Parallel Multigrid for Large-Scale Least Squares Sensitivity

by

Steven A. Gomez

B.S., Massachusetts Institute of Technology (2011)

Submitted to the Department of Aeronautical and Astronautical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautical and Astronautical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Aeronautical and Astronautical Engineering
May 23, 2013

Certified by .....
Qiqi Wang
Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Etyan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Parallel Multigrid for Large-Scale Least Squares Sensitivity

by

Steven A. Gomez

Submitted to the Department of Aeronautical and Astronautical Engineering
on May 23, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautical and Astronautical Engineering

Abstract

This thesis presents two approaches for efficiently computing the "climate" (long-time average) sensitivities for dynamical systems. Computing these sensitivities is essential to performing engineering analysis and design. The first technique is a novel approach to solving the "climate" sensitivity problem for periodic systems. A small change to the traditional adjoint sensitivity equations results in a method which can accurately compute both instantaneous and long-time averaged sensitivities. The second approach deals with the recently developed Least Squares Sensitivity (LSS) method. A multigrid algorithm is developed that can, in parallel, solve the discrete LSS system. This generic algorithm can be applied to ordinary differential equations such as the Lorenz System. Additionally, this parallel method enables the estimation of climate sensitivities for a homogeneous isotropic turbulence model, the largest scale LSS computation performed to date.

Thesis Supervisor: Qiqi Wang
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank my advisor Professor Qiqi Wang for the continued support over the past two years. The work required for this thesis would not have been possible without his intuition, his extraordinary patience, and his ability to quickly identify any bugs. I would like to thank ANSYS, Inc. for their ongoing financial support during this project. I would also like to thank my fellow students Patrick Blonigan for his extensive help in developing the multigrid method presented here, and Eric Dow for his help utilizing the ACDL cluster to implement my parallel algorithms. Finally, I would like to thank my parents Julio and Rosemary Gomez for their lifelong support, my girlfriend Julie Moyer for putting up with me, and our awful cat Clifton.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Sensitivity analysis is an important tool for engineering design. It deals with finding the sensitivity of the outputs of a system, due to arbitrary perturbations to system parameters. Gradient based optimization techniques rely on sensitivity information to find optimal design specifications for complex engineering systems. Therefore, accurately and cheaply computing this sensitivity can make a large impact on the ability of engineers to effectively explore a particular design space. Adjoint and forward sensitivity approaches can be used to efficiently compute the sensitivity of a system with respect to perturbations in the equations that govern that system. However, while these methods have been very successful in the realm of steady state problems, there has been some difficulties in applying the methods toward time-dependent, or dynamical, systems.

The long-time-averaged properties of a dynamical system are of particular interest when dealing with unsteady systems. In the field of climate science, for example,

there is a substantial need to predict the effects of climate forcing from humans or otherwise. Likewise, in the Aerospace field, the issue of understanding, and designing in situations where unsteady, turbulent fluid flow occurs is of great importance. There are many problems where current turbulence models fail, and only direct simulation of the flow field can model all of the complex dynamics that occur. In these situations scientists and engineers are often most interested in the "climate", that is, the averaged properties of the system, rather than information and one particular instance in time.

However, analyzing the "climate" of a dynamical system, presents many difficulties even for simple chaotic maps and ordinary differential equations (ODE). The problem worsens for the chaotic partial differential equations (PDE) that appear in turbulent fluid flow problems. These problems even persist in comparatively simple periodic systems.Climate sensitivity analysis, which is critical for understanding these systems, has remained computationally out of reach for all but the most simple problems. Only simple finite difference methods have been effective at estimating sensitivities for general chaotic systems. Computation of sensitivities with these methods is inefficient and the computational effort scales poorly with the problem size, parameter space, and time scale needed. The more advanced adjoint and tangent sensitivity methods have been difficult to adapt to finding long-time averaged quantities for ergodic dynamical systems [9, 14, 15, 20]. A recent technique, Least Squares Sensitivity (LSS) has proposed a method for accurately computing the climate sensitivity for chaotic and periodic systems, but the computational feasibility is still an issue.

1.2 Climate Sensitivity Problem

This thesis will tackle the climate sensitivity problem, and methods for computing these sensitivities efficiently. The climate problem starts with some set of differential equations (1.1) the model the dynamics of the system of interest.

$$\frac{du}{dt} = f(u;\beta) \tag{1.1}$$

$$u \in \mathbb{R}^M, \quad \beta \in \mathbb{R}^p \tag{1.2}$$

Solutions to this differential equations u exist in some M-dimensional phase space where it orbits throughout time infinitely. This $u(t)$ represent anything from the abstract solution vector of an ODE, to the spatially discretized velocity field in an incompressible Navier-Stokes problem. The system dynamics f, are parameterized by some vector β, that represent all the possible problem specific parameters that can affect the solution u. For example, this may include a set of scalar parameters or an entire field of data that can affect the properties of the system. In the climate problem, the long-time average of some function J is examined.

$$\bar{J}(\beta) = \lim_{T\to\infty} \frac{1}{T} \int_0^T J(\ u(t),\ \beta)\ dt \tag{1.3}$$

If the original governing equations are ergodic then the long time average quantities are independent of any initial condition and are dependent only on the system itself, and the parameters β. The goal is then to find the sensitivity of these long time average quantities, $\bar{J}$, with respect to perturbations in the governing differential equations, via β.

$$\frac{d\bar{J}}{d\beta} = ??? \tag{1.4}$$

This thesis will discuss ways of finding this sensitivity $\frac{d\bar{J}}{d\beta}$, and computing its value efficiently for large scale problems.

1.3 Previous Work

There have been many attempts at dealing with the problem of climate sensitivity for dynamical systems and chaos in general. Early work by Edward Lorenz first identified the properties of chaos [16] in a chaotic ODE later named the Lorenz System. The main property of chaotic systems is a sensitivity to initial conditions, in other words, small perturbations in the solution at a given time can result in vastly different solutions at a later time. For this reason, the climate, rather than instantaneous solution information, is a more feasible property to examine in chaotic systems. However, Lorenz quickly noticed the difficulty of computing the climate from system information alone [17].

Lea et. al., [14] quantified the problems of using adjoint sensitivity analysis naively on simple chaotic systems, and later confirmed the issue persists in chaotic ocean simulations [15]. Using a typical adjoint formulation, the adjoint equations are solved backwards in time from some terminal condition. However, in chaotic settings, these adjoint equations are unstable and solutions grow unbounded backwards in time [14]. The longer the integration time, the larger the adjoint solutions would grow. One possible solution to this issue is to limit the integration time, but average many these short-time adjoints together. This ensemble adjoint approach was examined by Lea et a. [15] and Eyink et al. [9] for the Lorenz System. However the ensemble approach requires a large number of ensemble solutions to compute approximate sensitivities [15, 9].

Thurburn proposed an alternate approaches, the Fokker-Planck adjoint, describ-

ing the solution space by a probability density [22], and computing sensitivities via density perturbations. However, the difficulty of discretizing the phase-space of solutions limits its feasibility. In 2007 Abramov and Majda [1] developed a method using fluctuation-dissipation theory from statistical dynamics to estimate sensitivities for statistical quantities, and has been shown to be successful for certain classes of chaotic systems [2]. Krakos et. al. [13] have developed a method for periodic climate sensitivity problems. This windowing method has made periodic problems feasible even for large scale systems, but this thesis will propose an alternative with several advantages. Recently, Wang [24], proposed a method for computing sensitivities using a shadow trajectory, and relying on the computation of Lyapunov covariant vectors. This shadow-trajectory based method was refined into the development of Least Squares Sensitivity [23]. The climate sensitivity problem is turned into an optimization problem and ultimately the adjoint and tangent sensitivities come as a result of solving a boundary value problem in time, rather than a terminal or initial value problem. This thesis will also examine methods for efficiently solving the LSS equations for large-scale systems.

Chapter 2

Split Periodic Adjoint

2.1 Introduction

The breakdown of traditional sensitivity occurs even in the most simple of periodic dynamical systems. While the occurrence of true periodic systems is rare, the difficulty in efficiently computing sensitivities for periodic systems can provide some insight into the problem for chaotic systems. This section 2.2 will formulate the adjoint sensitivity problem in a new way by splitting the hypothetical perturbation applied to the system into multiple parts. Then section 2.3 will illustrate how this formulation results in a simple set of equations for computing adjoint sensitivities. Finally section 2.4 will describe a method for computing these periodic sensitivity gradients in a more efficient way than previously attainable.[13]

2.2 Formulation

Again, there is a differential equation, defined by f, for finding solutions $u(t)$ given some parameters β.

$$\frac{du}{dt} = f(u;\beta) \tag{2.1}$$

This defines a periodic system for some range of β's. Solutions for $u(t)$ approaches some unique periodic limit cycle with a period $T = T(\beta)$. The objective function of interest is $J(u;\beta)$, and its long-time average $\bar{J}(\beta)$. We assume J does not have an explicit dependence on β, because the sensitivity due to an explicit β dependence can be computed separately. The goal is to find the sensitivity of this long-time average to perturbations in the parameters β.

$$\bar{J}(\beta) = \lim_{T\to\infty} \frac{1}{T} \int_0^T J(\ u(t);\beta\ )\ dt \tag{2.2}$$

$$\frac{\delta \bar{J}}{\delta \beta} = ???$$

As in traditional sensitivity methods we can define the forward tangent equations governing v and the adjoint equations. In this case there will be two different adjoint solutions, ϕ and ψ, governed by the homogeneous (2.4) and inhomogeneous (2.5) adjoint equations respectively.

$$\frac{d\delta u}{dt} - \frac{\partial f}{\partial u}\delta u = \frac{\partial f}{\partial \beta} \tag{2.3}$$

$$\frac{d\phi}{dt} + \frac{\partial f}{\partial u}^T \phi = 0 \tag{2.4}$$

$$\frac{d\psi}{dt} + \frac{\partial f}{\partial u}^T \psi = \frac{\partial J}{\partial u} \tag{2.5}$$

Solutions for $u(t)$, $\delta u(t)$, $\phi(t)$, and $\psi(t)$ are all periodic with the same period T. Each of these variables, when run from a random initial condition, should converge to some periodic limit-cycle. However, the two adjoint solutions are not unique. The homogeneous adjoint ϕ satisfies and additional invariant property.

$$\begin{aligned}\frac{d}{dt}(\phi^T f) &= \phi^T \frac{df}{dt} + \frac{d\phi^T}{dt} f \\ &= \phi^T \frac{\partial f}{\partial u} f + \left(-\phi^T \frac{\partial f}{\partial u} \right) f \\ &= 0\end{aligned}$$

Since $\phi^T f$ is constant along a trajectory, we can additionally scale ϕ so that $\phi^T f = 1$, this will simplify computations. Because the adjoint equations are linear, new inhomogeneous solutions may be created by adding a constant multiple of ϕ to any valid ψ solution.

All possible perturbations in β result in perturbations of f. In a periodic problem there are two fundamental directions that this perturbation (δf) can span, the tangent and stable directions. These special directions will be explained and their use motivated in sections 2.2.1 and 2.2.2 respectively. This will result in a decomposition of the perturbation into

$$\delta f = \delta f_{\text{stable}} + \alpha f \tag{2.6}$$

δf_{stable} perturbations will will cause transient deviations from the original limit-cycle. α is some scalar forcing magnitude, and αf is some forcing that will always be tangent to the original limit-cycle. This decomposition will result in a way to compute $\delta J / \delta \beta$

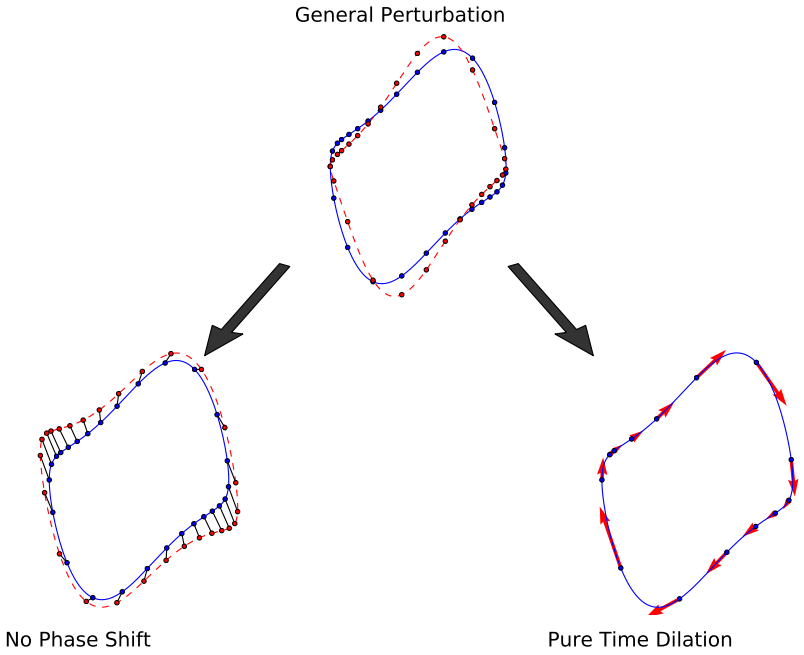

Figure 2-1: Two Possible Perturbations Made to a Periodic Orbit

from stable perturbations and tangent perturbations separately.

$$\frac{\delta \bar{J}}{\delta \beta} = \frac{\delta \bar{J}}{\delta \beta}\bigg|_{\text{stable}} + \frac{\delta \bar{J}}{\delta \beta}\bigg|_{\text{tangent}} \tag{2.7}$$

2.2.1 Tangential Component

Density Definition

To compute the tangent contribution, recognize that perturbations tangent to the 1D attractor do not change the shape of the attractor. If a stationary distribution of points along the attractor is computed, then these tangent perturbations simply shift this distribution around. To compute this density the 1D attractor is parameterized

by its arc length s defined by the 1D ODE below

$$\frac{ds}{dt} = \|f\|_2 \tag{2.8}$$

The one-dimensional stationary distribution, $\rho(s)$, is effectively a probability density. It represents the probability that after a long time, a trajectory started from a random initial condition will be in that region of phase-space. The total arc-length of the limit cycle is defined as $S \equiv s(T)$, and is the total length of the limit-cycle in phase-space. Assuming ergodicity, the density must fulfill the following equation.

$$\bar{b} = \frac{1}{T}\int_0^T b(s(t))\ dt = \int_0^S \rho(s)b(s)\ ds \tag{2.9}$$

In other words, the time average of any quantity, b, along the attractor can be computed by integrating the density times the quantity over the arc-length s. This equivalence allows climate estimates to be computed in the time space, as well as the density space. Using this definition, the density in 1D has a simple form, namely that the density at any point along the attractor is inversely proportional the speed at that point.

$$\rho(s) = \frac{1}{T}\frac{1}{\|f(s)\|_2} \tag{2.10}$$

Density Perturbations

Under tangential forcing ($\delta f = \alpha f$, $\alpha \ll 1$), the density will be perturbed. This section will derive the form that density perturbation will take. For clarity the

following norms will be assumed to be 2-norms ($\|\cdot\| \equiv \|\cdot\|_2$)

$$\rho + \delta\rho = \frac{1}{T + \delta T} \frac{1}{\|f + \delta f\|} \tag{2.11}$$

$$\frac{1}{\rho + \delta\rho} = (T + \delta T)\|f + \delta f\| \tag{2.12}$$

$$\delta\left(\frac{1}{\rho}\right) = (T + \delta T)\|f + \delta f\| - T\|f\| \tag{2.13}$$

Under tangential forcing $\delta f = \alpha f$

$$\delta\left(\frac{1}{\rho}\right) = (T + \delta T)(1 + \alpha)\|f\| - T\|f\| \tag{2.14}$$

$$= \|f\|\left(T + \alpha T + \delta T + \alpha\delta T - T\right) \tag{2.15}$$

$$= \|f\|\left(\alpha T + \delta T + o(\alpha\delta T)\right) \tag{2.16}$$

$$= \|f\|\left(\alpha T + \delta T\right) \tag{2.17}$$

$$= \|f\| T\left(\alpha + \frac{\delta T}{T}\right) \tag{2.18}$$

To compute δT recognize the following:

$$\int_0^S \rho ds = 1 \tag{2.19}$$

$$\int_0^S \frac{1}{T}\frac{1}{\|f\|} ds = 1 \tag{2.20}$$

$$\int_0^S \frac{1}{\|f\|} ds = T \tag{2.21}$$

So that

$$T + \delta T = \int_0^S \frac{1}{\|f + \delta f\|} ds \tag{2.22}$$

$$= \int_0^S \frac{1}{(1+\alpha)\|f\|} ds \tag{2.23}$$

$$\approx \int_0^S \frac{1 - \alpha + o(\alpha^2)}{\|f\|} ds \tag{2.24}$$

$$\delta T = -\int_0^S \frac{\alpha}{\|f\|}\ ds \tag{2.25}$$

Then $\delta\left(\frac{1}{\rho}\right)$ and $\delta\rho$ can be computed.

$$\delta\left(\frac{1}{\rho}\right) = \|f(s)\| T\left(\alpha(s) + \frac{\delta T}{T}\right) \tag{2.26}$$

$$= \|f(s)\| T\left(\alpha(s) - \int_0^S \frac{\alpha(\xi)}{\|f(\xi)\| T} d\xi\right) \tag{2.27}$$

$$= \frac{1}{\rho}\left(\alpha(s) - \int_0^S \rho(\xi)\ \alpha(\xi)\ d\xi\right) \tag{2.28}$$

$$\delta\rho = -\rho^2\ \delta\left(\frac{1}{\rho}\right) \tag{2.29}$$

$$= -\rho^2 \frac{1}{\rho}\left(\alpha(s) - \int_0^S \rho(\xi)\ \alpha(\xi)\ d\xi\right) \tag{2.30}$$

$$= \rho\left(\int_0^S \rho(\xi)\alpha(\xi)\ d\xi - \alpha(s)\right) \tag{2.31}$$

$$= \rho\ (\bar{\alpha} - \alpha) \tag{2.32}$$

Tangential Sensitivity

Using (2.9), the objective function J can be computed. As well as perturbations $\delta\bar{J}$.

$$\bar{J} = \frac{1}{T}\int_0^T J(u(t))\ dt \tag{2.33}$$

$$= \int_0^S \rho(s)\ J(u(s))\ ds \tag{2.34}$$

$$\delta\bar{J} = \int_0^S (\rho\ \delta J + J\ \delta\rho\ )\, ds \tag{2.35}$$

$$= \int_0^S \left(\rho\frac{\partial J}{\partial u}\delta u + J\ \delta\rho\ \right) ds \tag{2.36}$$

Under tangential forcing the shape of the attractor does not change ($\delta u = 0$). Therefore all of the sensitivity comes from the $\delta\rho$ term.

$$\delta\bar{J}_{\text{tangent}} = \int_0^S J(u)\ \delta\rho\ ds \tag{2.37}$$

Additionally because $\delta\rho$ is proportional to ρ, this integral can be converted back into a time integral.

$$\delta J_{\text{tangent}} = \int_0^S J(u)\delta\rho\ ds \tag{2.38}$$

$$= \int_0^S \rho(s)\ (\bar{\alpha} - \alpha)\ J(\ u(s)\ )\ ds \tag{2.39}$$

$$= \frac{1}{T}\int_0^T (\bar{\alpha} - \alpha)\ J(\ u(t)\ )\ dt \tag{2.40}$$

By transforming back into the time domain the problem of having to explicitly compute the arc-length and density can be completely avoided. Equation (2.40) has

an intuitive interpretation. A constant tangential forcing will cause no change in the sensitivity gradient, $(\bar{\alpha} - \alpha(t) = 0)$. An increase in the tangential forcing at a point on the attractor causes a proportional decrease in the local density, and therefore a proportional drop in the objective function.

2.2.2 Stable Component

δf_{stable} is defined as being orthogonal to the unforced adjoint ϕ, $(\phi^T \delta f_{\text{stable}} = 0)$. Because $\phi^T f$ is constant, δf_{stable} will never become collinear with f, and the decomposition will be well defined. Using this definition, computing δf_{stable} from some arbitrary δf becomes:

$$\delta f = \delta f_{\text{stable}} + \alpha f$$

$$\phi^T \delta f = \phi^T \delta f_{\text{stable}} + \alpha \ \phi^T f \tag{2.41}$$

$$= \alpha \tag{2.42}$$

$$\delta f_{\text{stable}} = \delta f - \alpha f \tag{2.43}$$

$\delta \bar{J}_{\text{stable}}$ can then be computed using the regular equation for adjoint sensitivity.

$$\delta \bar{J}_{\text{stable}} = -\frac{1}{T} \int_0^T \psi^T \delta f_{\text{stable}} \, dt \tag{2.44}$$

2.3 Sensitivity

Using the two decomposed sensitivities the total sensitivity to some generic perturbation δf can be computed.

$$\delta\bar{J} = \delta\bar{J}_{\text{tangent}} + \delta\bar{J}_{\text{stable}} \tag{2.45}$$

$$= \frac{1}{T}\int_0^T \left[(\bar{\alpha}-\alpha)J(x) - \psi^T \delta f_{\text{stable}}\right] dt \tag{2.46}$$

$$= \frac{1}{T}\int_0^T \bar{\alpha}\ J(x)dt - \frac{1}{T}\int_0^T \left(\alpha J(x) + \psi^T \delta f_{\text{stable}}\right) dt \tag{2.47}$$

$$= \bar{\alpha}\bar{J} - \frac{1}{T}\int_0^T \left(J(x)\phi^T \delta f + \psi^T\left(\delta f - \alpha f\right)\ \right) dt \tag{2.48}$$

$$= \bar{\alpha}J - \frac{1}{T}\int_0^T \left[J(x)\phi^T \delta f + \psi^T \delta f - \psi^T f \phi^T \delta f\ \right] dt \tag{2.49}$$

$$= \frac{J}{T}\int_0^T \phi^T \delta f\ dt - \frac{1}{T}\int_0^T \left[J(x)\phi^T \delta f + \psi^T \delta f - \left(\psi^T f\right)\left(\phi^T \delta f\right)\ \right] dt \tag{2.50}$$

$$= -\frac{1}{T}\int_0^T \left[-\bar{J}\phi^T + J(x)\phi^T + \psi^T - (\psi^T f)\phi^T\right] \delta f\ dt \tag{2.51}$$

$$= -\frac{1}{T}\int_0^T \left[\left(J(x) - \bar{J} - \psi^T f\right)\phi^T + \psi^T\right] \delta f\ dt \tag{2.52}$$

This takes the form of the normal adjoint sensitivity equation

$$\delta\bar{J} = -\frac{1}{T}\int_0^T \eta^T \delta f\ dt \tag{2.53}$$

With the adjoint variable

$$\eta \equiv \left(J(x) - \bar{J} - \psi^T f\right)\phi + \psi \tag{2.54}$$

To see how η evolves in time first look at the coefficient of ϕ.

$$\frac{d}{dt}(f^T\psi) = f^T\frac{d\psi}{dt} + \frac{df^T}{dt}\psi \tag{2.55}$$

$$= f^T\left(-\frac{\partial f}{\partial x}^T\psi + \frac{\partial J}{\partial x}\right) + \left(\frac{\partial f}{\partial x}f\right)^T\psi \tag{2.56}$$

$$= -f^T\frac{\partial f}{\partial x}^T\psi + f^T\frac{\partial J}{\partial x} + f^T\frac{\partial f}{\partial x}^T\psi \tag{2.57}$$

$$= f^T\frac{\partial J}{\partial x} = \frac{\partial J}{\partial x}^T f \tag{2.58}$$

$$\frac{d}{dt}\Big(\ J(\ x(t)\ )\ \Big) = \frac{\partial J}{\partial x}^T\frac{dx}{dt} \tag{2.59}$$

$$= \frac{\partial J}{\partial x}^T f = f^T\frac{\partial J}{\partial x} \tag{2.60}$$

$$\frac{d}{dt}\left(J - \bar{J} - \psi^T f\right) = \frac{dJ}{dt} - \frac{d\bar{J}}{dt} - \frac{d}{dt}(\psi^T f) \tag{2.61}$$

$$= f^T\frac{\partial J}{\partial x} - 0 - f^T\frac{\partial J}{\partial x} \tag{2.62}$$

$$= 0 \tag{2.63}$$

The coefficient in front of the unforced adjoint ϕ is simply a constant along the attractor.

$$c \equiv \left(J - \bar{J} - \psi^T f\right) \tag{2.64}$$

$$\frac{d\eta}{dt} = \frac{d}{dt}\Big(c\ \phi + \psi \Big) \tag{2.65}$$

$$= c\frac{d\phi}{dt} + \frac{d\psi}{dt} \tag{2.66}$$

$$= c\left(-\frac{\partial f}{\partial x}^T \phi\right) - \frac{\partial f}{\partial x}^T \psi + \frac{\partial J}{\partial x} \tag{2.67}$$

$$= -\frac{\partial f}{\partial x}^T \Big(c\ \phi + \psi \Big) + \frac{\partial J}{\partial x} \tag{2.68}$$

$$= -\frac{\partial f}{\partial x}^T \eta + \frac{\partial J}{\partial x} \tag{2.69}$$

η is also a solution of the same forced adjoint equations and can be computed using any periodic ϕ and ψ solutions started from random initial conditions. η is simply a special solution that will satisfy:

$$f^T\eta = f^T\left(\left(J - \bar{J} - f^T\psi\right)\phi + \psi\right) \tag{2.70}$$

$$= \left(J - \bar{J} - f^T\psi\right) f^T\phi + f^T\psi \tag{2.71}$$

$$= J - \bar{J} - f^T\psi + f^T\psi \tag{2.72}$$

$$= J - \bar{J} \tag{2.73}$$

2.3.1 Period Sensitivity

This formulation also provides a method for finding the sensitivity of the period T using equation (2.25).

$$\delta T = -\int_0^S \frac{\alpha}{\|f\|}\, ds \tag{2.74}$$

$$= -\int_0^S \frac{\phi^T \delta f}{\|f\|}\, ds \tag{2.75}$$

$$= -T\int_0^S \left(\frac{1}{T}\frac{1}{\|f\|}\right) \phi^T \delta f\ ds \tag{2.76}$$

$$= -T\int_0^S \rho\ \phi^T \delta f\ ds \tag{2.77}$$

$$= -\int_0^T \phi^T \delta f\ dt \tag{2.78}$$

Taking this into the more familiar adjoint form, it can be shown that the specific homogeneous adjoint chosen has a physical interpretation. As seen in equation (2.80), the homogeneous adjoint predicts the sensitivity to the log-period ($\log T$) of the system.

$$\delta \log T = \frac{\delta T}{T} \tag{2.79}$$

$$= -\frac{1}{T}\int_0^T \phi^T \delta f\ dt \tag{2.80}$$

2.4 Algorithm

This method forces the arbitrary adjoint ψ towards the correct adjoint trajectory η. After the adjoints have reached a periodic solution, steps 22-24 should be only very small corrections. The quadrature for computing $\bar{J}$ and $\delta\bar{J}$ should be included in

1: # Solve Primal Problem
2: $x_0 =$ random vector
3: **for** $i = 1 \to N$ **do**
4: $\quad x_i = \text{ode\_step}(f,\ x_{i-1},\ \Delta t)$ # some ode stepping (e.g. crank-nicolson)
5: **end for**
6: # Compute n and k where:
7: # x takes roughly n steps to reach a periodic trajectory with a period of k
8: # then choose quadrature weights, w_i, such that
9: $w_i = 0$ if $i \leq n$ or $i > n + k$
10: $w_i =$ some chosen weights for the k steps from $i = \{n+1, \ldots, n+k\}$. (e.g. trapezoidal)
11:
12: $\bar{J} = \sum_{i=0}^{N} w_i\ J(x_i) = \sum_{i=n+1}^{n+k} w_i\ J(x_i)$ # quadrature to find $\bar{J}$
13:
14: # Solve Adjoint Problem
15: $\phi_N =$ random vector
16: $\psi_N =$ random vector
17: **for** $i = N - 1 \to n + 1$ **do**
18: $\quad \phi_i =$backward_step$\big(f,\ \frac{\partial f}{\partial x},\ x_i,\ x_{i+1},\ \phi_{i+1},\ \Delta t,\ \frac{\partial J}{\partial x} = 0\big)$
19: $\quad \psi_i =$backward_step$\big(f,\ \frac{\partial f}{\partial x},\ x_i,\ x_{i+1},\ \psi_{i+1},\ \Delta t,\ \frac{\partial J}{\partial x} = \frac{\partial J}{\partial x}\big)$
20: $\quad$ # some backward stepping using same method as ode_step
21:
22: $\quad \phi_i = \frac{\phi_i}{\phi_i^T f(x_i)}$ # Normalize ϕ
23: $\quad c_i = J(x_i) - \bar{J} - \psi_i^T f(x_i)$ # Compute coefficient, should approach zero
24: $\quad \psi_i = \psi_i + c_i \phi_i$ # Correct ψ onto correct trajectory
25: **end for**
26:
27: # Compute Sensitivity
28: $\delta J = -\sum_{i=0}^{N} w_i\ \psi_i^T \delta f(x_i) = -\sum_{i=n+1}^{n+k} w_i\ \psi_i^T \delta f(x_i)$ # another quadrature to find δJ
29:
30:

the two loops to avoid storing all of the trajectories. This method provides several advantages to the windowing adjoint method [13]. Firstly this estimate only requires one full period of the primal and adjoint dynamics to compute sensitivities. After spin-up the windowing method requires a weighted integration over several period lengths. The split-perturbation technique, however, does require two simultaneous adjoint simulations, nearly doubling the computational time. This will still beat the windowing methods if the window length is more than two periods long. Also, this additional adjoint simulation also provides important sensitivity information about the period of the system without additional cost. Finally, the resulting adjoint solutions that are computed provide a time-accurate view of the sensitivity information. This extra information could be vital, for example, in automatic control situations, where time dependent sensitivities could be used for time dependent control inputs.

2.5 Van der Pol Oscillator

The Van der Pol Oscillator is a second order ODE and one of the simplest systems exhibiting periodic dynamics. This section will examine the application of the periodic adjoint algorithm described in section 2.4. The dynamics of this system are described in equation (2.81) below.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ \beta(1-x^2)y - x \end{bmatrix} \tag{2.81}$$

Solutions to the differential equation approach a single periodic limit-cycle whose shape is controlled by the parameter β, as is shown in Figure 2-2. The traditional adjoint equations, do not have one unique solution. An example of this phenomenon

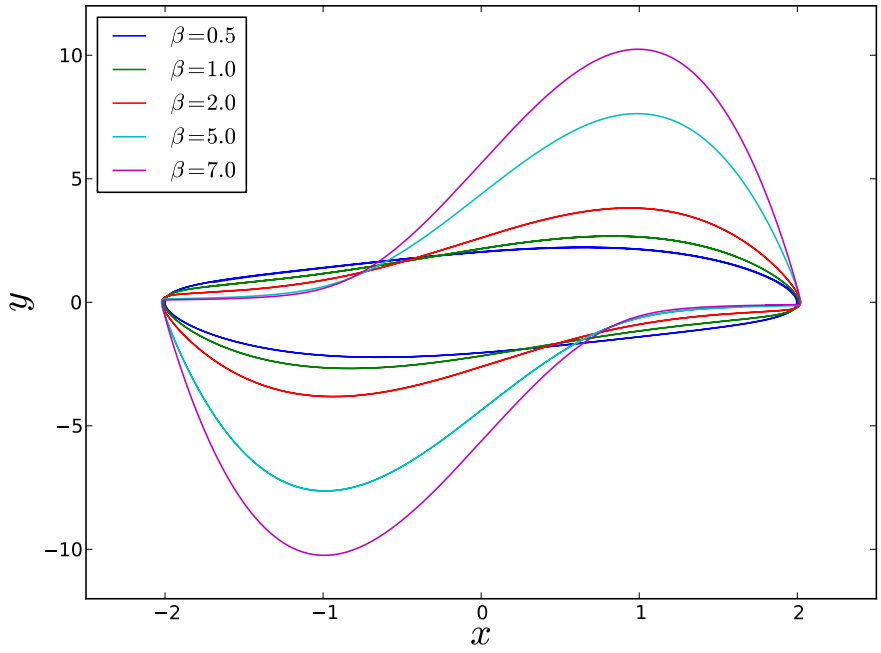

Figure 2-2: Several primal solutions to the Van der Pol equations

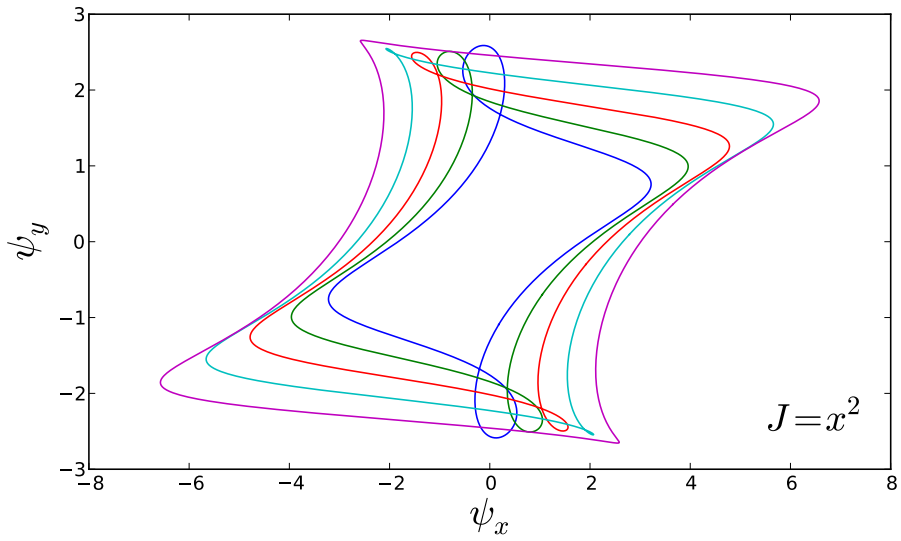

Figure 2-3: Multiple solutions to the adjoint Van der Pol equations, $\beta = 1.0$, $J([x,\ y]) = x^2$

is shown in figure 2-3. Despite the multiple adjoint solutions there is one "true" adjoint that will predict the correct sensitivities via equation (2.69). The objective function examined here is the averaged squared x position along the attractor. As can be seen in 2-2, a longer and longer proportion of the limit-cycle is further away from the origin as β increases, figure 2-4 confirms this trend.

The algorithm from section 2.4 was applied to the Van der Pol system. The adjoint sensitivity estimates are computed for a range of β values. These estimates are compared to a finite difference estimate of the sensitivity. Figure 2-5 shows these sensitivity estimates. Note, that the finite-difference estimates seem to provide worse estimates for a given choice of Δt, and as Δt decreases the finite-difference estimates approach the adjoint estimate. Additionally figure 2-6 shows information pertaining the sensitivity of the log-period of the Van der Pol Oscillator. These estimates agree very well with finite-difference estimates.

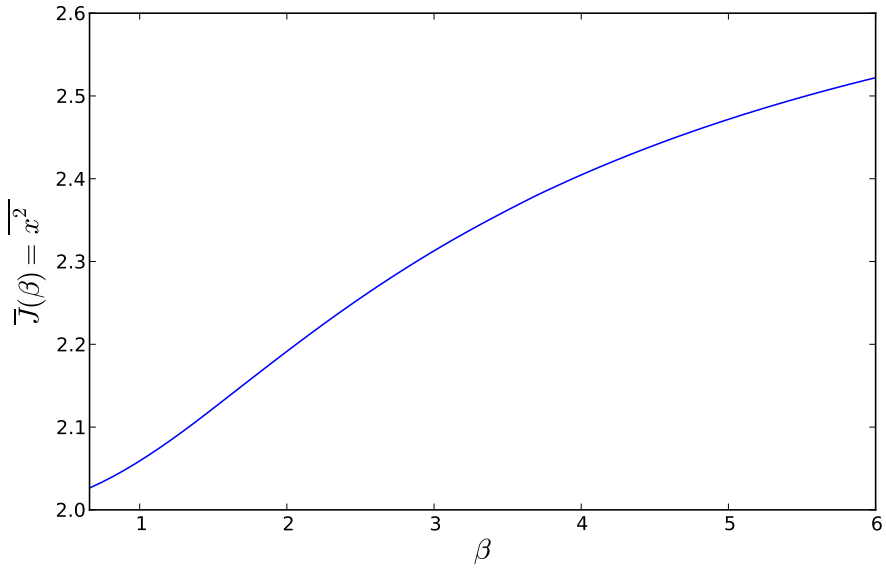

Figure 2-4: Objective Function (x^2) variation versus β

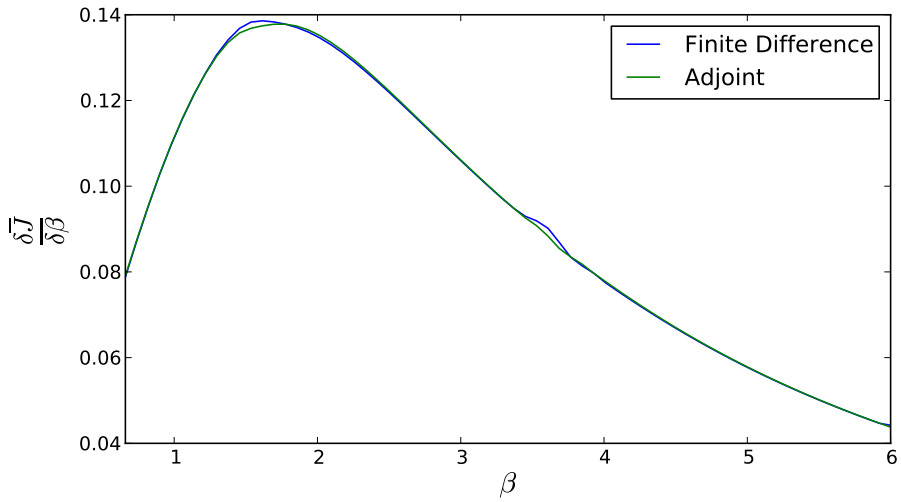

Figure 2-5: Predicted sensitivity comparison between split periodic adjoint formulation and finite difference ($\Delta t = 10^{-3}$, $\Delta\beta \approx 0.16$)

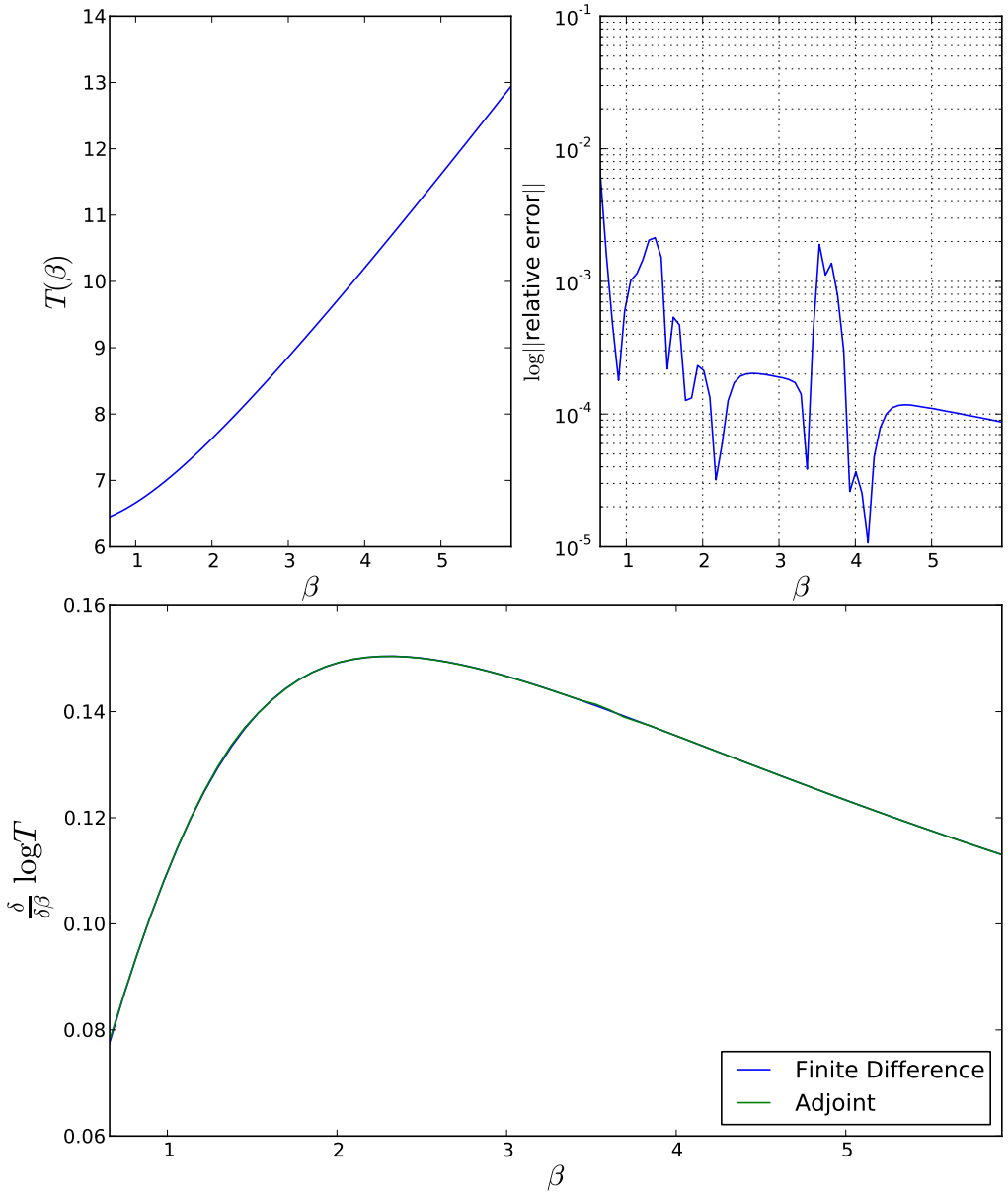

Figure 2-6: Predicted period variation (top left), Comparison between predicted sensitivities by FD vs Adjoint (bottom center), and relative error in period sensitivity predictions ($\Delta t = 10^{-3}$, $\Delta\beta \approx 0.16$) (top right)

Chapter 3

Least Squares Sensitivity

3.1 Introduction

Least Squares Sensitivity (LSS) is a new technique that can be used to find the sensitivity to long-time average quantities for ergodic dynamical systems. Traditional sensitivity methods derive a initial-value or final-value problem which can then be solved to find sensitivity gradients. However these methods breakdown for the chaotic systems that often occur in engineering settings. The linearized forward-tangent and adjoint equations are unstable and solutions grow unbounded as solutions are solved forward or backward in time. In Least Squares Sensitivity analysis, the problem is instead cast as an optimization problem for finding the smallest perturbation across all time, thus relaxing the initial or terminal condition. The result is a boundary-value problem in time requiring solution for the linearized perturbations across all time simultaneously.

3.2 Formulation of Continuous LSS Equations

To formulate the LSS method we, again, start with a non-linear differential equation for $u(t)$, that represents either an ODE or a spatially discretized PDE of size M ($u \in \mathbb{R}^M$). We also must obtain some solution to those primal equations $u(t)$.

$$\frac{du}{dt} = f(u) \tag{3.1}$$

We still want to find a perturbation $v(t)$ that satisfies the linearized equations given some small input perturbation δf

$$\frac{dv}{dt} = \frac{\partial f}{\partial u} v + \delta f \tag{3.2}$$

However instead of a traditional initial value constraint (i.e. $v(t = 0) = 0$). We instead require that this perturbation remains small **in phase space** across time. For computational reasons we also introduce a local time-dilation parameter $\eta(t)$ that represents the amount of perturbation **in time** from the primal trajectory $u(t)$ to its shadow trajectory $(u + v)$. α is a constant describing the relative weighting between minimizing the time-dialation and tangent solution magnitude.
Together these conditions result in the minimization condition

$$v, \eta = \text{argmin } \frac{1}{2} \int_0^T \|v\|_2^2 + \alpha^2 \eta^2 \; dt \tag{3.3}$$

$$s.t. \quad \frac{dv}{dt} = \frac{\partial f}{\partial u} v + \eta f \tag{3.4}$$

To solve this problem we introduce the Lagrange multiplier $w(t)$ and the Lagrange function Λ

$$\Lambda = \frac{1}{2}\int_0^T v^T v + \alpha^2\eta^2 \; dt + \int_0^T w^T \left(\frac{dv}{dt} - \frac{\partial f}{\partial u}v - \eta f\right) \; dt \tag{3.5}$$

$$\begin{aligned}\delta\Lambda &= \int_0^T v^T\delta v + \alpha^2\eta\delta\eta + w^T\frac{d\delta v}{dt} - w^T\frac{\partial f}{\partial u}\delta v - f\delta\eta \; dt \\ &= \int_0^T \delta v^T\left(v - \frac{dw}{dt} + \frac{\partial f}{\partial u}^T w\right) + \delta\eta\left(\alpha^2\eta - w^T f\right) \; dt \quad + \quad w^T\delta v\Big|_0^T\end{aligned} \tag{3.6}$$

Enforcing the optimality condition that $\delta\Lambda = 0$ for any possible variation $(\delta v,\, \delta\eta)$ the following equations fall out.

$$v = \frac{dw}{dt} + \frac{\partial f}{\partial u}^T w \tag{3.7}$$

$$\eta = \frac{1}{\alpha^2}f^T w \tag{3.8}$$

$$w(0) = w(T) = 0 \tag{3.9}$$

Finally, these equations can be manipulated into one second-order linear PDE for w.

$$-\frac{d^2w}{dt^2} - \left(\frac{d}{dt}\frac{\partial f}{\partial u}^T - \frac{\partial f}{\partial u}\frac{d}{dt}\right)w + \left(\frac{\partial f}{\partial u}\frac{\partial f}{\partial u}^T + \mathcal{P}\right)w = \delta f \tag{3.10}$$

$$\text{where} \quad \mathcal{P}w = \frac{1}{\alpha^2}f\left(f^T w\right)$$

$$w(0) = w(T) = 0$$

The continuous linear operators $\mathcal{B}$ and $\mathcal{E}$ can be defined across time

$$(\mathcal{B}w)(t) \equiv \left(\frac{d}{dt} - \frac{\partial f}{\partial u}\Big|_{u(t)} \right) w(t) \tag{3.11}$$

$$(\mathcal{E}w)(t) \equiv \frac{1}{\alpha} f(u(t)) \left(f(u(t))^T w(t) \right) \tag{3.12}$$

Then Equation (3.10) can be re–written as

$$\left(\mathcal{B}\mathcal{B}^T + \mathcal{E}\mathcal{E}^T \right) w = \delta f \tag{3.13}$$

In this form it is clear that solving for $w(t)$ involves inverting a symmetric positive definite (SPD) linear operator. Finding $w(t)$ directly gives the tangent solution $v(t)$ and time-dilatation $\eta(t)$ via equations (3.7) and (3.8) respectively. Once v and η have been found the climate sensitivity can be computed.

$$\bar{J} = \frac{1}{T} \int_0^T J(u(t))\ dt \tag{3.14}$$

$$\delta \bar{J} = \frac{1}{T} \int_0^T \frac{\partial J}{\partial u} v + (\eta - \bar{\eta}) J\ dt \tag{3.15}$$

To actually compute these sensitivities this linear PDE must be discretized and solved.

3.2.1 Discretization of LSS Equations

In order to discretize this problem first the primal solution must first be discretized. Any traditional ODE forward time solving method may be used. This thesis will

assume an implicit trapezoidal method, as a relatively stable and accurate scheme.

$$\frac{u_{n+1} - u_n}{\Delta t} = \frac{f(u_{n+1}) + f(u_n)}{2} \tag{3.16}$$

Because of ergodicity, starting from any initial condition ($u(0) = u_0$) will eventually produce the same long-time average quantities. Since a finite time interval (T) is required an initial condition will be chosen to lie along some attractor. The primal equations can then be solved forward in time to some sufficiently "large" time. From now on u will refer to the vector of all time steps from $0 \to N$. ($u_n \equiv u(n\Delta t)$)

$$u \equiv [u_0,\ u_1,\ u_2,\ \ldots\ u_{N-1},\ u_N]^T, \quad u \in \mathbb{R}^{(N+1)M} \tag{3.17}$$

The linear operators $\mathcal{B}$ and $\mathcal{E}$ are then discretized at the N midpoints between each time-step. Let's define

$$b_n \equiv \frac{1}{2}\left(\left.\frac{\partial f}{\partial \beta}\right|_{u_n} + \left.\frac{\partial f}{\partial \beta}\right|_{u_{n+1}}\right) \qquad b \in \mathbb{R}^{NM} \tag{3.18}$$

$$f_n \equiv \frac{1}{2}\Big(f(u_n) + f(u_{n+1})\Big) \qquad f \in \mathbb{R}^{NM} \tag{3.19}$$

$$A_n \equiv \left.\frac{\partial f}{\partial u}\right|_{\frac{u_n + u_{n+1}}{2}} \qquad A \in \mathbb{R}^{N \times M \times M} \tag{3.20}$$

Then the discretized operators become

$$(Bv)_n = \frac{v_n - v_{n+1}}{\Delta t} - A_n \frac{v_n + v_{n+1}}{2} \tag{3.21}$$

$$(E\eta)_n = \frac{1}{\alpha} f_n \eta_n \tag{3.22}$$

$$B \in \mathbb{R}^{NM \times (N+1)M}, \quad E \in \mathbb{R}^{NM \times N}$$

B is a bidiagonal block matrix with $M \times M$ blocks. With $F_n = \frac{I}{\Delta t} - \frac{A_n}{2}$, and $G_n = -\frac{I}{\Delta t} - \frac{A_n}{2}$. This corresponds to a trapezoidal discretization of $\mathcal{B}$.

$$B = \begin{bmatrix} F_0 & G_0 & & & \\ & F_1 & G_1 & & \\ & & & \ddots & \\ & & & F_{N-1} & G_{N-1} \end{bmatrix}$$

$$E = \frac{1}{\alpha} \begin{bmatrix} f_0 & & & \\ & f_1 & & \\ & & \ddots & \\ & & & f_{N-1} \end{bmatrix}$$

Using this form BB^T becomes a block tridiagonal matrix and EE^T becomes a block diagonal matrix with rank 1 blocks along the diagonal. The final matrix ($S = BB^T + EE^T$) retains the SPD form of the continuous linear operator, and remains block tri-diagonal with size $NM \times NM$. This tridiagonal property makes this problem amenable to parallelization, as will be shown in section 4.2.

The final discretized linear equation keeps a form identical to the continuous case as shown below in equation (3.23)

$$Sw = (BB^T + EE^T)w = b \tag{3.23}$$

$$v = -B^T w \tag{3.24}$$

$$\eta = \frac{1}{\alpha} E^T w \tag{3.25}$$

Chapter 4

Solution of the Discretized Least Squares Sensitivity System

4.1 Geometric Multigrid on the LSS System

In order to perform multigrid on the discrete LSS system we must be able to construct the system at different coarsening levels and be able to restrict and interpolate solutions between levels. For this problem we use a simple V-cycle, the pseudocode for the algorithm is below.

To fully implement this algorithm we must specify the four functions used above; SMOOTHING, COARSEOPERATOR, RESTRICT, and INTERPOLATE. As well as the recursion conditions and the number of pre and post iterations.

4.1.1 V-Cycle Smoothing Operations

Traditionally in multigrid methods, smoothing iterations are performed using some fixed-point iteration like Jacobi, Gauss-Seidel, SOR, SSOR, etc. The symmetric

```
1: function VCYCLE( S, b )
2:     w_0 ← 0
3:     w_1 ← SMOOTHING( S, b, w_0, n_pre )
4:     if (Can Coarsen) then
5:         r ← b − S w_1
6:         r_c ← RESTRICT( r )
7:         S_c ← COARSEOPERATOR( S )
8:         e_c ← VCYCLE( S_c, r_c)
9:         e ← INTERPOLATE( e_c )
10:        w_2 ← w_1 + e
11:    else
12:        w_2 ← w_1
13:    end if
14:    w_3 ← SMOOTHING( S, b, w_2, n_post )
15:    return w_3
16: end function
```

tridiagonal block structure of our operator makes something make the block version of these iterations (i.e. block Jacobi, block Gauss-Seidel) possible. For large-scale systems with large M it may be infeasible to actually form the block matrices required to construct a numerical S. Specifically for large PDE systems constructing the Jacobian ($\frac{\partial f}{\partial u}$) itself may be difficult. Instead a matrix-free representation for multiplying $\frac{\partial f}{\partial u}$ by some vector is used. These fixed iterators require solution of the $M \times M$ blocks within S repeatedly which can be slow for large M. Additionally, of these methods, only Jacobi iterations can be easily parallelized. A parallelizable Gauss-Siedel/Jacobi hybrid method was tested for this problem. However, because of the SPD features of our matrix more specialized krylov solvers such as Conjugate Gradient and MINRES may be utilized.[3, 6] For the implementation described here standard MINRES iterations were used in all smoothing operations. Using conjugate gradient gave similar convergence characteristics, but MINRES ensured a smoother, monotonic decrease in the solution residual.

Number of Smoothing Iterations

The correct number of pre-smoothing and post-smoothing iterations must be selected to run these smoothing iterations. There unfortunately not much theory on selecting these parameters optimally. In practice, for LSS problems, a few heuristics for selecting n_{pre} and n_{post} have been identified:

- Convergence is largely insensitive to the number of pre-iterations (n_{pre}).
- When the number of time-steps is large, a larger number of post-iterations (n_{post}) improves convergence and is often necessary for convergence at all.
- Post-smoothing rapidly reduces the residual at first, but the rate of reduction slows after a few iterations.
- Too much post-smoothing at one level is unproductive and at some point it is better to run a new cycle rather than to continue.

With these rules in mind the number of pre-iterations is chosen to be some small constant number, (e.g. $n_{\text{pre}} = 5$). The number of post-iterations was chosen dynamically at runtime. The residual was tracked over many cycles, once the convergence of the residual is below some specified rate the post-smoothing iterator stops. If this minimum rate is chosen correctly this greatly decreases the time it takes for a solution to converge.

4.1.2 Restriction and Interpolation of Solution Residual

For a generic ODE problem all restriction and interpolation are done **in time**. If the primal problem is a PDE then restriction and interpolation may also be performed

in space. There are potentially two restriction operators R_t and R_x

$$R_t : \mathbb{R}^{NM} \to \mathbb{R}^{N_c M} \tag{4.1}$$

$$R_x : \mathbb{R}^{NM} \to \mathbb{R}^{N M_c} \tag{4.2}$$

Typically in multigrid $N_c = N/2$ (i.e. restrict to a time-grid of half the size). The same convention will be used here. The scheme for spatial coarsening, however, is completely dependent on the the differential equations used. This issue will be left for section 4.4 when the specific application to a PDE problem is discussed.

When performing time coarsening, a wide windowed average across time-steps i used to coarsen from a time grid of size Δt to $2\Delta t$. Figure 4-1 shows one way in which the solution on the fine time-grid can restricted to the coarse time-grid.

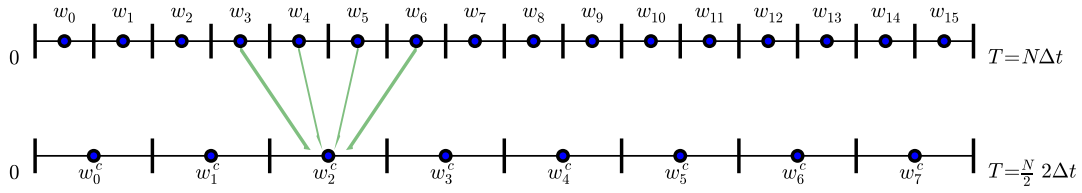

Figure 4-1: Diagram of Restriction in Time

The restriction operator in this implementation uses a 4 neighbor weighted average on the fine grid to compute the values on the coarse grid. At the internal points

the operator is given by

$$R_t = \frac{1}{8}\begin{bmatrix} \ddots & & & & & & & \\ & 1 & 3 & 3 & 1 & & & \\ & & & 1 & 3 & 3 & 1 & \\ & & & & & & & \ddots \end{bmatrix}$$

At the time boundaries $t = 0$, and T the boundary condition that $w(0) = w(T) = 0$ must be enforced. The restriction operator must respect this condition when solutions are moved from the fine to coarse grid. In the discrete setting ghost points are chosen so that $w_{-\frac{1}{2}} = w_{\left(N-\frac{1}{2}\right)} = 0$.

Using this approach, the equations for the first and last coarsened time-steps are as follows.

$$w_0^c = \frac{2}{17}\left(\frac{3}{2}w_{-\frac{1}{2}} + 3w_0 + 3w_1 + w_2\right)$$
$$w_{N/2-1}^c = \frac{2}{17}\left(w_{N-3} + 3w_{N-2} + 3w_{N-1} + \frac{3}{2}w_{N-1+\frac{1}{2}}\right)$$

The final time restriction operator R_t is shown in equation (4.3)

$$R_t = \begin{bmatrix} \frac{6}{17} & \frac{6}{17} & \frac{2}{17} & & & & & & & \\ & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & & & & \\ & & & & \ddots & & & & & \\ & & & & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & \\ & & & & & & & \frac{2}{17} & \frac{6}{17} & \frac{6}{17} \end{bmatrix} \tag{4.3}$$

The interpolation operator, is then simply a constant multiple of the adjoint of the Restriction operator, where the constant is determined so that each of the non-

boundary rows will sum to 1. For this restriction scheme that implies that $c = 2$.

$$I_t = c\ R_t^T \tag{4.4}$$

4.1.3 LSS Operator Coarsening

The variational form of the coarse grid operator (S_c) is derived from applying the restriction and interpolation operators to the fine-grid matrix ($S_c = RSI$). However, for large PDE problems, S may not be explicitly computed and therefore neither will S_c. One possible method would be to use a functional form of S_c, in which the operation of the coarse grid operator is computed by the composition of interpolation, application of the fine grid operator, and then restriction. This however would mean each operator application on the coarse grid would be even more expensive than on the fine grid. Because we are using a krylov method as a smoother, almost all the cost of the algorithm is from repeatedly applying the operator to some vector w. This method of forming the coarse grid operator would not take advantage of the fact that solving on the coarser grids should be faster.

To alleviate this problem notice that the fine grid operator is parameterized by some primal trajectory $u \in \mathbb{R}^{(N+1)M}$. This primal trajectory, along with the time-dilation weighting α fully defines the operator. Instead of applying restriction the the whole operator, restriction is applied to the primal trajectory in time and/or space. That coarsened primal solution is then used to define the coarse grid operator.

The obvious way to coarsen u from $N + 1$ time-steps to $N/2 + 1$ time-steps is to

take every other point in time as shown in (4.5).

$$\begin{gathered} u = [u_0,\ u_1,\ u_2,\ \dots\ u_{N-1},\ u_N]^T \\ \downarrow \\ u^c = [u_0,\ u_2,\ u_4,\ \dots\ u_{N-2},\ u_N]^T \end{gathered} \tag{4.5}$$

However, as u is recursively coarsened in time, the variation from time-step to time-step grows. Figure 4-2 shows what can happen when the primal is naively coarsened in this fashion. This "jagged" primal will lead to a non-smooth variation in the block matrices along the diagonals of the linear operator. With a non-smooth operator high frequency errors cannot be removed from the solution. The correct coarse grid solution will retain high frequency modes. Instead, when the primal is coarsened it is also smoothed in time. Similar to how the solution residuals are coarsened, a linear operator ($R_{t,p}$) can be constructed to restrict the primal solution. This is shown in equation (4.6). This method keeps the end-points of the primal trajectory fixed but smooths the interior points using a weighted average of nearby time-steps. Figure 4-2 shows this compared to the naive method on a primal solution from the Lorenz System. This smoothened primal results in smoother variation across the diagonal of the coarse linear operator. This effectively mimics the important property of the variational coarse operator, namely that solutions on the coarse grid will have smaller

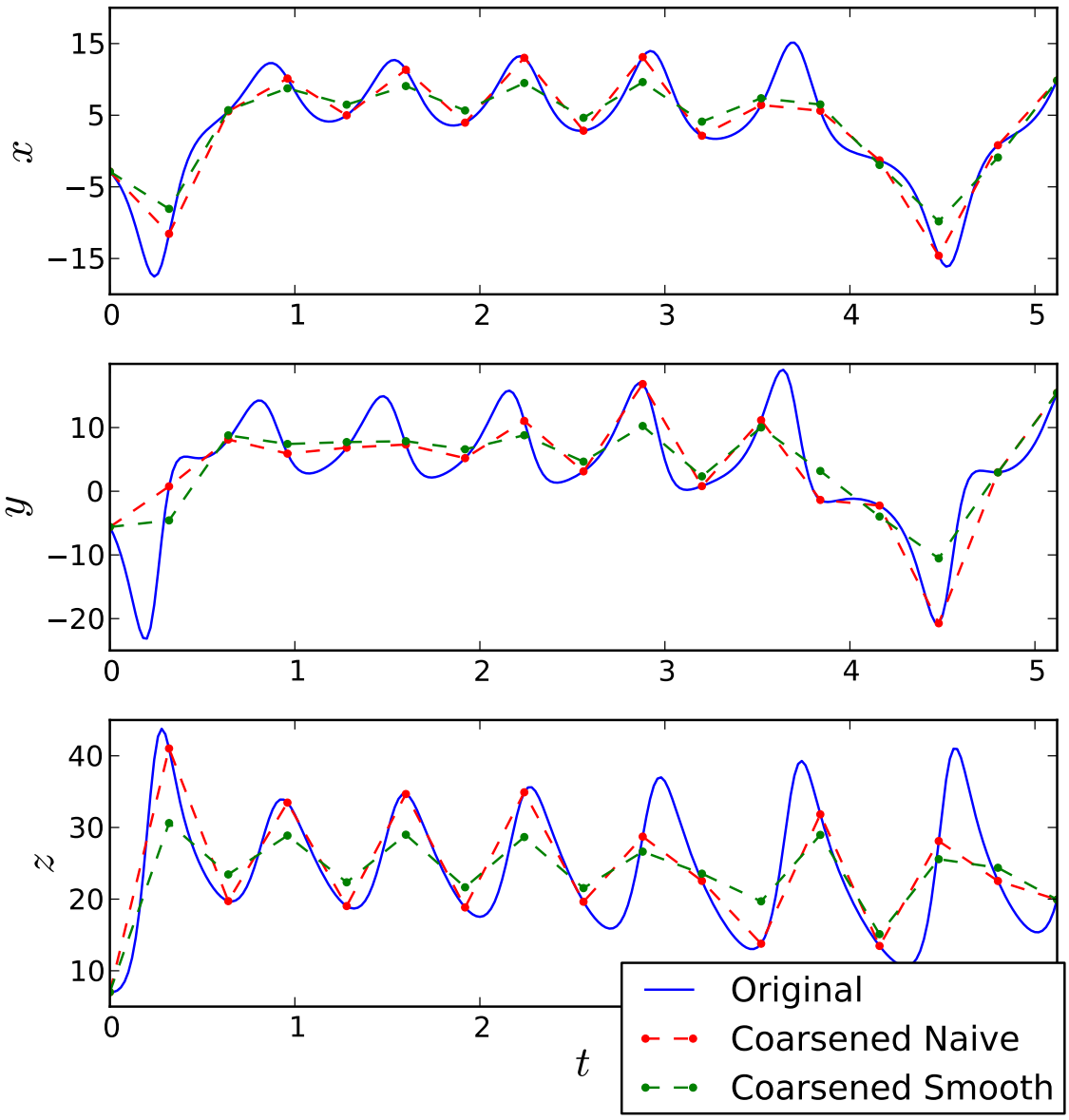

Figure 4-2: Results of coarsening primal four times using naive method (red), and smoothed method (green) (Lorenz Equations)

high-frequency components.

$$R_{t,p} = \begin{bmatrix} 1 & & & & & & & & & \\ & 1 & 3 & 1 & & & & & & \\ & & & 1 & 3 & 1 & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & 3 & 1 & \\ & & & & & & & & & 1 \end{bmatrix} \tag{4.6}$$

For PDE problems the primal trajectory must also be coarsened in space. However, the same spatial restriction operator (R_x) used to spatially coarsen the solution residual may be reused.

4.1.4 Coarsening Order (For PDE's)

At any given coarsening step there are three options for coarsening. The residual maybe be coarsened in space, time, or both. This choice on the order of coarsening has a profound impact on the convergence of the multigrid scheme. To examine the optimal coarsening strategy a simple sample PDE (Viscous Burger's Equation (4.7)) will be used. Viscous Burger's models a simple 1D fluid flow problem, however the equations themselves do no result in chaotic solutions. Instead, the linear LSS system that results from viscous burgers will be used. However, the will be coupled with a primal trajectory from a chaotic system. The primal solution used comes from a 1D slice of a homogeneous isotropic turbulence simulation. The solution of the LSS system for this contrived problem is meaningless, it predicts no relevant sensitivities.

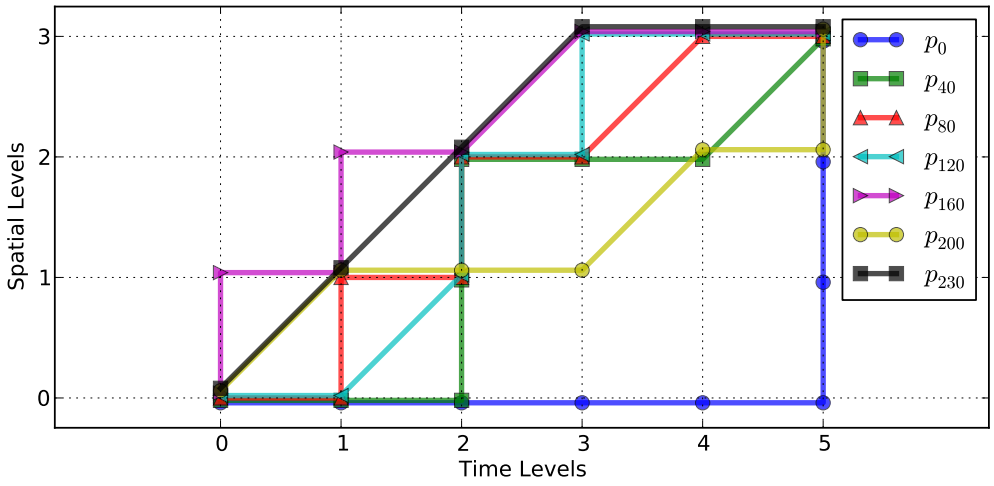

Figure 4-3: Example of space-time coarsening paths (5 Time-Levels $\times$ 3 Spatial-Levels)

It only provides a simple test-case for examining different multigrid parameters.

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \tag{4.7}$$

The problem of choosing the best coarsening order is one of choosing the optimal path from the finest space-time grid to the coarsest space-time grid. For some chosen problem the solution may be coarsened in time L_t times, and coarsened in space L_x times. At any instance in the cycle the current level is defined as $l = (l_t, l_x)$. The sequence of forward steps (p) from $l = (0,\ 0)$ to (L_t, L_x) will define the order in which the coarsening occurs. In this problem a forward step may be one of $(1, 0)$, $(0, 1)$, or $(1, 1)$ which corresponds to time-coarsening, spatial-coarsening, and both respectively. The full V-cycle is then defined by this choice of steps. Figure 4-3 shows several examples of how this path may be selected for $L_t = 5, L_x = 3$. Table 4.1 shows the steps that make up those paths.

path	steps
p_0	$[(1,0),(1,0),(1,0),(1,0),(1,0),(0,1),(0,1),(0,1)]$
p_{40}	$[(1,0),(1,0),(0,1),(0,1),(1,0),(1,0),(1,1)]$
p_{80}	$[(1,0),(0,1),(1,0),(0,1),(1,0),(1,1),(1,0)]$
p_{120}	$[(1,0),(1,1),(0,1),(1,0),(0,1),(1,0),(1,0)]$
p_{160}	$[(0,1),(1,0),(0,1),(1,0),(1,1),(1,0),(1,0)]$
p_{200}	$[(1,1),(1,0),(1,0),(1,1),(1,0),(0,1)]$
p_{230}	$[(1,1),(1,1),(1,1),(1,0),(1,0)]$

Table 4.1: Example Paths taken in Figure 4-3

In order to determine the best path a brute force algorithm is used to solve the burger's LSS system for every possible path from $(0,0)$ to $(5,3)$. For this (L_t, L_x) pair there are 231 unique, single-step, forward paths to try. Figure 4-4 shows the residual of the linear system vs time for all 231 unique paths. The convergence rate varies wildly depending on the coarsening path chosen. For this system, however, the best path is clearly p_0. The V-Cycle defined by p_0 is one which fully coarsens in time before coarsening in space. Figure 4-5 shows the 5 paths which provide the fastest rate of convergence. Each of the 5 best paths includes coarsening in time several times before coarsening in space. The ability to further coarsen in time will prove to be very important for the convergence of a multigrid LSS solver.

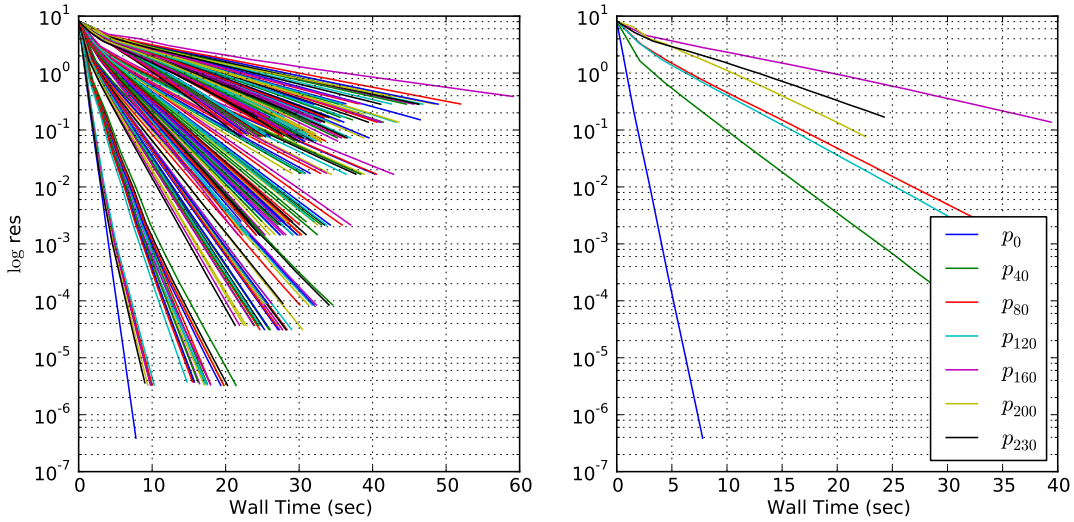

Figure 4-4: Residual v.s. Time For all possible paths, ($N = 2048, M = 48$)

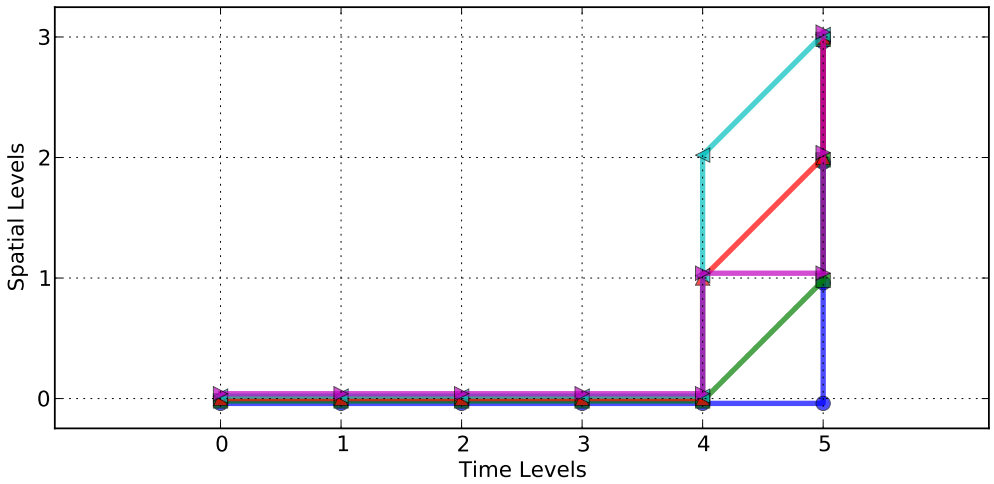

Figure 4-5: Five Best Coarsening Paths for Burger's LSS

4.2 Parallel-In-Time Multigrid

This section will address the techniques used to parallelize the multigrid method described in section 4.1. Specifically, the method for parallelizing this problem along the time axis will be examined. Parallelization along the space axis will be necessary for large classes of PDE's, however spatial parallelization has been examined in detail elsewhere and maybe added in to this time parallel approach without much difficulty.

There are many features of the discrete LSS system derived in section 3.2.1, that can be exploited to more easily parallelize operations on the linear system. Specifically, the block tridiagonal structure greatly reduces the communication complexity. For example, when multiplying by the LSS operator, the result at time-step n only depends on data from time-step $n-1$ and $n+1$.

To implement the multigrid solver, there are only a few operations that must become parallel. The krylov smoother chosen here, MINRES, requires only the ability to apply some linear operator, to take inner products, and to perform simple arithmetic on vectors, and scalars [18]. The rest of the multigrid algorithm only requires application of the restriction/interpolation linear operators, and application of the primal restriction operator. Once all these operations are parallelized the rest of the multigrid algorithm can run unchanged from the sequential version.

4.2.1 Parallel Distribution of Data

The Lagrange multiplier solution vector, w, for N time-steps, is a size NM vector. The data for this vector will be split across n_p processes. Assuming that N is a multiple of n_p, each process will be responsible for $N_c = \frac{N}{n_p}$ time-steps. The implementation described here will perform all coarse cycles on the same n_p processes, this puts a limit on the relationship between the number of time-steps (N), processes

(n_p), and time-levels (L_t) in the V-cycle.

$$N = c\ n_p\ 2^{L_t}, \quad c \in \mathbb{Z}^+ \tag{4.8}$$

4.2.2 Parallel Application of the LSS Operator

Application of the LSS linear operator S is the dominant cost of the multigrid cycle. The tridiagonal blocks of the S matrix may not be pre-computed in some PDE cases. In those cases it is more simple to use the decomposition of S from equation (3.23). Making the individual matrices B, B^T, E, and E^T parallel turns out to be an easier problem. Multiplying by E, and E^T is trivial to parallelize in this setting. Each of these operators are diagonal, as is their product.

$$(EE^T y)_i = f_i\ (f_i^T\ y_i) \tag{4.9}$$

The more interesting problem is to parallelize applications of B and B^T. If the original primal trajectory of length $N+1$ is split up into N_c+1 overlapping sub-chunks as shown in figure 4-6. By forming the B matrix over each sub trajectory, there are n_p sub-matrices (B_n). Each sub-matrix transforms an $(N_c{+}1)M$ vector into n N_cM. The sub-matrices can be combined to form the full B matrix as demonstrated in equation (4.10).

$$B = \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_{n_p-1} \end{bmatrix} \tag{4.10}$$

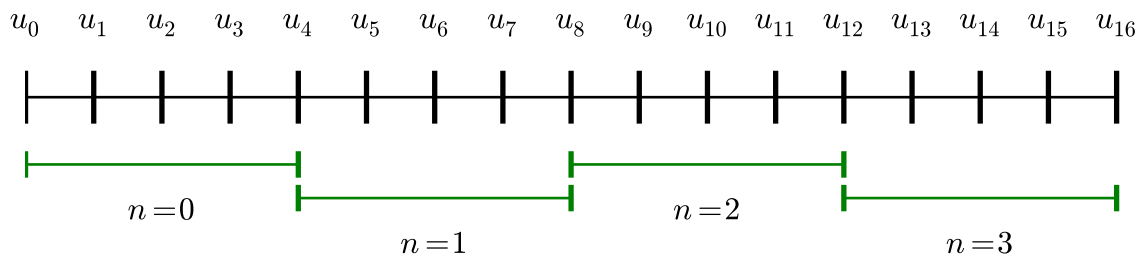

Figure 4-6: Spiting the primal trajectory into chunks ($N = 16,\ n_p = 4$)

Because the rows of the local operators do not overlap, multiplication by B in parallel simply involves each process multiplying its local $(N_c + 1)M$ vector by its local sub-matrix. In this step no communication is required.

Applying B^T is the final step towards applying the LSS operator in parallel. More care is required to multiply by B^T. The transpose of the same B_n sub-matrices make up B^T as shown in equation (4.11). If each sub-matrix is multiplied locally the result will be mostly identical to multiplication by the global B^T. However, because the rows between matrices overlap the local result is not the whole answer. At each of the overlapping time-steps seen in figure 4-6, the results will be incorrect, and will not match each other. At each of these $n_p - 1$ overlapping points the local solution data is summed between neighboring processes. Once this summation is done, each

piece of local data will be correct.

$$B^T = \begin{bmatrix} B_0^T & & & \\ & B_1^T & & \\ & & \ddots & \\ & & & B_{n_p-1}^T \end{bmatrix} \tag{4.11}$$

4.2.3 Parallel Restriction and Interpolation

During restriction and interpolation, the same data splitting described previously is used across all n_p processes at all L_t time-levels. Because, the parallelization is only over the time domain, spatial restriction and interpolation method may remain unchanged. The time restriction and interpolation operators described in section 4.1.2, are not purely local. That is, in order to restrict the local data on process n, data is required from process $n-1$ and $n+1$.

Take, for example, an $N = 16$, time-domain restricted to an $N = 8$ time-domain, split across $n_p = 4$ processes. Process number 1 is responsible for time-steps 4-7 on the fine grid, and 2-3 on the course grid. However, the coarse residual at time-step 2 requires knowledge of the fine-step 3 which is stored on process 0, not locally on process 1. An identical problem occurs when interpolating back from the coarse grid to the fine grid. However, because of the specific restriction operator chosen, only one additional point is required from the neighboring processes to apply the operator. Figure 4-7 shows one example of

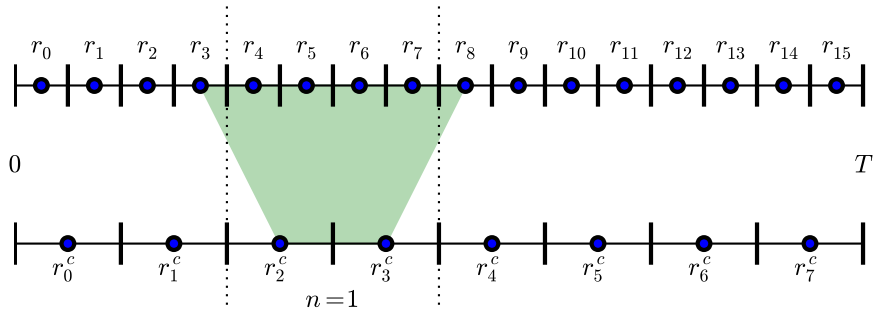

Figure 4-7: Data dependence for parallel restriction, ($N = 16$, $n_p = 4$)

how this data dependence works out.

Restriction

Each local data chunk is padded with one additional time-step worth of data on each side (Note: the first step of process 0 and the last step of process $n_p - 1$ are padded with an M-dimensional zero vector) . These additional ghost time-steps are sufficient so that restriction can be done locally. The local time-restriction operators (R_t^n) are $\frac{1}{2}\frac{N}{n_p} \times \left(\frac{N}{n_p} + 2\right)$

matrices, and are shown in equation (4.12).

$$
R_t^0 = \begin{bmatrix}
0 & \frac{6}{17} & \frac{6}{17} & \frac{2}{17} & & & & & & \\
 & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & & & \\
 & & & & & \ddots & & & & \\
 & & & & & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
\end{bmatrix}
$$

$$
R_t^n = \begin{bmatrix}
\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & & & & & \\
 & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & & & \\
 & & & & & \ddots & & & & \\
 & & & & & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
\end{bmatrix} \qquad 0 < n < n_p - 1
$$

$$
R_t^{n_p-1} = \begin{bmatrix}
\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & & & & & \\
 & & & \ddots & & & & & & \\
 & & & & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & & \\
 & & & & & & \frac{2}{17} & \frac{6}{17} & \frac{6}{17} & 0
\end{bmatrix} \tag{4.12}
$$

Interpolation

The local interpolation operators I_t^n can be found in a similar way. These size $\frac{N}{n_p} \times \left(\frac{1}{2}\frac{N}{n_p} + 2\right)$ sub-matrices can then be applied locally to achieve the same interpolation

described before. The local interpolation matrices are shown in equation (4.13)

$$I_t^0 = \frac{1}{4}\begin{bmatrix} 0 & 2 & & & & \\ & 3 & 1 & & & \\ & 1 & 3 & & & \\ & & & \ddots & & \\ & & & & 1 & 3 \\ & & & & & 3 & 1 \end{bmatrix} \qquad I_t^{n_p-1} = \frac{1}{4}\begin{bmatrix} 1 & 3 & & & & \\ & 3 & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & 3 \\ & & & & & 2 & 0 \end{bmatrix}$$

$$I_t^n = \frac{1}{4}\begin{bmatrix} 1 & 3 & & & & \\ & 3 & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & 3 \\ & & & & & 3 & 1 \end{bmatrix} \qquad 0 < n < n_p - 1 \qquad (4.13)$$

Primal Restriction

The primal restriction operation is the final operator that must be parallelized. The parallel primal trajectory already contains some redundancy, one time-step shared between each pair of processes. However, the specified primal restriction operator requires the primal to be padded similarly to the residual restriction operators. This local primal restriction operators $R_{t,p}^n$ is shown below in equation (4.14).

$$R^0_{t,p} = \begin{bmatrix} 0 & 1 & & & & & \\ & & 1 & 3 & 1 & & \\ & & & & \ddots & & \\ & & & & 1 & 3 & 1 \end{bmatrix} \qquad R^{n_p-1}_{t,p} = \begin{bmatrix} 1 & 3 & 1 & & & & \\ & & 1 & 3 & 1 & & \\ & & & & \ddots & & \\ & & & & & 1 & 3 & 1 \end{bmatrix}$$

$$R^n_{t,p} = \begin{bmatrix} 1 & 3 & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & 3 & 1 \\ & & & & & 1 & 0 \end{bmatrix} \qquad 0 < n < n_p - 1 \quad (4.14)$$

4.2.4 Parallel Arithmetic

The array arithmetic required for parallel MINRES, and other operations required for multigrid are largely trivial. Addition and subtraction can be done locally within each process without any communication. A parallel dot product is simply computed using the sum of all local dot products. Once these operations are parallel there are no other necessary modifications to make the multigrid solver run in parallel.

4.2.5 Implementation Details

The parallel multigrid solver was written in the Python™ programming language. Heavy use was made of the NumPy and SciPy projects [12]. The parallelization was developed with MPI using OpenMPI, and specifically the `mpi4py` Python bindings [8]. The Fast Fourier Transform (FFT) implementation used later in section 4.4, is provided by `anfft` [7], a NumPy compatible wrapper to the FFTW library [10].

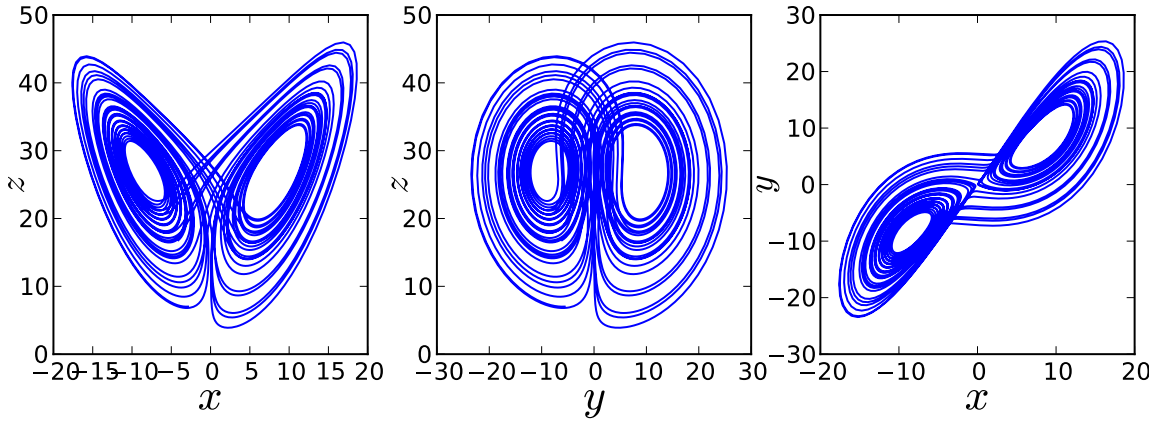

Figure 4-8: Example Primal Trajectory for Lorenz System ($\rho = 28$, $\sigma = 10$, $\beta = 8/3$), over time span $T = 40.96$

4.3 ODE Results (Lorenz System)

The Lorenz equations were first formulated by Edward Lorenz as a very low order model of Rayleigh-Bernard Convection [16]. For certain combinations of parameters (ρ, σ, β) this third order ODE exhibits chaotic properties. Figure 4-8 shows an example of a solution to the Lorenz equations in the chaotic regime.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{bmatrix} \tag{4.15}$$

$$\vec{u} \equiv [\ x\ \ y\ \ z\ ]^T$$

In climate sensitivity problems for the Lorenz attractor, people often examine the time-averaged coordinate positions [14, 19, 20, 24]. Effectively looking at the sensitivity of the center of mass of the attractor with respect to changes in the parameters (ρ, σ, β), another simple perturbation is applying a shift to a coordinate (i.e.

$z' = z + z_0$) as shown in (4.16). Perturbations in z_0 will then produce an equal perturbation in the average z coordinate.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z' \end{bmatrix} = \begin{bmatrix} \sigma(y - x) \\ x(\rho - (z' - z_0)) - y \\ xy - \beta(z' - z_0) \end{bmatrix} \tag{4.16}$$

Unlike for PDE problems, the LSS operator S, can be fully precomputed for Lorenz system problems. The result is a $3N \times 3N$ sparse matrix with 3×3 blocks. This matrix could be feasibly solved using sparse direct methods, or various krylov solvers. A finely tuned sequential direct solve, could very easily out perform the parallel multigrid implementation described in this thesis. Also, in small ODE problems, like the Lorenz System, communication costs required to parallelize these algorithms are non-trivial compared to the overall solution time. Therefore, this section is most useful to verify the parallel multigrid implementation.

4.3.1 Results

The "true" sensitivity to the different parameters in the Lorenz equations have been estimated in previous works.[14, 24]. These can be used as baselines to verify the parallel LSS solver. Table 4.2 shows various possible objective functions and perturbations, and how the LSS solver compares at computing these sensitivities. In all cases the sensitivity predicted is fairly accurate, in the cases with non-analytical solutions, the estimates lie within previously estimated ranges.

Figure 4-9 shows how the convergence of the multigrid LSS system compares to parallel implementations of conjugate gradient and MINRES on the smae machine. For the chosen parameters, the multigrid solver does not dramatically improve on these more simple solvers, however it is slightly better. Figure 4-10 shows the trajectories of the Lagrange multipliers (w) and tangent solution (v) for this problem.

Objective (J)	Perturbation	Sensitivity	LSS Prediction
z	z_0	1	0.9991
x^2	z_0	0	1.044×10^{-3}
y^2	z_0	0	-9.372×10^{-3}
z	ρ	1.01 ± 0.04 [24]	1.008
x^2	ρ	2.70 ± 0.10 [24]	2.681
y^2	ρ	3.87 ± 0.18 [24]	4.027

Table 4.2: Climate Sensitivity values of Lorenz System at ($\rho = 28$, $\sigma = 10$, $\beta = 8/3$, $z_0 = 0$). LSS parameters ($\alpha = 10\sqrt{10}$, $T = 81.96$, $\Delta t = 0.01$). Equations solved to a relative tolerance of 10^{-8}

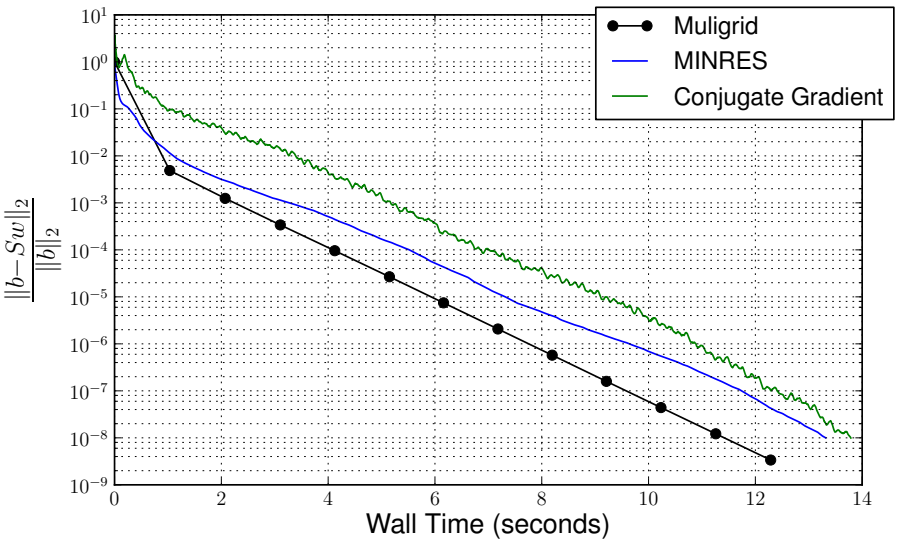

Figure 4-9: Comparison of convergence rates for parallel multigrid versus parallel MINRES, and parallel conjugate gradient implementations for solving discrete LSS equations for Lorenz System, perturbations to ρ, ($\rho = 28$, $\sigma = 10$, $\beta = 8/3$), ($\alpha = 10\sqrt{10}$, $T = 81.96$, $\Delta t = 0.01$), $n_p = 2$

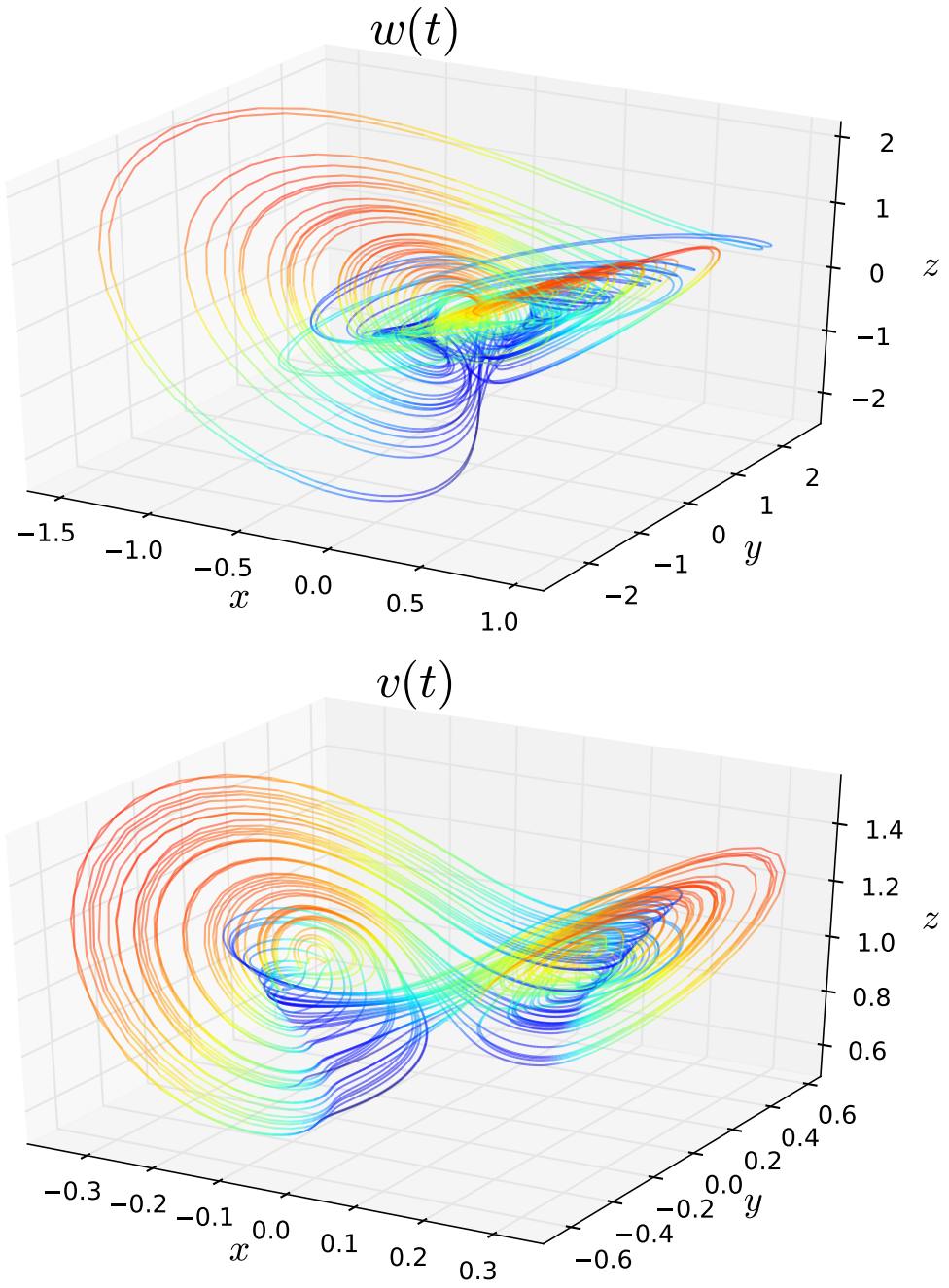

Figure 4-10: Solutions to the LSS equations, the Lagrange multipliers $w(t)$, and the tangent solution $v(t)$, colored by the local time-dilation $\eta(t)$

4.4 PDE Problem (Homogeneous Isotropic Turbulence)

To validate the LSS solver further, the same solver was run on a model of Homogeneous Isotropic Turbulence (HIT). A traditional HIT solver involves a direct numerical simulation of the three-dimensional Navier-Stokes equations on cube of fluid. The boundaries of the cube are periodic in each direction, and the fluid is forced, deterministically, to drive the flow [11]. Given the correct choice of forcing F, the result is a fully three-dimensional, unsteady, flow, that is chaotic with some bounded energy distribution. Typically, this flow is assumed to be incompressible, as shown in equation (4.17).

$$\frac{\partial}{\partial t}U(x,t) + \nabla \cdot \Big(U(x,t) \otimes U(x,t)\Big) = -\nabla P(x,t) + \nu \nabla^2 U(x,t) + F\left(\ U(x,t)\ \right) \qquad (4.17)$$
$$\nabla \cdot U(x,t) = 0$$

The simple, periodic, domain lends itself to a pseudo-spectral solution method. Given a spatial Fourier transform $\mathcal{F}$, equation (4.19) shows the pseudo-spectral transformation of incompressible Navier-Stokes. $\hat{G}$ represents the nonlinear convection operator, and is computed in the physical domain (4.20).

$$\hat{U}(\mathbf{k},t) = \mathcal{F}\left[U(x,t)\right] \qquad (4.18)$$

$$\frac{\partial}{\partial t}\hat{U}(\mathbf{k},t) + \hat{G}\left(\hat{U}(\mathbf{k},t)\right) = -i\mathbf{k}\hat{P}(\mathbf{k},t) - \nu|\mathbf{k}|^2\hat{U}(\mathbf{k},t) + \hat{F}\left(\ \hat{U}(\mathbf{k},t)\right) \qquad (4.19)$$
$$\mathbf{k} \cdot \hat{U}(\mathbf{k},t) = 0$$

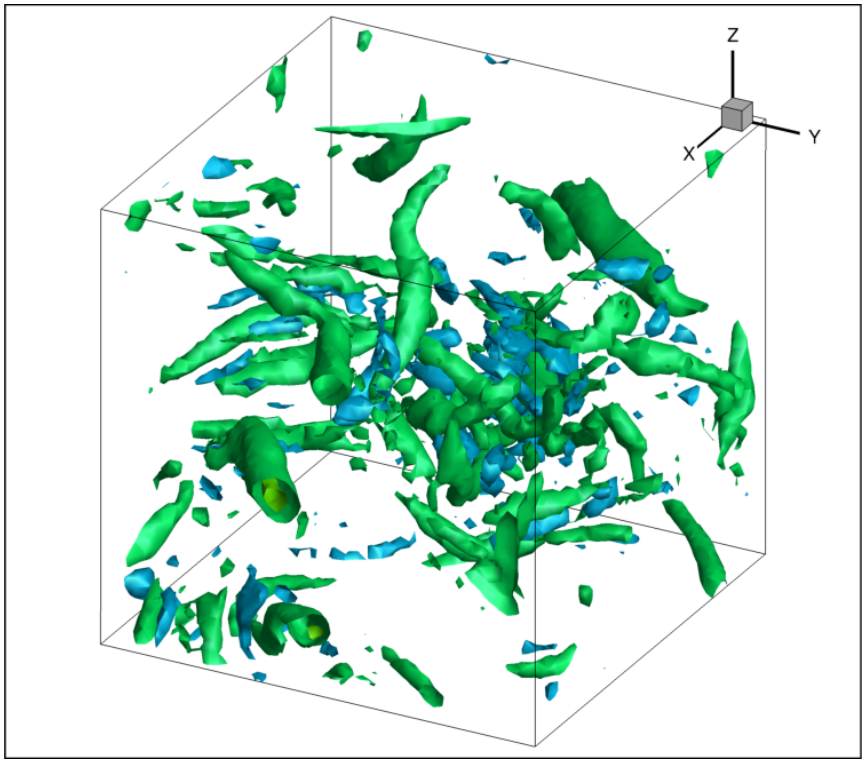

Figure 4-11: Iso-surfaces of the Q-Criterion for an example flow field produced by HIT, ($Q = \frac{1}{2}\left(\|\Omega\|_f^2 - \|S\|_f^2\right)$, $\Omega = \frac{1}{2}\left(\nabla U - \nabla U^T\right)$, $S = \frac{1}{2}\left(\nabla U + \nabla U^T\right)$)

$$U(x,t) = \mathcal{F}^{-1}\left[\hat{U}(\mathbf{k},t)\right]$$
$$\hat{G}\left(\hat{U}(\mathbf{k},t)\right) = i\mathbf{k}\cdot\mathcal{F}\left[\ U\otimes U\ \right] \tag{4.20}$$

Spectral Forcing

The spectral forcing, $\hat{F}(\hat{U})$, is chosen to inject energy at some rate to balance the energy loss from viscosity. In the spectral domain this forcing is applied to the lower frequency modes ($\tilde{\mathbf{k}}$) with wavenumber given in (4.21).

$$\tilde{\mathbf{k}}_0 \equiv (1,0,0), \quad \tilde{\mathbf{k}}_1 \equiv (0,1,0), \quad \tilde{\mathbf{k}}_2 \equiv (0,0,1) \tag{4.21}$$

The energy ϵ is computed on these low-frequency modes (4.22). The forcing is proportional to some constant power P divided by the low-frequency energy ϵ, and proportional to the low-frequency spectral velocity (4.23). The higher-frequencies are left unforced.

$$\epsilon = \sum_{i=0}^{2}\left|\hat{U}(\tilde{\mathbf{k}}_i,t)\right|^2 \tag{4.22}$$
$$\hat{F}(\tilde{\mathbf{k}}_i,\ t) = \frac{P}{\epsilon}\ \hat{U}(\tilde{\mathbf{k}}_i,\ t) \tag{4.23}$$
$$\hat{F}(\mathbf{k}\notin\tilde{\mathbf{k}},\ t) = 0$$

The spectral forcing coefficient, P, may be chosen to achieve the desired Taylor micro-scale Reynolds number. Here the coefficient is chosen to be proportional to the kinematic viscosity (ν) (4.24). β is a fractional deviation from the nominal power coefficient, and will be the perturbation parameter considered here.

$$P \equiv 2\nu\ (1+\beta) \tag{4.24}$$

Objective Function

The objective function considered in this problem is the cumulative energy spectrum of the flow field (4.25), and its long-time average (4.26). When increasing β, the energy at all values of the wavenumber magnitude ($|\mathbf{k}|$) are expected to increase on average, however, the exact amount is difficult to obtain. The goal will be to simulate one primal flow field. That one primal trajectory is then used to predict the sensitivity of the energy spectrum $d\bar{E}(|\mathbf{k}|)/d\beta$, across all frequencies.

$$E(|\mathbf{k}|,\ t) = \sum_{k \geq |\mathbf{k}|} \left| \hat{U}(k,\ t) \right|^2 \tag{4.25}$$

$$\bar{E}(|\mathbf{k}|) = \lim_{T \to \infty}\ \frac{1}{T} \int_0^T E(|\mathbf{k}|,\ t)\ dt \tag{4.26}$$

4.4.1 Discretization

The spatial domain is a cube of size $[0,\ 2\pi]^3$, each velocity component is discretized on a $(3k \times 3k \times 3k)$ Cartesian grid ($U \in \mathbb{R}^{3\times 3k\times 3k\times 3k}$). In the spectral domain the 2/3 rule is applied to avoid aliasing, giving a maximum wave number of $2k$ in each dimension. Additionally because the flow is real valued, the fourier modes will be symmetric and may be further reduced by about a half ($\hat{U} \in \mathbb{C}^{3\times 2k\times 2k\times (k+1)}$). The discrete Fourier transforms, and inverse transforms are computed via a real valued Discrete Fourier Transform [7, 10].

4.4.2 Application to LSS

For the purposes of the LSS solver, and to emphasize the generality of the method, these equations will be treated as a black-box differential equation. The discrete isotropic flow solver will implement a simple interface, providing access to evaluating f and multiplication by $\frac{\partial f}{\partial u}$ and $\frac{\partial f}{\partial u}^T$. In fact, the LSS solver does not even know that the isotropic turbulence model is solved in the complex domain, it simply treats the discretized solution as real

system of twice the size, ($M_k = 2 \times 3 \times 2k \times 2k \times (k+1) = 24k^2(k+1)$). The flow solver will also provide the spatial restriction and interpolation methods. The temporal restriction and interpolation techniques described previously still apply.

Spatial Restriction and Interpolation

The spatial restrictions and interpolations are performed in the frequency domain. To coarsen in space from size M_k to $M_{k/2}$, the higher frequency modes are simply truncated. At grid size k, all the discrete frequencies ($\mathbf{k}$) satisfy the inequality $0 \leq \|\mathbf{k}\|_\infty \leq 2k$. Therefore at size $k/2$ the coarse solution is given by

$$\hat{w}_c(\mathbf{k},\ t) = \hat{w}(\mathbf{k},\ t) \qquad \forall \mathbf{k} :\ 0 \leq \|\mathbf{k}\|_\infty \leq k \tag{4.27}$$

$$M_{k/2} = 6k^2(k/2+1) \tag{4.28}$$

Similarly, to interpolate back from size $k/2 \to k$, the solution is simply padded with zeros for all wave numbers above the threshold.

$$\hat{w}(\mathbf{k},\ t) = \begin{cases} \hat{w}_c(\mathbf{k},\ t) & 0 \leq \|\mathbf{k}\|_\infty \leq k \\ 0 & k < \|\mathbf{k}\|_\infty \leq 2k \end{cases} \tag{4.29}$$

4.4.3 Results

One primal solution of the Isotropic Flow solver is produced using the discretized isotropic flow equations. The solution is obtained using the same implicit trapezoidal method described previously. For all future results assume the isotropic flow is solved using the

following parameters.

$$\nu = 0.01$$

$$k = 16$$

$$\Delta t = \frac{2\pi}{16}$$

The Taylor micro-scale for this flow is $\lambda = 1.22$ and the associated Reynolds number $Re_\lambda = 33.06$. The primal solution is solved for $N+1$ time-steps from $0 \rightarrow N\Delta t$, here N will be assumed to be $2^{12} = 4096$. Several example solutions are shown in figure 4-12. Figure 4-13 shows the mean energy spectrum for one such run. The LSS equations for the long scale $N = 2^{12}$ problem are solved utilizing $n_p = 64$ processes. Because of a limitation in the implementation of the parallelization, the multigrid cannot coarsen in time below n_p time-steps. Again the time-dilation weighting, $\alpha = 10\sqrt{10}$, is used.

The climate sensitivity is then computed in two ways. A finite difference estimate of the β sensitivity is performed. β is perturbed $\Delta\beta = 0.05$ which produces a 5% change to the input power P.

$$\beta^+ = 0.05 \qquad \beta^- = -0.05 \tag{4.30}$$

Two new primal trajectories are simulated with these two perturbed β values.

$$\hat{U}^+ = \hat{U}(\mathbf{k},\ t;\ \beta^+) \tag{4.31}$$

$$\hat{U}^- = \hat{U}(\mathbf{k},\ t;\ \beta^-) \tag{4.32}$$

The difference in the instantaneous cumulative energy spectrum (4.26) between the two

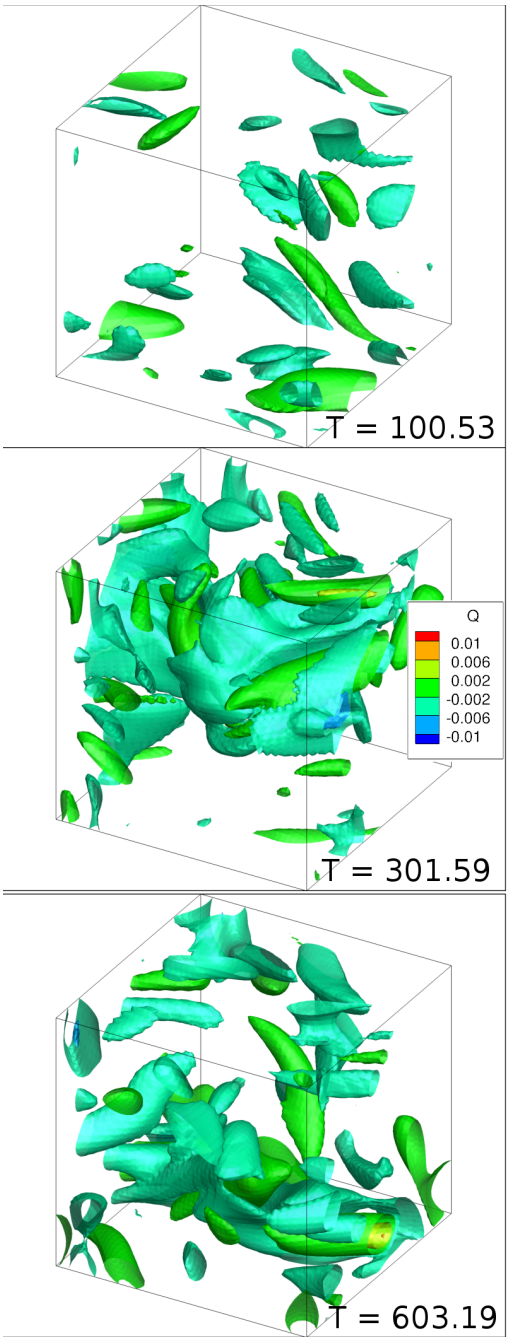

Figure 4-12: Q-Criterion iso-surfaces for several primal solutions ($Re_\lambda = 33.06$)

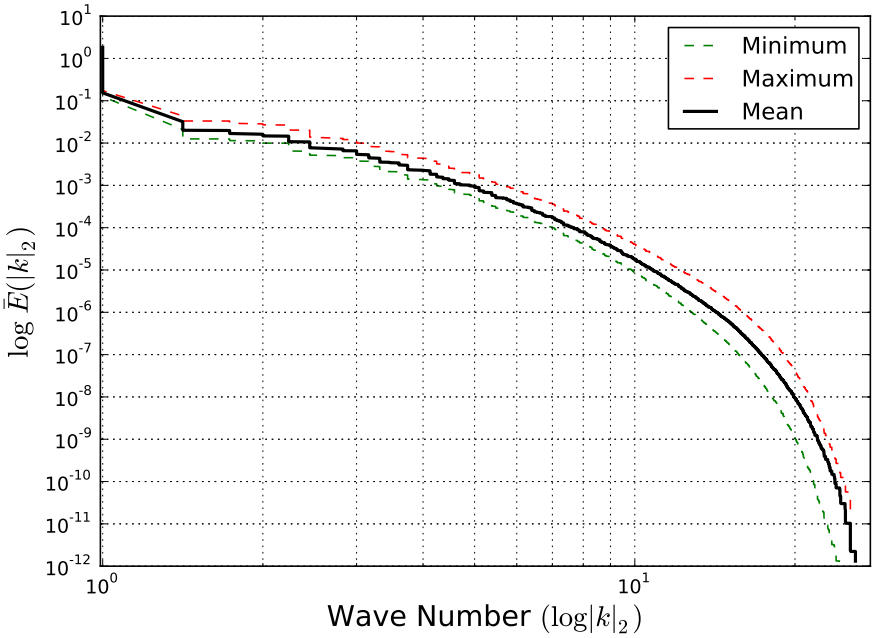

Figure 4-13: Average cumulative energy spectrum $\bar{E}$, for primal solution to HIT

perturbed runs is computed, and time-averaged.

$$E^+ = E(|k|,t)\Big|_{\hat{U}^+} \qquad E^- = E(|k|,t)\Big|_{\hat{U}^-} \tag{4.33}$$

$$\frac{d}{d\beta}E(|k|,\ t) \approx \frac{E^+ - E^-}{2\Delta\beta} \tag{4.34}$$

$$\frac{d}{d\beta}\bar{E}(|k|)_{(FD)} = \frac{1}{N+1}\sum_{i=0}^{N}\frac{E_i^+ - E_i^-}{2\Delta\beta} \tag{4.35}$$

These are compared against the sensitivity estimate given by the LSS equations given by the unperturbed trajectory ($\hat{U}$). The LSS solution is obtained using the parallel multigrid algorithm to compute w, v, and η.

$$b \leftarrow \frac{\partial f}{\partial \beta} = \frac{\partial \hat{F}}{\partial \beta} \tag{3.18}$$

$$w \leftarrow Sw = b \qquad \text{(parallel multigrid)}$$

$$v,\ \eta \leftarrow w \qquad (3.24),\ (3.25)$$

Then the sensitivities from that solution are approximated using equation (3.15).

$$\bar{\eta} = \frac{1}{N}\sum_{j=0}^{N-1}\eta_j \tag{4.36}$$

$$\hat{U}_j^m = \frac{1}{2}\left(\hat{U}_j + \hat{U}_{j+1}\right) \tag{4.37}$$

$$\frac{\partial}{\partial\beta}\bar{E}(|k|)_{(LSS)} = \frac{1}{N+1}\sum_{i=0}^{N}\frac{\partial E}{\partial\hat{U}}\Big|_{\hat{U}_i} v_i + \frac{1}{N}\sum_{j=0}^{N-1}\left(\bar{\eta}-\eta_j\right)\ E\Big|_{\hat{U}_j^m} \tag{4.38}$$

Figure 4-14 shows how these two methods compare. The LSS method seems to produce relatively good sensitivity estimates compared with finite difference. Figure 4-15 shows the absolute value of the instantaneous spectrum sensitivity. Figure 4-16 contains a visualization of the Q-criterion of the Lagrange multiplier solution of the LSS system.

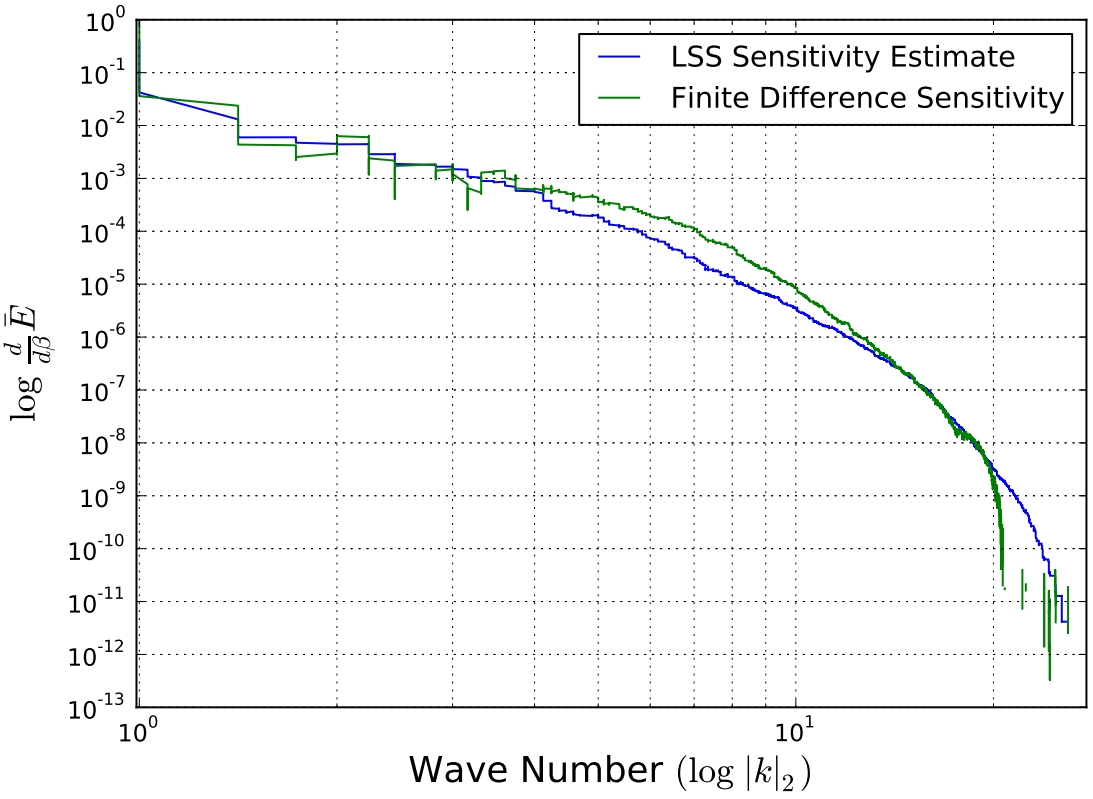

Figure 4-14: LSS vs. Finite Difference sensitivity estimates. Finite differences computed using two primal solutions with the forcing magnitude altered by $\pm 5\%$ ($\Delta\beta = \pm 0.05$). LSS estimate computed using parallel multigrid and equation (3.15).

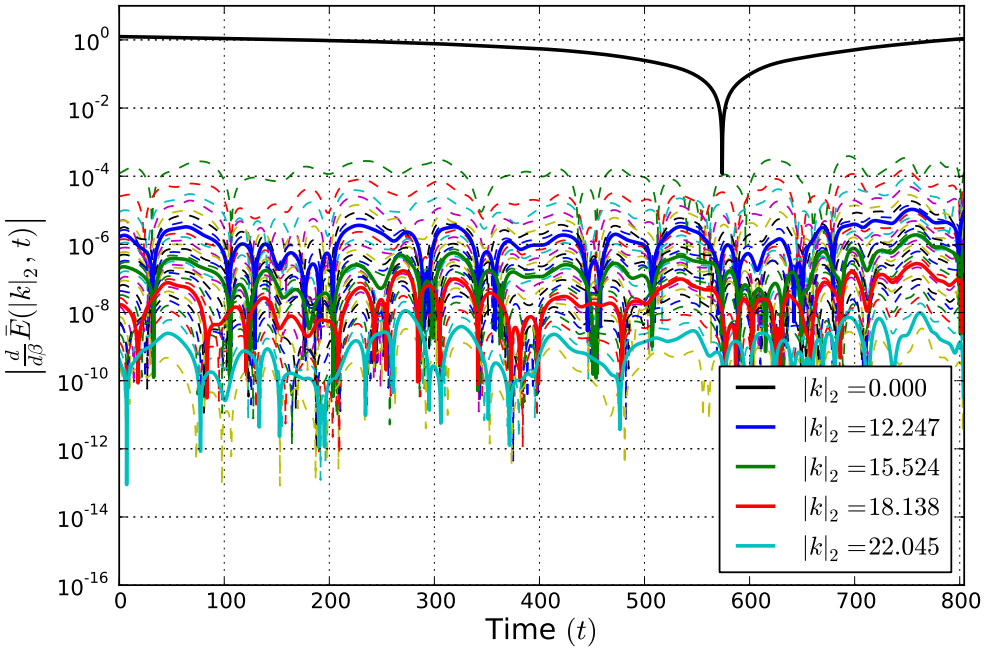

Figure 4-15: Absolute value of instantaneous spectrum sensitivity for various wave number magnitudes ($|k|_2$) computed from LSS. Dashed lines illustrate this instantaneous sensitivity measure at several values of $|k|_2$. Solid lines show a few examples curves where the wave number magnitudes are explicitly labeled.

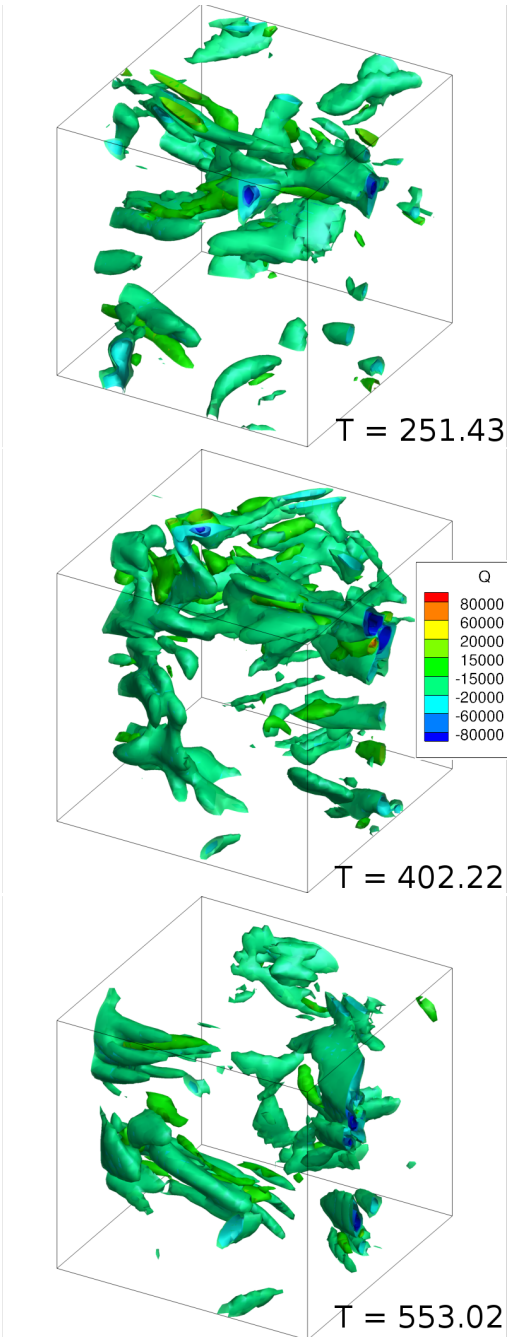

Figure 4-16: Q-Criterion of Lagrange multiplier field (w) at several points in time

Chapter 5

Conclusions

There are many difficulties with computing accurate sensitivity gradients for periodic and chaotic dynamical systems. Looking at "climate" properties of dynamical systems can give a window into understanding the dynamics of a chaotic system. However, even by reducing the problem scope to climate sensitivities, general methods for efficiently computing these sensitivities have been difficult to obtain. In this thesis, two approaches were examined for computing sensitivities to long-time averaged quantities in dynamical systems.

The first approach involved the special case of periodic systems. Here the instability that comes with solving the adjoint equations on chaotic systems is not a problem. Instead, the adjoint equations do not produce unique solutions. This is rectified by introducing the concept of the homogeneous adjoint equations, which in periodic and chaotic systems, have non-trivial solutions. Solutions to the homogeneous adjoint equations can be added to any inhomogeneous solution to make more valid adjoint solutions. The result is an additional constraint that is placed on the inhomogeneous adjoint solution. The solution that satisfies this constraint is the "true" adjoint solution that can predict both averaged and instantaneous sensitivities. An algorithm for computing this corrected adjoint is also presented, involving the additional cost of the simultaneous solution of an additional set

of homogeneous adjoint equations. This approach is verified on the Van der Pol Oscillator ODE system.

The second approach deals with the Least Squares Sensitivity (LSS) method of computing sensitivities to "climate" properties. Contrary to traditional sensitivity methods, an optimization problem is formed to minimize the linear solution perturbation v, and the time-dilation η. This minimization problem results in a large sparse SPD linear system for computing the Lagrange multipliers of the sensitivity. For ODE systems this resulting LSS system may be directly solved, however for large scale systems, there are difficulties constructing this system, let alone solving it. A parallel multigrid algorithm for solving the resultant LSS equations is developed to deal with both small scale chaotic ODEs and large scale PDEs. This algorithm makes no assumptions about the ability to numerically form the LSS system, and instead only requires a functional interface for computing the system dynamics, and multiplying by the Jacobian and its adjoint. This parallel LSS solver is then verified using a chaotic ODE, the Lorenz System, and also a chaotic PDE, from a direct numerical simulation of Homogeneous Isotropic Turbulence. Solution of both systems yield approximate sensitivity estimates that agree with literature and finite difference approximations.

These two methods increase the efficiency with which periodic and chaotic climate sensitivities may be computed. While periodic systems are very rare, understanding these problems can lead to better understanding of the more difficult chaotic problems. The parallel-in-time multigrid algorithm presented here shows great promise of making sensitivity computations for large scale chaotic systems more feasible than they have previously been. The success of the approach for a relatively small turbulent fluid flow problem could be extended to more complex turbulent flows seen in engineering applications. The structure of the LSS system lends itself to massive parallelization. In addition, the generality of the multigrid method developed here allows for a generic LSS solver to handle a variety of different dynamical systems without modification. This ability to compute sensitivities us-

ing arbitrary systems is an important step towards the development of push-button design tools that can handle chaotic dynamical systems.

Bibliography

[1] R.V. Abramov and A.J. Majda, *Blended response algorithms for linear fluctuations-dissipation for complex nonlinear dynamical systems*. Nonlinearity, 20, Issue 12, 2007, pp. 2793-2821

[2] R.V. Abramov and A.J. Majda, *A new algorithm for low-frequency climate response*. Journal of Atmospheric Sciences, 66, 2008, pp. 286-309.

[3] R.E. Bank and C.C. Douglas, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*. SIAM journal on numerical analysis, 22.4:617633, 1985.

[4] P.J. Blonigan, R. Chen, Q. Wang, and J. Larsson, *Towards adjoint sensitivity analysis of statistics in turbulent flow simulation*. Stanford, 2012, pp. 229-239

[5] V. Bugnion, C. Hill, P. Stone, *An adjoint analysis of the meridional overturning circulation in an ocean model* Journal .of Climate, V.19, 2005 pp. 3732-3750

[6] T. F. Chan and W. L. Wan. *Robust multigrid methods for nonsmooth coefficient elliptic linear systems*. Journal of Computational and Applied Mathematics, 123.1:323352, 2000.

[7] A. Collette, et. al., **ANFFT**: An FFT package for Python$^{\text{TM}}$, based on FFTW, 2012, "https://code.google.com/p/anfft/"

[8] L. Dalcin, et. al., **MPI4PY**: MPI for Python$^{\text{TM}}$, 2012, "https://code.google.com/p/mpi4py/"

[9] G. Eyink, T. Haine, and D. Lea, *Ruelles linear response formula, ensemble adjoint schemes and Levy flights*. Nonlinearity, Vol. 17, 2004, pp. 18671889.

[10] M. Frigo and S.G. Johnson, *The design and implementation of FFTW3*. Proceedings of the IEEE 93 (2), pp 216-231

[11] T. Ishihara, T. Gotoh, Y. Kaneda, *Study of HighReynolds Number Isotropic Turbulence by Direct Numerical Simulation*. Annual Review of Fluid Mechanics, Vol. 41, 2008, pp. 165-180.

[12] E. Jones, T. Oliphant, P. Peterson and others, **SciPy**: Open source scientific tools for Python^{TM}, 2001–2013, "http://www.scipy.org"

[13] J.A. Krakos, Q. Wang, S.R. Hall, and D.L. Darmofal, *Sensitivity analysis of limit cycle oscillations*. Journal of Computational Physics. Volume 231, issue 8, pages 3228-3245, 2012.

[14] D. Lea, M. Allen, and T. Haine, *Sensitivity analysis of the climate of a chaotic system*. Tellus, Vol. 52A, 2000, pp. 523532.

[15] D.J. Lea, T. Haine, M. Allen, and J. Hansen, *Sensitivity analysis of the climate of a chaotic ocean circulation model*. Q.J.R Meteorol. Soc. 2002, pp 2587-2605

[16] E.N. Lorenz, *Deterministic nonperiodic flow*. Journal of Atmospheric Sciences, 1963, V.20, pp. 130-141

[17] E.N. Lorenz, *The problem of deducing the climate from the governing equations*. . Tellus, 1964, V.20, pp. 130-141

[18] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*. SIAM J. Numerical Analysis 12, 1975, pp. 617-629.

[19] T.N. Palmer, *A nonlinear dynamical perspective on climate prediction*. Journal of Climate, V.12, 1999, pp 575-591

[20] T.N. Palmer, *Extended-Range atmospheric prediction and the Lorenz model*. American Meteorological Soc. V.74, 1993, pp. 49-65

[21] D. Ruelle, *General linear response formula in statistical mechanics, and the fluctuations-dissipation theorem far from equilibrium*. Phys. Lett, A245, 220-4

[22] J. Thuburn, *Climate sensitivities via a FokkerPlanck adjoint approach*. Quarterly Journal of the Royal Meteorological Society, Vol. 131, No. 605, 2005, pp. 7392.

[23] Q. Wang and R. Hu, *Sensitivity computation of periodic and chaotic limit cycle oscillations*. submitted to SIAM Journal of Scientific Computing. revised and submitted. arXiv:1204.0159

[24] Q. Wang, *Forward and adjoint sensitivity computation for chaotic dynamical systems*. Journal of Computational Physics, Vol. 235, No. 15, 2013, pp. 115.

[25] Q. Wang, S. Gomez, and P. Blonigan, *Towards scalable parallel-in-time turbulent flow simulations*. submitted to Physics of Fluids, 2013