Reduction of Temporal Discretization Error in an Atmospheric General Circulation Model (AGCM)

Final Presentation

Apr. 30, 2013
Daisuke Hotta
hotta@umd.edu

Advisor: Prof. Eugenia Kalnay
Dept. of Atmospheric and Oceanic Science,
University of Maryland, College Park
ekalnay@atmos.umd.edu
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Deliverable

5. References
Numerical Weather Prediction (NWP):

= Initial Value Problem of PDE

**Atmospheric Phenomena**

**Governing Equations**

\[
\frac{\partial \zeta}{\partial t} = \frac{1}{a(1 - \mu^2)} \frac{\partial F_V}{\partial \lambda} - \frac{1}{a} \frac{\partial F_U}{\partial \mu} - K_R(\sigma)(\zeta - \langle \zeta \rangle) - K_v\left(\nabla_\sigma^2 \zeta - \frac{2}{a^2}\right) + \cdots
\]

\[
\frac{\partial D}{\partial t} = \frac{1}{a(1 - \mu^2)} \frac{\partial F_U}{\partial \lambda} + \frac{1}{a} \frac{\partial F_V}{\partial \mu} - \nabla_\sigma^2(\Phi + R\bar{T}\pi + KE) - K_R(\sigma)(D - \langle D \rangle) - K_v\left(\nabla_\sigma^2 D - \frac{2}{a^2}\right)
\]

\[
\frac{\partial T}{\partial t} = -\frac{1}{a(1 - \mu^2)} \frac{\partial U_T'}{\partial \lambda} - \frac{1}{a} \frac{\partial V_T'}{\partial \mu} + T'D - \sigma \frac{\partial \bar{T}}{\partial \sigma} + K_T\left(\frac{\partial \bar{\pi}}{\partial t} + v_H \cdot \nabla_\sigma \bar{\pi} + \frac{\partial \bar{\sigma}}{\partial t}\right) - K_N(\sigma)(T - \langle T \rangle) - K_h\left(\nabla_\sigma^2 T - \frac{2}{a^2}\right) + T + \frac{\alpha_1}{\partial t}_{\text{forcing}}
\]

\[
\frac{\partial \pi}{\partial t} + v_H \cdot \nabla_\sigma \pi = -D - \frac{\partial \bar{\sigma}}{\partial \sigma}
\]

**Numerical Discretization**

**Solve! (Simulate)**

(Real Atmosphere)

(from JMA website)

Simulated Atmosphere


**Hydrodynamic PDE**

\(O(10^9)\)-dimensional ODE

**AGCM**: Atmospheric General Circulation Model

= a computer program which simulates the flow of global atmosphere by numerically integrating the governing fluid dynamical PDEs

Introduction: Motivation

Due to computational restrictions ...

• most AGCMs adopt low-order time-integration schemes, such as
  - Leap-frog with Robert-Asselin filter (1\textsuperscript{st} order)
  - Explicit Backward Euler (aka. Matsuno; 1\textsuperscript{st} order)

• Often, $\Delta t$ is taken as the largest value for which computational instability is suppressed,

• under the premise that temporal discretization errors are negligible compared to those associated with spatial discretization or Physical Parameterizations.
Introduction: Motivation

However ...

- Spatial resolutions become finer and finer as the supercomputers become faster.
- Is the premise that time truncation errors are negligible justified?
- If not, how can we alleviate such errors?

Goal of the Project: Reduction of such model errors

Approaches:

Phase 1: Use a more accurate scheme with the same computational cost

Phase 2: Identify and parameterize the error, and reduce it using data assimilation
Phase I: Implementation of Lorenz $N$-cycle to SPEEDY model
Phase 1: A Better integration scheme (Lorenz N-cycle)

Lorenz (1971) proposed an incredibly smart time-integration scheme which:
• requires only 1 function evaluation per step
• but yet (every $N$ steps) it is of
  - (up to) 4\textsuperscript{th}-order accuracy (for nonlinear systems)
  - arbitrary order of accuracy (for linear systems)

However, this scheme seems to have remained forgotten. No applications have been made to AGCMs.

⇒ Apply Lorenz $N$-cycle to an AGCM (Phase 1)
Phase 1: Approach

- Implement Lorenz $N$-cycle to an existing AGCM
- Implement 4$^{th}$ order Runge-Kutta (RK4) as well as a reference

- Compare the accuracy and efficiency of the newly introduced schemes with the original scheme
- Perform verification by the Jablonowski-Williamson (2006) Dynamical-core Tests
## Phase 1: Algorithms

### Lorenz $N$-cycle

\[ G = 0, \, w^1 = 1, \, w^k = \frac{N}{N-k}, \]

\begin{align*}
& \text{do } k = 1, \ldots, N-1 \\
& \quad G \rightarrow w^k F(u) + (1 - w^k)G \\
& \quad u \rightarrow u + G \Delta t \\
& \text{end do}
\end{align*}

**Memory consumption:** $2 \times \dim\{\text{model state}\}$

**$F$-evaluation:** 1 per time step

**Accuracy:** $(N \leq 4)$

\[ O((N \Delta t)^N) \text{ (every } N \text{ steps)} \]

\[ O(N \Delta t) \text{ (in between)} \]

### (existing) Leap-frog with Robert-Asselin Filter

\begin{align*}
& \text{do } k = 1, \ldots, \\
& \quad u^{+1} \rightarrow u^{-1} + 2 \Delta t F(u^0) \\
& \quad u^0 \rightarrow u^0 + \alpha(u^{+1} - 2u^0 + u^{-1}) \\
& \quad u^{-1} \rightarrow u^0 \\
& \quad u^0 \rightarrow u^{+1} \\
& \text{end do}
\end{align*}

**Memory consumption:** $2 \times \dim\{\text{model state}\}$

**$F$-evaluation:** 1 per time step

**Accuracy:**

\[ O(\Delta t) \]

### 4th order Runge-Kutta

\begin{align*}
& \text{do } k = 1, \ldots, \\
& \quad h^1 \rightarrow F(u) \\
& \quad h^2 \rightarrow F(u + \frac{\Delta t}{2} h^1) \\
& \quad h^3 \rightarrow F(u + \frac{\Delta t}{2} h^2) \\
& \quad h^4 \rightarrow F(u + \Delta t h^3) \\
& \quad u^{+} = \frac{\Delta t}{6}(h^1 + 2h^2 + 2h^3 + h^4) \\
& \text{end do}
\end{align*}

**Memory consumption:** $4 \times \dim\{\text{model state}\}$

**$F$-evaluation:** 4 per time step

**Accuracy:**

\[ O(\Delta t^4) \]

### ODE to be solved

\[
\frac{du}{dt} = F(u)
\]
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. **Model description**
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Delivurable

5. References
AGCM: SPEEDY model

- A fast AGCM with simplified physical parameterizations
- Developed in ICTP (Italy) by Drs. F. Molteni and F. Kucharski
- Horizontal Discretization:
  - Spectral Representation with Spherical Harmonics truncated at total wavenumber 30 (T30) ≈ 400km mesh
- Vertical Discretization:
  - 8-layers Finite Difference on $\sigma$-coordinate
- Temporal Discretization:
  - Leap-Frog scheme with Robert-Asselin Filter
  - (1$^{st}$ order Forward Euler for the physical parameterizations)
The equations solved:
the "primitive" equation system (PDEs)
on a spherical geometry + parametrized processes

Dynamical Core

Sub-grid Parametrizations
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. **Stiffness-problem and semi-implicit method**
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Deliverable

5. References
Stiffness Problem

- The “Primitive” Equations = Stiff system:
- Fast but insignificant modes (= gravity waves) are superposed on the slow but meteorologically meaningful modes (= Rossby waves)
- Due to the CFL condition for the fast modes, we need very small timestepping $\Delta t$

→ Semi-implicit method (Robert, 1969)
Semi-Implicit for Leapfrog (1) Formulation

ODE to be solved:

\[
\frac{du}{dt} = F^E(u) + L^Iu
\]

Slow modes \hspace{1cm} Fast modes

Solve this discretized equation

\[
\frac{u^{n+1} - u^{n-1}}{2\Delta t} = F^E(u^n) + L^I(\alpha u^{n+1} + (1 - \alpha)u^{n-1})
\]

Explicit \hspace{1cm} Implicit
Semi-Implicit for Leapfrog (2)

How to solve

Define: \[ \delta u = \frac{u^{n+1} - u^{n-1}}{2\Delta t} \]

Substitute \[ u^{n+1} = u^{n-1} + 2\Delta t \delta u \]
to the discretized equation,

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

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

Once you get \( \delta u \), you can integrate the equation by

\[ u^{n+1} = u^{n-1} + 2\Delta t \delta u \]
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climates
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Deliverable

5. References
Semi-implicit Lorenz $N$-cycle

Explicit (original)

\[
\text{do } k = 0, \ldots \\
\quad w \leftarrow w^{\text{mod}(k,N)} \\
\quad G \leftarrow wF(u) + (1 - w)G \\
\quad u \leftarrow u + G\Delta t \\
\text{end do}
\]

Semi-implicit

\[
\text{do } k = 0, \ldots \\
\quad w \leftarrow w^{\text{mod}(k,N)} \\
\quad G \leftarrow wF^E(u) + (1 - w)G \\
\quad \delta u = (I - \alpha \Delta t L^I)^{-1}(G + L^I u) \\
\quad u \leftarrow u + \Delta t \delta u \\
\text{end do}
\]
Accuracy(Consistency) & Stability Analysis: Method

- Following Durran (1991, 1999; MWR) and Williamson (2011; MWR), apply semi-implicit modification to the second term of the equation:
  \[ \frac{du}{dt} = i\omega_L u + i\omega_H u \]
- Examine the modulus of Amplification factor \( A \).
- If \( |A| < 1 \), the scheme is stable.
- Range of interest:
  \[ \omega_L \Delta t < 0.5, \quad \omega_H \Delta t > 2 \]
i.e., CFL condition is met for low-frequency part but is violated for high-frequency part.
Accuracy(Consistency)

• Truncation Error:

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

• By taking \( \alpha=1/2 \) (Crank-Nicolson), the scheme becomes 2\textsuperscript{nd}-order
Plots of $|A|-1$ \((\alpha=1/2: \text{Crank-Nicolson})\)

Leapfrog with R/A filter: 
Stable almost everywhere

Lorenz 1-cycle: 
(=Forward Euler): 
Unstable everywhere

Lorenz 2-cycle: 
Unstable everywhere

Lorenz 3-cycle: 
Absolutely Unstable for \(> 1.5\)

Lorenz 4-cycle: 
Absolutely Unstable for \(> 2.7\)
Plots of $|A|^{-1}$ \ ($\alpha=1$: Backward Euler)

Stability is good, but damping is too strong (~50%)
Stability for $\alpha=1/2$: Crank-Nicolson

$N=1,...,6$

White: stable $|A|<1$

Gray: Unstable $|A|>1$
Stability for $\alpha=1$: Backward Euler

$N=1,\ldots,6$

White: stable $|A| < 1$

Gray: Unstable $|A| > 1$
Stability: Summary

- Crank-Nicolson semi-implicit $N$-cycle is unstable for $N=1,2$
- Stability region is largest for $N=4$
- More unstable than Leapfrog.

- Backward Euler semi-implicit $N$-cycle is more stable, but damping is too strong
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. **Code Validation**
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Deliverable

5. References
Code Validation: Idea

• Lorenz $N$-cycle:
  – Lorenz 1-cycle is equivalent to Forward Euler, which is built-in in the SPEEDY model.
  – Compare Lorenz 1-cycle with Forward Euler.

• RK4:
  – For a linear system, RK4 with $4\Delta t$ is equivalent to Lorenz 4-cycle with $\Delta t$.
  – Remove all nonlinear terms from SPEEDY and Compare RK4 ($4\Delta t$) with Lorenz 4-cycle ($\Delta t$).
Code Validation: Results

• Outputs of single step integrations of Lorenz 1-cycle and Forward Euler from the same initial condition are compared using UNIX diff command.
  – Result → no difference (Success)!

• Similarly, outputs of single step integration of RK4 and four-step integration of Lorenz 4-cycle from the same initial condition are compared using UNIX diff command.
  – Result → no difference (Success)!
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Delivarable

5. References
Verification: Dynamical-Core test cases

• Standardized tests for dynamical cores of AGCM proposed by Jablonowski and Williamson (2006) which consists of two tests:

1. Steady-State test:
   – A model is integrated from an analytical steady-state solution of the primitive equation
   – The model is evaluated by how well it can keep the steady-state intact.

2. Baroclinic-Wave test:
   – The model is integrated with initial and boundary conditions which is designed to produce an idealized baroclinic wave (= extratropic cyclones and highs)
Baroclinic-wave Test: Result (Lorenz 4-cycle)
Snapshots of surface-pressure at Day 9

Hi-res. reference solution

Leapfrog (dt=20min)

RK4 (dt=1min)

Leapfrog (dt=1min)

N-cycle (dt=20min)
Order estimation

• Lorenz 4-cycle is supposedly of 4\textsuperscript{th}-order, while Leapfrog (with R/A filter) is only of 1\textsuperscript{st}-order
• Confirm this by plotting $L^2$ error vs. $\Delta t$ on a log-log plane.
• Experimental set-up: Jablonowski-Williamson baroclinic-wave test
• Measure of the error: difference in surface pressure with respect to the reference solution produced by RK4 with $\Delta t=0.5\min$ in $L^2$-norm
Result: $L^2(P_s)$ at $t=3$ days
Result: $L^2(Ps)$ at $t=5$ days
Result: $L^2(Ps)$ at $t=10$ days
Dynamical-core test
Summary of the results

• For $\Delta t \leq 10\text{min}$, the order of Lorenz 4-cycle is $3^{rd} \sim 4^{th}$

• For a large $\Delta t$, A-B-B-A cycle is inferior to version A or version B, which contradicts with Lorenz (1971)’s claim.

• Possible reasons:
  – Cancellation of truncation errors of version A and B does may hold because of the introduction of semi-implicit method.
  – Cancellation between A and B itself is not attained for the AGCM.
  – A-B-B-A cycle is more unstable for nonlinear models than A-only or B-only.

→ To be examined
Outline

1. Introduction

2. Phase I: Implementation of Lorenz $N$-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz $N$-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Delivarsable

5. References
Inclusion of Physics

• Having proved that $N$-cycle works without physical parameterizations, I next tried to include physics in the $N$-cycle SPEEDY model.
Problem encountered in the Last Semester: 
*N*-cycle and RK4 blow up when run with physics on

Resolved: It was a Stupid Bug!
Stability

• After fixing the bug, I tried to find the largest $\Delta t$ with which the $N$-cycle can integrated stably.

Bad news: the largest stable $\Delta t$
• for $N$-cycle : 15 mins, whereas
• for Leapfrog: 40mins.

Possible remedy (with some accuracy degradation) :
• the largest stable $\Delta t$ can be made 30mins
• by using Backward Euler for the gravity-wave part (instead of the default Crank-Nicolson)

• $N$-cycle can be integrated with $\Delta t$=15mins. for at least 100 years (=3,504,000 steps)
Comparison of Climatology (= long-term mean)

• In climate application, it is important that the long-term mean of the atmospheric state (called *climatology*) does not change.

• ➔ Statistically compared the climatologies using Welch’s *t*-test.

• Examined Climatologies:
  two seasons (DJF: 12-2 and JJA: 6-8), each from 3 models:
  – Leapfrog with dt=40mins (default)
  – 4-cycle (Crank-Nicolson for gravity-wave) dt=15mins
  – 4-cycle (Backward Euler for gravity-wave) dt=30mins
**t-test for the difference of climatologies**

- **Experimental design:**
  - Run all the models from the same initial condition (state of rest)
  - Integrate for 30 years.
  - Discard the first 10 years as a spin-up.
  - Use the remaining 20 years as the samples.

- **Null-hypothesis:**
  - Climatologies (= sample means) are taken from the same population.

- The probability of the differences between two sampled climatologies being larger than the observed difference under the null-hypothesis is computed.

- If this probability is larger than 95%, then the null-hypothesis is not rejected.
Results for T, Z, U and V, MSLP and OLR: no significant difference!

- all results are uploaded on
  http://www.atmos.umd.edu/~dhotta/speedy_clim2/clim.html

Example: 
- Leapfrog + CrankNicolson
- 4-cycle + CrankNicolson
- 4-cycle + Backward Euler

climatology

Probability of null-hypothesis not being rejected

.7  .9  >.95

.7  .9  >.95
Conclusion for Phase I

• Designed semi-implicit version of $N$-cycle
  – Analyzed stability using a toy-model
  – 4-cycle is the most stable
  – For semi-implicit method, Crank-Nicolson is more accurate than Backward-Euler but is more unstable

• Verification through Dynamical-Core test:
  – 4-cycle with Crank-Nicolson exhibits order-of-accuracy which is higher ($2^{nd} \sim 3^{rd}$) than filtered Leapfrog ($1^{st}$-order)
Schedule for Phase 1:
Planned and Actual

**Planned**
- Implement RK4 and $N$-cycle, **Nov.**
- Write the mid-year report, prepare the oral presentation, **Dec.**
- Switch-off physical parameterizations, prepare flat topography, **Jan.**
- Perform the dynamical core tests, **Feb.**

**Actual**
- Formulate semi-implicit $N$-cycle, **Oct.**
- Implement RK4 and $N$-cycle, **Nov.**
- Switch-off physical parameterizations, prepare flat topography, **Dec.**
- Perform the dynamical core tests, **Dec.**
- Write the mid-year report, prepare the oral presentation, **Dec.**
- Coded a bug, and fixed the bug, **Jan. (winter break)**
- Compare climatologies, performed statistical test, **Feb.**
Phase II: Empirical characterization of Model Errors
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Delivaraible

5. References
Phase 2: Approach

• Objective: Characterize the model errors due to temporal discretizations

• Take the Truth from NCEP/NCAR reanalysis (Kalnay et al. 1996)
  NCEP=National Centers for Environmental Prediction
  NCAR=National Center for Atmospheric Research

• Extract model errors by applying the method of Danforth et al. (2007) to the models with:
  1. the original scheme (Leap-Frog; $M^{LF}$)
  2. Lorenz N-cycle scheme ($M^{NCYC}$)

• (time permitting) Correct the model errors on-line during the course of model integration
  (⇒ Phase 3&4)
Phase 2: Algorithm

1. Generate initial values from the Truth (NCEP/NCAR reanalysis)
2. Perform short-range forecasts using the 2 models ($M^{LF}$, $M^{NCYC4}$) from the initial conditions
3. find the bias of the model errors for each model
4. Build the covariance matrix
   \[
   \left\langle (x(t) - \bar{x}) (M^{true}(x(t)) - M^{LF}(x(t)))^T \right\rangle
   \]
5. Extract the dominant modes by conducting SVD
Interpolation from NCEP/NCAR grid to SPEEDY grid

• Obtained the original code (used in Danforth et.al (2007)) and wrote a code that does exactly the same operation

• Original code:
  – written in MatLab script
  – performs Simple linear interpolation in 3D

• My code:
  – wrote in NCL (NCAR Command Language)
  – Basically a line-by-line translation from MatLab to NCL

• Validation:
  – Method:
    • produce data on SPEEDY grid from NCEP/NCAR data using the original code and my code
    • compare the two outputs, one from the original, the other from my code
  – Result: the two outputs agreed within single-precision rounding error
Outline

1. Introduction

2. Phase I: Implementation of Lorenz N-cycle to SPEEDY model
   1. Approach
   2. Algorithms
   3. Model description
   4. Stiffness-problem and semi-implicit method
   5. Semi-implicit Lorenz N-cycle
   6. Code Validation
   7. Verification: Dynamical-core test
   8. Inclusion of Physics & Comparison of Climatologies
   9. Conclusion
   10. Schedule, planned and actual

3. Phase II: Empirical Characterization of Model Errors
   1. Approach
   2. Algorithm
   3. Interpolation
   4. Model Error Bias: results
   5. Schedule, planned and actual

4. Outcome/Deliverable

5. References
Model Error Bias : Results
Zonal wind at 200hPa

From Danforth et.al (2007)
Model Error Bias: Results
Temperature at 850hPa

From Danforth et. al (2007)
Model Error Bias : Results
Specific Humidity at 300hPa

From Danforth et.al (2007)
Model Error Bias: Results
Specific Humidity at 700hPa

From Danforth et al. (2007)
Model Error Bias : Summary

• For all variables, Leapfrog and $N$-cycle produce almost identical bias pattern
  • Interpretation:
    bias is dominated by the model’s deficiencies associated with physical parameterizations

• They are consistent with Danforth et.al (2007)
Plan for Phase II

In Danforth et.al (2007),

1. Error samples $\{\delta x\}$ are produced with the original model $M$
2. Bias $<\delta x>$ is computed
3. The model is modified by incorporating nudging to $-<\delta x>$ to yield $M^+$
4. Error samples $\{\delta x^+\}$ are resampled using the de-biased model $M^+$
5. Diurnal errors are extracted from $\{\delta x^+\}$ by performing EOF analysis and retaining the first two dominant modes
6. The de-biased model $M^+$ is again modified by incorporating nudging to negative of diurnal biases to yield $M^{++}$
7. Error samples $\{\delta x^{++}\}$ are again resampled using $M^{++}$
8. Perform SVD analysis to extract dominant co-variation of model error $\delta x^{++}$ and the anomaly of the model states

while, in my original plan,

1. Error samples $\{\delta x\}$ are produced with the original model $M$
2. Bias $<\delta x>$ is computed
3. The model is modified by incorporating nudging to $-<\delta x>$ to yield $M^+$
4. Error samples $\{\delta x^+\}$ are resampled using the de-biased model $M^+$
5. Diurnal errors are extracted from $\{\delta x^+\}$ by performing EOF analysis and retaining the first two dominant modes
6. The de-biased model $M^+$ is again modified by incorporating nudging to negative of diurnal biases to yield $M^{++}$
7. Error samples $\{\delta x^{++}\}$ are again resampled using $M^{++}$
8. Perform SVD analysis to extract dominant co-variation of model error $\delta x^{++}$ and the anomaly of the model states

3. Perform SVD analysis to extract dominant co-variation of model error $\delta x^{++}$ and the anomaly of the model states
4. Validation: Results are compared with Danforth et.al (2007)
Plan for Phase II

- Danforth et.al (2007) involves much more tricks than I originally planned.
- Reproducing all the procedures in Danforth et.al (2007) is impossible given that I have only 2-weeks left.

→
- I continue the original plan, but modify the Validation part.
- New Validation:
  - Check orthogonality between the extracted modes
Conclusion for Phase II

• Generated samples of model error
• Computed biases for Leapfrog and Lorenz 4-cycle, compared them with Danforth et.al (2007)

→ Result:
  – No significant bias improvements by using a better temporal scheme. Perhaps dominated by physics errors
  – Consistent with Danforth et.al (2007)

• TODO (in two weeks):
  – SVD analysis to identify the dominant co-varying modes between model state anomaly and model error
  – Comparison with Danforth et.al (2007)
## Schedule for Phase 2: Planned and Actual

**Planned**

- Generate initial values from the NCEP/NCAR reanalysis, end of Feb.
- build the bias and covariance matrix, Mar.
- Code and test a program for SVD, Apr.
- Compare the model errors for the new and the original schemes, May.
- Write the final report (paper draft), May.

**Actual**

- Generate initial values from the NCEP/NCAR reanalysis, end of Feb.
- Compute the bias, Mar.
- Plot the bias and compare it with Danforth et.al, Apr. (← Now I’m here)
- Code and test a program for SVD, May.
- Compare the model errors for the new and the original schemes, May.
- Write the final report (paper draft), May.
Outcome/Deliverables

Phase 1:
• Upgraded code for SPEEDY model ✔
  - subroutines for Lorenz N-cycle and 4th order Runge-Kutta ✔
• Test-case results for the SPEEDY model (both for the original scheme and the new schemes) ✔

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

Phase 2:
• Archive of the model errors ✔
• Bias of model errors ✔
• Pairs of Singular Vectors for the model state and the model error
• Code for performing SVD

To be completed in two weeks
Bibliography

Lorenz N-cycle

SPEEDY model

SPEEDY-LETKF

Atmospheric GCM Dynamical Core test cases

NCEP/NCAR reanalysis

Model Error Correction
back-up slides
Swinging-pendulum problem

- A simple nonlinear test-bed for semi-implicit schemes.
- Fast oscillation = elastic spring
- Slow oscillation = pendulum
- Fast mode is treated implicitly, slow mode explicitly.

For this test, I use $\Delta t=0.075$ which gives $\omega_{\text{LOW}}\Delta t=0.225$, $\omega_{\text{HIGH}}\Delta t=2.25$,

\[
\begin{align*}
\dot{\eta} &= v_{\eta} , \\
\dot{v}_{\eta} &= -\omega_{\text{low}}^2 (1-\cos \theta) - \omega_{\text{high}}^2 \eta + (1+\eta)v_{\theta}^2 , \\
\dot{\theta} &= v_{\theta} , \quad \text{and} \\
\dot{v}_{\theta} &= \frac{-\omega_{\text{low}}^2 \sin \theta - 2v_{\eta}v_{\theta}}{1+\eta} .
\end{align*}
\]
Leapfrog with RA filter ($\alpha=0.01$)
4-cycle with semi-implicit correction on every 4 steps (Crank-Nicolson)

\[ \theta \]

\[ \eta \]

Energy

RK4 with \( \Delta t=10^{-6} \)

+ Leapfrog
4-cycle with semi-implicit correction on every 4 steps (Backward)
➔ stable but very dissipative

θ

η

Energy

RK4 with $\Delta t=10^{-6}$

+ Leapfrog
4-cycle with semi-implicit correction on every time step ➔ unstable
Explicit 4-cycle $\to$ unstable

- $\theta$
- $\eta$
- Energy

RK4 with $\Delta t=10^{-6}$

Leapfrog
Summary

Consistent with the linear analysis,

• N-cycle with semi-implicit on each time step
  ➔ unstable

• N-cycle with semi-implicit on every N steps
  ➔ Crank-Nicolson: stable, but comparable accuracy with Leapfrog
  ➔ Backward Euler: stable, very dissipative
Reference:

Runge-Kutta 4^{th}-order scheme with semi-implicit correction
Introduction

• I implemented semi-implicit Runge-Kutta 4th-order scheme to SPEEDY model and found that
  – for Crank-Nicolson, the model blows up, even for very small dt (1min).
  – for Backward Euler, the model is too diffusive that the baroclinic-wave dynamical core test fails to produce the baroclinic wave.

• Explicit Runge-Kutta 4th-order scheme with small dt (5min) works fine for the dynamical core test.

• → examine stability using the toy-model
Method

• Following Durran (1991, 1999; MWR) and Williamson (2011; MWR), apply semi-implicit modification to the second term of the equation:
  \[
  \frac{du}{dt} = i\omega_L u + i\omega_H u
  \]

• Examine the modulus of Amplification factor \( A \).
• If \( |A| < 1 \), the scheme is stable.
• Range of interest:
  \[
  \omega_L \Delta t < 0.5, \quad \omega_H \Delta t > 2
  \]
  i.e., CFL condition is met for low-frequency part but is violated for high-frequency part.
Algorithm:
RK4 for $\omega_L$, Crank-Nicolson for $\omega_H$

\[
\begin{align*}
    h^1 &= i\omega_L u^n \\
    u_2^* &= u^n + \frac{\Delta t}{2} h^1 \\
    h^2 &= i\omega_L u_2^* \\
    u_3^* &= u^n + \frac{\Delta t}{2} h^2 \\
    h^3 &= i\omega_L u_3^* \\
    u_4^* &= u^n + \Delta t h^3 \\
    h^4 &= i\omega_L u_4^* \\
    G &= \frac{1}{6}(h^1 + 2h^2 + 2h^3 + h^4) \\
    \frac{u^{n+1} - u^n}{\Delta t} &= G + \{\beta i\omega_H u^{n+1} (1 - \beta) i\omega_H u^n\}
\end{align*}
\]

Truncation Error:

\[
\frac{u^{n+1} - u^{Exact}}{u^n} = \frac{1}{2} \omega_H ((1 - 2\beta)\omega_H - 2(\beta - 1)\omega_L) \Delta t^2 + O(\Delta t^3)
\]

the accuracy is only 1st order
Plot of $|A|^{-1}$

$\beta = 1/2$: Crank-Nicolson
- Absolutely unstable

$\beta = 1$: backward Euler
- Absolutely stable, but extremely dissipative
Summary

• Runge-Kutta 4\textsuperscript{th}-order scheme with semi-implicit time is
  – only of first order (with respect to fast modes)
  – absolutely unstable if Crank-Nicolson is used
  – absolutely stable if Backward Euler is used, but the numerical damping is too strong (more than halving on every time step)

• all of the above are consistent with what I found for the SPEEDY model.
Conclusion

• Runge-Kutta 4\textsuperscript{th}-order scheme with semi-implicit time-stepping for gravity waves is impossible (either unstable or too dissipative)
• The accuracy becomes only of 1\textsuperscript{st} order.

• Since the motivation for implementing Runge-Kutta 4\textsuperscript{th}-order scheme is to produce a reference solution, I will not try to resolve this issue, and use the explicit scheme with small dt.
Stability analysis of Forward Euler “split physics” for Leapfrog, Lorenz 4-cycle and RK4
Toy model: linear advection-diffusion equation

• Undiscretized equation:

\[
\frac{du}{dt} = i\omega u - \beta u
\]

• The first term of the RHS simulates the dynamics of AGCM, and the second the physics.
Discretization: Leapfrog(dyn) + Euler(phys)

- Discretized equation:
  \[ \frac{u^{n+1} - u^{n-1}}{2\Delta t} = i\omega u^{n} - \beta u^{n-1} \]

- The amplification factor \( A \) satisfies the following equation:
  \[ \frac{A^2 - 1}{2\Delta t} = i\omega A - \beta \]
  \[ \Rightarrow A = i\omega \Delta t \pm \sqrt{1 - (\omega \Delta t)^2 - 2\Delta t\beta} \]

- (+) and (-) correspond, respectively, to physical and computational modes.

- The scheme is stable if the maximum of the moduli of them is less than 1.
Discretization: Leapfrog for both dyn & phys

• Discretized equation:
  \[ \frac{u^{n+1} - u^{n-1}}{2\Delta t} = i(\omega + i\beta)u^n \]

• The amplification factor \( A \) satisfies the following equation:
  \[ \frac{A^2 - 1}{2\Delta t} = i(\omega + i\beta)A \]
  \[ \Rightarrow A = i(\omega + i\beta)\Delta t \pm \sqrt{1 - (\omega + i\beta)^2\Delta t^2} \]

• (+) and (-) correspond, respectively, to physical and computational modes.
• The scheme is stable if the maximum of the moduli of them is less than 1.
Discretization: Lorenz 4-cycle (dyn) + Euler(phys)

• Discretized equation:

\[ u^{n+N} = u^* - N \beta \Delta t u^n, \]

where \( u^* = u^n \left( \sum_{k=0}^{N} \frac{1}{k!} (iN\omega \Delta t)^k \right) \)

• Amplification factor (per time step):

\[ A = \left\{ \left( \sum_{k=0}^{N} \frac{1}{k!} (iN\omega \Delta t)^k \right) - \beta N \Delta t \right\}^{1/N} \]
Discretization: Lorenz 4-cycle for both dyn & phys

- Discretized equation:

\[ u^{n+N} = u^n \left( \sum_{k=0}^{N} \frac{1}{k!} (iN(\omega + i\beta)\Delta t)^k \right) \]

- Amplification factor (per time step):

\[ A = \left( \sum_{k=0}^{N} \frac{1}{k!} (iN\omega\Delta t)^k \right)^{1/N} \]
Discretization: RK4(dyn) + Euler(phys)

1. Discretized equation:

\[ u^{n+1} = u^* - \beta \Delta t u^n, \]

where \( u^* = u^n \left( \sum_{k=0}^{N} \frac{1}{k!} (i \omega \Delta t)^k \right) \)

2. Amplification factor:

\[ A = \left( \sum_{k=0}^{N} \frac{1}{k!} (i \omega \Delta t)^k \right) - \beta \Delta t \]
Discretization: RK4 for both dyn&phys

- Discretized equation:

\[ u^{n+1} = u^n \left( \sum_{k=0}^{N} \frac{1}{k!} (i(\omega + i\beta)\Delta t)^k \right) \]

- Amplification factor (per time step):

\[ A = \sum_{k=0}^{N} \frac{1}{k!} (i\omega \Delta t)^k \]
Leapfrog:
“split physics” stabilizes the otherwise absolutely unstable scheme

Leapfrog for both dyn & phys: absolutely unstable

Leapfrog (dyn) + Euler (phys): Stable within the triangle

\[
\max(|A_{\text{phys}}|, |A_{\text{compl}}|) \text{ for Leapfrog (for dynamics and physics)}
\]

\[
\max(|A_{\text{phys}}|, |A_{\text{compl}}|) \text{ for Leapfrog + Euler (split physics)}
\]
Lorenz 4-cyle: “split physics” destabilizes the scheme

Lorenz 4-cycle for dyn & phys:
absolutely unstable

L4-cycle (dyn) + Euler (phys):
Stable within the triangle
RK4:
again, “split physics” destabilizes the scheme

RK4 for dyn & phys: absolutely unstable

RK4 (dyn) + Euler (phys): Stable within the triangle
Summary

• For leapfrog, doing “split physics” works very well (stabilizes the scheme)
• However, for Lorenz 4-cycle (and equivalently, for RK4), “split physics” acts to destabilized the scheme.
• The latter is consistent with what I found by doing “split physics” with SPEEDY model.
The Bug (1)

Leapfrog (default)

\[\text{PROGRAM agcm} \]
\[\text{CALL iniall ()} \]
\[\text{IDAY}=0; \text{CALL FORDATE()} \]
\[\text{CALL STEPONE()}! 1^{\text{st}} \text{ step by Euler Forward} \]
\[\text{DO}! \text{ loop over a month} \]
  \[\text{DO}! \text{ loop over a day} \]
    \[\text{CALL FORDATE()} \]
    \[\text{CALL STLOOP()}! \text{ integrate for a day} \]
  \[\text{END DO} \]
\[\text{END DO} \]

N-cycle (with bug)

\[\text{PROGRAM agcm} \]
\[\text{CALL iniall ()} \]
\[\text{DO}! \text{ loop over a month} \]
  \[\text{DO}! \text{ loop over a day} \]
    \[\text{CALL FORDATE()} \]
    \[\text{CALL STLOOP()}! \text{ integrate for a day} \]
  \[\text{END DO} \]
\[\text{END DO} \]
The Bug (2)

N-cycle (fixed)

```
PROGRAM agcm
CALL iniall()
IDAY=0; CALL FORDATE()

DO ! loop over a month
  DO ! loop over a day
    CALL FORDATE()
    CALL STLOOP() ! integrate for a day
  END DO
END DO
```

N-cycle (with bug)

```
PROGRAM agcm
CALL iniall()

DO ! loop over a month
  DO ! loop over a day
    CALL FORDATE()
    CALL STLOOP() ! integrate for a day
  END DO
END DO
```