TRUE);
C         XK-diffusion coefficient K (m^4/s).
```

```
C     OUPUT:
C         FO-output field after smoothing (phi)^(n+1).
C
C     LOCAL:
C         A-array storing the FFT coefficinets;
C         AM-coefficient array for the 2nd-order ODE (for the zonal mean)
C             at j-1 point;
C         AMK-coefficient array for the 2nd-order ODE for the FFT coefficients
C             at j-1 point;
C         B-working array for the FFT;
C         BM-coefficient array for the 2nd-order ODE (for the zonal mean)
C             at j point;
C         BMK-coefficient array for the 2nd-order ODE for the FFT coefficients
C             at j point;
C         CM-coefficient array for the 2nd-order ODE (for the zonal mean)
C             at j+1 point;
C         CMK-coefficient array for the 2nd-order ODE for the FFT coefficients
C             at j+1 point;
C         CPH-COS(theta) where theta is the latitude;
C         DN1-real part of the forcing term for the 2nd-order ODE or forcing
C             for the equation for the zonal mean;
C         DN1K-real part of the forcing term for the 2nd-order ODE or forcing
C             to the equation for the FFT coefficients;
C         DN2K-imaginery part of the forcing term for the 2nd-order ODE or forcing
C             to the equation for the FFT coefficients;
C         F-array storing the forcing term for the \del^4 equation;
C         FM-zonal mean of the input field (phi)^(n);
C         FN1-the solution to the 2nd-order ODE for the zonal mean;
C         FN1K-real part of the solution to the 2nd-order ODE for the FFT
C             coefficients;
C         FN2K-imaginery part of the solution to the 2nd-order ODE for the FFT
C             coefficients;
C         IX-working array for the FFT routine;
C         TR-working array for the FFT routine.
C         U-coeffcients for intermediate calculation;
C         V-coeffcients for intermediate calculation;
C         W-coeffcients for intermediate calculation;
C         WVSQ-array storing coefficients for the X-direction derivative term
C              in the \del^2 equation;
C
C*************************************************************************
      INTEGER FORWARD, BACKWARD
      PARAMETER(FORWARD = -1, BACKWARD = 1)

      DIMENSION FI(IMA,JM), FO(IMA,JM)

      DIMENSION IX(IM), TR(3*(IM+2)+1), A(IM+2,JM-2), B(IM*2,JM-2)

      DIMENSION AM(2,2,JM), BM(2,2,JM), CM(2,2,JM), F(IMA,JM)
     #,         DN1(2,JM), DN2(2,JM), FN1(2,JM), FN2(2,JM)
     #,         FM(JM), CPH(JM,2), U(JM), V(JM), W(JM)
     #,         WV(IM/2), WVSQ(IM/2)

      DIMENSION AMK(IM/2,2,2,JM), BMK(IM/2,2,2,JM), CMK(IM/2,2,2,JM)
     #,          DN1K(IM/2,2,JM),  DN2K(IM/2,2,JM)
     #,          FN1K(IM/2,2,JM),  FN2K(IM/2,2,JM)


      LOGICAL INIT1, INIT2, DIFF, FIRST

      PI = ACOS(-1.0)
      TWOPI = 2.0 * PI
      PIO2 = PI/2.0
      IMB2 = IM/2
      IJM = (IMB2-1)*JM
      IJMM = IJM*22 + JM
      IJM2 = IJMM + (IM+2)*(JM-2)
```

```
CALL CAL_R(XK,DT,R,IMPL)
CALL FORCE(IMA,IM,JM,FI,F,AE,R,IMPL)
CALL WRAPD(IMA,IM,JM,F)

IF (INIT1) THEN
   CALL FFTFAX(IM,IX,TR)
   JMM1 = JM - 1
   JMM2 = JM - 2
   IMP1 = IM + 1
   IMP2 = IM + 2
   LEN = IMB2 - 1
   FIM = 1.0 / FLOAT(IM)
   DLM = TWOPI*FIM
   DPH = PI/FLOAT(JMM1)

   RDLM = 1.0/DLM
   RDPH = 1.0/DPH

   DY = AE*DPH
   AESQ = AE*AE

   DO J = 1, JM
      TEM = - PIO2 + (J-1)*DPH
      CPH(J,1) = COS(TEM+0.5*DPH)
      CPH(J,2) = COS(TEM)
   ENDDO
   CPH(1,2) = 0.0
   CPH(JM,2) = 0.0

   FIRST = .TRUE.
   INIT2 = .TRUE.
ENDIF

IF (INIT2) THEN
   DO J = 2, JMM1
      U(J) = CPH(J,1)/CPH(J,2)
      W(J) = CPH(J-1,1)/CPH(J,2)
      V(J) = - 1.0 * (U(J)+W(J))
   ENDDO

   IF (DIFF) THEN
      DO I = 1, IMB2
         WVSQ(I) = (SIN(0.5*I*DLM)*(RDLM*2.0))**2
      ENDDO
   ELSE
      DO I = 1, IMB2
         WVSQ(I) = I*I
      ENDDO
   ENDIF

ENDIF

A(:,:) = 0.0

DO J = 1, JM
   FM(J) = 0.0
   DO I = 1, IM
      FM(J) = FM(J) + F(I,J)
   ENDDO
ENDDO

DO J = 1, JM
   FM(J) = FM(J)*FIM
ENDDO

DO J = 2, JMM1
   DO I = 1, IM
      A(I,J-1) = F(I,J) - FM(J)
   ENDDO
```

```
      A(IM+1,J-1) = A(1,J-1)
      A(IM+2,J-1) = A(2,J-1)
   ENDDO

   CALL RFFTMLT(A,B,TR,IX,1,IMP2,IM,JMM2,FORWARD)

   A(1,:) = 0.0
   A(2,:) = 0.0
   A(IMP2,:) = 0.0


   DO J = 2, JMM1
      DO I = 1, IMB2
         AMK(I,1,1,J) = W(J)
         AMK(I,2,2,J) = W(J)
         AMK(I,1,2,J) = 0.0
         AMK(I,2,1,J) = 0.0

         CMK(I,1,1,J) = U(J)
         CMK(I,2,2,J) = U(J)
         CMK(I,1,2,J) = 0.0
         CMK(I,2,1,J) = 0.0

         BMK(I,1,1,J) = - (DY**2)*WVSQ(I)/(AE*CPH(J,2))**2
#                       + V(J)
         BMK(I,1,2,J) = R*(DY**2)
         BMK(I,2,1,J) = - DY**2
         BMK(I,2,2,J) = - (DY**2)*WVSQ(I)/(AE*CPH(J,2))**2
#                       + V(J)

         DN1K(I,1,J) = (DY**2)*A(I+I+1,J-1)
         DN1K(I,2,J) = 0.0

         DN2K(I,1,J) = (DY**2)*A(I+I+2,J-1)
         DN2K(I,2,J) = 0.0
      ENDDO
   ENDDO

   DN1K(:,:,1) = 0.0
   DN2K(:,:,1) = 0.0

   AMK(:,1,1,1) = 1.0
   AMK(:,2,2,1) = 1.0
   AMK(:,1,2,1) = 0.0
   AMK(:,2,1,1) = 0.0

   BMK(:,:,:,1) = 0.0

   DN1K(:,:,JM) = 0.0
   DN2K(:,:,JM) = 0.0

   AMK(:,:,:,JM) = 0.0

   BMK(:,1,1,JM) = 1.0
   BMK(:,2,2,JM) = 1.0
   BMK(:,1,2,JM) = 0.0
   BMK(:,2,1,JM) = 0.0

   CALL S2ODE(IMB2,DN1K,DN2K,FN1K,FN2K,AMK,BMK,CMK,JM)

   DO J = 2, JMM1
      DO I = 1, IMB2
         A(I+I+1, J-1) = FN1K(I,2,J)
         A(I+I+2, J-1) = FN2K(I,2,J)
      ENDDO
   ENDDO


   DO J = 2, JMM1
```

```
      AM(1,1,J) = W(J)
      AM(2,2,J) = W(J)
      AM(1,2,J) = 0.0
      AM(2,1,J) = 0.0

      CM(1,1,J) = U(J)
      CM(2,2,J) = U(J)
      CM(1,2,J) = 0.0
      CM(2,1,J) = 0.0

      BM(1,1,J) = + V(J)
      BM(1,2,J) = R*(DY**2)
      BM(2,1,J) = - DY**2
      BM(2,2,J) = BM(1,1,J)

      DN1(1,J) = (DY**2)*FM(J)
      DN1(2,J) = 0.0
   ENDDO

   DN1(1,1) = (DY**2)*FM(1)
   DN1(2,1) = 0.0

   AM(1,1,1) = - 4.0
   AM(1,2,1) = R*(DY**2)
   AM(2,1,1) = - (DY**2)
   AM(2,2,1) = - 4.0

   BM(1,1,1) = 4.0
   BM(1,2,1) = 0.0
   BM(2,1,1) = 0.0
   BM(2,2,1) = 4.0


   DN1(1,JM) = (DY**2)*FM(JM)
   DN1(2,JM) = 0.0

   AM(1,1,JM) = 4.0
   AM(1,2,JM) = 0.0
   AM(2,1,JM) = 0.0
   AM(2,2,JM) = 4.0

   BM(1,1,JM) = - 4.0
   BM(1,2,JM) = R*(DY**2)
   BM(2,1,JM) = - (DY**2)
   BM(2,2,JM) = - 4.0

   FN1(:,:) = 0.0

   CALL SOMEAN(DN1, FN1, AM, BM, CM, JM)

   CALL RFFTMLT(A,B,TR,IX,1,IMP2,IM,JMM2,BACKWARD)

   DO J = 1, JMM2
      DO I = 1, IM
         FO(I,J+1) = A(I,J) + FN1(2,J+1)
      ENDDO
   ENDDO

   DO I = 1, IM
      FO(I,1) = FN1(2,1)
      FO(I,JM) = FN1(2,JM)
   ENDDO

   CALL WRAP1(IMA,IM,JM,FO)

   RETURN
   END
   SUBROUTINE CAL_R(XK,DT,R,IMPL)
   AB = (1.0+FLOAT(IMPL))*XK*DT
```

```
      R = 2.0/AB
      RETURN
      END
      SUBROUTINE DEL2(IMA,IM,JM,A,DA,AE)
C*********************************************************************
C                Subroutine DEL2
C*********************************************************************
C WRITTEN BY: Yong Li, GSFC code 910.3, Apr 1994
C
C PURPOSE: calculate the \del^2(A) on the sphere.
C
C CALLED BY:
C       DEL4
C
C SUBROUTINES CALLED:     None
C
C VARIABLES:
C
C     INPUT:
C       A-input field;
C       AE-radius of earth;
C       IM-number of total grid points in X-direction;
C       IMA-dimension size in X-direction;
C       JM-number of total grid points in Y-direction;
C
C     OUTPUT:
C       DA-\del^2 of A
C*********************************************************************
      dimension a(ima,jm), da(ima,jm), dx(jm)
      dimension a1(500), a3(500), bp(500), bm(500)

      imp1 = im + 1
      imp2 = im + 2
      pi = acos(-1.0)
      twopi = 2.0*pi
      dlm = twopi/float(im)
      dph = pi/float(jm-1)
      dy = ae*dph

      a2=1.0d0/dy**2

      do j = 2, jm-1
         theta = - pi*0.5 + float(j-1)*dph
         bp(j) = cos(theta+0.5*dph)/cos(theta)
         bm(j) = cos(theta-0.5*dph)/cos(theta)
         dx(j) = ae*dlm*cos(theta)
         a1(j) = 1.0/dx(j)**2
         a3(j) = 2.0*a1(j)+a2*(bp(j)+bm(j))
      enddo

      do 20 i=2,imp1
        do 30 j=2,jm-1
        da(i,j)=a1(j)*(a(i+1,j)+a(i-1,j))
     #               +a2*(bp(j)*a(i,j+1)+bm(j)*a(i,j-1))
     #               -a3(j)*a(i,j)
 30      continue
 20   continue

      do j = 2, jm-1
         da(1,j) = da(im+1,j)
         do i = im+2, ima
            da(i,j) = da(i-im,j)
         enddo
      enddo

C
C          Laplace at the south pole
C
      AAC = 0.0
```

```
      do 40 i = 2, impl
         AAC = AAC + a(i,2)
 40   continue

      da(1,1) = (2.0/dy)**2*(AAC/float(im)-a(1,1))

      do 50 i = 2, ima
        da(i,1) = da(1,1)
 50   continue

C
C          Laplace at the north pole
C
      AAC = 0.0
      do 60 i = 2, impl
         AAC = AAC + a(i,jm-1)
 60   continue

      da(ima,jm) = (2.0/dy)**2*(AAC/float(im)-a(1,jm))

      do i = 1, ima-1
         da(i,jm) = da(ima,jm)
      enddo

      return
      end
      SUBROUTINE DEL4(IMA,IM,JM,A,DA,AE)
C**********************************************************************
C              Subroutine DEL4
C**********************************************************************
C WRITTEN BY: Yong Li, GSFC code 910.3, Apr 1994
C
C PUEPOSE: calculate the \del^4(A) by applying a conservative \del^2 twice.
C
C CALLED FROM:
C       FORCE: calculate the forcing term
C
C SUBROUTINES CALLED:
C       DEL2: \del^2 operator;
C       WRAPD: wrap up the ghost points in X-direction.
C
C VARIABLES:
C
C     INPUT:
C       A-input field;
C       AE-radius of earth;
C       IM-number of total grid points in X-direction;
C       IMA-dimension size in X-direction;
C       JM-number of total grid points in Y-direction.
C
C     OUTPUT:
C       DA-\del^4(A)
C**********************************************************************
      DIMENSION A(IMA,JM), DA(IMA,JM), B(IMA,JM)

      CALL DEL2(IMA,IM,JM,A,B,AE)
      CALL WRAPD(IMA,IM,JM,B)

      CALL DEL2(IMA,IM,JM,B,DA,AE)
      CALL WRAPD(IMA,IM,JM,DA)

      RETURN
      END
      SUBROUTINE FORCE(IMA,IM,JM,A,F,AE,R,IMPL)
C**************************************************************
C                Subroutine FORCE
C**************************************************************
C WRITTEN BY: Yong Li, GSFC code 910.3, Apr 1994
C
```

```
C PURPOSE: calculate the forcing term for the \del^4-equation.
C
C CALLED FROM: DIRSOL4
C
C SUBROUTINES CALLED:
C     DEL4: calculate the \del^4 operator of field A;
C     WRAPD: wrap up the ghost points in X-direction;
C
C VARIABLES:
C
C     INPUT:
C       A-array storing the field to be smoothed;
C       AE-radius of earth;
C       IM-number of total grid points in X-direction;
C       IMA-dimension size in X-direction;
C       IMPL-for implicit (=1), for semi-implicit (=0);
C       JM-number of total grid points in Y-direction;
C       R-coefficient for the zero-order term in the \del^4 equation;
C
C     OUTPUT:
C       DA-array storing \del^4(A);
C       F-array storing the forcing term for the \del^4 equation
C*************************************************************************
      DIMENSION A(IMA,JM), DA(IMA,JM), F(IMA,JM)

      IF(IMPL.EQ.1) THEN
         F(:,:) = R*A(:,:)
      ELSE
         CALL DEL4(IMA,IM,JM,A,DA,AE)
         CALL WRAPD(IMA,IM,JM,DA)
         F(:,:) = R*A(:,:) - DA(:,:)
      ENDIF

      RETURN
      END

      SUBROUTINE INVERSE1(A,B)
C*****************************************************************
C               Subroutine INVERSE1
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: Find the inverse of a 2x2 matrix
C
C VARIABLES:
C     INPUT:
C          A-input matrix to be inverted;
C     OUTPUT:
C          B-inverse of matrix A;
C*****************************************************************
      DIMENSION A(2,2), B(2,2)

      DD = A(1,1)*A(2,2)-A(1,2)*A(2,1)

      B(1,1) = A(2,2)/DD
      B(1,2) = - A(1,2)/DD
      B(2,1) = - A(2,1)/DD
      B(2,2) = A(1,1)/DD

      RETURN
      END
      SUBROUTINE INVERSE2(LEN,A,B)
C*****************************************************************
C               Subroutine INVERSE2
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
```

```
C PURPOSE: Find the inverse of a a number (LEN) of 2x2 matrix
C
C VARIABLES:
C     INPUT:
C          A-input of LEN 2x2 matrices to be inverted simultaneously;
C     OUTPUT:
C          B-inverse the LEN 2x2 matrices A;
C*****************************************************************

      Dimension A(LEN,2,2), B(LEN,2,2), DD(LEN)

      DO KK = 1, LEN
         DD(KK) = A(KK,1,1)*A(KK,2,2)-A(KK,1,2)*A(KK,2,1)
      ENDDO

      DO KK = 1, LEN
         B(KK,1,1) = A(KK,2,2)/DD(KK)
         B(KK,1,2) = - A(KK,1,2)/DD(KK)
         B(KK,2,1) = - A(KK,2,1)/DD(KK)
         B(KK,2,2) = A(KK,1,1)/DD(KK)
      ENDDO

      RETURN
      END
      SUBROUTINE MULTAB(A,B,C,N)
C*****************************************************************
C            Subroutine MULTAB
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: Multiplication of two NxN matrix C = A B
C
C VARIABLES:
C     INPUT:
C          A-input matrix;
C          B-input matrix;
C     OUTPUT:
C          C-output matrix.
C*****************************************************************
      DIMENSION A(N,N), B(N,N), C(N,N)
      DO I = 1, N
         DO J = 1, N
            C(I,J) = 0.0
            DO K = 1, N
               C(I,J) = C(I,J) + A(I,K)*B(K,J)
            ENDDO
         ENDDO
      ENDDO
      RETURN
      END
      SUBROUTINE MULTABK(LEN,A,B,C,N)
C*****************************************************************
C            Subroutine MULTABK
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: Multiplication of a series of two NxN matrix C = A B
C
C VARIABLES:
C     INPUT:
C          A-input of LEN 2x2 matrices;
C          B-input of LEN 2x2 matrices;
C     OUTPUT:
C          C-output matrix.
C*****************************************************************
      DIMENSION A(LEN,N,N), B(LEN,N,N), C(LEN,N,N)
     #,         WRK1(LEN),  WRK2(LEN)
```

```
      DO I = 1, N
         DO J = 1, N
            C(:,I,J) = 0.0
            DO K = 1, N

               WRK1(:) = C(:,I,J)

               DO KK = 1, LEN
                  WRK2(KK) = A(KK,I,K)*B(KK,K,J)
               ENDDO

               C(:,I,J) = WRK1(:) + WRK2(:)

            ENDDO
         ENDDO
      ENDDO
      RETURN
      END
      SUBROUTINE MULTAV(A,V,W,N)
C*****************************************************************
C             Subroutine MULTAV
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: Multiplication of an NxN matrix A and a vector V: W = A V
C
C VARIABLES:
C     INPUT:
C          A-input matrix;
C          B-input vector;
C     OUTPUT:
C          C-output vector.
C*****************************************************************
      DIMENSION A(N,N), V(N), W(N)
      DO I = 1,  N
         W(I) = 0.0
         DO J = 1, N
            W(I) = W(I) + A(I,J)*V(J)
         ENDDO
      ENDDO
      RETURN
      END
      SUBROUTINE MULTAVK(LEN,A,V,W,N)
C*****************************************************************
C             Subroutine MULTAVK
C*****************************************************************
C
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: Multiplication of an NxN matrix A and a vector V: W = A V
C
C VARIABLES:
C     INPUT:
C          A-input of LEN 2x2 matrices;
C          B-input of LEN vectors;
C     OUTPUT:
C          C-output vector.
C*****************************************************************
      DIMENSION A(LEN,N,N), V(LEN,N), W(LEN,N), WRK(LEN)
      DO I = 1,  N
         W(:,I) = 0.0
         DO J = 1, N
            WRK(:)= W(:,I)
            W(:,I) = WRK(:) + A(:,I,J)*V(:,J)
         ENDDO
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE S2ODE(LEN, DN1, DN2, FN1, FN2, AM, BM, CM, JM)
C*********************************************************************
C                  Subroutine S2ODE
C*********************************************************************
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: This routine solves the second-order ODE which originally results from
C the \del^4 equation on the sphere.  But appplication is not so restricted
C as far as the discretized equation is in the format as (1).
C The unknown to be solved is a 2x1 vector (Lindzen and Kuo, 1969, MWR).
C Therefore this routine is used here to solve for the FFT  coefficients (k > 0)
C in the \del^4 direct solver if the the B.C's are chosen to be zero for the poles.
C
C The discretized form of the ODE is:
C
C               A(j) X(j-1) + B(j) X(j) + C(j) X(j+1) = F(j) ......(1)
C                       (for j = 2, 3, 4, ....., JM-1)
C
C The boundary conditions are:
C
C                   A(1) X(1) + B(1) X(2)  = F(1)          ........(2.1)
C
C           and     A(JM) X(JM-1) + B(JM) X(JM) = F(JM)    ........(2.2)
C
C The coefficient A, B and C are all real.  The forcing F and hence the
C solution X have both real and imaginery parts.
C
C CALLED FROM:
C     DIRSOL4
C
C SUBROUTINES CALLED:
C     INVERSE1: inversion of one 2x2 matrix;
C     INVERSE2: inversion of a number of 2x2 matrices;
C     MULTAB:   multiplication of two matrices;
C     MULTABK:  multiplication of a number of two matrix set;
C     MULTAV :  multiplication of a matrix and a vector.
C     MULTAVK : multiplication of a number of a matrix-vector sets.
C
C VARIABLES:
C
C     INPUT:
C        AM-array storing the coefficient matrix A as in (1);
C        BM-array storing the coefficient matrix B as in (1);
C        CM-array storing the coefficient matrix C as in (1);
C        DN1-array storing the real part of the vector of the forcing
C            term F in (1);
C        DN2-array storing the imaginery part of the vector of the forcing
C            term F in (1);
C        JM-number of total grid points including the two boundary points.
C
C     OUTPUT:
C        FN1-array storing the real part of the solution vector X of (1);
C        FN2-array storing the imaginery part of the solution vector X of (1);
C
C     LOCAL:
C        ALPHA-array storing intermediate coefficient matrix alpha
C              (Lindzen and Kuo, 1969);
C        BETA1-real part of vector beta (Lindzen and Kuo, 1969);
C        BETA2-imaginery part of vector beta (Lindzen and Kuo, 1969);
C        VRK1-working array;
C        VRK2-working array;
C        VRK3-working array;
C        VRK4-working array;
C        VRK5-working array;
C        VRK6-working array;
C        WRK1-working array;
C        WRK2-working array;
C        WRK3-working array;
C        WRK4-working array;
```

```
C*********************************************************************
      DIMENSION DN1(LEN,2,JM), DN2(LEN,2,JM), FN1(LEN,2,JM)
     #,         FN2(LEN,2,JM)

      DIMENSION AM(LEN,2,2,JM), BM(LEN,2,2,JM), CM(LEN,2,2,JM)

      DIMENSION ALPHA(LEN,2,2,JM), BETA1(LEN,2,JM), BETA2(LEN,2,JM)

      DIMENSION WRK1(LEN,2,2), WRK2(LEN,2,2), WRK3(LEN,2,2)
     #,         WRK4(LEN,2,2)
     #,         VRK1(LEN,2), VRK2(LEN,2), VRK3(LEN,2), VRK4(LEN,2)
     #,         VRK5(LEN,2), VRK6(LEN,2)

      WRK1(:,:,:) = AM(:,:,:,1)

      CALL INVERSE2(LEN, WRK1, WRK2)

      WRK1(:,:,:) = - WRK2(:,:,:)
      WRK4(:,:,:) = BM(:,:,:,1)
      CALL MULTABK(LEN, WRK1, WRK4, WRK3, 2)

      ALPHA(:,:,:,1) = WRK3(:,:,:)

      VRK1(:,:) = DN1(:,:,1)
      VRK2(:,:) = DN2(:,:,1)

      CALL MULTAVK(LEN, WRK2, VRK1, VRK3, 2)
      CALL MULTAVK(LEN, WRK2, VRK2, VRK4, 2)

      BETA1(:,:,1) = VRK3(:,:)
      BETA2(:,:,1) = VRK4(:,:)

      DO J = 2, JM-1
         WRK1(:,:,:) = ALPHA(:,:,:,J-1)
         WRK2(:,:,:) = AM(:,:,:,J)
         CALL MULTABK(LEN, WRK2, WRK1, WRK3, 2)

         WRK1(:,:,:) = WRK3(:,:,:) + BM(:,:,:,J)

         CALL INVERSE2(LEN,WRK1,WRK2)

         WRK1(:,:,:) = CM(:,:,:,J)
         CALL MULTABK(LEN, WRK2, WRK1, WRK3, 2)

         ALPHA(:,:,:,J) = - WRK3(:,:,:)

         WRK1(:,:,:) = AM(:,:,:,J)
         VRK1(:,:) = BETA1(:,:,J-1)
         VRK2(:,:) = BETA2(:,:,J-1)

         CALL MULTAVK(LEN, WRK1, VRK1, VRK3, 2)
         CALL MULTAVK(LEN, WRK1, VRK2, VRK4, 2)

         VRK3(:,:) = DN1(:,:,J) - VRK3(:,:)
         VRK4(:,:) = DN2(:,:,J) - VRK4(:,:)

         CALL MULTAVK(LEN, WRK2, VRK3, VRK1, 2)
         CALL MULTAVK(LEN, WRK2, VRK4, VRK2, 2)

         BETA1(:,:,J) = VRK1(:,:)
         BETA2(:,:,J) = VRK2(:,:)
      ENDDO

      VRK1(:,:) = DN1(:,:,JM)
      VRK2(:,:) = DN2(:,:,JM)
      VRK3(:,:) = BETA1(:,:,JM-1)
      VRK4(:,:) = BETA2(:,:,JM-1)

      WRK1(:,:,:) = AM(:,:,:,JM)
```

```
      CALL MULTAVK(LEN, WRK1, VRK3, VRK5, 2)
      CALL MULTAVK(LEN, WRK1, VRK4, VRK6, 2)

      VRK1(:,:) = DN1(:,:,JM) - VRK5(:,:)
      VRK2(:,:) = DN2(:,:,JM) - VRK6(:,:)

      WRK1(:,:,:) = ALPHA(:,:,:,JM-1)
      WRK2(:,:,:) = AM(:,:,:,JM)
      CALL MULTABK(LEN, WRK2, WRK1, WRK3, 2)

      WRK4(:,:,:) = BM(:,:,:,JM) + WRK3(:,:,:)
      CALL INVERSE2(LEN, WRK4, WRK1)

      CALL MULTAVK(LEN, WRK1, VRK1, VRK3, 2)
      CALL MULTAVK(LEN, WRK1, VRK2, VRK4, 2)

      FN1(:,:,JM) = VRK3(:,:)
      FN2(:,:,JM) = VRK4(:,:)

      DO J = JM-1, 1, -1
         WRK1(:,:,:) = ALPHA(:,:,:,J)
         VRK1(:,:) = FN1(:,:,J+1)
         VRK2(:,:) = FN2(:,:,J+1)

         CALL MULTAVK(LEN, WRK1, VRK1, VRK3, 2)
         CALL MULTAVK(LEN, WRK1, VRK2, VRK4, 2)

         FN1(:,:,J) = VRK3(:,:) + BETA1(:,:,J)
         FN2(:,:,J) = VRK4(:,:) + BETA2(:,:,J)
      ENDDO

      RETURN
      END
      SUBROUTINE SOMEAN(DN1, FN1, AM, BM, CM, JM)
C*****************************************************************
C                 Subroutine SOMEAN
C*****************************************************************
C WRITTEN BY: Yong Li, GSFC Code 910.3, Apr 1994
C
C PURPOSE: This routine solves the second-order ODE which originally results from
C the \del^4 equation on the sphere.  But appplication is not so restricted
C as far as the discretized equation is of the following format.
C The unknown to be solved is a 2x1 vector (Lindzen and Kuo, 1969, MWR).
C This routine is used here to solve for the zonal mean, equivalent to FFT
C coefficients (k = 0) in the \del^4 direct solver.
C
C The discretized form of the ODE is:
C
C               A(j) X(j-1) + B(j) X(j) + C(j) X(j+1) = F(j) ......(1)
C                       (for j = 2, 3, 4, ....., JM-1)
C
C The boundary conditions are:
C
C                   A(1) X(1) + B(1) X(2)  = F(1)          ........(2.1)
C
C           and     A(JM) X(JM-1) + B(JM) X(JM) = F(JM)    ........(2.2)
C
C The coefficient A, B and C are all real.  The forcing F and hence the
C solution X also real.
C
C CALLED FROM:
C     DIRSOL4
C
C SUBROUTINES CALLED:
C     INVERSE: matrix inversion;
C     MULTAB: multiplication of two matrix;
C     MULTAV: multiplication of a matrix and a vector.
C
C VARIABLES:
```

```
C
C     INPUT:
C        AM-array storing the coefficient matrix A as in (1);
C        BM-array storing the coefficient matrix B as in (1);
C        CM-array storing the coefficient matrix C as in (1);
C        DN1-array storing the vector of the forcing term F in (1);
C        JM-number of total grid points including the two boundary points.
C
C     OUTPUT:
C        FN1-array storing the solution vector X of (1);
C
C     LOCAL:
C        ALPHA-array storing intermediate coefficient matrix alpha
C              (Lindzen and Kuo, 1969);
C        BETA1-real part of vector beta (Lindzen and Kuo, 1969);
C        VRK1-working array;
C        VRK2-working array;
C        VRK3-working array;
C        VRK4-working array;
C        VRK5-working array;
C        VRK6-working array;
C        WRK1-working array;
C        WRK2-working array;
C        WRK3-working array;
C        WRK4-working array;
C*********************************************************************
      DIMENSION AM(2,2,JM), BM(2,2,JM), CM(2,2,JM)
     #,         DN1(2,JM), FN1(2,JM), ALPHA(2,2,JM), BETA1(2,JM)

      DIMENSION WRK1(2,2), WRK2(2,2), WRK3(2,2), WRK4(2,2)

      DIMENSION VRK1(2), VRK2(2), VRK3(2), VRK4(2), VRK5(2)
     #,         VRK6(2)

      WRK1(:,:) = AM(:,:,1)
      CALL INVERSE1(WRK1, WRK2)

      WRK1(:,:) = - WRK2(:,:)
      WRK4(:,:) = BM(:,:,1)
      CALL MULTAB(WRK1, WRK4, WRK3, 2)
      ALPHA(:,:,1) = WRK3(:,:)

      VRK1(:) = DN1(:,1)

      CALL MULTAV(WRK2, VRK1, VRK3, 2)

      BETA1(:,1) = VRK3(:)

      DO J = 2, JM-1
         WRK1(:,:) = ALPHA(:,:,J-1)
         WRK2(:,:) = AM(:,:,J)
         CALL MULTAB(WRK2,WRK1,WRK3,2)
         WRK1(:,:) = WRK3(:,:) + BM(:,:,J)

         CALL INVERSE1(WRK1,WRK2)

         WRK1(:,:) = CM(:,:,J)
         CALL MULTAB(WRK2,WRK1,WRK3,2)

         ALPHA(:,:,J) = - WRK3(:,:)

         WRK1(:,:) = AM(:,:,J)
         VRK1(:) = BETA1(:,J-1)

         CALL MULTAV(WRK1, VRK1, VRK3, 2)

         VRK3(:) = DN1(:,J) - VRK3(:)

         CALL MULTAV(WRK2, VRK3, VRK1, 2)
```

```
   BETA1(:,J) = VRK1(:)
ENDDO

VRK1(:) = DN1(:,JM)
VRK3(:) = BETA1(:,JM-1)

WRK1(:,:) = AM(:,:,JM)
CALL MULTAV(WRK1, VRK3, VRK5, 2)

VRK1(:) = DN1(:,JM) - VRK5(:)

WRK1(:,:) = ALPHA(:,:,JM-1)
WRK2(:,:) = AM(:,:,JM)
CALL MULTAB(WRK2, WRK1, WRK3, 2)

WRK4(:,:) = BM(:,:,JM) + WRK3(:,:)
CALL INVERSE1(WRK4, WRK1)

CALL MULTAV(WRK1, VRK1, VRK3, 2)

FN1(:,JM) = VRK3(:)

DO J = JM-1, 1, -1
   WRK1(:,:) = ALPHA(:,:,J)
   VRK1(:) = FN1(:,J+1)

   CALL MULTAV(WRK1, VRK1, VRK3, 2)

   FN1(:,J) = VRK3(:) + BETA1(:,J)
ENDDO

RETURN
END
SUBROUTINE WRAP1(IMA,IM,JM,A)
DIMENSION A(IMA,JM)

DO J = 1, JM
   DO I = IM+1, IMA
      A(I,J) = A(I-IM,J)
   ENDDO
ENDDO

RETURN
END
SUBROUTINE WRAPD(IMA,IM,JM,A)
DIMENSION A(IMA,JM)

DO J = 1, JM
   A(1,J) = A(IM+1,J)
   DO I = IM+2, IMA
      A(I,J) = A(I-IM,J)
   ENDDO
ENDDO

RETURN
END
```

References

Bates, J. R., S. Moorthi and R. W. Higgins, 1993: A global multilevel atmospheric model using a vector semi-Lagrangian finite-difference scheme. Part I: Adiabatic formulation. *Mon. Wea. Rev.*, **121**, 244-263.

Bates, J. R., Y. Li, A. Brandt, S. F. McCormick and J. Ruge, 1994: A global shallow water model based on the semi-Lagrangian advection of potential vorticity. *Accepted for publication in Q. J. Roy. Met. Soc.*

Gravel, S., A. Staniforth and J. Côté, 1993: A stability analysis of a family of baroclinic semi-Lagrangian forecast models. *Mon. Wea. Rev.*, **121**, 815-824.

Jakimow, G., E. Yakimiw and A. Robert, 1992: An implicit formulation for horizontal diffusion in gridpoint models. *Mon. Wea. Rev.*, **120**, 124-130.

Lindzen, R. S. and H.-L. Kuo, 1969: A reliable method for the numerical integration of a large class of ordinary and partial differential equations. *Mon. Wea. Rev.*, **97**,732-734.

Lynch, P. and Huang, X.-Y., 1992: Initialization of the HIRLAM model using a digital filter. *Mon. Wea. Rev.*, **120**, 1019-1034.

McDonald, A., 1994: Using second, fourth and sixth order implicit horizontal diffusion to control noise in three dimensional semi-Lagrangian, semi-implicit, limited area, gridpoint models of the primitive equations. *Unpublished Note* (available from Irish Meteorological Service, Glasnevin Hill, Dublin 9, Ireland).

McDonald, A. and J. R. Bates, 1989: Semi-Lagrangian integration of a gridpoint shallow water model on the sphere. *Mon. Wea. Rev.*, **117**, 130-137.

Mesinger, F. and A. Arakawa, 1976: Numerical methods used in atmospheric models, Vol.1. *GARP Publications Series No.17.*, World Meteorological Organization, Geneva, Switzerland.

Moorthi, S. and R. W. Higgins, 1992: A fast direct solver for a class of two-dimensional seperable elliptic equations on the sphere. *NASA Tech. Memo. 104567.*

Moorthi, S. and R. W. Higgins, 1993: Application of fast Fourier transformations to the direct solution of a class of two-dimensional separable elliptic equations on the sphere. *Mon. Wea. Rev.*, **121**, 290-296.

Moorthi, S., R. W. Higgins and J. R. Bates, 1994: A global multilevel atmospheric model using a vector semi-Lagrangian finite-difference scheme. Part 2: Version with physics.

Accepted for publication in Mon. Wea. Rev.

Moorthi, S., 1994: Numerical weather prediction experiments with a semi-Lagrangian semi-implicit grid-point general circulation model at NMC. *10th Conference on Numerical Weather Prediction*, July 18-22, 1994, Portland, Oregon.

Raymond, W. H. and A. Garder, 1991: A review of recursive and implicit filters. *Mon. Wea. Rev.*, **119**, 477-495.

Ritchie, H., C. Temperton, A. Simmons, M. Hortal, T. Davies, D. Dent and M. Hamrud, 1994: Implementation of the semi-Lagrangian method in a high resolution version of the ECMWF forecast model. *Mon. Wea. Rev.* (in press).

Semazzi, F. H. M. and P. Dekker, 1994: Optimal accuracy in semi-Lagrangian models. Part I. Linear formulation. *Mon. Wea. Rev.* (in press).

Shapiro, R., 1971: The use of linear filtering as a parameterization of atmospheric diffusion. *J. Atmos. Sci.*, **28**, 523-531.

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.

Shuman, F. G., 1957: Numerical methods in weather prediction: II. Smoothing and filtering. *Mon. Wea. Rev.*, **85**, 357-361.

Tanguay, M., E. Yakimiw, H. Ritchie and A. Robert, 1992: Advantages of spatial averaging in semi-implicit semi-Lagrangian schemes. *Mon. Wea. Rev.*, **120**, 113-123.

Williamson, D. L. and J. G. Olson, 1994: Climate simulations with a semi-Lagrangian version of the NCAR CCM2. *Mon. Wea. Rev.*, **122**, 1594-1610.

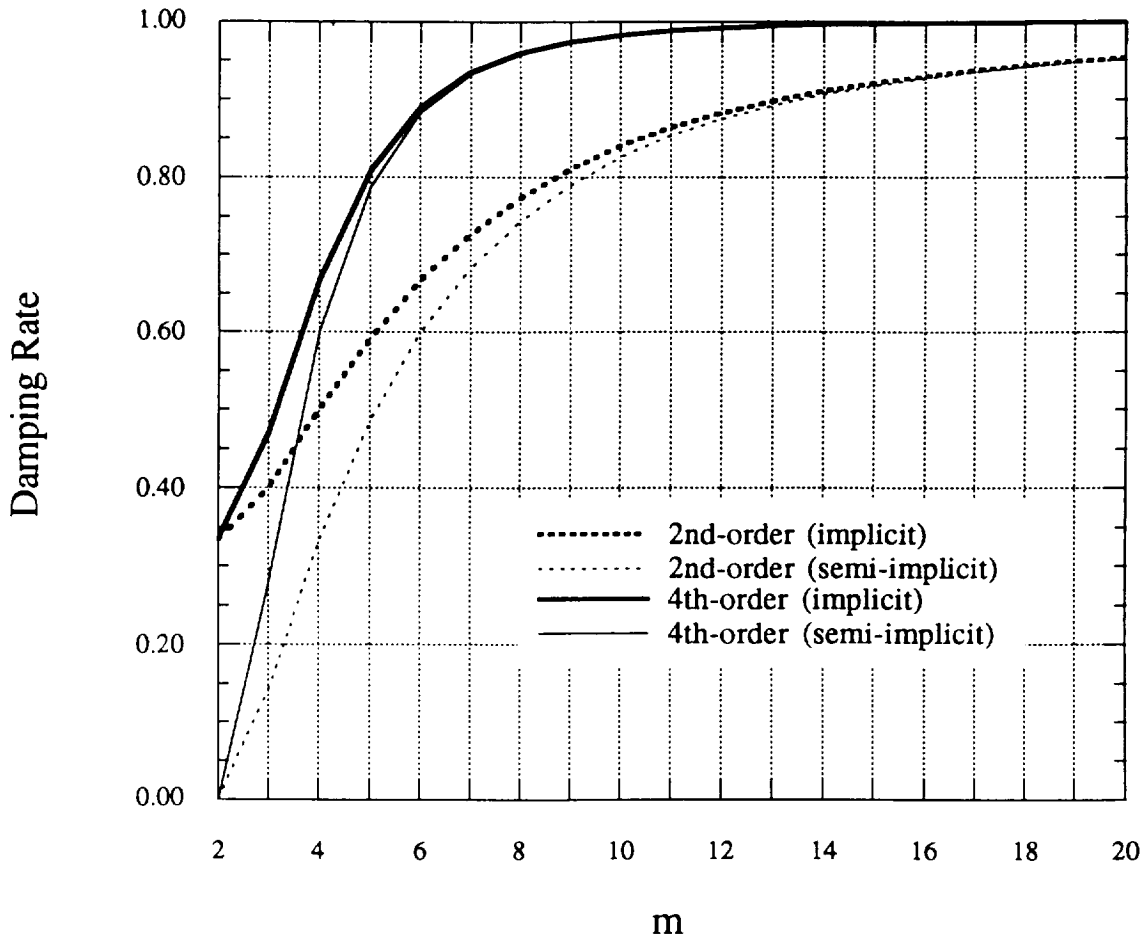
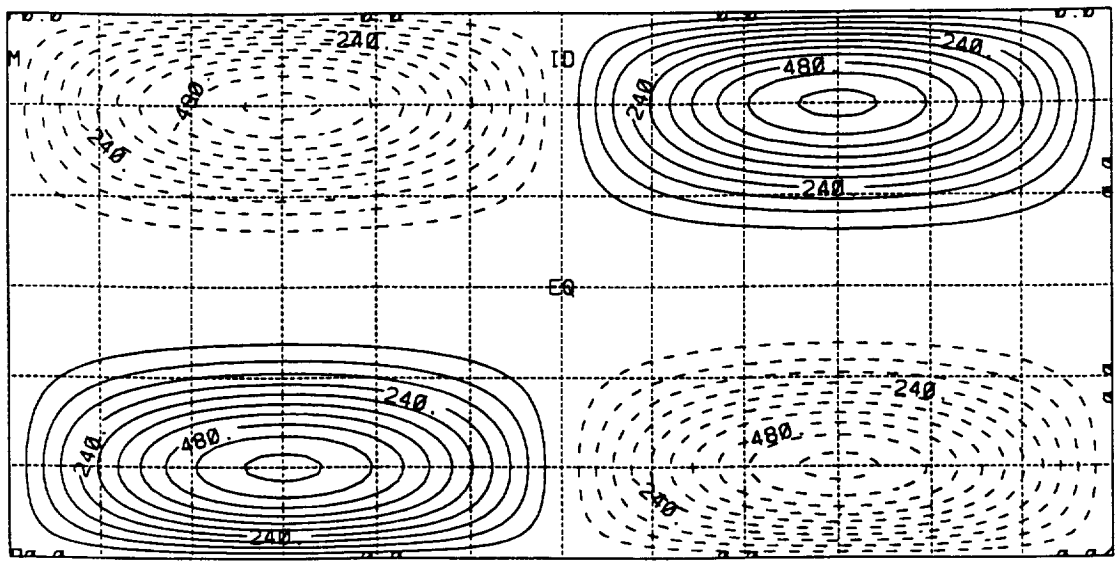


Figure 1: The damping rates as a function of wavenumber for the second and the fourth order diffusion in both implicit and explicit formulations.

(a)



(b)

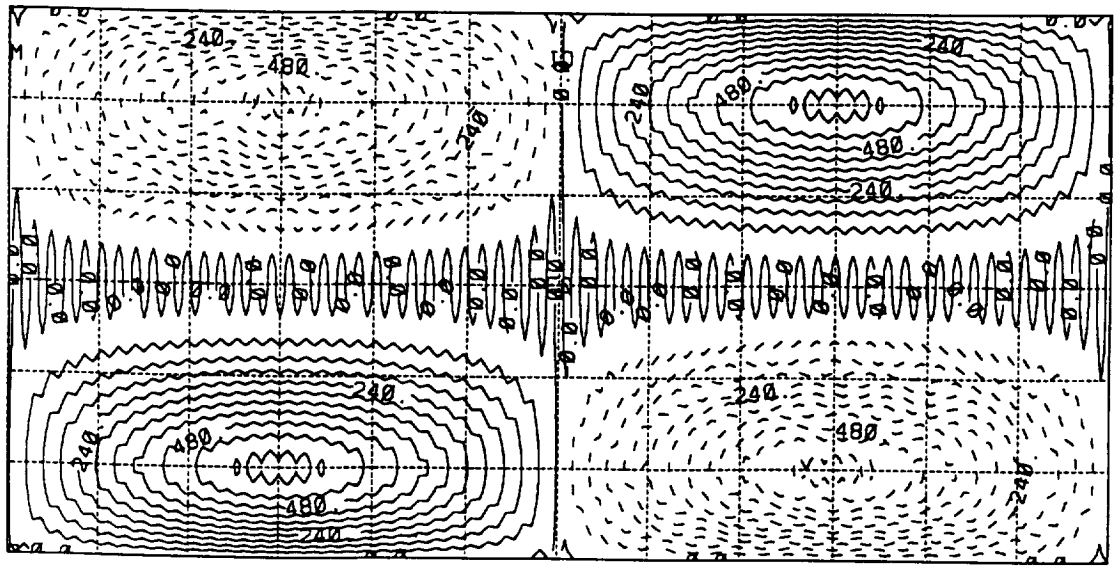
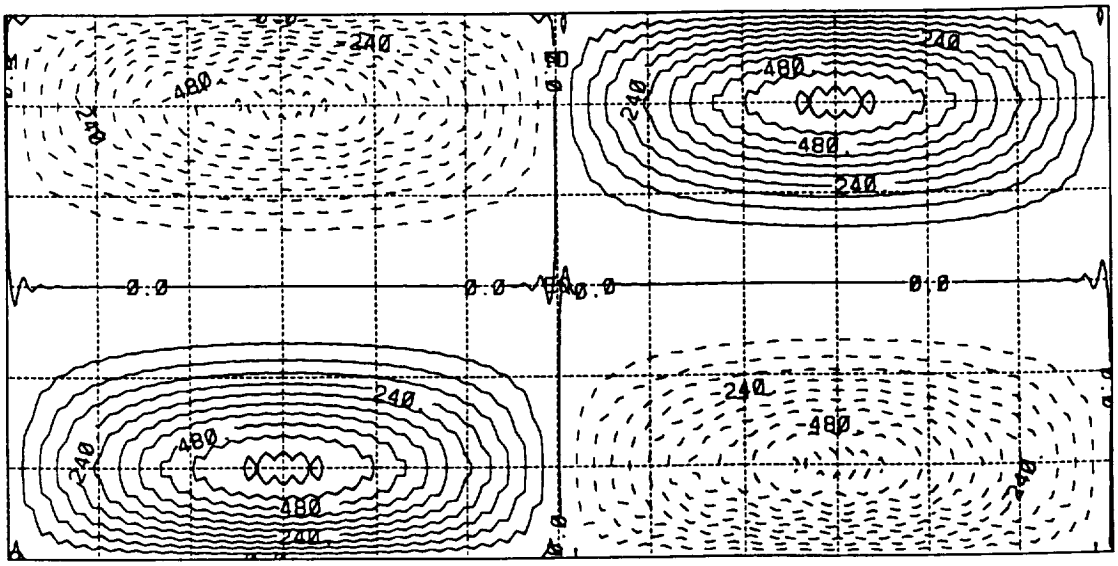


Figure 2: (a) Spherical harmonic wave number one; (b) Same as (a), but with $2\Delta x$ waves embedded; (c) After applying the semi-implicit formulation of fourth order diffusion to (b); (d) After applying the implicit formulation to (b); (e) After applying both the semi-implicit and the implicit formulations to (b).

(c)



(d)

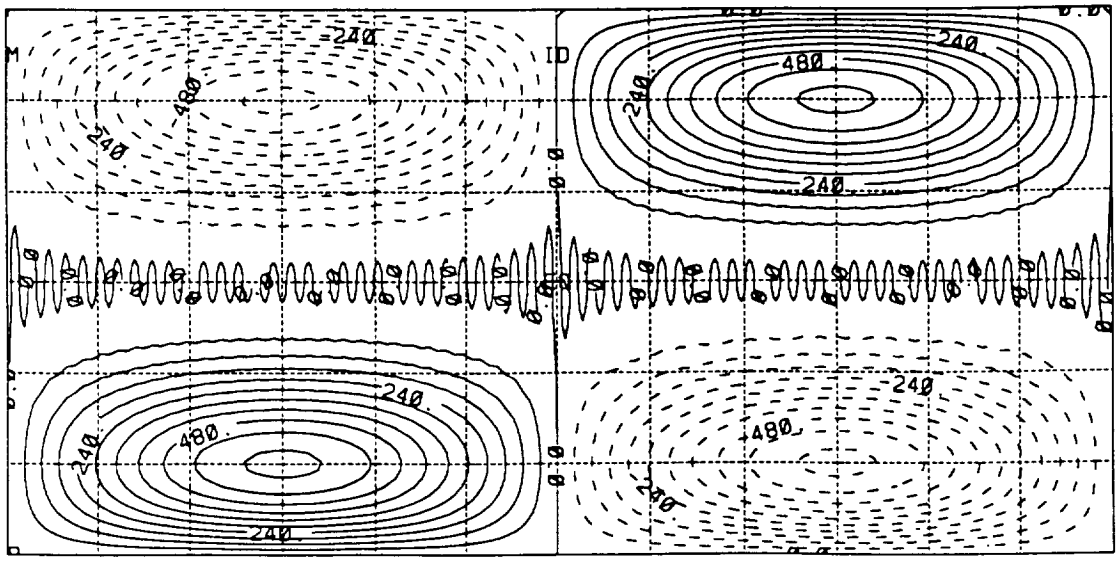


Figure 2 (continued)

(e)

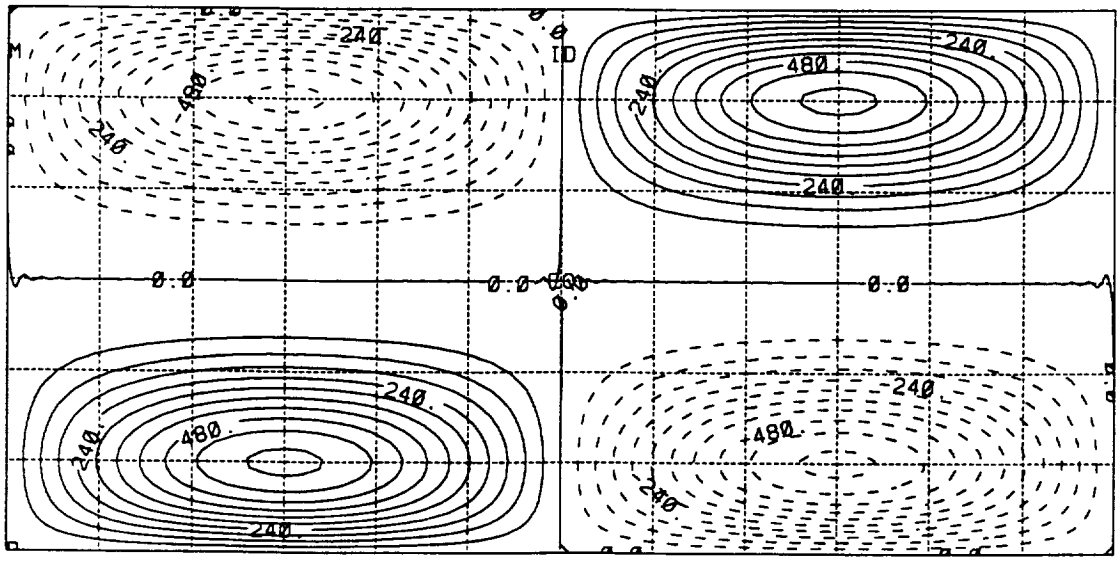


Figure 2 (continued)

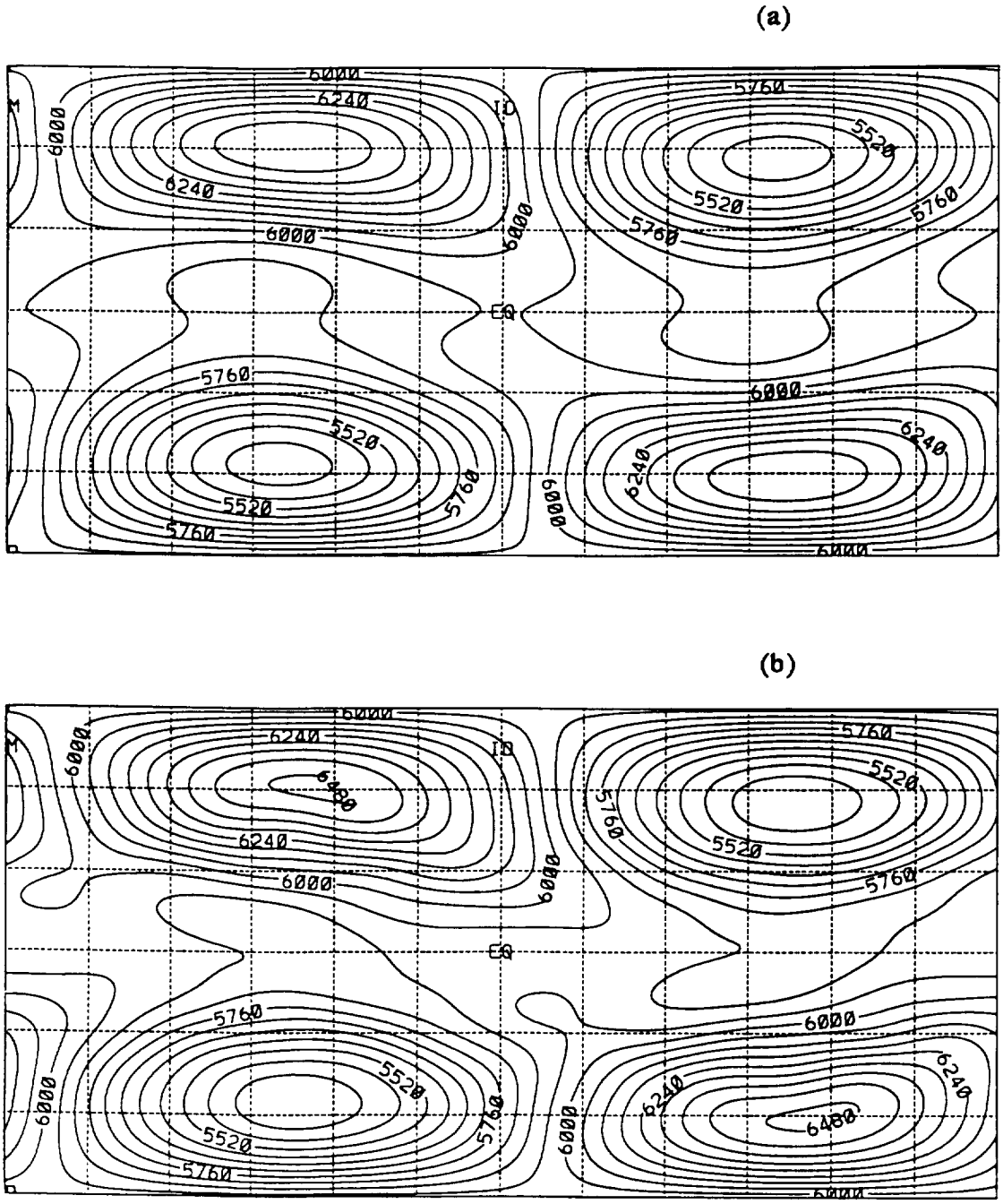
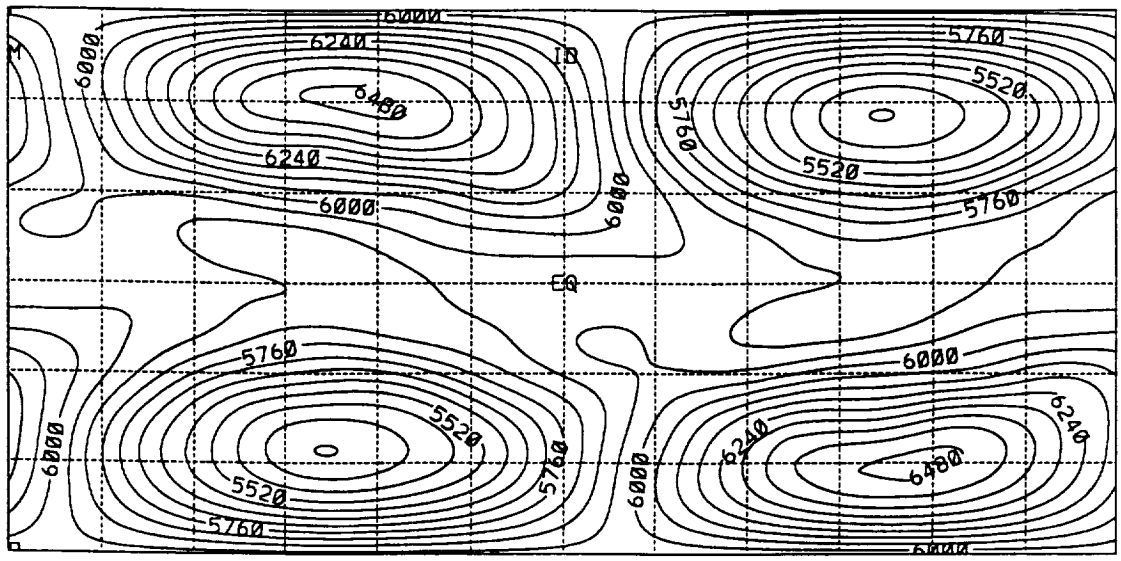


Figure 3: (a) Geopotential at 50 days for the integration with the initial condition of McDonald and Bates (1989) and a value of 0.03 for the uncentering parameter(contours every 60.0 meters); (b) Same as (a) but with the fourth order diffusion applied to geopotential(contours every 60.0 meters); (c) Same as (a) but with the fourth order diffusion applied to divergence (contours every 60.0 meters); (d) Same as (a) but with the fourth order diffusion applied to both geopotential and divergence(contours every 60.0 meters).

(c)



(d)

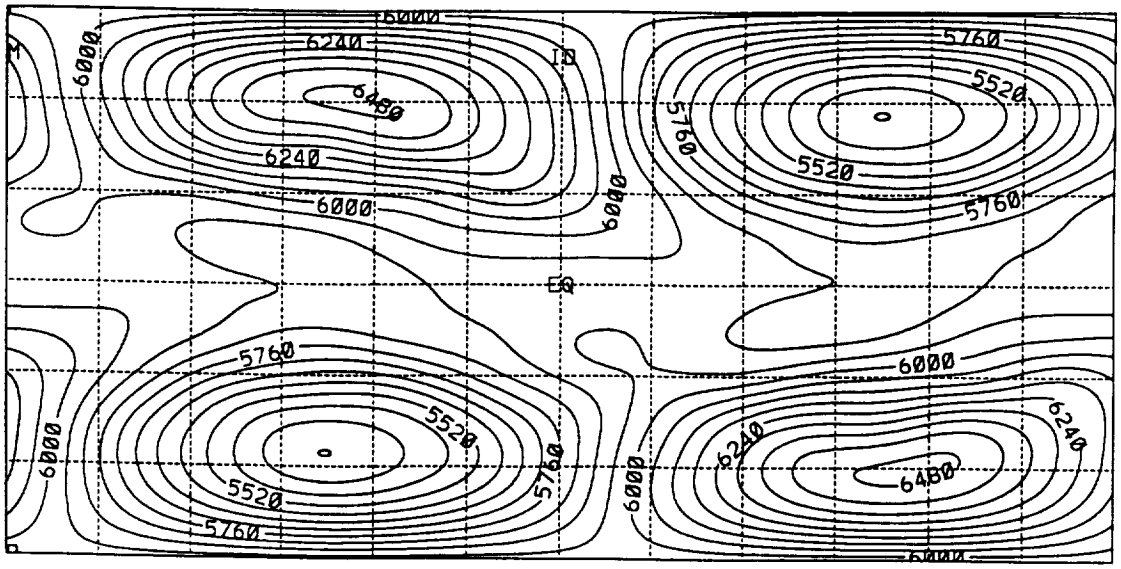


Figure 3 (continued)

(a)

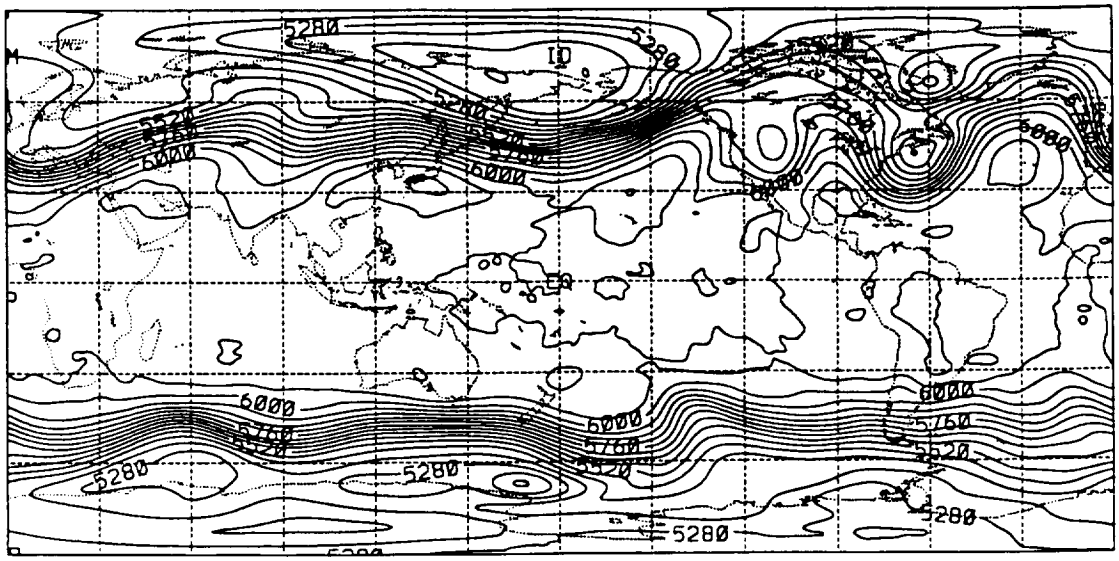
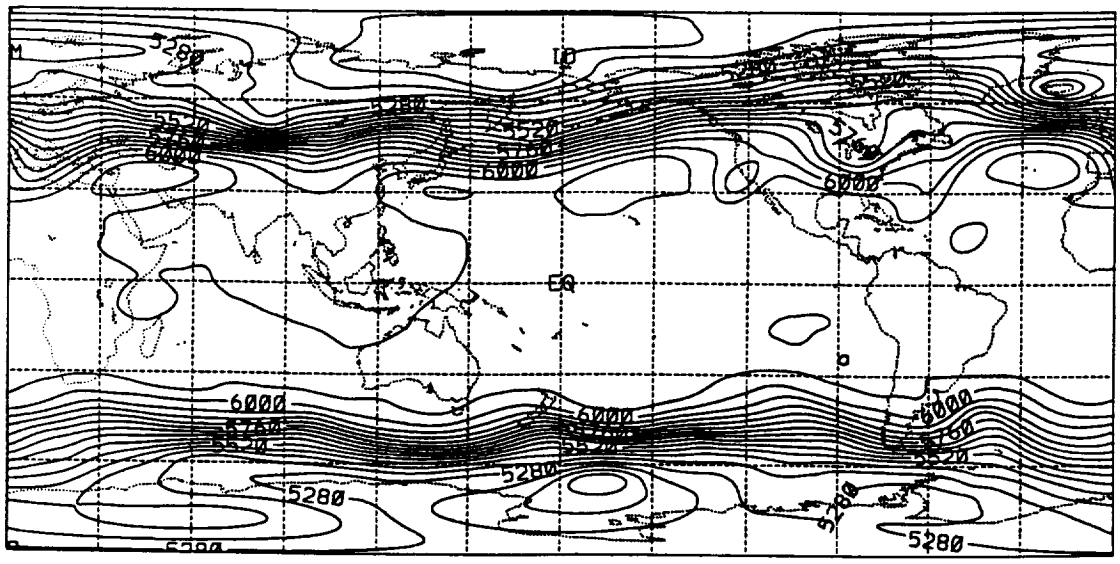


Figure 4: (a) Geopotential of the ECMWF analysis of 0000 UTC, 1 December, 1992 (contours every 60.0 meters); (b) Geopotential after 3 days of integration with a value of 0.05 for the uncentering parameter(contours every 60.0 meters); (c) Same as (b) but with a value of 0.015 for the uncentering parameter and the the fourth order diffusion applied to both geopotential and divergence (contours every 60.0 meters).

(b)



(c)

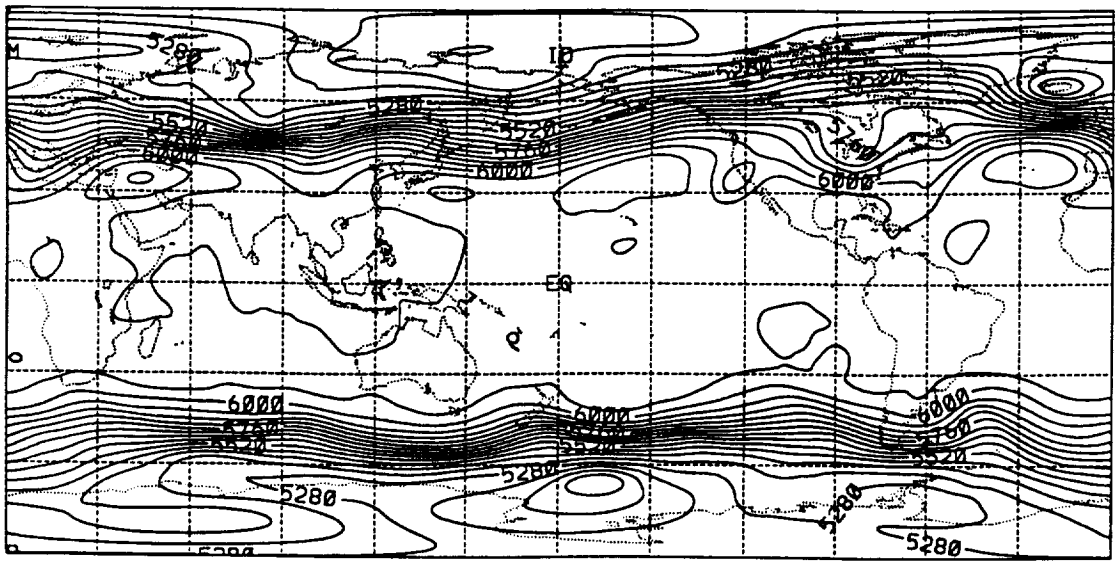


Figure 4 (continued)

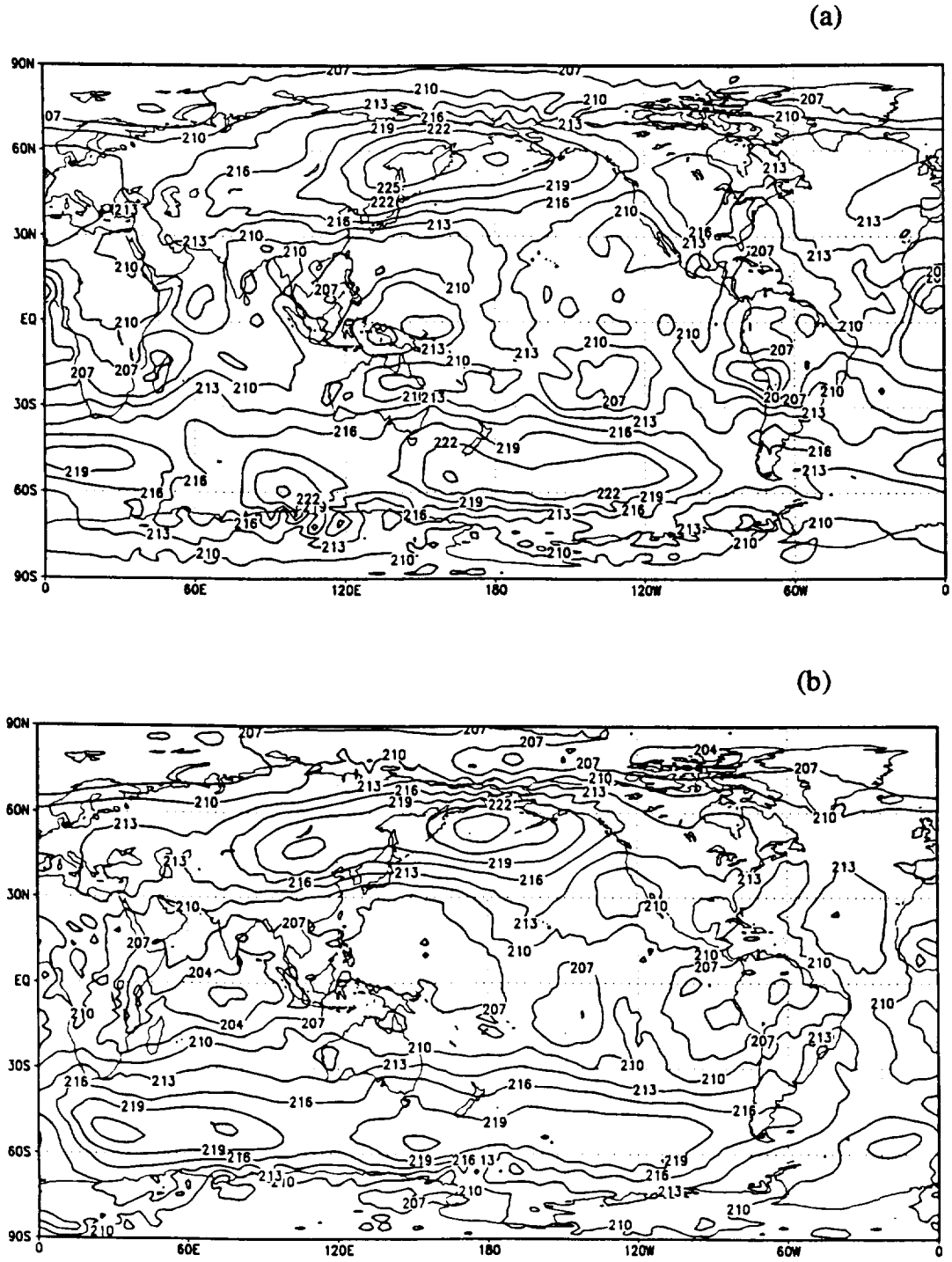
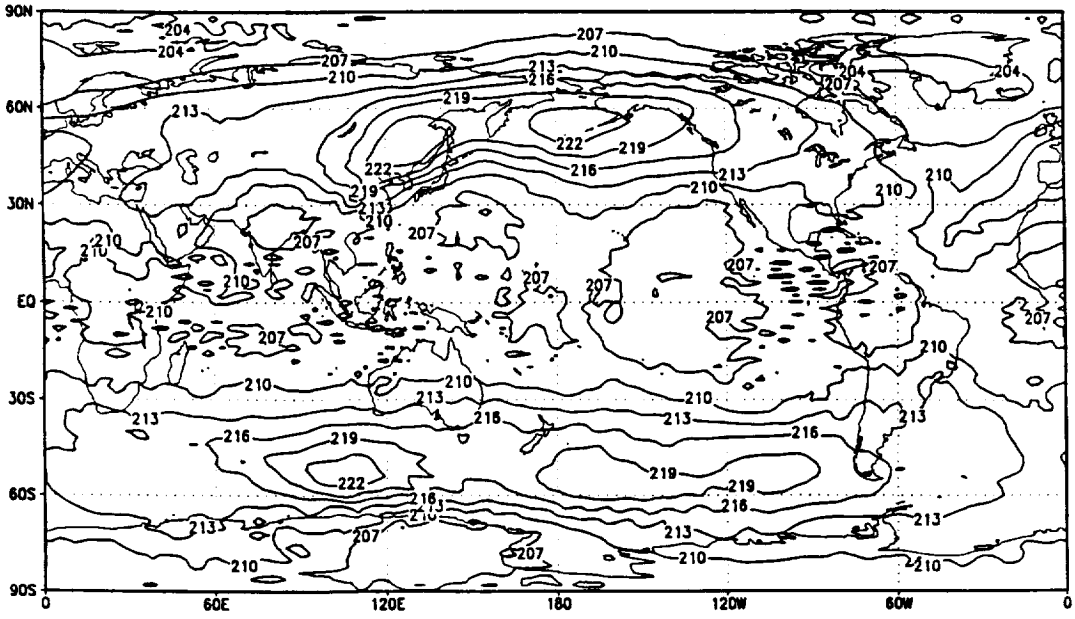


Figure 5: 50 mb temperature field from NASA/GLA 3-D semi-Lagrangian semi-implicit GCM without the application of the horizontal diffusion. The value for the off-centering parameter is 0.05. (a) 3-day forecast; (b) 5-day forecast; (c) 7-day forecast; (d) 10-day forecast. (Contour interval is 3 degrees).

(c)



(d)

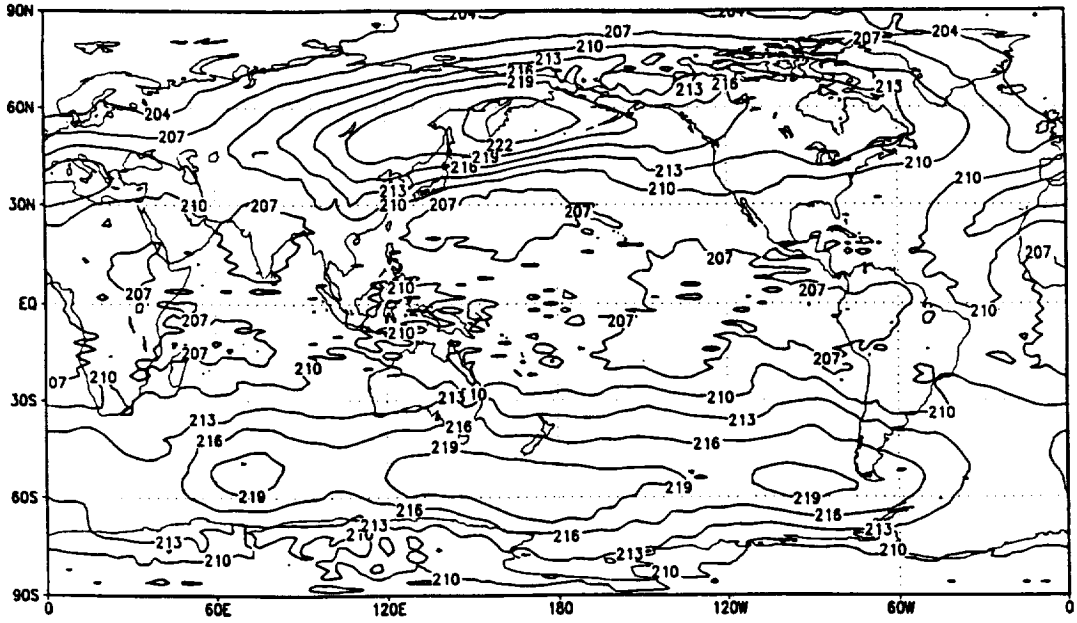


Figure 5 (continued)

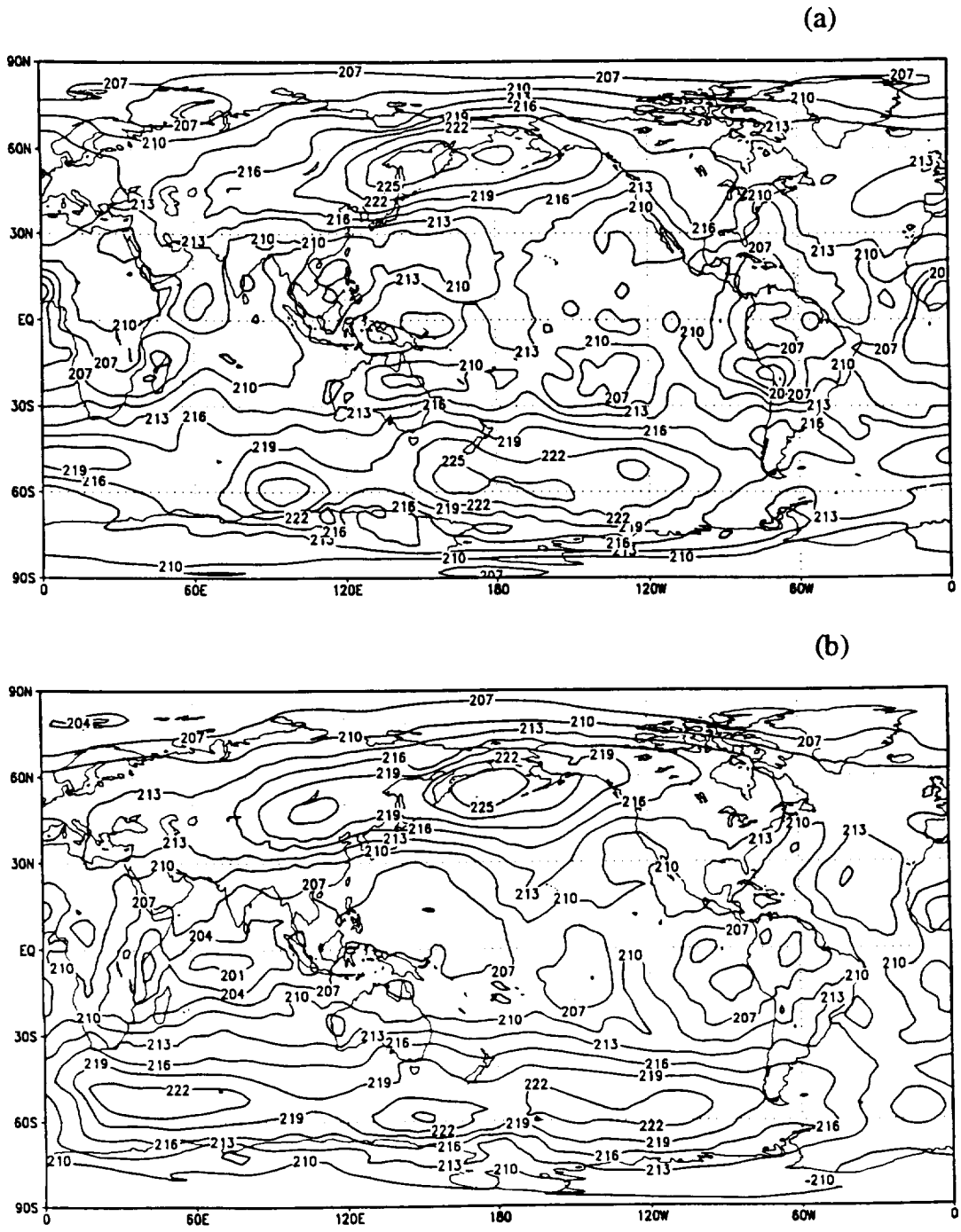


Figure 6: 50 mb temperature field from NASA/GLA 3-D semi-Lagrangian semi-implicit GCM with the application of the horizontal diffusion. The value for the off-centering parameter is 0.05. (a) 3-day forecast; (b) 5-day forecast; (c) 7-day forecast; (d) 10-day forecast. (Contour interval is 3 degrees).

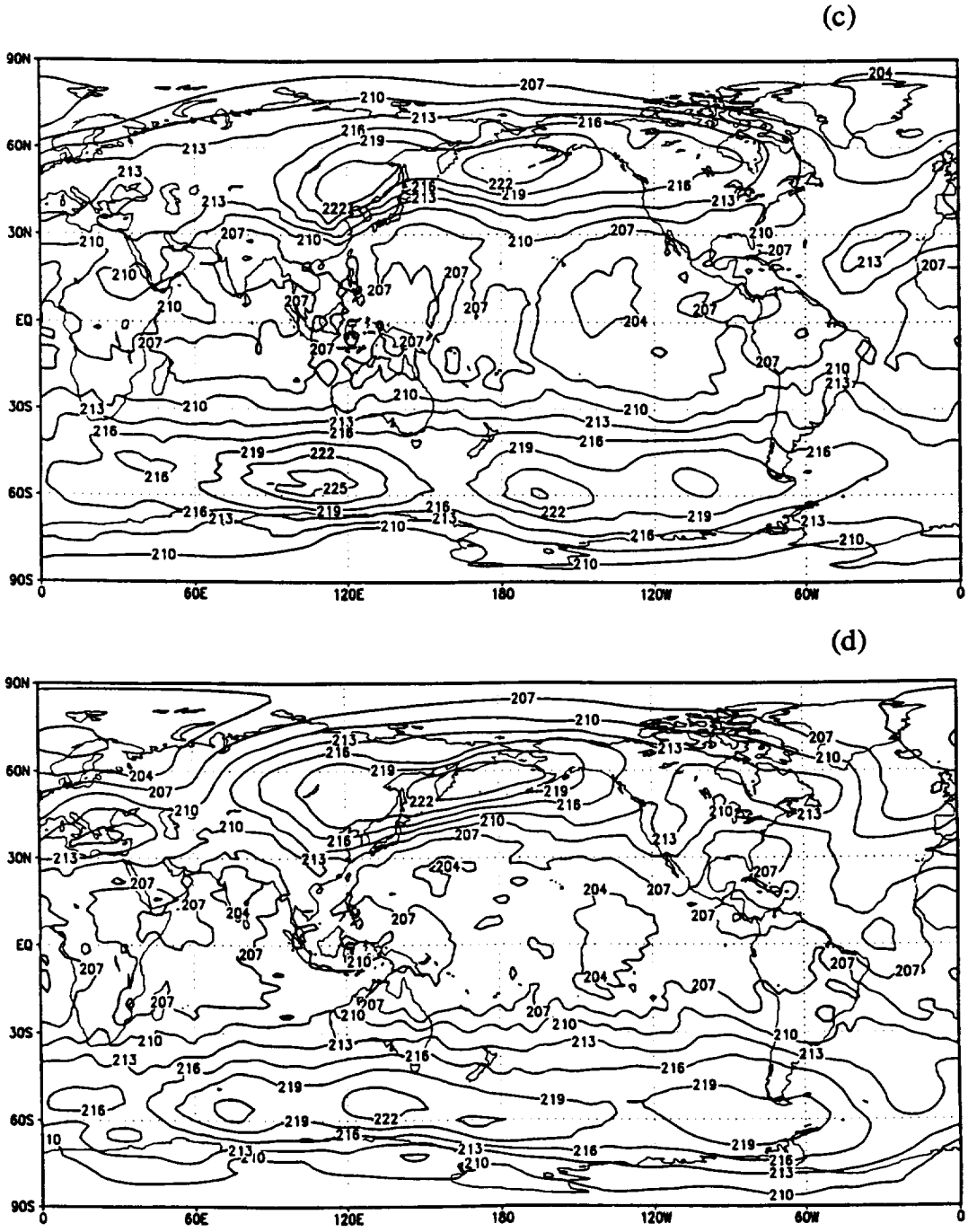


Figure 6 (continued)

Previous Volumes in this Series

Volume 1	Documentation of the Goddard Earth Observing System (GEOS) general circulation model - Version 1
September 1994	L.L. Takacs, A. Molod, and T. Wang

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1994	Technical Memorandum

4. TITLE AND SUBTITLE
Technical Report Series on Global Modeling and Data Assimilation
Volume 2 - Direct Solution of the Implicit Formulation of Fourth Order Horizontal Diffusion for Gridpoint Models on the Sphere

5. FUNDING NUMBERS
C - NAS5-32332
Code 910

6. AUTHOR(S)
Yong Li, S. Moorthi, and J. Ray Bates
Max J. Suarez, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS (ES)
Laboratory for Atmospheres
Data Assimilation Office
Goddard Space Flight Center
Greenbelt, Maryland 20771

8. PEFORMING ORGANIZATION REPORT NUMBER
94B00143

9. SPONSORING / MONITORING ADGENCY NAME(S) AND ADDRESS (ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING / MONITORING ADGENCY REPORT NUMBER
TM-104606, Vol. 2

11. SUPPLEMENTARY NOTES
Y. Li: General Sciences Corporation, Laurel, Maryland (at GSFC Data Assimilation Office, Greenbelt, Maryland); S. Moorthi, National Meteorological Center, Camp Springs, Maryland; M. Suarez and J. Bates: Goddard Space Flight Center, Greenbelt, Maryland

12a. DISTRIBUTION / AVAILABILITY STATMENT
Unclassified - Unlimited
Subject Category 46
This publication is available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090-2934, (301)621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

High order horizontal diffusion of the form $K\nabla^2$ is widely used in spectral models as a means of preventing energy accumulation at the shortest resolved scales. In the spectral context, an implicit formation of such diffusion is trivial to implement. The present note describes an efficient method of implementing implicit high order diffusion in global finite difference models. The method expresses the high order diffusion equation as a sequence of equations involving ∇^2. The solution is obtained by combining fast Fourier transforms in longitude with a finite difference solver for the second order ordinary differential equation in latitude.

The implicit diffusion routine is suitable for use in any finite difference global model that uses a regular latitude/longitude grid. The absence of a restriction on the timestep makes it particularly suitable for use in semi-Lagrangian models. The scale selectivity of the high order diffusion gives it an advantage over the uncentering method that has been used to control computational noise in two-time-level semi-Lagrangian models.

14. SUBJECT TERMS	15. NUMBER OF PAGES
GEOS, General Circulation Model, GCM, DAO, Semi-Lagrangian, Implicit, Diffusion	54
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL