Reduction of Temporal Discretization Error in an Atmospheric General Circulation Model

Final Report

May 14, 2013

Daisuke Hotta
hotta (at) umd.edu

Advisor: Prof. Eugenia Kalnay
Department of Atmospheric and Oceanic Science
ekalnay (at) atmos.umd.edu

Abstract

An atmospheric general circulation model (AGCM) is a computer program which numerically integrates the system of partial differential equations which describes the physical laws governing the flow of the Earth's atmosphere. Integration of AGCM is a central part of weather predictions and climate projections. Currently, due to restrictions on computational costs, most AGCMs adopt relatively low-order temporal discretizations, under the premise that errors arising from temporal discretizations are negligible compared to those associated with spatial discretizations or physical parametrizations. However, recent studies have shown that temporal discretizations is likely to be a significant source of model errors. The goal of this project is to reduce the temporal discretization error. Two different approaches will be examined: the first approach is to test a new time integration scheme, called Lorenz N-cycle scheme, which requires the same computational costs as the conventional scheme but yet can yield forecasts of higher-order accuracy. The second approach is to empirically estimate and correct

systematic model errors using the methodology presented by Danforth et al. (2007).

Contents

1 Introduction

Current weather forecasts and climate projections are based on numerical integration of the fluid dynamical partial differential equations which govern the evolution of the global atmosphere. The models which discretize those governing equations of the global atmospheric motion are called **Atmospheric General Circulation Model**s, or **AGCM**s. Currently, most AGCMs, even the state-of-the-art, most comprehensive models, adopt rather simple, low-order temporal discretization schemes, including the second order leapfrog scheme (which falls into first order when used in conjunction

with temporal filters such as Robert-Asselin (R/A) filter (Asselin, 1972) for stabilization), or the first order Matsuno scheme (Matsuno, 1966). The rationale behind the use of such low-order schemes in AGCMs is that, the model errors are dominated by the components arising from spatial discretization or from physical parametrizations and thus, in order to strike the best balance between computational efficiency (both in terms of speed and memory consumption) and accuracy, temporal discretizations should be made economical.

However, given the trend of increased spatial resolutions fueled by increase of computing powers, and the accelerating betterment of physical parametrizations, it might be natural to cast doubt on the validity of the assumption that errors due to temporal discretizations are less significant, and to consider refinements to temporal integration schemes. In fact, recent research shows that outputs of AGCMs show sensitive dependence on time stepping. For example, Teixeira et al. (2007) has shown that NOGAPS, the US Navy Operational Global Atmospheric Prediction System, exhibits considerably different model climates when integrated with different time steps, indicating that temporal discretization errors can account for a substantial component of the total model errors. In this project, we will consider two possible remedies to the issue presented above. The approaches of the two remedies are outlined in the next section.

2 Approaches

The first approach is to use a temporal discretization scheme which is computationally as economical as the conventional schemes but yet can be substantially more accurate. The efficient and accurate scheme, called the Lorenz N-cycle scheme, is implemented into an existing AGCM, called SPEEDY (**S**implified **P**arametrizations, primitiv**E** **E**quation **DY**namics) model, which adopts the leapfrog scheme with R/A filter as its temporal discretization. As a reference, the tried-and-truth Runge-Kutta scheme of 4th-order will be implemented as well. The performance of the Lorenz N-cycle in terms of computational efficiency and accuracy will be compared to the original scheme (leapfrog). Runge-Kutta scheme. The implementation, validation and verification of this approach will comprise the Phase I of this project.

The second approach is to estimate systematic model error and then reduce it by subtracting the estimated error during the course of model integration. For this purpose, we will reproduce the empirical error correc-

tion methods introduced by Danforth et al. (2007) who used model error bias statistics and the singular value decomposition (SVD) technique on the cross-covariance matrix between model state and model error to estimate, respectively, the state-independent and the state-dependent component of the model error in the SPEEDY model. The methods are applied both to the original SPEEDY model and to the newly implemented Lorenz N-cycle. The implementation, validation and verification of a subset of this approach will comprise the Phase II of this project.

3 Phase I: Methods

3.1 Lorenz N-cycle

In 1971 Edward Norton Lorenz devised an incredibly smart time-integration scheme for a system of first-order ordinary differential equations (ODEs) which achieves high-order accuracy and minimal memory consumption at the same time (Lorenz, 1971). Consider a problem of numerically integrating the system of ODEs:

$$\frac{du}{dt} = F(u) \tag{1}$$

where $u = (u_1(t), \ldots, u_M(t))$ is an M-dimensional vector function of t and $F(u) = (F_1(u_1, \ldots, u_M), \ldots, F_M(u_1, \ldots, u_M))$ is a function from $\mathbb{R}^M \to \mathbb{R}^M$. Lorenz (1971) derived two "isomeric" versions for the above problem whose algorithms are shown schematically as pseudo-codes below:

N-cycle A	N-cycle B
$w^0 = 1,$ (2)	$w^0 = 1,$ (7)
$w^k = \frac{N}{N-k}$ $(k = 1, \ldots, N-1)$ (3)	$w^k = \frac{N}{k}$ $(k = 1, \ldots, N-1)$ (8)
do $k = 0, \ldots$	**do** $k = 0, \ldots$
$w \leftarrow w^{\mathrm{mod}(k,N)}$ (4)	$w \leftarrow w^{\mathrm{mod}(k,N)}$ (9)
$G \leftarrow wF(u) + (1-w)G$ (5)	$G \leftarrow wF(u) + (1-w)G$ (10)
$u \leftarrow u + G\Delta t$ (6)	$u \leftarrow u + G\Delta t$ (11)
end do	**end do**

(Here we used the elegant notation introduced in Purser and Leslie (1997) rather than that in the original paper by Lorenz.)

Mathematical idea behind the above algorithms is simple: to "reuse" the previously computed tendencies $(F(u^{k-j}), j = 1, 2, \ldots, k-1,$ where

u^{k-j} denotes the value of u at the $(k-j)$-th step of each cycle) by forming a weighted average of them to produce a tendency which yields the highest order of accuracy after the completion of N-th step, under the constraint that each intermediate step retains at least first order accuracy. In this sense, the Lorenz N-cycle can be regarded as a special type of Runge-Kutta family. In fact, it can be shown that for linear systems, the Lorenz 2-cycles, both version A and B, are equivalent to the Heun scheme with time step $2\Delta t$ and the Lorenz 4-cycles, both version A and B, are equivalent to the classical 4-step 4th-order Runge-Kutta scheme (Section 3.2) with time step $4\Delta t$.

The two versions differ only in the choice of the "weight" coefficients w^k. Lorenz (1971) showed that, for a case where F is linear with respect to u, for any positive integer N, the both versions yield numerical solution of N-th order every N time steps, although in the intermediate steps, the accuracy is only of first order. If F is nonlinear, the accuracy of N-cycle reduces to 2nd order for $N \geq 3$. However, for $N = 3$ and for $N = 4$, the N-th order term in the truncation error of the versions A and B can be shown to be of same magnitude with opposite signs. Thus, for $N = 3$ and for $N = 4$, N-th order accuracy can be attained by running both A and B cycles simultaneously and averaging the predictions, at the expense of doubling the computational cost both in speed and memory consumption.

In order to avoid doubling of computational costs, Lorenz (1971) proposed to use the versions A and B in a suitably designed alternating sequence, based on the intuition that the errors the both versions should tend to cancel each other. In fact, he showed that, for $N = 3$, true 3rd order accuracy can be retained even for a nonlinear case by alternating versions A and B. Likewise, full 4th order accuracy can be achieved for $N = 4$ by forming a $4N(= 16)$-cycle of A,B,B,A. In this project, we will implement this $4N(= 16)$-cycle version and call it by the Lorenz N-cycle for convenience.

Advantages of Lorenz N-cycle

The principal advantage of Lorenz N-cycle beside the high-order accuracy is its computational efficiency, both in terms of speed and memory consumption. As is clear from the pseudo-code listed above, the scheme requires only one evaluation of $F(u)$ per time step which, in most cases, is the most computationally demanding part of the algorithm. Also, the scheme consumes only $2M$ words of memory. Thus, the Lorenz N-cycle has the same computational cost as the widely-used leapfrog scheme. Compared to the 4th-order Runge-Kutta scheme which is very accurate but is impractically

too expensive for the purpose of AGCMs, the Lorenz N-cycle consumes less than half memory and can run 4 times faster.

Another great advantage of the Lorenz N-cycle is the absence of computational modes: the Lorenz N-cycle, being a two time-level method (i.e., the state at time t alone completely determines the state at time $t + \Delta t$) rather than a three time-level method like leapfrog scheme, does not suffer from the presence of computational mode. This feature proves to be particularly useful for nonlinear systems for which computational modes tend to grow causing divergence of numerical solution from the physical solution.

Being two-time level scheme also facilitates the initialization process for which, in the case of three or more time-level schemes, a special treatment of the very first step(s) are required.

Despite these practical merits, the Lorenz N-cycle scheme has remained "forgotten". Although there are some oceanic models which uses this method as its temporal discretization, and Purser and Leslie (1997) developed and tested a scheme which they devised inspired by the Lorenz N-cycle scheme, no direct application has been made to AGCMs.

3.2 4th-order Runge-Kutta scheme

As a reference, we will also implement the tried-and-tested classical 4th-order Runge-Kutta scheme. In a pseudo-code, its algorithm is:

$$
\begin{aligned}
&\textbf{do } k = 0, \ldots, \\
&\quad h_1 \leftarrow F(u), \quad v \leftarrow u + \tfrac{\Delta t}{2} h_1 && (12)\\
&\quad h_2 \leftarrow F(v), \quad v \leftarrow u + \tfrac{\Delta t}{2} h_2 && (13)\\
&\quad h_3 \leftarrow F(v), \quad v \leftarrow u + \Delta t h_3 && (14)\\
&\quad h_4 \leftarrow F(v) && (15)\\
&\quad u \leftarrow u + \tfrac{\Delta t}{6} \left(h_1 + 2h_2 + 2h_3 + h_4\right) && (16)\\
&\textbf{end do}
\end{aligned}
$$

As we can clearly see, this algorithm requires 4 evaluations of F per time step. The above algorithm consumes $6M$ words of memory (M words for each of u, v, h_1, h_2, h_3 and h_4), but the memory consumption can be reduced to $4M$ words by adopting the following operation rearrangements, for which

scheme	leapfrog with R/A filter	N-cycle ($N \leq 4$)	4th-order Runge-Kutta
accuracy	$O(\Delta t)$	$O((N\Delta t)^N)$	$O(\Delta t^4)$
memory consumption	$2M$	$2M$	$4M$
# of F-evaluation(s)	1	1	4

Table 1: The accuracy, memory consumption and the number of F-evaluations per time step of each schemes. Note that "R/A filter" signifies the Robert-Asselin filter defined by Eq.(45) in Section 3.4.

we have only to store u, v, h, p:

$$
\begin{aligned}
&\textbf{do } k = 0, \ldots, \\
&\quad h \leftarrow F(u),\ p \leftarrow h, \qquad v \ \leftarrow u + \tfrac{\Delta t}{2} h && (17)\\
&\quad h \leftarrow F(v),\ p \leftarrow p + 2h, \quad v \ \leftarrow u + \tfrac{\Delta t}{2} h && (18)\\
&\quad h \leftarrow F(v),\ p \leftarrow p + 2h, \quad v \ \leftarrow u + \Delta t h && (19)\\
&\quad h \leftarrow F(v),\ p \leftarrow p + h && (20)\\
&\quad u \ \leftarrow u + \tfrac{\Delta t}{6} p && (21)\\
&\textbf{end do}
\end{aligned}
$$

The accuracy, memory consumption and the number of F-evaluations per time step ($\sim$ CPU cost) of each schemes are summarized in Table 1.

3.3 Semi-implicit method

3.3.1 Introduction

The equations solved by the AGCMs are stiff: the external inertia-gravity waves, which are contained in the solution of these equations but of little meteorological interest, exhibit very fast phase speed (approximately ~ 300 m/s), whereas, other waves such as internal inertia-gravity waves or Rossby waves, which are relevant to the actual weather phenomena exhibit an order-of-magnitude smaller phase speed. In order to resolve this stiffness problem, most AGCMs adopt a so-called semi-implicit scheme which treats terms responsible for external gravity waves implicitly and other terms explicitly (Robert, 1969). However, no semi-implicit modification has been given in previous studies to the Lorenz N-cycle. In this subsection, we propose a simple semi-implicit modification to the Lorenz N-cycle and analyze its accuracy and stability. Similar analysis is also performed to a semi-implicit version of 4th-order Runge-Kutta scheme.

3.3.2 Semi-implicit leapfrog

To clarify the concept of semi-implicit method, we first outline the formulation of the semi-implicit version of the leapfrog scheme which is adopted by the SPEEDY model.

Consider integrating the following equation:

$$\frac{du}{dt} = F^E(u) + L^I u \tag{22}$$

where $F^E : \mathbb{R}^M \to \mathbb{R}^M$ is a nonlinear function and L^I is a $M \times M$ matrix. It is assumed that the term $L^I u$ is responsible for the fast external inertia-gravity waves. The semi-implicit modification to the leapfrog scheme takes the form:

$$\frac{u^{n+1} - u^{n-1}}{2\Delta t} = F^E(u^n) + L^I \left(\alpha u^{n+1} + (1-\alpha)u^{n-1}\right) \tag{23}$$

where $0 \leq \alpha \leq 1$ is a "centering factor". $\alpha = 1/2$ corresponds to the centered Crank-Nicolson scheme, $\alpha = 1$ the backward Euler, and $\alpha = 0$ explicit Euler.

To solve Eq.(23) for u^{n+1}, let us first denote

$$\delta u = \frac{u^{n+1} - u^{n-1}}{2\Delta t} \tag{24}$$

and express u^{n+1} on the right hand side as $u^{n+1} = u^{n-1} + 2\Delta t \delta u$. Then, we have,

$$\delta u = F^E(u^n) + L^I u^{n-1} + 2\alpha \Delta t L^I \delta u \tag{25}$$

$$\Leftrightarrow \delta u = (I - 2\alpha\Delta t L^I)^{-1}(F^E(u^n) + L^I u^{n-1}) \tag{26}$$

Once δu is obtained, the integration can be completed by

$$u^{n+1} = u^{n-1} + 2\Delta t \delta u \tag{27}$$

Namely, to solve Eq.(23) for u^{n+1}, we first evaluate the nonlinear tendency $F^E(u^n)$ at the central step, and then evaluate and add the linear tendency $L^I u^{n-1}$ at the older step. We then multiply it by the inverse matrix $(I - 2\alpha\Delta t L^I)^{-1}$ and finally integrate the equation by Eq.(27). Note that the matrix inversion for $(I - 2\alpha\Delta t L^I)^{-1}$ needs to be carried out only once for the whole integrations because the matrix is constant and thus can be stored and reused.

3.3.3 Semi-implicit Lorenz N-cycle

Our motivation of using Lorenz N-cycle is in its computational economy, both in the amount of computation and the consumption of memory. Thus, in designing a semi-implicit version, we seek to reserve these advantages. A straightforward semi-implicit modification to the Lorenz N-cycle which retains the economy of memory usage is to apply tendency-modification similar to Eq.(26) on each step of the N-cycle:

$$w^0 = 1, \tag{28}$$
$$w^k = \tfrac{N}{k} \ (k = 1, \ldots, N-1), \tag{29}$$
do $k = 0, \ldots$
$$w \leftarrow w^{\mathrm{mod}(k,N)} \tag{30}$$
$$G \leftarrow wF^E(u) + (1-w)G \tag{31}$$
$$\delta u = (I - \alpha \Delta t L^I)^{-1}(G + L^I u) \tag{32}$$
$$u \leftarrow u + \Delta t \delta u \tag{33}$$
end do

Later in this subsection, we show that, by taking $\alpha = 1/2$, this scheme becomes of second order and has a reasonable stability characteristics.

3.3.4 Semi-implicit 4th-order Runge-Kutta

A naïve formulation of semi-implicit 4th-order Runge-Kutta scheme is the following:

do $k = 0, \ldots,$
$$h_1 \leftarrow F^E(u), \quad v \leftarrow u + \tfrac{\Delta t}{2} h_1 \tag{34}$$
$$h_2 \leftarrow F^E(v), \quad v \leftarrow u + \tfrac{\Delta t}{2} h_2 \tag{35}$$
$$h_3 \leftarrow F^E(v), \quad v \leftarrow u + \Delta t h_3 \tag{36}$$
$$h_4 \leftarrow F^E(v) \tag{37}$$
$$\delta u = (I - \alpha \Delta t L^I)^{-1} \cdot \frac{1}{6}(h_1 + 2h_2 + 2h_3 + h_4) \tag{38}$$
$$u \leftarrow u + \Delta t \delta u \tag{39}$$
end do

One can also formulate a semi-implicit 4th-order Runge-Kutta scheme based on the memory-efficient version (Eq.(17-21)) in an exactly same way. Unfortunately, a linear analysis reveals that this scheme is only of first order regardless of the choice of α.

3.3.5 Accuracy and Stability analysis

Following Durran (1991), we examine the stability of the above semi-implicit schemes by applying them to the following linear equation:

$$\frac{du}{dt} = F^E(u) + L^I u \tag{40}$$

with

$$F^E(u) = i\omega_L u,\ L^I = i\omega_H. \tag{41}$$

We can find the truncation errors of these schemes by carrying out the algorithms. The truncation errors of the semi-implicit Lorenz N-cycle and the 4th-order Runge-Kutta are, respectively,

$$\frac{u^N - u^{\text{Exact}}}{u^0} = \frac{1}{2N}(1-2\alpha)\omega_H(\omega_H+\omega_L)(N\Delta t)^2 + O(\Delta t^3) \tag{42}$$

$$\frac{u^1 - u^{\text{Exact}}}{u^0} = \frac{1}{2}\omega_H\left((1-2\alpha)\omega_H - 2(\alpha-1)\omega_L\right)\Delta t^2 + O\left(\Delta t^3\right) \tag{43}$$

For the Lorenz N-cycle, the semi-implicit scheme can be of second order by taking $\alpha = 1/2$. On the other hand, Runge-Kutta scheme can never be of second order: it is only of first order.

Stability of these schemes for each ω_L and ω_H can be visualized by plotting the modulus of corresponding amplification factor $|A|$ as a function of (ω_L, ω_H). The scheme is unstable in the region on (ω_L, ω_H)-plane where $|A| - 1$ is positive.

Figure 1 shows the contour plots of $|A|$ for the semi-implicit leapfrog scheme (with R/A filter), semi-implicit Runge-Kutta 4th-order scheme and the semi-implicit Lorenz 1,2,3,4,5,6-cycle. All schemes use Crank-Nicolson scheme for their semi-implicit part. The $|A| = 1$ contours which divide stable and unstable regions are drawn in thick black lines and unstable regions are shaded in gray. Semi-implicit leapfrog is stable if $\omega_L < \omega_H$, a condition which is always met for practical purposes. Interestingly, the semi-implicit version of 4th-order Runge-Kutta is stable if the slow wave and fast wave are in the same direction ($\omega_L > 0, \omega_H > 0$), while it is stable if they are in the opposite direction ($\omega_L < 0, \omega_H > 0$). In meteorological applications, fast waves (gravity waves) can propagate in all directions, which makes the semi-implicit Runge-Kutta scheme inapplicable. In fact, we confirmed that the SPEEDY model with semi-implicit Runge-Kutta scheme blows up even with very small time steps.

scheme	leapfrog with R/A filter	N-cycle ($N \leq 4$)	4th-order Runge-Kutta
formulation	Eq.(23)	Eq.(28)–(33)	Eq.(34)–(39)
stability	Stable	Good for $N = 4$	Unstable
formal accuracy	$O(\Delta t)$	$O(\Delta t^2)$	$O(\Delta t^2)$
accuracy for nonlinear terms	$O(\Delta t)$	$O((N\Delta t)^N)$	$O(\Delta t^4)$
memory consumption	$2M$	$2M$	$4M$
# of F-evaluation(s)	1	1	4

Table 2: The stability, accuracy, memory consumption and the number of F-evaluations per time step of each semi-implicit schemes. All schemes are assumed to use Crank-Nicolson scheme for semi-implicit part. Note that "R/A filter" signifies the Robert-Asselin filter defined by Eq.(45) in Section 3.4.

The stability region of semi-implicit Lorenz N-cycle enlarges as N gets larger until it becomes 5. For $N \geq 5$, in contrast, the scheme becomes more unstable as N gets larger. Lorenz 4-cycle exhibits a reasonably large stability region, albeit much smaller than that of the leapfrog.

As Lorenz 4-cycle was found to be the most stable, in the implementation and verification, we did not try 3-cycle.

Also, as the 4th-order Runge-Kutta was found to be unstable, and the purpose of implementing Runge-Kutta is only to use it as a reference, I chose to run the explicit Runge-Kutta with a very small time-step (1 minutes in contrast to the default 40 minutes).

The stability, accuracy and computational efficiency of each semi-implicit schemes are summarized in Table 2. In designing the semi-implicit versions of Lorenz N-cycle and Runge-Kutta scheme, we sought to retain the same amount of memory consumption and function evaluations as their explicit versions. Hence, their computational efficiency is essentially identical to that of the explicit counterparts listed in Table 1.

3.4 The SPEEDY model

In this project, the Lorenz N-cycle scheme and the Runge-Kutta 4th-order scheme will be implemented to an existing AGCM known as the SPEEDY model (Molteni, 2003). It is a primitive equation model with 8 vertical layers in σ-coordinate whose horizontal discretization is spectral representation with respect to spherical harmonics triangularly truncated at total wave number of 30 (T30). As its temporal discretization, leapfrog scheme is used

for the dynamics but for the physical parametrizations, 1st order Forward Euler scheme with time step of $2\Delta t$ is used. The R/A filter is also applied (Eq.(45) below). Namely, in a pseudo-code, the algorithm is:

do $k = 1, \ldots$

$$u^{+1} \leftarrow u^{-1} + 2\Delta t(F_{\mathbf{Dyn}}(u^0) + F_{\mathbf{Phys}}(u^{-1})) \tag{44}$$

$$u^0 \leftarrow u^0 + \nu(u^{+1} - 2u^0 + u^{-1}) \tag{45}$$

$$t \leftarrow t + \Delta t \tag{46}$$

$$u^{-1} \leftarrow u^0, \quad u^0 \leftarrow u^{+1} \tag{47}$$

end do

where ν is a filtering parameter of R/A filter which is assumed to be positive small constant. In SPEEDY model, it is set to be $\nu = 0.05$. Semi-implicit treatment of the fastest external gravity waves is also applied which enables stable integration with a large value of Δt by circumventing the CFL condition associated with external gravity waves. The default time step for the SPEEDY model is $\Delta t = 40$min.

Details of the prognostic variables (u) and the equations solved by the model, including complete specification of the forcing $F^{\mathbf{Dyn}}(u)$ and an outline of the forcing $F^{\mathbf{Phys}}(u)$ are described in the Appendix 1.

Simplified physical process and coarse resolution enable the SPEEDY model to be integrated very fast. Despite such simplification, this model is able to produce realistic simulations of a wide range of the atmospheric phenomena, including mid-latitude synoptic features, precipitation and climatology.

The SPEEDY model is written in Fortran 77 and can be run on virtually any machine which supports Fortran 77 compiler. A Linux server hosted by the AOSC department will be used in this project.

3.5 Code Validation

For the validation of the N-cycle code, we will exploit the mathematical fact that Lorenz 1-cycle is equivalent to Forward Euler scheme. Since Forward Euler scheme is already included in the SPEEDY model, we can validate our N-cycle code by comparing the outputs of Lorenz 1-cycle and the built-in Forward Euler scheme initiated from the same initial condition. Since these schemes are unstable, we will integrate them only for single time step.

For the 4th-order Runge-Kutta code, we exploit the fact that, for a linear equation, single step integration of Runge-Kutta with time step of $4\Delta t$ is equivalent to 4-step integration of Lorenz 4-cycle with the time step of Δt.

Therefore, the Runge-Kutta code can be validated by first eliminating all nonlinear terms from SPEEDY model and then, comparing single step integration of Runge-Kutta code with time step of $4\Delta t$ with 4-step integration of Lorenz 4-cycle with the time step of Δt.

The results of code validation are described in Section 4.2.

3.6 Verification: Dynamical Core test

Verification of the implementation is conducted by carrying out the bench mark for AGCMs called "Jablonowski-Williamson dynamical core test" (Jablonowski and Williamson, 2006). The test consists of two test cases. The first one is the "steady-state test case" in which the model is run from an analytic steady-state initial condition. The validity of the model is judged based on to what extent the model can maintain the steady state.

In the second test case, called the "baroclinic wave test case", the steady-state initial condition is perturbed so that the model, if run from this initial condition, will yield baroclinic wave. Baroclinic waves are the unstable waves which drive extra-tropic synoptic weather disturbances. Hence, successful simulation of baroclinic waves is of crucial importance for AGCMs. Unfortunately, however, no analytic solution is known for the baroclinic wave test case. In lieu of this, Jablonowski and Williamson (2006) provides a reference solution which is generated from four distinct high-resolution (approximately corresponds to 50km-mesh) AGCMs. The uncertainty estimate of the reference solution is also provided which are evaluated as the differences among those high-resolution models. The data of reference solution along with its uncertainty estimate is publicly available from the University of Michigan website:

`http://esse.engin.umich.edu/groups/admg/ASP_Colloquium.php`
`http://www-personal.umich.edu/~cjablono/dycore_test_suite.html`

In order to conduct the above test cases, we first switch-off physical parametrizations from the SPEEDY model. This can be done by simply commenting-out a call to the driver subroutine of the physics package. Second, the orography (mountains) is made flat. Since the orography is implemented as an external input to the SPEEDY model, removal of orography can be done simply by making a new orography file containing zeros for all grids on the Earth.

The Jablonowski-Williamson dynamical core tests also requires that Rayleigh drag, which is imposed at the model top in many models including SPEEDY to suppress spurious reflection of vertically propagating waves, be switched

ExpID	Scheme	Initial Condition	Reference Data
STDY_LF STDY_NCYC STDY_RK4	Leapfrog N-cycle Runge-Kutta 4	Analytic steady-state	Initial condition itself.
BRCL_LF BRCL_NCYC BRCL_RK4	Leapfrog N-cycle Runge-Kutta 4	Steady-state superposed with a disturbance	Reference Solution.

Table 3: Set-ups of the experiments of the dynamical-core tests.

off. Thus, we also switch off Rayleigh drag in the SPEEDY model, which can be done simply by setting the coefficient to zero.

The results of baroclinic test case obtained for the the newly-implemented Lorenz N-cycle and 4th-order Runge-Kutta scheme, along with the original leapfrog scheme, are compared to the provided reference solution. If the newly-integrated schemes are closer or equally close to the reference solution, we conclude that the implementation has been made successfully.

The set-ups of the experiments are summarized in the following table:

The results of the dynamical-core tests are described in Section 4.3.

3.7 Verification: Model Climate

The verification described in the previous section only evaluates the validity of the newly-implemented schemes in a specific configuration. Notably, the verification procedure does not evaluate the performance of the schemes in conjunction with physical parametrizations.

In order to evaluate the full performance of the newly-implemented schemes, we produced and plot model climatology and compare them with the official plots by the developers published at the following web site:

`http://users.ictp.it/~kucharsk/speedy8_clim.html`

The results for the verification with model climate are described in Section 4.5.2.

4 Phase I: Implementation and Results

4.1 Implementation

Although the SPEEDY model is implemented in Fortran77, an old, "petrified" language, thanks to the modular design of the codes, the implementation of the subroutines for Lorenz N-cycle and Runge-Kutta scheme was possible in a quite straightforward and clean way.

The environment necessary to reproduce the results in this report, including the code, initial and boundary data, can be retrieved from the following Google Code site:

`https://code.google.com/p/speedy-lorenz-ncycle/`

The code for subroutines which we implemented ourselves are in `SPEEDY/model/update2` directory. The environment for the code validation is in `SPEEDY/model/validation` directory. The data and code for the verification (Dynamical Core Tests) are in `SPEEDY/model/dyncore_test`.

4.2 Results of Code Validation

This subsection describes the result of code validation described in Section 3.5, The result was successful both for the Lorenz N-cycle and for the 4th-order Runge-Kutta scheme: the outputs exactly agreed, which was confirmed by comparing the binary outputs by using the UNIX's `diff(1)` command.

4.3 Results of the Dynamical Core tests

This subsection describes the results of the dynamical core tests whose procedures are described in Section 3.6.

4.3.1 Steady-state test

Jablonowski and Williamson (2006) proposes two metrics of model performance for the steady-state test. The first metric, $l_2(u(t) - \bar{u}(t))$ evaluates the symmetry-deviations from the zonal average of zonal winds at a given instant:

$$l_2(u(t) - \bar{u}(t)) = \left[\frac{1}{4\pi} \int_0^1 \int_{-\pi/2}^{\pi/2} \int_0^{2\pi} \{u(\lambda, \varphi, \sigma, t) - \bar{u}(\varphi, \sigma, t)\}^2 \cos\varphi d\lambda d\varphi d\sigma \right]^{1/2} \quad (48)$$

where the overbar ($\bar{}$) denote the zonal average.

The second metric, $l_2(\bar{u}(t) - \bar{u}(t=0))$, measures the degradation of the zonal average of the zonal wind with respect to the analytic solution:

$$l_2(\bar{u}(t) - \bar{u}(t=0)) = \left[\frac{1}{2}\int_0^1 \int_{-\pi/2}^{\pi/2} \{\bar{u}(\varphi,\sigma,t) - \bar{u}(\varphi,\sigma,t=0)\}^2 \cos\varphi d\varphi d\sigma\right]^{1/2} \quad (49)$$

For the SPEEDY model, the first metric $l_2(u(t) - \bar{u}(t))$ was found to be zero up to rounding precision, for all of the leapfrog, Lorenz 4-cycle and explicit Runge-Kutta. This means that the SPEEDY model can maintain the all the non-zero wavenumber components of the analytic initial condition. The validity of the wavenumber-zero component is evaluated with the second metric.

Figure 2 shows the second metric $l_2(\bar{u}(t)-\bar{u}(t=0))$ of the leapfrog (with Δt = 20min.), Lorenz 4-cycle (with Δt = 20min.), Lorenz 4-cycle (with Δt = 1min.), and 4th-order Runge-Kutta (with Δt = 1min.). All schemes have virtually identical performance which is far from being "steady". This is because the analytic steady-state solution turns out not to be a steady state for the SPEEDY model. The SPEEDY model, having only 8 vertical layers, imposes very strong horizontal diffusion at the model's top layer which corresponds to lower stratosphere in order enhance numerical stability. This violates the "free atmosphere" assumption which is assumed in the design of the dynamical core test, which makes "analytic steady state" not a steady state.

Figure 3 shows the meridional cross sections of zonal mean zonal wind ($\bar{u}(\varphi,\sigma,t)$) at $t = 0$ and $t = 30$ days for the leapfrog with time step $\Delta t = 20$ min. We can observe that the strong mid-latitude westerlies are significantly decreased and they diffuse into the tropics at the model top ($\sigma = 0.08$) due to the stronger horizontal diffusion. We can also observe that, in the troposphere, especially below $\sigma \sim 0.5$, the zonal wind is pretty much preserved. This pattern of change in zonal wind is shared in all other schemes as well.

4.3.2 Baroclinic wave test

It was found that the SPEEDY with default leapfrog scheme blows up for the baroclinic wave test configuration if the default time step of $\Delta t = 40$ min. is used. For this reason, we used $\Delta t = 20$ min. for the leapfrog and Lorenz 4-cycle throughout this test.

All of the tested schemes, leapfrog with $\Delta t = 20$min. and $\Delta t = 1$min., Lorenz 4-cycle with $\Delta t = 20$min. and $\Delta t = 1$min. and Runge-Kutta scheme with $\Delta t = 1$ min., successfully reproduced a plausible development of baroclinic wave trains followed by realistic weather pattern.

In order to grasp the qualitative features of different schemes, we show in Figure 4, the snapshots of surface pressure at the 9th day for the different schemes. The figure also shows the reference solution provided by Jablonowski and Williamson (2006) which is produced from a high-resolution version of NCAR CAM (an community AGCM developed at the National Center for Atmospheric Research). In the reference solution (top panel), a deep low with a minimum of about 940hPa develops to the east, as well as a weaker low to its west. The leapfrog with $\Delta t = 20$ min. (upper-left panel) fails to reproduce this deep low, showing a minimum of about 970hPa. The leapfrog with smaller time step of $\Delta t = 1$ min. (lower-left panel) is more successful in reproducing the deep low, showing a minimum of about 960hPa. Runge-Kutta with small time step $\Delta t = 1$ min. (upper-right panel) and Lorenz 4-cycle with the larger time step $\Delta t = 20$ min, are also successful in reproducing the low.

Lorenz 4-cycle is clearly the most advantageous in that it alone successfully reproduces the deep low with the larger time step.

In order to quantitatively compare the accuracy of the leapfrog and Lorenz 4-cycle, we computed the RMS errors in surface pressure of them regarding Runge-Kutta with $\Delta t = 1$min. as the truth, which is shown in Figure 5. For both $\Delta t = 20$min. (left) and $\Delta t = 1$min. (right), Lorenz 4-cycle is consistently more accurate than the leapfrog, and the difference is quite dramatic for the smaller time step ($\Delta t = 1$min.). These results clearly confirm the superiority of Lorenz 4-cycle over the leapfrog.

4.4 Order Estimation

As we described in Section 3.3.5, Lorenz N-cycle with Crank-Nicolson semi-implicit scheme is formally of second order. In order to verify this, we conducted the dynamical-core test for a number of time steps Δt and estimated the order of accuracy by plotting L^2 errors of surface pressure against Δt on a log-log plane. The L^2 errors are computed with respect to the reference solution obtained from explicit Runge-Kutta 4-th order scheme with time step $\Delta t = 30$ sec. The results for 3-day forecast are shown in Figure 6. We can observe that, for each scheme, the errors are aligned linearly for small time step ($\Delta t < 200$ sec.) but they begin to saturate as Δt becomes larger. Lorenz 4-cycle of both versions A and B, and their combinations A-

B and A-B-B-A, show almost identical performance. The order of accuracy estimated by least-square fitting to a regression line using small time steps (Δt =60, 120 and 180 sec.) is 1.98 for Lorenz 4-cycle with Crank-Nicolson and 1.12 for Leapfrog with Crank-Nicolson and R/A filter. These results are consistent with our expectation that semi-implicit Lorenz schemes N-cycle with Crank-Nicolson are of second-order and that the semi-implicit leapfrog with Crank-Nicolson and R/A filter is of first-order. Unlike what is claimed in Lorenz (1971), however, alternating the versions A and B of Lorenz 4-cycle did not improve the accuracy of forecast in our case. We found, in fact, alternation of the two versions leads to instability for large Δt. Figure 7, which shows the results for 10-day forecasts, show that alternated versions (AB and ABBA) of Lorenz 4-cycle with Crank-Nicolson show very large errors for larger time steps $\Delta t > 300$ sec. These alternated schemes with $\Delta t = 1200$ sec. blow up if we continue integrating them up to 29 days.

4.5 Inclusion of Physics

Having established the forecast-skill improvement of semi-implicit Lorenz N-cycle under the framework of dynamical-core tests, we then introduced physical parametrizations to the model. In the mid-year report, I reported about the instability of SPEEDY model with Lorenz 4-cycle integrated in conjunction with physical parametrizations. It turned out that this problem was merely due to a bug which I introduced. The bug was fixed during the winter break.

4.5.1 Stability

The largest time step Δt with which the model can be integrated stably is usually not predictable and modelers must resort to trial-and-error, varying time steps and performing long-term integration with each time steps.

In SPEEDY model, the default, semi-implicit leapfrog scheme with Crank-Nicolson and R/A filter, $\Delta t = 40$ min. is the largest time step with which the model can be stably integrated. We tried to find the largest Δt for the semi-implicit Lorenz N-cycle with Crank-Nicolson and found that Δt must be no larger than 15 minutes. This is unfortunate in the sense that the new scheme requires smaller time step than the default scheme.

To confirm the stability, we conducted a 100-year integration of the SPEEDY model with Lorenz 4-cycle with Crank-Nicolson using time step Δt =15 min. and found no evidence instability. 100 years correspond to

3,504,000 time steps which is way larger than the model's degree of freedom.

4.5.2 Comparison of Model Climate

As described in Section 3.7, it is important that the long-term mean of the atmospheric state (called climatology) does not change due to change of time integration scheme. Thus, we computed climatologies for the new and original schemes and statistically compared them using a version of Student's $t-$test Welch's test.

Examined climatologies are for two seasons, one for winter (December, January and February) and the other for summer (June, July and August). The climatologies are computed for the three schemes listed below:

- Leapfrog scheme with Crank-Nicolson semi-implicit with time step Δt =40 min. (the default scheme)
- Lorenz 4-cycle with Crank-Nicolson semi-implicit with time step Δt =15 min.
- Lorenz 4-cycle with Backward-Euler semi-implicit with time step Δt =30 min.

To compute climatologies, we integrated all the models from the same initial condition (state of rest) for 30 years. The first 10 years are discarded as spin-up and the remaining 20 years are used as as samples.

In our significance test for difference of computed climatologies (i.e., sample means), the null-hypothesis is:

- All the climatologies are taken from the same population

We compute the probability of the difference between two sampled climatologies being larger than the computed (observed) difference under null-hypothesis stated above. If this probability is larger than 95%, then the null-hypothesis cannot be rejected, which supports our claim that climatologies are not affected by the change of time integration schemes.

We conducted this statistical test for geopotential height at 500hPa (Z500), mean sea-level pressure (MSLP), temperature at 925hPa, 850hPa, 700hPa (T925, T850, T700), surface temperature, zonal wind at 925hPa (U925), outgoing longwave radiation (OLR), soil wetness, zonal mean zonal wind, and zonal mean temperature. We found no significant differences in all the above meteorological elements for any model for both winter and

summer. As an example, we show the result for winter-time T925 in Figure 8. The climatologies of all models are almost identical and we cannot discern any differences by visual inspection. In fact, the p-values are larger than 0.95 everywhere. Results for all other variables and models are available at:

`http://www.atmos.umd.edu/~dhotta/speedy_clim2/clim.html`

4.6 Phase I: Conclusion

In the Phase I, we designed a semi-implicit version of Lorenz N-cycle and analyzed its stability and accuracy using the "split-frequency" linear equation. It is found, through the stability analysis, that 4-cycle is the most stable and accurate. The semi-implicit scheme is then implemented to the dynamical core of the SPEEDY model. Verification through the dynamical-core test confirmed that, as expected, Lorenz 4-cycle with Crank-Nicolson scheme exhibits 2nd-order accuracy which is higher than the conventional leapfrog scheme with R/A filter.

In contrast to what is claimed in Lorenz (1971), however, alternation of the two versions of the 4-cycle does not improve the accuracy of the scheme but rather leads to destabilization. There may be several possible reason to this contradiction. The amelioration of the scheme by alternation of the two versions relies on the assumption that truncation errors of the two versions have opposite sign and thus tend to cancel each other. Since we introduced semi-implicit modification to the Lorenz N-cycle, there is no guarantee that the two versions still have truncation errors of equal amount with opposite sign. Moreover, Purser (2012, personal communication) pointed out that this cancellation mechanism itself is highly questionable even for fully explicit N-cycle when it is applied to strongly nonlinear systems which possess a number of mutually-interacting modes such as AGCM. We will explore more on this issue to understand the mechanism behind it.

The SPEEDY model with Lorenz 4-cycle is then extended to include physical parametrizations. It was found that Lorenz 4-cycle, despite being more accurate, requires shorter time step than the original leapfrog scheme. The stability of the scheme is confirmed by a very long (100-year) integration. The model climatologies of Lorenz 4-cycle and the original leapfrog were then compared with statistical significance test. No significant differences were detected to any of the meteorological elements, which confirms

that model climatologies are not sensitive to the choice of time integration schemes.

5 Phase II

In Phase II of this project, the training stage of Danforth et al. (2007) is reproduced, both for the original SPEEDY model with leapfrog scheme and for the SPEEDY model with the newly implemented schemes. We planned to address the second stage of Danforth et al. (2007), the estimation-correction stage in Phase II provided that time permit, but we did not have enough time to work on this phase. The outline of plans for Phase III are described in Appendix 2.

5.1 Overall Algorithm

The first step of the training stage of Danforth et al. (2007) is to generate many samples of model errors. First, a 6-hour forecast is performed by running the SPEEDY model with the original leapfrog scheme, denoted $M_{6h}^{LF}(x)$ here after, from the initial condition taken from the National Centers for Environmental Prediction (NCEP)/National Center for Atmospheric Research (NCAR) reanalysis (Kalnay and Coauthors, 1996), which will be denoted $x^{true}(t)$ here after. The discrepancy between the forecast $M_{6h}^{LF}(x^{true}(t))$ and the corresponding atmospheric state from NCEP/NCAR reanalysis $x^{true}(t + 6\text{hours})$, namely

$$\delta x(t) = M_{6h}^{LF}(x^{true}(t)) - x^{true}(t + 6\text{hours}),$$

is regarded as the model error. The 6-hour forecasts are repeated many times using different initial conditions to collect many samples of model errors $\delta x(t)$.

The next step in the training stage is to extract state-independent component, or the bias, of the model error. The bias is estimated as the mean of the sampled model errors $\langle \delta x \rangle$, where the angle brackets denotes averaging over all samples.

The third step is to build the cross-covariance matrix between the model state and the model error:

$$C = \left\langle \left(x^{true}(t) - \langle x^{true} \rangle\right) \left(\delta x(t) - \langle \delta x \rangle\right)^T \right\rangle.$$

In prior to building the cross-covariance matrix, the mean is removed, and they are standardized by dividing them by the standard deviation, both for the model state and the model error.

Finally, to extract the dominant modes of the co-variation of the two quantities, singular value decomposition (SVD) is performed on the matrix C:

$$C = U\Sigma V^T$$

where U and V are square orthogonal matrices and Σ is a diagonal matrix. The right singular vector v_i can be interpreted as the shape of model error which is most likely to be assumed if the anomaly of the model state $(x^{true}(t) - \langle x^{true} \rangle)$ is in the direction of the corresponding left singular vector u_i. Thus, pairs of left and right singular vectors provide valuable information about the characteristics of the model error. In the estimation-correction stage of Danforth et al. (2007), which, was planned to be addressed, time permitted, in Phase III of this project, the singular vectors and the singular values are used to build a regression model with which we can estimate the most likely state-dependent component of model error given the current state of the model.

In this project, the above procedures are repeated for the SPEEDY model with the original leapfrog scheme (M^{LF}), the Lorenz N-cycle scheme (M^{Ncyc}) and that with the 4th-order Runge-Kutta scheme (M^{RK4}).

The algorithm can be summarized as the following:

for models M in $M^{LF}, M^{Ncyc}, M^{RK4}$, **do**
  do $i = 1, 2, \ldots$
    - perform forecast $x^{fcst}(t_i + 6\text{hours}) = M_{6h}(x^{true}(t_i))$
    - compute the error $\delta x(t_i) = x^{fcst}(t_i + 6\text{hours}) - x^{true}(t_i + 6\text{hours})$
  end do
  - compute the bias $\langle \delta x \rangle = \frac{1}{\#i} \sum_i \delta x(t_i)$
  - compute the climatology $\langle x^{true} \rangle = \frac{1}{\#i} \sum_i x^{true}(t_i)$
  - compute the cross-covariance matrix
    $C = \frac{1}{\#i - 1} \sum_i \left(x^{true}(t_i) - \langle x^{true} \rangle\right) \left(\delta x(t_i) - \langle \delta x \rangle\right)^T$
  perform SVD for C: $C = U\Sigma V^T$
end do

The NCEP/NCAR reanalysis data which is used as truth are publicly available from NCEP or Earth System Research Laboratory (ESRL) of National Oceanic and Atmospheric Administration (NOAA) at:

`http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html`
`http://www.nomad3.ncep.noaa.gov/ncep_data/`

5.2 Covariance Localization

In the SPEEDY model, the size of cross-covariance matrix C becomes huge (approximately $150,000 \times 150,000$) for which performing SVD is prohibitively expensive. In order to reduce the size of matrix, Danforth et al. (2007) introduced the following "covariance localization":

1. Neglect correlation between different meteorological elements
2. Neglect further correlation between the same meteorological element at different vertical levels. These two restrictions introduce block-diagonal structure to the matrix C.
3. Apply horizontal localization by setting zero to covariances among two grid points whose distance exceeds 3,000km.
4. Perform SVD independently to each blocks of C.

The SPEEDY model has 96 grids in longitude and 48 grids in latitude. Thus, the above localization reduces the size of the matrix to $(96 \cdot 48) \times (96 \cdot 48) = 4608 \times 4608$.

5.3 Choice of SVD Algorithm

Although the covariance localization described in the previous section significantly reduces the size of matrix, 4608×4608 is still very large and SVD on this size of matrix can be computationally expensive. In fact, performing SVD using LAPACK routine for dense matrices (`DGESVD`) on this size takes 4 hours under our environment. However, since each submatrix of C is sparse due to the horizontal localization applied in Step 3, it is possible to reduce computational cost by using Krylov-type iterative SVD algorithms for sparse matrices. In this project, we used the following simple version of Lanczos algorithm.

First, we attribute the SVD on C to the eigendecomposition on the symmetric matrix

$$A := CC^T$$

, and perform eigendecomposition of A using Lanczos method. Then, the left singular vectors of C can be obtained as eigenvectors of A. Similarly, the singular values of C are obtained as square roots of eigenvalues of A.

Lanczos method uses the following fact to obtain approximate eigenvalues for the $n \times n$ matrix A (where $n = 4608$ in our case):
Let $m \ll n$ and suppose we can obtain $m \times m$ symmetric tridiagonal matrix T which satisfies

$$T = V^T A V \tag{50}$$

where V is a $n \times m$ matrix whose column vectors are orthonormal. Then, the eigenvalues of this smaller tridiagonal matrix are approximation to the eigenvalues of A, and the eigenvectors can be approximated by $Vu_i,\ i = 1, \dots m$ where $\{u_i\}$ are the eigenvectors of T. Since T is tridiagonal and thus upper Hessenberg, its eigendecomposition can be performed efficiently using QR algorithm.

The algorithm for triangulation can be derived as follows:
Let $T_{ii} = a_i, T_{i+1,i} = T_{i,i-1} = b_i$. Then, we have

$$\begin{aligned}
Av_1 &= a_1 v_1 + b_1 v_1 && (51)\\
Av_2 &= b_1 v_1 a_2 v_2 + b_2 v_3 && (52)\\
&\vdots && (53)\\
Av_i &= b_{i-1} v_{i-1} a_i v_i + b_i v_{i+1} && (54)\\
&\vdots && (55)\\
Av_n &= b_{n-1} v_{n-1} + a_n v_n && (56)
\end{aligned}$$

By taking inner product of the i-th line and v_i, we have

$$a_i = v_i^T A v_i \tag{57}$$

Also, we have

$$\begin{aligned}
b_{i+1} v_{i+1} &= Av_i - (b_i v_{i-1} + a_i v_i) && (58)\\
b_{i+1} &= \|Av_i - (b_i v_{i-1} + a_i v_i)\| && (59)
\end{aligned}$$

Thus, starting from an arbitrary v_1, we can compute a_1 from (57), b_1 from (59) and v_2 from (58); we can repeat this process until we get v_n.

Lanczos method only gives approximate eigenvalues and eigenvectors. Thus, it is important to bear in mind that the accuracy of estimation depends on the number of iterations m. In this project, we are interested only in the first ~ 20 dominant modes. Thus, we chose the number of iterations to $m = 100$. To perform the eigendecomposition of the smaller $m \times m$ matrix T, we used a LAPACK subroutine `DSYEVX` for dense symmetric matrix.

5.4 Difference in Procedures between this project and Danforth et al. (2007)

By carefully reading Danforth et al. (2007) I realized, in early April, that the procedure which I proposed for this project (which is described in Section 5.1) differs in some part from that of Danforth et al. (2007). Danforth et al. (2007) follows the following procedure:

1. Produce error samples $\{\delta x\}$ from the original model M
2. Compute bias $\langle \delta x \rangle$
3. De-bias the original model M by incorporating nudging to $-\langle \delta x \rangle$ to yield M^+
4. Resample errors $\{\delta x^+\}$ using M^+
5. Extract diurnal systematic errors by performing EOF analysis on $\{\delta x^+\}$
6. The de-biased model M^+ is again modified by incorporating nudging to the negative of diurnal systematic error estimated in the previous step to yield M^{++}
7. Resample errors again to produce $\{\delta x^{++}\}$ using M^{++}
8. Perform SVD on cross-covariance matrix between model errors $\{\delta x^{++}\}$ and anomalous model states

In our algorithm, the steps from 3. to 7. are missing. Comparing our results with those of Danforth et al. (2007) allows us to understand the effect of two-stage de-biasing (steps 3. and 5.) which was applied in Danforth et al. (2007) but not in our procedure.

5.5 Implementation and Platform

Programs implemented in Phase II are the following:

1. a program to compute the bias
2. a program to compute the covariance matrix C
3. a program to perform SVD on C

These programs are implemented in partially in Fortran 90 and partially in NCL (NCAR Command Language) on a Linux server hosted on the network of the AOSC department.

5.6 Validation

Computation of bias is validated by comparing the results with Danforth et al. (2007).

SVD code which is implemented based on Lanczos algorithm is validated by checking the orthogonality of singular vectors.

5.7 Verification

In the verification of Phase I, the accuracy of Lorenz N-cycle will be tested only for the dynamical core of the AGCM. In the verification process of Phase II, its accuracy as a comprehensive AGCM including physical parametrizations will be tested by comparing the amplitude of model error extracted as the bias and covariance with those for the original model. If the amplitudes are smaller for the Lorenz N-cycle than for the original leapfrog scheme, it means that the model with Lorenz N-cycle is more accurate.

5.8 Results of Bias Estimation

Figure 9 and 10 show the 6-hour forecast errors of, respectively, 200hPa zonal wind (U200) and 850hPa temperature (T850). Results of both leapfrog and N-cycle look almost identical to the result of Danforth et al. (2007). Resemblance of our result for leapfrog and Danforth et al. (2007) validates our implementation. Resemblance of our results for leapfrog and N-cycle implies that, in terms of bias, no improvement can be attained by the improvement of time integration scheme. This is perhaps due to the fact that bias of SPEEDY's model errors are dominated by defects of physical parametrizations and/or coarse spatial resolution.

Similar results are obtained for other meteorological elements such as meridional wind, specific humidity and precipitation.

5.9 Results: SVD analysis

This section presents the results of the second step of Phase II, namely, SVD analysis. The purpose of SVD analysis is to extract pairs of dominant modes of co-variation between the anomalous model states and the model errors. The pairs of dominant spatial patterns can be conveniently represented by "correlation maps".

In this section, we first show the result of validation of SVD code in subsection 5.9.1. We then present the main results, the correlation maps, in subsection 5.9.2.

5.9.1 Code Validation

Figure 11 shows the inner products of left and right singular vectors of the cross-covariance matrix for zonal wind at $\sigma = 0.2$ of leapfrog scheme obtained using Lanczos method with 100 iterations. The shading of (i, j) grid represents $u_i^T u_j$ (or $v_i^T v_j$). Theoretically, this matrix is expected to coincide with identity matrix. We can observe that, for both left and right singular vectors, orthonormality property holds quite well. The orthogonality is particularly accurate for the dominant modes ($i, j < 20$); most of the off-diagonal components are smaller than 10^{-12} for this domain. The inner products become larger as i or j increases but they never exceed 10^{-9}. These results support the validity of the implementation of SVD.

5.9.2 Correlation maps

The obtained results were quantitatively similar for all meteorological elements at all vertical levels. Thus, we show only the results of temperature at $\sigma = 0.95$ level (T0.95).

Figure 12 shows the homogeneous (contour) and heterogeneous (shade) right correlation maps for the first five dominant modes for T0.95 of leapfrog scheme. The corresponding amplitude timeseries $a_k(t)$ and $b_k(t)$ (see Appendix II for their definitions) are also shown, with solid and dashed lines, respectively. Figure 13 shows the same plots for the next five dominant modes. Figure 14 and 15 shows the same plots for Lorenz 4-cycle.

Figure 12–13 and 14–15 are almost identical, which means that the change in time integration scheme has little effect on the linear relation between the model error and the anomalous model state.

Consistent with what was found in Danforth et al. (2007), the heterogeneous and homogeneous correlation maps for each mode exhibit very similar spatial patterns, indicating that model errors can be successfully estimated from the anomalous model state. An interesting feature that we can discern from Figure 12 or 13 is that most of dominant modes are concentrated around the two poles, which is not consistent with Danforth et al. (2007). This is perhaps related to the strong diurnal oscillations of $a_k(t)$ which are evident in all modes. Unlike Danforth et al. (2007) who successfully eliminated diurnal oscillation in the model errors, we left diurnal biases in the model errors "as is". The diurnal model errors show clear large-scale horizontal structure with zonal wavenumber 1. However, in constructing the cross-covariance matrix, we applied horizontal localization which destroys zonal-wavenumber-1 structure except near the poles. This is perhaps why

the dominant modes exaggeratedly pick up spatial patterns which are localized at the two poles.

To summarize, conclusions that we can draw from these results are:

- Changing the time integration scheme has little effect on the dominant linear relation between the model state and the model error
- Whether or not correct the diurnal bias of model error has large impact on the dominant modes

6 Schedule and Milestones

The planned and actual schedules for Phase I and the corresponding milestones are as follows:

6.1 Planned Schedule

- Phase I:
  - Implement Lorenz N-cycle and 4th-order Runge-Kutta scheme to the SPEEDY model: September through end of November
  - Write the mid-year report, prepare the oral presentation: December
  - Switch-off physical parametrizations and prepare flat orography: January
  - Perform the dynamical core tests: early February
- Phase II:
  - Generate initial values from the NCEP/NCAR reanalysis (or from SPEEDY with Runge-Kutta run without physics): end of February
  - Build the bias and covariance matrix: March
  - Code and test a program for SVD: April
  - Compare the model errors for M^{LF} with those published in Danforth et al. (2007): early May
  - Compare the model errors for M^{LF} and M^{Ncyc}, M^{RK4}: mid-May
  - Write the final report and prepare for the oral presentation.

6.2 Actual schedule

- Phase I:
  - Formulate semi-implicit N-cycle: October
  - Implement Lorenz N-cycle and 4th-order Runge-Kutta scheme to the SPEEDY model: November
  - Switch-off physical parametrizations and prepare flat orography: December
  - Perform the dynamical core tests: December
  - Write the mid-year report, prepare the oral presentation: December
  - Coded a bug, and fixed it: January
  - Compare climatologies and perform statistical significance test: February
- Phase II:
  - Generate initial values from the NCEP/NCAR reanalysis: end of February
  - Compute the bias: March
  - Plot the bias and compare it with Danforth et al. (2007): April
  - Code and test a program for SVD: May
  - Compare the model errors for the new and the original schemes: May
  - Write the final report, May.

6.3 Milestones

- Phase I:
  - The SPEEDY model with Lorenz N-cycle and 4th-order Runge-Kutta scheme both runs: November
  - Complete validation of Lorenz N-cycle and 4th-order Runge- Kutta scheme: December
- Phase II:
  - Complete computation of error bias: March
  - Complete computation of error covariance: April
  - Complete validation of the SVD program: May

7 Deliverables

Deliverables of Phase I are:

1. subroutines for Lorenz N-cycle and 4th-order Runge-Kutta of the SPEEDY model (c.f. Sections 3.1, 3.2 and 4.1) (**delivered**)

2. results of Jablonowski-Williamson dynamical core test cases for the SPEEDY model, both with the original leapfrog scheme and the newly implemented schemes. (c.f. Section 4.3) (**delivered**)

3. plots of model climatology for the SPEEDY model, both with the original leapfrog scheme and the newly implemented schemes.(c.f. Section 3.7 and 4.5.2) (**delivered**)

Deliverables of Phase II are:

1. an archive of the model errors (**delivered**)

2. model error bias (c.f. Section 5.8) (**delivered**)

3. pairs of singular vectors for the model state and the model error along with the corresponding singular values (c.f. Section 5.9) (**delivered**)

4. code for performing SVD. (c.f. Section 5.3) (**delivered**)

Deliberables also includes

1. two presentations, mid-year and final

2. two reports, mid-year and final

3. and the project proposal.

References

Asselin, R., 1972: Frequency filter for time integrations. *Mon. Wea. Rev.*, **100**, 487–490.

Danforth, M., C, E. Kalnay, and T. Miyoshi, 2007: Estimating and correcting global weather model error. *Mon. Wea. Rev.*, **135**, 281–299.

Durran, D. R., 1991: The third-order adams-baashforth metod:h an attractive alternative to leapfrog time differencing. *Mon. Wea. Rev.*, **119**, 702–720.

Jablonowski, C. and D. L. Williamson, 2006: A baroclinic instability test case for atmospheric model dynamical cores. *Q. J. R. Meterol. Soc.*, **132**, 2943–2975.

Kalnay, E. and Coauthors, 1996: The ncep/ncar 40-year reanalysis project. *Bull. Amer. Meteor. Soc.*, **77**, 437–471.

Lorenz, N., E., 1971: An n-cycle time-differencing scheme for step-wise numerical integration. *Mon. Wea. Rev.*, **119**, 1612–1623.

Matsuno, T., 1966: Numerical integrations of the primitive equations by a similated backward difference method. *J. Meterol. Soc. Japan*, **44**, 76–84.

Molteni, F., 2003: Atmospheric simulations using a gcm with simplified physical parametrizations. i: Model climatology and variability in multi-decadal experiment. *Climate Dyn.*, **20**, 175–191.

Purser, J., R. and L. M. Leslie, 1997: High-order generalized lorenz n-cycle schemes for semi-lagrangian models employing second derivatives in time. *Mon. Wea. Rev.*, **125**, 1261–1276.

Robert, A. J., 1969: The integration of a spectral model of the atmosphere by the implicit method. *Proc. WMO-IUGG Symp. on Numerical Weather Prediction*, Tokyo, Japan, Japan Meteorological Agency, Vol. VII, 19–24.

Teixeira, J., C. A. Reynolds, and K. Judd, 2007: Time step sensitivity of nonlinear atmospheric models: Numerical convergence, truncation error growth, and ensemble design. *J. Atmos. Sci.*, **64**, 175–189.

A1 Appendix 1: Detailed Description of the SPEEDY model

A1.1 The equations

The SPEEDY model is based on the nonlinear primitive equations for moist atmosphere on a vertical σ-coordinate with spherical geometry. Its prognostic variables are horizontal vorticity ζ, horizontal divergence D, temperature T, specific humidity q and the natural logarithm of surface pressure $\pi \equiv \ln p_s$. These equations are vertically discretized by a finite differencing and then horizontally discretized by Galerkin spectral method with respect to spherical harmonics. Namely, horizontal structure of any variable on a given vertical level is represented expansion coefficients of spherical harmonics, and horizontal differentiations are performed with respect to such coefficients. Thus, the actual prognostic variables in the computer code (expressed as u in the pseudo-codes in Section 3) are therefore the expansion coefficients of the above 5 prognostic variables on specified vertical levels.

The set of prognostic equations can be written explicitly as follows:

$$\frac{\partial \zeta}{\partial t} = \frac{1}{a(1-\mu^2)}\frac{\partial \mathcal{F}_V}{\partial \lambda} - \frac{1}{a}\frac{\partial \mathcal{F}_U}{\partial \mu} - K_\nu\left(\nabla_\sigma^4 - \frac{2^2}{a^2}\right)\zeta + \left.\frac{d\zeta}{dt}\right|_{\mathbf{Phys}} \tag{A.1}$$

$$\frac{\partial D}{\partial t} = \frac{1}{a(1-\mu^2)}\frac{\partial \mathcal{F}_U}{\partial \lambda} + \frac{1}{a}\frac{\partial \mathcal{F}_V}{\partial \mu} - \nabla_\sigma^2(\Phi + R\bar{T}\pi + KE) - K_\nu\left(\nabla_\sigma^4 - \frac{2^2}{a^2}\right)D \tag{A.2}$$

$$\begin{aligned}\frac{\partial T}{\partial t} = &- \frac{1}{a(1-\mu^2)}\frac{\partial UT'}{\partial \lambda} - \frac{1}{a}\frac{\partial VT'}{\partial \mu} + T'D \\ &-\dot{\sigma}\frac{\partial T}{\partial \sigma} + \kappa T\left(\frac{\partial \pi}{\partial t} + \boldsymbol{v}_H \cdot \nabla_\sigma \pi + \frac{\dot{\sigma}}{\sigma}\right) \\ &-K_h\left(\nabla_\sigma^4 - \frac{2^2}{a^2}\right)T + \left.\frac{dT}{dt}\right|_{\mathbf{Phys}}\end{aligned} \tag{A.3}$$

$$\frac{\partial q}{\partial t} = -\boldsymbol{v}_H \cdot \nabla_\sigma q - \dot{\sigma}\frac{\partial q}{\partial \sigma} + \left.\frac{dq}{dt}\right|_{\mathbf{Phys}} \tag{A.4}$$

$$\frac{\partial \pi}{\partial t} = -\boldsymbol{v}_H \cdot \nabla_\sigma \pi - \frac{\partial \dot{\sigma}}{\partial \sigma} - D \tag{A.5}$$

where

$$\theta \equiv T\,(p/p_0)^{-\kappa} \tag{A.6}$$

$$\kappa \equiv R/C_p \tag{A.7}$$

$$\Phi \equiv gz \tag{A.8}$$

$$= \Phi|_{\sigma=1} - \int_1^\sigma \frac{RT}{\sigma} d\sigma \tag{A.9}$$

$$\pi \equiv \ln p_S \tag{A.10}$$

$$\dot{\sigma} \equiv \frac{d\sigma}{dt} \tag{A.11}$$

$$\mu \equiv \sin\varphi \tag{A.12}$$

$$U \equiv u\cos\varphi \tag{A.13}$$

$$V \equiv v\cos\varphi \tag{A.14}$$

$$\zeta \equiv \frac{1}{a(1-\mu^2)}\frac{\partial V}{\partial\lambda} - \frac{1}{a}\frac{\partial U}{\partial\mu} \tag{A.15}$$

$$D \equiv \frac{1}{a(1-\mu^2)}\frac{\partial U}{\partial\lambda} + \frac{1}{a}\frac{V}{\mu} \tag{A.16}$$

$$\mathcal{F}_U \equiv (\zeta+f)V - \dot{\sigma}\frac{\partial U}{\partial\sigma} - \frac{RT'}{a}\frac{\partial\pi}{\partial\lambda} \tag{A.17}$$

$$\mathcal{F}_V \equiv -(\zeta+f)U - \dot{\sigma}\frac{\partial V}{\partial\sigma} - \frac{RT'}{a}(1-\mu^2)\frac{\partial\pi}{\partial\mu} \tag{A.18}$$

$$KE \equiv \frac{U^2+V^2}{2(1-\mu^2)} \tag{A.19}$$

$$\boldsymbol{v}_H\cdot\nabla \equiv \frac{u}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial}{\partial\varphi}\right)_\sigma$$

$$= \frac{U}{a(1-\mu^2)}\left(\frac{\partial}{\partial\lambda}\right)_\sigma + \frac{V}{a}\left(\frac{\partial}{\partial\mu}\right)_\sigma \tag{A.20}$$

$$\nabla_\sigma^2 \equiv \frac{1}{a^2(1-\mu^2)}\frac{\partial^2}{\partial\lambda^2} + \frac{1}{a^2}\frac{\partial}{\partial\mu}\left[(1-\mu^2)\frac{\partial}{\partial\mu}\right]. \tag{A.21}$$

Here, θ in Eq.(A.6) denotes potential temperature, Φ in (A.8) denotes geopotential height, φ and λ denote, respectively, latitude and longitude, a denotes the radius of the Earth, and KE in (A.19) denotes the kinetic energy per unit mass.

In some terms, temperature is divided into the horizontal global mean and the deviation therefrom:

$$T \equiv \bar{T}(\sigma) + T' \tag{A.22}$$

Vertical wind speed in the σ coordinate $\dot{\sigma}$ can be diagnosed as

$$\dot{\sigma} = -\sigma \frac{\partial \pi}{\partial t} - \int_0^\sigma D d\sigma - \int_0^\sigma \boldsymbol{v}_H \cdot \nabla_\sigma \pi d\sigma, \tag{A.23}$$

A Crank-Nicolson-type semi-implicit scheme is applied to filter-out external gravity waves with fast phase speed, whereby enabling an efficient long time-stepping in the integration.

A1.2 "Dynamics" and "Physics"

In the literature of meteorology, the tendencies of prognostic equations are conventionally divided into "dynamics" part and "physics" part. The physics part, denoted by $F_{\mathbf{Phys}}(u)$ in the pseudo-code in Section 3.3, is comprised of the terms of the form $\left.\dfrac{d(\cdot)}{dt}\right|_{\mathbf{Phys}}$ in Eq. (A.1)-(A.5). The dynamics part, denoted by $F_{\mathbf{Dyn}}(u)$ in the pseudo-code in Section 3.3, corresponds to all the terms in the right hand side of Eq. (A.1)-(A.5) except the physics part defined.

In the SPEEDY model, the physics tendencies $F_{\mathbf{Phys}}(u)$ includes contributions from:

- Convection (cumulonimbus)
- Large-scale condensation and Clouds
- Shortwave and Longwave radiation
- Surface fluxes of momentum and energy
- and Vertical diffusion (planetary boundary layer (PBL)).

The details for these parametrizations can be found in the model description by the developers which is available at

`http://users.ictp.it/~kucharsk/speedy-net.html`

A1.3 Boundary conditions

Boundary conditions used in the SPEEDY model are:

- Orography, expressed as the geopotential height at the surface ($\Phi|_{\sigma=1}$)
- land-sea mask

- sea surface temperature (SST)
- sea ice fraction
- soil temperature
- snow depth
- bare-surface albedo
- vegetation coverage

The first two and the last two remain constant during the integration. The remaining fields are given to the model as prescribed climatologies and vary with seasonal march. These boundary conditions are input to the model from external files.

A1.4 Control of Boundary Conditions in the Validation

For the validation of Phase I described in Section 3.4, physics tendencies must be turned off and the orography must be made flat. This subsection describes how these can be controlled.

Among the boundary conditions listed in the previous section, the orography $\Phi|_{\sigma=1}$ alone directly affects the dynamics part: it affects the divergence tendency through Eq. (A.2) and (A.9). All the other boundary conditions enter the prognostic equations (A.1)–(A.5) only through the physics tendencies. Therefore, effects of boundary conditions other than the orography onto the dynamical core automatically vanishes by switching the physics off.

In the SPEEDY model, physics tendencies $\left.\dfrac{d(\cdot)}{dt}\right|_{\mathbf{Phys}}$ are computed and added to the total tendencies in the subroutine called `PHYPAR()`. Therefore, switching-off of physical parametrizations can be done simply by commenting-out calls to this subroutine.

A2 Appendix 2: Phase III

A2.1 Approach

In Phase III, we will reproduce the estimation-correction stage of Danforth et al. (2007). On each time step of the integration of the model M^{LF}, the model error statistics obtained in Phase II will be used to estimate the model error. The model error will be reduced by subtracting the estimated error from the predicted model state. The procedure will be repeated for the models with the newly implemented schemes (M^{Ncyc} and M^{RK4}) as well.

A2.2 Algorithm

The detailed algorithm for Phase III is as follows:
During the integration of M^{LF} (or M^{Ncyc}, M^{RK4}), on each time step $t = t_0$:

1. We estimate the state-dependent component of model error by regressing the current model state $x(t_0)$ onto the model error in the space spanned by the singular vectors. Namely, the anomaly of the current model state $x(t_0) - \langle x^{true} \rangle$ is expressed as a linear combination of left singular vectors u_i. The coefficients $a_i(t_0)$ can be computed by simply taking projection of $x(t_0) - \langle x^{true} \rangle$ onto u_i because of the orthonormality of u_i. Having obtained u_i for sufficiently many modes, we can now "reconstruct" the model error by performing regression in the space spanned by singular vectors:

$$\begin{aligned} a_i(t_0) &= \left(x(t_0) - \langle x^{true} \rangle\right) \cdot u_i \\ \delta x^{dep}(t_0) &= \sum_i \sigma_i a_i(t_0) v_i \\ \because \frac{\langle a_i(t), b_i(t) \rangle}{\langle a_i(t)^2 \rangle} &= \sigma_i \quad \text{from the property of SVD} \end{aligned}$$

   The total estimated model error for 6-hour forecast is then expressed as the sum of state-independent component (i.e. the bias) and the state-dependent component $\delta x^{dep}(t_0)$ obtained above:

$$\delta x^{tot}(t_0) = \langle \delta x(t) \rangle + \delta x^{dep}(t_0)$$

2. The single step forecast is then corrected by subtracting the model error which is scaled to fit the time stepping of the model ($2\Delta t$ for the

leapfrog, and Δt) for Runge-Kutta or Lorenz N-cycle):

$$\begin{aligned} M^{LF}(x(t_0)) &\leftarrow M^{LF}(x(t_0)) - \frac{2\Delta t}{6\text{hours}}\delta x^{tot}(t_0) \\ \text{or} \quad M^{Ncyc}(x(t_0)) &\leftarrow M^{Ncyc}(x(t_0)) - \frac{\Delta t}{6\text{hours}}\delta x^{tot}(t_0), \\ M^{RK4}(x(t_0)) &\leftarrow M^{RK4}(x(t_0)) - \frac{\Delta t}{6\text{hours}}\delta x^{tot}(t_0). \end{aligned}$$

A2.3 Implementation

The subroutines to perform error estimation and correction will be embedded to the SPEEDY model. Since the SPEEDY model is coded in Fortran 77, the subroutines will also be coded in Fortran 77.

A2.4 Validation

Again, validation of the implementation will be conducted by comparing the results for M^{LF} with those published in Danforth et al. (2007).

A2.5 Deliverables

The delivarables of this phase will be the subroutines of the SPEEDY model which performs model error estimation and correction.

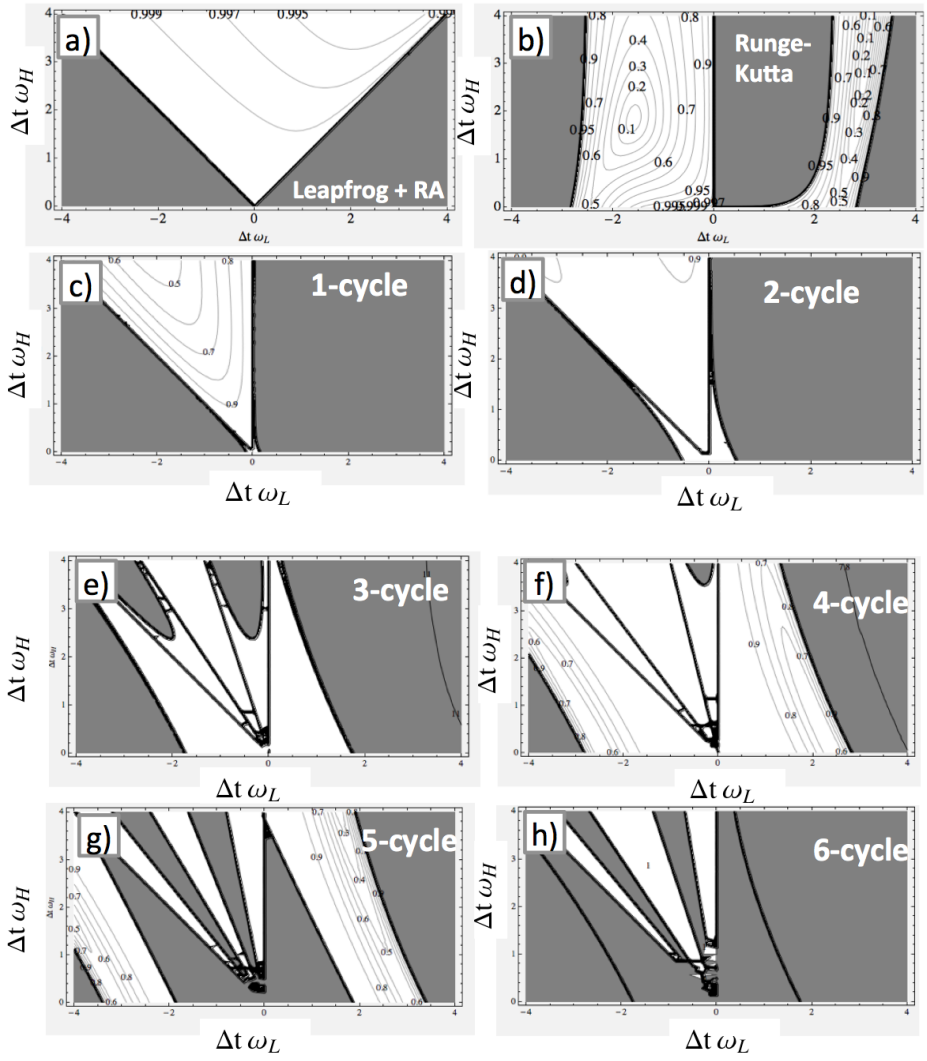

Figure 1: Stability of semi-implicit schemes. Plotted are the modulus of the amplification factor $|A|$ for (a) leapfrog with R/A filter (with parameter $\nu = 0.05$), (b) Runge-Kutta 4th-order scheme, (c) Lorenz 1-cycle, (d) Lorenz 2-cycle, (e) Lorenz 3-cycle, (f) Lorenz 4-cycle, (g) Lorenz 5-cycle and (h) Lorenz 6-cycle. All schemes use Crank-Nicolson scheme for the semi-implicit part. The contours of $|A| = 1$ which divides stable and unstable regions are drawn with thick black lines. The contour intervals are 0.002 for (a) and 0.1 for the rest.

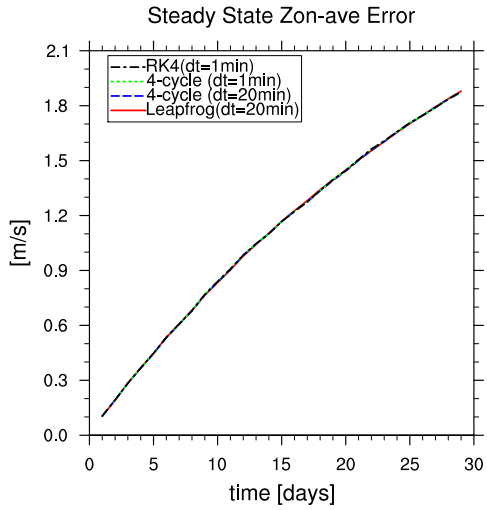

Figure 2: The metric $l_2(\bar{u}(t) - \bar{u}(t=0))$ of leapfrog (with $\Delta t = 20$min.), Lorenz 4-cycle (with $\Delta t = 20$min.), Lorenz 4-cycle (with $\Delta t = 1$min.), and 4th-order Runge-Kutta (with $\Delta t = 1$min.).

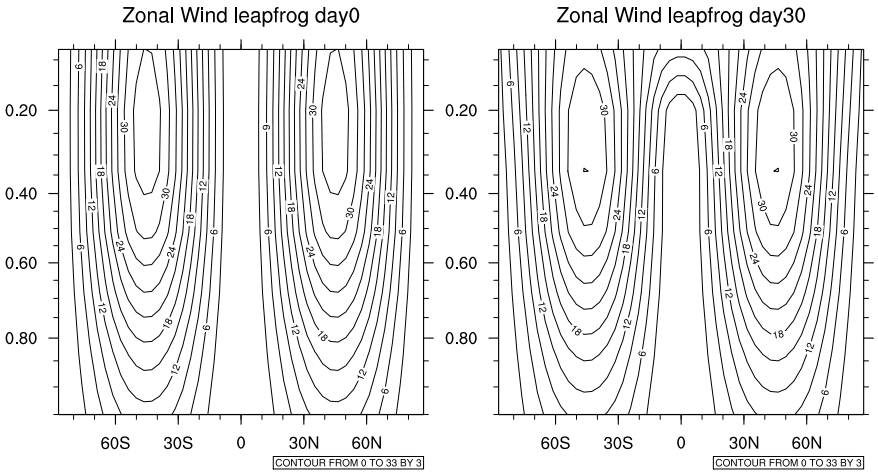

Figure 3: meridional cross sections of zonal mean zonal wind ($\bar{u}(\varphi, \sigma, t)$) at (left)$t = 0$ and (right)$t = 30$ days for the leapfrog with time step $\Delta t = 20$ min. Contour intervals are 3 m/s.

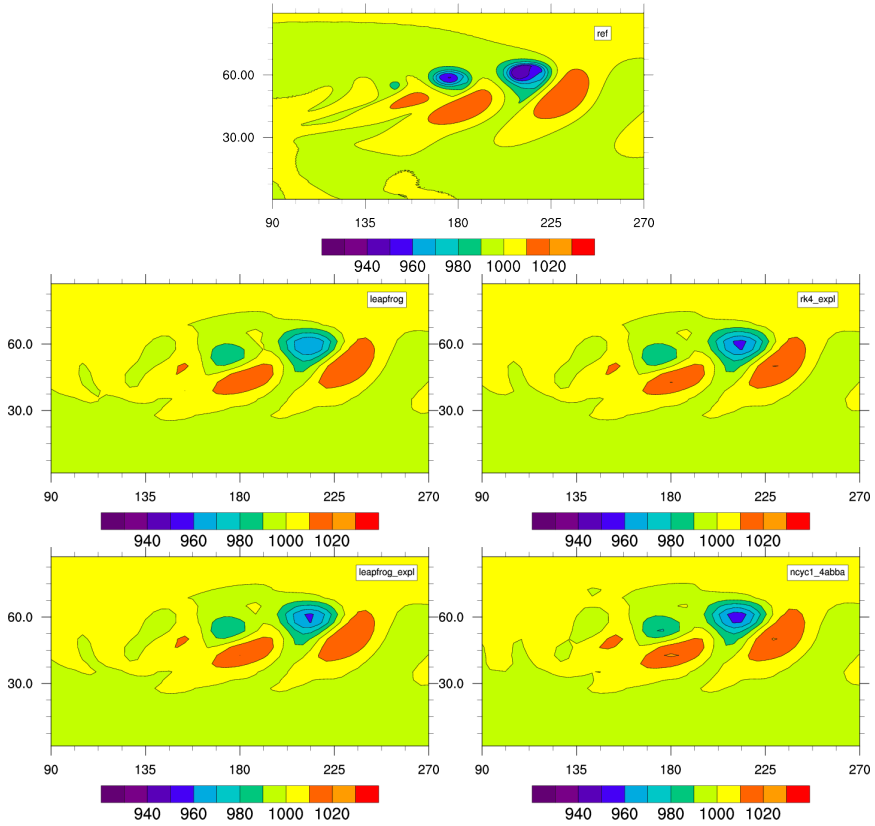

Figure 4: Snapshots of surface pressure (in hPa) at the 9th day. (Top) Reference solution provided by Jablonowski and Williamson (2006), (Upper Left) leapfrog with $\Delta t = 20$min., (Upper Right) Runge Kutta with $\Delta t = 1$min., (Lower Left) leapfrog with $\Delta t = 1$min., (Lower Right) Lorenz 4-cycle with $\Delta t = 20$min.

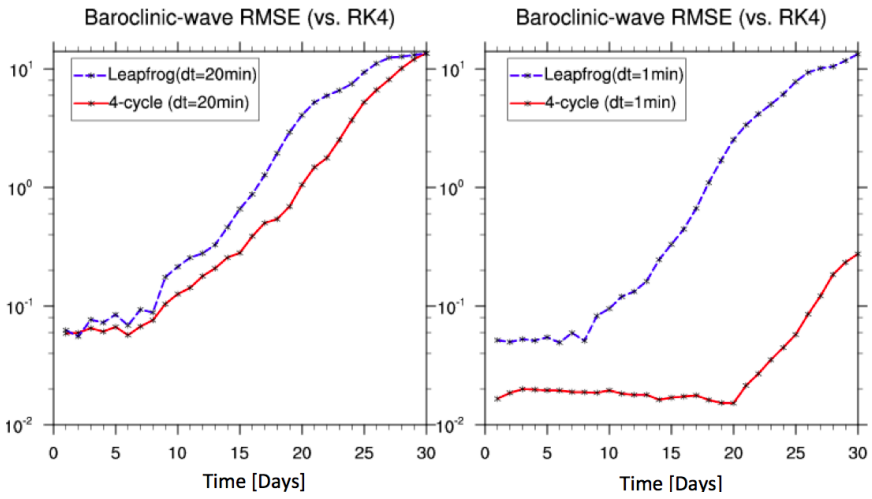

Figure 5: RMSE of surface pressure with respect to Runge-Kutta with $\Delta t =$ 1min. (left) Leapfrog and Lorenz 4-cycle with $\Delta t = 20$min. (right) as in the left, but for $\Delta t = 1$min.

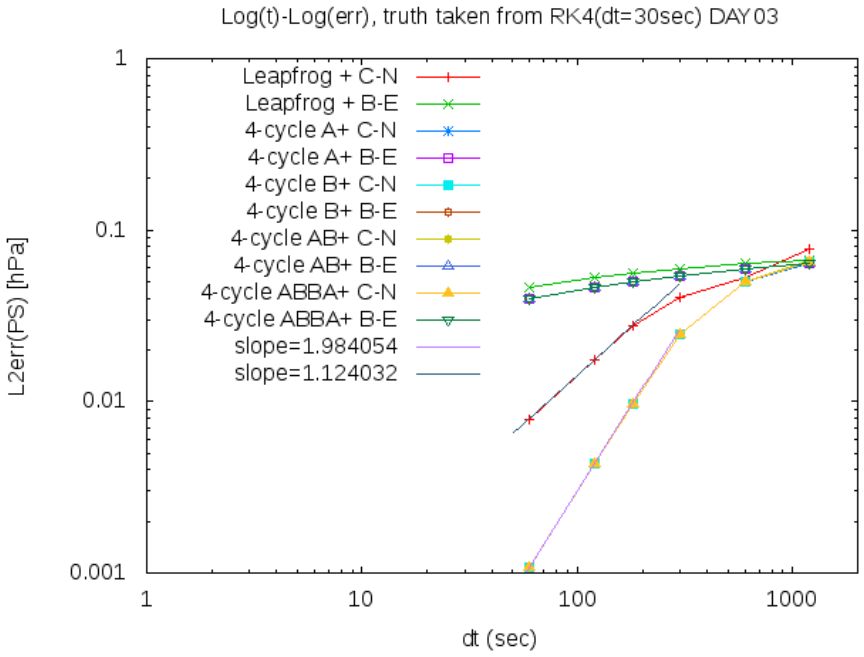

Figure 6: L^2 errors in surface pressure for various time integration schemes plotted against time step Δt shown on log-log plane. Regression lines are also shown for the semi-implicit Lorenz 4-cycle (version B) with Crank-Nicolson scheme and the semi-implicit leapfrog with Crank-Nicolson scheme and R/A filter which are estimated using time steps $\Delta t = 60, 120$ and 180 sec. The slopes (i.e. the order of accuracy) for the two schemes are, respectively, 1.98 and 1.12, which are consistent with our theoretical expectation.

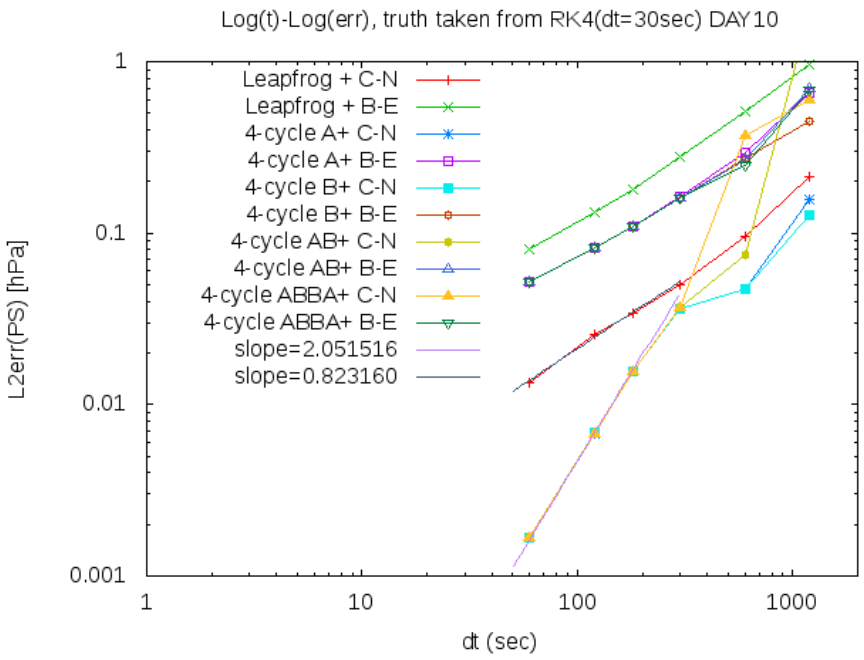

Figure 7: As in Figure 6, but for 10-day forecasts.

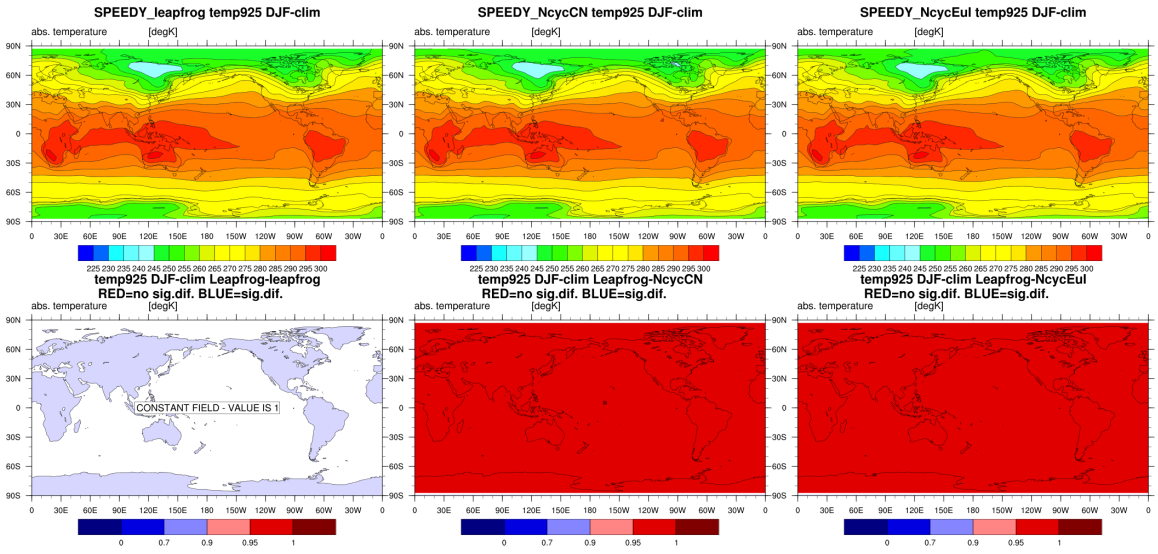

Figure 8: Top panels: Climatologies of winter-time 925hPa temperature computed for (left) leapfrog, (middle) Lorenz 4-cycle with Crank-Nicolson and (right) Lorenz 4-cycle with Backward-Euler. Units are in Kelvin and contour intervals are 15K. Bottom panels: as in the top panels, but for the p-values of the null-hypothesis (see the text).

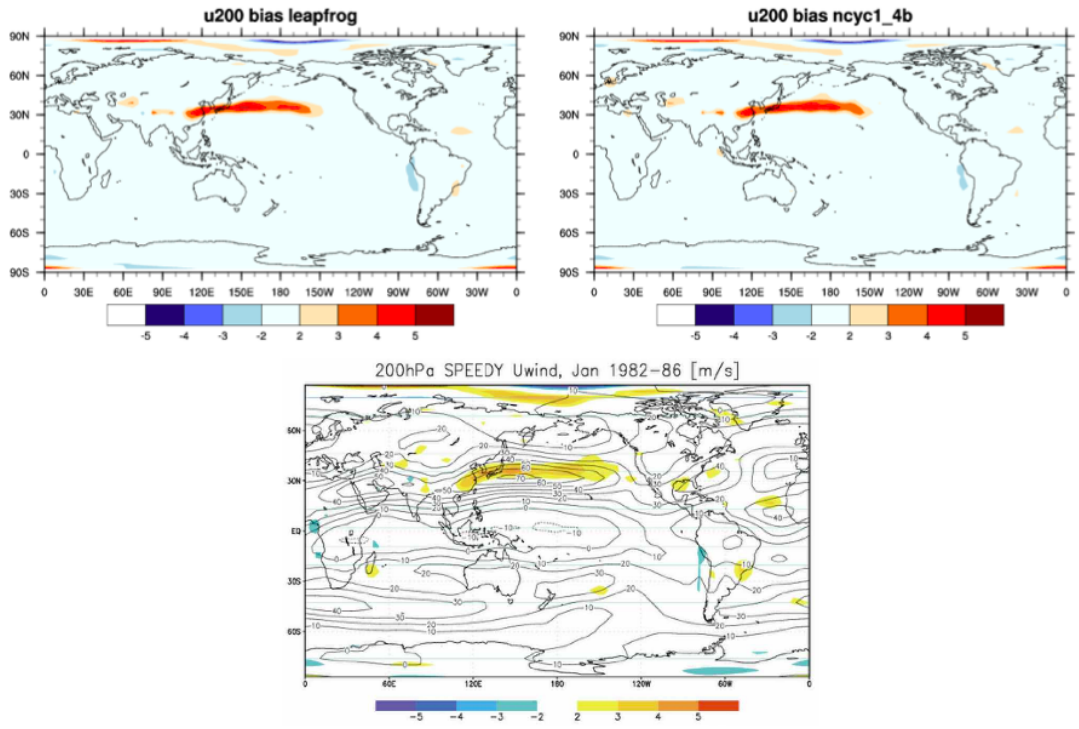

Figure 9: Bias of 6-hour forecasts for 200hPa zonal wind (U200) for (top left) leapfrog scheme, (top right) Lorenz 4-cycle. The bottom panel shows the corresponding result from Danforth et al. (2007).

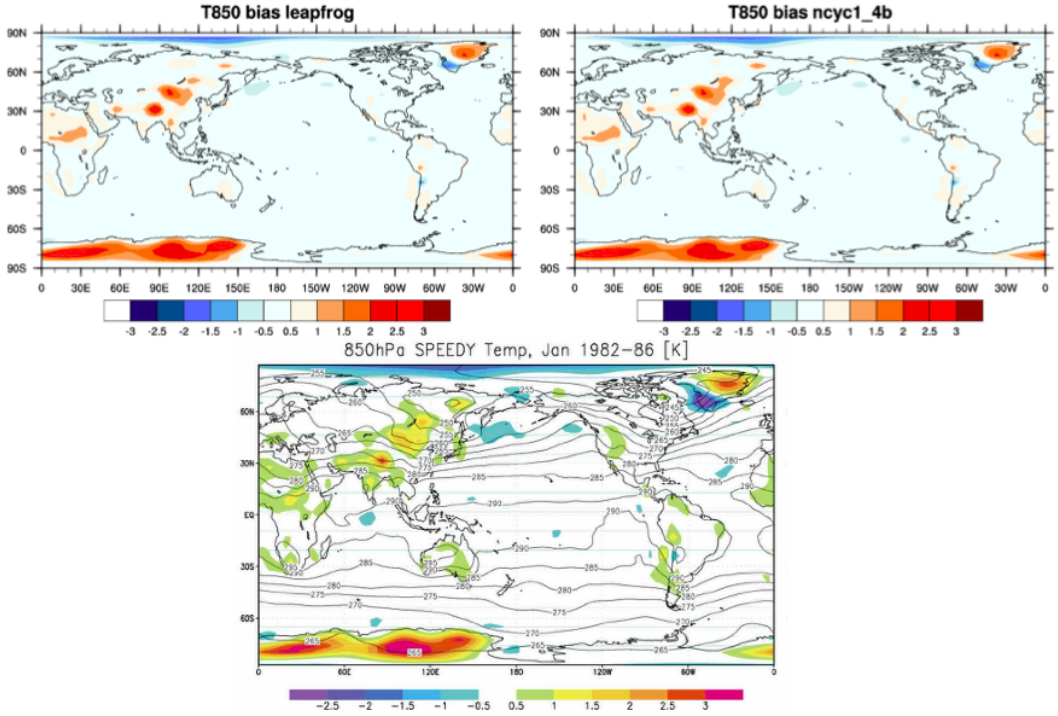

Figure 10: As in Figure 9, but for 850hPa temperature (T850).

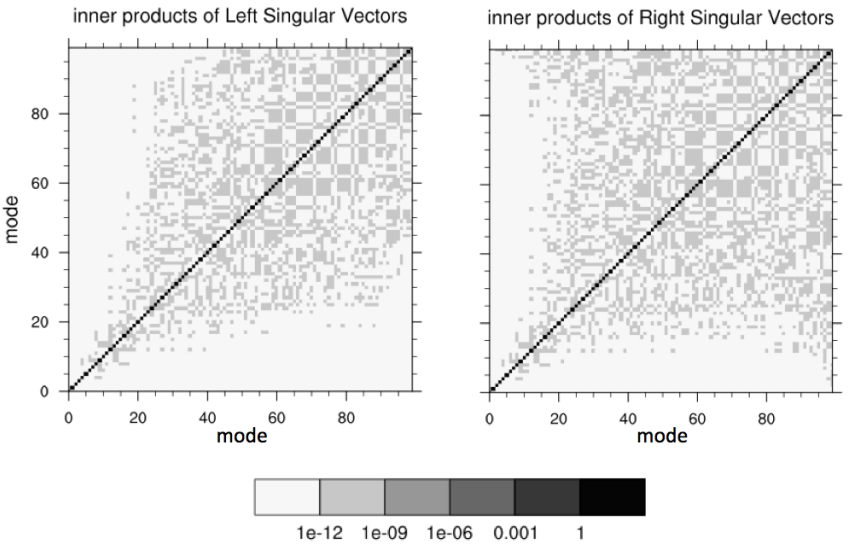

Figure 11: Orthogonality of left and right singular vectors of the cross-covariance matrix for zonal wind at $\sigma = 0.2$ of leapfrog scheme obtained using Lanczos method with 100 iterations. The shading of (i, j) grid represents $u_i^T u_j$ (or $v_i^T v_j$).

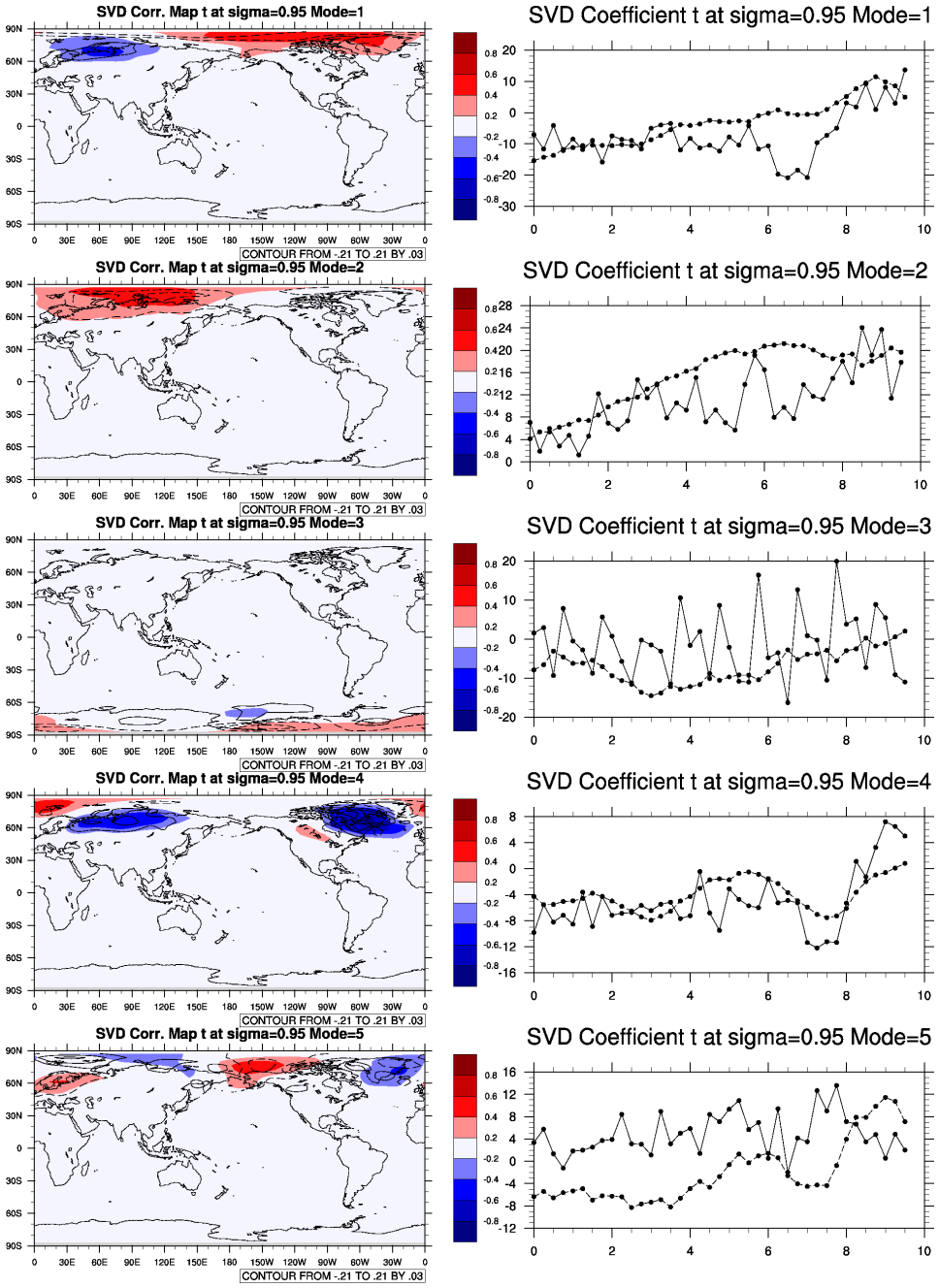

Figure 12: The homogeneous (contour) and heterogeneous (shade) right correlation maps for the first 10 dominant modes for T0.95 of leapfrog scheme identified through SVD analysis using the Lanczos algorithm. The corresponding amplitude timeseries $a_k(t)$ and $b_k(t)$ (see Appendix II for their definitions) are also shown, respectively, with solid and dashed lines on the right. Contour intervals for homogeneous correlation maps are 0.03 and the zero contours are omitted. For heterogeneous correlation maps, the intervals for color shadings are 0.2.

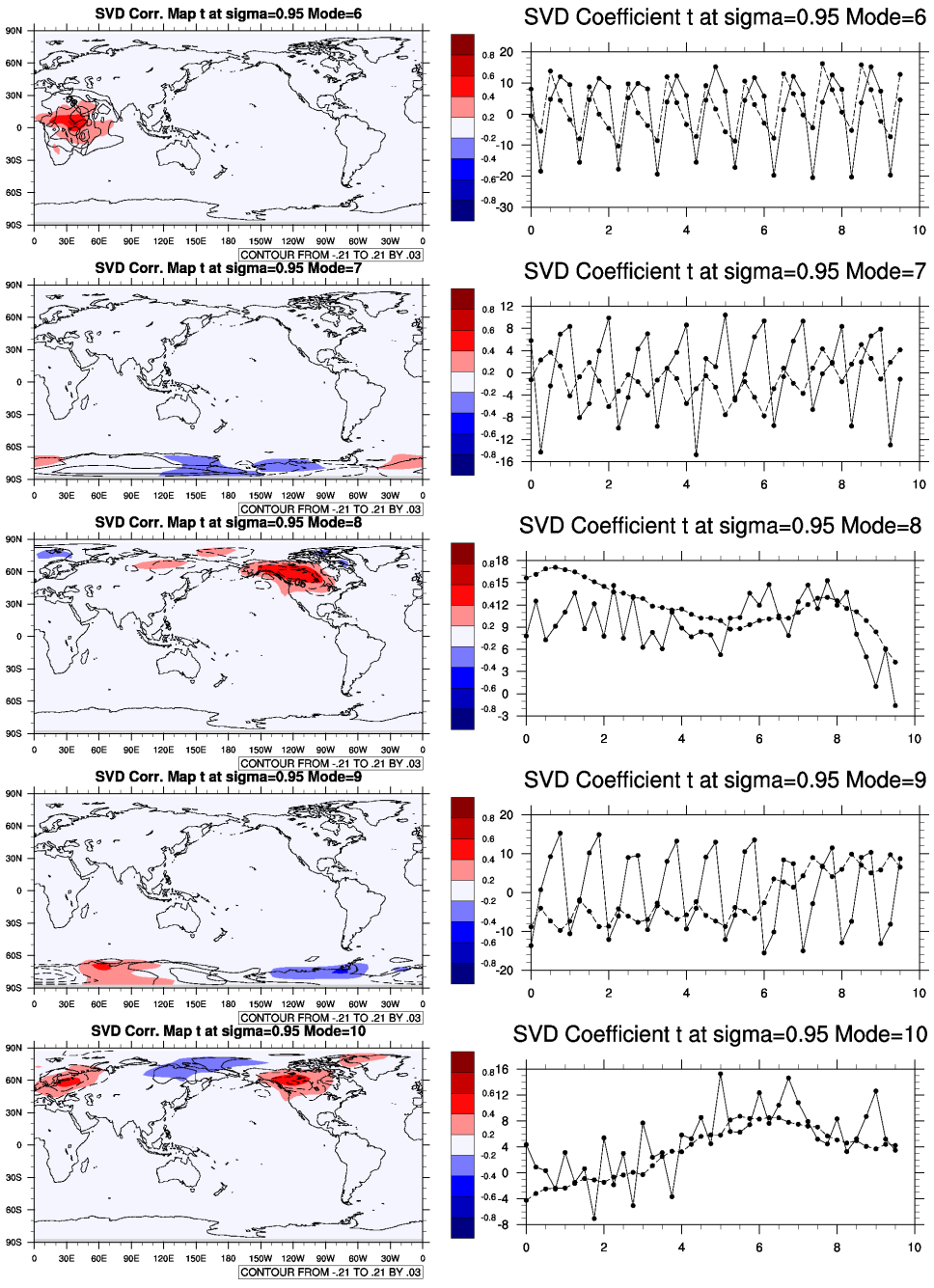

Figure 13: As in Figure 12, but for 6th to 10th dominant modes.

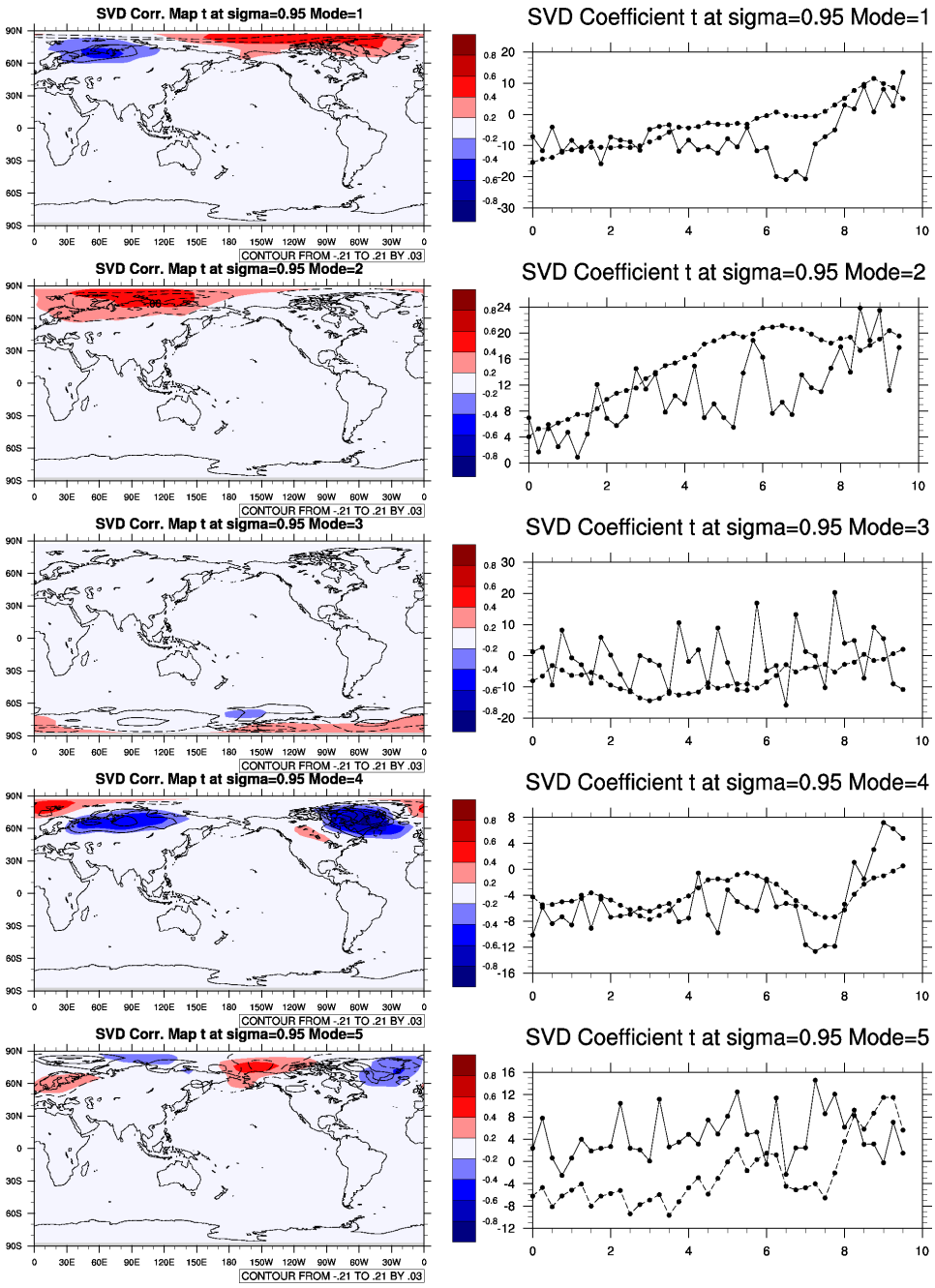

Figure 14: As in Figure 12, but for Lorenz 4-cycle.

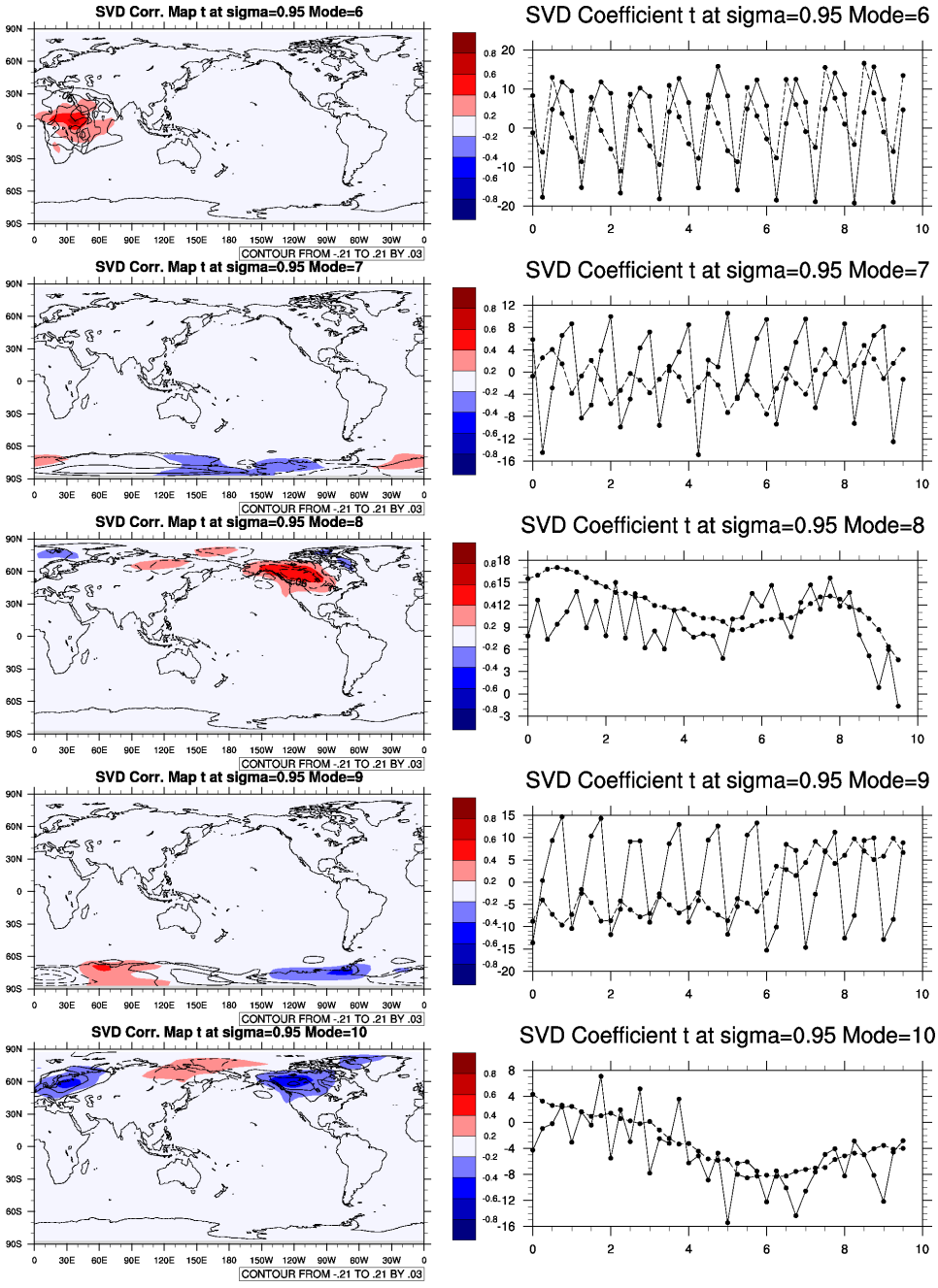

Figure 15: As in Figure 13, but for Lorenz 4-cycle.