Reduction of Temporal Discretization Error in an Atmospheric General Circulation Model

Mid-year Progress Report

December 14, 2012

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

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 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, will be implemented to an existing AGCM, called SPEEDY (**S**implified **P**arametrizations, primitiv**E** **E**quation **DY**namics) model, which adopts the leapfrog scheme with Robert-Asselin 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) and the 4th-order 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 the 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 correction methods introduced by Danforth et al. (2007) who used model error bias statistics and the singular value decomposition (SVD) technique on the 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 will be 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

```
$w^0 = 1,$                                      (2)
$w^k = \frac{N}{N-k}$ $(k = 1, \ldots, N-1)$    (3)
do $k = 0, \ldots$
    $w \leftarrow w^{\mathrm{mod}(k,N)}$         (4)
    $G \leftarrow wF(u) + (1-w)G$               (5)
    $u \leftarrow u + G\Delta t$                (6)
end do
```

N-cycle B

```
$w^0 = 1,$                                      (7)
$w^k = \frac{N}{k}$ $(k = 1, \ldots, N-1)$      (8)
do $k = 0, \ldots$
    $w \leftarrow w^{\mathrm{mod}(k,N)}$         (9)
    $G \leftarrow wF(u) + (1-w)G$               (10)
    $u \leftarrow u + G\Delta t$                (11)
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 (described in the next subsection) 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:

```
do $k = 0, \dots,$
    $h_1 \leftarrow F(u), \quad v \leftarrow u + \frac{\Delta t}{2} h_1$                (12)
    $h_2 \leftarrow F(v), \quad v \leftarrow u + \frac{\Delta t}{2} h_2$                (13)
    $h_3 \leftarrow F(v), \quad v \leftarrow u + \Delta t h_3$                          (14)
    $h_4 \leftarrow F(v)$                                                               (15)
    $u \leftarrow u + \frac{\Delta t}{6}(h_1 + 2h_2 + 2h_3 + h_4)$                      (16)
end do
```

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

```
do $k = 0, \dots,$
    $h \leftarrow F(u),\ p \leftarrow h, \quad v \leftarrow u + \frac{\Delta t}{2} h$          (17)
    $h \leftarrow F(v),\ p \leftarrow p + 2h, \quad v \leftarrow u + \frac{\Delta t}{2} h$     (18)
    $h \leftarrow F(v),\ p \leftarrow p + 2h, \quad v \leftarrow u + \Delta t h$               (19)
    $h \leftarrow F(v),\ p \leftarrow p + h$                                                  (20)
    $u \leftarrow u + \frac{\Delta t}{6} p$                                                    (21)
end do
```

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

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

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 ~ 1000 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. 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

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

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,

$$\begin{aligned} \delta u &= F^E(u^n) + L^I u^{n-1} + 2\alpha \Delta t L^I \delta u & (25) \\ \Leftrightarrow \delta u &= (I - 2\alpha\Delta t L^I)^{-1}(F^E(u^n) + L^I u^{n-1}) & (26) \end{aligned}$$

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

A straightforward semi-implicit modification to the Lorenz N-cycle is to apply tendency-modification similar to Eq.(26) on each step of the N-cycle:

```
$w^0 = 1,$                                              (28)
$w^k = \frac{N}{k}\ (k = 1, \ldots, N-1),$              (29)
**do** $k = 0, \ldots$
    $w \leftarrow w^{\mathrm{mod}(k,N)}$                (30)
    $G \leftarrow wF^E(u) + (1-w)G$                     (31)
    $\delta u = (I - \alpha\Delta t L^I)^{-1}(G + L^I u)$   (32)
    $u \leftarrow u + \Delta t \delta u$                (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 $k = 0, \ldots,$
    $h_1 \leftarrow F^E(u), \quad v \leftarrow u + \frac{\Delta t}{2} h_1$    (34)
    $h_2 \leftarrow F^E(v), \quad v \leftarrow u + \frac{\Delta t}{2} h_2$    (35)
    $h_3 \leftarrow F^E(v), \quad v \leftarrow u + \Delta t h_3$    (36)
    $h_4 \leftarrow F^E(v)$    (37)
    $\delta u = (I - \alpha \Delta t L^I)^{-1} \cdot \frac{1}{6}(h_1 + 2h_2 + 2h_3 + h_4)$    (38)
    $u \leftarrow u + \Delta t \delta u$    (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 $|A| - 1$ for the semi-implicit leapfrog scheme (with Robert-Asselin filter), the semi-implicit Lorenz 3-cycle, semi-implicit Lorenz 4-cycle and semi-implicit 4th-order Runge-Kutta. The zero contours which divide stable and unstable regions are drawn in thick black lines. Semi-implicit leapfrog is stable if $\omega_L < \omega_H$, a condition which is always met for practical purposes. The stability region for Lorenz 3-cycle is very narrow and notably, it is unconditionally unstable (i.e. regardless of the value of ω_L) for $\omega_H > 1.5$. This means that Lorenz 3-cycle cannot be used for a practical AGCM. Lorenz 4-cycle exhibits a reasonably large stability region, albeit much smaller than that for the leapfrog. Interestingly, the semi-implicit version of 4th-order Runge-Kutta is unconditionally unstable.

As Lorenz 4-cycle was found to be more stable than 3-cycle, 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).

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 Robert-Asselin filter is also

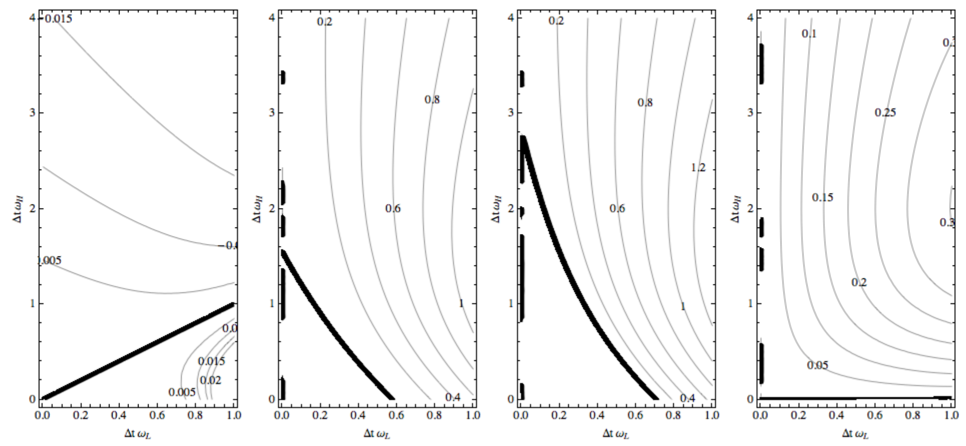

Figure 1: Stability of semi-implicit schemes. Plotted are $|A| - 1$ for (from left to right) leapfrog, 3-cycle, 4-cycle and Runge-Kutta scheme. The zero contours are drown with thick black line. The contour intervals are, from left to right, 0.005, 0.2, 0.2 and 0.05. Note that the vertical thick black lines along the ordinate are part of zero contours although they may appear to be distinct broken lines.

applied. Namely, in a pseudo-code, the algorithm is:

```
do $k = 1, \dots$
    $u^{+1} \leftarrow u^{-1} + 2\Delta t(F_{\mathbf{Dyn}}(u^0) + F_{\mathbf{Phys}}(u^{-1}))$    (44)
    $u^0 \leftarrow u^0 + \alpha(u^{+1} - 2u^0 + u^{-1})$    (45)
    $t \leftarrow t + \Delta t$    (46)
    $u^{-1} \leftarrow u^0, \quad u^0 \leftarrow u^{+1}$    (47)
end do
```

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 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 will be 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 will 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) will be 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 off. Thus, we will 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, will be 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:

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

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 will produce 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`

Due to the problem described in Section 4.4, the results for the verification with model climate are not available yet.

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:

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

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:

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

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 $\Delta t = 20$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.). 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

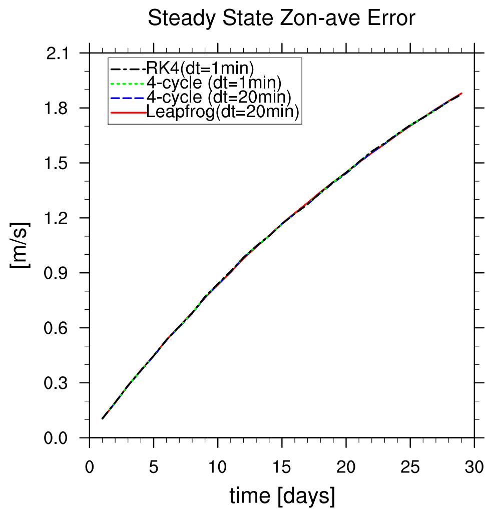

Figure 2: The metric $l_2(\bar{u}(t) - \bar{u}(t = 0))$ of leapfrog (with $\Delta t = 20$min.), Lorenz 4-cycle (with $\Delta t = 20$min.), Lorenz 4-cycle (with $\Delta t = 1$min.), and 4th-order Runge-Kutta (with $\Delta t = 1$min.).

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

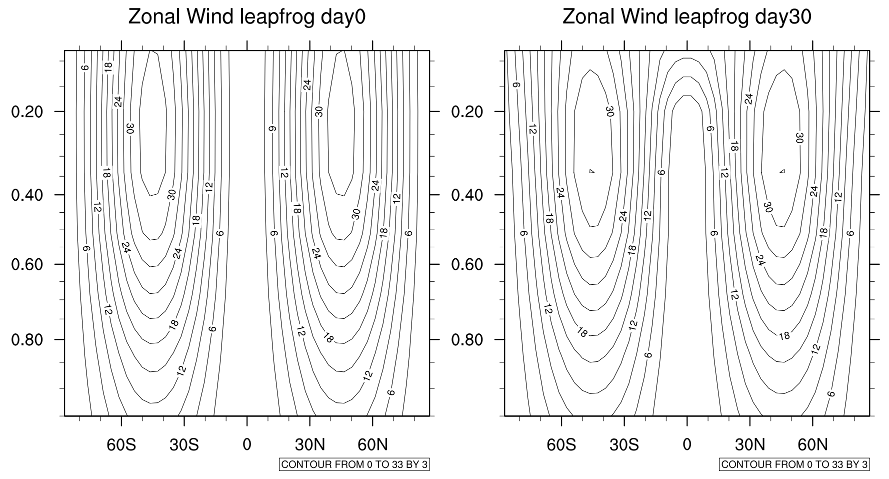

Figure 3: meridional cross sections of zonal mean zonal wind ($\bar{u}(\varphi, \sigma, t)$) at (left)$t = 0$ and (right)$t = 30$ days for the leapfrog with time step $\Delta t = 20$ min. Contour intervals are 3 m/s.

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 $\Delta t =$ 1min. as the truth, which is shown in Figure 5. For both $\Delta t =$ 20min. (left) and $\Delta t =$ 1min. (right), Lorenz 4-cycle is consistently more accurate than the leapfrog, and the difference is quite dramatic for the smaller time step ($\Delta t =$ 1min.). These results clearly confirm the superiority of Lorenz 4-cycle over the leapfrog.

4.4 Phase I: Unexpected Problem

Although we succeeded in clearly showing that Lorenz N-cycle is more accurate for the dynamical core test, we encountered a problem that we did not

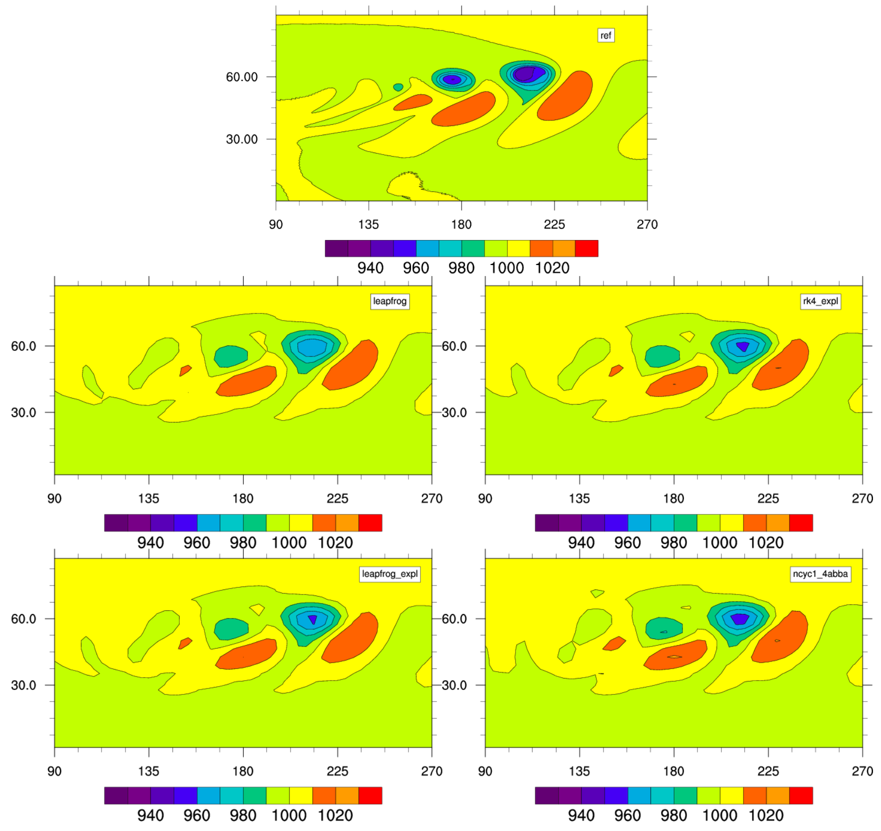

Figure 4: Snapshots of surface pressure (in hPa) at the 9th day. (Top) Reference solution provided by Jablonowski and Williamson (2006), (Upper Left) leapfrog with $\Delta t = 20$min., (Upper Right) Runge Kutta with $\Delta t = 1$min., (Lower Left) leapfrog with $\Delta t = 1$min., (Lower Right) Lorenz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

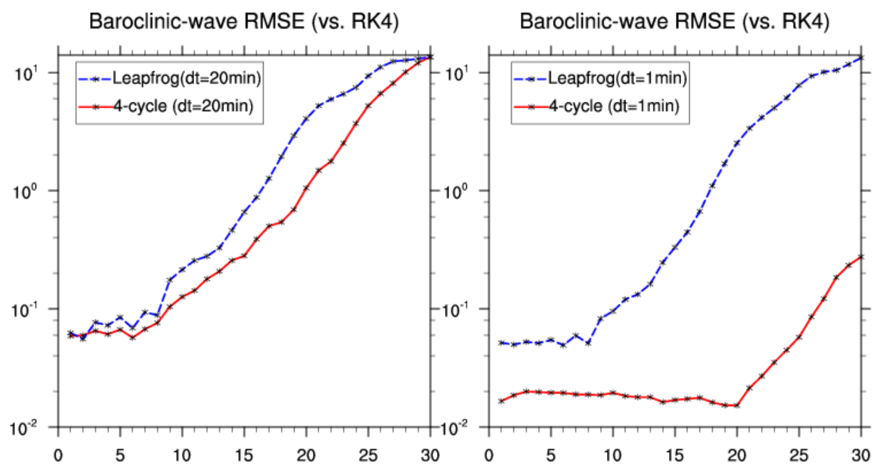

Figure 5: RMSE of surface pressure with respect to Runge-Kutta with $\Delta t =$ 1min. (left) Leapfrog and Lorenz 4-cycle with $\Delta t = 20$min. (right) as in the left, but for $\Delta t =$ 1min.

expect: Lorenz N-cycle blows up if physical parametrizations are included in the model. This blow-up occurs to Runge Kutta and Forward Euler scheme as well.

Figure 6 shows the time-height hovmüller plot of global averages of (left) temperature, (center) rotational kinetic energy and (right) divergent kinetic energy, for (top) Runge-Kutta, (middle) Lorenz 4-cycle, and (bottom) leapfrog. Unlike in the case of leapfrog where temperature does not change significantly from the prescribed initial profile, in the cases of Runge-Kutta or Lorenz 4-cycle, the temperature at the lower troposphere continues to grow linearly until it reaches impossibly high 320K.

In order to carry out the original plan for Phase II which is describe in the next section, the SPEEDY model with Lorenz N-cycle needs to run stably with physical parametrizations switched on. We plan to work on this issue until the end of the December 2012. If we succeed to fix this problem and Lorenz N-cycle becomes able to run stably with the physical parametrizations switched on, then we will continue Phase II as originally planned. If we do not succeed, then we will modify Phase II as described in Section 6.

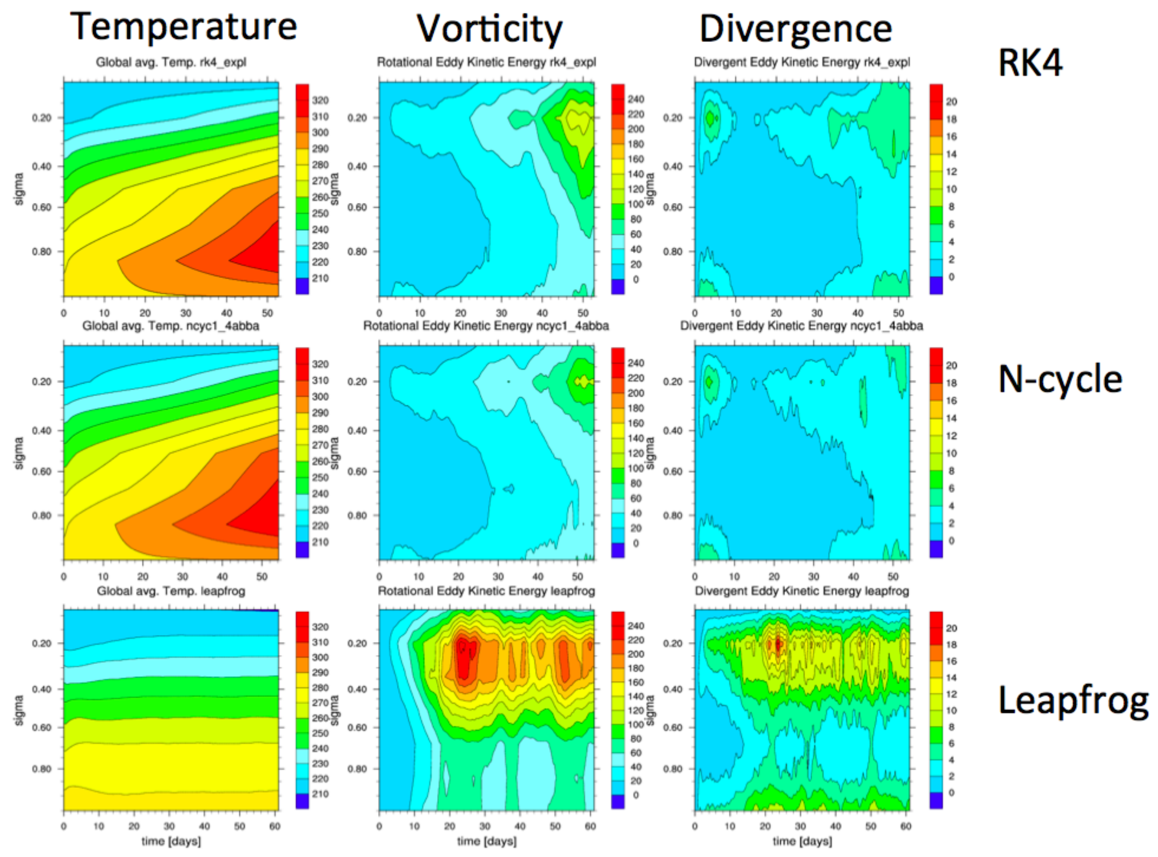

Figure 6: Time-height hovmüller plot of global averages of (left) temperature, (center) rotational kinetic energy and (right) divergent kinetic energy, for (top) Runge-Kutta, (middle) Lorenz 4-cycle, and (bottom) leapfrog.

5 Phase II: original plan

In Phase II of this project, the training stage of Danforth et al. (2007) will be reproduced, both for the original SPEEDY model with leapfrog scheme and for the SPEEDY model with the newly implemented schemes. The second stage of Danforth et al. (2007), the estimation-correction stage, will be addressed in Phase III of this project provided that time permits. For the outline of Phase III, see the Appendix 2.

5.1 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 covariance matrix between the model state and the model error:

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

In prior to building the covariance matrix, the mean will be removed, 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) will be 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, time permitted, will be addressed in Phase III of this project, the singular vectors and the singular values will be 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 will be 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 covariance matrix
      $C = \frac{1}{\#i-1} \sum_i (x^{true}(t_i) - \langle x^{true} \rangle) (\delta x(t_i) - \langle \delta x \rangle)^T$
  perform SVD for $C$:   $C = U \Sigma V^T$
end do
```

The NCEP/NCAR reanalysis data which will be 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 Implementation and Platform

Programs to be 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 will be implemented in Fortran 90 on a Linux server hosted on the network of the AOSC department.

5.3 Validation

For the SVD code, validation will be conducted by first preparing a small-dimensional dummy data for the covariance and then performing the SVD on this dummy data both by running the implemented code and by running a Matlab's package routine. The implementation is judged successful if the two results agrees.

The entire implementation will be judged successful if the bias and singular vectors obtained for M^{LF} agrees with those published in Danforth et al. (2007).

5.4 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.

6 Phase II: possible modification

As described in Section 4.4, the original plan for Phase II described in the previous section requires that SPEEDY with Lorenz N-cycle runs with physical parametrizations. If we could not resolve the problem described in Section 4.4 by the end of December 2012, then we will make the following modifications to the Phase II:

1. SPEEDY model is run without physics
2. Instead of using NCEP/NCAR reanalysis as the truth, nature run of SPEEDY model with Runge-Kutta integrated with small time step (1 min.) will be regarded as the truth
3. For the validation, we will ask Dr. Danforth to provide his original code used in Danforth et al. (2007) and duplicate Phase II using his code.

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. (**not delivered**)

Deliverables of Phase II are:

1. an archive of the model errors
2. model error bias
3. pairs of singular vectors for the model state and the model error along with the corresponding singular values
4. code for performing SVD.

Deliberables also includes

1. two presentations, mid-year and final
2. two reports, mid-year and final
3. and the project proposal.

8 Revised Schedule and Milestones

The revised schedule for the project is as follows:

- Phase I
  - Implement Lorenz N-cycle and 4th-order Runge-Kutta scheme to the SPEEDY model: ~~September through end of November~~. (**done**)
  - Write the mid-year report, prepare the oral presentation: ~~December~~ (**done**)
  - Switch-off physical parametrizations and prepare flat orography: ~~January~~ (**done**)
  - Perform the dynamical core tests: ~~early February~~ (**done**)
  - (new) try to resolve physics problem: until the end of December
- 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.

The corresponding milestones are:

- Phase I
  - The SPEEDY model with Lorenz N-cycle and 4th-order Runge-Kutta scheme both runs: ~~end of November~~ (**done**)
  - Complete validation of Lorenz N-cycle and 4th-order Runge-Kutta scheme: ~~February~~ (**done**)
- Phase II
  - Complete computation of error bias and covariance: March

– Complete validation of the SVD program: April
– Complete validation of the entire procedure: early May
– Complete verification: mid-May

9 Summary of the Achievements of the First Semester

Here is the summary of the achievements so far:

- Analyzed the accuracy and stability of semi-implicit formulation of Lorenz N-cycle and Runge-Kutta
- Implemented Lorenz N-cycle and Runge-Kutta to the SPEEDY model
- Successfully performed code validation
- Performed verification with the Dynamical Core tests and confirmed the superiority of Lorenz N-cycle over the leapfrog
- Encountered unexpected problem: SPEEDY with physics parametrizations blows up if the temporal scheme is Lorenz N-cycle, Runge-Kutta, or Forward Euler (i.e. anything other than leapfrog)

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: *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 multidecadal 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.

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.

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:

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

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

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

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

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

where

$$\begin{aligned}
\theta &\equiv T\,(p/p_0)^{-\kappa} && \text{(A.6)}\\
\kappa &\equiv R/C_p && \text{(A.7)}\\
\Phi &\equiv gz && \text{(A.8)}\\
&= \Phi|_{\sigma=1} - \int_1^\sigma \frac{RT}{\sigma} d\sigma && \text{(A.9)}\\
\pi &\equiv \ln p_S && \text{(A.10)}\\
\dot{\sigma} &\equiv \frac{d\sigma}{dt} && \text{(A.11)}\\
\mu &\equiv \sin\varphi && \text{(A.12)}\\
U &\equiv u\cos\varphi && \text{(A.13)}\\
V &\equiv v\cos\varphi && \text{(A.14)}\\
\zeta &\equiv \frac{1}{a(1-\mu^2)}\frac{\partial V}{\partial \lambda} - \frac{1}{a}\frac{\partial U}{\partial \mu} && \text{(A.15)}\\
D &\equiv \frac{1}{a(1-\mu^2)}\frac{\partial U}{\partial \lambda} + \frac{1}{a}\frac{V}{\mu} && \text{(A.16)}\\
\mathcal{F}_U &\equiv (\zeta+f)V - \dot{\sigma}\frac{\partial U}{\partial \sigma} - \frac{RT'}{a}\frac{\partial \pi}{\partial \lambda} && \text{(A.17)}\\
\mathcal{F}_V &\equiv -(\zeta+f)U - \dot{\sigma}\frac{\partial V}{\partial \sigma} - \frac{RT'}{a}(1-\mu^2)\frac{\partial \pi}{\partial \mu} && \text{(A.18)}\\
KE &\equiv \frac{U^2+V^2}{2(1-\mu^2)} && \text{(A.19)}\\
\boldsymbol{v}_H \cdot \nabla &\equiv \frac{u}{a\cos\varphi}\left(\frac{\partial}{\partial \lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial}{\partial \varphi}\right)_\sigma \\
&= \frac{U}{a(1-\mu^2)}\left(\frac{\partial}{\partial \lambda}\right)_\sigma + \frac{V}{a}\left(\frac{\partial}{\partial \mu}\right)_\sigma && \text{(A.20)}\\
\nabla_\sigma^2 &\equiv \frac{1}{a^2(1-\mu^2)}\frac{\partial^2}{\partial \lambda^2} + \frac{1}{a^2}\frac{\partial}{\partial \mu}\left[(1-\mu^2)\frac{\partial}{\partial \mu}\right]. && \text{(A.21)}
\end{aligned}$$

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:

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

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 in to "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 $\left.\frac{d(\cdot)}{dt}\right|_{\mathbf{Phys}}$ 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 $\left.\frac{d(\cdot)}{dt}\right|_{\mathbf{Phys}}$ 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.

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:

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

   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.