Output-Based Error Estimation and Model Reduction for Chaotic Flows

by

Yukiko Sonya Shimizu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2019

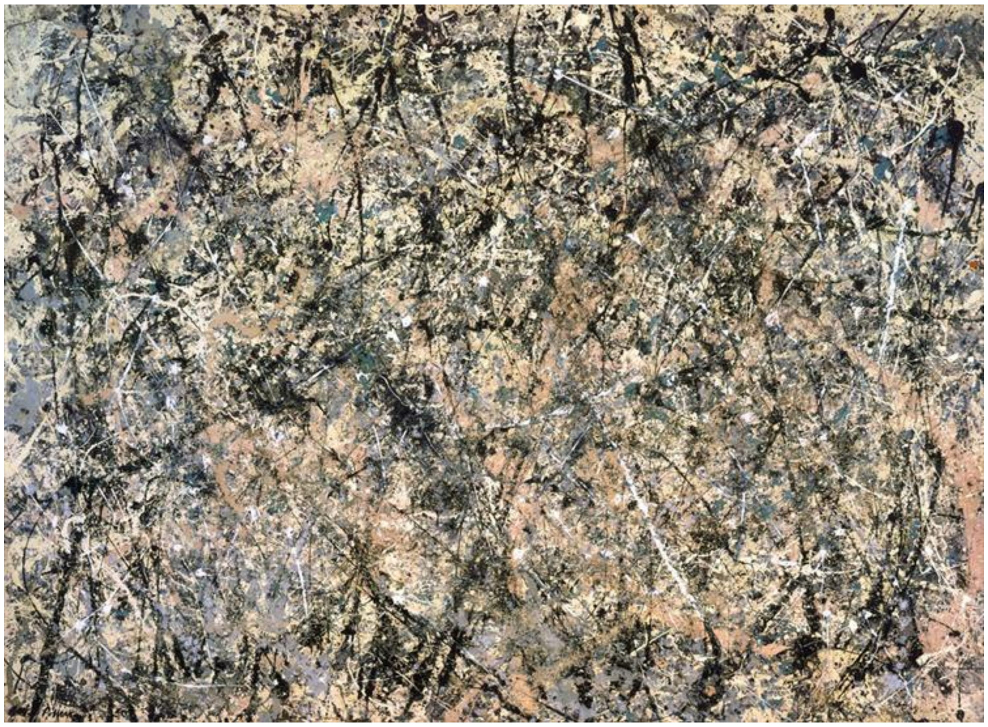


Figure 0.1: Jackson Pollock **No 1. Lavender Mist**, 1950

"*Through his drip action technique, Pollock would create layers upon layers of paint, created in a chaotic assemblage of drips and splashes. Physicists have studied Pollock's canvases, such as Lavender Mist, for fractals, which naturally occur out of chaos. Some art scientists, such as Richard Taylor, have determined that the more chaotic Pollock's drippings became, the closer they resembled naturally occurring fractals, and that his chronologically later paintings displayed these characteristics more so than his earlier works, which were less chaotic* [1]."

Yukiko Sonya Shimizu

ykmizu@umich.edu

ORCID iD: 0000-0003-1934-6688

To my Dad and my Mom
To my sister
To お祖母さん and お爺さん
To 할머니 and 할아버지

ACKNOWLEDGMENTS

I would like to first convey my deepest gratitude to my advisor and mentor, Professor Chris Fidkowski. Chris was the first person to believe that I was capable of pursuing a PhD. It was not easy going down the path of an independent researcher, and there were many ups and downs during my career here, but Chris was always encouraging and patient with me. Now that I have finished my PhD at the University of Michigan under his guidance, I have gained an even deeper appreciation for computational fluid dynamics and applied mathematics. I intend to continue to do research-level work in these fields and feel truly fortunate to have had the opportunity to be his student.

Additionally, I would like to thank Professor Kenneth Powell, Professor Yin Lu (Julie) Young, Joshua Krakos, and Professor Qiqi Wang for serving on my committee. I am grateful for their feedback and guidance on my research. In particular, I would like to give special thanks to Professor Wang for having supported me since my undergraduate years at MIT.

I would like to thank my father. When I was a kid, my father showed me Apollo 13 and The Right Stuff – two movies that inspired me to pursue a career in aerospace. They are still my favorites even now. In middle school I was heavily involved in a competition called Science Olympiad. My father was the one who encouraged me to participate, teaching me how to build bridges out of balsa wood and airplanes out of rubber band and Mylar. These early lessons laid the foundations for my engineering pursuits, and I wouldn't be submitting this thesis had it not been for those experiences.

I would like to thank my mother. My mother is an artist who taught me from an early age how to draw upon my creative intuition and how to approach my work with conviction and passion. She encourages me everyday and is my biggest supporter in whatever I do. Her steadfast commitment as a mother and her ability to persevere through anything have shaped who I am today. Whenever I had a moment of doubt in the midst of my research, she always picked me up with her love and encouragement.

Also, thank you to my dear sister, Aki, for being the most supportive and understanding sister one can ever have. She has always been there for me and has always been more than happy to help me in any way that she could during this process. I am truly lucky to have a sister like her in my life.

My graduate career at Michigan would not have been successful without my lab mates from the Computational Fluid Dynamics Group. I would like to say thank you to past and present members, Johann Dahm, Kyle Ding, Devina Sanjaya, Guodong Chen, Gustavo Halila, Gary Collins, Vivek Ojha, Matteo Franciolini, and Qingzhao Wang for being so welcoming of me and for helping me out whenever I needed it. I'm grateful that our lives crossed paths and hope that we may have more opportunities to work on research together in the future. Thank you to Johann for always reminding me that I *am* good at research, to Kyle for giving me the most hilarious memories, to Devina for always being so supportive, and to Guodong for being the best listener a friend could ever ask for.

I would like to thank all the friends I met in Michigan outside of the lab group as well. There are so many of you whom I have had the good fortune of meeting throughout my stay. Please know that I feel lucky to have met you all. I would like to thank the following people in particular: Doreen Fan, Nishant Narechania, José Márquez, Helen Zhang, Anand Singh, Luke Chiang, Yifan Bai, and Katrina Chaves. I will forever be in your debt.

Lastly, I would like to thank my undergraduate friends who have supported me throughout my PhD career: Kyumin Lee, Rex Lam, Mengfei Yang-Flint, Niels Zussblatt, and Carlos Ruvalcaba. Thank you for still being my friends after all these years.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF ALGORITHMS

ALGORITHM

ABSTRACT

Turbulent flows are characterized by chaotic variations in state variables and are commonly found in many applications such as jet engine mixing and flow over bluff bodies. Large Eddy Simulations (LES) of these chaotic flows have already proven to be useful to the design process. However, LES is resource and time-intensive. Application of output-based methods for error estimation and mesh adaptation would decrease the cost of these chaotic simulations while still retaining their overall accuracy. However, a direct application of unsteady adjoint-based methods is not possible due to the flows' inherent sensitivity to the initial conditions and the exponential growth of the corresponding adjoint solutions. This dissertation proposes the Hyper-Reduced Order Modeling-Least Squares Shadowing (HROM-LSS) method, which combines model reduction principles with adjoint sensitivity techniques for chaotic flows to calculate accurate adjoints that are cheaper to solve for than the Least Squares Shadowing (LSS) method on its own. All primal solutions are solved using the discontinuous Galerkin finite element method. Results of the HROM-LSS method for the Kuramoto-Sivashinsky equation and the NACA 0012 airfoil at high Reynolds numbers show promise for this combined method and have been shown to outperform the LSS method when calculating the effect of the discretization errors on the output. In particular, the average CPU times for the HROM-LSS method are reduced by as much as 97.44% for short time simulations and as much as 64% for longer simulations, making the HROM-LSS method a more practical option to calculate adjoint for chaotic flows in order to perform output-based error estimation for turbulent flows.

CHAPTER I

Introduction

1.1 Motivation

Computational fluid dynamics (CFD) has evolved to become more sophisticated to solve much more complicated problems. With the aid of increasing computational power, CFD has allowed aeronautical engineers and scientists to simulate flow over new conceptual aircraft configurations and to accurately predict desired outputs such as lift and drag that would otherwise be difficult or expensive to find from experiments. This increased prediction capability make it possible to design more innovative and more efficient aircrafts.

The computational power needed to perform high-fidelity CFD simulations has been increasing for many years. Moore's law, construed in 1965, first stated that transistors were shrinking so quickly that every year the number of transistors that could fit onto a chip doubled. In 1975, it was adjusted to double every two years. This prediction has been consistent for the past 50 years. However, the reality is that the number of transistors that can fit onto a chip is beginning to diverge away from Moore's law and to plateau. Therefore, computer chip manufacturers and researchers who rely on the increase in computational power will need to become more creative in the way the computational resources are used. More powerful algorithms and new advances in computational tools are needed to make simulations more efficient and less computationally expensive for CFD.

One way to reduce the computational resources required to run complicated CFD simulations is to analyze the effect of the errors on the simulations. This dissertation in particular focuses on the effect of the discretization errors on the output, which is quantified using the output-based error estimation method. Output-based error estimation has been proven to work successfully for steady and unsteady, non-chaotic cases. However, application of output-based error estimation for turbulent flows has been limited. Understanding and being able to predict the effect of the discretization errors on the output for turbulent flows, will allow one to provide more cost effective CFD simulations for much more complicated problems.

From the perspective of aerodynamics, turbulent flows are chaotic and are unique compared to steady and unsteady flows, because they are characterized by chaotic variations in state variables such as velocity and pressure. These variations stem from heavy mixing in the flow, which is found in many different aerospace applications such as jet engine mixing and flow over bluff bodies. In CFD, these chaotic flows can be found in numerical simulations of airfoils at high angles of attack and at high Reynolds numbers. The instantaneous drag and lift for these particular cases lack patterns and show chaotic variations with time. From the perspective of hydrodynamics, flexible hydrofoils experience complicated behaviors due to the interactions of the hydrofoil's natural frequencies and the fluid. A similar behavior can occur for flexible airfoils in aerodynamics. These particular conditions can lead to flutter when the structure reaches its natural frequencies. These instabilities at high Reynolds numbers can lead to chaotic responses that are dominated by the natural frequencies of the structure. Flutter leads to structural failure, requiring the need to accurately predict the natural vibration frequencies and characteristics of the flexible foil. Application of output-based error estimation for these cases in aerodynamics and hydrodynamics can decrease the computational cost of these simulations without sacrificing accuracy by estimating the effect of the discretization errors on the output of interest.

To simulate these chaotic flows, Large Eddy Simulations of turbulent flows are used. Large Eddy Simulations (LES) are important due to the fidelity of information they provide in the design process; however, traditional prediction tools in CFD do not work for LES. This is due to the fact that turbulent flows are chaotic, meaning that they are inherently unpredictable. Turbulent flows are computationally expensive as well, and therefore direct numerical simulation is not a practical option for the foreseeable future. Overcoming this high computational cost will open new opportunities to efficiently simulate turbulent flows. This dissertation makes it possible to economically simulate turbulent flows by focusing on the capability of calculating adjoints for chaotic flows, which are used in output-based error estimation and mesh adaptation.

1.2 Background

Figure 1.1 shows that there are many different kinds of errors associated with the understanding, qualification, and quantification of a physical process. There are errors associated with the model chosen to emulate the processes in nature. There are convergence errors found in the process of acquiring the solution from the systems of equations found from the governing equations. Experiments allow scientists and engineers to find data that describe the natural processes; however, there are errors associated with the observations that are made to acquire these data. Then there are errors in the scientific process where the experimental results and the numerical results are compared. These errors compound, demanding the need to minimize the error throughout the entire scientific process. The main type of error relevant to this research is the discretization errors and how it affects the outputs of interest. Methods used in computational fluid dynamics to solve governing equations that model natural processes include the finite difference method, the finite volume method, and the finite element method. In finite differences, discretization errors can be demonstrated by looking at the derivative $f'(u)$ at $u = u^*$ in Figure 1.2. To find the derivative at point u^*, The simple approximation of $f'(u*)$ can be found in different ways including the backward difference, the central difference, and the forward difference as seen in Figure 1.2. The formulae are

$$\begin{aligned}
\text{Forward}: &\quad f'(u^*) \approx \frac{f(u^* + \Delta u) - f(u^*)}{\Delta u} \\
\text{Backward}: &\quad f'(u^*) \approx \frac{f(u^*) - f(u^* - \Delta u)}{\Delta u} \\
\text{Central}: &\quad f'(u^*) \approx \frac{f(u^* + \Delta u) - f(u^* - \Delta u)}{2\Delta u}
\end{aligned} \tag{1.1}$$

respectively. As seen in Figure 1.2, each numerical method to calculate $f'(u^*)$ produces a slightly different approximation for the derivative and is each different from the actual derivative. This error is referred to as the discretization error and is one of the main interests of this research.

1.2.1 Error Estimation and its Application to Chaotic Systems

In this dissertation, the discretization method of interest is the discontinuous Galerkin (DG) finite element method, which discretizes the governing solution for each element locally in space. This method discretizes the governing equation in space by using piecewise high-order polynomials for the approximate solution. These high-order polynomials, or basis functions, are used to approximate the solution for each element and are discontinuous between the elements. The approximate solution is found from the summation of the state coefficients and the corresponding polynomial basis functions. The number of high-order polynomials associated with the solution for each element depends on the approximation order of the method, p, where the order of the solution is defined as $p+1$. For this research, high-order solutions, $p > 2$, are of special interest for their high accuracy and relative to computational expenses. DG is used as well due to its ease of use with mesh adaptation due to its ability to recover high-order convergence. Along with their ability for high-order

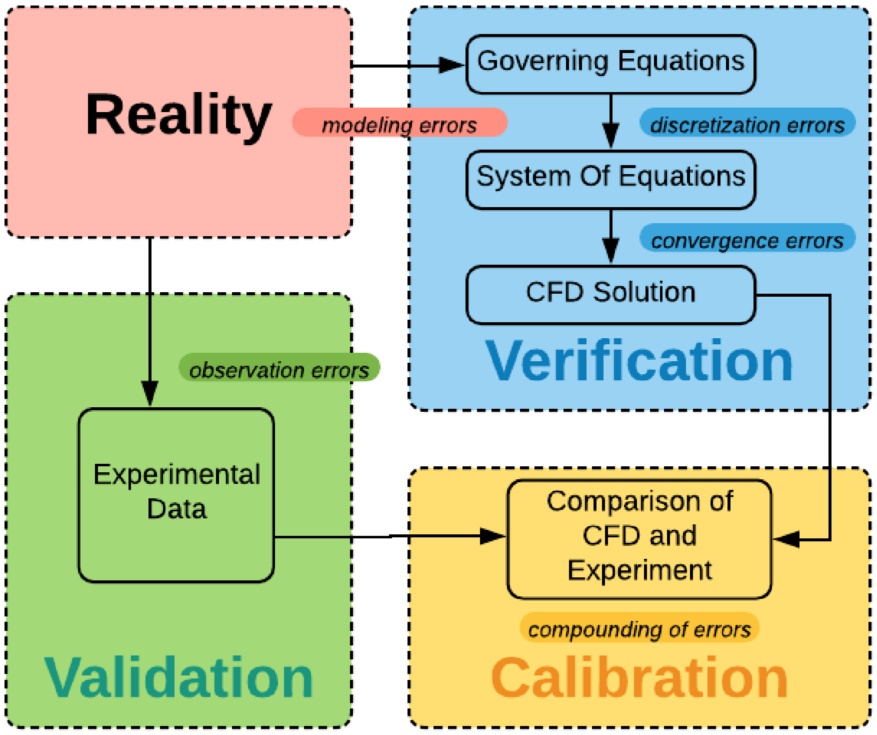


Figure 1.1: Flowchart highlights the different types of errors associated with the quantification and qualification of physical systems.

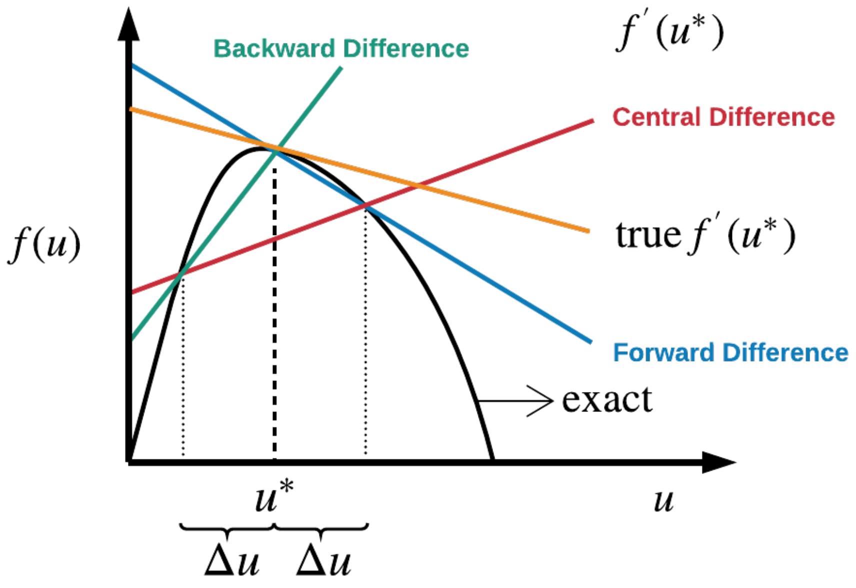


Figure 1.2: Example of discretization errors among different finite-difference methods

approximations, DG methods are highly parallelizable and can handle complicated geometries.

One particular prediction tool that can be used to understand the effects of discretization errors in computational fluid dynamics is output-based error estimation. In this method, one is interested in predicting how the discretization errors from the DG method affect the output of interest such as drag, in terms of the residual of the discretized system. Typically in output-based error estimation, one is interested in the discretized error between the coarse space solution, order p_H, and the fine space solution, order p_h, where typically $p_h = p_H + 1$. This information can be used to reduce the costs of these expensive simulations without sacrificing accuracy. Output-based error quantification and localization has the ability to provide more confidence in output values computed from chaotic simulations and to increase the efficiency of meshes through adaptation. Such adaptive capability would make LES simulations more efficient, cheaper, and more practical for analysis and design.

There has already been much successful research in steady and unsteady adjoint calculations and output-based error estimation, which rely on the linearization of the residual and the output of interest. Its success is dependent on the calculation of derivative quantities, adjoints, which are sensitivities of the outputs of interest in terms of the discretized residual. Methods like output-based error estimation for steady and unsteady systems that are dependent on adjoint calculations have advantages compared to other techniques [2]. First, these methods produce error estimates that can be used as confidence intervals to determine whether or not accuracy in the output has been obtained. Second, the errors can be localized elements in a mesh such that they can be used for mesh adaptation to reduce error. As a result, output-based methods produce better output convergence results than that of residual-based or uniform refinement techniques for deterministic problems [2]. The goal of the present work is to obtain similar improvements for chaotic problems.

However, application of the current traditional adjoint method for chaotic systems fails to produce useful adjoints for output-based error estimation, hindering the application of output-based error estimation for turbulent flows. The main goal is to study if it is possible to apply the alternative sensitivity technique, known as the Least Squares Shadowing method by Wang et. al to output-based error estimation, and whether or not it is possible to improve on it further. The challenge to this is the inherent nature of chaotic systems, which are highly sensitive to their initial conditions. Over a very short period of time, a perturbed trajectory of a chaotic system will diverge away from the original trajectory, preventing one from using the traditional linearization techniques used for steady and unsteady flows. Overcoming this issue will open new opportunities to efficiently simulate turbulent flows via output-based error estimation for mesh adaption. With better prediction capabilities, one can use output-based error estimation and mesh adaption to increase the accuracy and efficiency of these simulations.

1.2.2 The Least Squares Shadowing Method

The Least Squares Shadowing (LSS) method provides a solution to calculating usable chaotic adjoints that do not diverge compared to the traditional adjoint method. In order to use the LSS method, several theories and assumptions need to be made about chaotic systems. The first idea is ergodic theory, which states that for a chaotic system whose initial conditions are perturbed, the time average output will converge to its ensemble average output. In other words, the initial conditions have very little influence on the long time output averages. This leads to the idea that the initial conditions can be relaxed. Thus instead of using slightly perturbed conditions to calculate the adjoint, a shadow trajectory is found that exists very close to the reference trajectory. Note that this new shadow trajectory may not have initial conditions that are close to the original initial conditions, but this is inconsequential according to ergodic theory. The LSS method thus uses ergodic theory to search for a shadow trajectory that [2] follows the physics of the original problem. According to the shadowing lemma, this shadow trajectory does indeed exist for a hyperbolic or quasi hyperbolic system, which includes chaotic system' that have positive, neutral, and negative Lyapunov exponents. Similar to eigenvalues, these exponents and their corresponding covariant Lyapunov vectors, which describe the direction of the distortion dictated by the exponents, control the amount of folding and stretching that exists for a chaotic system. This continuous stretching and folding is what gives a chaotic system its characteristic sensitive behavior to its initial condition, also termed the butterfly effect.

Thus, for a quasi-hyperbolic chaotic system, the LSS method is a minimization problem that searches for a shadow trajectory that stays as close as possible to the reference trajectory with the constraint that the shadow trajectory is a solution to the perturbed governing equation. This method has already proven to be accurate at calculating chaotic sensitivities. Incorporating the LSS method in output-based error estimation will provide the ability to predict the effect of discretization errors on outputs of interest for turbulent flows.

1.2.3 Model Reduction for Chaotic Flows

The last part of this research is the incorporation of model reduction techniques for chaotic systems. Model reduction has already been shown to be successful for linear and nonlinear chaotic systems at reducing the size of the states by projecting the states to a smaller subspace. As a result of this projection process, the states and the reduced states can be related via a set of reduced order basis functions, $\mathbf{\Phi}$. Model reduction is an important part of the present research in output-based error estimation for chaotic flows, because application of the LSS method is computationally expensive. Applying LSS to large scale problem such as LES incurs high computational costs due to the need to solve a large linear system in LSS. In order to make LSS more practical, model reduction is needed. Before applying model reduction techniques, it will be shown that it is possible to find a reduced-order model for chaotic flows and that such a model can used with LSS to inexpensively find an adjoint. As a result, model reduction techniques make output-based error estimation for chaotic flows much more efficient and practical for large scale chaotic problems, such as LES.

1.3 Thesis Overview

This dissertation presents an alternative technique to calculating chaotic adjoint for output-based error estimation. By providing this ability for computational fluid dynamics, it will be possible to improve accuracy and decrease the computational costs associated with simulations of complicated chaotic flows such as turbulent flows.

The objective of this dissertation is to improve the accuracy and reduce the costs of high fidelity chaotic simulations using **output-based error estimation** with **least square shadowing,** and **model reduction techniques**.

In achieving this goal, this dissertation makes the following contributions to the field of computational fluid dynamics:

- Provides a way to efficiently and practically calculate accurate adjoints for chaotic flows via the Hyper-Reduced-Order Modeling-Least Squares Shadowing method.
- Extends the nonlinear model reduction technique, the Least-Squares Petrov-Galerkin method, to chaotic flows and shows that it is possible to find an accurate reduced order model that can still produce accurate time-average outputs of interest over long periods of time.
- Extends the Gauss-Newton with Approximated Tensors technique to chaotic systems and shows that it is possible to apply this reduction technique without losing accuracy in chaotic simulations.
- Implements the Least Squares Shadowing method using the discontinuous Galerkin method for the 2D Navier-Stokes equations and the Kuramoto-Sivashinsky equation.
- Shows successfully the implementation of output-based error estimation for high Reynolds number laminar flow that can be used further for mesh adaptation.

1.3.1 Organization

This dissertation provides more detailed background information and results where applicable in each chapter. The organization of this work is presented in the following way:

- Chapter II presents an overview of the finite-element method, specifically the discontinuous Galerkin method and derives the spatial and temporal discretization in its weak form. Included in this chapter is an explanation on how the fourth-order derivative is discretized, which is needed for the fourth order one-dimensional Kuramoto-Sivashinsky equation.
- Chapter III introduces the traditional continuous and discrete adjoint methods and the output-based error estimation method. The chapter explains what the adjoint is and why it is important in error estimation.

- Chapter IV delves into chaos and introduces the basics of chaos theory. In addition, several tools such as ergodic theory are introduced. Three different chaotic systems are introduced here, the Lorenz Attractor, the Kuramoto-Sivashinsky Equation, and the pseudo chaotic Navier-Stokes equations. The traditional adjoint method from Chapter III is applied to each of these equations to show that the method fails for chaotic systems. This inability to find usable adjoints for chaotic systems makes it difficult to apply output-based error estimation that would make LES cheaper to simulate.

- Chapter V introduces an alternative method for calculating adjoints, the Least Squares Shadowing Method, which aims to numerically calculate accurate chaotic adjoints by relaxing the initial conditions and solving for the adjoint using a shadow trajectory that does not diverge away from the original solution. Additionally, this chapter explains how LSS can be accurate, but expensive, for very large problems.

- Chapter VI introduces model reduction via the Least-Squares Petrov-Galerkin method and hypothesizes that the LSS method can benefit from calculating the adjoint from a reduced form of the original states. Before investigating the reduced form of the LSS method, Chapter VI investigates whether or not it is possible to find an accurate, robust, reduced-order model for a chaotic system. Additionally, this chapter introduces a method that further reduces the computational cost in calculating the reduced-order model: the Gauss-Newton with Approximated Tensors method, which decreases the cost of the calculation of the residuals and the Jacobian for nonlinear problems.

- Chapter VII finally introduces the combined hyper-reduced order model-least squares shadowing approach, which aims at reducing the cost of the original LSS method without sacrificing accuracy.

CHAPTER II

Discretization Methods

The objective of output-based error estimation is to accurately estimate the effects of discretization errors on scalar quantities of interest. For the implementation of output-based error estimation for chaotic flows, the finite element method is employed to discretize the chaotic governing equations. The finite element method is chosen over the finite difference method and the finite volume method for its variational formulation and flexibility in introducing high-order approximation [2], which are favorable for output-based error estimation and mesh adaptation.

Within the class of finite element methods, the discontinuous Galerkin method (DG) has been chosen to discretize the governing equations. The motivation for DG includes: ease of implementing high-order approximations [3, 4], suitability for error estimation [5, 6, 7], and hp-adaption [8], availability of stable viscous discretizations [9, 10, 11, 12, 13], and a straightforward extension to variational space-time algorithms [14, 15, 16, 17, 18, 19, 20, 21, 22]. DG in particular allows for enrichment and optimization of the solution approximation space [2].

The development of discontinuous Galerkin methods began in 1973 by Reed and Hill in the field of neutron transport [23]. Adoption of this method in computational fluid dynamics occurred afterwards due to the need to find a method that could efficiently solve complicated convection dominated problems, such as gas dynamics, modeling of shallow water, etc. However, Cockburn et. al introduced two main issues that arose with DG [24]: 1) the occurrence of discontinuities and 2) the occurrence of complicated structures near these discontinuities. These were addressed with the development of numerical fluxes and slope limiters from finite volume methods for nonlinear hyperbolic equations. DG can be considered to be a generalization of finite volume methods due to its localized nature. Cockburn et. al revealed some advantages that DG has over finite volume and finite difference methods [24].

- DG's built-in variational framework provides the ability for high-order approximations.

- The observed order of accuracy of DG depends on the regularity of the exact solution.
- DG methods are highly parallelizable due to the discontinuous nature of the method.
- DG methods can handle complicated geometries and boundary conditions.
- DG methods can be used for mesh adaption, because refinement of the mesh does not need to take into account of the continuity restrictions found in continuous finite element methods.

DG has been applied to nonlinear hyperbolic problems, convection-diffusion systems, Navier-Stokes equations, and elliptical problems. Hyperbolic problems use numerical fluxes and elliptical problems use penalty methods to help stabilize the solution and to enforce boundary conditions

Lastly, DG has been chosen over the continuous Galerkin method (CG), because DG finds solutions that are inherently locally conservative to each element and provides more stability for convection-dominated problems, albeit at the expense of more degrees of freedom associated with DG compared to CG [25]. In DG, communication happens between elements while for CG, communication happens between nodes.

This chapter will first introduce the general mechanics of the discontinuous Galerkin finite element method. Next, it will provide derivations for scalar advection, nonlinear advection (Burgers' equation), second order diffusion, and fourth order diffusion. Additionally, three unsteady solvers will be introduced: discontinuous Galerkin in time (DGTIME), backward difference formulae (BDF), and the diagonally-implicit Runge-Kutta (DIRK) methods.

2.1 Discontinuous Galerkin Finite Element Method

The prototypical partial differential (PDE) equation can be written as

$$\boldsymbol{r}(\boldsymbol{u}) = \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \vec{\boldsymbol{F}} = \boldsymbol{0}, \tag{2.1}$$

where $\boldsymbol{u} \in \mathbb{R}^s$ is the state vector and $\boldsymbol{F}_i$ is the i^{th} component for the flux, which is dependent on the spatial dimension, d. In the following derivation, the focus will be on unsteady problems, where $\frac{\partial \boldsymbol{u}}{\partial t} \neq \boldsymbol{0}$.

For DG methods, the state, $\boldsymbol{u}$, can be approximated continuously on each element in space. The continuous approximation within each element is performed by using linear combinations of basis functions. In this work, polynomial basis functions for the DG method are chosen to approximate the solution in space. Note that continuity is not enforced between elements. Using basis functions allows for the state for each element to be approximated as

$$\boldsymbol{u}(x, y, t) \approx \sum_{m=1}^{N} \sum_{j=1}^{N_p} \boldsymbol{U}_{m,j}(t)\phi_{m,j}(x, y) \tag{2.2}$$

where

$$\begin{aligned}
N &= \text{number of elements} \\
N_p &= \text{number of basis functions on an element for order } p \text{ approximation} \\
\phi_{m,j}(x, y) &= j^{\text{th}} \text{ polynomial basis function on element } m \\
p &= \text{order of spatial basis functions} \\
\boldsymbol{U}_{m,j}(t) &= \text{time-varying coefficients on basis nodes}
\end{aligned}$$

Note that the basis functions within an element have local support on that current element. Everywhere else along space, the basis functions are zero. This can be seen in 1D in Figure 2.2. The total number of unknowns for the system is $N(p+1)$. The objective of DG is to solve for the coefficients $\boldsymbol{U}_{m,j}(t)$ for a particular time t. Unlike CG methods, DG methods solve for solution approximations that are discontinuous between elements. This adds more spatial nodes to ultimately solve for, but has the benefit of providing convective stability and simplifies more complicated refinement such as hanging-node mesh refinement and local order enrichment. These comparisons between CG and DG can be seen in Figure 2.1, where T_H denotes the set of N elements in a non-overlapping tessellation of the domain. Given the state approximation for each element in Eqn 2.2, the state coefficients are stored

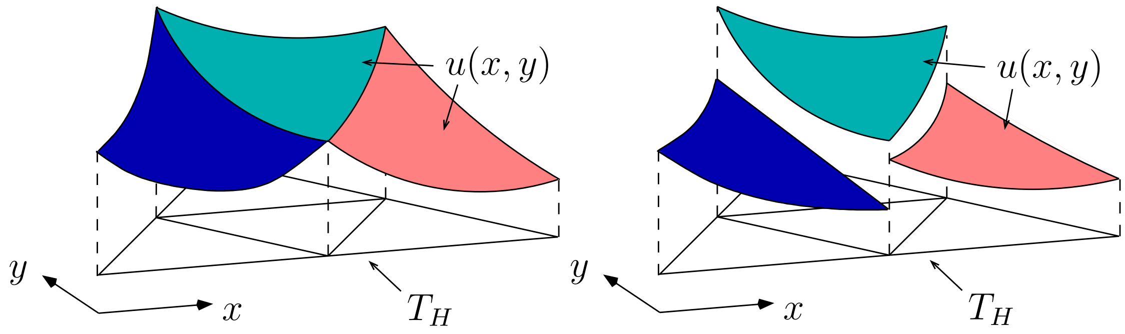


Figure 2.1: Solution approximation for continuous Galerkin and discontinuous Galerkin formulations. Note that flux terms depend on the interfaces between elements only.

in a vector for all the elements,

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_{\mathrm{elem}_1} \\ \boldsymbol{U}_{\mathrm{elem}_2} \\ \boldsymbol{U}_{\mathrm{elem}_3} \\ \boldsymbol{U}_{\mathrm{elem}_4} \\ \vdots \\ \boldsymbol{U}_{\mathrm{elem}_k} \end{bmatrix}, \quad \boldsymbol{U}_{\mathrm{elem}_k} = \begin{bmatrix} \boldsymbol{U}_0 \\ \boldsymbol{U}_1 \\ \vdots \\ \boldsymbol{U}_{p+1} \end{bmatrix}, \quad \boldsymbol{U}_{p+1} = \begin{bmatrix} \boldsymbol{U}_1 \\ \vdots \\ \boldsymbol{U}_s \end{bmatrix}. \tag{2.3}$$

Eqn 2.3 shows that each element contains a smaller vector pertaining to the collection of states at each basis node. Each element will consist of $p+1$ basis nodes. For each basis node, the state consists of another vector that contains information about the state approximation coefficients for the current element and the current basis node.

Before deriving the weak form of the DG method, important quantities at the interface between elements need to be defined. The average and normal jump quantities at the interfaces are defined as:

$$\begin{aligned} \{v\} &= \frac{1}{2}\left(v^+ + v^-\right), \qquad & \{\boldsymbol{u}\} &= \frac{1}{2}\left(\boldsymbol{u}^+ + \boldsymbol{u}^-\right) \\ [\![v]\!] &= v^+\vec{\boldsymbol{n}}^+ + v^-\vec{\boldsymbol{n}}^-, \qquad & [\boldsymbol{u}] &= \boldsymbol{u}^+\cdot\vec{\boldsymbol{n}}^+ + \boldsymbol{u}^-\cdot\vec{\boldsymbol{n}}^- \end{aligned} \tag{2.4}$$

In this work, Lagrange basis functions are used. Note that for $p > 0$, there are $p + 1$ nodes for each element. The formula for the Lagrange basis functions in 1D is

$$\phi_i = \prod_{j=1, j\neq i}^{p+1} \frac{\xi - \xi_j}{\xi_i - \xi_j}, \tag{2.5}$$

where ξ represents a point in reference space. In DG, the weak form will be written for a global element k, but will be evaluated using a reference element. The mapping between the global space (x) and the reference space (ξ) is

$$x = x_{k-\frac{1}{2}} + \frac{\xi + 1}{2}\Delta x, \quad \xi = 2\frac{x - x_{k-\frac{1}{2}}}{\Delta x} - 1, \tag{2.6}$$

where $k - \frac{1}{2}$ refers to the left side of the element and $k + \frac{1}{2}$ refers to the right side of the element. In reference space, $\xi = -1$ refers to the left side of the element, and $\xi = +1$ refers

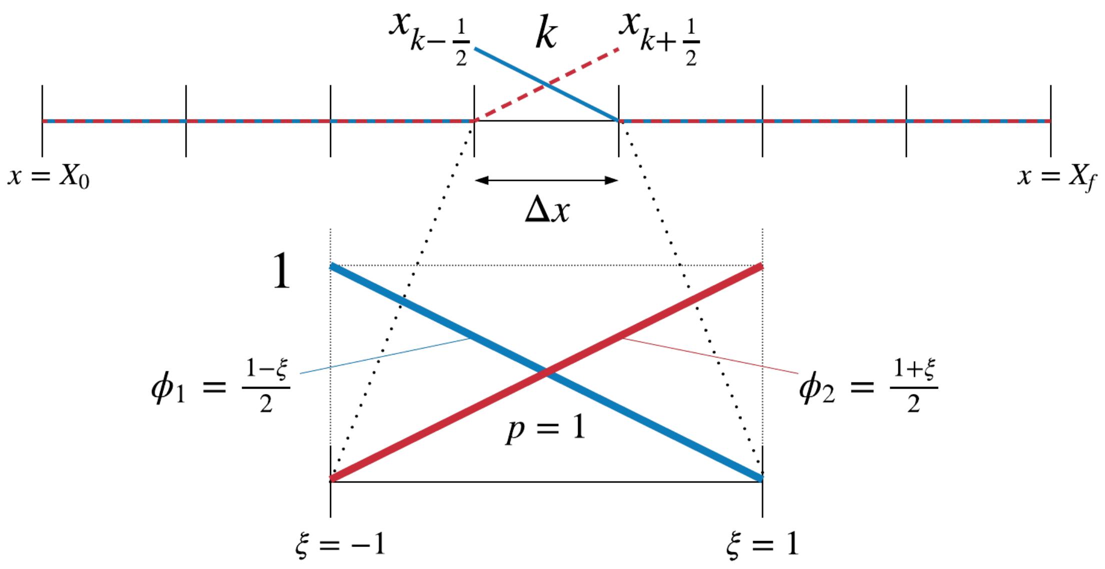


Figure 2.2: 1D mesh with $p = 1$ basis function defined for an element k.

to the right side of the element. The 1D basis functions, original space, and the reference space are shown in Figure 2.2 for $p = 1$. Again, note that the basis function is 0 everywhere but the local element, k.

2.1.1 Weak Form

The DG methods seek the solution of the state vector, u_h where h refers to a fine space mesh. The goal is to find the state solution from the semilinear residual form of the governing equation,

$$\boldsymbol{R}(\boldsymbol{u}_h, v) = 0 \quad \forall v \in \mathcal{V}_h. \tag{2.7}$$

where the v are the test functions. $\mathcal{V}_h$ is defined as

$$\mathcal{V}_h := \{v \in L^2(\Omega) |\; v|_{\Omega_e} \in \mathcal{P}^p(\Omega_e)\; \forall \Omega_e \in T_h\}. \tag{2.8}$$

Ω_e refers to the non-overlapping elements in the domain, Ω, p refers to the polynomial degree of the solution, and T_h refers to the tessellation of the fine mesh. The $L^2(\Omega)$ refers to standard Lebesgue space where $\Omega \in \mathbb{R}^2$. The states are defined such that

$$\boldsymbol{u}_h \in \boldsymbol{\mathcal{V}}_h, \tag{2.9}$$

which defines the approximation of the state solution in Eqn 2.2 when the test functions are set to the polynomial Lagrange basis functions. Thus, given the solution approximation in Eqn 2.2 and once the test functions have been assigned, Eqn 2.1 can be written in terms of basis functions in the weak form. The weak form is obtained by multiplying Eqn 2.1 by basis functions, and integrating over the domain,

$$\boldsymbol{R}_{k,i}(\boldsymbol{u}) = \int_{\Omega} \phi_{k,i} \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \vec{\boldsymbol{F}} \right] d\Omega = \boldsymbol{0}. \tag{2.10}$$

Integration only needs to be taken for the element that supports the basis function, $\phi_{k,i}$ where k again refers to the current element and i refers to the basis functions in that element. This can be simplified to

$$\boldsymbol{R}_{k,i}(\boldsymbol{u}) = \int_{\Omega_k} \phi_{k,i} \frac{\partial \boldsymbol{u}}{\partial t}\, d\Omega + \int_{\Omega_k} \phi_{k,i} \nabla \cdot \vec{\boldsymbol{F}}\, d\Omega = \boldsymbol{0}. \tag{2.11}$$

Integrating by parts for the spatial terms yields,

$$\boldsymbol{R}_{k,i}(\boldsymbol{u}) = \int_{\Omega_k} \phi_{k,i} \frac{\partial \boldsymbol{u}}{\partial t}\, d\Omega - \int_{\Omega_k} \nabla \phi_{k,i} \cdot \vec{\boldsymbol{F}} d\Omega + \int_{\partial \Omega_k} \phi_{k,i}^{+} \hat{\boldsymbol{F}}(\boldsymbol{u}^{+}, \boldsymbol{u}^{-}, \vec{\boldsymbol{n}})\, ds = \boldsymbol{0}, \tag{2.12}$$

where

$$\begin{aligned} \partial\Omega_k &= \text{boundary of element } k \\ k &= \text{current local element} \\ \phi_{k,i} &= i^{\text{th}} \text{ polynomial basis function for element } k \\ \vec{\boldsymbol{n}} &= \text{outward-pointing normal} \\ \vec{\boldsymbol{F}} &= \text{numerical flux} \\ +/- &= \text{superscript referring to interior/exterior of element } k \end{aligned}$$

The numerical flux $\widehat{\boldsymbol{F}}$ resolves the situation in DG where there are double values at the interface between element k and $k+1$, as seen in Figure 2.3. This numerical flux solution appears in finite volume methods as well. Thus, techniques already exist for computing the flux and will be explained in the next sections.

Substituting Eqn 2.2 into Eqn 2.12 gives the linear system

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{M} \frac{d\boldsymbol{U}}{dt} + \boldsymbol{R}_s = \boldsymbol{0}, \tag{2.13}$$

where $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ is the mass matrix and $\boldsymbol{R}_s$ contains the spatial residual terms. The mass matrix $\boldsymbol{M}$ is an element-wise block diagonal matrix where each smaller square matrix of

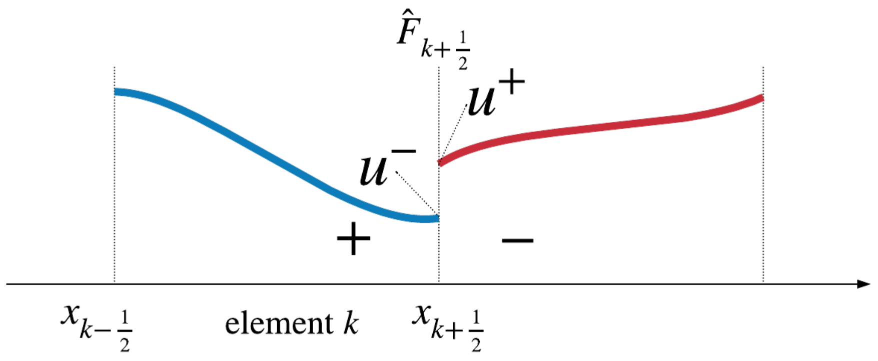


Figure 2.3: Flux definition for element k.

element k, $\boldsymbol{M}_{ij}$, is defined mathematically as

$$\boldsymbol{M}_{i,j} = \boldsymbol{I}_s \int_{\Omega} \phi_i \phi_j d\Omega, \tag{2.14}$$

where $\boldsymbol{I}_s \in \mathbb{R}^{s \times s}$ is the state identity matrix and i, j refers to the index of of the global degree of freedom where $0 \leq i, j < N$. Note that the mass matrix for each element, k, is the same, because ϕ_i is chosen to be the same basis for each element element. If the problem is linear then $\boldsymbol{R}_s = \boldsymbol{A}\boldsymbol{U}$ and $\boldsymbol{A}$ is the stiffness matrix. The following sections will go over how $\vec{\boldsymbol{F}}$ is discretized depending on what $\vec{\boldsymbol{F}}$ is. The $\vec{\boldsymbol{F}}$ term in the second integral of Eqn 2.12 is typically evaluated using quadrature.

2.1.2 Linear Scalar Advection

The linear advection term is characterized by its advection speed $\vec{V}$ and is written as

$$\vec{F} = \vec{V} u. \tag{2.15}$$

Substituting Eqn 2.15 into the the weak form, Eqn 2.12, gives

$$R_{k,i}(u) = \int_{\Omega_k} \phi_{k,i} \frac{\partial u}{\partial t} \, d\Omega + \int_{\Omega_k} \vec{V} \cdot \nabla \phi_{k,i} u \, d\Omega - \int_{\partial \Omega_k} \vec{V} \cdot \hat{u} \phi_{k,i} \, ds = 0. \tag{2.16}$$

The flux term, $\hat{F}(u^+, u^-, \vec{n})$, for advection is computed using the upwinding method with respect to the normal velocity . The upwinding method gives,

$$\hat{F}(u^+, u^-, \vec{n}) = \vec{V} \cdot \hat{u} = \frac{1}{2} \left(\vec{V} \cdot u^- + \vec{V} \cdot u^+ \right) - \frac{1}{2} |\vec{V}| \left(u^+ - u^- \right). \tag{2.17}$$

In 1D, Eqn 2.15 can be written with a scalar velocity as

$$\vec{F} = au. \tag{2.18}$$

Using Eqn 2.18 in the 1D version of the weak form, Eqn 2.12, gives

$$R_{k,i}(u) = \int_{\Omega_k} \phi_{k,i}\frac{\partial u}{\partial t}\, d\Omega - a\int_{\Omega_k} \frac{\partial \phi_{k,i}}{\partial x} u\, d\Omega + [\phi_{k,i} a\hat{u}]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} = 0. \tag{2.19}$$

Substituting Eqn 2.2, which is the solution approximating u gives the discretized form of the spatial part of the total residual,

$$R_{k,i}(U) = \underbrace{\sum_{j=1}^{p+1}\left[\int_{\Omega_k} \phi_{k,i}\phi_{k,j}\right]}_{\boldsymbol{M}} \frac{dU_{k,j}}{dt} - \underbrace{\sum_{j=1}^{p+1}\left[\int_{\Omega_k} \frac{\partial \phi_{k,i}}{\partial x}\phi_{k,j} dx\right] U_{k,j} + [\phi_{k,i} a\hat{u}]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}}}_{\boldsymbol{AU}} = 0. \tag{2.20}$$

Again, this discretized residual is in the form of Eqn 2.13.

2.1.3 Nonlinear Scalar Burgers' Equation: Scalar Conservation Law

The nonlinear scalar Burgers' term is written as

$$\vec{F} = \frac{1}{2}u^2 \tag{2.21}$$

Substituting Eqn 2.21 into the the weak form, Eqn 2.12, gives

$$R_{k,i}\left(u\right) = \int_{\Omega_k} \phi_{k,i}\frac{\partial u}{\partial t}\, d\Omega + \frac{1}{2}\int_{\Omega_k} \nabla\phi_{k,i} u^2\, d\Omega - \int_{\partial\Omega_k} \frac{1}{2}\hat{u}^2\phi_{k,i}\, ds = 0. \tag{2.22}$$

The flux term, $\hat{F}(u^+, u^-, \vec{n})$, for Burgers' equation, $\frac{1}{2}\widehat{u}^2$, is solved using the Godunov method [26, 27], which is an entropy-satisfying monotone scheme that generalizes the upwind method. The Godunov method computes the interface flux by solving the interface problem, also known as the Riemann problem exactly. It assumes a piece wise constant solution over each cell and is first order accurate. For this section, u_L will refer to the state to the left of the interface and u_R will refer to the state to the right of the interface. Applying the Godunov method to $\hat{F} = f(u)$ gives,

$$\hat{F} = \begin{cases} \min\limits_{u\in[u_L,u_R]} f(u), & u_L \le u_R \\ \max\limits_{u\in[u_R,u_L]} f(u), & u_L > u_R \end{cases} \tag{2.23}$$

Eqn 2.23 is valid for any scalar conservation law. For a concave or convex function, Eqn 2.23 can be expanded and written out as

$$\hat{F} = \begin{cases} f(u_R), & \text{if} \quad \dfrac{f(u_L)}{\partial u}, \dfrac{f(u_R)}{\partial u} \leq 0 \\ f(u_L), & \text{if} \quad \dfrac{f(u_L)}{\partial u}, \dfrac{f(u_R)}{\partial u} \geq 0 \\ f(u_s), & \text{if} \quad \dfrac{f(u_L)}{\partial u} < 0 < \dfrac{f(u_R)}{\partial u} \qquad \text{(expansion)} \\ f(u_L), & \text{if} \quad \dfrac{f(u_L)}{\partial u} \geq 0 \geq \dfrac{f(u_R)}{\partial u} \qquad \text{and} \quad \dfrac{[f]}{[u]} > 0 \\ f(u_R), & \text{if} \quad \dfrac{f(u_L)}{\partial u} \geq 0 \geq \dfrac{f(u_R)}{\partial u} \qquad \text{and} \quad \dfrac{[f]}{[u]} < 0, \end{cases} \tag{2.24}$$

where u_s is called the sonic point and refers to value of u for which the characteristic speed is zero.

Note that for Burgers' equation, the flux function is convex. Evaluation of the Godunov flux applied to Burgers' equation,

$$\widehat{F}_B = \begin{cases} \frac{1}{2}u_R^2, & \text{if} \quad u_L, u_R \leq 0 \\ \frac{1}{2}u_L^2, & \text{if} \quad u_L, u_R \geq 0 \\ 0, & \text{if} \quad u_L < 0 < u_R \qquad \text{(expansion)} \\ \frac{1}{2}u_L^2, & \text{if} \quad u_L \geq 0 \geq u_R \qquad \text{and} \quad \frac{1}{2}(u_R + u_L) > 0 \\ \frac{1}{2}u_R^2, & \text{if} \quad u_L \geq 0 \geq u_R \qquad \text{and} \quad \frac{1}{2}(u_R + u_L) < 0 \end{cases} \tag{2.25}$$

Eqn 2.22 can be written as Eqn 2.13 once the Godunov flux is applied.

2.1.4 Linear Scalar Second-Order Diffusion

The scalar diffusion term, which is a second order derivative, is written as

$$\vec{F} = \nabla u, \tag{2.26}$$

Instead of substituting Eqn 2.26 into the the weak form from Eqn 2.12, the governing equation, Eqn 2.1, is first converted to a first-order system

$$\begin{aligned} \nabla u &= \vec{\sigma}, \\ \frac{\partial u}{\partial t} - \nabla \cdot \vec{\sigma} &= 0. \end{aligned} \tag{2.27}$$

To find the weak form, first assume the same order p for both system in Eqn 2.27. Then dot the first system by $\vec{\tau}$ and multiply the second system by ϕ. Lastly, integrate both equations over element k to find,

$$\begin{aligned}\int_{\Omega_k} \nabla u \cdot \vec{\tau}\, d\Omega - \int_{\Omega_k} \vec{\sigma} \cdot \vec{\tau}\, d\Omega = 0,\\ \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega - \int_{\Omega_k} \nabla \cdot \vec{\sigma}\phi\, d\Omega = 0.\end{aligned} \tag{2.28}$$

Integrating by parts and using $+$ and $-$ once again to designate the interior and exterior element k gives,

$$\begin{aligned}-\int_{\Omega_k} u \nabla \cdot \vec{\tau} d\Omega + \int_{\partial\Omega_k} \widehat{u}\vec{\tau} \cdot \vec{n}\, ds - \int_{\Omega_k} \vec{\sigma} \cdot \tau\, d\Omega = 0,\\ \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega + \int_{\Omega_k} \vec{\sigma} \cdot \nabla\phi^{+}\, d\Omega - \int_{\partial\Omega_k} \phi^{+}\widehat{\sigma}\, ds = 0.\end{aligned} \tag{2.29}$$

Choosing $\vec{\tau} = \nabla\phi^{+}$ in Eqn 2.29,

$$\begin{aligned}-\int_{\Omega_k} u \nabla \cdot (\nabla\phi)\, d\Omega + \int_{\partial\Omega_k} \widehat{u}\nabla\phi^{+} \cdot \vec{n}\, ds - \int_{\Omega_k} \vec{\sigma} \cdot \nabla\phi^{+}\, d\Omega = 0,\\ \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega + \int_{\Omega_k} \vec{\sigma} \cdot \nabla\phi^{+}\, d\Omega - \int_{\partial\Omega_k} \phi^{+}\widehat{\sigma}\, ds = 0.\end{aligned} \tag{2.30}$$

Substituting the first equation into the second equation, one obtains

$$R_{k,i}(u) = \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega - \int_{\Omega_k} u \nabla \cdot (\nabla\phi)\, d\Omega + \int_{\partial\Omega_k} \widehat{u}\nabla\phi^{+} \cdot \vec{n}\, ds - \int_{\partial\Omega_k} \phi^{+}\widehat{\sigma}\, ds = 0. \tag{2.31}$$

Integrating the first term by parts gives,

$$\begin{aligned}R_{k,i}(u) = \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega + \int_{\Omega_k} \nabla u \cdot \nabla\phi\, d\Omega\\ - \int_{\partial\Omega_k} u^{+}\nabla\phi^{+} \cdot \vec{n}\, ds + \int_{\partial\Omega_k} \hat{u}\nabla\phi^{+} \cdot \vec{n}\, ds - \int_{\partial\Omega_k} \phi^{+}\widehat{\sigma}\, ds = 0\end{aligned} \tag{2.32}$$

Rearrange Eqn 2.32 to find the final form

$$\begin{aligned}R_{k,i}(u) = \int_{\Omega_k} \frac{\partial u}{\partial t}\phi\, d\Omega + \int_{\Omega_k} \nabla u \cdot \nabla\phi\, d\Omega\\ - \int_{\partial\Omega_k} \left(u^{+} - \hat{u}\right)\nabla\phi^{+} \cdot \vec{n}\, ds - \int_{\partial\Omega_k} \phi^{+}\, \widehat{\sigma}\, ds = 0,\end{aligned} \tag{2.33}$$

where

$$\hat{u} = \{u\} \tag{2.34}$$

$$\widehat{\sigma} = \left(\{\nabla u\} - \eta_f \{\vec{\delta}_f\}\right) \cdot \vec{n} \tag{2.35}$$

$\vec{\delta}_f$ represents a vector field corresponding with one face that is supported by two elements adjacent to the face. η_f is the non dimensional stabilization factor associated with a face f and should be set such that its value is greater than or equal to the maximum number of faces on the two adjacent elements [25]. In this research, the non dimensional stabilization factor is set to $\eta_f = 10$. Once in the weak form, the interior penalty (IP) method is chosen to calculate the fluxes. Known as the the poor man's DG diffusion flux, the IP method is defined as

$$\vec{\delta}_f^+ = \frac{1}{h^+}[\![u]\!], \quad \vec{\delta}_f^- = \frac{1}{h^-}[\![u]\!], \tag{2.36}$$

where h^+/h^- are the lengths of the elements adjacent to the face f [25]. In 1D, the weak form for the diffusion equation becomes Note that the interior penalty method is easy to implement, but lacks strict bounds for the stabilization factor, η_f.

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0 \tag{2.37}$$

$$\int_{\Omega_k} \phi_{k,i} \frac{\partial u}{\partial t} dx + \int_{\Omega_k} \frac{\partial \phi}{\partial x}\frac{\partial u}{\partial x} dx - \left[\left(u^+ - \hat{u}\right)\frac{\partial \phi_{k,i}}{\partial x}\right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \left[\phi_{k,i}^+ \hat{\sigma}\right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} = 0 \tag{2.38}$$

$$\hat{\sigma} = \frac{1}{2}\left(\frac{\partial u^+}{\partial x} + \frac{\partial u^-}{\partial x}\right) - \eta_f \frac{1}{2}\left(\delta_f^+ + \delta_f^-\right) \tag{2.39}$$

2.1.5 Linear Scalar Fourth-Order Diffusion

Some partial differential equations such as Kuramoto-Sivashinky require the discretization of a fourth-order derivative. These types of problems can be solved with finite-element methods, specifically discontinuous Galerkin methods. Discontinuous Galerkin methods allow for flexible and efficient discretizations of these more complicated problems. The purpose of this section is to go over how to properly discretize the fourth-order term and to define the fluxes needed in discontinuous Galerkin methods. The method is referred to as the IPDG FEM method and was devised by Georgoulis et. al [28]. The flux term for the fourth-order derivative term is written as

$$\vec{F} = \nabla^3 u \tag{2.40}$$

where the divergence of the flux gives the fourth-order derivative. The weak form of Eqn 2.12 applied to Eqn 2.40 gives

$$R_{k,i}(u) = \int_{\Omega_k} \frac{\partial u}{\partial t}\phi \; d\Omega - \int_{\Omega_k} \nabla\phi_{k,i} \cdot \nabla^3 u \; d\Omega + \int_{\partial\Omega_k} \phi_{k,i}\nabla^3 u \cdot \vec{n} \; ds = 0. \tag{2.41}$$

Integrating by parts one again gives,

$$\begin{aligned} R_{k,i}(u) = \int_{\Omega_k} \frac{\partial u}{\partial t}\phi \; d\Omega - \int_{\Omega_k} \nabla^2\phi_{k,i} \cdot \nabla^2 u \; d\Omega \\ + \int_{\partial\Omega_k} \phi_{k,i}\nabla^3 u \cdot \vec{n} \; ds - \int_{\partial\Omega_k} \nabla\phi_{k,i}\nabla^2 u \cdot \vec{n} \; ds = 0. \end{aligned} \tag{2.42}$$

However, Eqn 2.42 is not helpful and requires an additional step before the test functions v can be assigned to. Georgoulis et al. provides an alternate derivation for the discretization of the four-order derivative by first introducing a lifting operator [28],

$$\mathcal{L}(v) : \mathcal{W} := \mathcal{V}_h + \mathcal{H}_0(\Omega)^2 \to \mathcal{V}_h, \tag{2.43}$$

where H_0^2 refers to the standard Hilbertian Sobolev space of index $i = 2$ of real valued functions defined on domain Ω. The Sobolev space specifically is characterized by a norm that is a combination of L^2 norms of the function f itself and its weak derivatives up to a given order, where the weak derivative is defined as not necessarily differentiable but integrable. The Sobolev space becomes a Hilbertian space as well when $i = 2$ and is useful due to its relationship with Fourier series. Again, h refers to the fine mesh. Lifting the test function v to the Hilbertian Sobolev space gives,

$$\int_{\Omega} \mathcal{L}(v) w \; d\Omega = \int_{\Gamma} \left([\![v]\!] \cdot \{\nabla w\} - \{w\} [\nabla v] \right) \; ds, \quad \forall w \in \mathcal{V}_h. \tag{2.44}$$

Note that the states and test functions will be integrated over the entire domain Ω instead of the element Ω_k and the set of faces, Γ instead of the local faces of an element. To solve the fourth-order derivative, a version of the interior penalty method (IPDG) is use. The IPDG method finds the states $u_h \in \mathcal{W}_h$ such that the bilinear form, $B(u_h, v) = 0 \; \forall v \in \mathcal{V}$, where $B : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$. The bilinear form is given by

$$\begin{aligned} B(w,v) = \int_{\Omega} \left(\nabla_h^2 w \nabla_h^2 v + \mathcal{L}(w)\nabla_h^2 v + \nabla^2 w \mathcal{L}(v) \right) d\Omega + \int_{\Gamma} \left(\sigma [\![w]\!] \cdot [\![v]\!] + \tau [\nabla w] [\nabla v] \right) \; ds \\ \forall w, v \in \mathcal{W} \end{aligned} \tag{2.45}$$

When $w, v \in \mathcal{W}$, the formulation is inconsistent; however, the formulation is consistent when $u_h, v \in \mathcal{V}$. Hence, after setting the test functions v to the polynomial Lagrange basis functions, the bilinear form Eqn 2.45 written with the unsteady term for a general mesh becomes

$$\begin{aligned} R_{k,i}(u) = \int_\Omega \frac{\partial u}{\partial t}\phi \, d\Omega + \int_\Omega \nabla^2 u \nabla^2 \phi \, d\Omega + \int_\Gamma \big[\nabla^3 u \cdot [\![\phi]\!] + \{\nabla^3 \phi\} + \{\phi\} \cdot [\![u]\!] \\ - \{\nabla^2 u\}[\nabla \phi] - \{\nabla^2 \phi\}[\nabla u] + \sigma [\![u]\!] [\![\phi]\!] + \tau [\nabla u][\nabla \phi]\big] \, ds = 0 \end{aligned} \tag{2.46}$$

where

$$\sigma = \frac{1}{\{h\}^3}\sigma_0 \qquad \tau = \frac{1}{\{h\}^3}\tau_0. \tag{2.47}$$

h is the length of diameter of the element k. σ and τ are stabilization terms, similar to the stabilization terms found for the IPDG method used to solve for the diffusion terms, where σ_0 and τ_0 must be greater than zero and be sufficiently large [28].

2.2 Unsteady Solvers

Several unsteady solvers are used in this thesis: diagonally implicit Runge-Kutta methods (DIRK), backward differentiation formulae (BDF), and the discontinuous Galerkin in time (DGTIME) method. Extensive literature exist for BDF and DIRK schemes, which have been used for the Kuramoto-Sivashinsky equation and the Navier-Stokes equations. In this section, DG in time will be introduced as well, as this is used to solve the Lorenz Attractor. Note that only the temporal discretization errors are of interest for the Lorenz Attractor, and that only the spatial discretization errors are of interest for the Kuramoto-Sivashinsky equation and the Navier-Stokes equations.

2.2.1 Discontinuous Galerkin in Time

To find the unsteady solution using a discontinuous Galerkin finite element temporal discretization (DGTIME), consider the discrete system of Eqn 2.13. Like in space, the temporal domain is split into k, temporal elements, where each temporal element is denoted by $\left[t_{k-\frac{1}{2}}, t_{k+\frac{1}{2}}\right]$. In space, the approximation order is designated as p. In time, the aproximation order is referred to as r. Given r, the solution over a temporal element can be approximated as

$$\boldsymbol{U}(t) = \sum_{j=0}^{r} \boldsymbol{U}^j \phi^j(t). \tag{2.48}$$

The goal of DG in time is to solve for the coefficients of this approximation. Like in space, one multiplies Eqn 2.13 by a temporal basis function $\phi(t)^i$ and integrate by parts to find a temporal weak form,

$$\boldsymbol{R}^{k,i} = -\int_{t_{k-\frac{1}{2}}}^{t_{k+\frac{1}{2}}} \boldsymbol{M}\frac{d\phi^i}{dt}\boldsymbol{U}(t)dt + \left[\boldsymbol{M}\boldsymbol{U}(t)\phi^i(t)\right]_{t_{k-\frac{1}{2}}}^{t_{k+\frac{1}{2}}} + \int_{t_{k-\frac{1}{2}}}^{t_{k+\frac{1}{2}}} \phi^i(t)\boldsymbol{R}_s(\boldsymbol{U}(t))dt = \boldsymbol{0}. \tag{2.49}$$

Once the total residual has been discretized in time for a given unsteady discretization, the states can be solved. For this thesis, since the main focus is on nonlinear chaotic systems, $\boldsymbol{R}$ will be nonlinear, requiring the use of a Newton-Raphson method to solve Eqn 2.49. With this method the linear update for the i^{th} element is given by

$$\frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}^j}\boldsymbol{\Delta U}^j = -\boldsymbol{R}^i(\boldsymbol{U}^j). \tag{2.50}$$

where j refers to the j^{th} state coefficient from the temporal state approximation in Eqn 2.48. The $n+1$ Newton update to the solution for the i^{th} element is

$$\boldsymbol{U}^j_{n+1} = \boldsymbol{U}^j_n + \Delta\boldsymbol{U}^j. \tag{2.51}$$

2.2.2 Backward Differentiation Formula Method

The next time marching method of interest is the n^{th} order implicit backward difference formula (BDF), which can be written as

$$\boldsymbol{R}\left(\boldsymbol{U}^{i+1}\right) = \frac{\boldsymbol{M}}{\Delta t}\left(c_0\boldsymbol{U}^{i+1} + c_1\boldsymbol{U}^i + c_2\boldsymbol{U}^{i-1} + ...\right) + \boldsymbol{R}_s\left(\boldsymbol{U}^{i+1}\right) = \boldsymbol{0}. \tag{2.52}$$

The $n+1$ coefficients for the n^{th} BDF scheme are presented in Table 2.1.

Table 2.1: Coefficients for n^{th} order BDF, up until BDF3

	c_0	c_1	c_2	c_3
BDF1	1	-1		
BDF2	$\frac{3}{2}$	-2	$\frac{1}{2}$	
BDF3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$

When using a BDF scheme whose order is greater than 1, BDF1 is used for the first time step, BDF2 is used for the second time step, etc. until enough previous solutions exist to perform the n^{th} BDF scheme. Note that BDF1 and BDF2 is stable;however, BDF3 and higher are unstable and do not give the correct convergence rate. Thus the BDF1 and BDF2 time marching scheme is used to solve the Kuramoto-Sivashinsky equation. Since Kuramoto-Sivashinsky is nonlinear, the Newton-Raphson method is used once again to solve Eqn 2.52. The linear update is given by,

$$\frac{\partial \boldsymbol{R}\left(\boldsymbol{U}^{i+1}\right)}{\partial \boldsymbol{U}^{i+1}} \boldsymbol{\Delta U}^{i+1} = -\boldsymbol{R}\left(\boldsymbol{U}^{i+1}\right). \tag{2.53}$$

The $n+1$ Newton update to the solution for the $i^{\text{th}}+1$ temporal node is

$$\boldsymbol{U}_{k+1}^{i+1} = \boldsymbol{U}_{k}^{i+1} + \Delta \boldsymbol{U}^{i+1}. \tag{2.54}$$

2.2.3 Diagonally-Implicit Runge Kutta Method

The last time marching method of interest is the n_{stage} diagonally-implicit Runge-Kutta method (DIRK), where each time step requires n_{stage} nonlinear solves [29]. This time scheme will be used to solve the Navier-Stokes equations. The DIRK method is characterized by a lower triangular matrix with at least one nonzero diagonal entry and is sometimes referred to as semi-implicit or semi-explicit Runge-Kutta method [30]. This structure allows for each stage to be calculated individually rather than simultaneously [30]. The total residual for the n_{stage} DIRK scheme is given in Algorithm 2.1.

Algorithm 2.1 n_{stage} Diagonally-Implicit Runge-Kutta

```
Input: $t^n, \boldsymbol{U}^n$
Output: $t^{n+1}, \boldsymbol{U}^{n+1}$
1: $t^0 = t^n$
2: $\boldsymbol{U}^0 = \boldsymbol{U}^n$
3: for $i = 0, \dots n_{\text{stage}} - 1$ do
4:     $t_{i+1} = t^0 + b_{i+1}\Delta t$
5:     $\boldsymbol{S}_{i+1} = -\frac{\boldsymbol{M}}{\Delta t}\boldsymbol{U}^0 + \sum_{j=0}^{i} a_{ij}\boldsymbol{R}_s(\boldsymbol{U}^j, t_j)$
6:     Solve: $\boldsymbol{R}(\boldsymbol{U}^{i+1}) = \frac{\boldsymbol{M}}{\Delta t}\boldsymbol{U}^{i+1} + a_{ii}\boldsymbol{R}_s(\boldsymbol{U}^{i+1}, t_{i+1}) + \boldsymbol{S}_{i+1} = 0$
7: end for
8: end for
9: $\boldsymbol{U}^{n+1} = \boldsymbol{U}^{n_{\text{stage}}}$
```

To solve the Navier Stokes equations, the 3rd order DIRK3 scheme is used, where the a_{ij} coefficients are

$$a_{ij} = \begin{bmatrix} 0.435866521508459 & 0 & 0 \\ 0.2820667393 & 0.435866521508459 & 0.0 \\ 1.2084966492 & -0.6443631707 & 0.435866521508549, \end{bmatrix}. \tag{2.55}$$

and the b_i coefficients are

$$b_i = \begin{bmatrix} 0.435866521508459 \\ 0.7179332608 \\ 1.0 \end{bmatrix}. \tag{2.56}$$

Note that first-order one stage DIRK scheme is the same the first order implicit backward difference formula, BDF1.

2.3 Summary

To perform error estimation for chaotic systems, the governing equations are discretized and solved using discontinuous Galerkin methods, which are variations of finite-element methods. Localized to an element, DG methods have several advantages that make them practical for output-based error estimation and eventually mesh adaption of chaotic systems: built in variational framework for high-order solutions, highly parallelizable, and not dependent on continuity restrictions for mesh adaptation. DG will be used to discretize the governing equations of chaotic systems where the discretized residuals will be used to define the continuous and discrete adjoints, needed for error estimation. This chapter presented flux evaluation techniques borrowed from finite volume methods that will be needed when solving the Kuramoto-Sivashinsky equation and the Navier-Stokes equations. In addition, unsteady methods were presented to discretize the unsteady term of the governing equations. The next chapter will go over the fundamentals of output-based error estimation and the overall adjoint method.

CHAPTER III

Output-Based Error Estimation

Even with increasing computational power, the accuracy of computational fluid dynamic simulations does not necessarily improve. Estimation of discretization errors via output-based adaptive methods is essential for improving the accuracy and efficiency of steady and unsteady simulations. By providing numerical error bars, these methods allow for increased confidence in the accuracy of simulations. Output-based adaptive methods are fairly mature for steady problems [31, 32, 33, 34, 35, 36]; however, their application to unsteady simulations has been more limited [37, 38, 19, 20, 22, 39, 40] due to challenges in implementation and computational cost associated with fine-space adjoint solutions. Success of these methods for unsteady problems include: temporal-only error estimation and adaptation [37, 38]; spatial-only error estimation and adaptation [19, 41, 42]; combined temporal and spatial mesh refinement with a static geometry and mesh [22, 43, 40]; combined temporal and dynamic spatial refinement on static geometries [44, 39]; combined temporal and dynamic-order spatial refinement on deformable domains [45]. Additional work has been done for space-time discontinuous Galerkin (DG) and hybridized discontinuous Galerkin (HDG) [46, 47, 48, 49, 50] finite element discretizations using time slabs and an approximate space-time solver [51, 22, 29].

Output-based error estimation methods use adjoint solutions, which relate to residual perturbations to output changes. Adjoint solutions are calculated for unsteady problems by reverse time-integration and linearization about the primal solution. Figure 3.1 shows a schematic of the adaptive process, where the unsteady simulation is run multiple times, starting with a coarse space-time mesh that is successively improved.

However, the usual unsteady adjoint calculation fails to produce useful adjoints for chaotic flows, preventing the successful use of output-based error estimation and mesh adaptation for turbulent simulations like LES. This hindrance is due to the output-based methods' reliance on adjoints. The traditional unsteady adjoint calculation fails to produce useful adjoints for chaotic flows, due to the high sensitivity of chaotic problems to initial conditions. The inability to calculate useful adjoints for output-based error estimations can be attributed to the linear nature of the adjoint solution. Applying the linearized adjoint equation, which will be introduced in this section, fails as the chaotic adjoint increases exponentially backwards in time. Before delving into chaos theory in order to find a solution to calculate adjoints for chaotic flows, details of how the adjoint calculation is found traditionally for output-based methods will be shown in this chapter. The adjoint and the duality

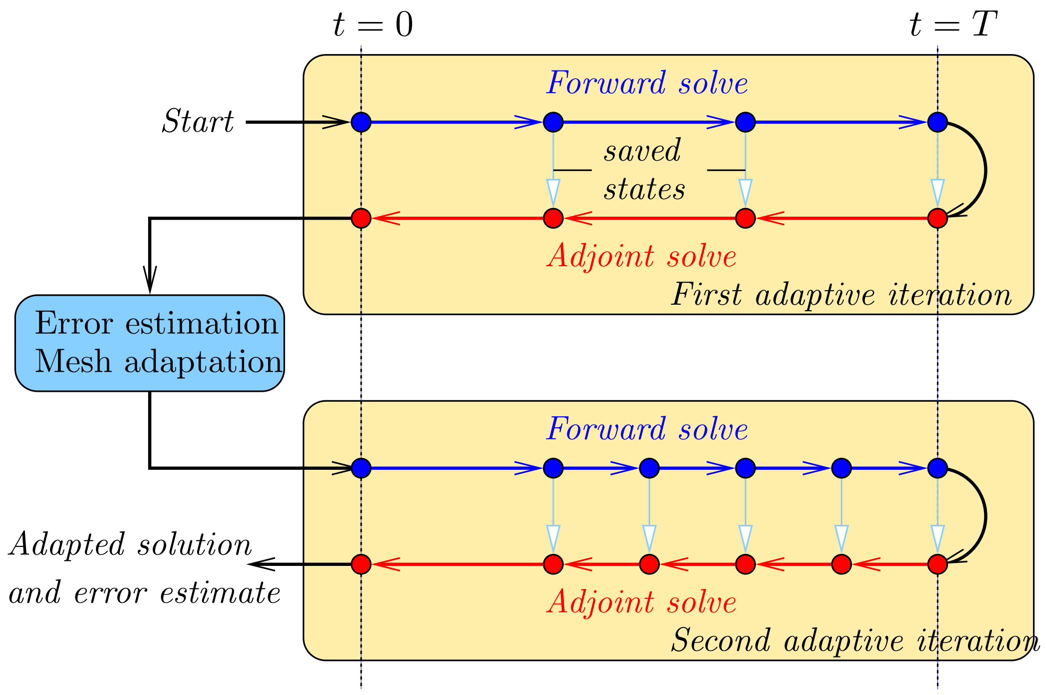


Figure 3.1: Unsteady adaptive iterations: Schematic of an adaptive primal and adjoint solution procedure for output-based unsteady simulations.

theory of a governing equation will be introduced in order to show the relationship between the primal solution and the adjoint solution. Next, the traditional adjoint method for steady and unsteady flows will be introduced. Adjoint methods are usually categorized into discrete adjoint and continuous adjoint methods [52]. One important difference between the two methods is that the discrete adjoint is first discretized before variations are taken, while the continuous adjoint method is derived by first taking variations and then discretizing the adjoint PDE. Lastly, the output-based error estimation mechanics and the adjoint-weighted residual method will be introduced.

3.1 Duality and Analytical Adjoint Formulation

Before formulating the discrete and continuous adjoint equations, it will be shown that for any governing equation, a dual equation, or the adjoint equation, can be found, which provides the linear sensitives of any objective function of interest. From this dual equation formulation, the error estimation will be defined as well. The technique presented here is explained in works by Giles et. al [53].

The governing equations can be written in terms of the state as

$$r(u) = 0, \tag{3.1}$$

where perturbing the geometry or the mesh for example leads to changes in the residual- a perturbed residual. If the residual is linear, the governing equations become,

$$Lu = f. \tag{3.2}$$

where L is a linear differential operator. If Eqn. 3.1 is nonlinear, the governing equations can be linearized to obtain the form of Eqn. 3.2,

$$r(u + \delta u) = r(u) + \frac{\partial r}{\partial u}\delta u + \mathcal{O}(\delta u^2) = 0. \tag{3.3}$$

Assuming small perturbations and rearranging the terms from the linearization gives

$$\underbrace{\frac{\partial r}{\partial u}}_{L}\underbrace{\delta u}_{u} = \underbrace{-r(u)}_{f}, \tag{3.4}$$

where $f = -r(u)$, u is the resulting linearization perturbation, and the sensitivity of r is L. The nonlinear output can be defined as

$$J(u) = \int_{\Omega} j(u) d\Omega. \tag{3.5}$$

If the output is nonlinear, the linearized output $J(u)$ in Eqn. 3.5 can be defined as

$$J(u) = \langle g, u \rangle \tag{3.6}$$

where $\langle \cdot\,, \cdot \rangle$ is the integral inner product over the domain Ω. Formulating the Lagrangian equation in terms of a linear output gives,

$$\mathcal{L} = \langle g, u \rangle - \langle \psi, Lu - f \rangle. \tag{3.7}$$

Where ψ is the Lagrange multiplier function associated with the PDE constraint. Taking variations of u, $u \to u + \delta u$ and requiring the Lagrangian to be stationary with respect to permissible δu gives

$$L[u]^* \psi = J'[u], \tag{3.8}$$

where the prime denotes the Fréchét linearization with respect to the arguments in the brackets. This can be simplified to

$$L^* \psi = g, \tag{3.9}$$

which is called the adjoint form, or the dual form. THe linear operator L^* is defined by the adjoint identity,

$$\langle \psi, Lu \rangle = \langle L^* \psi, u \rangle \quad \forall \psi, u \in \mathcal{V}. \tag{3.10}$$

Expanding Eqn. 3.7 gives

$$\mathcal{L} = \langle g, u \rangle - \langle \psi, Lu \rangle + \langle \psi, f \rangle, \tag{3.11}$$

where the adjoint identity can be substituted in to find

$$\begin{aligned} \mathcal{L} &= \langle g, u \rangle - \langle L^* \psi, u \rangle + \langle \psi, f \rangle \\ &= \langle v, f \rangle - \langle L^* v - g, u \rangle. \end{aligned} \tag{3.12}$$

Thus, the linear/linearized output for the adjoint form by definition of the Lagrangian formulation is

$$J(\psi) = \langle \psi, f \rangle, \tag{3.13}$$

which is the same as the linearized output from Eqn. 3.6. which shows how the adjoint operator, L^*, is related to the linearized operator, L. Note that the outputs of Eqn. 3.6 and Eqn. 3.13 are equivalent as long as the $\mathcal{L}^*$ is the adjoint operator of the original operator $\mathcal{L}$. Note that $\mathcal{V}$ is the function space for the state and adjoitn approximation.

This duality analysis can also be used to show that error estimation is possible. This form is not typically used but stresses the duality relationship that exists between the the original operator and the adjoint operator. Let

$$\delta r(u) = r(u + \delta u) - r(u), \tag{3.14}$$

where

$$r(u + \delta u) = r(u) + \frac{\partial r}{\partial u} \delta u + \mathcal{O}(\delta u^2). \tag{3.15}$$

Assuming small perturbations,

$$\delta r(u) = \frac{\partial r}{\partial u} \delta u. \tag{3.16}$$

Note that $f = \delta r$, $L = \frac{\partial r}{\partial u}$ and $u = \delta u$ in the form of Eqn. 3.2. From the duality analysis, it was seen that the following is true for the outputs,

$$\langle g, u \rangle = \langle \psi, f \rangle. \tag{3.17}$$

Thus, as done before to find the adjoint equation, the output error is defined as,

$$\delta J = \langle \psi, \delta r(u) \rangle. \tag{3.18}$$

This formulation is referred to the adjoint-weighted residual method used to find the output discretized error estimate, which is based on the perturbed residual; however, it gives some insight on the relationship between the governing equation and its dual form. In the next section, the adjoint will be defined and the discrete analysis analysis will be introduced, which is based on an already discretized primal solution.

3.2 Discrete Adjoint Sensitivity Analysis

In contrast to the way the adjoint was found in section 3.1, the adjoint can be computed from the discretized primal solution as well without having to discretize the adjoint solution once it is found. Thus the discretized residual that is used to derive the discrete adjoint is

$$\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\mu}) = \boldsymbol{0}, \tag{3.19}$$

where $\boldsymbol{U}$ represents the discretized states from the primal solution and $\boldsymbol{\mu}$ represents the inputs of the system. The scalar output of interest is

$$J = J\left(\boldsymbol{U}\right). \tag{3.20}$$

This output is computed from the discrete state vector. The output can be the instantaneous lift, instantaneous drag, or as simple as the state itself at a certain time t. Besides the output, the output sensitives are often important as well. These are as,

$$\frac{\partial J}{\partial \mu} \in \mathbb{R}^N. \tag{3.21}$$

The output sensitivity measures how the output changes given changes in the input parameters, $\boldsymbol{\mu}$. There are many different ways to find this derivative [54]:

1. Finite difference: increment the input μ by ϵ to find

$$\lim_{\epsilon \to 0} \frac{dJ}{d\mu_i} \approx \frac{J(\boldsymbol{\mu} + \epsilon_i) - J(\boldsymbol{\mu})}{\epsilon_i} \tag{3.22}$$

   A finite difference is easy to implement but is expensive for a large number of inputs as separate finite differences have to be computed for each input. It is reliable, but for unsteady problems, this calculation will need to be made for every time step; however, this procedure is useful for problems with a small number of inputs.

2. Tangent Linearization: One can rewrite the output, J, as N operations and use the chain rule to differentiate the sequence

$$\begin{aligned} J &= J_N(J_{N-1}(\mathcal{J}_0(\boldsymbol{\mu})) \\ \delta J &= J'_N J'_{N-1} \dots J'_0 \delta \boldsymbol{\mu} \end{aligned} \tag{3.23}$$

   The computational cost of the forward tangent linearization is on the order of finite difference; however, the derivatives are computed exactly without numerical approximation via $\boldsymbol{\epsilon}$. Unlike finite differences, a tangent linearization is difficult to implement, because it requires the calculation of derivative quantities.

3. The adjoint method: One can first solve for $\boldsymbol{\Psi}$ in the expression,

$$\frac{dJ}{d\mu} = \boldsymbol{\Psi}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}, \tag{3.24}$$

   where $\boldsymbol{\Psi} \in \mathbb{R}^N$ is the adjoint. Calculating sensitivities of the output with the adjoint method allows for sensitivity and output prediction without having to solve the forward simulation problem each time to find the change in the output. Additionally, the adjoint method is more useful compared to finite difference and tangent linearizations for computing a large number of sensitives for one given output. With the adjoint derivative, no forward analysis is needed every time a different input is used. For every residual calculation there is one vector product per sensitivity. Due to the flexibility of the adjoint, it is very useful when the forward problem is expensive and the number of inputs is large.

For this research on chaotic flows, the adjoint method is chosen for sensitivity calculation and error estimation due to its need to only calculate the forward problem once for any residual perturbation due to discretization errors.

For output-based error estimation specifically, it will be shown that the discretization errors are of the main interest and depend on the type of discretization and the discretization parameters used to solve the governing equation. This information is encapsulated in the residual. The residual takes into account how the solution state changes given a change in the time step, or a change in the spatial discretization.

To calculate the output sensitivity, the discrete adjoint is defined as

$$\boldsymbol{\Psi} = \left(\frac{\partial J}{\partial \boldsymbol{R}}\right)^T \in \mathbb{R}^N, \tag{3.25}$$

where N refers to the total size of the discretized system and $\boldsymbol{\Psi}$ is a vector of the sensitives. Note that the adjoint can also be used to find output sensitives. The adjoint can be found for steady and unsteady systems. To derive the corresponding adjoint equations for the governing equations of interest, a sensitivity analysis will be considered. The ideas from calculating the sensitivity in Eqn. 3.21 will be used for output-based error estimation as well.

3.2.1 Adjoint Derivation for Steady Systems

The adjoint equation can be derived from a discrete sensitivity analysisi [2, 25, 54]. Given the inputs $\boldsymbol{\mu}$ and the residual $\boldsymbol{R}$ defined in Eqn. 3.19, the following general relationship is true for a system of size N with output J.

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs}} \rightarrow \underbrace{\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\mu}) = 0}_{N \text{ equations}} \rightarrow \underbrace{\boldsymbol{U}}_{\text{state } \in \mathbb{R}^N} \rightarrow \underbrace{J(\boldsymbol{U})}_{\text{output(scalar)}} \tag{3.26}$$

To begin the adjoint derivation, the effects of a small perturbation in the inputs of interest is investigated first,

$$\boldsymbol{\mu} \rightarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}. \tag{3.27}$$

A small perturbation in the inputs will change the residual resulting in a perturbed residual that is non zero,

$$\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\mu}) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu} + \mathcal{O}(\delta\boldsymbol{\mu}^2) = \delta\boldsymbol{R} \neq \boldsymbol{0}, \tag{3.28}$$

where $\delta \boldsymbol{R}$ is nonzero and refers to the residual perturbation due to the change in the inputs. To satisfy the perturbed equation, the inputs state has to change as well,

$$\boldsymbol{R}(\boldsymbol{U}+\delta\boldsymbol{U},\boldsymbol{\mu}+\delta\boldsymbol{\mu}) = \boldsymbol{R}(\boldsymbol{U},\boldsymbol{\mu}) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu} + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right|_{\boldsymbol{U},\boldsymbol{\mu}} \delta\boldsymbol{U} + \mathcal{O}(\delta\boldsymbol{\mu}^2) + \mathcal{O}(\delta\boldsymbol{U}^2) = \boldsymbol{0} \tag{3.29}$$

In addition, the state perturbation affects the output,

$$J(\boldsymbol{U}+\delta\boldsymbol{U}) = J(\boldsymbol{U}) + \frac{\partial J}{\partial \boldsymbol{U}}\delta\boldsymbol{U} + \mathcal{O}(\delta\boldsymbol{U}^2). \tag{3.30}$$

Assuming small perturbations, subtracting 3.28 from 3.29 gives

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\delta\boldsymbol{U} = -\delta\boldsymbol{R} \rightarrow \delta\boldsymbol{U} = -\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^{-1}\delta\boldsymbol{R}. \tag{3.31}$$

Substituting Eqn. 3.31 into 3.30 gives the output perturbation in terms of the residual perturbation,

$$\delta J = \underbrace{-\frac{\partial J}{\partial \boldsymbol{U}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^{-1}}_{\boldsymbol{\Psi}^T\in\mathbb{R}^N}\delta\boldsymbol{R}. \tag{3.32}$$

Taking the transpose and rearranging Eqn. 3.32 gives the final steady discrete adjoint equation,

$$\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right)^T\boldsymbol{\Psi} + \left(\frac{\partial J}{\partial \boldsymbol{U}}\right)^T = \boldsymbol{0}. \tag{3.33}$$

Note that the each component of the adjoint, $\boldsymbol{\Psi}$, refers to the sensitivity of J to changes in the corresponding residual component. Note that Eqn. 3.33 looks similar to Eqn. 3.8 and Eqn. 3.9, which is expected since the discrete adjoint is an approximation to the continuous adjoint. More on this relationship will be explained in section 3.2.2. Once the adjoint is calculated from Eqn. 3.33, the sensitivities for the steady problem can be calculated from

$$\frac{dJ}{d\boldsymbol{\mu}} = \boldsymbol{\Psi}^T\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{U},\boldsymbol{\mu}}. \tag{3.34}$$

Once the adjoint has been found, no more solves are necessary to calculate the new sensitivity for a new perturbations of the same output.

3.2.2 Adjoint Consistency

In section 3.2.1, the discrete adjoint was derived. The results of this derivation show that the discrete adjoint equation is similar to the continuous adjoint equation derived in section 3.1. This suggests that the discrete adjoint must produce discretized solutions that approximate the continuous adjoint. This requirement is referred to as "adjoint consistency", which is necessary to produce accurate adjoints and adequate convergence of the output of interest. Adjoint consistency requirements have been analyzed theoretically and numerically in works by Oliver et. al and Lu [55, 56], and have been used in multifidelity PDE-constrained optimization by Chen et. al [57]. In its simpliest form, adjoint consistency states that the discrete adjoint needs to satisfy the continuous adjoint equation. One can recall the DG approximation for the states and make the connection that the discrete adjoint consists of coefficients for the approximation of the continuous adjoints, which uses the same basis functions as the the state.

The quantification of the error between the continuous and discrete adjoint is important for error estimation. Adjoint consistency has an effect not only on the convergence of the adjoint approximation but on the primal approximation as well. If the adjoint is not adjoint consistent, the solutions of the adjoint can be oscillatory for a non chaotic problem, which will pollute and affect the error estimate with noise, leading to incorrect areas for adaptation in mesh adaptation. Imposing adjoint consistency for traditional adjoint problems affects the output definition and the interior and boundary discretizations of the semi-linear form. To determine adjoint consistency, the same principles from section 3.1 are applied to the general semilinear form.

3.2.2.1 Discretized Adjoint Formulation

The semilinear residual form is

$$\mathcal{R}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{0}, \quad \forall \boldsymbol{v} \in \boldsymbol{\mathcal{V}}, \tag{3.35}$$

where $\mathcal{V}$ refers to the test space for the PDE. and the output it defined as

$$J(\boldsymbol{u}), \quad \boldsymbol{u} \in \mathcal{U}. \tag{3.36}$$

The state $\boldsymbol{u} \in \mathcal{U}$ is the solution of the governing equations lying the space $\mathcal{U}$, which is an infinite-dimensional space. Note, that the basis functions are chosen on the state space and adjoint space, $\mathcal{U} = \mathcal{V}$. The dual-continuous adjoint equation is found by considering the variations of the Lagrangian from section 3.1. The dual equation takes the form

$$\mathcal{R}'[\boldsymbol{u}](\boldsymbol{v},\boldsymbol{\psi}) = J'[\boldsymbol{u}](\boldsymbol{v}), \quad \forall \boldsymbol{v} \in \boldsymbol{\mathcal{V}}. \tag{3.37}$$

where $\boldsymbol{\psi}$ is introduced in the Lagrangian equation as a constraint multiplier. It is difficult to solve most PDEs analytically, thus the DG discretization on a finite-dimensional space $\mathcal{V}_h$ seeks $\boldsymbol{\psi}_h \in \mathcal{V}_h$ such that Eqn. 3.37 is discretized as

$$\mathcal{R}_h'[\boldsymbol{u}_h](\boldsymbol{v}_h,\boldsymbol{\psi}_h) = J_h'[\boldsymbol{u}_h](\boldsymbol{v}_h), \quad \forall \boldsymbol{v}_h \in \boldsymbol{\mathcal{W}}_h \tag{3.38}$$

where the prime symbols refer to the Fréchét derivatives. $\boldsymbol{\mathcal{W}}_h$ is defined as

$$\boldsymbol{\mathcal{W}}_h = \boldsymbol{\mathcal{V}}_h + \boldsymbol{\mathcal{V}} = \{\boldsymbol{h} = \boldsymbol{f} + \boldsymbol{g} : \boldsymbol{f} \in \boldsymbol{\mathcal{V}}_h, \boldsymbol{g} \in \boldsymbol{\mathcal{V}}\} \tag{3.39}$$

Note that Eqn. 3.38 produces the discretized adjoint and that the trial space and test space are assumed to be the same.

3.2.2.2 Continuous Adjoint Formulation

The exact adjoint equation can be found by manipulating the weak form of Eqn. 3.35 with a different technique by looking at perturbations of $\boldsymbol{u}$, $\delta\boldsymbol{u}$. Consider an infinitesimal state perturbation $\delta\boldsymbol{u}$ added to the state of the weak form of the semilinear residual. The sensitivity of the output to the residual perturbation is defined as

$$\delta J = J(\boldsymbol{u}+\delta\boldsymbol{u}) - J(\boldsymbol{u}) = \delta\mathcal{R}(\psi) = \mathcal{R}(\boldsymbol{u},\boldsymbol{\psi}) - \mathcal{R}(\boldsymbol{u}+\delta\boldsymbol{u},\boldsymbol{\psi}) \tag{3.40}$$

Given that $\mathcal{R}(\boldsymbol{u},\boldsymbol{\psi}) = 0$, the state, $\boldsymbol{u}$, is already the solution to the weak form, $\mathcal{R}$, one can find the dual form of Eqn. 3.40 via techniques from section 3.1,

$$\mathcal{R}'[\boldsymbol{u}](\delta\boldsymbol{u},\boldsymbol{\psi}) = J'[\boldsymbol{u}](\delta\boldsymbol{u}), \quad \forall\, \delta\boldsymbol{u} \in \mathcal{U}, \tag{3.41}$$

where primes denote the Fréchét derivative with respect to arguments in brackets. This technique produces the exact adjoint, $\boldsymbol{\psi}$.

3.2.2.3 Requirements for Adjoint Consistency

The original DG discretization is adjoint consistent if the exact adjoint, $\boldsymbol{\psi}$, and exact solution, $\boldsymbol{u}$, from the continuous adjoint equation, Eqn. 3.41 satisfy the discrete equation, Eqn. 3.38, such that,

$$\mathcal{R}_h'[\boldsymbol{u}](\boldsymbol{v},\boldsymbol{\psi}) = J_h'[\boldsymbol{u}](\boldsymbol{v}) \quad \forall \boldsymbol{v} \in \boldsymbol{\mathcal{W}}_h. \tag{3.42}$$

If Eqn. 3.42 is not satisfied, the discretization defined by the semilinear residual, $\mathcal{R}_h$, can still be asymptotically adjoint consistent if Eqn. 3.38 holds true when $h \to 0$ [55],

$$\lim_{h\to 0}\left(\sup_{v\in\mathcal{W}_h,\|v\|_{\mathcal{W}_h}=1} \left|\mathcal{R}_h'[\boldsymbol{u}](\boldsymbol{v},\boldsymbol{\psi}) - J_h'[\boldsymbol{u}](\boldsymbol{v})\right|\right) = 0. \tag{3.43}$$

Additionally, discretizations defined by $\mathcal{R}_h$ that are adjoint inconsistent can often be modified to be adjoint consistent by appending terms to the semilinear form or to the output definition.

3.3 Unsteady Discrete Adjoint Formulation

Unsteady adjoints require additional consideration of the effect of input perturbations on the output at different time indices. The linear adjoint equation is similar to Eqn. 3.33 except that instead of looking at the spatial residual, $\boldsymbol{R}$ refers to the total unsteady residual consisting of residuals at different time nodes i. The inputs affect the output via,

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs}} \to \underbrace{\boldsymbol{R}^i\left(\boldsymbol{U}^j,\boldsymbol{\mu}\right) = \boldsymbol{0}}_{N\text{ equations}} \to \underbrace{\boldsymbol{U}^j}_{\text{state }\in\mathbb{R}^N} \to \underbrace{J(\boldsymbol{U}^j)}_{\text{output(scalar)}}\ , \tag{3.44}$$

where j refers to the time indices in the unsteady discretization. Note that $j \le N_t$ where N_t refers to the total number of time indices. Eqn. 3.33 is applied to each time node for the total residual to give

$$\sum_{i=1}^{N_t}\left(\frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}^j}\right)^T \boldsymbol{\Psi}^i + \left(\frac{\partial J}{\partial \boldsymbol{U}^j}\right)^T = \boldsymbol{0}. \tag{3.45}$$

Backwards substitution can be used to solve for the unsteady adjoint. It is possible to use backwards substitution due to the structure of the unsteady Jacobian matrix in the linear adjoint equation, which can be seen in Eqn. 3.46 for BDF1 temporal discretization for the backwards difference method.

$$\left(\frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{U}^j}\right)^T = \begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & * \\ & & & & & & * \end{bmatrix} \tag{3.46}$$

Each $*$ refers to one $N \times N$ block, where the $*$ on the main diagonal for BDF1 is written as

$$\frac{\partial \boldsymbol{R}(\boldsymbol{U}^{n+1})}{\partial \boldsymbol{U}^{n+1}} = \frac{\boldsymbol{M}}{\Delta t} c_0 + \frac{\partial \boldsymbol{R}_s(\boldsymbol{U}^{n+1})}{\partial \boldsymbol{U}^{n+1}}, \tag{3.47}$$

and the $*$ on the off diagonal for BDF1 is written as

$$\frac{\partial \boldsymbol{R}(\boldsymbol{U}^{n+1})}{\partial \boldsymbol{U}^{n+1}} = \frac{\boldsymbol{M}}{\Delta t} c_1. \tag{3.48}$$

Once the unsteady adjoint solution has been found, the sensitivity is calculated by

$$\frac{dJ}{d\boldsymbol{\mu}} = \sum_{i=1}^{N_t} \left(\boldsymbol{\Psi}^i\right)^T \left(\frac{\partial \boldsymbol{R}^i}{\partial \boldsymbol{\mu}}\right). \tag{3.49}$$

Note that for nonlinear problems, the states need to be saved to disk in order to calculate the Jacobian at each time index.

The next section will go over how the continuous in time unsteady adjoint is computed for a given output.

3.4 Unsteady Temporal Continuous Adjoints Derivation

To derive the unsteady continuous adjoint equations, consider the continuous formulation of the governing equations,

$$\boldsymbol{r}(\boldsymbol{u}) = \frac{d\boldsymbol{u}}{dt} - \boldsymbol{f}. \tag{3.50}$$

Typically, the output is set to the instantaneous value of interest at a finite time; however, for the purpose of output-based error estimation for chaotic flows, the time-average output is analyzed,

$$\overline{J} = \frac{1}{T} \int_{T_0}^{T_f} J(\boldsymbol{u})\, dt. \tag{3.51}$$

To construct the continuous adjoint equations for the linearized output, variations are used. The Lagrangian is constructed with the introduction of the Lagrange multipliers, $\boldsymbol{\psi}$, which were used when deriving the exact adjoint. Note that $\boldsymbol{\psi}$ depends on the computational domain Ω. The Lagrangian $\mathcal{L}$ is defined and simplified as

$$\begin{aligned}\mathcal{L} &= \overline{J}(\boldsymbol{u}) - \int_{T_0}^{T_f} \boldsymbol{\psi}^T \boldsymbol{r}(\boldsymbol{u})\ dt \\ &= \frac{1}{T}\int_{T_0}^{T_f} J(\boldsymbol{u})\ dt - \int_T \boldsymbol{\psi}^T \left[\frac{d\boldsymbol{u}}{dt} - \boldsymbol{f}\right]\ dt \\ &= \frac{1}{T}\int_{T_0}^{T_f} J(\boldsymbol{u})\ dt - \int_{T_0}^{T_f} \boldsymbol{\psi}^T \frac{d\boldsymbol{u}}{dt} dt + \int_{T_0}^{T_f} \boldsymbol{\psi}^T \boldsymbol{f} dt\end{aligned} \tag{3.52}$$

One can integrate the third term by parts and combine like terms to obtain,

$$\mathcal{L} = -\left[\boldsymbol{\psi}^T \boldsymbol{u}\right]_{T_0}^{T_f} + \int_{T_0}^{T_f} \left[\frac{1}{T} J(\boldsymbol{u}) + \frac{d\boldsymbol{\psi}^T}{dt}\boldsymbol{u} + \boldsymbol{\psi}^T \boldsymbol{f}\right]\ dt. \tag{3.53}$$

Linearizing the Lagrangian and requiring that it be stationary with respect to permissible state variations, δu, produces,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\delta\boldsymbol{u} = -\left[\boldsymbol{\psi}^T \delta\boldsymbol{u}\right]_{T_0}^{T_f} + \int_{T_0}^{T_f} \left[\frac{1}{T}\frac{\partial J(\boldsymbol{u})}{\partial \boldsymbol{u}} + \frac{d\boldsymbol{\psi}^T}{dt} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right] \delta\boldsymbol{u}\ dt. \tag{3.54}$$

The terms multiplied by δu inside the time integral must equal to zero and form the continuous adjoint equation for $\boldsymbol{\psi}$. Taking the transpose of the equation yields,

$$\frac{d\boldsymbol{\psi}}{dt} + \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\psi} + \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}}^T = \boldsymbol{0}, \tag{3.55}$$

where the terminal conditions are the $t = T$ terms

$$\boldsymbol{\psi}\Big|_{t=T_f} = -\frac{\partial J}{\partial \boldsymbol{u}}^T\Big|_{t=T_f} \tag{3.56}$$

It will be shown that Eqn. 3.55 and the discrete form of the adjoint equation fail for chaotic flows. An alternate version of the unsteady continuous adjoint equation will be derived via the Least Squares Shadowing method, which can be compared and contrasted to the traditional adjoint methods derived in this section.

3.5 Output-Based Error Estimation

In computational fluid dynamics, the purpose of output-based error estimation is to quantify as best as possible the discretization errors associated with a simulation on a finite-dimensional approximation space. One can specifically consider two different levels of discretization: the coarse space (H) and the fine space (h). One can compute a scalar output of interest, J. Our goal is to predict the difference in outputs calculated using two different levels of discretization; this is important in order to successfully apply mesh adaptation, which leads to decreases in computational costs. Before analyzing the technique used to estimate discretization errors, the characteristics of the coarse and fine spaces need to be described in detail.

The discretization space refers to the mesh of the simulation on which the state and adjoint are approximated in space and time. In contrast, the fine space refers to the space that is obtained after mesh refinement or approximation order increment. Ideally, the fine solution computed on the fine mesh would be the exact solution, $\boldsymbol{U}$; however, this is usually unknown and instead represents a solution of order $p_h = p_H + 1$ and or $r_h = r_H + 1$. The coarse space state is associated with the coarse space residual and mesh, while the fine space state is associated with the fine space residual and mesh. These can be written as

$$\begin{aligned} \text{Coarse space} \rightarrow \boldsymbol{R}_H(\boldsymbol{U}_H) = \mathbf{0} \rightarrow \boldsymbol{U}_H \rightarrow J_H(\boldsymbol{U}_H), \\ \text{Fine space} \rightarrow \boldsymbol{R}_h(\boldsymbol{U}_h) = \mathbf{0} \rightarrow \boldsymbol{U}_h \rightarrow J_h(\boldsymbol{U}_h). \end{aligned} \tag{3.57}$$

The discretization error of interest is defined as

$$\delta J \equiv J_h(\boldsymbol{U}_h) - J_H(\boldsymbol{U}_H). \tag{3.58}$$

An estimate of δJ will be a function of the adjoint and the injection of the coarse space solution into the fine space.

3.5.1 The Adjoint-Weighted Residual

It was shown in section 3.1 that given a perturbed residual equation, the corresponding dual or adjoint equation can be found. This derivation led to the continuous error estimate equation of the perturbed output of interest as a function of the adjoint and the perturbed residual. However, this form is not useful for error estimation, because it requires the exact adjoint. A better form that takes into account discretization errors given an already discretized state will be more useful. To begin, the residuals for both levels of discretization

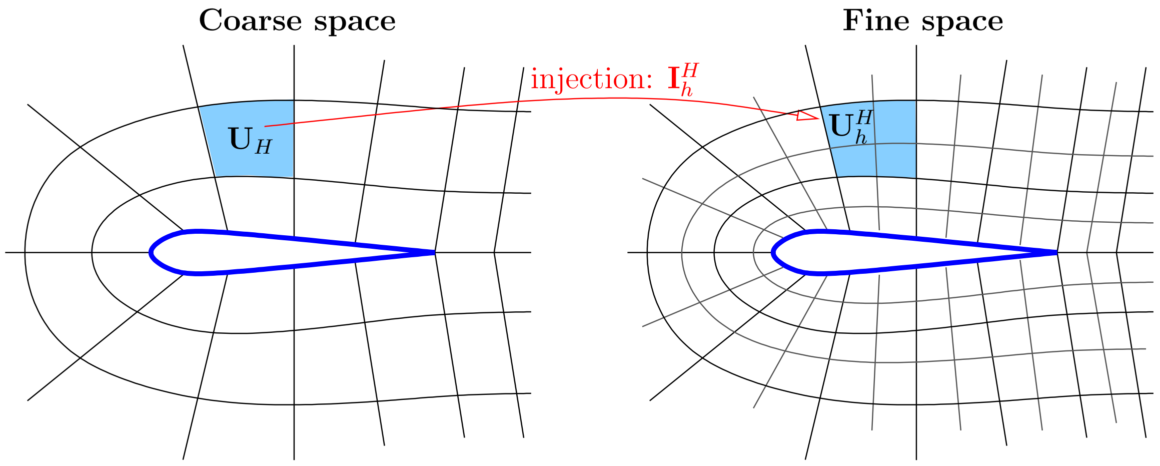


Figure 3.2: Coarse space solution injected in the fine space via an injection $\boldsymbol{I}_h^H$

are defined as

$$\boldsymbol{R}_h(\boldsymbol{U}_h) = \mathbf{0}, \quad \boldsymbol{R}_H(\boldsymbol{U}_H) = \mathbf{0}. \tag{3.59}$$

The discretization error can be found by looking at the coarse state injected into the fine space,

$$\boldsymbol{U}_h^H = \boldsymbol{I}_h^H \boldsymbol{U}_H, \tag{3.60}$$

where I_h^H is the injection operator. Figure 3.2 shows how the fine space is obtained by uniform refinement or incrementing the approximation order of each element. That process refers to the injection of the state from the coarse space to the fine space, where $\boldsymbol{U}_h^H$ contains the errors in the state between the coarse and fine spaces. This error is reflected in the calculation of the residual of the coarse solution on the fine space

$$\boldsymbol{R}_h(\boldsymbol{U}_h^H) \neq \mathbf{0}. \tag{3.61}$$

This residual is referred to as the perturbed residual and contains information on the error between the fine and coarse space solutions. The state injected from the coarse space will usually not be a fine space solution and thus will give non-zero fine space residuals. Along with the fine space adjoint, $\boldsymbol{\Psi}_h$, the perturbation of the residual will predict how the output will change between the two spaces. Next, the adjoint-weighted residual for output error estimation is introduced.

The adjoint-weighted residual is found by linearizing the output $J(\boldsymbol{U}_h)$ about $\boldsymbol{U}_h^H$ [58],

$$J_h(\boldsymbol{U}_h) = J_h(\boldsymbol{U}_h^H) + \left.\frac{\partial J_h}{\partial \boldsymbol{U}_h}\right|_{\boldsymbol{U}_h^H} \delta\boldsymbol{U} + \mathcal{O}(\delta\boldsymbol{U}^2), \tag{3.62}$$

where $\delta \boldsymbol{U}_h = \boldsymbol{U}_h - \boldsymbol{U}_h^H$. Additionally, linearizing the fine space residual about the injected coarse solution in the fine space gives,

$$\boldsymbol{R}_h(\boldsymbol{U}_h) = \boldsymbol{R}_h(\boldsymbol{U}_h^H) + \left.\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{U}_h}\right|_{\boldsymbol{U}_h^H} \delta \boldsymbol{U} + \mathcal{O}(\delta \boldsymbol{U}^2) = \boldsymbol{0}. \tag{3.63}$$

Dropping the high order terms and rearranging Eqn. 3.63 to find $\delta \boldsymbol{U}$ produces

$$\delta \boldsymbol{U} \approx -\left(\left.\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{U}_h}\right|_{\boldsymbol{U}_h^H}\right)^{-1} \boldsymbol{R}_h(\boldsymbol{U}_h^H). \tag{3.64}$$

Substituting Eqn. 3.64 into Eqn. 3.62 gives

$$\underbrace{J_h(\boldsymbol{U}_h) - J_h(\boldsymbol{U}_h^H)}_{\delta J_h} = -\underbrace{\left.\frac{\partial J_h}{\partial \boldsymbol{U}_h}\right|_{\boldsymbol{U}_h^H} \left(\left.\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{U}_h}\right|_{\boldsymbol{U}_h^H}\right)^{-1}}_{\boldsymbol{\Psi}_h^T} \boldsymbol{R}_h(\boldsymbol{U}_h^H) + \mathcal{O}(\delta \boldsymbol{U}^2). \tag{3.65}$$

Dropping the higher-order terms, the discrete adjoint-weighted residual output error estimate is

$$\delta J \approx -\boldsymbol{\Psi}_h^T \boldsymbol{R}_h(\boldsymbol{U}_h^H). \tag{3.66}$$

Note that this derivation assumes small perturbations in the state and residual when the output or equations are nonlinear. For this research $\boldsymbol{\Psi}_h$ is calculated exactly on the fine space to minimize additional approximations in the error estimates. The adjoint is calculated in a Galerkin variational form, which by Galerkin orthogonality means that the adjoint weighted residual can be modified without affecting the error estimate in the following way

$$\delta J = -\delta \boldsymbol{\Psi}_h^T \boldsymbol{R}_h(\boldsymbol{U}_h^H), \tag{3.67}$$

where the difference in the fine and injected coarse adjoints in the fine space is defined as

$$\delta \boldsymbol{\Psi} = \boldsymbol{\Psi}_h - \boldsymbol{\Psi}_h^H. \tag{3.68}$$

In certain cases, e.g. for viscous problems in DG, Galerkin orthogonality may not hold,

$$\boldsymbol{R}_h(\boldsymbol{U}_h^H, \boldsymbol{\Psi}_h^H) \neq \boldsymbol{0}. \tag{3.69}$$

In general, for linear inviscid problems, the fine-space residual of the injected coarse solution weighted by the injected coarse adjoint is zero; however, when Galerkin orthogonality does not hold, the error estimate tends to over estimate the actual discretization error. Using $\delta\boldsymbol{\Psi}$ leads to more accurate error estimates in these cases. The injected coarse adjoint into, $\boldsymbol{\Psi}_h^H$ is calculated by first projecting the fine adjoint solution into the coarse mesh, and then injecting the final solution back into the fine space. This is summarized with injection and projection operators $\boldsymbol{I}_h^H$ and $\boldsymbol{I}_H^h$,

$$\boldsymbol{\Psi}_h^H = \boldsymbol{I}_h^H \boldsymbol{I}_H^h \boldsymbol{\Psi}_h. \tag{3.70}$$

Substituting Eqn. 3.70 into Eqn. 3.67 gives,

$$\delta J = \left[(\boldsymbol{I}_h - \boldsymbol{I}_h^H \boldsymbol{I}_H^h)\boldsymbol{\Psi}_h\right]^T \boldsymbol{R}_h(\boldsymbol{U}_h^H), \tag{3.71}$$

where $\boldsymbol{I}_h$ is an identity matrix whose size is that of the fine-space solution. Note that two different methods are used for the projection matrix $\boldsymbol{I}_H^h$, least-squares projection and H_1 projection. H_1 projection is similar to the least-squares projection, but in addition the slopes at the basis nodes are constrained as well. For unsteady discrete error estimation, the error estimate is written as

$$\delta J \approx -\sum_{i=1}^{N_t} \left(\boldsymbol{\Psi}_h^i\right)^T \boldsymbol{R}_h^i \left(\boldsymbol{U}_h^H\right), \tag{3.72}$$

where N_t is the number of time nodes and $\boldsymbol{R}_h$ refers to the total fine-space residual. Due to the Galerkin orthogonality form, the unsteady discrete error estimate can be written as

$$\delta J \approx -\sum_{i=1}^{N_t} \left(\delta\boldsymbol{\Psi}_h^i\right)^T \boldsymbol{R}_h^i \left(\boldsymbol{U}_h^H\right). \tag{3.73}$$

A similar unsteady error estimate can be found from the unsteady continuous adjoint,

$$\delta J \approx -\int_{T_0}^{T_f} \boldsymbol{\psi}_h^T \boldsymbol{R}(\boldsymbol{U}_h^H)\, dt, \tag{3.74}$$

where once again $\boldsymbol{R}_h(\boldsymbol{U}_h) = \boldsymbol{0}$. Again, when Galerkin orthogonality holds, the adjoint can be modified such that the continuous error estimate can be written as

$$\delta J \approx -\int_{T_0}^{T_f} \delta\boldsymbol{\psi}_h^T \boldsymbol{R}(\boldsymbol{U}_h^H)\, dt, \tag{3.75}$$

where $\delta\boldsymbol{\psi}_h = \boldsymbol{\psi}_h - \boldsymbol{\psi}_h^H$. For unsteady problems, the integral is evaluated using quadrature for DGTIME and high-order multistep/multistage temporal methods. For low order multistep/multistage temporal methods, the integral is evaluated using the trapezoidal rule.

3.6 Summary

In this chapter, the output-based error estimation method is introduced. In order to successfully apply output-based error estimation to LES, the accurate calculation of the adjoint is needed. This chapter concentrated on the traditional adjoint method and derived both the discrete and continuous adjoint formulation for steady and unsteady systems.

Output-based error estimates rely on a perturbed residual that is calculated on a fine space using the injected coarse solution. This non zero residual drives the error estimate, which is calculated by the adjoint-weighted residual method. Both the discrete and continuous versions of the adjoint-weighted residual method were introduced. In the next chapter, chaos theory will be introduced and will be used to explain why the traditional adjoint and error estimation methods fail for chaotic systems.

CHAPTER IV

Chaotic Flows

Output-based error estimation for chaotic flows is challenging due to the unique nature of chaotic systems and its effect on the adjoint derivative. To fully appreciate the challenges involved in this process, this chapter introduces chaos theory and ergodic theory, which provide some indication of a possible way to formulate an accurate chaotic adjoint for error estimation. Next, different prototypical chaotic systems are introduced and analyzed. This chapter then summarizes the main points of chaotic systems that are important to keep in mind when applying output-based error estimation to chaotic systems in the next chapters.

In 1687, Issac Newton stated famously that our world is a predictable mechanical system, a “clockwork universe” [59]. In 1814, Pierre-Simon Laplace said that “nothing would be uncertain and the future, as the past, would be present in [our] eyes” [60]. Just like when the world thought the earth was the center of the system, civilization lived in a world of predictability defined by Newton and Laplace. Laplace in his essay described how Halley’s comet of 1456 was discovered. He described how it spread terror across Europe and how these fears were cause by “ignorance”. They were unaware that Halley’s comet followed the laws of universe, but soon after Hally’s comet was discovered, scientists were able to predict the comet’s next return trajectory. Laplace then describes how “the trajectory of a simple molecule of air or vapour is regulated in a manner as certain as that of the planetary orbits” and that “the only difference between them is that which is contributed by our ignorance” [60]. He boldly stated that astronomy shows us the movement of the comets takes place in all phenomena. As time goes on, it has been evident that his inability to truly understand the intricacies of nature is as he said “caused by ignorance”.

It was not until 1963 that Edward Lorenz at MIT was able to clearly highlight how predictability was not necessarily possible for all phenomena. As a meteorologist, Edward Lorenz found that some systems behave “in an irregular, seemingly haphazard manner, and even when observed for long periods of time, do not appear to repeat their previous history” [61]. He found that this was the case for turbulent flows, where instantaneous flow patterns were unpredictable but statistical behaviors of turbulence were more predictable.

Additionally, he found it a challenge to predict extreme weather patterns such as cyclones from a short term weather forecast. To model these unique convective weather patterns, Edward Lorenz formulated his famous Lorenz Attractor equation, which exhibited that small initial changes to the variables of the system led to completely different patterns and trajectories. This idea became known as the “butterfly effect” when Lorenz famously stated that “the flap of a butterfly’s wings might ultimately cause a tornado”. This observation that Lorenz made in 1963 is the reason why advanced simulation techniques have been difficult to apply to turbulent flows. This can be seen in more detail by looking at one technique, output-based error estimation, which relies on the successful calculation of an accurate adjoint. Applying output-based error estimation with the traditional adjoint methods from Chapter III to Edward Lorenz’s Lorenz Attractor would fail, because of how the adjoint $\boldsymbol{\psi}$, behaves over long periods of time,

$$\boldsymbol{\psi}^{\infty} \neq \lim_{T\to\infty} \boldsymbol{\psi}, \tag{4.1}$$

where the goal is to calculate $\boldsymbol{\psi}^{\infty}$. As $T \to \infty$, the adjoint, $\boldsymbol{\psi}$, will diverge exponentially due to the butterfly effect, making it more difficult to find $\boldsymbol{\psi}^{\infty}$.

This unpredictability is a driver for new updated techniques that will be able to make chaos more “predictable” in a way such that with output based error estimation, discretization errors can actually be quantified statistically and minimized. Quantification of discretization errors in chaotic flows can additionally be used with adaptive methods to increase accuracy and predictability of the simulation. Confidence intervals can be used to determine if the time average output of interest is within sufficient numerical accuracy, which is essential when studying time average outputs. Even if it is not possible to accurately calculate adjoints for chaotic flows, the error estimates can still be used as localized error indicators, which pin points where the greatest errors in the simulation are located in the mesh for adaptive methods. Overall, The application of output-based error estimations for estimation of discretization errors of chaotic systems can help chaos be effectively more “predictable” than without it.

4.1 Chaos Theory

Chaos is defined as aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on its initial conditions [62]. In the next three subsections, each of these behaviors will be defined. More details are given in Strogatz’s book.

4.1.1 Aperiodic Long Term Random Trajectories

To describe chaos, one can begin by looking at a fluid particle in space that is influenced by a vector field written as an ODE,

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}) \tag{4.2}$$

When $T > 0$, the particle begins to move along space with a certain undefined velocity. This space is referred to as the phase space. The particle is referred to as the phase point at the initial conditions and the position of the phase point will change as a function of time. This position is referred to as the trajectory of the particle. This trajectory is the solution of the differential equation. A collection of all the trajectories of the system is referred to the phase portrait where all the trajectories are generated from different initial conditions. Every phase portrait has fixed points which are the stagnation points of the system. At these fixed points, $\boldsymbol{f}(\boldsymbol{u}_{fixed}) = 0$ and are rare stable points where a particle would be captured or forced into. Fixed points can be referred to as equilibrium points as well, where small disturbances are damped out over short periods of time. The opposite of a fixed point is an unstable point, which in a chaotic system fuels its erratic, unpredictable behavior. For a system to be considered chaotic, the trajectories need to never reach these fixed equilibrium points and must not settle or reach periodic behavior as $T \to \infty$.

4.1.2 Deterministic Behavior

Along with exhibiting aperiodic behavior, chaotic systems are deterministic as well. In deterministic models, the output of interest is only determined by the given parameters of the system and its initial conditions. This is different from stochastic models, which contain another inherent level of randomness in time. Given the same set of parameters and initial conditions as a chaotic system, a stochastic system will lead to a completely unique ensemble of different output results at difference times, making stochastic systems much more complicated to work with. Overall, chaotic systems are predictable for short periods of time, while stochastic systems are unpredictable and random at all times. Methods and results described in this paper will not work on non-deterministic systems since temporal consistency -ability to reconstruct the same trajectory for a particular equation at different times- in its ensemble results is assumed.

4.1.3 Sensitivity to Initial Conditions: Butterfly Effect

Along with being deterministic, chaotic systems are highly sensitivity to its initial conditions. This type of indeterministic behavior in terms of its initial conditions is referred to as the butterfly effect, which was first coined by Edward Lorenz. Overall, the trajectory of a deterministic system exhibits indeterministic behavior as $T \to \infty$, but grows exponentially with time. Edward Lorenz experienced this first hand when he was running his simplified weather model in his attempt to make prediction of weather patterns more accurate. He began by looking at how bumps in the trajectories would be followed by multiple bumps and thought this may be used for future predictions. Simple observations like these gave Lorenz some ideas on how one would begin to predict future behavior of chaotic systems. In 1961, Lorenz began looking at how trajectories behaved over longer periods of time. Instead of restarting a simulation, he retrieved the final trajectory positions from the original simulation, by copying and pasting the numbers for the new run. He let the case run and expected to see the trajectories emulate the future evolution of the original solutions exactly; however, the weather diverged rapidly such that all resemblance had disappeared completely. After ruling out a malfunction or a bug in his program, he realized that the mysterious behavior was inherent to the model. For example, the final position of the original run was 0.506127, but Lorenz had copied 0.506 [63]. He was three decimal places off. However, this small, seemingly inconsequential, difference resulted in a completely different solution. He realized that a minute perturbation in the initial conditions, like a butterfly flapping outside in a weather simulation/system, could lead to large scale changes [63]. These minute differences lead to unpredictable behavior, which is contrary to what Laplace and Newton had the world believe for a long time. This sensitivity to initial conditions can be quantified mathematically by analyzing Lyapunov exponents. Define $\boldsymbol{u}(t = 0)$ as a point on the attractor. Define one addition point, $\boldsymbol{u}(t = 0) + \delta\boldsymbol{u}$ where $\delta\boldsymbol{u}$ is the chance in the trajectory that results in a change in the initial conditions.

$$\|\delta u(t)\| \sim \|\delta u_0\| e^{\lambda t} \tag{4.3}$$

Plotting $\ln \|\delta u\|$ versus time gives a curve that is approximately a straight line with a slope λ, which can be seen in Figure 4.2 for the chaotic Kuramoto-Sivashinsky Equation, a fourth order 1D PDE that will be described in more detail in the next sections.

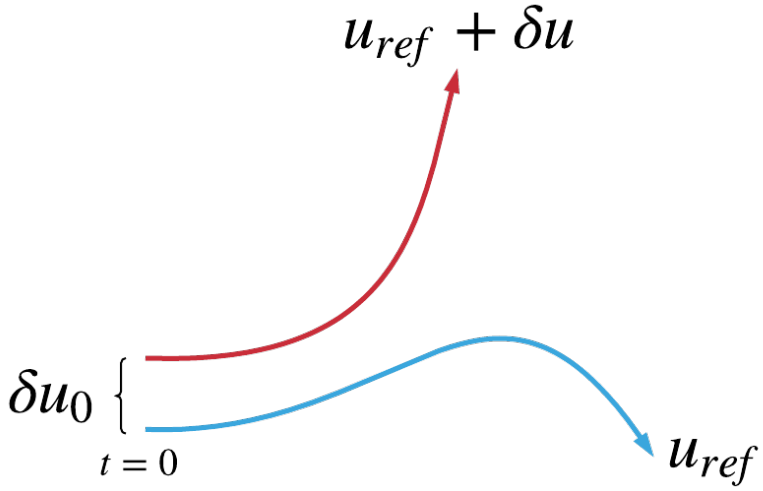


Figure 4.1: Reference and perturbed trajectory given a perturbed set of small change in initial conditions.

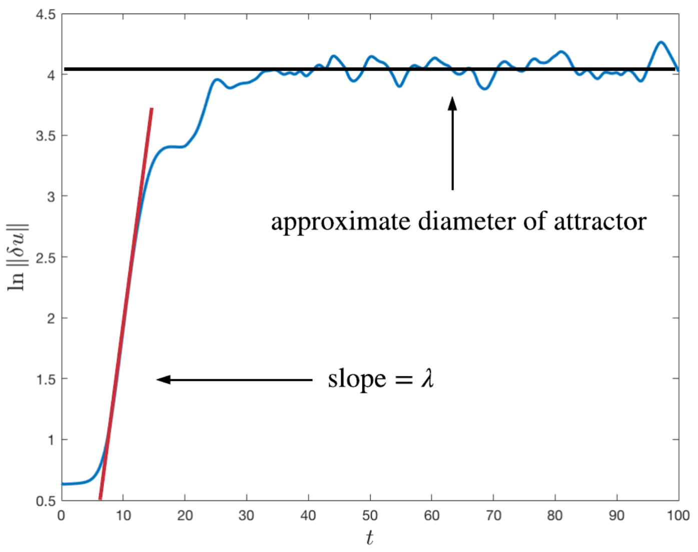


Figure 4.2: Plot of $\ln \|\delta u\|$ vs. t where the slope refers to the largest Lyapunov exponent of the system. $\ln \|\delta u\|$ steadies out once the fluid element reaches the bounds or the diameter of the attractor.

For chaotic systems, the slope is positive, which refers to the largest positive Lyapunov exponent of the system. The plotting of $||\delta u||$ is distracting from the fact that each chaotic system can have more than one Lyapunov exponent or exactly n different Lyapunov exponents for an n-dimensional systems. Thus Figure 4.2 only shows the largest Lyapunov exponent of the system. It is important to note that the Lyapunov exponent calculation from Eqn 4.3 is approximate and can be calculated numerically. Figure 4.2 shows that the largest positive Lyapunov Exponent for the chaotic Kuramoto-Sivashinsky equation is $\lambda \approx 0.3484$. The positive Lyapunov exponents are responsible for the system's sensitivity to its initial condition. The drastic weather patterns that Lorenz saw in his experiments after failing to copy and paste the exact parameters is due to the exponential growth dictated by the Lyapunov exponent. Because of the exponential growth, the time window for prediction is very short, making it necessary to find some other prediction methods for when $T \to \infty$. A system with no positive Lyapunov exponents will result in a non chaotic system.

Lyapunov exponents can be viewed as eigenvalues and have corresponding Lyapunov vectors, which can give more information on how the trajectories change given the magnitude of the largest Lyapunov exponent. Covariant Lyapunov vectors provide information about the local geometrical structure of the tangent space; literature shows how to find these vectors [64]. Covariant Lyapunov Vectors can be found through the following ordinary differential equation,

$$\frac{d}{dt}\phi_i(u(t)) = \frac{\partial f}{\partial u}_{u(t)} \cdot \phi_i(t(t)) - \Lambda_i \phi_i(u(t)) \tag{4.4}$$

The covariant Lyapunov vectors span both the stable and unstable manifolds and dictate the direction of the deformation of a fluid element while each Lyapunov exponent contribute to the total rate of deformation. Combined together, they dictate the trajectory of the flow. Covariant Lyapunov vectors and Lyapunov exponents influence how much a fluid element stretches and folds over time as seen in Figure 4.3. This process of stretching and folding, similar to a piece of paper folding over and over again, the reason why chaotic systems are sensitive to their initial conditions. This is specifically characteristic of a strange attractor, which has two main properties. First, trajectories on a strange attractor must remain in a bounded area and separate from their neighbors exponentially. The diameter of the bounded area of a strange attractor can be approximately calculated from Figure 4.2 when $\ln ||\delta u||$ levels off. The diameter for the Kuramoto-Sivashinsky equation from Figure 4.2 is $\ln ||\delta u|| \approx 4.032$. The attractor stays bounded, because there is a finite number of times a fluid particle in a chaotic system can he stretched and folded. Eventually, the distorted fluid particle must fold back on itself. Contracting and stretching behavior in fluid mechanics is seen through dissipation via friction. Lastly, the term "strange" when describing an attrac-

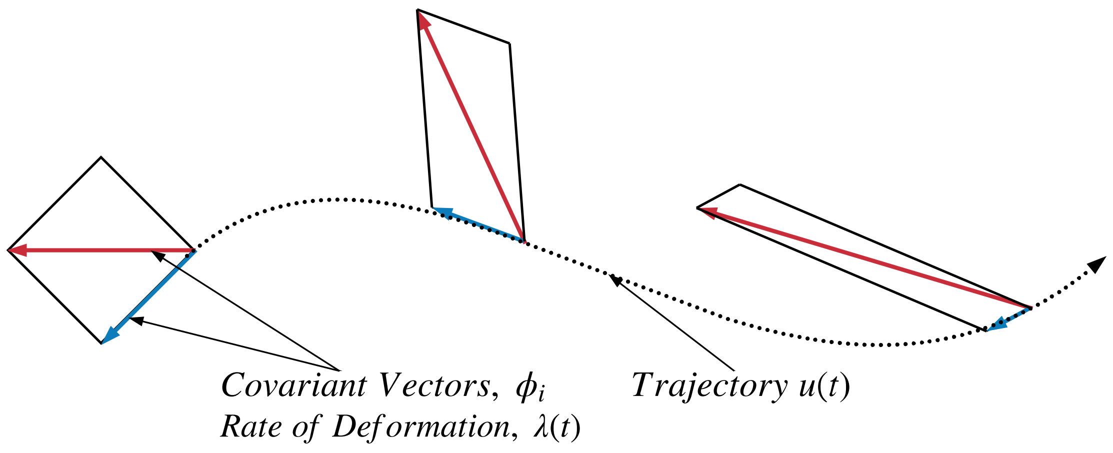

$Covariant\ Vectors,\ \phi_i$
$Rate\ of\ Deformation,\ \lambda(t)$
$Trajectory\ u(t)$

Figure 4.3: Illustration of covariant Lyapunov vectors and Lyapunov exponents acting on a fluid element

tor refers to the system's fractal structure, which are complex geometric shapes with fine structures at very small scales. If one were to zoom in on a fractal, the tiny shapes would emulate the larger shapes and exhibit similarity, which is found in many different aspects of nature.

4.1.4 Summary on Chaos Theory

Chaos was defined by three different characteristics: aperiodic behavior, deterministic, and sensitivity to initial conditions. A Chaotic system must not exhibit periodic behavior, but must be bounded. The system must as well be deterministic such that the trajectory can be reproduced at different times eluding to the idea that randomness is minimal for short periods of time. Lastly, a chaotic system is sensitivity to initial conditions as a result of the influences of the Lyapunov exponents and covariant Lyapunov vectors. In the next section, three main tools are introduced that are used to begin making error estimation possible for chaotic flows: statistical theory, ergodic theory, and chaotic hypothesis. The assumptions that be made from these ideas are essential to calculating accurate chaotic adjoints.

4.2 Statistical Theory and Ergodicity

Simple numerical simulations have given the scientific and mathematical community the opportunity to begin understanding what chaos is. When a system is chaotic and becomes unpredictable, there is a lack of information about the geometrical characteristic of the system. In order to make sense of seemingly random behavior, statistical theory allow for one to distinguish different levels of complexity in the system. Statistical theory is already being used and is the foundation of the study of turbulent flows. In statistical theory, time averages are used, making transients of the system irrelevant. Over time, the transients will disappear, leading to motion of a fluid particle that exists along the strange attractor. As mentioned before, the trajectory is folded and bonded to the strange attractor [62]. In order to obtain accurate statistics, simulations need to execute until the trajectory of the dynamical system is on the attractor, before beginning to compute statistical quantities.

Given the choice to use statistical theory to quantify outputs of interest of a chaotic system, it is necessary to be able to measure the dimension and other quantities of the system. Knowing this information will allow one to have a better idea of how long the simulation should run and what parameters to use in order to obtain accurate statistics. One can do this with ergodic theory, which allows for one to apply predicative tools for turbulent flows. Ergodic theory assumes the following [65]:

- For almost all initial conditions of an ergodic system, the time average of an interested variable J will reproduce the spatial average of the same interested variable J, meaning that the time averages are invariant (ergodic) under time evolution.

$$\frac{1}{T}\int J(x)\,dt = \frac{1}{X}\int J(x)\,dx \tag{4.5}$$

- In chaotic system, one only has to consider long-term behavior of a system and not worry about transients, thus initial conditions over long periods of time have little effect on the long term output.

- The largest Lyapunov exponent λ is independent of the initial parameters of the system.

These assumptions of ergodicity are useful for adjoint calculations for chaotic systems in later chapters.

4.3 Chaotic Hypothesis

Ergodic theory gives an overview of the relationship between a flow and its statistical quantities [66]. With ergodic theory, more information is needed in order to have enough understanding of chaotic systems in order to propose possible solutions to error estimation of turbulent flows. It is important to understand how chaotic flow is distributed on a smaller scale and how the particles in the flow behave over time. This behavior can first be described by the zeroth law of thermodynamics, which states that a closed control system consisting of a large number of particles will approach equilibrium, assuming at $t = 0$, the control system is at non-equilibrium [66]. Here, equilibrium is defined by the state when the time averages of the system have reached values that can be described by a probability distribution. Another important characteristic to note is how the particles behave up until the equilibrium point. The assumption made here is that macroscopic systems behave as hyperbolic systems, which are defined by three subspaces: one that is exponentially contracting, one that is expanding, and one which is one-dimensional and tangential to the flow direction; or in other words, stable, neutrally stable, and unstable. (Stable manifold, unstable manifold, neutral). These regions correspond to the negative, zero, and positive Lyapunov exponents, which each consist of covariant Lyapunov covariant vectors. Put simply, the chaotic hypothesis says that many high-dimensional chaotic systems behave as if they were hyperbolic or quasi-hyperbolic. Given this hypothesis, all chaotic systems in this thesis are assumed to be hyperbolic.

4.4 Prototypical Governing Chaotic Equations

In this thesis, three different governing equations are introduced and analyzed. For certain parameters, the traditional adjoint method was implemented. The results for these are shown to support the need to find an alternate way to calculate usable adjoints and to understand the impact of discretization errors on statistical outputs of chaotic problems. The three systems are the Lorenz Attractor, the Kuramoto-Sivashinsky 1D partial differential equation, and the 2D compressible Navier-Stokes equations. Each of these systems is assumed to be quasi hyperbolic due to the chaotic hypothesis. Ergodic principles will be used when analyzing the behaviors of these systems.

4.4.1 Lorenz Attractor

The first chaotic system to analyze is the Lorenz Attractor, which is the infamous set of equations that Edward Lorenz worked on when he discovered the unpredictability of chaotic systems [61]. Prototypical of a chaotic system, the Lorenz attractor is a simplified system that models atmospheric convection. The system is defined by three nonlinear

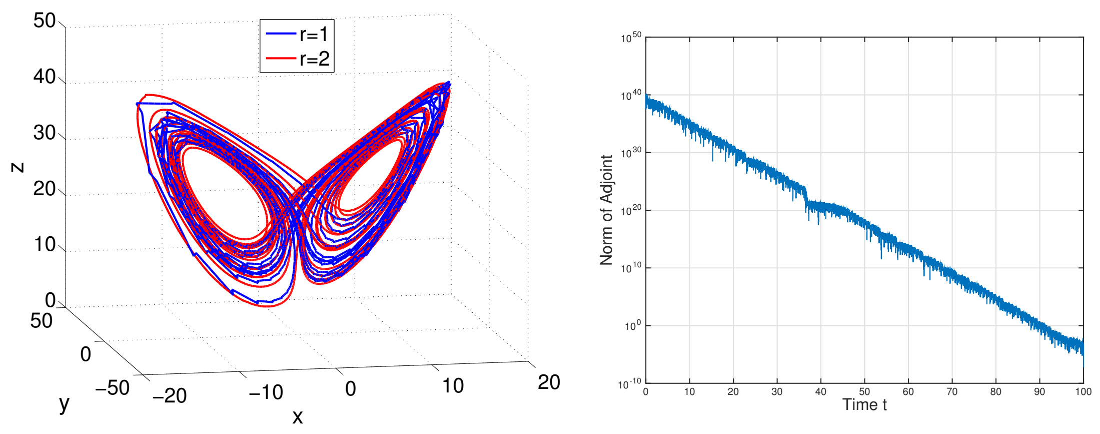


Figure 4.4: Lorenz Attractor orbits solved with DG in time for $r_H = 1$ and $r_h = 2$ and the corresponding norm of the adjoint for $r_h = 2$

equations and three parameters.

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}), \quad \boldsymbol{u} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{bmatrix}. \tag{4.6}$$

In the Lorenz Attractor, the parameter that is usually studied is ρ. Different values of ρ gives different types of attractors.

$$\begin{aligned} 0 \leq \rho \leq 1 &\Rightarrow \text{One stable fixed point attractor} \\ 1 < \rho < 24.74 &\Rightarrow \text{Two stable fixed points} \\ 24.06 < \rho < 31 &\Rightarrow \text{Quasi-hyperbolic strange attractors} \\ 31 < \rho < 99.5 &\Rightarrow \text{Non-hyperbolic quasi-attractors} \end{aligned} \tag{4.7}$$

The most studied set of parameters, σ, ρ, β, is set to 10, 28, $\frac{8}{3}$, respectively. It is important to note that these parameters do give quasi-hyperbolic strange attractors. To discretize the Lorenz Attractor in time, use a discontinuous Galerkin (DG) temporal discretization [67], in which the temporal approximation order, r, is varied. In error estimation, the goal is to compare temporal orders $r_H = 1$ (coarse) and $r_h = 2$ (fine) by integrating the system from $t = 0$ to $t = T$, the final time. The output of interest is the mean z-coordinate,

$$\bar{J} \equiv \frac{1}{T} \int_0^T z \, dt. \tag{4.8}$$

For $T \to \infty$ and exact temporal integration, $\bar{J}$ is a well-defined mean. However, for finite T and inexact temporal integration, statistical and discretization errors pollute $\bar{J}$. In order to gain insight into the magnitudes of these errors numerically for the Lorenz system, T is varied and the statistics for temporal orders $r = 1$ are compared with those for $r = 2$. Figure 4.4 illustrates sample trajectories obtained using $\Delta t = .05$ and $T = 20$, starting with the same initial conditions. The two solutions are initially close but then drift apart. This is expected in a chaotic system, and of interest is how the discretization error affects the desired statistical output. Additionally, the unsteady adjoint for the Lorenz Attractor is calculated and shown in Figure 4.4. In the graph, one can see that the norm of the adjoint diverges backwards in time for both temporal orders, consistent with that of a chaotic system. Over the studied time horizon, the magnitude of the norm of the adjoint reaches on order of 10^{40}, which makes the unsteady adjoint calculation useless for chaotic systems.

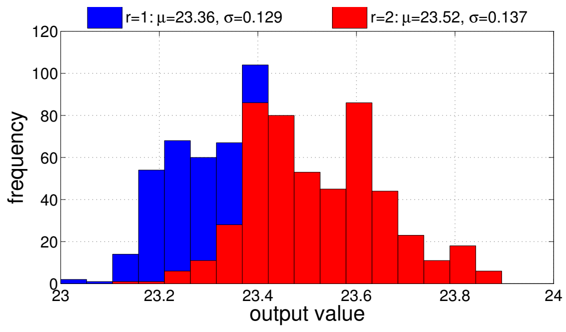
Next, understanding how the Lorenz Attractor behaves over long periods of time is important in order to begin thinking of ways to overcome the failure of the traditional unsteady adjoint for chaotic systems. Given the assumption that the chaotic systems of interest are ergodic, one can analyze the statistical results of the time average output of the chaotic system. Long integration times and high-fidelity temporal integration reduce errors but add computational expense, hence quantifying the relative importance of each error source is of interest in ensuring optimal-efficiency calculations.

Figure 4.5 shows a quantitative comparison of output differences for $\Delta t = .05$ and an ensemble of 500 different random initial conditions. A burn-time of $0.2T$ is used to allow the system to settle around the attractor for each choice of initial conditions.

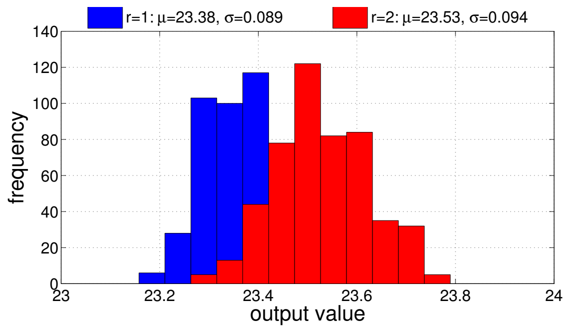
As expected, the statistics improve (ensemble standard deviation drops) as T increases, but there is a persistent discrepancy between the two temporal accuracy orders and the ensemble means converge to different values. The difference is similar for all T and larger than the statistical errors measured by the ensemble standard deviations, which indicates that this case warrants higher fidelity time integration in lieu of longer integration times. For more complex simulations, we cannot afford such a detailed convergence study, and we need to make this decision based on more efficient error estimates. Furthermore, for a discretization of partial differential equations, numerical error arises from both spatial and temporal discretizations, and an accurate distinction between the two is vital to efficiency and convergence of adaptation.

4.4.2 Kuramoto-Sivashinsky

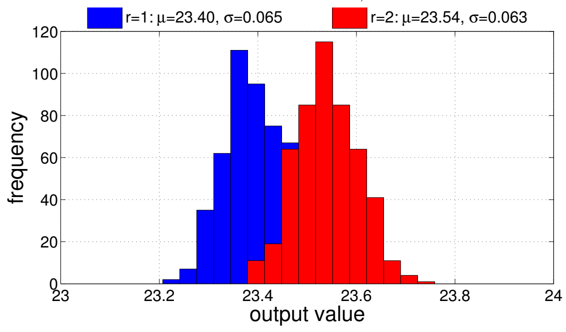
Another chaotic system is the Kuramoto-Sivashinsky (KS) 1D PDE, which was derived to model the Belousov-Zhabotinsky reactions in three dimensional space [68]. In addition, KS was derived to model small thermal diffusive instabilities in laminar flame fronts by



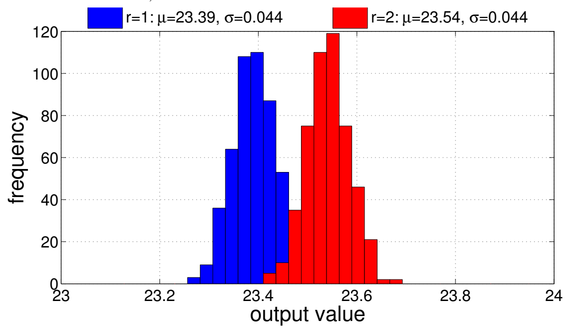
a) $T = 50$



b) $T = 100$



c) $T = 200$



d) $T = 400$

Figure 4.5: Lorenz Attractor: ensemble statistics of outputs computed using different integration times, T, and DG-in-time integration order, r. Each ensemble contains 500 experiments, and μ and σ_μ refer to the ensemble mean (more accurate with increasing r) and standard deviation (smaller with increasing T).

Sivashinsky. In the past, KS has been used to model small perturbations from a reference Poseuille flow of a film layer on an inclined plane. It was first dealt with by Kuramoto et. al in 1976 when Kuramoto began investigating many different types of instability and ordered structure that he found in equilibrium stead states [69]. The solution to KS exhibits local stabilities that are quickly ruined by disturbances that repel and eventually produce new localized patterns that are again stable for short periods of time. At the moment there is no explanation for how these local structures are connected to each other (similar to the way Lorenz tried to make sense of bumps and patterns in his weather model), making Kuramoto-Sivashinsky a natural chaotic system to use for error estimation before dealing with more applied problems. The KS equation for a primal scalar state $u(x, t)$ reads

$$\frac{\partial u}{\partial t} = -\alpha \frac{\partial u}{\partial x} - u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \beta \frac{\partial^4 u}{\partial x^4}, \tag{4.9}$$

where the role of the linear advection term ensures that the KS solutions exhibit ergodic behavior, which is essential for error estimation [70]. Along with the advection term, KS is characterized by a second-order unstable diffusion term, a fourth-order stabilizing viscosity term, and a nonlinear burgers term. Computations have confirmed that the norm of the diffusion term $\|\nabla u(t)\|^2$ remains bounded and that the linearly unstable modes are stabilized by the strong non linear coupling of the burgers term, $\frac{1}{2}\|\nabla u\|^2$.The initial conditions consists of a delta distribution,

$$u(x, t = 0) = u_0(x), \tag{4.10}$$

$$u_0(x) = \begin{cases} 1, & x = \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}. \tag{4.11}$$

and a burn time, t_{burn} is used to verify that the state after the burn time is on the strange attractor. The homogeneous Dirichlet boundary conditions are

$$u(x = X_0, t) = u(x = X_f, t) = \frac{\partial u}{\partial x}(x = X_0, t) = \frac{\partial u}{\partial x}(x = X_f, t) = 0. \tag{4.12}$$

where u is the state and is referred to as the primal solution. Dirichlet boundary conditions are required for KS to exhibit chaotic behavior. The output of interest is the spatial and temporal average of the state,

$$\overline{J}_{KS} = \lim_{T\to\infty} \frac{1}{T} \int_{T_0}^{T_f} J_{KS}(t)\, dt, \tag{4.13}$$

where

$$J_{KS} = \frac{1}{X} \int_{X_0}^{X_f} u(x,t)\, dx \tag{4.14}$$

where $T = T_f - T_0$ and $X = X_f - X_0$. This integral is evaluated using quadrature in space and trapezoidal time integration, since BDF2 is used for the time discretization. According to ergodic theory,

$$\lim_{T\to\infty} \overline{J}_{KS} = J_{KS} \tag{4.15}$$

There are different forms of the KS equation and different researchers have quantified different models of KS. For this particular case, adding an advection term, allows one to use ergodic principles to help predict how KS simulations will behave. However, an extensive study into how the KS model behaves with different parameters is necessary. Understanding how parameters affect the simulation will give different versions of the model to use for error estimation. Unlike the Lorenz Attractor, where it is accepted that $\rho = 28$ gives a quasi-hyperbolic attractor, this knowledge is unknown for KS. Thus, a mini investigation is required before proceeding. This small study will also give an idea of how long the burn time should be in order for the trajectory to reach the strange attractor.

4.4.2.1 Kuramoto-Sivashinsky Parameters Analysis

KS exhibits unique trajectories for different advection speed and the ”super viscosity”, (α, β), which can be seen in $x - t$ contour plots in Figure 4.6- 4.9 for ten different cases with zero burn time. To simplify the process, only the advection speed and the super viscosity will be analyzed in the following plots. These results provide some insight into which trajectory would make an adequate prototype for research in error estimation. In addition, these results help indicate what the ideal burn time (t_{burn}) should be for each set of parameters. The burn time should be long enough such that the initial conditions have as little effect on the statistical output as possible. The area where the initial conditions have the most effect on the statistical output is seen at the beginning with large zero value regions in the $x - t$ contour plots. The time-average output $\overline{\boldsymbol{J}}$ for all the runs is plotted in Figure 4.10, showing how it is affected by the magnitudes of the advection speed and the super viscosity. Figure 4.6 shows that with $\alpha = 1, \beta = 1$, the trajectory is always heavily influenced by the initial conditions in time. This type of behavior is caused by the high advection to super viscosity ratio, which extends the time required for the trajectory to reach a steady-state time-average output. This can be seen further in Figure 4.10 where the time-average solution for $p = 2$ and $p = 3$ differ considerably compared to the other cases. This reveals that the trajectory is possibly not ergodic, making it a poor choice as a prototypical trajectory

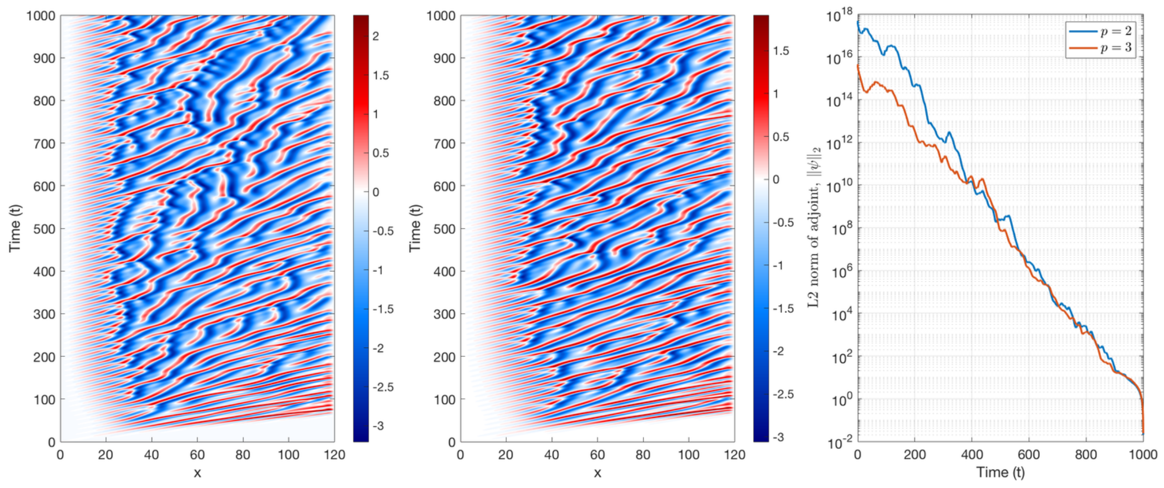
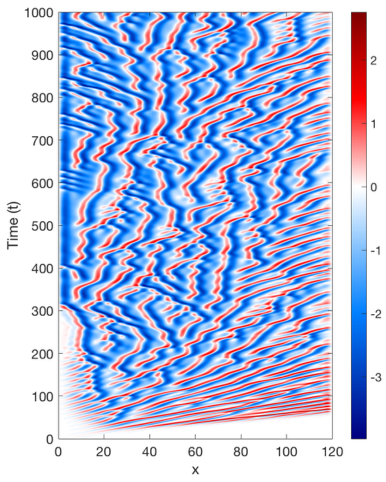

(a) $\alpha = 1,\ \beta = 1,\ p = 2$ (b) $\alpha = 1,\ \beta = 1,\ p = 3$ (c) $\alpha = 1,\ \beta = 1$

Figure 4.6: KS: Advection speed $\alpha = 1$, super viscosity rate $\beta = 1$ trajectories.

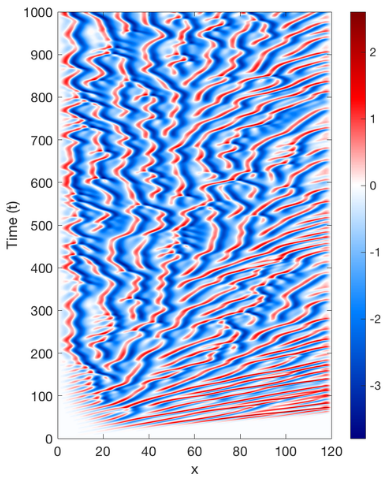
for numerical adjoint calculations.

The next set of cases investigates the effect of the super viscosity on the trajectories. This consists of keeping $\alpha = 1$ constant and varying β. This will give insight on how the magnitude of the trajectories behaves when the super diffusive characteristics of the system decreases. Figure 4.7 shows the trajectory for $\beta = 0.75$. The plots compared to Figure 4.7(a) and 4.7(b) show that the trajectories expand for $p = 3$ and $p = 2$ throughout the entire space with time starting at $T = 300$. The trajectories or coherent structures appear to be given the chance to form themselves now that the influence of the fourth order diffusion term is less than before. The minimum burn time for this case is $t_{\text{burn}} \approx 300$. The L_2 norm of the adjoint for $p_h = 3$ and $p_H = 2$ in Figure 4.7(c) shows that the slopes for each of the temporal discretization do not match precisely throughout the entire time simulation.

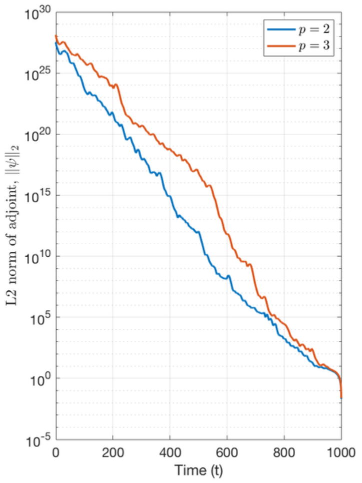
Figures 4.7(d) and 4.7(e) show the trajectories for $\alpha = 1, \beta = 0.5$. One thing to notice is that the thickness of the solution coherent structures has decreased, leading to higher average trajectory magnitudes and higher oscillations, which could mean that the system exhibits stronger chaotic behavior. Compared to Figure 4.6, the trajectory is able to reach a situation where the initial conditions have little influence on the time-averaged output. Compared to Figure 4.7(a) and 4.7(b), the trajectories span the entire one dimensional space sooner at $T \approx 100$, meaning that the minimum t_{burn} can be set to $t_{\text{burn}} \approx 150$, lower than before. From this we see that the lower β leads to lower super viscosity, which dilutes the higher advection speed resulting a system that is possibly more ergodic. Figure 4.7(f) shows that the slopes of the norm of the adjoint match closely for $p_H = 2$ and $p_h = 3$ than



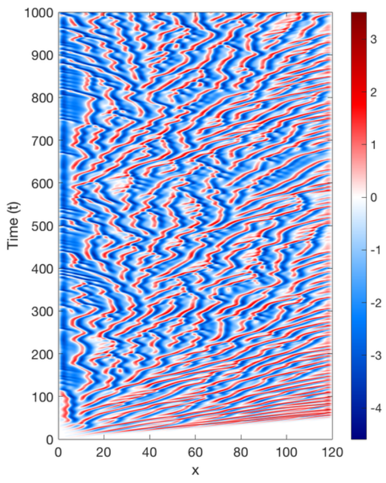
(a) $\alpha = 1,\ \beta = 0.75,\ p = 2$



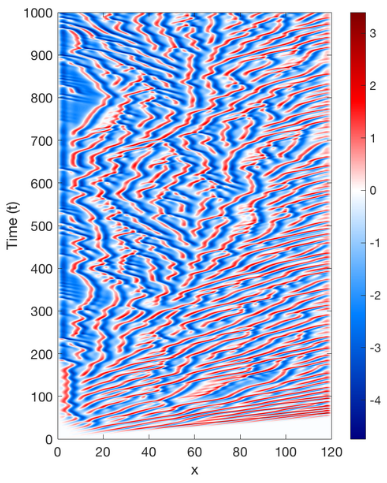
(b) $\alpha = 1,\ \beta = 0.75,\ p = 3$



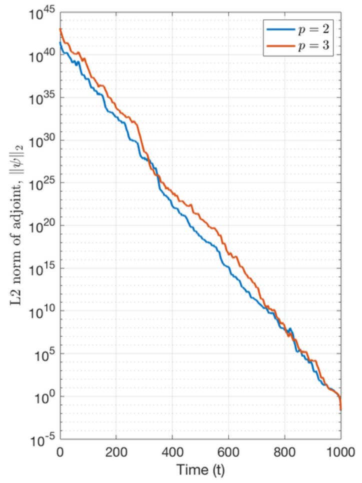
(c) $\alpha = 1,\ \beta = 0.75$



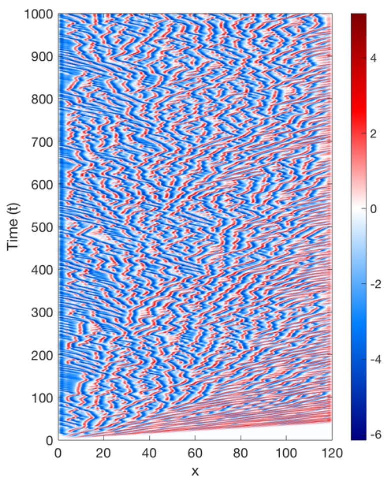
(d) $\alpha = 1,\ \beta = 0.5,\ p = 2$



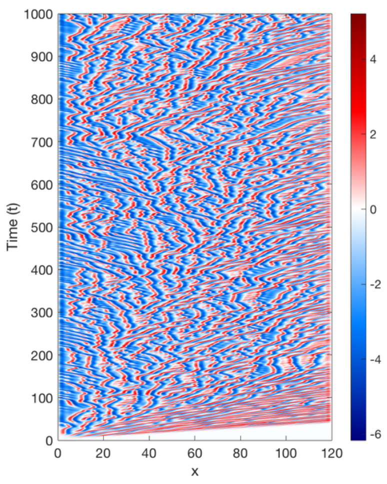
(e) $\alpha = 1,\ \beta = 0.5,\ p = 3$



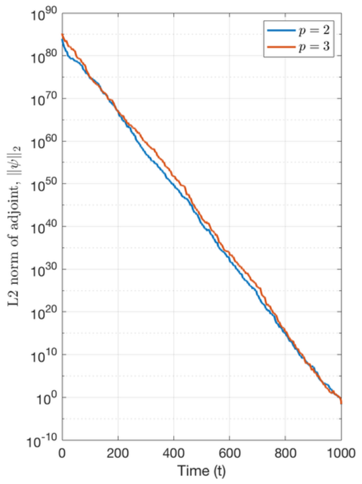
(f) $\alpha = 1,\ \beta = 0.5$



(g) $\alpha = 1,\ \beta = 0.25,\ p = 2$



(h) $\alpha = 1,\ \beta = 0.25,\ p = 3$



(i) $\alpha = 1,\ \beta = 0.25$

Figure 4.7: KS: Advection speed $\alpha = 1$, changing super viscosity rate β trajectories

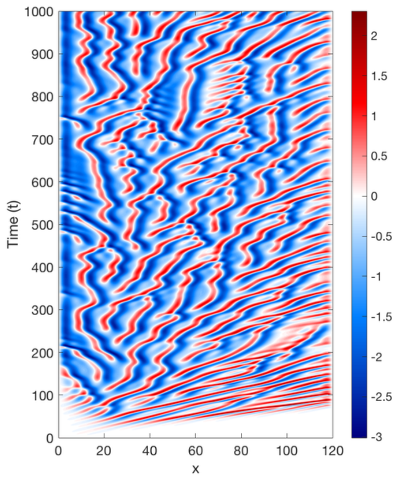
when $\beta = 0.75$.

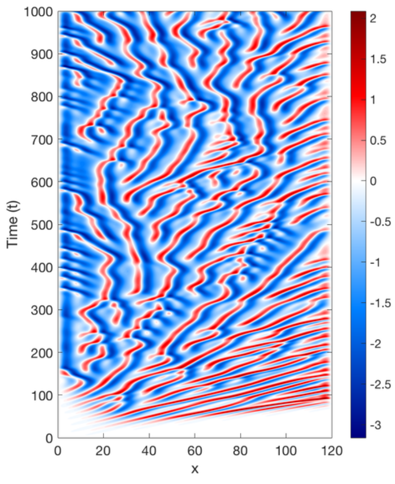
Next, the super viscosity is further decreased to $\beta = 0.25$ to see its more extreme effect on the trajectory. Figure 4.7(g) and 4.7(h) shows the solution for $\alpha = 1, \beta = 0.25$. The thickness of the coherent structures has decreased further implying the systems' stronger chaotic behavior and the effect of the initial conditions on the overall time-average output is decreased further. The minimum burn time according to the results is $t_{burn} \approx 50$, meaning the simulations do not have to be executed as long to reach the statistically steady-state time-averaged outputs even though ideally $T \to \infty$. This case shows that we have a stronger ergodic behavior compared to when $\beta = 0.5, 0.75$. Figure 4.7(i) shows that the norm for the adjoint is nearly the same for $p = 2$ and $p = 3$. The next set of cases to investigate involves the effect of the advection speed on the trajectories. Note that the higher the advection speed, the more ergodic the system may be. Figure 4.8 shows a collection of cases for varying α and constant β.

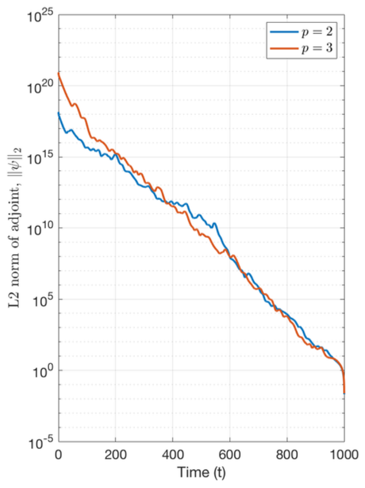
Figure 4.7(g) and 4.7(h) show the trajectories for $\alpha = 0.75$ and $\beta = 1$. Compared to Figure 4.6, the trajectories expand the entire space quickly, solidifying that along with the advection speed, the super viscosity does have important effect on ergodicity as well. This system's initial conditions have less effect than even the case where $\alpha = 1$ and $\beta = 0.75$ in Figure 4.7(a) and 4.7(b). This observation is shown as well in Figure 4.7(i) where the norms of the adjoint for $p = 2$ and $p = 3$ match up better than when $\alpha = 1$ and $\beta = 0.75$.

Figure 4.8(a) and 4.8(b) show results for $\alpha = 0.5$. The system with these parameters will theoretically be less ergodic. Like that in Figure 4.7(g) and 4.7(h), the initial conditions for $p = 2$ and $p = 3$ become "washed out" in a reasonable amount of time. The lower advection speeds do not overcome the super viscosity. For this trajectory, given how long the zero regions exist in the $x - t$ contour plots, the minimum burn time should be $t_{\text{burn}} \approx 125$. From these results, decreasing the advection speed, gives a less ergodic system compared to when $\alpha = 1$. This behavior can be seen in Figure 4.8(c) where the norms of the adjoints for the different discretization orders are more different than before. Figure 4.10 shows that when decreasing α, the overall time-average output steadies to a value that is larger than before, implying the less ergodic behavior.

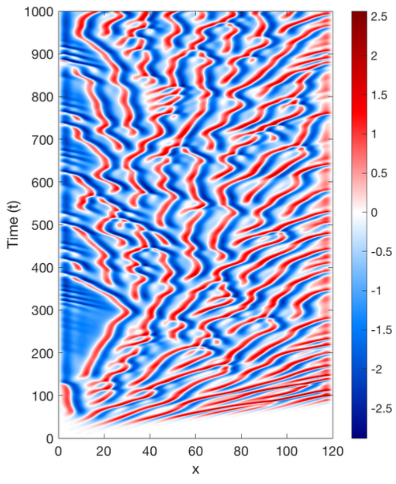
Figure 4.8(d) and 4.8(e) shows that by further decreasing α to $\alpha = 0.25$, t_{burn} can be set at a lower minimum $t_{\text{burn}} \approx 100$ allowing some savings in the computational costs in creating the primal solution. Figure 4.10 shows that compared to the $\alpha = 0.5$ and $\alpha = 0.75$ case, the $\alpha = 0.25$ trajectory reaches a steady state time-averaged solution that is most different than the rest of the parameters tested making it possibly less ergodic than the other cases. Figure 4.8(f) shows similar results to that of Figure 4.8(c). Overall, it appears that decreasing α leads to higher time-average outputs.

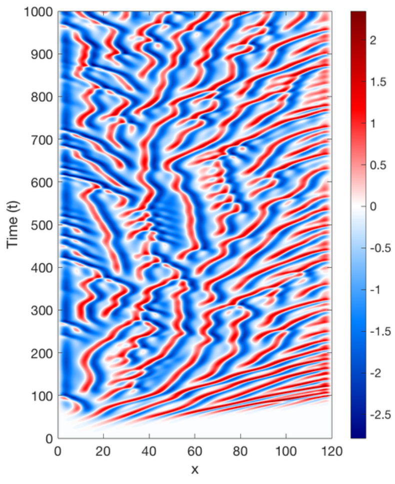

(a) $\alpha = 0.75,\ \beta = 1,\ p = 2$

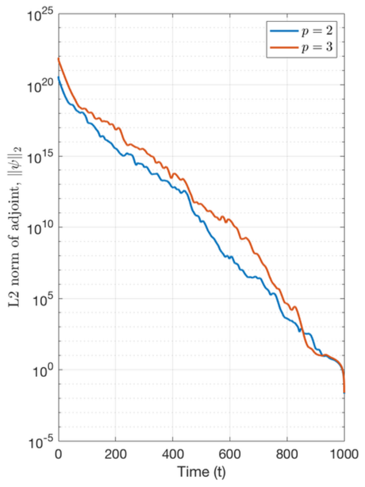

(b) $\alpha = 0.75,\ \beta = 1,\ p = 2$



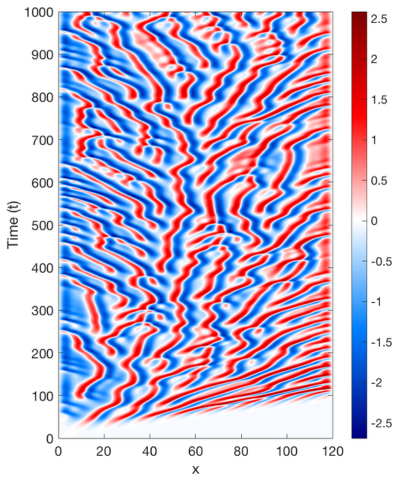
(c) $\alpha = 0.75,\ \beta = 1,\ p = 2$

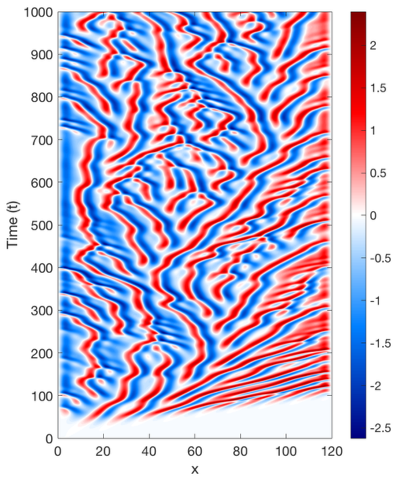

(d) $\alpha = 0.5,\ \beta = 1,\ p = 2$

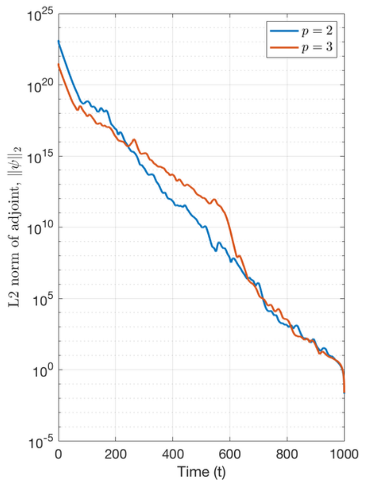

(e) $\alpha = 0.5,\ \beta = 1,\ p = 2$



(f) $\alpha = 0.5,\ \beta = 1,\ p = 2$

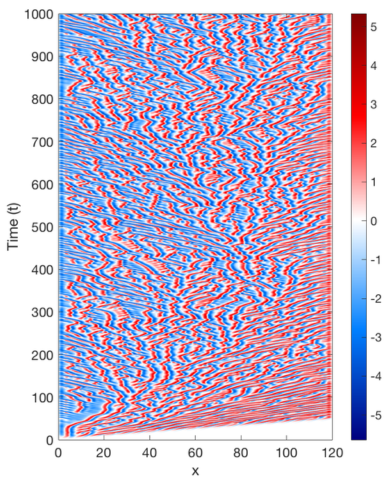

(g) $\alpha = 0.25,\ \beta = 1,\ p = 2$


(h) $\alpha = 0.25,\ \beta = 1,\ p = 2$

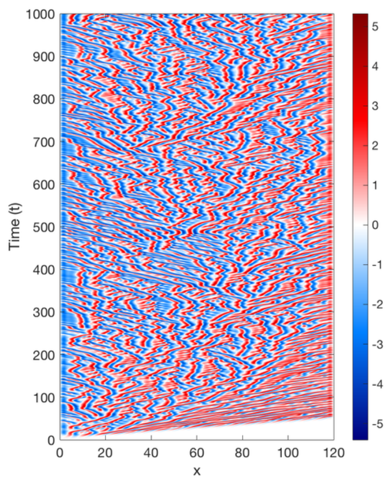


(i) $\alpha = 0.25,\ \beta = 1,\ p = 2$

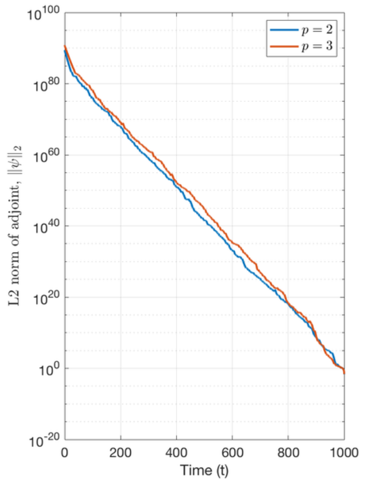
Figure 4.8: KS: Changing advection speed α, super viscosity rate $\beta = 1$ trajectories



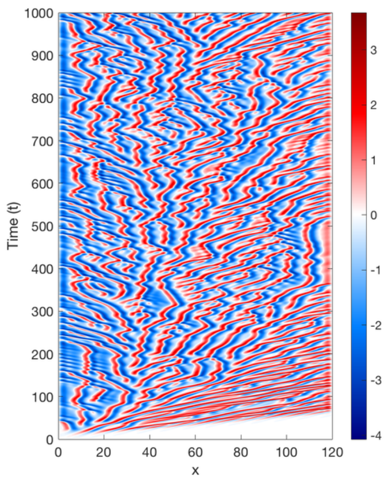
(a) $\alpha = 0.25,\ \beta = 0.25,\ p = 2$



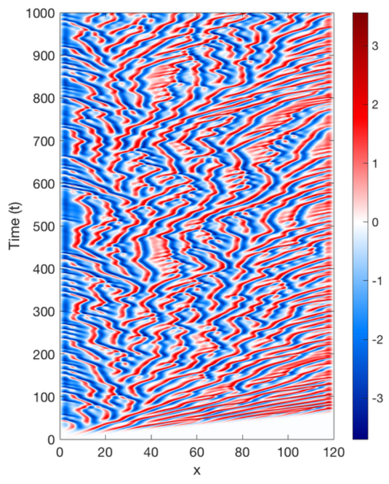
(b) $\alpha = 0.25,\ \beta = 0.25,\ p = 3$



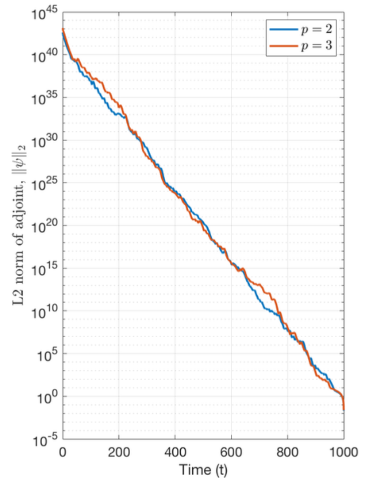
(c) $\alpha = 0.25,\ \beta = 0.25$



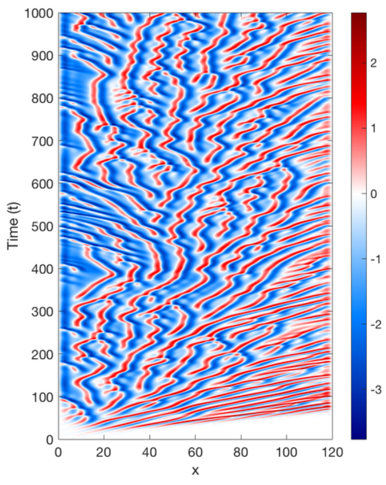
(d) $\alpha = 0.5,\ \beta = 0.5,\ p = 2$



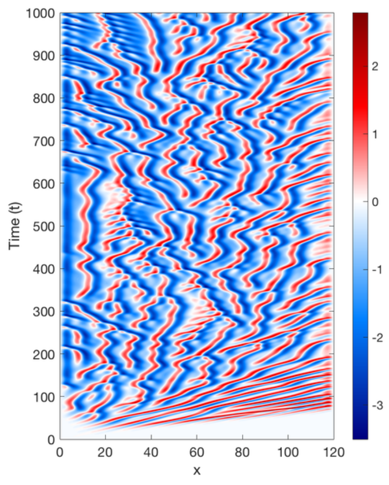
(e) $\alpha = 0.5,\ \beta = 0.5,\ p = 3$



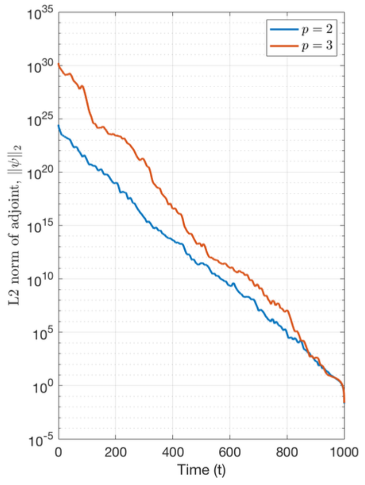
(f) $\alpha = 0.5,\ \beta = 0.5$



(g) $\alpha = 0.75,\ \beta = 0.75,\ p = 2$



(h) $\alpha = 0.75,\ \beta = 0.75,\ p = 3$



(i) $\alpha = 0.75,\ \beta = 0.75$

Figure 4.9: KS: advection speed and super viscosity speed $\frac{\alpha}{\beta} = 1$ trajectories

Lastly, it is important to investigate what happens when the ratio of α and β is one, but the magnitudes of these parameters are varied. Figure 4.9(a) and 4.9(b) show results for $\alpha = 0.25$ and $\beta = 0.25$, which looks similar to the case where $\alpha = 1$ and $\beta = 1$. The trajectories or warms for both $p_H = 2$ and $p_h = 3$ are considerably thinner than that of the last set of runs and the norms of the adjoints in Figure 4.9(c) are very close in values. In Figure 4.10 the time-average output for $\alpha = 0.25$ and $\beta = 0.25$ is drastically different compared to that of $\alpha = 1$ and $\beta = 0.25$ even though the thickness of the "worms" are similar in magnitude. It appears compared to the $\alpha = 1$ and $\beta = 0.25$ case, that the time average output for $\alpha = 0.25$ and $\beta = 0.25$ is more heavily influenced by the initial conditions, implying that is in less ergodic-there is large hump at $T \approx 80$. This leads to the conclusion that higher advection speed does lead to a more ergodic system.

Figure 4.9(d) and 4.9(e) show results for $\alpha = 0.5, \beta = 0.5$. These results further show that both the magnitudes of α and β, and the ratio of these parameters affect the solution behavior. Compared Figure 4.9(a) and 4.9(b), the coherent structures are thicker and the time-average output in Figure 4.10 is less influenced by the initial conditions, which is to be expected with a higher advection speed. The norm of the adjoint in Figure 4.9(f) show approximate agreement and the minimum burn time is low, but higher than when $\alpha = 0.25$ and $\beta = 0.25$, $t_{\text{burn}} \approx 75$.

Figure 4.9(g) and 4.9(h) show results for $\alpha = 0.75$ and $\beta = 0.75$. The coherent structures are thicker than before and the minimum burn time is higher than the last two cases. The time average output for this case in Figure 4.10 shows that its value is not highly influenced by its initial conditions- no bump. However, 4.9(i) shows some interesting results in the adjoint for these set of parameters. the norm of the adjoints for $p = 2$ and $p = 3$ actually diverge away from quickly. By $T = 0$ the magnitudes of the norm of the adjoints differ on the order of $\delta\|\psi\|_2 \approx 10^5$. This is not ideal and shows that this system is even less ergodic than before even though α is higher now.

4.4.2.2 Summary on Kuramoto Sivashinsky Analysis

Based on the previous study, a good prototypical chaotic system is one that is at least quasi-hyperbolic and ergodic. It is essential as well that the chosen parameters for the system gives a heavily chaotic system and a low minimal required burn time. Over a long period of time, the output of such system needs to reach statistically-converged time-averaged outputs as quickly as possible. The best parameters that fulfill these requirements are when α and the ratio of α to β is as high as possible. The case that best fulfills all of these requirements is the one presented in Figure 4.7(g)- 4.7(i). For the rest of this paper, all results are for $\alpha = 1, \beta = 0.25$.

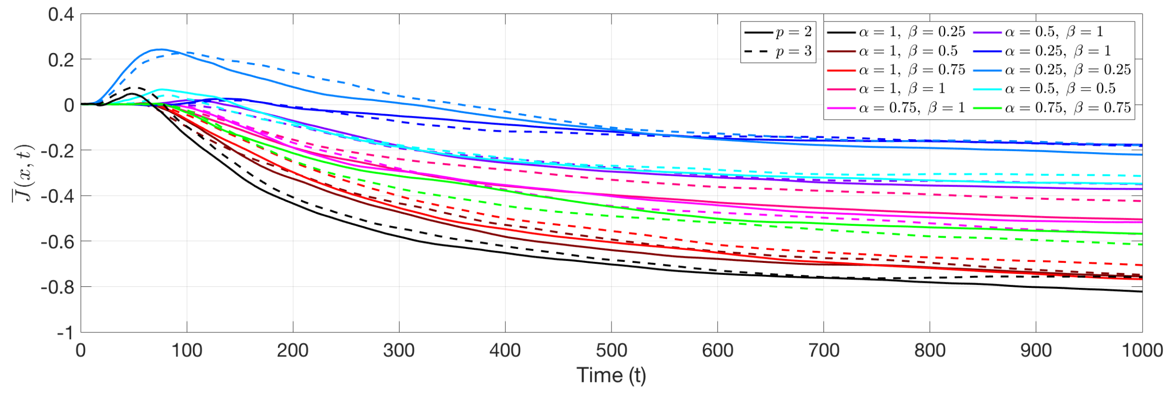


Figure 4.10: KS: Time-average output for different KS trajectories and spatial interpolation orders.

4.4.3 High Reynolds Number Compressible Navier-Stokes

For an application in aeronautical engineering, one can consider the compressible Navier-Stokes equations, which govern many flow processes. These equations describe the conservation of mass, momentum, and energy of a viscous fluid. For this research, only the two-dimensional compressible Navier-Stokes equations ($d = 2$) are considered. The particular case of interest is a NACA 0012 airfoil in viscous laminar flow at $M = 0.2$, $Re = 10^4$ at an angle of attack of $\alpha = 8°$. Unlike the Lorenz Attractor and the Kuramoto-Sivashinsky 1D PDE, this particular simulation is not chaotic; however, it is a pseudo-periodic unsteady case, which suffers the same issues as a chaotic case in terms of its adjoint calculations. Meaningful adjoint calculation for chaotic flows will help rectify the issues with adjoint calculation for this case as well.

The compressible Navier-Stokes equations are:

$$\begin{aligned} \text{Conservation of mass:} \quad & \partial_t \rho && + \partial_j(\rho u_j) && = 0 \\ \text{Conservation of momentum:} \quad & \partial_t(\rho u_i) && + \partial_j(\rho u_i u_j + p\delta_{ij}) && = \partial_j \tau_{ij} \\ \text{Conservation of energy:} \quad & \partial_t(\rho E) && + \partial_j(\rho u_j H) && = \partial_j(u_j \tau_{ij} + q_i) \end{aligned} \tag{4.16}$$

where i and j index the spatial dimension, ρ is the density, u_i and u_j are the components of velocity, and E is the total energy per unit mass. The viscous shear and normal stresses for a Newtonian fluid are

$$\tau_{ij} = \mu(\partial_i u_j + \partial_j u_i) + \delta_{ij}\lambda\partial_m u_m, \tag{4.17}$$

where μ is the dynamic viscosity and λ is the bulk viscosity. The heat transfer term is defined as

$$q_i = \kappa_T \partial_i T. \tag{4.18}$$

where κ_T is the thermal conductivity and δ_{ij} is the Kronecker delta function,

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases} \tag{4.19}$$

The important physical quantities for air are

$$\begin{aligned}
\text{Dynamics viscosity :} \quad & \mu && = \mu_{\text{ref}} \left(\frac{T}{T_{ref}}\right)^{1.5} \left(\frac{T_{ref}+T_s}{T+T_s}\right) \\
\text{Bulk viscosity coefficient :} \quad & \lambda && = -\frac{2}{3}\mu \\
\text{Thermal conductivity :} \quad & \kappa_T && = \frac{\gamma \mu R}{(\gamma-1)Pr} \\
\text{Specific-heat ratio :} \quad & \gamma && = 1.4 \\
\text{Prandtl number :} \quad & Pr && = 0.71 \\
\text{Gas constant for air :} \quad & R && = 287.05\ J/(kg \cdot K).
\end{aligned} \tag{4.20}$$

The output of interests for the compressible Navier-Stokes case is the time average of the drag coefficient. Figure 4.11 shows the computational mesh and a snapshot of the unsteady flow-field for this governing equation. The boundary conditions are full-state on the farfield

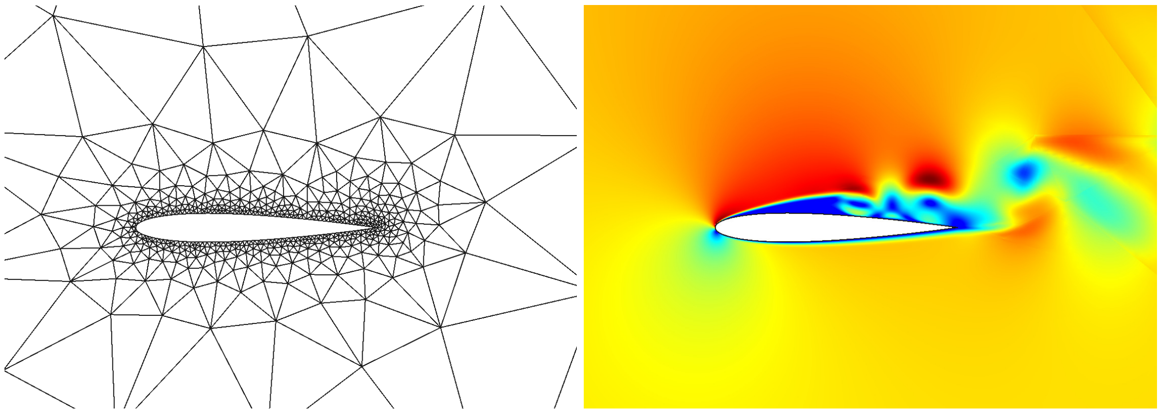


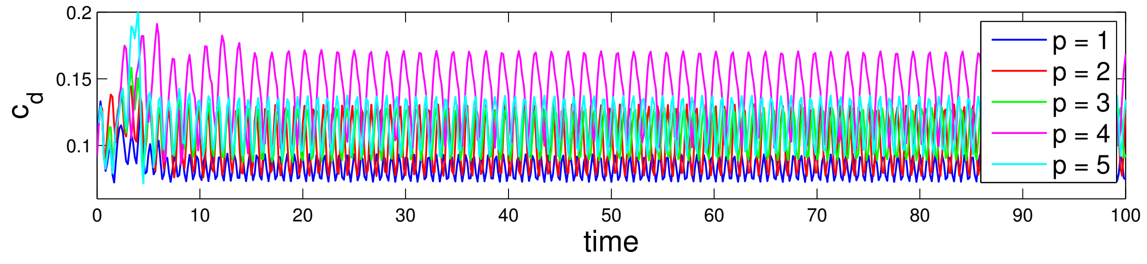
Figure 4.11: NACA 0012: M = 0.2, $Re = 10^4$, $\alpha = 8^\circ$, computational mesh and flow-field snapshot

and adiabatic no-slip wall on the airfoil. The state is initialized to free-stream and advanced forward for a time length T using third-order diagonally-implicit Runge-Kutta (DIRK3) time marching with $\Delta t = 0.2$. The farfield is approximately 100 chord lengths away from the airfoil, and the initial solution approximation order is uniform in space at $p = 1$ or

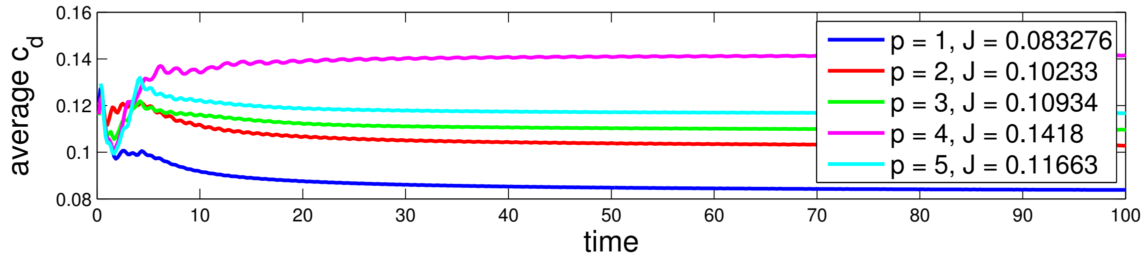
$p = 2$. The free-stream state is initialized to

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} = \begin{pmatrix} 1 \\ \cos(\alpha) \\ \sin(\alpha) \\ \frac{1}{\gamma(\gamma-1)M^2} + \frac{1}{2} \end{pmatrix} \tag{4.21}$$

Figure 4.12 shows the time histories of the drag coefficient and the convergence of its time-averaged value for different spatial orders p. The time simulation length for this particular case is $T = 100$. As the mesh is coarsened, discretization errors become quite large, even at moderate orders. Similarly as in the Kuramoto-Sivashinsky case, the effect of the discretization error on the output is not predictable and converges to different time-average values. For example, $p = 4$ behaves as an outlier in between $p = 3$ and $p = 5$. Calcu-



(a) Drag coefficient time histories



(b) Convergence of the average drag coefficient

Figure 4.12: NACA 0012: M = 0.2, $Re = 10^4$, $\alpha = 8^\circ$, time histories and average drag coefficient convergence for uniform order refinement. ”Under-resolved”

lation of the traditional adjoint of the pseudo chaotic Navier-Stokes equations in viscous flow case will result in an adjoint field that deteriorates quickly and that increases exponentially backwards in time given a instantaneous chaotic drag output seen in Figure 4.13. This makes it not possible with current techniques to use output-based error estimation for this simulation. The adjoint calculations show that at high Reynolds number, the solution is highly sensitive to its initial condition and further calls for estimating the impact of numerical error, i.e., discretization error on statistical outputs, because of its chaotic-like behavior.

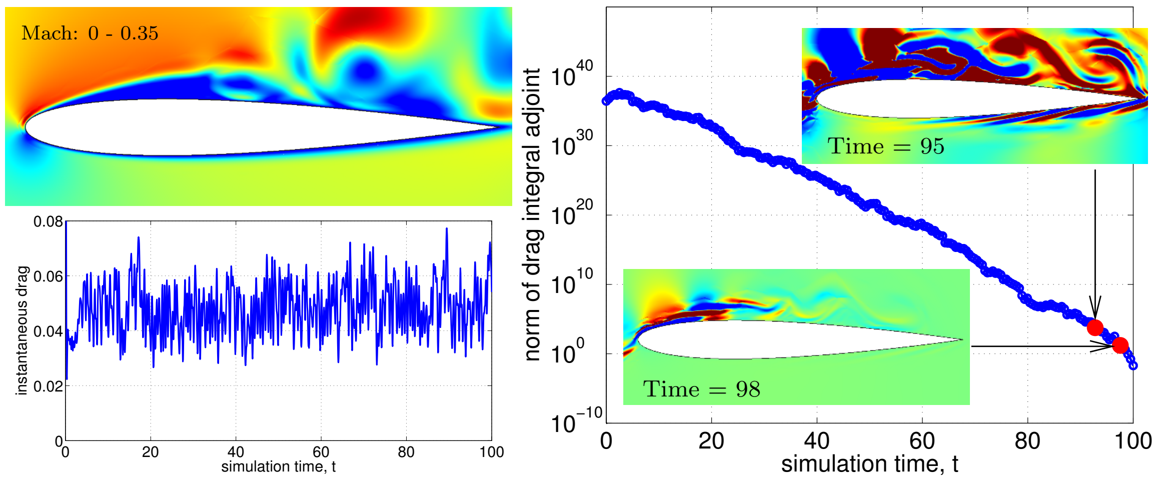


Figure 4.13: NACA 0012, $M = 0.2$, $Re = 210^4$, $\alpha = 8°$, ill-conditioning of average-drag prediction manifests itself through an unstable adjoint; i.e. the output is highly sensitive to initial conditions

4.5 Summary

Chapter III introduced the adjoint and went into details explaining how to traditionally calculate both its discrete and continuous versions for both the steady and unsteady cases. Due to the adjoints' useful characteristic as the sensitivity that can be used for any input perturbation, it has been used successfully for error estimation. However, for chaotic flows, traditional adjoint calculations break down, making error estimation difficult to implement.

The purpose of this chapter was to give insight into why the unsteady traditional adjoint breaks down and how it does so for real chaotic cases. These concepts were introduced via chaos theory which classifies a system as chaotic if it is prone to exhibit aperiodic long term random trajectories, is deterministic, and is highly sensitive to its initial conditions. Next, the butterfly effect was looked at in more detail by considering the Lyapunov exponents and the covariant Lyapunov vectors, which were shown to influence the trajectory of a chaotic system. The system's non predictive behavior was shown to be caused by positive Lyapunov exponents. The Lyapunov exponent was calculated numerically by looking at the exponential growth of a perturbed trajectory with slightly different initial conditions. The slope of the norm of the adjoint gives the magnitude of the largest positive Lyapunov exponent.

Several tools were introduced that could be used to analyze chaotic systems and used to develop a new chaotic adjoint technique to replace its traditional formulation. The first tool was the chaotic hypothesis, which assumes that many high-dimensional chaotic systems behave as if they were quasi-hyperbolic. This means that it can be assumed that most chaotic systems are made of three subspaces, one that is contracting, one that is expanding, and one that is neutral and tangential to the flow direction. By assuming most chaotic systems are hyperbolic or quasi-hyperbolic, the shadowing lemma for example, which will be introduced in the next chapter, can be used for adjoint method development.

The last tool is statistical theory, which allows for one to make sense of seemingly random behaviors. Without statistical theory, there is is not enough information about the geometrical characteristic of the system in order to begin to formulate solutions. By using statistics - i.e. studying time average outputs after running for long simulation times- one can better distinguish different levels of complexity in the system. This is an important concept for verification for error estimation of chaotic systems. Statistical theory leads to the theory or ergodicity, which allows for one to assume that over long periods of time ($T \to \infty$), certain assumptions can be made. The most important assumptions are that the initial conditions over long periods of time have little impact on the time-average output and that the time-average output of the system will eventually be equal to the spatial average output. Ergodic theory along with the two previous tools described will allow for one to attempt to provide a solution to calculating chaotic adjoints for output-based error estimation and other applications.

Lastly, this chapter introduced three different prototypical chaotic equations that will be used for the rest of the thesis. The three chaotic systems are the Lorenz Attractor, the one dimensional ergodic Kuramoto-Sivashinsky system, and the pseudo-chaotic high Reynolds number compressible Navier-Stokes equation. It was shown for all three systems that the traditional unsteady adjoint increases exponentially backwards in time, resulting in unusable adjoints for output-based error estimation. The following chapters will propose a method that can calculate accurate chaotic adjoints.

CHAPTER V

The Least Squares Shadowing-Adjoint Weighted Residual Method

In Chapter IV, chaos theory was introduced in order to explain why traditional unsteady adjoint-based techniques fail for chaotic flows. When calculated backwards in time, the adjoints increased exponentially, making them unusable for sensitivity calculations, or error estimation. This divergent behavior is caused by the chaotic system's sensitivity to initial conditions. The system was then described by its Lyapunov exponents, which is a quantify that defines that rate of separation between a reference trajectory and a trajectory with perturbed initial conditions. When perturbing the initial conditions, the new trajectory of a chaotic system will not in general stay close to the original trajectory, as seen in Figure 4.1. Note that a perturbed trajectory of a chaotic system will stay close to its reference trajectory for a short period time before diverging away. This behavior hinders sensitivity and error calculations based on linearization. Modifications to the traditional continuous adjoint method will be introduced in this chapter to enable calculation of meaningful chaotic adjoints that can be used for output-based error estimation and mesh adaptation.

One such technique that has been studied extensively by Wang et. al, uses optimization techniques to find an alternative perturbed trajectory that will not diverge away from the reference trajectory [70, 71, 72, 73]. This particular technique is referred to as the Least Squares Shadowing method and replaces the traditional adjoint method in the output-based error estimation routine as shown in Figure 5.1. The Least Squares Shadowing method (LSS) has been used successfully to compute adjoint sensitivities for chaotic systems. In this paper, several assumptions and theories will reveal why LSS can be used. Next, a derivation of the LSS method similar to that of the traditional continuous adjoint method will be presented. Once the LSS adjoint equations have been found, the chapter will go over how to practically implement the LSS method, which uses a technique called, the LSS checkpoint design. The LSS checkpoint design technique solves the adjoint for n_{seg} time windows. The time simulation is split into n_{Seg} time windows in order to prevent the

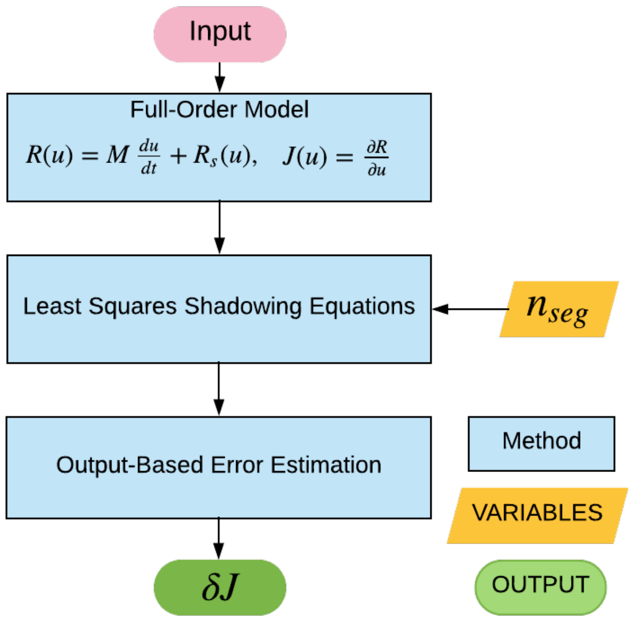

Input
Full-Order Model
$R(u) = M\frac{du}{dt} + R_s(u), \quad J(u) = \frac{\partial R}{\partial u}$
Least Squares Shadowing Equations
n_{seg}
Output-Based Error Estimation
Method
VARIABLES
δJ
OUTPUT

Figure 5.1: Flowchart of the Least Squares Shadowing-Adjoint Weight Residual Method for Output-Based Error Estimation.

adjoint from growing exponentially backwards in time. After the algorithms for LSS are presented, LSS is extended to output-based error estimation to formulate the Least Squares Shadowing-Adjoint Weighted Residual Method.

Finally, results for the implementation of LSS for the Lorenz attractor and the KS equations will be presented. LSS will be used to calculate usable adjoints to estimate temporal error for the Lorenz attractor, and LSS will be used to calculate usable adjoints to estimate spatial errors for the KS equations.

5.1 Shadowing Lemma

In Chapter IV, it was stated that by the first principle of ergodic theory, long time-averages of an output do not depend on the initial conditions. This concept leads to the first idea that the initial conditions for the perturbed trajectory can indeed be relaxed and chosen such that the perturbed trajectory does not diverge away from the reference trajectory, assuming such a trajectory exists. If a perturbed trajectory that does not diverge away from the reference exists, then theoretically, an accurate adjoint based on the non-diverged perturbed trajectory can be calculated for the system. Hence the duality principles from Chapter III can still be used, since the linearization principles can still be applied for the states and the

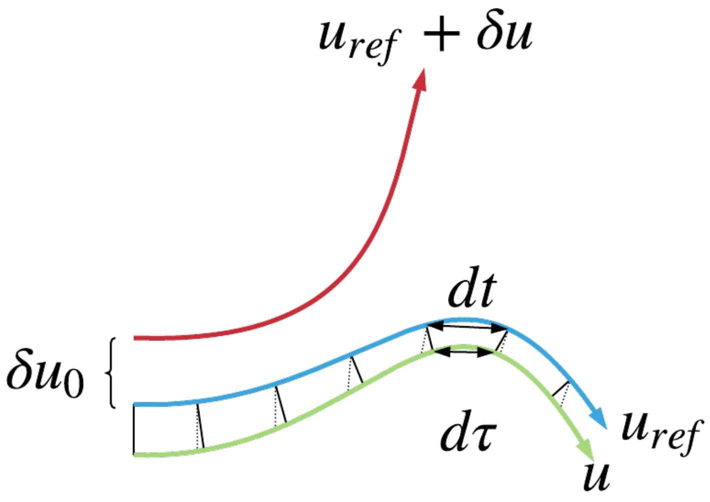


Figure 5.2: Shadow trajectory u with relaxed initial conditions along with the reference and perturbed trajectory. Dotted lines refer to the longer difference in distances (t) compared to the full line, which refer to the shorter difference in distances (τ).

residuals. Regardless of the initial conditions, ergodic theory states that the time-average of the output for the perturbed trajectory and the reference trajectory will converge as $t \to \infty$. This theory leads to the least squares principles which can be used to find the perturbed trajectories or "shadow trajectories" that do not diverge from the reference trajectory. More proof is needed in order to verify that such a trajectory actually exists.

In addition to ergodicity, it was shown in Chapter IV that most chaotic systems can be treated as quasi-hyperbolic. Given ergodicity and hyperbolicity of the system,there exists a lemma to support the existence of the shadow trajectory. If a chaotic system possesses a hyperbolic or quasi hyperbolic strange attractor, and the system is perturbed by initial condition or parameters,$\delta\mu$, then the ***shadowing lemma*** states:

Lemma 1 *For any $\delta u > 0$ there exists $\epsilon > 0$, such that for every u that satisfies $\|\frac{d\boldsymbol{u}}{dt} - f(u; \boldsymbol{\mu} + \delta\boldsymbol{\mu})\| < \epsilon$, $0 \leq t \leq T$, there exists a true solution $\boldsymbol{u}_{ref}$ and a time transformation $\tau(t)$, such that $\|\boldsymbol{u}_{ref}(\tau(t)) - \boldsymbol{u}(t)\| < \delta$, $|1 - \frac{d\tau}{dt}| < \delta$ and $\frac{d\boldsymbol{u}_{ref}}{d\tau} - f(\boldsymbol{u}_{ref}; \boldsymbol{\mu} + \delta\boldsymbol{\mu}) = 0$, $0 \leq \tau \leq \mathcal{T}$.*

where $\boldsymbol{u}$ refers to the shadow trajectory, $\boldsymbol{u}_{ref}$ refers to the reference trajectory, and τ refers to the temporal reference frame of the shadowing trajectory[74]. This time transformation pertains to the neutral Lyapunov exponent and relates the temporal reference frame of the reference trajectory to that of the shadowing trajectory, which can be seen in Figure 5.2. The time dilation term shown in Figure 5.2 relates the reference trajectory's temporal frame to the shadow trajectory's temporal frame. For the unperturbed trajectory, η is defined as

$$\eta = \frac{d\tau(t;\boldsymbol{\mu})}{dt} - 1. \tag{5.1}$$

and is necessary in order to find a shadow trajectory that is as close as possible to the reference trajectory. Overall, the shadowing lemma shows that for a hyperbolic system, a shadow trajectory exists close to the reference trajectory and that it is possible to use ergodic ideas to find this shadow trajectory, $\boldsymbol{u}$. The existence of a shadow trajectory makes it possible to calculate chaotic adjoints for output-based error estimation. The next section goes over the formulation of the LSS primal problem.

5.2 The Least Squares Shadowing Primal Problem

To formulate the LSS method, the governing equations are presented as

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{\mu}), \tag{5.2}$$

where $\boldsymbol{u}$ represents the states of the system and $\boldsymbol{\mu}$ represents the design parameters or inputs of interest. The design parameter, $\boldsymbol{\mu}$, can be the advection speed, α, for the Kuramoto-Sivashinsky problem or the thickness of an airfoil for the Navier-Stokes equations. Again, since the application of LSS is for chaotic systems, the output of interest is a time-average, defined as

$$\overline{J} = \frac{1}{T}\int_{T_0}^{T_f} J(\boldsymbol{u};\boldsymbol{\mu};t)\,dt, \tag{5.3}$$

where J is an instantaneous output, and $T_f - T_0$ is the length of the simulation in time.

A least-squares problem is defined to find a shadow trajectory that exists very close to the original reference trajectory while still satisfying the governing equations. Specifically, the least squares problem looks at minimizing the L_2 norm of the difference between the reference trajectory and the shadow trajectory. This problem can be written out mathematically as

$$\begin{aligned} &\min_{\boldsymbol{u}} \frac{1}{2}\int_{T_0}^{T_1} \|\boldsymbol{u}(\tau(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu}) - \boldsymbol{u}_{ref}(t;\boldsymbol{\mu})\|^2 dt \quad s.t. \\ &\qquad \frac{d\boldsymbol{u}(\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} = \boldsymbol{f}(\boldsymbol{u}(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu}) \quad T_0 < t < T_f \end{aligned} \tag{5.4}$$

where $\boldsymbol{u}_{ref}$ is the reference trajectory, $\boldsymbol{u}$ is the shadow trajectory, and τ is the time of the shadow phase frame explained earlier. Note that

$$\boldsymbol{u}_{ref} = \boldsymbol{u}(\tau(t;\boldsymbol{\mu}),\boldsymbol{\mu}). \tag{5.5}$$

When $T \rightarrow \infty$, the approximated shadow trajectory will converge to the exact shadow trajectory. Hence, solutions of this least-squares shadowing problem are more accurate with longer time simulations.

In sensitivity analysis, one is interested in how the inputs or parameters $\boldsymbol{\mu}$ affect the outputs or states of the system. For LSS, the sensitivity of the states in terms of the parameters is defined as

$$\boldsymbol{v} = \frac{\partial \boldsymbol{u}(\tau(t;\boldsymbol{\mu}),\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}. \tag{5.6}$$

The LSS equations can be rewritten in terms of $\boldsymbol{v}$ by first taking a Taylor-series expansion of $\boldsymbol{u}(\tau(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})$,

$$\begin{aligned}\boldsymbol{u}(\tau(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu}) &= \boldsymbol{u}(\tau(t;\boldsymbol{\mu}),\boldsymbol{\mu}) + \frac{\partial \boldsymbol{u}(\tau(t;\boldsymbol{\mu}),\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\delta\boldsymbol{\mu} + \mathcal{O}(\delta\boldsymbol{\mu}^2),\\ &= \boldsymbol{u}_{ref}(t) + \boldsymbol{v}(t)\delta\boldsymbol{\mu} + \mathcal{O}(\delta\boldsymbol{\mu}^2).\end{aligned} \tag{5.7}$$

Substituting Eqn. 5.7 into the minimization statements, Eqn. 5.4, and assuming small perturbations, one can obtain

$$\begin{aligned}\min_{\boldsymbol{v}} \frac{1}{2}\int_{T_0}^{T_1} &\|\boldsymbol{v}\delta\boldsymbol{\mu}\|^2 dt \quad s.t.\\ \frac{d\boldsymbol{u}(\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} &= f(\boldsymbol{u}(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu}) \quad T_0 < t < T_f\end{aligned} \tag{5.8}$$

The constraint of the least-squares problem can be written in terms of $\boldsymbol{v}$. Given Eqn. 5.7 and ignoring the higher-order terms, one can take the derivative via the chain rule and simplify

$$\begin{aligned}\frac{d\boldsymbol{u}(\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} &= \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt}\\ \frac{d\boldsymbol{u}(\tau;\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{d\tau}\frac{d\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} &= \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt}\\ \boldsymbol{f}(\boldsymbol{u}(\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})\frac{d\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} &= \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt}\end{aligned} \tag{5.9}$$

Taking the Taylor series of $\boldsymbol{f}(\boldsymbol{u}(\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})$ and rewriting Eqn. 5.9,

$$\left[\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu}) + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}\delta\boldsymbol{\mu} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\delta\boldsymbol{\mu}\right]\frac{d\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} = \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt} \tag{5.10}$$

Eqn. 5.10 is a function of $\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu})$, which represents the time dilation of the shadow trajectory when the input $\boldsymbol{\mu}$ is perturbed by $\delta\boldsymbol{\mu}$. Note that for the reference solution, u_{ref}, the the time dilation is $\tau(t;\boldsymbol{\mu}) = t$. In general, the time dilation will be different for the shadow trajectory compared to the reference trajectory, $\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu}) \neq \tau(t,\boldsymbol{\mu})$. To find the perturbed dilation, one can linearize $\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu})$ about $\boldsymbol{\mu}$,

$$\tau(t,\boldsymbol{\mu}+\delta\boldsymbol{\mu}) = \tau(t,\boldsymbol{\mu}) + \frac{\partial\tau(t,\boldsymbol{\mu})}{\partial t}\frac{\partial t}{\partial\boldsymbol{\mu}}\delta\boldsymbol{\mu} + \mathcal{O}(\delta\boldsymbol{\mu}^2). \tag{5.11}$$

Given Eqn. 5.1 and assuming small perturbations, Eqn. 5.11 can be simplified to

$$\eta(t) = \frac{d}{d\boldsymbol{\mu}}\left(\frac{d\tau(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} - 1\right). \tag{5.12}$$

Taking the integral of both sides from $\boldsymbol{\mu} \to \boldsymbol{\mu}+\delta\boldsymbol{\mu}$ and solving for $\frac{d\tau}{dt}$ gives,

$$\frac{d\tau(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu})}{dt} = 1 + \eta(t)\delta\boldsymbol{\mu} \tag{5.13}$$

Substituting Eqn. 5.13 into Equation 5.10, one can find,

$$\left[\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu}) + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}\delta\boldsymbol{\mu} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\delta\boldsymbol{\mu}\right][1+\eta(t)\delta\boldsymbol{\mu}] = \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt} \tag{5.14}$$

Simplifying and neglecting higher-order terms produces,

$$\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu}) + f(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\delta\boldsymbol{\mu} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}\delta\boldsymbol{\mu} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\delta\boldsymbol{\mu} = \frac{d(\boldsymbol{u}_{ref}(t)+\boldsymbol{v}(t)\delta\boldsymbol{\mu})}{dt} \tag{5.15}$$

Since $\frac{d\boldsymbol{u}_{ref}}{dt} = \boldsymbol{f}(\boldsymbol{u}_{ref},\boldsymbol{\mu})$, Eqn. 5.15 and the original minimization problem statement can be written as,

$$
\begin{aligned}
\min_{\boldsymbol{v}} \frac{1}{2}\int_{T_0}^{T_f} \|\boldsymbol{v}\delta\boldsymbol{\mu}\|^2 dt \quad s.t.& \\
\frac{d\boldsymbol{v}(t)}{dt}\delta\boldsymbol{\mu} =& \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}\delta\boldsymbol{\mu} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\delta\boldsymbol{\mu} + \\
&\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\delta\boldsymbol{\mu}, \quad T_0 < t < T_f
\end{aligned}
\tag{5.16}
$$

Dividing Eqn. 5.16 by $\delta\boldsymbol{\mu}$, one can obtain the modified problem statement in terms of $\boldsymbol{v}$:

$$
\begin{aligned}
&\min_{\boldsymbol{v}} \frac{1}{2}\int_{T_0}^{T_1} \|\boldsymbol{v}(t)\|^2 dt \quad s.t. \\
&\frac{d\boldsymbol{v}(t)}{dt} = \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} + \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t).
\end{aligned}
\tag{5.17}
$$

After the linearized form of the minimization problem has been found, the new governing equations that define the original problem need to be found in the optimization process.

5.3 Karush-Kuhn-Tucker Conditions: Modified Governing Equations

Once the linearization of the problem statement has been made, the goal is to find the new set of governing equations that encapsulates the minimization problem and the shadow trajectory definition. Given the new minimization problem written in terms of $\boldsymbol{v}$ in Eqn. 5.17, one can turn the constrained optimization problem into an unconstrained optimization problem by forming the Lagrangian,

$$
\boldsymbol{\mathcal{L}}(\boldsymbol{v},\boldsymbol{w}) = \boldsymbol{F}(\boldsymbol{v}) + \sum_{k=1}^{m} \boldsymbol{w}_k^T \boldsymbol{c}_k(\boldsymbol{v}),
\tag{5.18}
$$

where k refers to number of constraint equations, $\boldsymbol{w}$ refers to the Lagrange multipliers, $\boldsymbol{F}(\boldsymbol{v})$ refers to the function to be minimized, and $\boldsymbol{c}_k$ refers to the residuals of the constraint equations. Applying this to Eqn. 5.17 as a constrained variational problem gives the following Lagrangian,

$$
\begin{aligned}
&\boldsymbol{\mathcal{L}}(\boldsymbol{v},\boldsymbol{w}) = \frac{1}{2}\int_{T_0}^{T_f} \boldsymbol{v}(t)^T\boldsymbol{v}(t) + \\
&2\boldsymbol{w}^T\left(\frac{d\boldsymbol{v}(t)}{dt} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\right) dt.
\end{aligned}
\tag{5.19}
$$

To find the KKT equations (the new governing equations for this problem), one can take the derivatives of this equation with respect to $\boldsymbol{v}$ and $\boldsymbol{w}$ and set them to 0,

$$\frac{\partial \mathcal{L}(\boldsymbol{v},\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{d\boldsymbol{v}(t)}{dt} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t) = \boldsymbol{0}. \tag{5.20}$$

To find $\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}$, the Lagrangian needs to be manipulated more before taking the derivative. Eqn. 5.19 can be be rewritten by integrating by parts $\int_{T_0}^{T_f} 2\boldsymbol{w}^T \frac{d\boldsymbol{v}}{dt} dt$,

$$\mathcal{L}(\boldsymbol{v},\boldsymbol{w}) = \frac{1}{2}\int_{T_0}^{T_f} \boldsymbol{v}(t)^T\boldsymbol{v}(t)\;dt + [\boldsymbol{w}^T\boldsymbol{v}]_{T_0}^{T_f} +$$
$$\int_{T_0}^{T_f} -\frac{d\boldsymbol{w}^T(t)}{dt}\boldsymbol{v} - \boldsymbol{w}^T\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} - \boldsymbol{w}^T\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{w}^T\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\;dt \tag{5.21}$$

Then, taking the derivative with respect to $\boldsymbol{v}$ and setting it to 0, one can obtain (after a transpose),

$$\frac{\partial \mathcal{L}(\boldsymbol{v},\boldsymbol{w})}{\partial \boldsymbol{v}} = -\frac{d\boldsymbol{w}(t)}{dt} - \left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^* \boldsymbol{w}(t) + \boldsymbol{v}(t) = \boldsymbol{0}, \tag{5.22}$$

where $(\cdot)^*$ refers to the conjugate transpose. Thus the KKT conditions are

$$\frac{d\boldsymbol{v}(t)}{dt} = \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} + \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t), \tag{5.23}$$

$$\frac{d\boldsymbol{w}(t)}{dt} = -\left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^* \boldsymbol{w}(t) + \boldsymbol{v}(t). \tag{5.24}$$

Here, the Karush-Kuhn-Tucker (KKT) conditions take the form of the dual/adjoint problems. Note that there is a third KKT condition equation for the time transformation; however, this term will be included in the final LSS adjoint equations in a different way. for the time transformation term b In the next section, the adjoint equations will be found corresponding to these KKT conditions.

5.4 Least Squares Shadowing Adjoint Equations

To find the Least Squares Shadowing adjoint equations, the concepts of duality from chapter III for the continuous adjoint derivations can be used. First, the chaotic output of interest is considered as,

$$\bar{J} = \frac{1}{T}\int_0^{T_f} J(\boldsymbol{u}(t;\boldsymbol{\mu}+\delta\boldsymbol{\mu});\boldsymbol{\mu}+\delta\boldsymbol{\mu})\;dt. \tag{5.25}$$

Since the output is nonlinear for chaotic systems, it needs to be linearized about the reference primal trajectory before the dual equation can be found. The linearized output is defined as,

$$\bar{J}'[\boldsymbol{u}] = \frac{1}{T}\int_{T_0}^{T_1} \frac{\partial J}{\partial \boldsymbol{u}}\boldsymbol{v}\; dt, \tag{5.26}$$

where, the prime values are the Féchét derivatives. Note that $\bar{J}'[\boldsymbol{u}]$ indicates the state about which the lienarization is performed. Next, the same procedure for the unsteady continuous adjoint derivations can be used to find the chaotic LSS adjoint equations. First, the adjoint Lagrangian is defined as,

$$\mathcal{L}_{adj} \equiv \bar{J}'[\boldsymbol{u}] - \int_{T_0}^{T_f} \sum_{k=1}^{k=m} \boldsymbol{\psi}_k^T, \boldsymbol{r}_k(\boldsymbol{v}, \boldsymbol{w})\; dt. \tag{5.27}$$

where $\boldsymbol{\psi}_k$ is the adjoint and $\boldsymbol{r}_k(\boldsymbol{v}, \boldsymbol{w})$ is the residual of the governing equations (in our case the KKT conditions found previously). Putting all terms in Eqns. 5.23 and 5.24 on one side, the constraints for the adjoint Lagrangian are

$$\boldsymbol{r}_{k=1} = \frac{d\boldsymbol{v}(t)}{dt} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}(t) - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t), \tag{5.28}$$

$$\boldsymbol{r}_{k=2} = \frac{d\boldsymbol{w}(t)}{dt} + \left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^T \boldsymbol{w}(t) - \boldsymbol{v}(t). \tag{5.29}$$

Substituting Eqn. 5.28 and 5.29 into Equation 5.27 gives,

$$\begin{aligned}\mathcal{L}_{adj} = {} & \frac{1}{T}\int_{T_0}^{T_f} \frac{\partial J}{\partial \boldsymbol{u}}\boldsymbol{v}dt \\ & - \int_{T_0}^{T_f} \boldsymbol{\psi}_1^T \left[\frac{d\boldsymbol{v}(t)}{dt} - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}(t)\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\right] dt \\ & - \int_{T_0}^{T_f} \boldsymbol{\psi}_2^T \left[\frac{d\boldsymbol{w}(t)}{dt} + \left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^T \boldsymbol{w}(t) - \boldsymbol{v}(t)\right] dt.\end{aligned} \tag{5.30}$$

Integrating by parts gives,

$$\begin{aligned}\mathcal{L}_{adj} = \frac{1}{T}\int_{T_0}^{T_f} \frac{\partial J}{\partial \boldsymbol{u}}\boldsymbol{v}dt - [\boldsymbol{\psi}_1^T\boldsymbol{v}]_{T_0}^{T_f} - [\boldsymbol{\psi}_2^T\boldsymbol{w}]_{T_0}^{T_1} + \int_{T_0}^{T_f} \frac{d\boldsymbol{\psi}_1^T}{dt}\boldsymbol{v}dt \\ - \int_{T_0}^{T_f} \boldsymbol{\psi}_1^T\left[-\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v}(t) - \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} - \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t)\right]dt \\ + \int_{T_0}^{T_f} \frac{d\boldsymbol{\psi}_2^T}{dt}\boldsymbol{w}dt - \int_{T_0}^{T_f} \boldsymbol{\psi}_2^T\left[\left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^T\boldsymbol{w}(t) - \boldsymbol{v}(t)\right]dt\end{aligned} \tag{5.31}$$

Now to find the adjoint equation that governs $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_1$, one can take variations of $\mathcal{L}_{adj}$ with respect to $\boldsymbol{v}$ and $\boldsymbol{w}$,

$$\frac{\partial \mathcal{L}_{adj}}{\partial \boldsymbol{w}} = -[\boldsymbol{\psi}_2^T\delta\boldsymbol{w}]_{T_0}^{T_f} + \int_{T_0}^{T_f}\left[\frac{d\boldsymbol{\psi}_2^T}{dt} - \boldsymbol{\psi}_2^T\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T\right]\delta\boldsymbol{w}\,dt \tag{5.32}$$

$$\frac{\partial \mathcal{L}_{adj}}{\partial \boldsymbol{v}} = -[\boldsymbol{\psi}_1^T\delta\boldsymbol{v}]_{T_0}^{T_f} + \int_{T_0}^{T_f} \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}} + \frac{d\boldsymbol{\psi}_1^T}{dt} + \boldsymbol{\psi}_1^T\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right) + \boldsymbol{\psi}_2^T\,dt. \tag{5.33}$$

Setting these derivatives to 0 gives,

$$\frac{d\boldsymbol{\psi}_2^T}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T\boldsymbol{\psi}_2^T, \tag{5.34}$$

$$\frac{d\boldsymbol{\psi}_1^T}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)\boldsymbol{\psi}_1^T - \boldsymbol{\psi}_2^T - \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}}. \tag{5.35}$$

Taking the transpose of both equations,

$$\frac{d\boldsymbol{\psi}_2}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)\boldsymbol{\psi}_2, \tag{5.36}$$

$$\frac{d\boldsymbol{\psi}_1}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T\boldsymbol{\psi}_1 - \boldsymbol{\psi}_2 - \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}}^T. \tag{5.37}$$

In summary, the adjoint system is given by Eqn. 5.40 and Eqn. 5.41. The temporal boundary conditions, which derive from the boundary terms in the integration by parts, dictate that the equation for $\boldsymbol{\psi}_2$ is solved forward in time, whereas the equation for $\boldsymbol{\psi}_1$ is solved backward in time. These chaotic unsteady adjoint equations look similar to the traditional continuous unsteady adjoint equations; however, the chaotic adjoint depends on two, not one, equations, already making it more expensive of a calculation than the traditional equation. Due to the coupled chaotic adjoint equations, a more complicated solver is used.

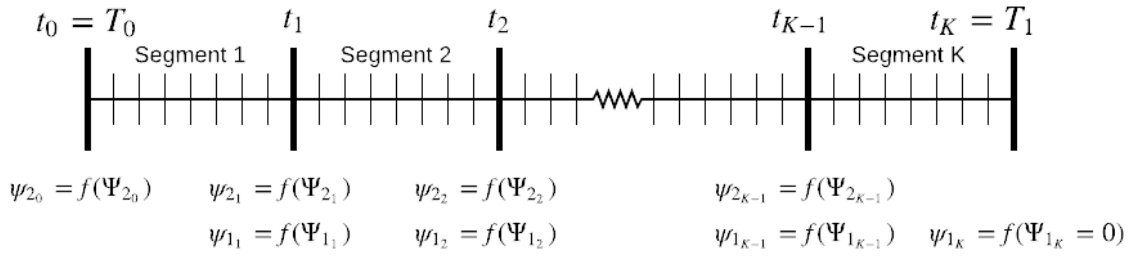
5.5 Solving the Least Squares Adjoint Equations

In the previous section, the LSS adjoint equations were derived for a quasi-hyperbolic ergodic equation. To solve for these equations, an additional tangent solver is needed before the reverse adjoint solve can be made, complicating the process. There are two different ways to solve the adjoint equations

1. **The Full trajectory design** discretizes the LSS adjoint equations directly and solves the linear set of systems, $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{x}$ includes the the tangent, Lagrange, and time transformation variable, η. This technique consists of finding a full trajectory at once where the tangent $\boldsymbol{\psi}_2$ is the main constraint of the system.

2. **The Checkpoint design** divides the time simulation into n_{seg} windows. Instead of relaxing the initial conditions and then searching for a set of new initial conditions for the full trajectory, the tangent and Lagrange adjoint equations are solved for different checkpoints throughout the simulation time as shown in Figure 5.3. By minimizing over n_{seg} time windows, the problem has fewer variables than the full trajectory design, since the η is taken into account via projections of the tangent and Lagrange solutions, making it a cheaper method to use to solve for the LSS adjoint equations [70]. This method solves a smaller adjoint system compared to the full trajectory design.

For this dissertation, the cheaper checkpoint design is used to solve the LSS adjoint equations. The main variable of this method is the number of time segments, n_{seg} or the length of each time segment, T_k. The time simulation is split up such that the adjoint does not grow exponentially within each segment and such that information of the inherent problem is not lost. By using an iterative checkpoint process [70] for each time window, it is



$t_0 = T_0$
t_1
t_2
t_{K-1}
$t_K = T_1$
Segment 1
Segment 2
Segment K
$\psi_{2_0} = f(\Psi_{2_0})$
$\psi_{2_1} = f(\Psi_{2_1})$
$\psi_{2_2} = f(\Psi_{2_2})$
$\psi_{2_{K-1}} = f(\Psi_{2_{K-1}})$
$\psi_{1_1} = f(\Psi_{1_1})$
$\psi_{1_2} = f(\Psi_{1_2})$
$\psi_{1_{K-1}} = f(\Psi_{1_{K-1}})$
$\psi_{1_K} = f(\Psi_{1_K} = 0)$

Figure 5.3: Least Squares Shadowing Checkpoint design for adjoint calculation.

not required to solve the tangent and the adjoint for the entire time simulation, decreasing the overall computational time. The checkpoint design reduces the size of the KKT system by searching for the shadow trajectory at $K + 1$ checkpoints, where the total number of time segments is $n_{seg} = K$. This technique can be thought of as an iterative solve of the following system of linear equations [70],

$$\boldsymbol{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\Psi}_{2,0} \\ \boldsymbol{\Psi}_{2,1} \\ \vdots \\ \boldsymbol{\Psi}_{2,K-1} \\ \boldsymbol{\Psi}_{1,1} \\ \vdots \\ \boldsymbol{\Psi}_{1,K-1} \end{bmatrix} \tag{5.38}$$

where $\boldsymbol{x}$ starts off at the beginning as the initial guesses for the tangent and adjoint values of the shadow trajectory. Note that the number of $\boldsymbol{\Psi}_2$ terms in $\boldsymbol{x}$ is one fewer than that of $\boldsymbol{\Psi}_1$. $\boldsymbol{\Psi_2}$ and $\boldsymbol{\Psi_1}$ represent the adjoints for the time checkpoint nodes. Thus, the length of $\boldsymbol{x}$ is $m = N(2n_{seg} - 1)$ where N is the total number of states in the system. Hence $\mathcal{A} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$. In order to take into account the time-dilation term, η, LSS solves for the projection of the adjoint inputs onto $\boldsymbol{f}(t)$ via

$$\boldsymbol{\psi} = P_{t_i}\boldsymbol{\Psi} \equiv \boldsymbol{\Psi} - \frac{\boldsymbol{\Psi}^T \boldsymbol{f}(t_i)}{\boldsymbol{f}(t_i)^T \boldsymbol{f}(t_i)} \boldsymbol{f}(t_i), \tag{5.39}$$

where t_i refers to the checkpoint at t_i. In addition, taking into account the time dilation term makes the LSS adjoint method, adjoint consistent [70]. Thus, the LSS adjoint equations take its final form as

$$\frac{d\boldsymbol{\psi}_2}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right) \boldsymbol{\psi}_2, \tag{5.40}$$

$$\frac{d\boldsymbol{\psi}_1}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\psi}_1 - P_t\boldsymbol{\psi}_2 - \beta\frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}}^T. \tag{5.41}$$

where β is used in an iterative solver. The initial conditions for $\boldsymbol{\psi}_2$ on each time segment is

$$\boldsymbol{\psi}_2(t_i) = P_{t_i}\boldsymbol{\Psi}_2 \tag{5.42}$$

and the terminal conditions for $\boldsymbol{\psi}_1$ for each time segment is

$$\boldsymbol{\psi}_1(t_{i+1}) = P_{t_i}\boldsymbol{\Psi}_1 - \beta\frac{J(t_{i+1}) - \overline{J}}{T\boldsymbol{f}_{i+1}^T \boldsymbol{f}_{i+1}} \boldsymbol{f}_{i+1} \tag{5.43}$$

Before solving Eqn. 5.38, a burn time, T_{burn}, is executed to ensure that the trajectory is on the attractor at the start of actual simulation. Note that perturbed initial conditions are assigned before the burn time takes place. Once the burn time has been done, the primal solution for the entire simulation that spans all the time segments is needed. Once this is found, the adjoint LSS checkpoint solver presented in Algorithm 5.1 is used. A guess for the tangent solution and adjoint solution for each checkpoint is made and then used to find the corresponding tangent and adjoint solutions. This process is performed for all time segments and repeated until the tangent and adjoint solution converges via an iterative matrix-free solver, GMRES, to a prescribed tolerance. Within the GMRES algorithm, matrix vector multiplication and system solves are needed. For these calculations, the MATVEC algorithm is presented in Algorithm 5.2 used specifically for the LSS checkpoint solver.

Algorithm 5.1 Adjoint LSS Checkpoint Iterative Solver for $\mathcal{A}\,\boldsymbol{x} = \boldsymbol{b}$

```
Input: $\boldsymbol{\Psi}_1$, $\boldsymbol{\Psi}_1$
Output: $\boldsymbol{\psi}_1$
1: Choose an iterative solver e.g. GMRES
2: To calculate $\mathcal{A}\overline{\boldsymbol{x}}$ in interative solver, set $\beta = 0$, and perform $\mathrm{MATVEC}(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1, \beta)$
3: To calculate $\mathcal{A}\overline{\boldsymbol{x}} - \boldsymbol{b}$ in iterative solver, set $\beta = 1$, and perform $\mathrm{MATVEC}(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1, \beta)$
4: To compute $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$, find $\boldsymbol{\psi}_2$ by setting $\beta = 1$, and performing $\mathrm{MATVEC}(\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1, \beta)$
```

Algorithm 5.2 Adjoint MATVEC Algorithm

```
Input: $\boldsymbol{\Psi}_2, \boldsymbol{\Psi}_1, \beta$
Output: $\boldsymbol{R}_i^{\boldsymbol{\psi}_2}, \boldsymbol{R}_i^{\boldsymbol{\psi}_1}$
1: for $t = 1, K$ time segments do
2:     Set tangent initial conditions to $\boldsymbol{\psi}_2(t_i) = P_{t_i}\boldsymbol{\Psi}_2$
3:     Set adjoint terminal conditions to $\boldsymbol{\psi}_1(t_i) = P_{t_i}\boldsymbol{\Psi}_1 - \beta\frac{J(t_{i+1}) - \overline{J}}{T\boldsymbol{f}_{i+1}^T\boldsymbol{f}_{i+1}}\boldsymbol{f}_{i+1}$
4:     Time integrate $\frac{d\boldsymbol{\psi}_2}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)\boldsymbol{\psi}_2$
5:     Time integrate backwards $\frac{d\boldsymbol{\psi}_1}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T\boldsymbol{\psi}_1 - \boldsymbol{\psi}_2 - \frac{1}{T}\frac{\partial J}{\partial u}^T$
6:     Compute $\boldsymbol{R}_i^{\boldsymbol{\psi}_2} = \boldsymbol{\Psi}_1 - P_{t_i}\boldsymbol{\psi}_1(t_i^-)$
7:     Compute $\boldsymbol{R}_i^{\boldsymbol{\psi}_1} = \boldsymbol{\Psi}_2 - P_{t_i}\boldsymbol{\psi}_2(t_i^+)$
8: end for
9: end for
```

For a large number of time segments, the stiffness of Eqn. 5.38 grows and demands efficient preconditioning strategies.

5.6 Output-Based Error Estimation via the Adjoint-Weighted Residual

The least-squares shadowing (LSS) approach has been used successfully to compute efficient sensitivities for chaotic systems [71, 72, 70]. Once $\boldsymbol{\psi}_1$ has been found for each time segment, the LSS method can be extended to output-based error estimation.

After the adjoint system solve, output sensitivities can be calculated from $\boldsymbol{\psi}_1$, since this is the adjoint that weights the residual term containing $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}}$.

Specifically,

$$\frac{d\bar{J}}{d\boldsymbol{\mu}} = \int_0^T \boldsymbol{\psi}_1^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}} dt. \tag{5.44}$$

To estimate the output error using the LSS adjoint, Eqn. 5.44 is used with a residual perturbation computed from two different discretization spaces. Doing so gives a perturbation in the residual arising from the discretization error, which is weighted by the fine-space adjoint to obtain the error estimate,

$$\delta \bar{J} = -\int_{T_0}^{T_f} \boldsymbol{\psi}_{1,h}^T \left[\frac{d\boldsymbol{u}_H}{dt} - \boldsymbol{f}(\boldsymbol{u}_H) \right]_h dt = -\int_{T_0}^{T_f} \boldsymbol{\psi}_{1,h}^T \boldsymbol{M}^{-1} \boldsymbol{R}_h(\boldsymbol{u}_h^H), \tag{5.45}$$

where $\boldsymbol{R}_h(\boldsymbol{u}_h^H)$ is the fine-space residual vector evaluated with the coarse solution injected into the fine space.

5.7 LSS Adjoint-Weighted Residual for Lorenz Attractor

To test the LSS method, the Lorenz attractor was chosen for to its simplicity. The primal solution was first found by implementing a burn time of $T_{\text{burn}} = 0.5T$. DG in time was used for the temporal discretization with a time step of $\Delta t = 0.05$. The goal of this is exercise is to determine the effect of discretization errors on the time-average output for the Lorenz attractor between temporal approximation orders of $r_H = 1$ and $r_h = 2$. Before the burn time, the initial conditions were chosen to be $x = 1,\ y = 1,\ z = 1$ and then perturbed by a small $\delta \boldsymbol{u}$, the components of which were randomly chosen from $[-0.1, 0.1]$. The unsteady output of interest was the time-average of the z trajectory. Figure 5.4 shows the comparisons of the output-based error estimates to the actual difference of the time-average outputs with the same initial conditions. Note that only the temporal errors are being estimated in this problem. The length of the time segments for all time simulations was set to $T_k = 4$, which was found to be close to the maximum possible values for keeping the adjoint calculations and the linearized calculations stable. If the time window were changed to be larger, the

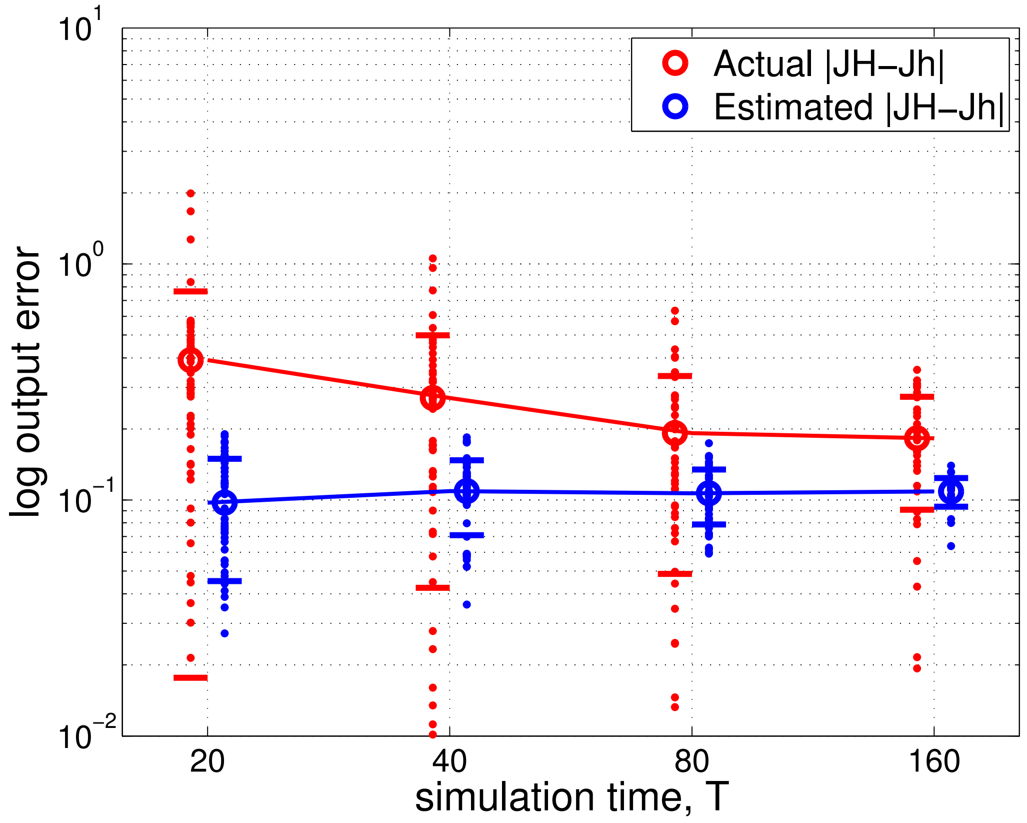


Figure 5.4: Lorenz attractor: temporal output based error estimation using the LSS-adjoint weighted residual method. The points are the output based errors computed from 50 individuals runs in each ensemble. The circles are the means of the ensemble errors, and the horizontal line segments are drawn at $\pm$ one standard deviation

adjoint would begin to increase exponentially backwards in time. Hence, the time window is an important parameter to find reasonable chaotic adjoints. For the Lorenz Attractor, the estimated discretization numerical error in blue under-predicts the actual error in red especially at earlier time simulations, which is expected since the statistical errors tend to dominate. At longer time simulations, the numerical error estimates do improve and the statistics improve as well. It is important to note that the output-based error estimates are more tightly clustered than the actual errors, which indicates a high level of confidence in the output based error results, even for shorter simulations times.

The results of the output-based error estimates using LSS show promise. Implementing this method on a more complicated chaotic system is important to see how the method behaves for different Lyapunov exponents values and with estimation of spatial errors.

5.8 LSS Adjoint Weighted Residual for Kuramoto-Sivashinsky

In addition to the Lorenz Attractor, the LSS method is used to calculate chaotic adjoints for the Kuramoto-Sivashinsky equation. Unlike the Lorenz Attractor, the KS equations is a one-dimensional PDE, and thus the error estimates of interest can include the spatial discretization error. For both the coarse and fine space residual, BFD2 is used with a time step of $\Delta t = 0.2$ to solve the PDE in time, and the spatial interpolation order is set to $p_H = 2$ and $p_h = 3$. Note that $p = 1$ does not give a chaotic system as its trajectories are under resolved and reach a fixed point in time; the states reach a steady state solution. The burn time for KS is set to $T_{\text{burn}} = 500$. Again, like the Lorenz attractor case, the initial conditions are perturbed by a small random value of $\delta u \in [-0.2, 0.2]$ at all the spatial nodes before the burn time is implemented. Figure 5.5 shows the error estimate results for KS at $T = 20, 40, 60, 80, 100, 120, 140, 160, 180$ for time segments of length $T_k = 5$, and Figure 5.6 shows the error estimate results for KS at $T = 20, 40, 60, 80, 100, 120, 140, 160, 180$ for time segments of length $T_k = 4$. In addition, the error estimates for both results are shown for three different adjoint calculations, $\boldsymbol{\psi}, \delta\boldsymbol{\psi}, \delta\boldsymbol{\psi}_{H_1}$. Note that Figure 5.5 and Figure 5.6 show similar results for $T_k = 4$ and $T_k = 5$; decreasing the length of the time window beyond $Tk = 5$ has little effect on the output-based error estimates.

Originally, the error estimates using $\boldsymbol{\psi}$ were investigated for both $T_k = 5$ and $T_k = 4$, and these are shown in red. However, the error estimates behave differently than that of the Lorenz attractor. First, the error estimates over predict the actual errors, shown in in blue, and begin to converge to the actual error estimate by $T = 180$ at a much slower rate compared to the results of the Lorenz attractor. The average difference between the error estimate and the actual error estimate is just above an order of magnitude larger. Another difference between the results for KS versus those of the Lorenz attractor is that the statistics do not improve with time. Instead, the statistics stay relatively the same with time, which is likely due to the fact that the KS equation does not become chaotic as fast as the Lorenz Attractor. To see if the error estimates can be improved, the error estimates in terms of $\delta\psi$ were calculated in turquoise. Unfortunately, the error estimates with $\delta\boldsymbol{\psi}$ for $T_k = 5$ and $T_k = 4$ becomes worse, possibly due to poor projections of the fine solution onto the coarse space. When investigating the behavior of the projection of the fine space solution to the coarse mesh used to calculate, $\delta\boldsymbol{\psi}$, it was found that the projection was of poor quality. This poor quality in the projection process is most likely due to the very chaotic nature/lack of smoothness of the adjoint itself, making it difficult to properly negate the influence of the coarse space adjoint.

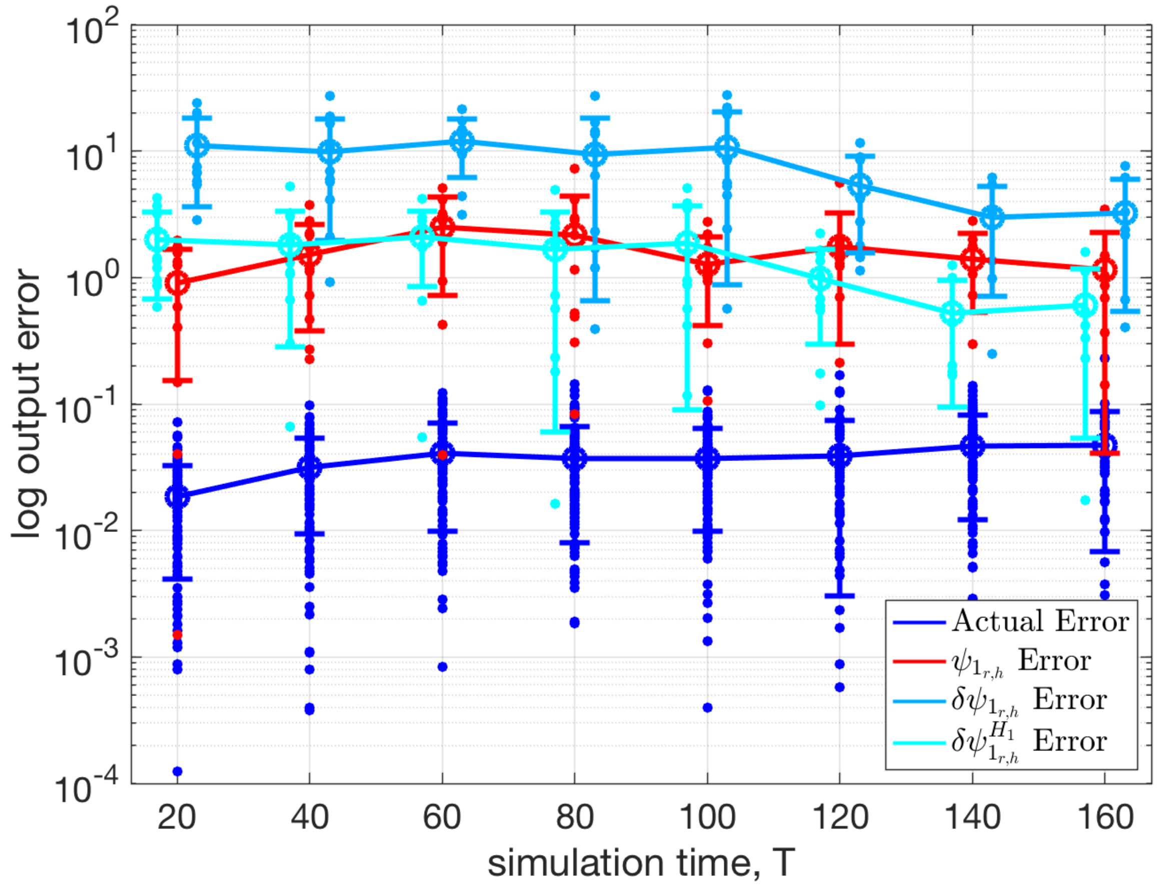


Figure 5.5: KS: spatial output based error estimation that uses the LSS-adjoint weighted residual method for time segment length of $T_k = 5$. The points are the output based errors computed from 10 individuals runs in each ensemble. The circles are the means of the ensemble errors, and the horizontal line segments are drawn at $\pm$ one standard deviation

In an attempt to improve the projection process and to therefore improve the error estimates, the H_1 projection was used, which restricts the slopes of the coarse and fine solution to match at the coarse basis nodes as well. H_1 projection is described in Chapter III. The results of the error estimates with the H_1 projection for $T_k = 5$ and $T_k = 4$ can be seen in both Figure 5.5 and Figure 5.5 in cyan, which on average matches the original error estimate in red. Using the H_1 projection does improve the error estimate by a small amount compared to the original projection, but does not improve the overall error estimate significantly. These results suggest that the projection of a chaotic adjoint for KS does not eliminate the undesirable error contributions. Even though the results for LSS are not as accurate as that of the Lorenz attractor, the error estimates are reasonable. This is most likely due to the fact that the KS equation compared to the Lorenz attractor has more larger positive Lyapunov exponents, making it more difficult to predict the discretization errors of KS.

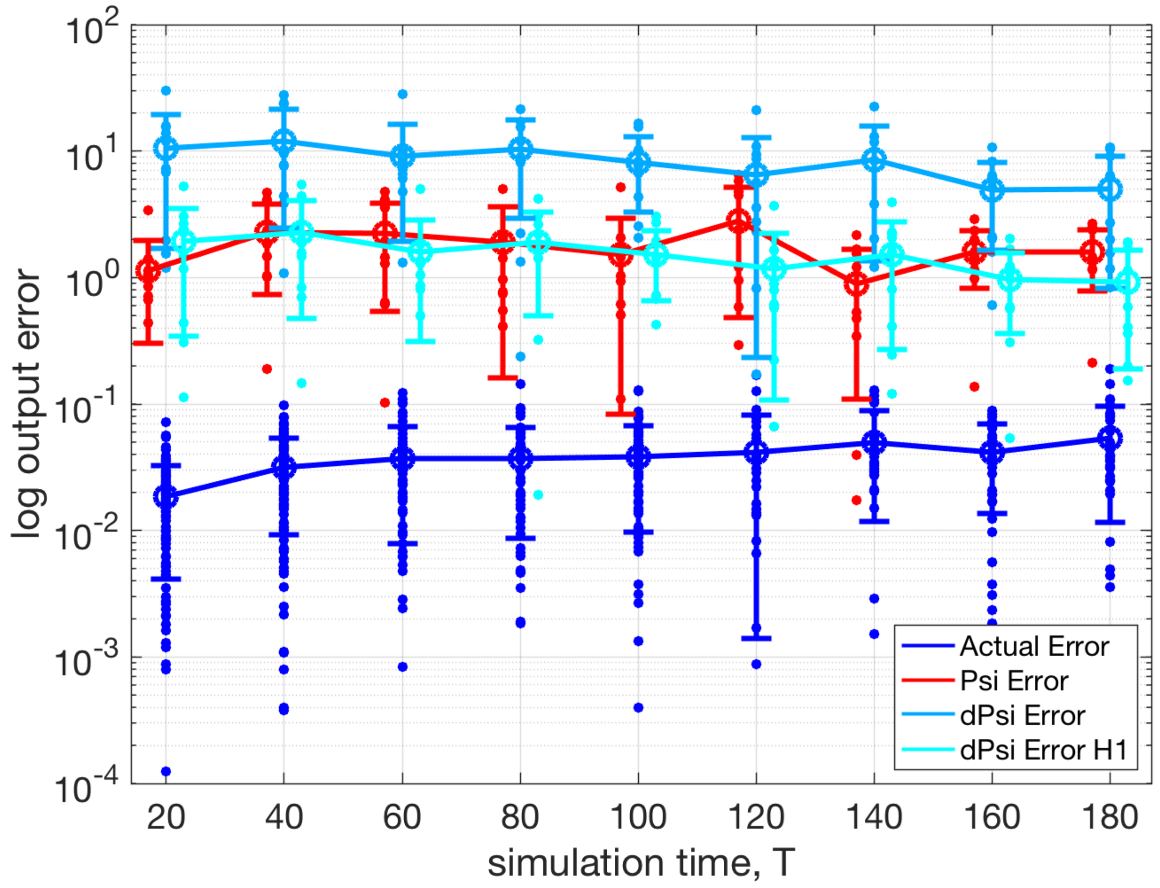


Figure 5.6: KS: spatial output based error estimation that uses the LSS-adjoint weighted residual method for time segment length of $T_k = 4$. The points are the output based errors computed from 10 individuals runs in each ensemble. The circles are the means of the ensemble errors, and the horizontal line segments are drawn at $\pm$ one standard deviation

One observation about the implementation of the LSS method for the KS equations is that due to the fact that the KS equations are a function of both one dimensional space and time, the size of the system is much larger than that of the Lorenz Attractor. The fine space adjoint is a function of $N = 480$ spatial nodes and $T/\Delta t$ temporal nodes, which in contrast to the Lorenz attractor is much larger and requires many more calculations to find the adjoint. It was found that the LSS method for systems larger than the Lorenz attractor is quite expensive. For $T = 20$ with $T_k = 4$, the LSS system has $n_{seg} = 5$ time windows. When solving the linear system of LSS, the $\mathcal{A}$ matrix has size 4320×4320. The error tolerance in GMRES was set to 10^{-6} preventing the need for all 4320 iterations, but this small modification is not enough to produce efficient chaotic adjoints for much longer time simulations. In order to implement LSS for larger systems, the LSS method needs to be modified in order to make it more practical for more extreme simulations. In the next chapter, reduced-order modeling will be introduced to reduce the size of the overall system, by reducing N spatial nodes to only n_r spatial nodes where $n_r \ll N$.

5.9 Summary

Chapter III introduced the traditional adjoint methods and output-based error estimation for use with unsteady simulations, which when applied to chaotic systems, fail to produce accurate error estimates for the purpose of mesh adaptation. Chapter IV revealed that traditional adjoint calculations for unsteady flows fail due to the "butterfly effect", characteristic of chaotic systems. The "butterfly effect" describes the chaotic system's high sensitivity to its initial conditions. Due to this behavior, the linearizations used to derive the traditional adjoint equations fail, leading to exponentially large adjoint magnitudes when calculated backwards in time. In this chapter, the LSS method was introduced, which provides additional steps to the duality process. When perturbing the initial conditions of a chaotic trajectory, the new trajectory will diverge away from the reference trajectory over a small period of time. The rate at which the perturbed trajectory diverges is dependent on the Lyapunov exponent of the system. However, due to the ergodic assumptions that state that the initial conditions will have zero effect on the time-average output as $T \to \infty$, the initial conditions can be relaxed. Instead of restricting the initial conditions, the perturbed trajectory will be restricted such that the trajectory exists very close to the original trajectory. This perturbed trajectory will have completely different initial conditions that are not of concern, which is again allowed under ergodic theory assumption of chaos. The LSS method is a robust method that finds this new perturbed trajectory, the "shadow trajectory". It does so by solving a minimization problem given the constraint of the system, the original governing equations. In other words, the new shadow trajectory must still satisfy the original nature of the problem stated by the governing equation. The KKT conditions were found from the minimization problem and then the ideas of duality from section 3.1 were used to find the modified adjoint equations. Instead of one adjoint equation, the LSS method requires that one solves two equations, a tangent and an adjoint equation. The LSS method is simplified by solving these two equations over short time windows via a checkpoint design method. Given an initial guess for the shadow trajectory, the MATVEC algorithm is used to find the new shadow trajectory. Note that the LSS problem can be seen as solving a set of linear equations, $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ where the size of $\boldsymbol{x}$ is $N(2n_{seg} - 1)$.

Results of LSS for the Lorenz attractor and the Kuramoto-Sivashinsky equations were presented to show how the LSS method works for output-based error estimation. The results for the Lorenz attractor showed that LSS produces accurate error estimates over long periods of time. As the simulation times increased, the error estimates improved; this is due to the ergodic assumption that initial conditions have little to no effect as $T \rightarrow \infty$. Hence, as $T \rightarrow \infty$, the error estimates should converge as it does for the Lorenz attractor. The statistics do improve as well with time, which shows that the confidence in the error estimates increase with time as well.

Lastly, the results for Kuramoto-Sivashinsky equation were shown. The KS equations have more larger positive Lyapunov exponents, making them less suitable for error estimation. The results showed that implementing LSS for KS did not produce as accurate results compared to those of the Lorenz attractor. On average, the error estimates for KS, were over a magnitude larger throughout time than the actual error. The statistics did not improve and remained on average constant with time. Multiple error estimates using $\delta\psi$ and $\delta\psi_{H_1}$ were attempted to see if the projection of the adjoint would improve the error estimates. However, the projection was not found to improve the error estimates. The H_1 projection process did not change the error estimate on average, either signaling the poor quality of the H_1 projection and/or that the adjoint is not strongly affected by its coarse counterpart.

LSS is a robust and accurate method that works well for the Lorenz Attractor; however, application of LSS for KS and for larger problems has proven to be computational expensive. The next chapter will go over several reduction techniques that will decrease the size of the primal solution and hence the size of the LSS adjoint problem, decreasing the number of iterations it takes for LSS to find the shadow trajectory and to predict error estimates.

CHAPTER VI

Model Reduction for Chaotic Flows

In Chapter V, the Least Squares Shadowing Adjoint-Weighted Residual (LSS) method was introduced and implemented to calculate adjoints that do not increase in magnitude exponentially backwards in time for the purpose of output-based error estimation. Implementation of LSS with output-based error estimation successfully estimated temporal discretization errors for the Lorenz attractor, an ordinary differential equation when using DG in time. The method implemented for the Kuramoto-Sivashinsky (KS) equation as well, a more complicated one-dimensional PDE where the goal was to estimate the spatial discretization error only. The results for KS showed that the error estimates over predicted the actual errors by an order of magnitude or more depending on the number of time windows chosen. The error estimates from LSS for KS became worse as the number of time windows decreased, i.e. as the length of each window increased. For both the Lorenz attractor and the KS equation, the error estimates did improve with total simulation time. Furthermore, the computational requirements for the the LSS Adjoint-Weighted Residual method are extreme, making it inefficient in its present state for very large problems. Hence, this chapter introduces model reduction techniques, which are used to reduce the computational costs of LSS, specifically for output-based error estimation.

First, this chapter will introduce projection-based model reduction techniques. Next, model reduction for linear problems and the Least-Squares Petrov-Galerkin (LSPG) method for nonlinear problems will be introduced. Once the reduced-order model (ROM) is found from the LSPG method for nonlinear chaotic systems, hyper-reduced techniques in the Gauss Newton with Approximated Tensors (GNAT) family of methods are used to further reduce the costs associated with the calculation of the residuals and Jacobians of the primal system. Applying GNAT(GNAT) produces the hyper-reduced-order model (HROM). Algorithms for all these techniques will be shown as well as results of GNAT for chaotic systems, KS and Navier-Stokes (NS). Only reduction in space will be investigated.

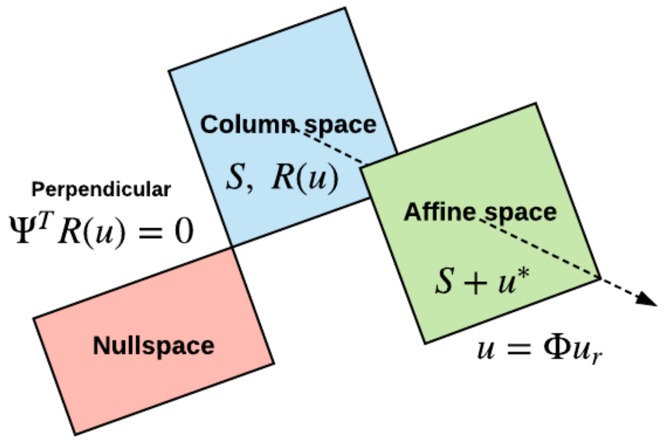


Figure 6.1: Relationship among the subspaces of the full-order model in three-dimensions. The affine space is translated upward from the column space plane by $u*$.

6.1 Projection-Based Model Reduction

Projection-based model reduction techniques have already been proven to work well for linear and nonlinear problems where the state solution can be approximated accurately as a member of an affine subspace whose dimension is significantly smaller than that of the full-order primal solution [75]. The governing equations of the full-order model are represented by a set of linear or nonlinear ODEs,

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{0}. \tag{6.1}$$

For a linear system, Eqn. 6.1 can be written as

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{u} \tag{6.2}$$

When reducing this system, the information associated with the space of the system needs to be captured as best as possible. This process can be done by considering the subspaces of the system seen in Figure 6.1. First, the information of the full-order homogeneous system is captured by the column space, $C(A)$, which is the span of its column vectors, $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{0}$. Note that the column space contains the origin of the system, ($\boldsymbol{b} = \boldsymbol{0}$). To solve the inhomogeneous system in Eqn. 6.2, $\boldsymbol{b}$ must be in the column space of $\boldsymbol{A}$. This is possible by looking to another subspace of the full-order system, the affine subspace which similar to its column space except that the affine subspace does not contain the origin and is instead translated by u^* seen in Figure 6.1. u^* is defined by $\boldsymbol{b}$. The affine subspace is important, because it contains the set of state solutions to $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$ or $\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{0}$ for a nonlinear problem. The projection to the affine subspace allows one to relate the solutions of the full-order model and the affine subspace via a set of basis functions of the affine subspace, $\boldsymbol{\Phi}$, that has dimensions lower than that of the full space. Note that the basis functions for the column space is different than that of the affine subspace, which is dependent on the vector $\boldsymbol{b}$. Thus, the approximated state solution is represented via the projection process as

$$\boldsymbol{u} \approx \boldsymbol{\Phi}\boldsymbol{u}_r, \tag{6.3}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times n_r}$ is the reduced-order basis matrix whose columns store the discrete reduced-order basis functions that describe the affine subspace. N refers to the number of states in the original system and n_r refers to the number of states in the reduced-order model. The reduced-order basis, $\boldsymbol{\Phi}$, contains n_r linearly independent columns, and the

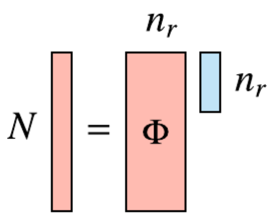


Figure 6.2: Relationship between the full-order solution and the reduced solution via the reduced-order basis functions of the affine subspace. N is the number of states in the full-order system and n_r is the number of states in the reduced-order system.

reduced solution contains n_r reduced states as well ($\boldsymbol{u}_r \in \mathbb{R}^{n_r}$). The relationship between the full-order states and the reduced states can be seen with its corresponding sizes in Figure 6.2.

Substituting Eqn. 6.3 into Eqn. 6.1 produces an overdetermined system. The dimension of $\boldsymbol{\Phi}$ is smaller than the rank of the full-order system. To work with only n_r unknowns and n_r equations, one can look to the left nullspace of the system, which is perpendicular to the column space seen in Figure 6.1. Since the column space is perpendicular, or orthogonal to the left nullspace of the system, the orthogonality relationship can be enforced as

$$\boldsymbol{\Gamma} = \boldsymbol{\varphi}^T \boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{u}_r) = \boldsymbol{0}, \tag{6.4}$$

where $\boldsymbol{\varphi} \in \mathbb{R}^{N \times n_r}$ is the reduced-order basis of the left nullspace of the system, which has dimensions of n_r as well. The left nullspace reduced-order basis is chosen here to minimize the L_2 norm of the residual. In other words, Eqn. 6.4 can be written as a minimization statement,

$$\boldsymbol{\Gamma} = \min_{u_r} \|\boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{u}_r)\|_2. \tag{6.5}$$

To distinguish between the two reduced-order bases, $\boldsymbol{\varphi}$ is referred to as the left nullspace basis function matrix and $\boldsymbol{\Phi}$ is referred to as the basis function matrix. By looking to the left nullspace of the system, the number of inputs and equations is the same, n_r. Before going further into the different model reduction techniques for linear and nonlinear systems, the proper orthogonal decomposition (POD) is introduced as a technique to calculate the reduced-order basis function matrix, $\boldsymbol{\Phi}$ [75, 76].

6.2 Proper Orthogonal Decomposition

Proper orthogonal decomposition (POD) is a technique used to compute the basis of the affine search subspace for the approximate solution. POD is chosen for its optimal compression property, which minimizes the sum of squares distances to vector snapshots of the available data [77, 76]. For model reduction of the primal solution, the vectors of interest are the states of the system. The collections of states is referred to as the snapshot matrix, $\boldsymbol{S}$.

When calculating the POD of the state snapshot matrix, $\boldsymbol{S}$, there are two critical parameters: the number of snapshots (n_s) and the number of reduced-order basis functions (n_r). n_s is the number of primal solutions in the snapshot matrix, $\boldsymbol{S} \in \mathbb{R}^{N \times n_s}$. It is important to note that $n_r \leq n_s$.

To generate the snapshot matrix, $\boldsymbol{S}$ in Figure 6.3, which consists of n_s primal solutions, experimentation is generally required to determine which precomputed primal solutions to use. Typically these primal solutions are taken from every few time steps in an unsteady calculation. A poor set of snapshots that do not exemplify the entire solution will lead to an ill-conditioned matrix. If there are too few snapshots, the resulting reduced solution will be under resolved.

$$\boldsymbol{S} = [\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_{n_s}] \in \mathbb{R}^{N \times n_s}, \tag{6.6}$$

where $\boldsymbol{u}_i$ is the primal state snapshots at time t_i. To implement POD, the singular-value decomposition of the snapshot matrix is found, $\boldsymbol{S} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$, and then the reduced-order basis matrix is obtained from the first n_r columns of $\boldsymbol{U}$,

$$\boldsymbol{\Phi} = \boldsymbol{U}(:, 1 : n_r) \in \mathbb{R}^{N \times n_r}. \tag{6.7}$$

Again, experimentation for chaotic flows is required to determine the optimal value of n_r

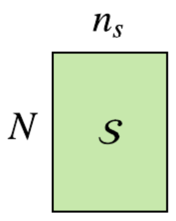


Figure 6.3: Snapshots of states are columns of the matrix, $\mathcal{S}$

in order to obtain an economical and accurate ROM. Higher n_r means more information of the full-order model is contained in the reduced-order model, but additional, it increases the computational cost of the reduced-order model construction and use.

6.3 Linear Model Reduction

For model reduction of linear systems, the governing equations of Eqn. 6.1 can be further linearized and used, given that only small perturbations in the flow exist. The first model reduction technique, which is the simplest, utilizes the POD method for the basis. In computational fluid dynamics, the unsteady governing equation can be represented as

$$\boldsymbol{M}\frac{d\boldsymbol{u}}{dt} + \boldsymbol{R}_s(\boldsymbol{u}, \boldsymbol{\mu}) = 0 \tag{6.8}$$

This can be written in the form of $\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}$,

$$\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{M}^{-1}\boldsymbol{R}_s(\boldsymbol{u}, \boldsymbol{\mu}). \tag{6.9}$$

One can linearize $\frac{d(\boldsymbol{u}+\delta\boldsymbol{u})}{dt}$ at point $(\boldsymbol{u}, \boldsymbol{\mu})$ to find

$$\frac{d\boldsymbol{u}}{dt} + \frac{d(\delta\boldsymbol{u})}{dt} = -\boldsymbol{M}^{-1}\boldsymbol{R}_s(\boldsymbol{u}, \boldsymbol{\mu}) - \boldsymbol{M}^{-1}\left.\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u},\boldsymbol{\mu}} \delta\boldsymbol{u} - \boldsymbol{M}^{-1}\left.\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{u},\boldsymbol{\mu}} \delta\boldsymbol{\mu}. \tag{6.10}$$

Moving $-\boldsymbol{M}^{-1}\boldsymbol{R}_s(\boldsymbol{u}, \boldsymbol{\mu})$ to the left side and given Eqn. 6.9, the linearize equation becomes

$$\frac{d(\delta\boldsymbol{u})}{dt} = -\boldsymbol{M}^{-1}\left.\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u},\boldsymbol{\mu}} \delta\boldsymbol{u} - \boldsymbol{M}^{-1}\left.\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{u},\boldsymbol{\mu}} \delta\boldsymbol{\mu}. \tag{6.11}$$

Setting the following to new terms,

$$\begin{aligned}
\boldsymbol{x} &= \delta \boldsymbol{u}, \\
\boldsymbol{y} &= \delta \boldsymbol{\mu}, \\
\boldsymbol{A} &= -\boldsymbol{M}^{-1} \left. \frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}} \right|_{\boldsymbol{u},\boldsymbol{\mu}}, \\
\boldsymbol{B} &= -\boldsymbol{M}^{-1} \left. \frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{\mu}} \right|_{\boldsymbol{u},\boldsymbol{\mu}}
\end{aligned} \tag{6.12}$$

will give the linearization in the familiar general form represented as

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}. \tag{6.13}$$

The reduced-order model of the system can be found by first substituting Eqn. 6.13 and Eqn. 6.3 into Eqn. 6.4, giving the reduced set of equations,

$$\boldsymbol{\varphi}^T \left[\boldsymbol{\Phi} \frac{d\boldsymbol{x}_r}{dt} = \boldsymbol{A}\boldsymbol{\Phi}\boldsymbol{x}_r + \boldsymbol{B}\boldsymbol{y} \right]. \tag{6.14}$$

The goal is to rewrite Eqn. 6.14 in terms of just x_r and find the new matrices $\boldsymbol{A}_r$ and $\boldsymbol{B}_r$,

$$\frac{d\boldsymbol{x}_r}{dt} = \underbrace{\left(\boldsymbol{\varphi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{A} \boldsymbol{\Phi}}_{\boldsymbol{A}_r} \boldsymbol{x}_r + \underbrace{\left(\boldsymbol{\varphi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{B}}_{\boldsymbol{B}_r} \boldsymbol{y} \tag{6.15}$$

$$\frac{d\boldsymbol{x}_r}{dt} = \boldsymbol{A}_r \boldsymbol{x} + \boldsymbol{B}_r \boldsymbol{y}. \tag{6.16}$$

Note that these matrices, $\boldsymbol{A}_r$ and $\boldsymbol{B}_r$, are referred to as offline matrices, because these matrices only have to be calculated once and is used to solve the governing equation at all time nodes. Substituting $\boldsymbol{A}$ and $\boldsymbol{B}$ into the reduced offline matrices gives,

$$\begin{aligned}
\boldsymbol{A}_r &= -\left(\boldsymbol{\varphi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{M}^{-1} \left. \frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}} \right|_{\boldsymbol{u},\boldsymbol{\mu}} \boldsymbol{\Phi} \\
\boldsymbol{B}_r &= -\left(\boldsymbol{\varphi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{M}^{-1} \left. \frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{\mu}} \right|_{\boldsymbol{u},\boldsymbol{\mu}}
\end{aligned} \tag{6.17}$$

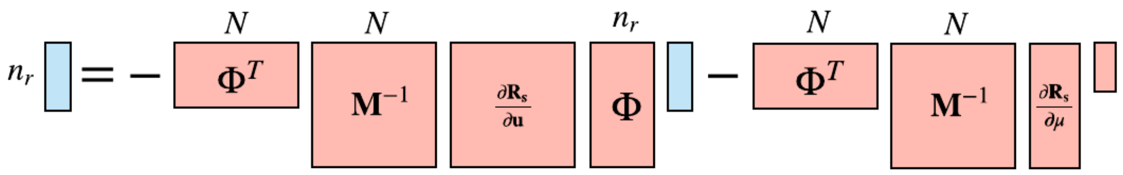
For a linear system, a Galerkin projection can be used for the left nullspace basis function, $\boldsymbol{\varphi}$ to evaluate the offline matrices:

$$\boldsymbol{\varphi} = \boldsymbol{\Phi} \tag{6.18}$$

Note that since $\boldsymbol{\Phi}$ is orthogonal, $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{I}$, an identity matix, giving,

$$\begin{aligned}\boldsymbol{A}_r &= -\boldsymbol{\Phi}^T\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u},\boldsymbol{\mu}}\boldsymbol{\Phi}, \\ \boldsymbol{B}_r &= -\boldsymbol{\Phi}^T\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u},\boldsymbol{\mu}}\end{aligned} \tag{6.19}$$

Once the basis is known, the offline matrices $\boldsymbol{A}_r$ and $\boldsymbol{B}_r$ can be calculated, and the new



n_r
N
N
n_r
N
N
$= -$
Φ^T
M^{-1}
$\frac{\partial R_s}{\partial u}$
Φ
$-$
Φ^T
M^{-1}
$\frac{\partial R_s}{\partial \mu}$

Figure 6.4: Galerkin Projection: Model Reduction for Linear Systems

reduced linearized equation can be solved forward in time to find the reduced states. The relative sizes of the offline matrices can be seen in the reduced linearized system in Figure 6.4. Once the reduced states have been found, the full states can be found by multiplying the reduced states by the reduced-order basis matrix $\boldsymbol{\Phi}$. This technique with POD does provide the most efficient representation of the states from the snapshot matrix and has been shown to be as stable as the full-order model; however, application of this method to a nonlinear system produces poor quality and inaccurate reduced-order models since the linearized POD can be inaccurately further away from its linearization point. The nonlinear evaluation of the residuals and the Jacobians is still on the order of the high fidelity model, cancelling out the strengths of the linear model reduction technique. For a chaotic system especially, a more robust technique that can truly eliminate the high cost of the forward problem is needed; however, the POD technique can still be used for nonlinear model reduction.

6.4 Nonlinear Model Reduction

There are numerous nonlinear techniques that can be used for model reduction of a nonlinear system. These techniques include the trajectory piecewise interpolation method [78, 79], the dynamic mode decomposition method [80, 81, 82], the combined least-squares Petrov-Galerkin (LSPG) and Gauss-Newton with approximated tensors (GNAT) method [83, 77, 76], and the discrete empirical interpolation method (DEIM) [84, 85, 86].

The trajectory piecewise interpolation method reduces the full-order model by constructing a continuous piecewise model of the linear PODs within the solution space; however, the method only produces a first-order approximation of the nonlinear problem. The dynamic mode decomposition method is generally more accurate than POD and evaluates the nonlinear dynamics of the problem by approximating the Koopman operator. The Koopman operator is a linear operator that governs observables, e.g. Mach number at a given point, along trajectories of a nonlinear system. Specifically, the operator lifts the dynamics from the state space to the observable space, where the observables can be represented as a linear expansion of the Koopman invariants that are part of the Koopman invariant subspace. DEIM applies a gappy POD on the nonlinear portion of the model, where the gappy algorithm reduces the cost of the nonlinear evaluation by sampling and interpolation; however, it is more inaccurate compared to the other methods. Carlberg et. al shows that DEIM is less accurate compared to the other methods due to nonlinear instabilities that occur over time [83]. Of the four methods, GNAT is the most accurate and stable for long time integrations; however, compared to the other methods, it is considerably more difficult to implement and intrusive to the main simulation program. In addition, GNAT has high offline costs that may not be suitable for some problems. For the purpose of output-based error estimation for chaotic systems, accuracy of long time-averages is essential, and hence the GNAT method is chosen to find the a reduced-order model for nonlinear chaotic systems. The reduced-order model calculated from GNAT will then be used in the reduced form of LSS in Chapter VII.

The least-squares Petrov-Galerkin method (LSPG) builds on the linear model reduction technique for nonlinear problems. It uses the POD method to find the basis of the affine search subspace, $\mathbf{\Phi}$, for the states $\boldsymbol{u}$ and looks to the left reduced-order basis $\boldsymbol{\varphi}$ to minimize the L_2 norm of the residual as seen in Eqn. 6.4. For the linear model reduction method, the Galerkin projection was used for $\mathbf{\Phi}$. For the least squares Petrov-Galerkin method, a different kind of projection that is able to encapsulate the nonlinear behaviors more accurately is used.

6.4.1 Orthogonal Projection with Newton's Method

After projecting the solution to a smaller affine subspace to find $\mathbf{\Phi}$, the reduced solution of the nonlinear problem is found by implementing the Newton iteration method [75, 76].

Each Newton iteration, k, begins by working on Eqn. 6.4. Orthogonalizing via the left nullspace basis functions, $\boldsymbol{\varphi} \in \mathbb{R}^{N \times n_r}$, gives

$$\boldsymbol{\Gamma} \equiv \boldsymbol{\varphi}^T \boldsymbol{R}(\boldsymbol{\Phi} \boldsymbol{u}_r) = \boldsymbol{\varphi}^T \left[\boldsymbol{M} \frac{d\left(\boldsymbol{\Phi} \boldsymbol{u}_r\right)}{dt} + \boldsymbol{R}_s(\boldsymbol{\Phi} \boldsymbol{u}_r) \right] = \boldsymbol{0}. \tag{6.20}$$

Applying the Newton method to Eqn. 6.20 in terms of the reduced states, $\boldsymbol{u}_r$, gives the state update $\boldsymbol{p}$,

$$\begin{aligned} \boldsymbol{p} = \boldsymbol{u}_{i+1} - \boldsymbol{u}_i &= -\frac{\boldsymbol{\Gamma}(\boldsymbol{\Phi} \boldsymbol{u}_{r_{i-1}})}{\nabla \boldsymbol{\Gamma}(\boldsymbol{\Phi} \boldsymbol{u}_{r_{i-1}})}, \\ &= -\frac{\boldsymbol{\varphi}^T \boldsymbol{R}(\boldsymbol{\Phi} \boldsymbol{u}_r)}{\boldsymbol{\varphi}^T \boldsymbol{J} \boldsymbol{\Phi}} \end{aligned} \tag{6.21}$$

where, i, refers to the time node. Simplifying Eqn. 6.21 gives,

$$\boldsymbol{\varphi}^T \boldsymbol{J} \boldsymbol{\Phi} \boldsymbol{p} = -\boldsymbol{\varphi}^T \boldsymbol{R}(\boldsymbol{\Phi} \boldsymbol{u}_r). \tag{6.22}$$

where $\boldsymbol{J}$ is the Jacobian of the residual, $\boldsymbol{R}$. The update for the reduced solution $\boldsymbol{u}_r$ for Newton iteration k at time node, i is

$$\boldsymbol{u}_{r,i}^{k+1} = \boldsymbol{u}_{r,i}^{k} + \alpha \boldsymbol{p}_i^k \tag{6.23}$$

where α represents the magnitude at which $\boldsymbol{p}$ should be imposed; for this research, α is set to 1. The Newton method gives the direction $\boldsymbol{p}$ and the update to $\boldsymbol{u}_r$. Once the Newton method drives $\|\boldsymbol{p}\|_2$ within a certain tolerance, ϵ, the full-order solution from the new reduced solution for the current time step can be found from Eqn. 6.3. ϵ is chosen to be close to zero; in this research it was set to $\epsilon = 10^-8$. The next step is to choose the left nullspace basis functions, $\boldsymbol{\varphi}$, that will best minimize the error in the search direction for each Newton iteration. This can be written as

$$\boldsymbol{p} = \arg\min \|\boldsymbol{\Phi} \boldsymbol{p} - \boldsymbol{J}^{-1} \boldsymbol{R}\|_2 \tag{6.24}$$

where $\boldsymbol{J}^{-1}\boldsymbol{R}$ is the full-order search direction. Note that Eqn. 6.24 decreases monotonically as the number of the basis functions in $\boldsymbol{\Phi}$ increases [76]. The left nullspace basis function matrix, φ should be chosen such that Eqn. 6.24 can be minimized correctly for the chosen problem.

6.4.2 Galerkin Projection

Setting the left nullspace basis function to be the reduced-order basis function matrix, $\boldsymbol{\Phi}$, -a typical choice for linear systems- is referred to as the Galerkin projection. Eqn. 6.22 then becomes

$$\boldsymbol{\Phi}^T \boldsymbol{J}\boldsymbol{\Phi}\boldsymbol{p} = -\boldsymbol{\Phi}^T \boldsymbol{R}(\Phi \boldsymbol{u}_r). \tag{6.25}$$

However, the Jacobians of a nonlinear problem are not in general symmetric positive definite (SPD) matrices. For Galerkin projection to work well, the Jacobians need to be SPD matrices. When the Jacobians, J, are SPD matrices, the search direction, $\boldsymbol{p}$ is optimal in Eqn. 6.24. In addition, Galerkin projection is a poor choice for nonlinear systems, because it will not be able to capture the nonlinear effects for each Gauss-Newton iteration in the orthogonal projection process used to find the search direction, $\boldsymbol{p}$. A different type of projection of the left nullspace basis function matrix is therefore needed.

6.4.3 Least-Squares Petrov-Galerkin Projection

For the the least-squares Petrov-Galerkin (LSPG) method, the left nullspace basis function matrix is chosen to be,

$$\boldsymbol{\varphi} = \boldsymbol{J}\boldsymbol{\Phi}. \tag{6.26}$$

The Petrov-Galerkin projection is chosen over the Galerkin projection, because Petrov-Galerkin projection is more accurate at capturing non-linear effects and is more stable for unsteady non-linear model reduction performed at the discrete level [76]. Since the projection is still based on the reduced-order basis function matrix, $\boldsymbol{\Phi}$, the fidelity of the reduced-order model from the LSPG method is mainly dependent on one input, the number of reduced-order basis functions, n_r. Setting $\boldsymbol{\varphi} = \boldsymbol{J}\boldsymbol{\Phi}$ and simplifying Equation 6.22 gives

$$\boldsymbol{\Phi}^T \boldsymbol{J}^T \boldsymbol{J}\boldsymbol{\Phi}\boldsymbol{p} = -\boldsymbol{\Phi}^T \boldsymbol{J}^T \boldsymbol{R}. \tag{6.27}$$

To avoid computing $\boldsymbol{J}^T\boldsymbol{J}$, it is important to note that Eqn. 6.27 is the normal equation form of the least-squares problem,

$$\boldsymbol{p} = \arg \min_{a \in \mathbb{R}^{n_r}} \|\boldsymbol{J}\boldsymbol{\Phi}\boldsymbol{a} + \boldsymbol{R}\|_2, \tag{6.28}$$

which can be solved using the thin QR factorization ($\boldsymbol{J}\boldsymbol{\Phi} = \boldsymbol{Q}_{\boldsymbol{J}\Phi}\boldsymbol{R}_{\boldsymbol{J}\Phi}$). Overall, the LSPG method refers to the use of the Petrov-Galerkin projection and the solving of the least-squares problem for each Gauss-Newton iteration. To find the direction of the update $\boldsymbol{p}$, the result of the QR factorization is used,

$$\boldsymbol{R}_{\boldsymbol{J}\Phi}\boldsymbol{p} = -\boldsymbol{Q}_{\boldsymbol{J}\Phi}^T \boldsymbol{R}. \tag{6.29}$$

6.4.4 The Gauss-Newton Method for Nonlinear Least-Squares Problems

The projection in section 6.4.3 is referred to as the Least-Squares Petrov-Galerkin projection. However, a nonlinear least-squares problem was not explicitly solved. The orthogonal projection used to solve for the reduced states can be seen as least squares problem as well,

$$\boldsymbol{\Gamma} = \min_{\boldsymbol{u}_r} \|\boldsymbol{R}(\boldsymbol{u}(\boldsymbol{u}_r))\|_2 \tag{6.30}$$

which is the minimization of the residual. The Gauss-Newton method is used to solve the nonlinear least-squares problem and is defined for the minimization problem as

$$\boldsymbol{p} = \boldsymbol{u}_t - \boldsymbol{u}_{t-1} = -\frac{\nabla\boldsymbol{\Gamma}(\boldsymbol{u}_{t-1})}{\nabla^2\boldsymbol{\Gamma}(\boldsymbol{u}_{t-1})}, \tag{6.31}$$

where $\nabla^2\boldsymbol{\Gamma}$ is the Hessian of the residual. The search direction used in this minimization satisfies

$$\boldsymbol{J}\Phi\boldsymbol{p} = -\boldsymbol{R}. \tag{6.32}$$

To improve the numeric conditioning of this equation, the thin QR decomposition is computed for $\boldsymbol{J}\Phi = \boldsymbol{Q}_{\boldsymbol{J}\Phi}\boldsymbol{R}_{\boldsymbol{J}\Phi}$ and then solved for $\boldsymbol{p}_i$ by

$$\boldsymbol{R}_{\boldsymbol{J}\Phi}\boldsymbol{p} = -\boldsymbol{Q}_{\boldsymbol{J}\Phi}^T\boldsymbol{R}. \tag{6.33}$$

Note that this equation is the same as Eqn. 6.29, which shows that the Petrov Galerkin projection is a optimal choice and further shows that setting $\boldsymbol{\varphi} = \boldsymbol{\Phi}\boldsymbol{J}$ for each Newton iteration is the least squares problem. As a result of this relationship, the projection method is referred to as the Least-Squares Petrov Galerkin method.

To solve Eqn. 6.33, use the Gauss-Newton iteration method similar to that in section 6.4.1. Once $\boldsymbol{p}$ is found at Gauss-Newton iteration k, calculate the step, α; in this research, α is set 1 for convenience. The update for the reduced solution at time i is

$$\boldsymbol{u}_{r,i}^{k+1} = \boldsymbol{u}_{r,i}^{k} + \alpha\boldsymbol{p}_i^k. \tag{6.34}$$

where the initial conditions for the reduced state at $t = 0$ is

$$\boldsymbol{u}_{r_0} = \boldsymbol{\Phi}^\dagger\boldsymbol{u}_0, \tag{6.35}$$

where $(\cdot)^\dagger$ refers to the Moore-Penrose pseudo-inverse of a matrix. Again, once the Gauss-

Algorithm 6.1 Reduced-Order Modeling Using LSPG with Gauss-Newton Method

```
Input: $\mathbf{\Phi}$, $\boldsymbol{u}_0$
Output: $\boldsymbol{u}$, $\boldsymbol{u}_r$
 1: for $i = 1, \dots n_T - 1$ do
 2:     $k = 0$
 3:     $\boldsymbol{u}_i = \boldsymbol{u}_{i-1}$, $\boldsymbol{u}_{r,i} = \mathbf{\Phi}^{\dagger} u_i$
 4:     while $\|\boldsymbol{p}\|_2 > \epsilon$ do                    ▷ $\epsilon$ is the desired tolerance value
 5:         Compute $\boldsymbol{R}_i^k(\boldsymbol{u}_i^k) = \boldsymbol{M}\frac{d\boldsymbol{u}}{dt} + \boldsymbol{R}_{s}{}_i^k(\boldsymbol{u}_i^k)$
 6:         Compute $\boldsymbol{J}_i^k(\boldsymbol{u}_i^k)\Phi$ where $\boldsymbol{J}_i^k(\boldsymbol{u}_i^k) = \frac{\partial \boldsymbol{R}_i^k(\boldsymbol{u}_i^k)}{\partial \boldsymbol{u}}$
 7:         Compute the thin QR factorization $\boldsymbol{J}_i^k(\boldsymbol{u}_i^k)\Phi = \boldsymbol{Q}_{\boldsymbol{J}\Phi,i}\boldsymbol{R}_{\boldsymbol{J}\Phi,i}$
 8:         Solve $\boldsymbol{R}_{\boldsymbol{J}\Phi,i}^k \boldsymbol{p}_i^k = -\boldsymbol{Q}_{\boldsymbol{J}\Phi}^{T^k} \boldsymbol{R}_i^k$
 9:         Set $\alpha_i^k = 1$
10:         Compute $\boldsymbol{u}_{r,i}^{k+1} = \boldsymbol{u}_{r,i}^k + \alpha_i^k \boldsymbol{p}_i^k$
11:         Compute $\boldsymbol{u}_i^{k+1} = \Phi \boldsymbol{u}_{r,i}^{k+1}$
12:         $k = k + 1$
13:     end while
14:     end while
15:     $\boldsymbol{u}_i = \boldsymbol{u}_i^{k+1}$
16:     $\boldsymbol{u}_{r,i} = \boldsymbol{u}_{r,i}^{k+1}$
17: end for
18: end for
```

Newton method drives $\|\boldsymbol{p}\|_2$ within a certain tolerance, ϵ, the full-order solution from the new reduced solution for the current time step can be found from Eqn. 6.3. An overview of LSPG with the Gauss-Newton method can be found in Algorithm 6.1.

6.4.5 Reduced-Order Model of the Kuramoto-Sivashinsky Equation

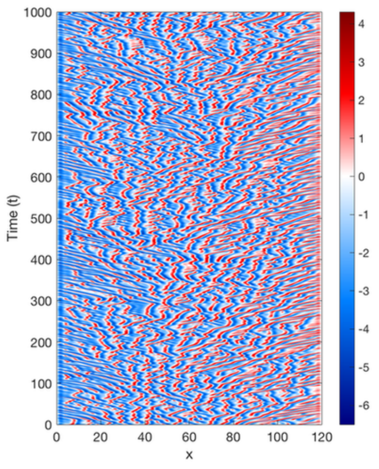
The LSPG method has been used successfully for nonlinear problems; however, there is a dearth of literature on the application of model reduction for nonlinear chaotic problems. In this section, the LSPG method is applied to the KS equation and the fidelity of the resulting ROMs is investigated. Again, the main parameter of LSPG is the number of basis functions, n_r, from the reduced-order basis function matrix, $\mathbf{\Phi}$. This parameter is varied in order to gain information on which value of n_r gives the best ROM for KS. The resulting ROM solutions are shown in Figure 6.5 and 6.6. The time-average output for the corresponding results are shown in Figure 6.7. For this particular case, the simulation time was set to $T = 1000$ and the burn time was set to $t_{\text{burn}} = 500$. To show the effectiveness of the ROM for KS for different values of n_r, the initial conditions are the same for each result in this section. The simulation contains $n_T = 50000$ temporal nodes and $n_s = 25000$ snapshots. The spatial interpolation order is set to $p = 3$ and second-order backward differencing (BDF2) is used for the time discretization.

In Figures 6.5 and 6.6, each unique value of n_r produces a ROM with significantly different trajectories. This is to be expected due to "butterfly effect", where any small perturbations can produce a different trajectory that diverges away from the original trajectory. Here, the small perturbation is due to the number of basis functions, n_r. For comparison, the original full-order solution can be seen in Figure 6.5(a). Figures 6.5(b)- 6.6(c) show trajectories for $n_r = 480 \to 150$. These produce unique trajectories compared to the original full-order model solution; however, the thickness of the coherent structures is similar. From $T = 0$ to $T = 1000$, the solutions continue to diverge and never reach visually periodic behavior, signifying that these values of n_r still do produce chaotic solutions. However, Figure 6.7 shows some distinctive differences among the trajectories, even though they look similar in $x - t$ plots. However, when the final time is set to $T = 1000$, the time-average outputs reach different values for different n_r. Note that as $T \to \infty$, not all of the time-average output histories for ROM will converge to one value. Several trajectories do attain approximately similar time-average outputs as the full-order model. These trajectories are for $n_r = 480, 375, 325$. Overall, for these n_r, the time-average outputs are within $\pm 5\%$ of the time-average output of the full-order model. Note that in Figure 6.6(c), the maximum magnitudes of the states begin to decrease compared to that of the full-order model.

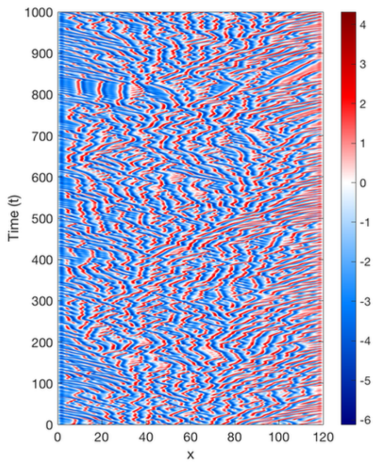
When n_r is decreased past $n_r = 150$, the trajectories begin to change as seen in Figure 6.6(d)- 6.6(i). The maximum magnitudes of the states further decrease with decreasing n_r. The spaces between the maxima and the minima of the states start to increase and some subtle periodic behavior begins to appear, which becomes even more evident by Figure 6.5(h). As the ROM becomes more periodic, the ROM solutions begin to act less chaotic and reach almost steady behaviors, as seen in Figure 6.6(i), where the trajectories are unrecognizable.

This behavior can be seen as well in Figure 6.7, where for $n_r = 5 - 125$, the time-average output is not the same as that of the full-order model by $T = 1000$. Through $n_r = 125$, the time-average output has lost enough information such that it reaches a steady time-average out prematurely. The non-chaotic behavior of Figure 6.6(i) is evident as the time-average output completely flattens out earlier than the other trajectories.

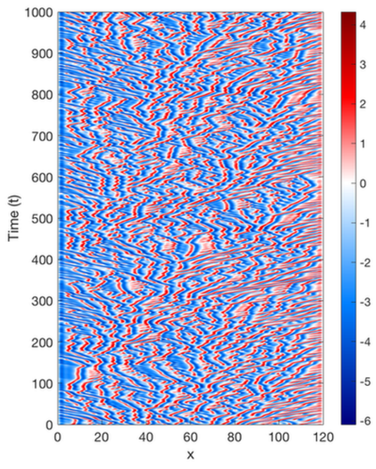
The increase in the non-chaotic nature with decreasing n_r affects the accuracy of the estimated time-averaged solution. Figure 6.7 indicates that one requirement to estimate the time-averaged solution accurately is that the ROM should also exhibit chaotic behavior, because the time-average of $n_r = 5$ reaches steady state quite quickly. One reason why the



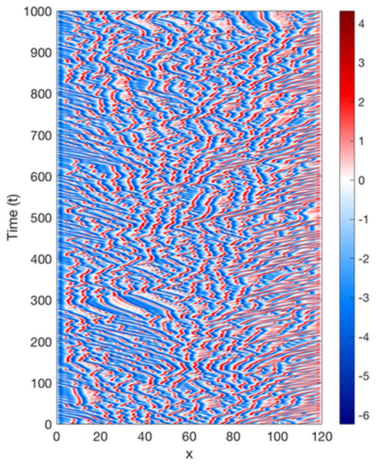
(a) primal



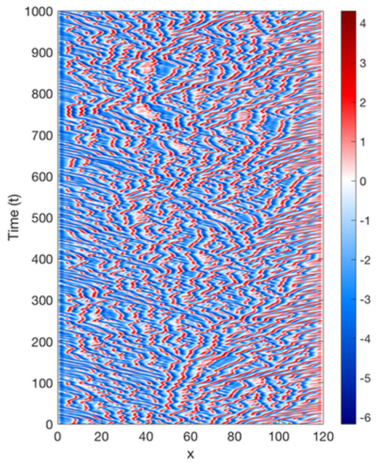
(b) $n_r = 480$



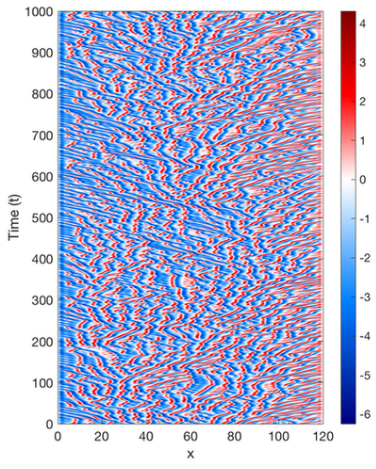
(c) $n_r = 400$



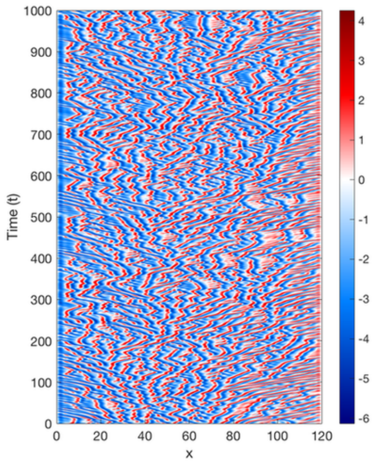
(d) $n_r = 375$



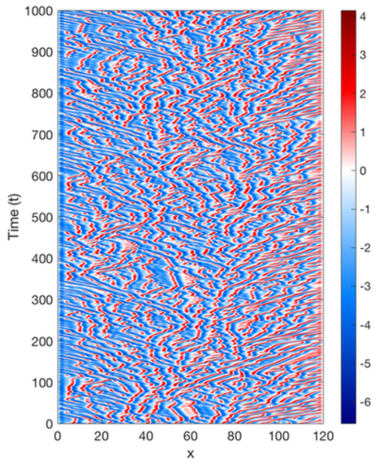
(e) $n_r = 350$



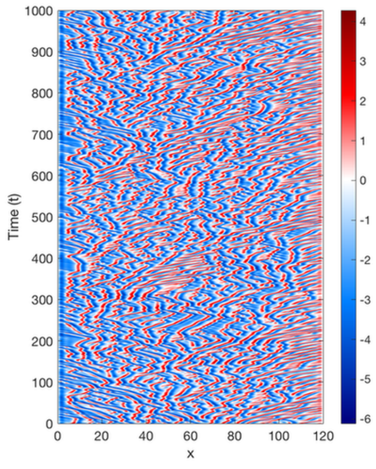
(f) $n_r = 325$



(g) $n_r = 300$

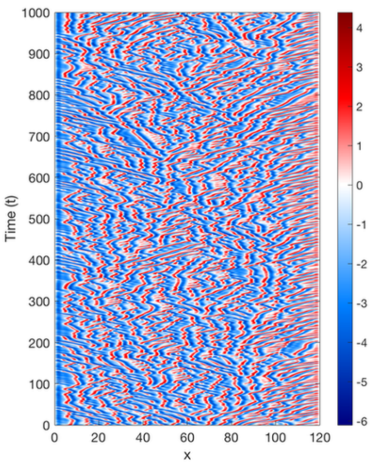


(h) $n_r = 275$

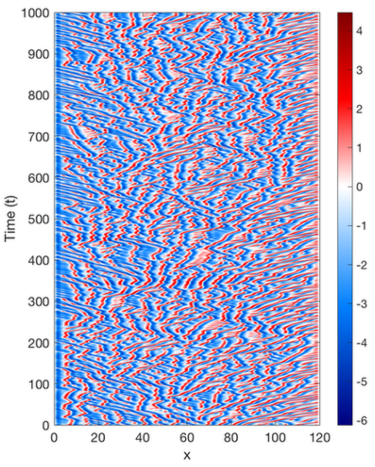


(i) $n_r = 250$

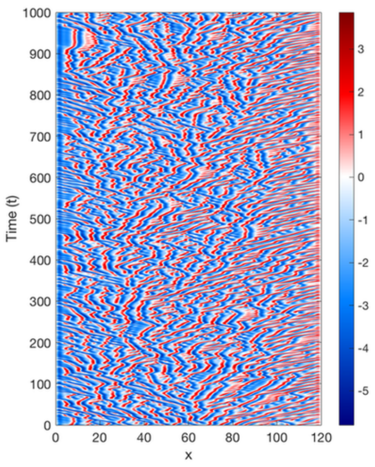
Figure 6.5: KS: state trajectories for different values of n_r.



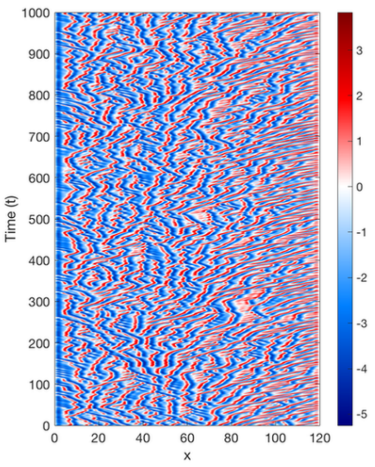
(a) $n_r = 225$



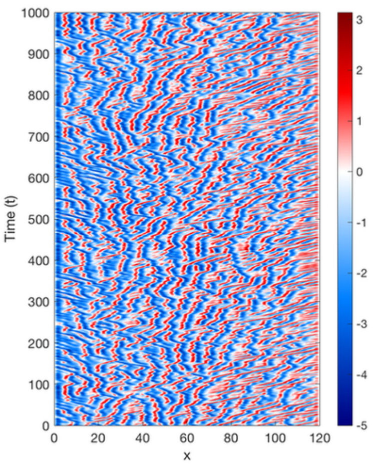
(b) $n_r = 175$



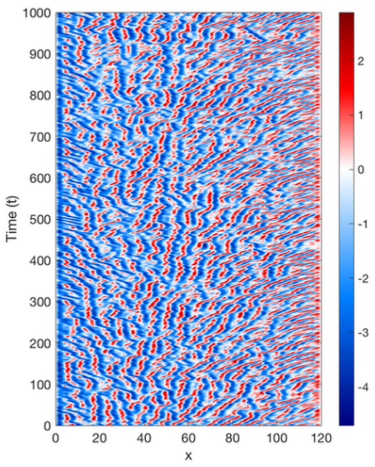
(c) $n_r = 150$



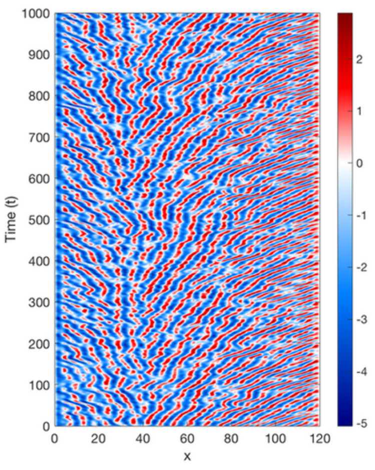
(d) $n_r = 125$



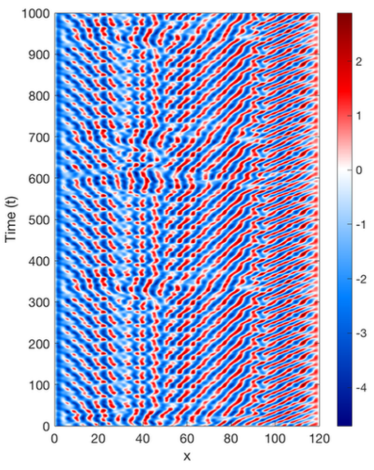
(e) $n_r = 75$



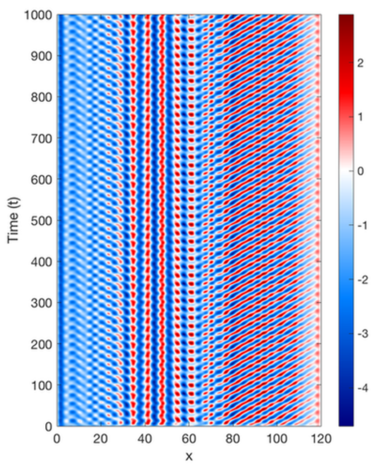
(f) $n_r = 50$



(g) $n_r = 25$



(h) $n_r = 10$



(i) $n_r = 5$

Figure 6.6: KS: state trajectories for different values of n_r.

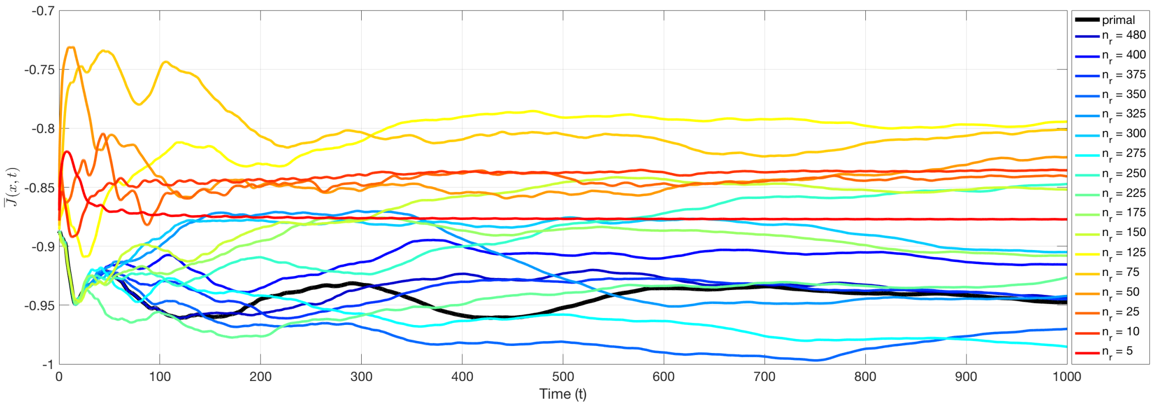


Figure 6.7: Time-average output for ROM with different values for n_r

trajectory for $n_r = 5$ may not accurately estimate the time-averaged output is that the ROM trajectory is under resolved. In order to preserve the chaotic characteristics of the original full-order model, more basis functions are needed.

Even though the ROMs for varying values of n_r produce trajectories that do not completely emulate the original model, these ROMs are still important. The most effective ROMs to use for error estimation is one in which n_r is small enough to be efficient without compromising the chaotic nature of the original problem. Based on the results of Figures 6.5, 6.6, and 6.7, it appears possible to find a working and accurate ROM for a chaotic system given enough reduced-order basis functions.

6.5 Hyper Model Reduction via Gauss-Newton with Approximated Tensors

For nonlinear model reduction, the LSPG method was introduced and implemented to see if it would be possible to find ROMs for chaotic systems, specifically for the KS equation. LSPG uses model reduction techniques developed for linear problems and sets the left nullspace basis function matrix to be $\boldsymbol{\varphi} = \boldsymbol{J\Phi}$ which is a more stable choice for nonlinear systems. The ROM is found via a minimization problem– setting $\boldsymbol{\varphi} = \boldsymbol{J\Phi}$ allows one to rewrite the orthogonal projection as a minimization problem– of the residual and the search direction is used to update the initial guess for the reduced solution $\boldsymbol{u}_r$. All of these take place for each Gauss-Newton iteration where the tolerance of the search direction is driven to ϵ, set by the user. Again, the LSPG method is named accordingly due to use of the Petrov-Galerkin projection and the solving of the least-squares problem for each Gauss-Newton iteration.

Next, it was shown that implementation of the LSPG method did give visually accurate ROMs for KS when comparing trajectories and the time-average output to that of the full-order solution. It was found that as the number of reduced-order basis functions decreased though, the ROMs became less accurate, less chaotic, and more periodic. By single digit n_r values, the trajectory and the time-average outputs exhibited steady-state behavior, making these unusable for further chaotic error estimation work. The best value for n_r is one that is as small as possible, yet still exhibits somewhat chaotic behaviors. By reducing the number of basis functions as much as possible, the computational costs savings of LSS can be reduced as well, leading to cheaper discretization error estimates for chaotic flows.

The next step for output-based error estimation for chaotic flows is to investigate whether or not the computational costs of finding the ROM can be further reduced. It is important to note that when finding the ROM for any n_r, the calculation of the residual and Jacobian is still dependent on the full-order model system size, N. The true savings in the computational costs will come from decreasing the time it takes to calculate the residual and Jacobian. This leads to the idea of decreasing the associated costs by only calculating n_i values for the residuals and the Jacobians where $n_i \leq N$ is the number of sample nodes. By doing this, a new hyper-reduction procedure can be put in place that produces a modified ROM that is even cheaper than LSPG on its own. This new hyper-reduction procedure for approximating the residuals and the Jacobians produces what is called the hyper-reduced-order model (HROM). To achieve hyper-reduction, the Gauss-Newton with approximated tensor quantities (GNAT) method is used; this technique introduces additional steps and approximations into the model reduction procedure for the residuals and the Jacobians [83]. For this section, we use GNAT to approximate the residuals and Jacobians by projecting them with the reduced-order basis matrices, $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$. The orthogonal projection method is then further expanded and manipulated to obtain the hyper-reduced-order model (HROM). The basic principles of HROM start with the an altered version of the approximate solution defined in Equation 6.3. The next section will go over how this new HROM is found and will compare results to its ROM counterpart.

6.5.1 Modified Projection-Based Model Reduction

To reduce the computational cost further, the full-order states are evaluated at n_e nodes, creating the restricted full-order state,

$$\overline{\boldsymbol{u}} \approx \overline{\boldsymbol{\Phi}}\boldsymbol{u}_r. \tag{6.36}$$

Given an identity matrix $\boldsymbol{I}_e$, where n_e specified rows -calculated indirectly from a greedy

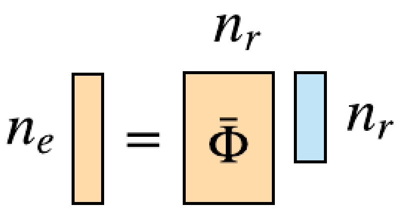


Figure 6.8: modified stated and reduced states based on the restricted number of elements and nodes, n_e

algorithm- are kept, the altered vector or matrix with n_e nodes can be found from

$$\overline{\boldsymbol{u}} = \boldsymbol{I}_e \boldsymbol{u}. \tag{6.37}$$

n_e is referred to the restricted nodes in the simulation. Note that $n_e \leq N$ and refers to the number of state vector indices required to reconstruct and approximate the residual and Jacobian accurately. The relationship between the restricted full-order states and the reduced states can be seen with its corresponding sizes in Figure 6.8. Note that how many n_e nodes to keep is dependent on the discretization method.

Another parameter of importance is the number of sample nodes n_i, which is the total number of nodes at which the residual and the Jacobian are calculated given n_e state nodes. n_i is determined via a Greedy algorithm, which in addition indirectly determines the total number of elemental nodes, n_j, where elemental nodes refer to all the nodes in each element that contain the n_i^{th} node. Note that $n_i \leq n_j$. In GNAT, the residual and the Jacobian functions need to be further modified in order to take into account the overall $N - n_i$ states that are not being calculated. These $N - n_i$ states can be referred to as "gaps". This modification is performed using the gappy reconstruction method [76, 77, 83]. This nodal framework is created this way to work with existing DG solvers. More on how $n_i,\ n_j,\ n_e$ are related will be explained in the next sections.

6.5.2 Greedy Algorithm for Computing Sample Nodes

Once the states have been defined mathematically, an efficient method is needed to calculate the number of sample nodes, or entries (nodes) to calculate important quantities such as the residual and the Jacobian. In total there are n_i of these nodes. By not sampling at all nodes, the computational cost of calculating a reduced-order model can be further reduced. Note that the number of sample nodes required for the residual and the Jacobian is not necessarily the same as the number of nodes required for the state, due to the fact that the residual and Jacobian evaluations are not completely local, i.e. component-wise, though they are element-wise compact. For this thesis, n_i was chosen to be the same for the states, the residuals, and the Jacobians. The chosen method "greedily" determines nodes that minimize the error in the gappy reconstruction of the nonlinear function in question. The output vector of sample node indices from the greedy algorithm is used reduce any vector and matrix to the number of sample nodes as shown in Figure 6.9. The greedy algo-

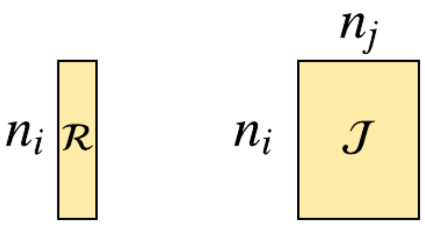


Figure 6.9: Residual and Jacobian matrices with only n_i sample nodes and n_j corresponding columns

rithm treats all state variables equally and avoids the need to sample indices individually. n_i refers to the number of sampled nodes from the greedy algorithm. Note that $n_i \geq n_r$. Once n_i is found, n_j can be determined. In DG, each element receives information from elements surrounding it. n_j includes all the nodes of n_i and all the nodes associated with the surrounding elements. The relationship among n_i, n_j, and n_e can be seen in an example mesh in Figure 6.10. The greedy algorithm is reproduced in Algorithm 6.2 from Carlberg et. al. [83].

6.5.3 Gappy Data Reconstruction for the Residual and Jacobian

Once the greedy algorithm is used to determine both the number of sample nodes, n_i, and the number of states, n_j, from elements that contain n_i, the resulting data needs to be reconstructed as a result of the missing values or "gaps". Define

$$\widehat{\boldsymbol{u}} = \boldsymbol{I}_i \boldsymbol{u}, \tag{6.38}$$

where the permutation matrix $\boldsymbol{I}_i$ contains only n_i rows pertaining to the specific samples node chosen from the greedy algorithm. This reconstruction process is called the gappy data reconstruction and was first introduced for image reconstruction [87]. In particular, it is important to find a way to recreate an approximate version of the full residuals and the Jacobians from smaller data sets, which can be viewed as data compression [76]. The

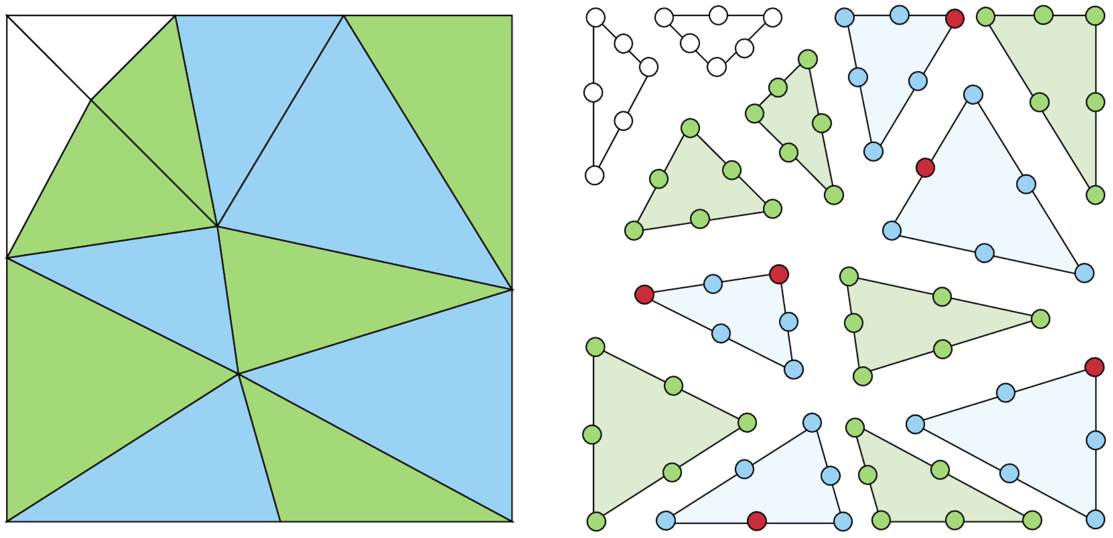


Figure 6.10: For this 2D $p = 2$ mesh example, red circles refer to the n_i sample nodes. Blue and red dots together refer to the n_j elemental nodes. Blue, red, and green dots together refer to the n_e restricted nodes, which define $\overline{\boldsymbol{\Phi}}$. Note, these choices apply to the DG framework.

gappy data reconstruction method computes an approximation of a desired vector $\boldsymbol{\Lambda}$ as

$$\tilde{\boldsymbol{\Lambda}} = \boldsymbol{\Phi}_{\boldsymbol{\Lambda}} \boldsymbol{\Lambda}_g, \tag{6.39}$$

where $\tilde{\boldsymbol{\Lambda}}$ refers to the approximation of $\boldsymbol{\Lambda}$. $\boldsymbol{\Lambda}_g$ is defined as,

$$\boldsymbol{\Lambda}_g = \arg \min_{\boldsymbol{a} \in \mathbb{R}^{n_{\boldsymbol{\Lambda}}}} \|\widehat{\boldsymbol{\Phi}}_{\boldsymbol{\Lambda}} \boldsymbol{a} - \widehat{\boldsymbol{\Lambda}}\|_2. \tag{6.40}$$

The solution to Equation 6.40 is

$$\boldsymbol{\Lambda}_g = \widehat{\boldsymbol{\Phi}}_{\boldsymbol{\Lambda}}^{+} \widehat{\boldsymbol{\Lambda}}. \tag{6.41}$$

where $(\cdot)^\dagger$ refers to the Moore-Penrose pseudo-inverse of a matrix. Next, the non-linear residual and Jacobian can be approximated using the gappy reconstruction method as

$$\widetilde{\boldsymbol{R}} \approx \boldsymbol{\Phi}_R \boldsymbol{R}_g, \tag{6.42}$$

$$\widetilde{\boldsymbol{J\Phi}} \approx \boldsymbol{\Phi}_J \boldsymbol{J}_g, \tag{6.43}$$

where $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$ are new reduced-order basis function matrices for the residual and Jacobian, which are chosen to of accurately approximate the full residual and Jacobian values

Algorithm 6.2 Greedy Algorithm for Gauss-Newton with Approximated Tensors Method

```
Input: $\mathbf{\Phi}_R$, $\mathbf{\Phi}_J$, initialized vector $\mathcal{N}$, $n_i$, $n_{RJ}$ where $n_{RJ} \le n_i$
Output: Complete vector, $\mathcal{N}$
 1: $n_a = n_i - |\mathcal{N}|$        ▷ Compute the number of sample nodes needed
 2: $n_b = 0$        ▷ Initialize counter for number of workings basis vectors used
 3: $n_{it} = \min(n_{RJ}, n_a)$        ▷ Set the number of greedy iterations to perform
 4: $n_{RHS} = \left\lceil \frac{n_{RJ}}{n_a} \right\rceil$        ▷ Maximum number of right-hand sizes in least-squares problem
 5: $n_{RJi,\min} = \left\lfloor \frac{n_{RJ}}{n_{it}} \right\rfloor$        ▷ Minimum number of working basis vectors per iteration
 6: $n_{ai,\min} = \left\lfloor \frac{n_a n_{RHS}}{n_{RJ}} \right\rfloor$        ▷ Minimum number of sample nodes to add per iteration
 7: for $i = 1, \dots n_{it}$ do        ▷ Greedy iteration loop
 8:     $n_{RJi} \leftarrow n_{RJ,\min}$        ▷ Number of working basis functions for $i$
 9:     if $i \le n_{RJ}$ mod $n_{it}$ then
10:         $n_{RJi} \leftarrow n_{RJi} + 1$
11:     end if
12:     $n_{ai} \leftarrow n_{ai,\min}$        ▷ Number of sample nodes to add for $i$
13:     if ($n_{RHS} = 1$ and $i \le n_a$ mod $n_{RJ}$) then
14:         $n_{ai} \leftarrow n_{ai} + 1$
15:     end if
16:     if i = 1 then
17:         $[R^1 \dots R^{n_{RJi}}] \leftarrow [\phi_R^1 \dots \phi_R^{n_{RJi}}]$        ▷ $\phi_R^i$ refers to column of $\mathbf{\Phi}_{\boldsymbol{R}}$
18:         $[J^1 \dots J^{n_{RJi}}] \leftarrow [\phi_J^1 \dots \phi_J^{n_{RJi}}]$        ▷ $\phi_J^i$ refers to column of $\mathbf{\Phi}_{\boldsymbol{J}}$
19:     else
20:         for $j = 1, \dots, n_{RJi}$ do        ▷ basis vector loop
21:             $\alpha = \arg \min_{\gamma \in \mathbb{R}^{n_b}} \| [Z\phi_R^1 \dots Z\phi_R^{n_b}]\, \gamma - Z\phi_R^{n_b+j} \|_2$        ▷ $Z$ is sample matrix
22:             $\beta = \arg \min_{\gamma \in \mathbb{R}^{n_b}} \| [Z\phi_J^1 \dots Z\phi_J^{n_b}]\, \gamma - Z\phi_J^{n_b+j} \|_2$
23:             $R^j \leftarrow \phi_R^{n_b+j} - [\phi_R^1 \dots \phi_R^{n_b}]\, \alpha$
24:             $J^j \leftarrow \phi_J^{n_b+j} - [\phi_J^1 \dots \phi_J^{n_b}]\, \beta$
25:         end for
26:     end if
27:     for $j = 1, \dots, n_{ai}$ do        ▷ Sample node loop
28:         $n \leftarrow \arg\max_{l \notin \mathcal{N}} \sum_{j=1}^{n_{RJi}} \sum_{i \in \delta(l)} \left[ (R_i^j)^2 + (J_i^j)^2 \right],$        ▷ $\delta(l)$ is the DOF for node, $l$
29:         $\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$        ▷ Add new sample nodes to sample node set
30:     end for
31:     $n_j \leftarrow n_j + n_{RJi}$        ▷ total number of sample nodes found
32: end for
```

given the sample node solutions. The overall new approximations for these quantities are

$$\widetilde{\boldsymbol{R}} \approx \boldsymbol{\Phi}_R \widehat{\boldsymbol{\Phi}}_R^+ \widehat{\boldsymbol{R}}, \qquad \widetilde{\boldsymbol{J\Phi}} \approx \boldsymbol{\Phi}_J \widehat{\boldsymbol{\Phi}}_J^+ \widehat{\boldsymbol{J\Phi}}. \tag{6.44}$$

The corresponding matrices and their sizes for the approximated residuals and Jacobians can be seen in Figure 6.11. The approximation of the residuals and the Jacobians is a

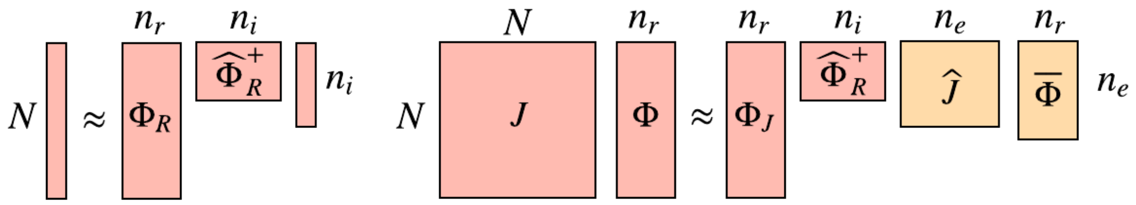


Figure 6.11: Approximations for the residual and the Jacobian as a function of reduced-order basis functions

function of $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$, which are reduced-order basis function matrices for the residuals and the Jacobians respectively and are similar to the reduced-order basis function for the states, $\boldsymbol{\Phi}$. The next section will go over how $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$ are calculated.

6.5.4 Calculation of Reduced-order Basis Functions for the Residual and Jacobian

Unlike the calculation of the state basis functions, $\boldsymbol{\Phi}$, there is only one critical parameter to set for the calculation of the residual reduced-order basis matrix, $\boldsymbol{\Phi}_R$, and the Jacobian reduced-order basis matrix, $\boldsymbol{\Phi}_J$: the number of residual reduced-order basis functions (columns) n_{RJ}, which is different than n_s. It is important that $n_i \geq n_{RJ}$ enable a unique approximation for the residuals and the Jacobians. The calculation of $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$ begins with the generation of the residual and the Jacobian snapshot matrices, $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$. To find the state snapshots $\boldsymbol{\mathcal{S}}$, the user chooses which states at a particular time t to add; however, this is not the case for $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$. Instead, the snapshot matrix is generated during the ROM stage where, during each Gauss-Newton iteration, the information of the residual and the Jacobian is saved. The number of columns of the $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$ matrices is the total number of Gauss-Newton iterations in the entire reduced-order model process, n_{ST}. This is necessary in order to preserve the nonlinear behavior of governing equation. Due to large number of Gauss-Newton iterations to find ROM, the residual and Jacobian data are saved to a file on disk before being called to create the snapshot matrix. $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$ take the form,

$$\boldsymbol{S}_R = [\boldsymbol{R}_0,\ \boldsymbol{R}_1,\ \ldots,\ \boldsymbol{R}_{n_{ST-1}}] \in \mathbb{R}^{N \times n_{ST-1}}, \tag{6.45}$$

$$\boldsymbol{S}_J = [\boldsymbol{J}_0 \boldsymbol{\Phi} \boldsymbol{p}_0,\ \boldsymbol{J}_1 \boldsymbol{\Phi} \boldsymbol{p}_1,\ \ldots,\ \boldsymbol{J}_{n_{ST-1}} \Phi \boldsymbol{p}_{n_{ST-1}}] \in \mathbb{R}^{N \times n_{ST-1}}. \tag{6.46}$$

where the indices on R, J, and p refer to the corresponding Gauss-Newton iteration. Other

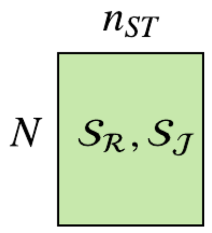


Figure 6.12: Snapshots for the residual and the Jacobian.

alternatives to find the $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$ can be found in Carlberg et. al. [76]; however, for chaotic systems, Eqns. 6.45 and 6.45 have been found to work well. Once the snapshot matrices are defined, POD is performed on these matrices. After performing singular value decomposition of the snapshot matrices, $\boldsymbol{S_R} = \boldsymbol{U_R} \boldsymbol{\Sigma} \boldsymbol{V_R}^T$ and $\boldsymbol{S_J} = \boldsymbol{U_J} \boldsymbol{\Sigma} \boldsymbol{V_J}^T$, the residual-reduced-order basis matrix and the Jacobian-reduced-order basis matrix are chosen as

$$\boldsymbol{\Phi}_R = \boldsymbol{U_R}(:, 1 : n_{RJ}) \quad \boldsymbol{\Phi}_J = \boldsymbol{U_J}(:, 1 : n_{RJ}). \tag{6.47}$$

6.5.5 LSPG Projection with GNAT

The addition of the GNAT algorithm to LSPG to find HROM requires additional work offline compared to standard ROM. During the offline stage, residual and Jacobian information is retrieved and used for each Gauss-Newton iteration of ROM. Once the residual-reduced-order basis matrix and the Jacobian-reduced-order basis matrix are created, the offline matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are created and are used to help approximate the residuals and the Jacobians for each new Gauss-Newton iteration of HROM. More computational effort is required in the offline stage, but this cost is outweighed by the cost savings associated with the Gauss-Newton stage. $\boldsymbol{A}$ and $\boldsymbol{B}$ are first created by substituting both Eqn. 6.42 and Eqn. 6.43 into Eqn. 6.28 to get a modified minimization problem,

$$\boldsymbol{p} = \arg \min_{\boldsymbol{a} \in \mathbb{R}^{n_r}} \|\boldsymbol{\Phi}_J \widehat{\boldsymbol{\Phi}}_J^+ \widehat{\boldsymbol{J}\boldsymbol{\Phi}} \boldsymbol{a} + \boldsymbol{\Phi}_R \widehat{\boldsymbol{\Phi}}_R^+ \widehat{\boldsymbol{R}}\|_2. \tag{6.48}$$

$\boldsymbol{C}$ is defined as

$$\boldsymbol{C} = \boldsymbol{\Phi}_J \widehat{\boldsymbol{\Phi}}_J^+ \widehat{\boldsymbol{J\Phi}}. \tag{6.49}$$

One can then perform a QR factorization

$$\boldsymbol{C} = \boldsymbol{Q}_c \boldsymbol{R}_c, \quad \boldsymbol{A} = \boldsymbol{R}_c, \quad \boldsymbol{B} = \boldsymbol{Q}_c^T \Phi_R \widehat{\boldsymbol{\Phi}}_R^+. \tag{6.50}$$

We rearrange Equation 6.48 to obtain

$$\boldsymbol{p} = \arg \min_{\boldsymbol{a} \in \mathbb{R}^{n_r}} \|\boldsymbol{A}\widehat{\boldsymbol{\Phi}}_J \boldsymbol{a} + \boldsymbol{B}\widehat{\boldsymbol{R}}\|_2. \tag{6.51}$$

We solve this minimization problem in the same way as we did for ROM to obtain,

$$\boldsymbol{R}_{A\hat{J}\hat{\Phi}} \boldsymbol{p} = -\boldsymbol{Q}^T_{A\widehat{J\Phi}} \boldsymbol{B}\widehat{\boldsymbol{R}}. \tag{6.52}$$

Once again, α is set to 1 and the reduced state is updated via

$$\overline{\boldsymbol{u}}_r^{k+1} = \overline{\boldsymbol{u}}_r^k + \alpha^k \boldsymbol{p}^k, \tag{6.53}$$

$$\boldsymbol{u}^{k+1} = \overline{\Phi} \boldsymbol{u}_r^{k+1}, \tag{6.54}$$

where the initial reduced state at $t = 0$ is the same as that of ROM in Eqn. 6.35. Once the Gauss-Newton method drives $\|\boldsymbol{p}\|_2$ to within a certain tolerance, ϵ, the full-order solution from the new hyper-reduced solution for the current time step can be found from Eqn. 6.36. Note that k refers to the k^{th} Gauss-Newton iteration.

The online calculation of the Gauss-Newton method for HROM can be found in Algorithm 6.3, which is a modified version of Algorithm 6.1. Overall, these residual and Jacobian approximations eliminate the dependency of the solution procedure on N, the large dimension of the full-order state approximation. Once these approximate substitutions are made, the iteration process can proceed to find the HROM model of KS and NS, which is a better candidate than ROM for use with LSS.

6.5.6 Hyper-Reduced-Order Model of the Kuramoto-Sivashinsky Equation

Section 6.4.5 demonstrated that with the correct number of reduced-order basis functions, n_r, recovering an accurate ROM of a chaotic system is possible. It was shown that the overall size of the full-order system for KS can be reduced significantly; however, a minimum number of reduced-order basis functions for the states is required. Providing fewer basis

Algorithm 6.3 Hyper-Reduced-Order Modeling with GNAT

```
Input: $\overline{\Phi}$, $\overline{\boldsymbol{u}}_0$, $\boldsymbol{A}$, $\boldsymbol{B}$
Output: $\overline{\boldsymbol{u}}$, $\boldsymbol{u}_r$
 1: for $t = 1, \ldots n_T - 1$ do
 2:     $k = 0$
 3:     $\boldsymbol{u}_r = 0, \overline{\boldsymbol{u}}_t = \overline{\boldsymbol{u}}_{t-1}$
 4:     while $\|\boldsymbol{p}\|_2 > \epsilon$ do                          ▷ $\epsilon$ is the desired tolerance value
 5:         Compute $\widehat{\boldsymbol{R}}_t^k(\boldsymbol{u}_t^k)$
 6:         Compute $\widehat{\boldsymbol{J}_t^k(\overline{\boldsymbol{u}}_t^k)}\overline{\Phi}$ where $\widehat{\boldsymbol{J}_t^k(\overline{\boldsymbol{u}}_t^k)} = \frac{\partial \widehat{\boldsymbol{R}}_t^k(\overline{\boldsymbol{u}}_t^k)}{\partial \boldsymbol{u}}$
 7:         Compute $\boldsymbol{D}_t^k = \boldsymbol{B}\widehat{\boldsymbol{R}}_t^k$
 8:         Compute $\boldsymbol{E}_t^k = \boldsymbol{A}\widehat{\boldsymbol{J}_t^k(\boldsymbol{u}_t^k)}\overline{\Phi}$
 9:         Compute the thin QR factorization $\boldsymbol{E}_t^k = \boldsymbol{Q}_E\boldsymbol{R}_E$
10:         Solve $\boldsymbol{R}_E\boldsymbol{p}_t^k = -\boldsymbol{Q}_E^T\boldsymbol{D}_t^k$
11:         Compute $\alpha_t^k$ from a line search
12:         Compute $\boldsymbol{u}_{r,t}^{k+1} = \boldsymbol{u}_{r,t}^k + \alpha_t^k\boldsymbol{p}_t^k$
13:         Compute $\overline{\boldsymbol{u}}_t^{k+1} = \overline{\Phi}\boldsymbol{u}_{r,t}^{k+1}$
14:         $k = k + 1$
15:     end while
16:     end while
17:     $\overline{\boldsymbol{u}}_t = \overline{\boldsymbol{u}}_t^{k+1}$
18:     $\boldsymbol{u}_{r,t} = \boldsymbol{u}_{r,t}^{k+1}$
19: end for
20: end for
```

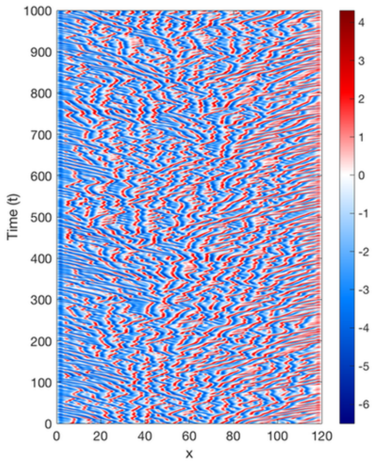
functions than required produces a periodic system that may not be accurate enough to be used for adjoint calculation for output-based error estimation.

It is important to remember that the process of creating the ROM, which represents the solution for the KS equation, in Section 6.4.5, is still a function of the full-order system size, N. As a result, calculating the residuals and the Jacobians of the ROM is still a function of N as well. To remedy this high cost, the GNAT method shown in Algorithm 6.3 to approximate the residual and the Jacobians. The combined GNAT method produces the hyper-reduced-order model (HROM), which does not require the calculation of the states at all N spatial nodes. As a result, the GNAT method introduces two additional parameters: the number of basis functions for the residual and Jacobian, n_{RJ}, and the number of sample nodes, n_i. Here, $n_i \leq N$ and $n_{RJ} \leq N$. An investigation is needed into whether or not it is possible to further approximate the states for a chaotic system; the GNAT method is applied to KS to find HROM.

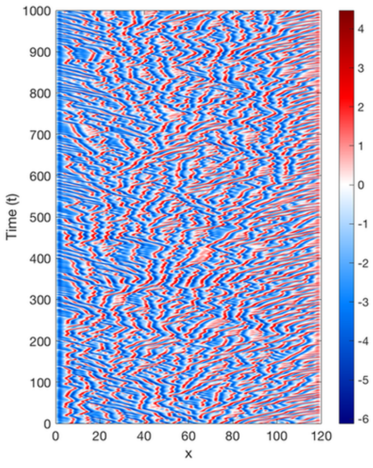
To quantify the relationship among the number of basis functions n_r, the number of sample nodes, n_i, and the number of basis functions for the residuals and the Jacobians, n_{RJ}, two different n_r choices are investigated for various values of n_i and n_{RJ} on the same test case from section 6.4.5, where the final simulation time was set to $T = 1000$. In order to ensure that the approximated residuals and Jacobians are unique, it is important that $n_i \geq n_{RJ}$, which dictates how large n_{RJ} can be.

First a large n_r is investigated. The HROM result for $n_r = 175$ is shown in Figure 6.13. Each sub graph refers to HROM for a unique number of sample nodes with various corresponding values of n_{RJ}. Figure 6.13(a) and Figure 6.13(b) are the same primal and ROM solution from section 6.4.5. Due to the requirement that $n_i \geq n_{RJ}$, the minimum number of sample nodes, n_i, for $n_r = 175$ is, $n_i = 380$. The HROM for $n_i = 380,\ n_{RJ} = 340$ is shown in Figure 6.13(f). For $n_r = 175$, the minimum number of basis functions for the residuals and Jacobians was found to be $n_{RJ} \approx 340$. Figures 6.13(c), 6.13(d), 6.13(e) show HROMs for intermediate values of the number of sample nodes and the number of basis functions for the residuals and Jacobians. Figure 6.13 shows HROM trajectories that diverge away from the primal and ROM trajectories, characteristic of chaotic systems. Each of these trajectories is unique and still exhibits chaotic behavior with time; however, to access the validity of these HROMs, the time-average output for each of these trajectories needs to be compared to that of the ROM. The time-average output for the $n_r = 175$ test case can be seen in Figure 6.14 for $n_i = 480, 460, 440, 420, 380$. Note that $n_i = 480$ is the maximum number of sample nodes for a $p = 3$ DG discretization. More test cases than the one shown in Figure 6.13 are shown for the time-average output in order to better understand the time-average output trends. The test cases from Figure 6.14 that are not shown in Figure 6.13 exhibit unique chaotic trajectories as well. Each subfigure in Figure 6.14 refers to a set of trajectories for a different number of sample nodes, n_i, and the corresponding ROM for comparison. Each unique trajectory for a given n_i, refers to the time-average output with a different number of basis functions for the residual and the Jacobian. The number of basis functions for the residual and Jacobian, n_{RJ}, and the number of samples nodes, n_i, were chosen in increments of 20. The goal is to see if the trajectories with a fewer number of sample nodes and a fewer number of basis functions for the residual and Jacobian still produce an accurate HROM, by looking at the time-average output.

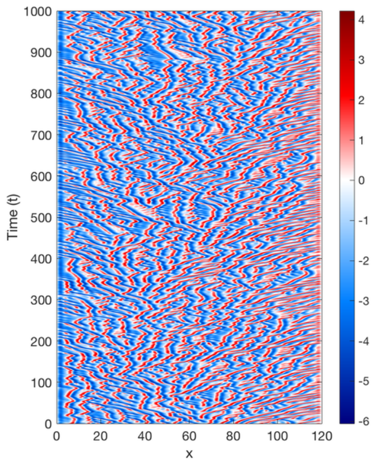
Figure 6.14(a) shows the time-average output for $n_i = 480$ and $n_{RJ} \in [480, 340]$. For $n_r = 175$ and $n_i = 480$, the minimum number of sample nodes required to successfully produce a HROM is $n_i = 340$. Below $n_i = 340$, the GNAT method fails to converge. For $n_i = 480$, the time-average output $\overline{J}$ of the HROMs underestimates the time-average output of the ROM. At $T = 1000$, $\overline{J}$ of the ROM is within ± 0.04 of the $\overline{J}$ of the HROM. In other words, at $T = 1000$, the percentage error of the HROMs is about 4.4% of the



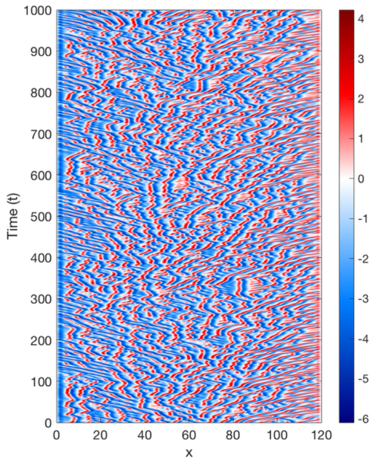
(a) primal



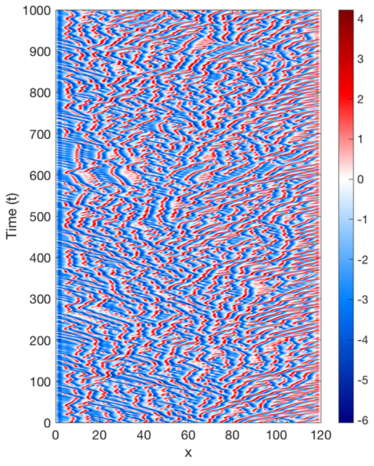
(b) ROM



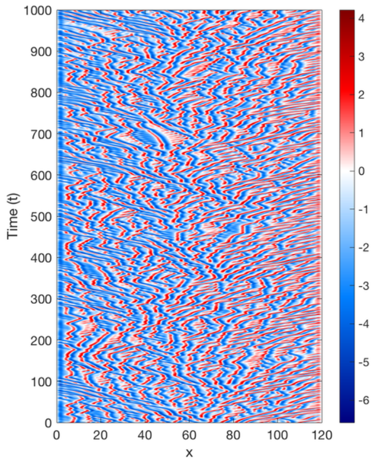
(c) $n_i = 480,\ n_{RJ} = 480$



(d) $n_i = 480,\ n_{RJ} = 360$



(e) $n_i = 420,\ n_{RJ} = 360$

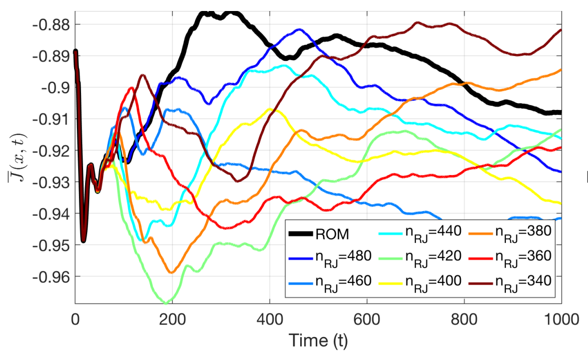


(f) $n_i = 380,\ n_{RJ} = 340$

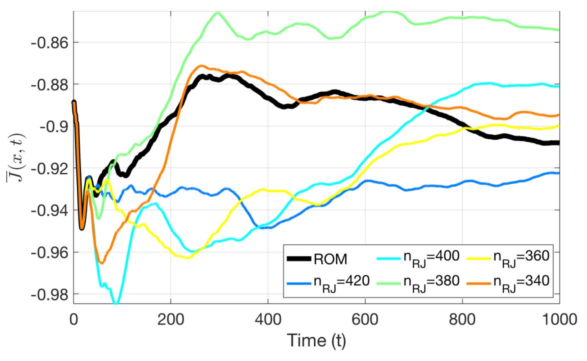
Figure 6.13: KS: Primal, ROM, and selected HROM trajectories for $n_r = 175$

ROM. Figure 6.14(b) shows the time-average output for $n_i = 460$ and $n_{RJ} \in [420, 340]$. $n_{RJ} = 460$ and $n_{RJ} = 440$ did not converge. For $n_i = 460$, there are fewer possible values of n_{RJ} that can be used in order to obtain a unique approximation for the residuals and the Jacobians. It is interesting to note that even though $n_{RJ} = 340$ is the minimum number of basis functions for the residuals and Jacobians required for $n_i = 460$, it closely emulates the time-average trajectory of the ROM. At $T = 1000$, the time-average output of n_{RJ} is off by approximately 2.1%.

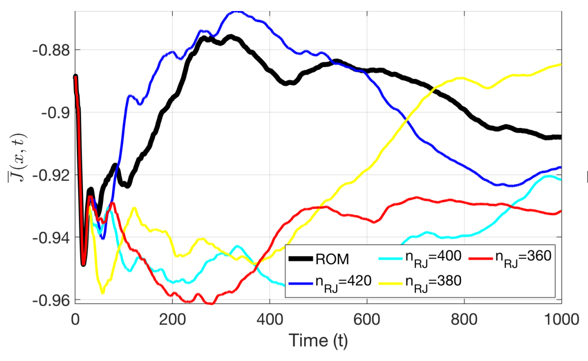
Figures 6.14(c), 6.14(d), and6.14(e) show similar behaviors for $n_i = 440, 425, 440$; however, as the number of sample nodes decreases the possible number of basis functions for the residuals and the Jacobians decreases as well, reducing the number of possible HROMs overall. The HROMs for $n_i = 440, 425$, and 440 show similar levels of accuracy



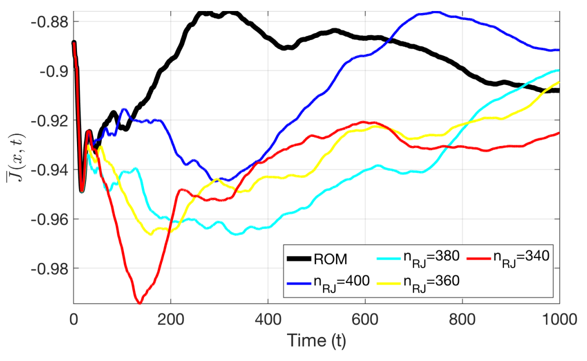
(a) $n_i = 480$



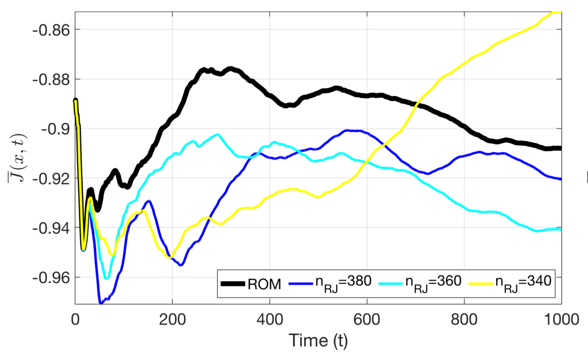
(b) $n_i = 460$



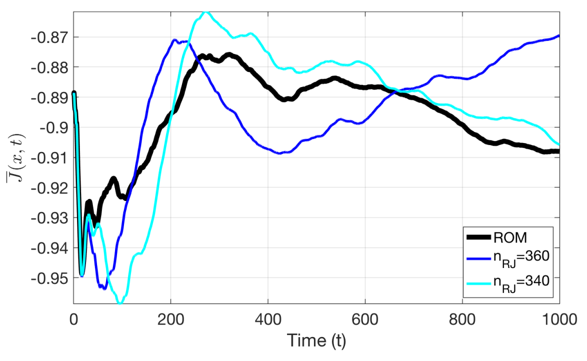
(c) $n_i = 440$



(d) $n_i = 420$



(e) $n_i = 400$



(f) $n_i = 380$

Figure 6.14: KS: Time-average outputs for HROM trajectories for $n_r = 175$.

as that of Figure 6.14(a) and Figure 6.14(b).

Figure 6.14(f) shows the time-average output for $n_i = 380$, which is approximately the minimum number of sample nodes possible to produce a HROM for $n_r = 175$. The trajectory for $n_{RJ} = 340$ is interesting, because even though it has the fewest number of sample nodes for n_r shown in the Figure 6.14, it emulates the ROM the best. This is most likely do to chance and better understanding of this phenomena requires running these test cases for much longer times.

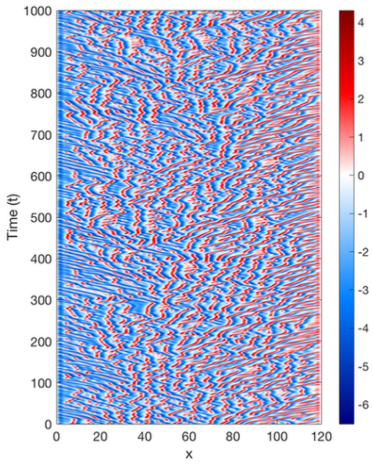
Overall, the HROM results for n_r shows that it is possible to further reduce the cost associated with ROM by using GNAT to approximate the residuals and the Jacobians as shown in Figure 6.14. The smallest number of sample nodes and basis functions for the residuals and Jacobians will give a HROM associated with the lowest computational cost. Hence, for the $n_r = 175$ test case, the best HROM to use for adjoint calculations for $T = 1000$ is $n_i = 380$ and $n_{RJ} = 340$. However, 380 sample nodes out of the possible $N = 480$ spatial nodes for $p = 3$, is still high. This refers to only a 20% reduction in the number of evaluations for the residuals and Jacobians. In addition, because the number of sample nodes is still quite high, the number of evaluated nodes, n_e will be quite high and/or equal to N. Reducing the number of sample nodes and hence the basis nodes for the residual and the Jacobian, n_{RJ}, even more will be more beneficial for output-based error estimation of chaotic flows. The next test case for $n_r = 50$, will be investigated to see if it produces similar results as $n_r = 175$ and requires fewer samples nodes, n_i. Figure 6.15 shows sample test cases for $n_r = 50$. Each sub graph refers to HROM as well for a unique number of sample nodes with corresponding values of n_{RJ}. Again, Figure 6.15(a) and Figure 6.15(b) are the same primal and ROM solution from Section 6.4.5. The requirement that $n_i \geq n_{RJ}$ still applies. As a result, the minimum number of sample nodes n_i is approximately $n_i = 180$, which is far lower than that of $n_r = 175$. For $n_r = 50$, the minimum number of basis functions for the residuals and the Jacobians was found to be $n_{RJ} \approx 160$, which again is far lower than that of $n_r = 175$.

Figures 6.15(c)-6.15(e) show a HROM for $n_i = 480$ and $n_{RJ} = 480,\ 320,\ 180$. As before with the $n_r = 175$ test case, the GNAT produces HROM trajectories that diverge away from the corresponding ROM. Even for $n_{RJ} = 180$, the HROM trajectory in Figure 6.15(e) is still chaotic and exhibits similar coherent structures.

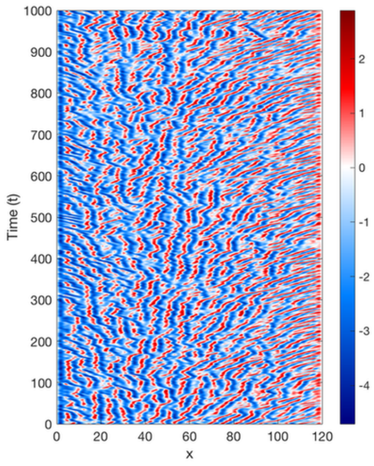
Figures 6.15(f)-6.15(h) show a HROM for $n_i = 380$, which was found to be the minimum number of sample nodes required for $n_r = 175$ case. The coherent structures once again for these three trajectories are unique and chaotic. Figure 6.15(h) once again shows the HROM for $n_{RJ} = 180$, which produces similar trajectories to Figure 6.15(e).

Figure 6.15(i) shows a HROM for the minimum possible number of sample nodes and reduced-order basis functions for the residual and the Jacobians. With only $n_i = 180$, which is a 62.5% reduction in spatial nodes to evaluate, the HROM trajectory in Figure 6.15(i) is still chaotic and exhibits similar coherent structures as its corresponding $n_i = 480$ case. Compared to the other HROM in Figure 6.15, Figure 6.15(i) produces a HROM that requires the least amount of computational cost to evaluate the residuals and Jacobians, making it an ideal case for adjoint calculations, assuming that its time-average output is relatively accurate compared to that of the ROM.

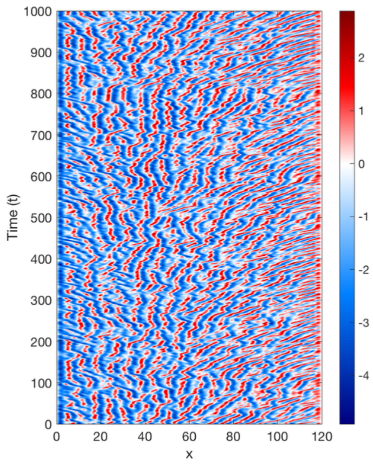
Figures 6.16 and 6.17 show the time-average outputs for different numbers of sample



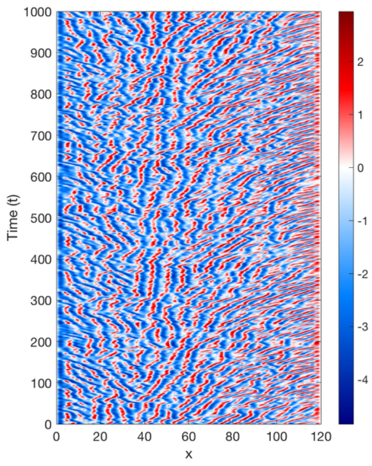
(a) primal



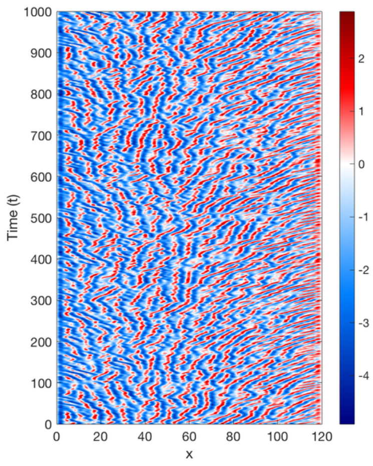
(b) ROM



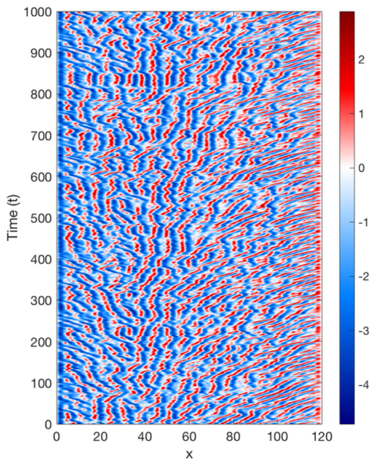
(c) $n_i = 480,\ n_{RJ} = 480$



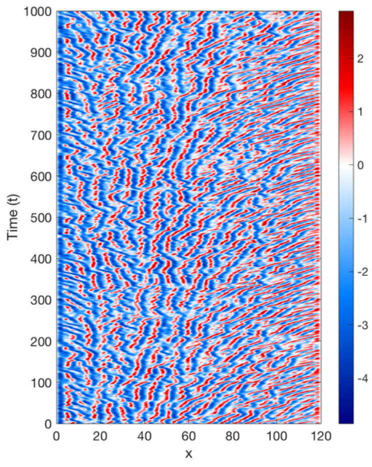
(d) $n_i = 480,\ n_{RJ} = 320$



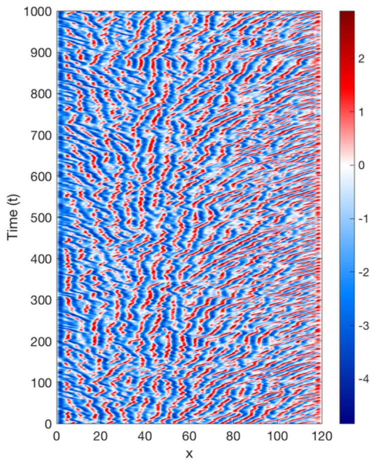
(e) $n_i = 480,\ n_{RJ} = 180$



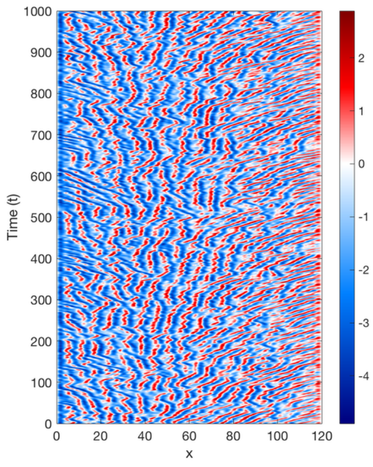
(f) $n_i = 380,\ n_{RJ} = 360$



(g) $n_i = 380,\ n_{RJ} = 260$

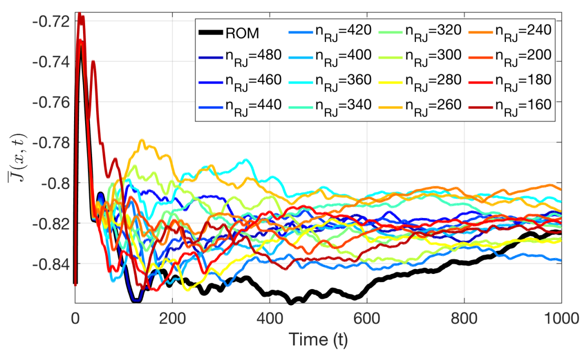


(h) $n_i = 380,\ n_{RJ} = 180$

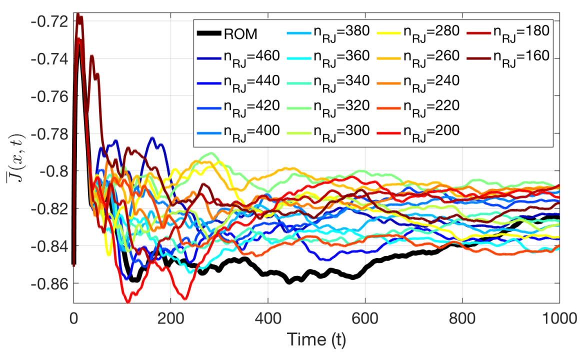


(i) $n_i = 180,\ n_{RJ} = 160$

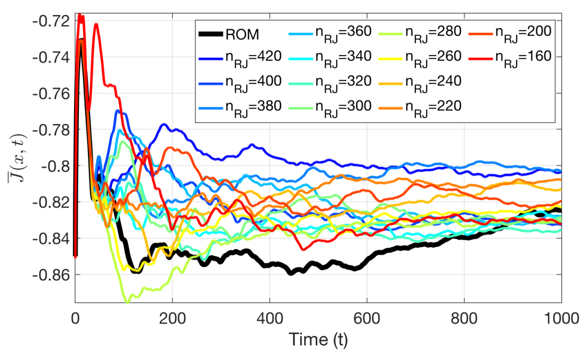
Figure 6.15: KS: Primal, ROM, and selected HROM trajectories for $n_r = 50$.



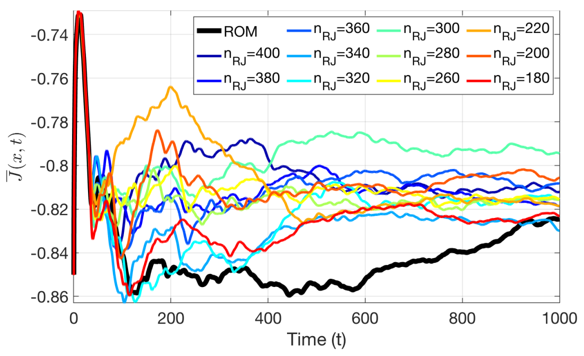
(a) $n_i = 480$



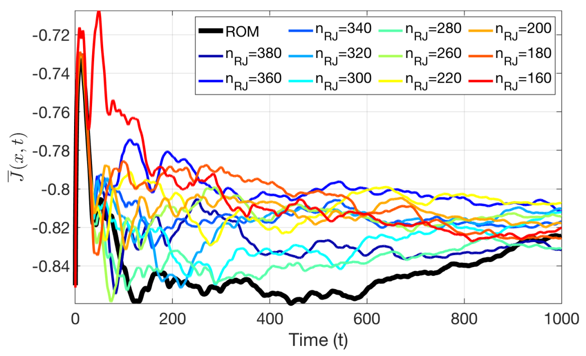
(b) $n_i = 460$



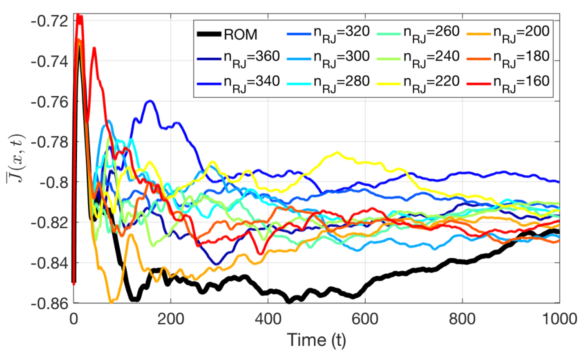
(c) $n_i = 440$



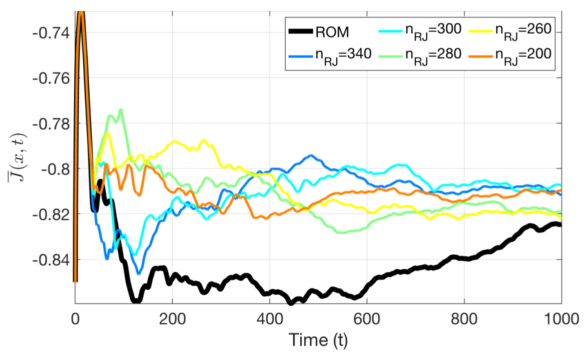
(d) $n_i = 420$



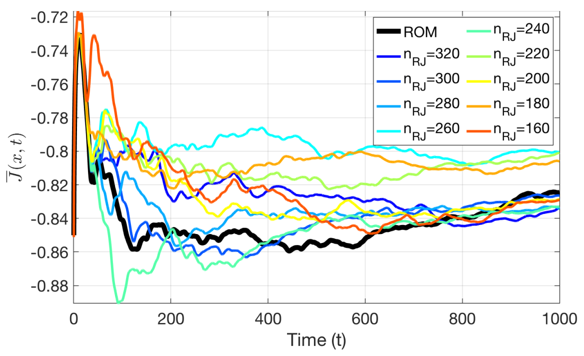
(e) $n_i = 400$



(f) $n_i = 380$

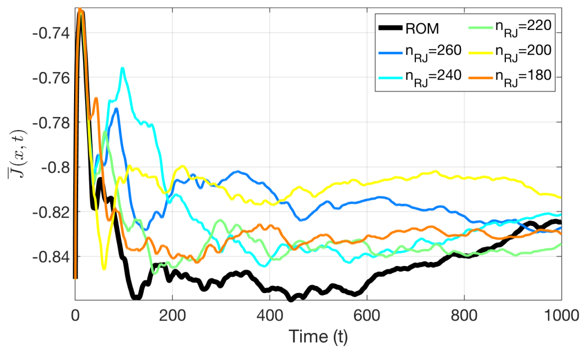


(g) $n_i = 360$

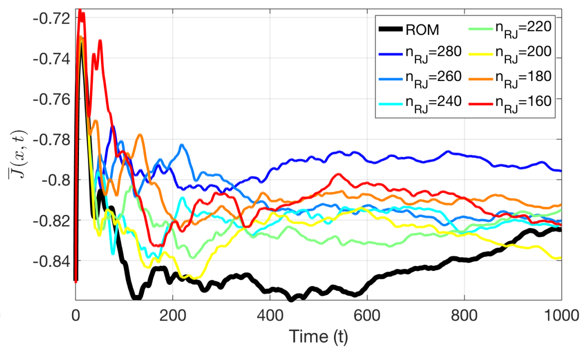


(h) $n_i = 340$

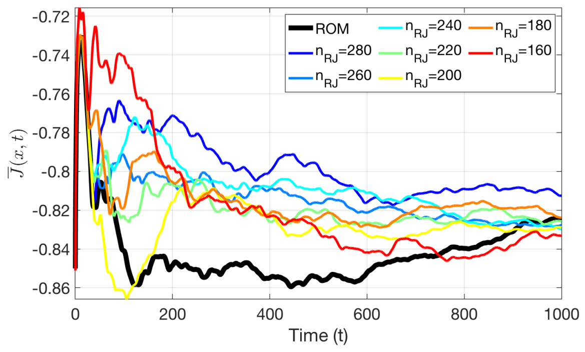
Figure 6.16: KS: Time-average outputs, $\overline{J}$, for $n_r = 50$ and $n_i \in [480, 340]$.



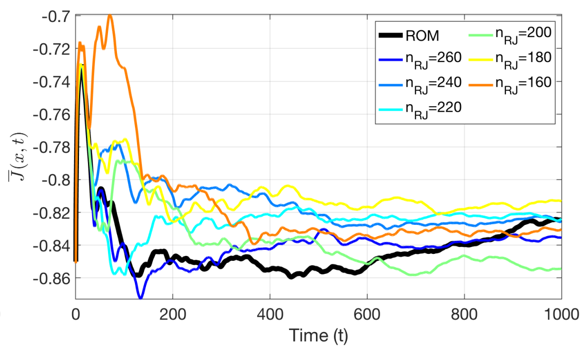
(a) $n_i = 320$



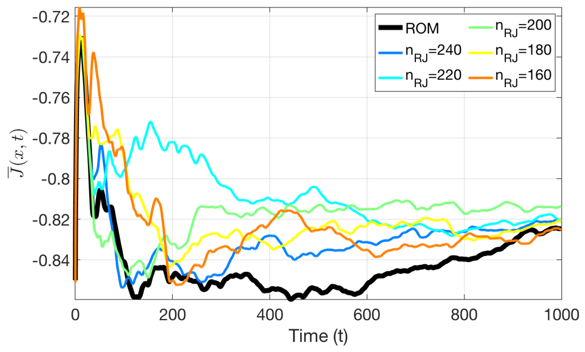
(b) $n_i = 300$



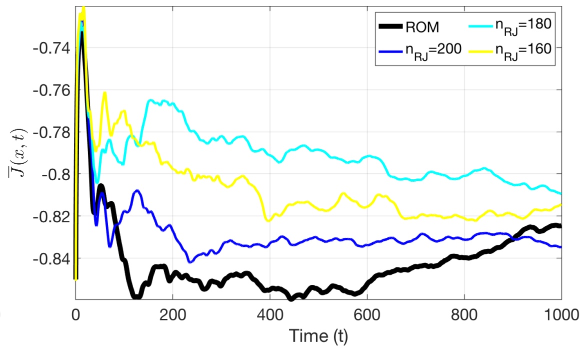
(c) $n_i = 280$



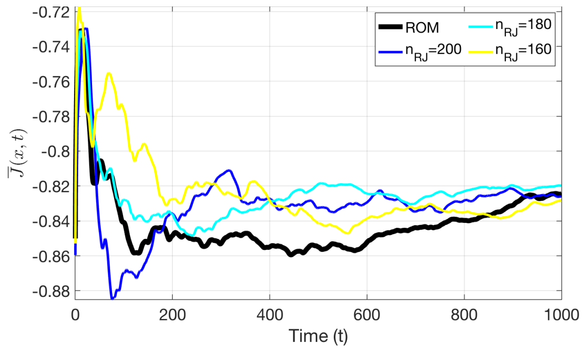
(d) $n_i = 260$



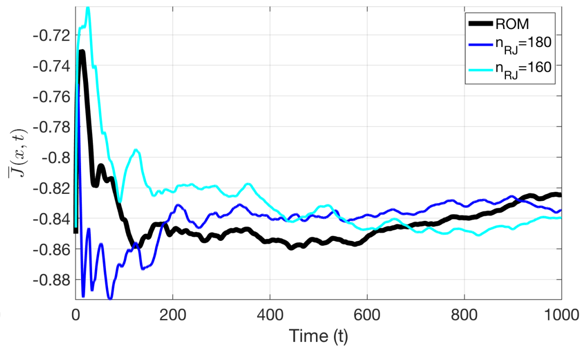
(e) $n_i = 240$



(f) $n_i = 220$



(g) $n_i = 200$



(h) $n_i = 180$

Figure 6.17: KS: Time-average outputs, $\overline{J}$, for $n_r = 50$ and $n_i \in [320, 180]$.

nodes, $n_i \in [480, 180]$ and reduced basis functions for the residuals and Jacobians. Once again, the trajectories are found for n_i and n_{RJ} in intervals of 20. One difference to highlight between the $n_r = 175$ and $n_r = 50$ test case is that the number of possible HROMs is greater for $n_r = 175$. For $n_r = 50$, the number of sample nodes n_i can be as low as $n_i = 180$ meaning that for the $n_i = 480$ case, twice as many HROMs can be found.

Figures 6.17(g) and 6.17(h) show the minimum case of $n_i = 200$ and $n_i = 180$. For a small number of sample nodes, the time-average output still emulates the output of the ROM case. For a corresponding smaller number of reduced-order basis functions for the residuals and Jacobians, the time-average output agrees well at $T = 1000$. It is important to note that as the number of sample nodes decreases, the number of possible HROMs with the corresponding reduced basis functions for residuals and Jacobians decreases as well such that by the time $n_i = 180$, only three HROMs converge for this setup, when $n_{RJ} = 180$ and $n_{RJ} = 160$ in Figure 6.17(h). For all sample nodes in Figures 6.16 and 6.17, the HROMs at $T = 1000$ are within 2.4% compared to that of the ROM.

Overall, these results show that the higher the number of basis functions, n_r, the higher the minimum number of sample nodes and reduced basis functions for the residuals and Jacobians is needed, $n_{RJ} \propto n_r$. Higher n_r will produce an overall more computationally expensive HROM. Note that based on the results for HROM, the minimum number of sample nodes needs to be at least half of the total number of spatial nodes for $n_r < 175$ for KS. If the number of sample nodes is approximately less than half of the number of total spatial nodes, GNAT will either not converge or the HROM will be unresolved. However, this guideline on how many sample nodes should be used is not necessarily true for other governing equations. When analyzing the time-average output $\overline{J}$ of the HROMs compared to the corresponding ROM, it is found that the HROM is able to emulate the ROM's time-average output even though the trajectories themselves may diverge away from the ROM completely. To minimize the computational cost associated with finding a HROM, the smallest possible n_i and n_{RJ} are ideal. However, this isn't necessarily true for n_r. Hence, it is overall not difficult to find a HROM that reduces the overall size of the system. n_r has a greater impact on the accuracy than n_i and n_{RJ} of the HROM compared to the primal solution. The ROM for $n_r = 175$ is more accurate than the $n_r = 50$ case compared to the primal solution; however, significant cost savings can be found by using $n_r = 50$. If some accuracy can be sacrificed, then the HROM corresponding to $n_r = 50$, $n_i = 180$, and $n_{RJ} = 160$ for $T = 1000$ for example would be an acceptable model to use to reduce the overall cost of LSS. The cost savings of using a HROM is even more apparent for a larger system such as the Navier-Stokes equations.

6.6 Summary

Chapter V introduced the LSS method, an alternative algorithm for the calculation of chaotic adjoints. Results from Chapter V for the Lorenz attractor and KS showed that the LSS method can produce accurate adjoints for output-based error estimation. However, the LSS method solves a minimization problem that requires a linear solver such as GMRES to solve a LSS system of size $N(2n_{seg} - 1)$, which is expensive for N or n_r. This cost is tractable for a simple system such as the Lorenz attractor, but it is quite expensive for a large system, such as the 2D Navier-Stokes equations.

Due to the expensive nature of LSS, model reduction was introduced to reduce the overall size of the system from N to n_r, where n_r is the number of basis functions of the state. The idea is that by reducing the overall size of the system to n_r without sacrificing too much on the overall accuracy of the model, the reduced system can be used instead of the full-order model in the LSS method, such that the LSS method solves a linear system of size $n_r(2n_{seg} - 1)$ instead of $N(2n_{seg} - 1)$. In this chapter, several model reduction techniques were introduced for this purpose.

The technique chosen for the model reduction of nonlinear system was the Least-Squares Petrov-Galerkin (LSPG) technique which is a modified version of the linear model reduction technique. By looking at the affine subspace, the solution can be reduced to size n_r via a set of basis functions in matrix form, $\boldsymbol{\Phi}$. Model reduction in general utilizes the POD method to solve for $\boldsymbol{\Phi}$ given a state snapshot matrix and looks to the left nullspace of the system in order to reduce the set of residual equations via orthogonality and a projection onto a test space, $\boldsymbol{\varphi}$. LSPG is unique in that the test functions are set to the Jacobian multiplied by the reduced-order basis function matrix, $\boldsymbol{J\Phi}$. For nonlinear cases, the Petrov-Galerkin projection, $\boldsymbol{\varphi} = \boldsymbol{J\Phi}$, is used instead of the Galerkin projection, $\boldsymbol{\varphi} = \boldsymbol{\Phi}$ to improve stability. In addition, setting the test function to $\boldsymbol{\varphi} = \boldsymbol{J\Phi}$ captures information on the nonlinearity of the system since for each Gauss-Newton iteration used to solve the reduced system in the minimization problem, $\boldsymbol{J\Phi}$ changes.

Implementing the LSPG technique produces a ROM. It was shown through the ROM of KS that the accuracy of the ROM can be assessed by looking at the time-average outputs $\overline{J}$, which showed that the accuracy of $\overline{J}$ is highly dependent on n_r. Higher n_r gives a more accurate ROM, but will produce an overall more expensive system to solve. Lower n_r produces a less accurate ROM, but will produce an overall less expensive system to solve. However, there is a minimum number of n_r that is required before the ROM fails and instead produces a non-chaotic periodic system.

ROM can be used with LSS directly; however, the ROM is not truly independent of the full-order model size, N. LSS requires the calculation of residuals and Jacobians which still depend on N. In order to further reduce the computational costs associated with the reduced model, the GNAT method was introduced which uses the greedy algorithm to select the n_i nodes at which to calculate the residuals and the Jacobians. As a result of the GNAT method, the residuals and the Jacobians are functions of n_i, n_j, n_e nodes, whose relationship is dependent on the choice of discretization for the primal solution. The gappy algorithm requires the creation of the snapshot matrix for the residuals and the Jacobians and then performs POD for both to produce $\mathbf{\Phi}_R$ and $\mathbf{\Phi}_J$, which have n_{RJ} basis functions. The combined GNAT method produces the HROM method which is a more efficient model to use than ROM for LSS.

Given that $n_i \geq n_{RJ}$ is required to produce unique approximated residuals and Jacobians, the GNAT method was shown to be able to produce many HROMs for KS. As the number of sample nodes decreased, the number of possible basis functions, n_{RJ}, decreased as well. It was shown that given n_i and n_{RJ}, as long as the HROM converges, the HROM is accurate and emulates the ROM of the chaotic system. Note that from the results, higher n_r requires more n_{RJ} but is again limited by the number of sample nodes. Thus, n_{RJ} is proportionally related to n_r, $n_{RJ} \propto n_r$. For comparison to the full-order solution, the accuracy of HROM is more strongly affected by the number of basis functions of the states n_r, which was seen with the creation of the ROM as well. Thus the minimum n_i and n_{RJ} is ideal for a given n_r.

The next chapter will go over how to use the HROM with LSS and show with results that using HROM instead of the full-order model can reduce the overall computational cost of LSS, making the GNAT method useful for the modified reduced LSS approach.

CHAPTER VII

Error Estimation Using Hyper-Reduced Order Modeling-Least Squares Shadowing

In this chapter, the LSS method, the model reduction techniques, and the output-based error estimation method will be combined to calculate usable adjoints in order to estimate the effect of discretization errors on the output. This combined method will be referred to as the HROM-LSS method. An overview of how these methods are used in relationship to each other is shown in a flowchart in Figure 7.1. This new procedure can be compared with the original LSS method in Figure 5.1. In Chapter V, the Least Squares Shadowing (LSS) method was introduced as an alternative way to calculate adjoints for chaotic flows. Instead of using the traditional adjoint method from Chapter III, the LSS method is used to find the shadow trajectory that is designed to stay close to the reference trajectory. The LSS method finds this new trajectory by relaxing the initial conditions and solving an optimization problem. The results of LSS for the Lorenz attractor and the KS equation show promise. However, despite the level of accuracy LSS is able to produce, the LSS optimization routine is expensive and requires up to $N(2n_{seg} - 1)$ iterations of the GMRES linear solver when using a checkpoint design to find the tangent and adjoint values for each of the time windows. This makes the LSS method a costly method to use for larger problems such as the Navier-Stokes equations, compared to the traditional adjoint method.

To remedy the high cost associated with LSS, model reduction techniques were introduced in order to decrease the overall size of the primal problem. The reduced primal solution was found by looking to the associated affine subspace of the system. There, the basis functions, were found to map the full-order solution to the reduced solution. It was shown in Chapter VI, that model reduction principles can be applied to chaotic systems and can produce reduced-order models whose time-average values converge to the time-average values of the primal solution, as long as enough basis functions are used.

Despite the success of ROM, the residual and the Jacobian for each Gauss-Newton iteration of the model reduction procedure is still dependent on N spatial nodes, which

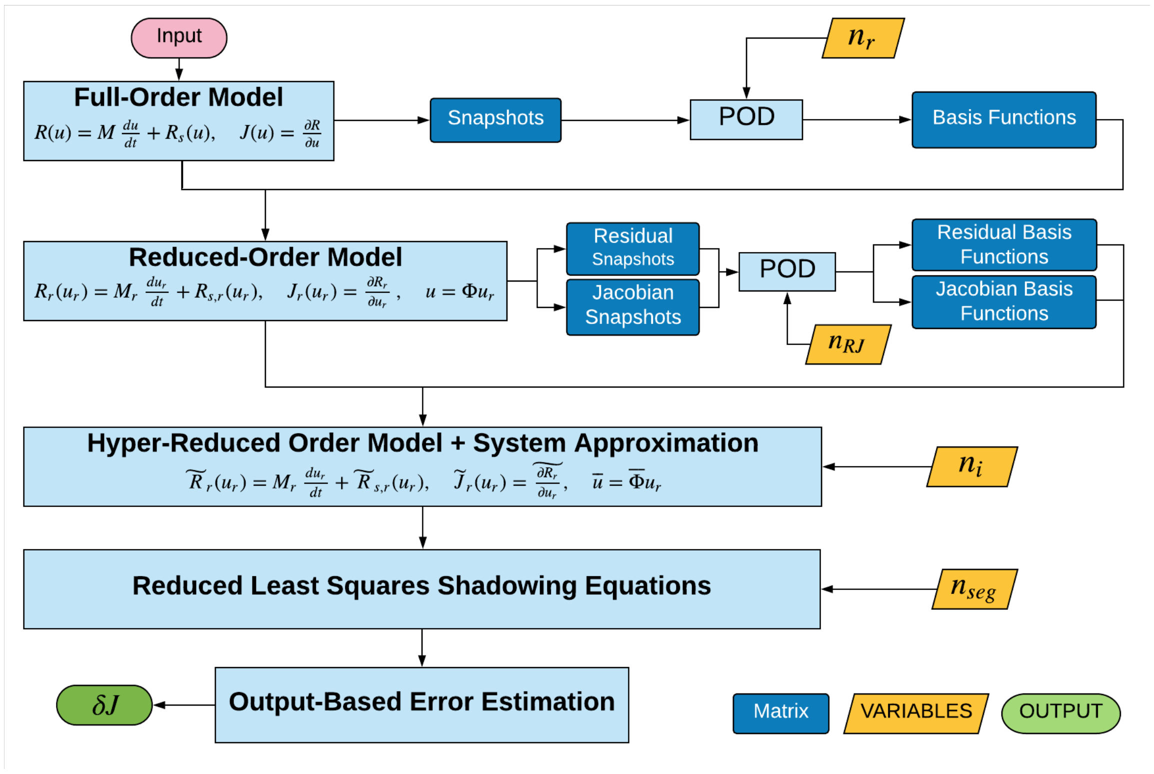

Input
Full-Order Model
$R(u) = M\frac{du}{dt} + R_s(u), \quad J(u) = \frac{\partial R}{\partial u}$
Snapshots
POD
n_r
Basis Functions
Reduced-Order Model
$R_r(u_r) = M_r\frac{du_r}{dt} + R_{s,r}(u_r), \quad J_r(u_r) = \frac{\partial R_r}{\partial u_r}, \quad u = \Phi u_r$
Residual Snapshots
Jacobian Snapshots
POD
n_{RJ}
Residual Basis Functions
Jacobian Basis Functions
Hyper-Reduced Order Model + System Approximation
$\widetilde{R}_r(u_r) = M_r\frac{du_r}{dt} + \widetilde{R}_{s,r}(u_r), \quad \widetilde{J}_r(u_r) = \widetilde{\frac{\partial R_r}{\partial u_r}}, \quad \overline{u} = \overline{\Phi}u_r$
n_i
Reduced Least Squares Shadowing Equations
n_{seg}
Output-Based Error Estimation
δJ
Matrix
VARIABLES
OUTPUT

Figure 7.1: HROM-LSS flowchart: discretization error estimates are found by first solving the primal solution in both the coarse and fine space. The ROM is found from the snapshots of the fine-space solution. POD is performed to find the basis functions. The basis functions are used to find the reduced-order model. From the reduced-order model, the residual and Jacobian snapshots are found. POD is performed as well to find the basis functions for the residuals and the Jacobians. The new basis functions are used to find the HROM, which is then used in the reduced form of the LSS equations. The adjoints are then used to find the error estimates via the adjoint weighted-residual in the output-based error estimation method. The four main tuning parameters of this combined method are the number of basis functions n_r, the number of basis functions for the residuals and the Jacobians, n_{RJ}, the number of sample nodes, n_i, and finally the number of time windows, n_{seg}.

can be cumbersome for large chaotic problems. To truly reduce the computational cost associated with the primal solution, Chapter VI introduces a technique to reduce the number of evaluations of the residual and Jacobian from N nodes to n_i sample nodes.

By reducing the size of the primal solution to n_r and by reducing the number of nodes needed to calculate the residuals and Jacobians to n_i via the GNAT method, the number of iterations required to find the shadow trajectory via LSS is $n_r(2n_{neg} - 1)$ where $n_r \ll N$. Due to only n_i nodes being calculated to find the residuals and the Jacobians, the time it takes to perform an individual GMRES iteration in LSS is reduced as well.

In this chapter, it will be shown how the LSS equations will be converted into its reduced form using the same projection-based and hyper-model reduction techniques from Chapter VI such that it can accommodate the newly developed HROM solution. It will be shown how the gappy algorithm is used to reduce the LSS equations as well. Once the reduced LSS equations are found, the shadow trajectory based on the hyper-reduced order model will be calculated. Next, the modified output-based error estimation procedure that is used to calculate the effect of the discretization error estimate on the output will be shown. Finally, results for this combined HROM-LSS method will be shown for KS and the Navier-Stokes equations.

7.1 Reduced Form of the Least Squares Shadowing Equations

In order to use the HROM found from GNAT (Chapter VI), the original LSS equations from Chapter V need to be converted into corresponding reduced form. In Chapter VI, the full-order solution is related to its orthogonal left nullspace via test functions, $\boldsymbol{\varphi}$. The same principles are applied to the LSS adjoint equations and its initial and terminal conditions to find the reduced tangent ($\boldsymbol{\psi}_2$) and reduced Lagrange ($\boldsymbol{\psi}_1$) equations. First, one needs to apply projection based model reduction techniques to the original governing equation in order to define the initial and terminal conditions for the reduced LSS adjoint equations. The original governing equations multiplied by the transpose of the test functions give,

$$\boldsymbol{\varphi}^T \left[\frac{d(\boldsymbol{\Phi} \boldsymbol{u}_r)}{dt} = \boldsymbol{f}(\boldsymbol{\Phi} \boldsymbol{u}_r) \right], \tag{7.1}$$

where its reduced form becomes

$$\frac{d\boldsymbol{u}_r}{dt} = \underbrace{\left[\boldsymbol{\varphi}^T \boldsymbol{\Phi}\right]^{-1} \boldsymbol{\varphi}^T \boldsymbol{f}(\boldsymbol{\Phi}\boldsymbol{u}_r)}_{\boldsymbol{f}_r}. \tag{7.2}$$

Next, the same procedure is used to find the reduced form of the LSS adjoint equations; however it is not obvious as to how the full adjoints is related to the reduced adjoints. Given the linear equation from Chapter III,

$$\boldsymbol{L}\boldsymbol{u} = \boldsymbol{f}, \tag{7.3}$$

it can be shown that after substituting Eqn. 6.3 and multiplying the linear system by test function $\boldsymbol{\varphi}$, the linear equation becomes

$$\boldsymbol{\varphi}^T \left[\boldsymbol{L}\boldsymbol{\Phi}\boldsymbol{u}_r = \boldsymbol{f}\right]. \tag{7.4}$$

Formulating the Lagrangian equation gives,

$$\mathcal{L}_r = \langle \boldsymbol{g}, \boldsymbol{\Phi}\boldsymbol{u}_r\rangle - \langle \boldsymbol{\psi}_r, \boldsymbol{\varphi}^T \boldsymbol{L}\left(\boldsymbol{\Phi}\boldsymbol{u}_r - \boldsymbol{f}\right)\rangle, \tag{7.5}$$

where $\boldsymbol{g}$ is a function used to calculate the linearized output, $\langle \boldsymbol{g}, \boldsymbol{\Phi}\boldsymbol{u}_r\rangle$. Taking variations of $\boldsymbol{u}_r$, $\boldsymbol{u}_r \to \boldsymbol{u}_r + \delta\boldsymbol{u}_r$, requiring the Lagrangian to be stationary with respect to permissible δu_r, and integrating by parts gives the reduced adjoint equations,

$$\boldsymbol{\varphi}^T \left[\boldsymbol{L}^* \boldsymbol{\Phi}\boldsymbol{\psi}_r = \boldsymbol{g}\right], \tag{7.6}$$

where L^* is adjoint operator defined after integrating the Lagrangian equation by parts. the When comparing the dual reduced adjoint with the full adjoint equation from Chapter III it can be seen that that the following is true,

$$\boldsymbol{\psi} = \boldsymbol{\Phi}\boldsymbol{\psi}_r. \tag{7.7}$$

where $\boldsymbol{\Phi}$ is the reduced order basis matrix from the reduced state. This approximation of the full adjoint solution as a function of its reduced adjoint solution can be used for the both the tangent and Lagrange solutions in the LSS adjoint equations. By substituting Eqn. 7.7 into the LSS adjoint equations and performing a projection using a test function, $\boldsymbol{\varphi}$, one can find that

$$\boldsymbol{\varphi}^T \left[\frac{d\boldsymbol{\Phi}\boldsymbol{\psi}_{2,r}}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\boldsymbol{\Phi}u_r)\right)\boldsymbol{\Phi}\boldsymbol{\psi}_{2,r}\right], \tag{7.8}$$

$$\boldsymbol{\varphi}^T \left[\frac{d\boldsymbol{\Phi}\boldsymbol{\psi}_{1,r}}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \left(\boldsymbol{\Phi} u_r \right) \right)^T \boldsymbol{\Phi}\boldsymbol{\psi}_{1,r} - \boldsymbol{\Phi}\boldsymbol{\psi}_{2,r} - \frac{1}{T} \frac{\partial J}{\partial \boldsymbol{u}}^T \right], \tag{7.9}$$

where Eqn. 6.3 is used to substitute for the full-order states. Using the same linear model reduction technique from Chapter VI, the LSS adjoint equations can be rewritten as

$$\frac{d\boldsymbol{\psi}_{2,r}}{dt} = \underbrace{\left[\boldsymbol{\varphi}^T \boldsymbol{\Phi} \right]^{-1} \boldsymbol{\varphi}^T \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \left(\boldsymbol{\Phi} u_r \right) \right) \boldsymbol{\Phi}}_{\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \right)_r} \boldsymbol{\psi}_{2,r}, \tag{7.10}$$

$$\begin{aligned} \frac{d\boldsymbol{\psi}_{1,r}}{dt} = &- \underbrace{\left[\boldsymbol{\varphi}^T \boldsymbol{\Phi} \right]^{-1} \boldsymbol{\varphi}^T \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \left(\boldsymbol{\Phi} u_r \right) \right)^T \boldsymbol{\Phi}}_{\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}^T \right)_r} \boldsymbol{\psi}_{1,r} - \underbrace{\left[\boldsymbol{\varphi}^T \boldsymbol{\Phi} \right]^{-1} \boldsymbol{\varphi}^T \boldsymbol{\Phi}}_{\boldsymbol{C}_r} \boldsymbol{\psi}_{2,r} \\ &- \underbrace{\left[\boldsymbol{\varphi}^T \boldsymbol{\Phi} \right]^{-1} \boldsymbol{\varphi}^T \frac{1}{T} \frac{\partial J}{\partial \boldsymbol{u}}^T}_{\left(\frac{1}{T} \frac{\partial J}{\partial \boldsymbol{u}}^T \right)_r}, \end{aligned} \tag{7.11}$$

where $\boldsymbol{\psi}_{2,r}$ refers to the reduced tangent solution and $\boldsymbol{\psi}_{1,r}$ refers to the reduced adjoint solution. The goal is to write the tangent equation and Lagrange equation in terms of known quantities. Like the full-order residual and Jacobian, calculating $\boldsymbol{f}(\boldsymbol{\Phi}\boldsymbol{u}_r)$ is expensive since it is dependent on N spatial nodes. In order to reduce the costs of calculating $\boldsymbol{f}_r$ and $\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}$, the GNAT method is used. First, only n_i nodes are evaluated from the greedy algorithm. The reduced terms are then approximated using the gappy reconstruction procedure from GNAT to find,

$$\widetilde{\boldsymbol{f}} \approx \boldsymbol{\Phi}_f \widehat{\boldsymbol{\Phi}}_f^+ \widehat{\boldsymbol{f}}, \tag{7.12}$$

where $\widehat{(\cdot)}$ refers to the n_i rows retained from the greedy algorithm seen in Eqn. 6.38. $\widehat{\boldsymbol{f}}$ is calculated from the known spatial residual and the inverse of the mass matrix used in the DG discretization,

$$\widehat{\boldsymbol{f}} = -\widehat{\boldsymbol{M}^{-1}\boldsymbol{R}_s}. \tag{7.13}$$

To calculate the approximation for $\boldsymbol{f}$, the reduced-order basis matrix, $\boldsymbol{\Phi}_f$, is required; however, instead of creating a snapshot matrix for $\boldsymbol{f}$ and then performing POD on it, the reduced-order basis matrix for $\boldsymbol{f}$ can be said to be approximately the same as $\boldsymbol{\Phi}_R$, the reduced-order basis matrix for the residual. The main difference between the two would be the mass matrix, but nonlinearity of the problem will still be preserved, which is what is most important. For this dissertation the following relationship will be assumed,

$$\boldsymbol{\Phi}_f \approx \boldsymbol{\Phi}_R \tag{7.14}$$

and will be shown to provide accurate results in the error estimation process. Substituting the approximation of the full-order $\boldsymbol{f}$ found in Eqn. 7.12 into $\boldsymbol{f}_r$ from Eqn. 7.2 gives the final form for reduced $\boldsymbol{f}$,

$$\boldsymbol{f}_r = \boldsymbol{A}\widehat{\boldsymbol{M}^{-1}\boldsymbol{R}}_s, \quad \boldsymbol{A} = -\left[\boldsymbol{\varphi}^T\boldsymbol{\Phi}\right]^{-1}\boldsymbol{\varphi}^T\boldsymbol{\Phi}_R\widehat{\boldsymbol{\Phi}}_R^{+}. \tag{7.15}$$

where $\boldsymbol{A}$ is an offline matrix, calculated before the LSS routine takes place. Note that $(\cdot)^{\dagger}$ is the Moore-Penrose pseudo inverse and that the approximation for $\boldsymbol{f}_r$ is mainly used to calculate the initial and terminal conditions of LSS. Next, the derivative of $\boldsymbol{f}$ need to defined for both the tangent and Lagrange equation. From Eqn. 7.11, the reduced derivative of $\boldsymbol{f}$ is defined as

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)_r = \left[\boldsymbol{\varphi}^T\boldsymbol{\Phi}\right]^{-1}\boldsymbol{\varphi}^T\frac{\partial \boldsymbol{f}\left(\Phi \boldsymbol{u}_r\right)}{\partial \boldsymbol{u}}\boldsymbol{\Phi}. \tag{7.16}$$

As with the $\boldsymbol{f}_r$ case, the GNAT procedure is used to find $\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)_r$ using the greedy and gappy algorithm,

$$\widetilde{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}} \approx \boldsymbol{\Phi}_{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}}\widehat{\boldsymbol{\Phi}}^{+}_{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}}\widehat{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}}, \tag{7.17}$$

where

$$\widehat{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}} = -\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}}_s, \quad \boldsymbol{J}_s = \frac{\partial \boldsymbol{R}_s}{\partial \boldsymbol{u}}. \tag{7.18}$$

Since the GNAT procedure is used, $\overline{\boldsymbol{\Phi}}$ is used instead of $\boldsymbol{\Phi}$ in the gappy approximations. To calculate the approximation for $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$, the reduced-order basis matrix, $\boldsymbol{\Phi}_{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}}$ is required; however, again, instead of creating a snapshot matrix for $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ and then performing POD on it, the reduced-order basis matrix for $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ can be said to be approximately the same as $\boldsymbol{\Phi}_J$, the reduced-order basis matrix for the Jacobian. Like the reduced-order basis function matrix for $\boldsymbol{f}$, the main difference between the two would be the mass matrix, but nonlinearity of the problem will still be preserved. As a result,

$$\widehat{\boldsymbol{\Phi}}_{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}} = \boldsymbol{\Phi}_J. \tag{7.19}$$

Substituting Eqn. 7.18 into Eqn. 7.16 gives the final reduced form of $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$,

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)_r = \boldsymbol{B}\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}}_s\overline{\boldsymbol{\Phi}} \quad \boldsymbol{B} = \left[\boldsymbol{\varphi}^T\boldsymbol{\Phi}\right]^{-1}\boldsymbol{\varphi}^T\boldsymbol{\Phi}_J\widehat{\boldsymbol{\Phi}}_J^{+}, \tag{7.20}$$

where $\boldsymbol{B}$ is the second offline matrix that needs to be calculated before the LSS routine takes place. The same procedure can be used to define $\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}^T\right)_r$,

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}^T\right)_r = \boldsymbol{B}\left(\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}_s}\right)^T \overline{\boldsymbol{\Phi}} \quad \boldsymbol{B} = \left[\boldsymbol{\varphi}^T\boldsymbol{\Phi}\right]^{-1}\boldsymbol{\varphi}^T\boldsymbol{\Phi}_J\widehat{\boldsymbol{\Phi}}_J^+, \tag{7.21}$$

After finding approximations for the reduced $\boldsymbol{f}$ and its derivative, a choice needs to be made about what type of projection to use for $\boldsymbol{\varphi}$. For model reduction of nonlinear systems in Chapter VI, it was shown that the Least-Squares Petrov-Galerkin projection was the best choice due to its ability to retain information of the nonlinearity of the problem. However, due to the fact that the LSS adjoints equations are linear, the Galerkin projection is adequate and used,

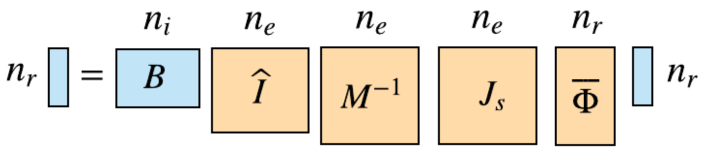
$$\boldsymbol{\varphi} = \boldsymbol{\Phi}. \tag{7.22}$$

Since $\boldsymbol{\Phi}$ is orthogonal, many terms simplify and $\boldsymbol{C}_r$ becomes an identity matrix. The reduced LSS equations have the final form,

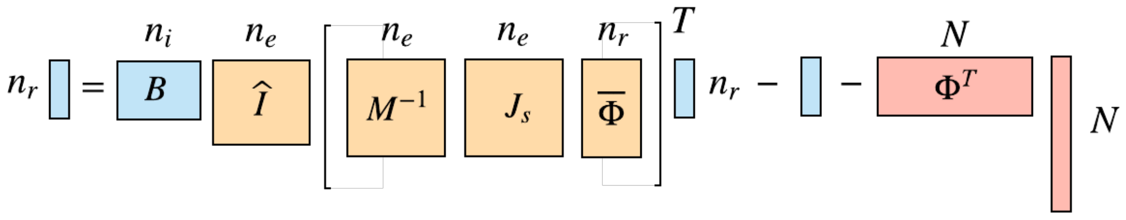
$$\frac{d\boldsymbol{\psi}_{2,r}}{dt} = \left[\boldsymbol{B}\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}_s}\overline{\boldsymbol{\Phi}}\right]\boldsymbol{\psi}_{2,r}, \tag{7.23}$$

$$\frac{d\boldsymbol{\psi}_{1,r}}{dt} = -\left[\boldsymbol{B}\left(\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}_s}\right)^T\overline{\boldsymbol{\Phi}}\right]\boldsymbol{\psi}_{1,r} - \boldsymbol{\psi}_{2,r} - \frac{1}{T}\boldsymbol{\Phi}^T\frac{\partial J}{\partial \boldsymbol{u}}^T. \tag{7.24}$$

The reduced LSS equations' relative sizes and structure can be seen in Figure 7.2. As with



(a) Structure for the reduced tangent equation for ψ_2.



(b) Structure for the reduced Lagrange equation for ψ_1.

Figure 7.2: Vectors and matrices for the reduced Least Squares Shadowing equations that show how the reduced states and full-order vectors relate to each other.

the full-order LSS adjoint equations in Chapter V, the reduced LSS adjoint equations need to take into account the dilation term as well, by looking at the projections of the tangent and Lagrange adjoint states. This projection process also ensures that the adjoints are adjoint consistent.

The final form of the reduced LSS equations for the checkpoint design with the dilation term taken into account is

$$\frac{d\boldsymbol{\psi}_{2,r}}{dt} = \left[\boldsymbol{B}\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}_s\overline{\boldsymbol{\Phi}}}\right]\boldsymbol{\psi}_{2,r}, \tag{7.25}$$

$$\frac{d\boldsymbol{\psi}_{1,r}}{dt} = -\left[\boldsymbol{B}\left(\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}_s}\right)^T\overline{\boldsymbol{\Phi}}\right]\boldsymbol{\psi}_{1,r} - P_t\boldsymbol{\psi}_{2,r} - \beta\frac{1}{T}\boldsymbol{\Phi}^T\frac{\partial J}{\partial \boldsymbol{u}}^T. \tag{7.26}$$

These reduced LSS equations are solved with the same iterative checkpoint process outlined in Chapter VI. A guess for the reduced tangent solution and reduced adjoint solution for each check point is made. Once again, GMRES is chosen as the iterative solver to find the desired reduced-adjoint solution.

7.2 Output-Based Error Estimation via the Adjoint-Weighted Residual

After solving the reduced-adjoint LSS equations via the iterative checkpoint method, output sensitives can be calculated from $\boldsymbol{\psi}_{1,r}$, since this is the adjoint that weights the residual term with $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mu}}$. Specifically,

$$\frac{\partial \bar{J}}{\partial \boldsymbol{\mu}} = \int_{T_0}^{T_f} \boldsymbol{\psi}_{1_r}^T \frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{\mu}} dt. \tag{7.27}$$

To estimate the output error using the adjoint from HROM-LSS, one can apply Eqn. 7.27 as in the LSS case, using a residual perturbation computed from two different discretization spaces: a coarse one with spatial order p_H, and a fine one with spatial order $p_h = p_H + 1$. The primal solution of order p_H is injected into the fine space. Given the perturbation in the residual, the error estimate is

$$\delta\bar{J} = -\int_{T_0}^{T_f} \boldsymbol{\psi}_{1_{r,h}}^T \boldsymbol{\Phi}^T \left[\frac{d\boldsymbol{u}_H}{dt} - \boldsymbol{f}(\boldsymbol{u}_H)\right]_h dt = -\int_{T_0}^{T_f} \boldsymbol{\psi}_{1_{r,h}}^T \boldsymbol{\Phi}^T \boldsymbol{M}^{-1}\boldsymbol{R}(\boldsymbol{u}_h^H), \tag{7.28}$$

where $\boldsymbol{\psi}_{1_{r,h}}$ is the vector of all of the $\boldsymbol{\psi}_h$ unknowns and $\boldsymbol{R}_h(\boldsymbol{u}_h^H)$ is the order p_h residual vector evaluated with the order p_H injected solution. To perform error estimation given

$\delta\boldsymbol{\psi}_{1_{r,h}}$,

$$\delta\bar{J} = -\int_{T_0}^{T_f} \delta\boldsymbol{\psi}_{1_{r,h}}^T \boldsymbol{\Phi}^T \left[\frac{d\boldsymbol{u}_H}{dt} - \boldsymbol{f}(\boldsymbol{u}_H)\right]_h dt = -\int_{T_0}^{T_f} \delta\boldsymbol{\psi}_{1_{r,h}}^T \boldsymbol{\Phi}^T \boldsymbol{M}^{-1} \boldsymbol{R}(\boldsymbol{u}_h^H), \quad (7.29)$$

where the reduced $\delta\boldsymbol{\psi}_{1_{r,h}}$ is defined from Chapter III as

$$\delta\boldsymbol{\psi}_{1_{r,h}} = \boldsymbol{\Phi}^T \left[\boldsymbol{I}_h - \boldsymbol{I}_h^H \boldsymbol{I}_H^h\right] \boldsymbol{\Phi}\boldsymbol{\psi}_{1_{r,h}}, \quad (7.30)$$

where $\boldsymbol{I}_h$ is the identity matrix of the fine space, $\boldsymbol{I}_h^H$ is the projection permutation matrix, and $\boldsymbol{I}_H^h$ is the injection permutation matrix. Eqn. 7.30 takes into account of the non-zero coarse adjoint. Results for the error estimate using $\boldsymbol{\psi}_{1_{r,h}}$, $\delta\boldsymbol{\psi}_{1_{r,h}}$, and $\delta\boldsymbol{\psi}_{1_{r,h}}^{H_1}$ will be shown in the next section.

7.3 Results for HROM-LSS for Kuramoto-Sivashinsky

In this section, several different results relating the four main parameters of HROM-LSS will be presented. For all the results presented for KS, the burn time is the same as in the LSS case, $t_{\text{burn}} = 500$. The spatial order for the coarse space is $p_H = 2$ and the spatial order for the fine space is $p_h = 3$. Each graph will have three different types of error estimates corresponding to different forms of the adjoints that are used, $\boldsymbol{\psi}_{1_{r,h}}$ shown in red, $\delta\boldsymbol{\psi}_{1_{r,h}}$ shown in turquoise, and $\delta\boldsymbol{\psi}_{1_{r,h}}^{H_1}$ shown in cyan. The actual error is shown in blue. The goal is find error estimates that are close to the mean error and that fall within $\pm$ one standard deviation of the actual error.

The first set of results consists of the error estimates versus the number of basis functions, n_r, of the states for time simulations, $T = [20, 120]$. Note that these results reflect the change in the number of basis functions for the residuals and the Jacobians, n_{RJ}, given the simulation time, T. The number of sample nodes is set to $n_i = 480$. These results are shown in Figure 7.3. The shaded blue region in Figure 7.3 represents the $\pm$ one standard deviation of the actual error, and the bold blue line refers to the mean actual area. The best parameters are chosen for each simulation time and are presented in Figure 7.4.

The second set of results will show how decreasing the number of sample nodes from $n_i = 480$ to $n_i = 200$ affects the error estimates overall. The results are shown in Figure 7.5. The primal solution, the HROM solution, and the corresponding adjoint are shown as well for one test case for each time simulation.

Lastly, the results of HROM-LSS are compared to that of LSS for KS. It is shown that HROM-LSS is a more efficient method that is able to retain/improve accuracy compared to

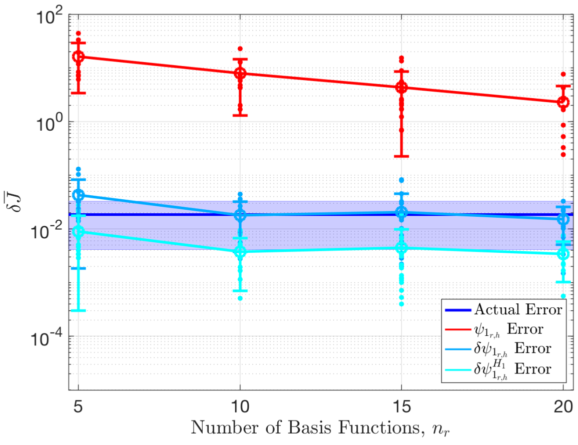
LSS.

7.3.1 HROM-LSS Error Estimate for $n_i = 480$ and Varied n_r

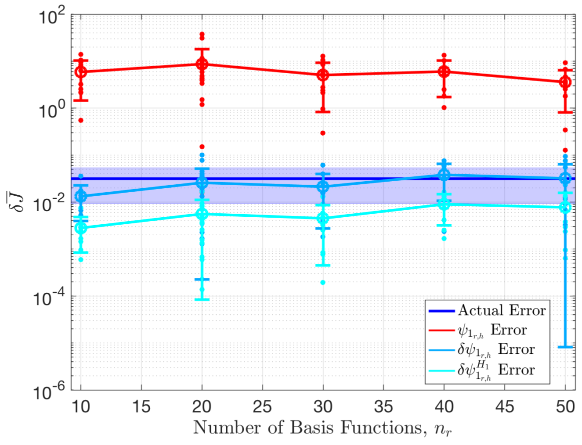
Figure 7.3 presents results for an ensemble of approximately ten cases with randomly chosen perturbed initial conditions for each unique n_r. The goal of these results is to understand the relationship between the output error estimate and the number of basis functions, n_r.

Figure 7.3(a) shows results for the simulation time of $T = 20$ for $n_r = 5, 10, 15, 20$. When the number of basis functions is set to $n_r = 5$, the number of basis functions for the residuals and Jacobians is set to $n_{RJ} = 50$. The full adjoint error estimate, $\boldsymbol{\psi}_{1_{r,h}}$, overestimates the actual error by approximately three orders of magnitude. When the influence of the coarse space adjoint is taken away by projecting the fine adjoint onto the coarse space, $\delta\boldsymbol{\psi}_{1_{r,h}}$, the error estimates improve significantly and produce results that are within an order magnitude of the actual error estimate. The error estimate that use the adjoint with H_1 projection,$\delta\boldsymbol{\psi}^{H_1}_{1_{r,h}}$ further decrease the error estimates and produce similar results as that of the original projected adjoint, $\delta\boldsymbol{\psi}_{1_{r,h}}$. Next, the number of basis functions, n_r is decreased to $n_r = 10$ to see if the error estimate improves. For $n_r = 10$, the number of basis functions for the residuals and the Jacobians is set to $n_{RJ} = 30$. As found in Chapter VI, the higher the number of basis functions, n_r, the more basis functions for the residuals and the Jacobians, n_{RJ} are needed. The full adjoint error estimation shown in red decreases and the mean error of the projected adjoint matches up well with the mean actual error. This trend is seen as well when n_r is increased to $n_r = 15$ and $n_r = 20$. The number of basis functions for the residuals and the Jacobians respectively are set to $n_{RJ} = 50$ and $n_{RJ} = 60$. The error estimates for the projected adjoint for $n_r = 15$ and $n_r = 20$ stay relatively constant while the full adjoint error estimate continues to decrease, which shows that if the number of basis functions, n_r increases, the full adjoint error estimate may match the actual error. However, increasing n_r and n_{RJ} leads to a more expensive model, and hence, based on this small test case, the best HROM to use to estimate the effect of discretization errors on the output for $T = 10$ is one that has parameters set to $n_r = 10$, $n_{RJ} = 30$. Note that the statistics become worse for the full adjoint with increasing n_r while the statistics are better when the error estimates are calculated with $\delta\boldsymbol{\psi}_{1_{r,h}}$.

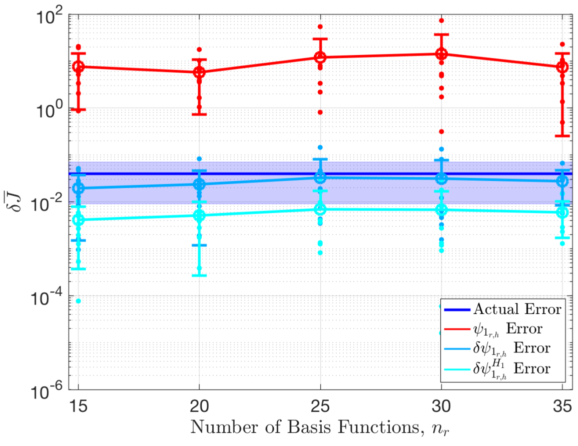
Figure 7.3(b) shows results for simulation time of $T = 40$ for $n_r = 10, 20, 30, 40, 50$. When the number of basis functions is set to $n_r = 10$, the number of basis functions for the residuals and Jacobians is set to $n_{RJ} = 60$. The error estimates for $n_r = 10$ when using the full adjoint, once again overestimate the actual error just under three orders of magnitude. However, the projected adjoint error estimates under predicts the actual errors. Again, the



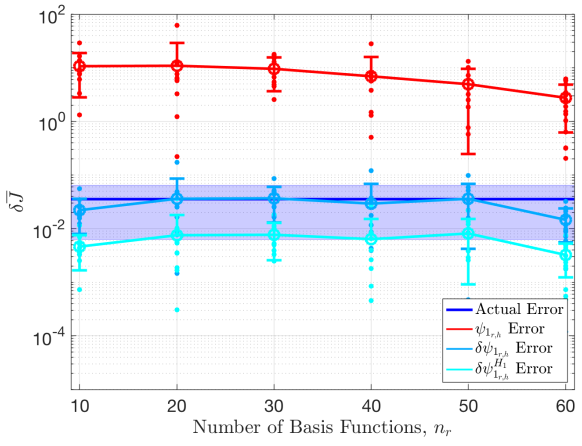
(a) $T = 20$



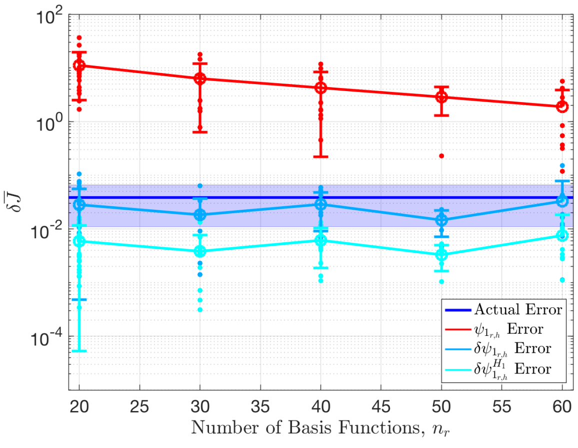
(b) $T = 40$



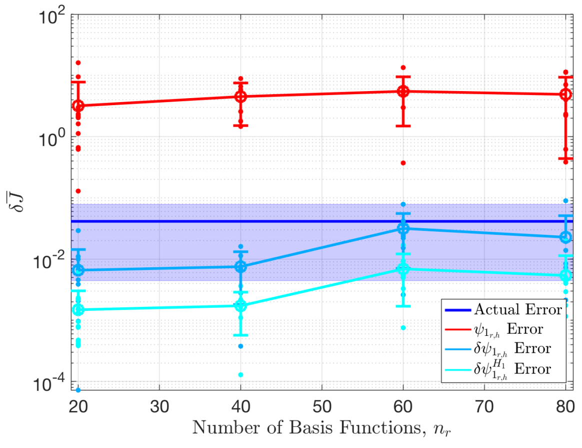
(c) $T = 60$



(d) $T = 80$



(e) $T = 100$



(f) $T = 120$

Figure 7.3: KS: relationship between error estimates and reduced basis functions n_r for time simulation $T = [20, 120]$.

goal is to see if increasing n_r will improve the error estimates. Next, the number of basis functions is set to $n_r = 20$, where the number of basis functions for the residuals and the Jacobians is set to $n_{8J} = 60$ as well. The full adjoint error estimate for $n_r = 20$ increases slightly, giving a worse error estimate; however, the projected adjoint error estimate for both types of projections improves. Next, results are shown for $n_r = 30, 40, 50$, where $n_{RJ} = 80, 85, 100$ respectively. By $n = 50$, the projected adjoint error estimate matches that of the mean actual error, showing that increasing n_r has an influential effect on the accuracy of the error estimates. The full adjoint error estimate improves as well with increasing n_r, but at a much slower rate. Based on this test case, the best set of parameters for $T = 40$ that can be used for error estimate is $n_r = 20$ and $n_{RJ} = 60$. Any more and the HROM becomes overly expensive to use. Note that there is no clear trend on how the statistics for $T = 40$ behave.

Figure 7.3(c) shows results for the simulation time of $T = 60$ for $n_r = 15, 20, 25, 30, 35$. When the number of basis functions is set to $n_r = 15$, the number of basis functions for the residuals and Jacobians is set to $n_{RJ} = 70$. Again, the full adjoint error estimate over estimates the actual error by three orders of magnitude. The error estimate with the projected adjoint underestimates the actual error by a small amount. The full adjoint error estimate continues to over estimate the actual error for $n_r = 20, 25, 30, 35$. The corresponding number of reduced-order basis functions for the residual and the Jacobian is $n_{RJ} = 50, 60, 70, 80$. It can be seen based on the trends for the projected adjoint error estimate, that n_{RJ} is unnecessarily large for $n_r = 15$. By $n_r = 25$, the projected error estimate shown in turquoise is able to estimate the mean of the actual error. However, by $n_r = 35$, the error estimate for the projected adjoint becomes slightly worse than before. More basis functions will improve the accuracy of the HROM, but it is possible that it doesn't necessarily improve the error estimates. More of this behavior will be seen in longer time simulations. The H_1 projected adjoint error estimate continues to under predict the actual error as before. Based on this test case, the best error set of parameters for $T = 60$ that can be used for error estimation is $n_r = 25$ and $n_{RJ} = 60$. Again, any more and the HROM becomes more expensive to use and more inaccurate, possibly due to the adjoint becoming less smooth with more basis functions. Smoother adjoints produce better projections for projected adjoint error estimates. Note that there is no clear trend on how the statistics for $T = 60$ behave.

Figure 7.3(d) shows results for the simulation time for $T = 80$ for $n_r = 10, 20, 30, 40, 60$. When the number of basis functions is set to $n_r = 10$, the number of basis functions for the residuals and Jacobians is set to $n_{RJ} = 40$. Again, the full adjoint error estimate over predicts the actual error by over three orders of magnitude. The error estimate for the projected adjoint just slightly under estimates the actual error. For $n_r = 20, 30, 40$, the corresponding number of basis functions for the residuals and the Jacobians is set to $n_{RJ} = 80, 80, 90$. For these basis functions, the error estimate for the full adjoint becomes slightly more accurate, while the projected error estimate closely resembles the actual estimates quite well. However, by $n_r = 40$ and $n_r = 80$, an interesting trend begins. At $n_r = 80$, the number of basis functions for the residuals and Jacobians is set to $n_{RJ} = 160$, which is quite high and more than needed to find the error estimate compared to previous time simulations. Due to this high value, convergence should not be an issue. However, between $n_r = 40$ and $n_r = 60$, the error estimates for the projected adjoint and H_1 projected adjoint become worse and begin to diverge away from the actual error, while the full adjoint error estimate continues to improve. These results show that there is an indeed a maximum number of basis functions n_r that will give accurate error estimates when using the projected adjoint, but that increasing n_r will improve the error estimates for the full adjoint. However, increasing n_r does again lead to a more expensive model. Based on this test case, the best set of parameters for $T = 80$ is one that gives error estimates with the fewest number of basis functions despite the fact that the error estimate for the full adjoint improves with more basis functions. The parameters that match this are $n_r = 20$ and $n_{RJ} = 80$. Note that there is no clear trend on how the statistics for $T = 80$ behave; however, the statistics at $n_r = 60$ are quite better than those at $n_r = 20$.

Figure 7.3(e) shows results for the simulation time for $T = 100$ for $n_r = 20, 30, 40, 50$. When the number of basis functions is set to $n_r = 20$, the number of basis functions for the residuals and Jacobian is set to $n_{RJ} = 100$. The full adjoint error estimate over predicts the actual error by just over two orders of magnitude showing that the error estimations are improving for the full adjoint with longer time simulations. The projected adjoint error estimation just slightly under predicts the error at $n_r = 20$. For $n_r = 30, 40, 50$, the corresponding number of basis functions for the residuals and the Jacobians is set to $n_{RJ} = 100, 100, 120$, respectively. As the number of basis functions n_r increases, the full adjoint error estimation improves and seems to approach the actual error estimation. In order to see if this trend continues, more cases with higher reduced-order basis functions are needed. The projected adjoint error estimates on the other hand begin to decrease in accuracy between $n_r = 40$ and $n_r = 50$. This could be due to the $n_r = 50$ case not having enough reduced-order basis functions for the residuals and Jacobian, but based on the results of Chapter VI, which shows that n_{RJ} has a small effect on accuracy and more influence on convergence, it is more likely that the projected adjoint error estimates decrease in accuracy when too many basis functions, n_r are used. These results are similar to those seen for $T = 60$ and $T = 100$ as well. Based on this test case, the optimal set of parameters for $T = 100$ that can be used for error estimation is $n_r = 20$ and $n_{RJ} = 100$. Again, any more and the HROM becomes more expensive to use then is needed, and possibly more inaccurate. Note that there is no clear trend to how the statistics for $T = 60$ behave; however, the standard deviations for all the error estimates for $n_r = 50$ decrease. Figure 7.3(f) shows results for the simulation time for $T = 120$ for $n_r = 20, 40, 60, 80$. When the number of basis functions is set to $n_r = 20$, the number of basis functions for the residuals and Jacobian is set to $n_{RJ} = 110$. The full adjoint error estimates over predict the actual errors, but do so only over one order of magnitude, an improvement from the previous time simulations. However, when the number of basis functions is increased to $n_r = 40, 80$, the full adjoint error estimates increases, which seems to be contrary to results from earlier time simulations. Note that the corresponding number of reduced basis functions for the residual and the Jacobian is set to $n_{RJ} = 120$ and $n_r = 160$, respectively. The projected adjoint error estimation for $n_r = 20$ underestimates the actual estimate error, but does more so than previous time simulations by under an order of magnitude. More basis functions are needed to improve the projected error estimates. When $n_r = 40$, the projected error estimates do not improve. But when $n_r = 80$, the projected error estimates improve significantly, producing results similar in accuracy as those of previous time simulations. This leads to a conclusion for $T = 120$, that more basis functions and hence more basis functions for the residuals and Jacobians are now needed to acquire accuracy in projected adjoint error estimates. To see if the projected adjoint error estimates can be improved further, more basis functions need to be executed with more basis functions for the residuals and the Jacobians. However, the full adjoint error estimates do not improve with more basis functions. Based on this test case, the best error set of parameters for $T = 120$ that can be used for error estimation is $n_r = 80$ and $n_{RJ} = 160$. Any less, and the error estimates are not accurate enough. Note that there is no clear trend on how the statistics for $T = 120$ behave.

Figure 7.4 shows the collective results from Figure 7.3 and results for longer simulation times, $T = 140, 160, 180, 200$. As seen in Figure 7.3, the projected error estimates estimate the actual errors well for each sub figure between $T = 20$ and $T = 120$. However, by $T = 140$, it is more difficult and a challenge to find the right number of basis functions and the right number of basis functions for the residuals and the Jacobians that would produce as accurate error estimates. Note that the mean projected error estimates are still within one order of magnitude of the mean actual error. This behavior can be a result of the KS equation becoming chaotic at a slower rate than the Lorenz attractor. In Chapter IV, it was seen that the adjoint even for the most chaotic case when the advection speed was set to $\alpha = 1$ and the super viscosity speed was set to $\nu = 0.25$ grew exponentially backwards slower than that of the Lorenz attractor. This can be reflected in the results where one can

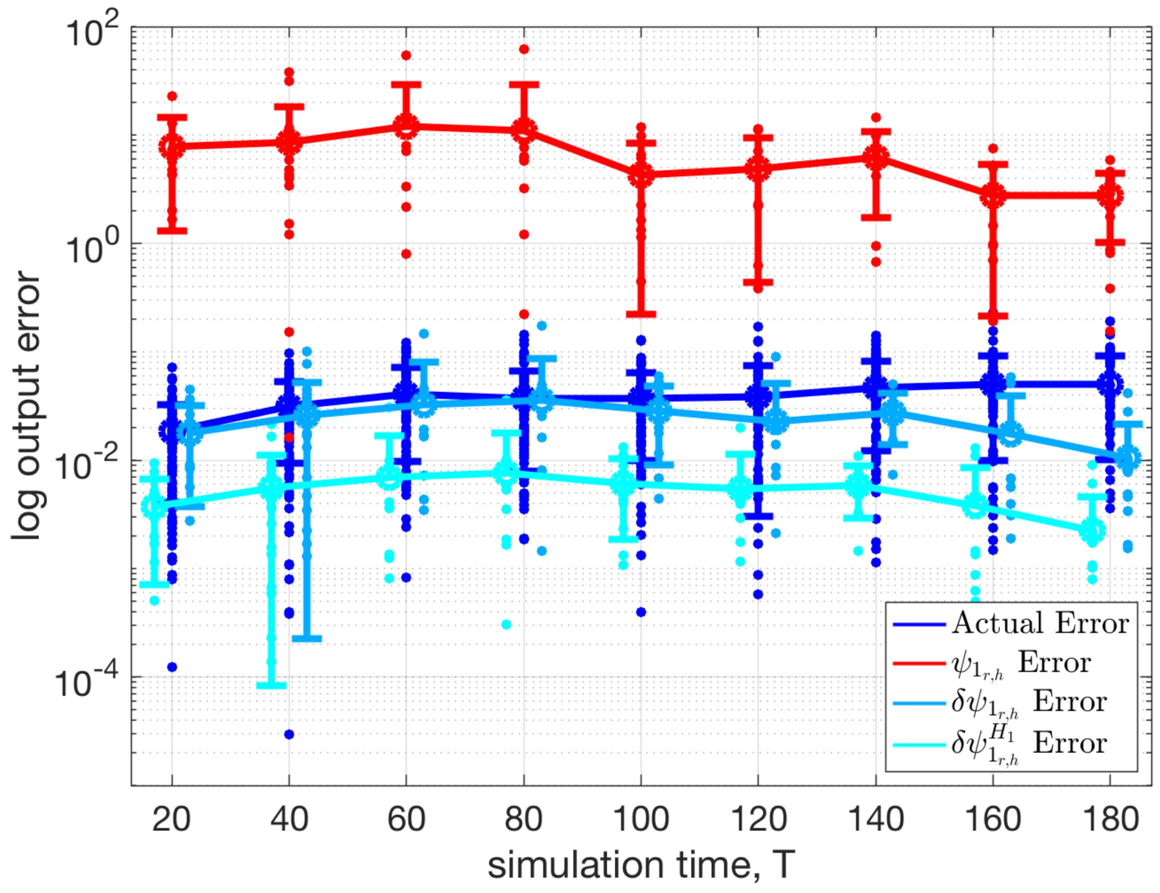


Figure 7.4: KS: LSS results for full adjoint error estimation, projected adjoint error estimation, and H_1 projected error estimation. The number of basis functions and number of basis functions for the residuals and the Jacobians is based on the results from Figure 7.3. The number sample nodes for all of these simulations is $n_i = 480$. The full adjoint error estimate over predicts the actual error for $T = [20, 200]$. The projected adjoint error estimation performs better when the influence of the coarse adjoint is taken away from the fine space adjoint. However, by $T = 140$, the projected adjoint error estimates become worse, but still within reasonable values. Note that KS adjoints increase exponential backwards in time at a slower rate than those of the Lorenz attractor. For the KS equation, the system does not become chaotic until $T \approx 100$. This could be the reason why it is more difficult to find accurate error estimates after $T \approx 100$.

see that the error estimates for the projected adjoint begin to diverge. It is important to note as well, that the full adjoint error estimates become more accurate by $T = 200$. It it possible that with longer time simulations, the full adjoint error estimations will converge with that of the actual error. However, with the projected adjoint error estimation, the error estimates are obtained faster for shorter periods of time, making the use of projected adjoints for error estimation more efficient for larger chaotic systems.

7.3.2 HROM-LSS Error Estimate and Adjoint Results for $n_i = 200$

In section 7.3.1, results were shown for HROM-LSS for when the number of sample nodes is set to $n_i = 480$. The next step is to see if similar results can be found when the computational costs of the HROM-LSS model is reduced even further by changing the number of sample nodes from $n_i = 480$ to $n_i = 200$. Note that from Chapter VI, reducing the number of sample nodes decreases the number of allowable reduced basis functions for the residuals and the Jacobians, n_{RJ}.

Figure 7.5 shows the results for the same parameters as that of Figure 7.4 except that the number of sample nodes is changed form $n_i = 480$ to $n_i = 200$. The purpose in lowering the number of sample nodes is to determine whether or not it is possible to obtain the same level of accuracy as the $n_i = 480$ case with fewer number of sample nodes. If it is possible to obtain the same level of accuracy, then less computational expense is needed, making the output-based error estimation method for chaotic systems overall more efficient than without the implementation of the GNAT method. The corresponding number of basis functions and reduced basis functions for the residuals and the Jacobians for each of the time simulations, T, are $n_r = 5, 20, 15, 20, 30, 40, 45, 50, 55$ and $n_{RJ} = 50, 60, 70, 80, 100, 120, 150, 160, 180$. Similar results as for $n_i = 480$ are obtained for $n_i = 200$. The trends for the full adjoint error estimate, the projected adjoint error estimate, and the H_1 projected error estimate are very similar. The growing error in the error estimate for $T = 140, 160, 180$ exists as well for the projected adjoint error estimate for $n_i = 200$, leading to the conclusion that more basis functions or basis residual and Jacobian basis functions may be required to improve the accuracy of the error estimates for the projected adjoint error estimate. Overall, reducing the number of sample nodes does not affect the accuracy of the error estimation.

The primal trajectory, HROM trajectory, and the corresponding adjoint for one test case from each time simulation for $n_i = 200$ can be seen in Figures 7.6, 7.7, 7.8, 7.9 for $T = [20, 180]$. The full adjoint was found from Eqn. 7.7. These figures show a comparison of the primal solution to the adjoint solution where regions of zero adjoint indicate areas where

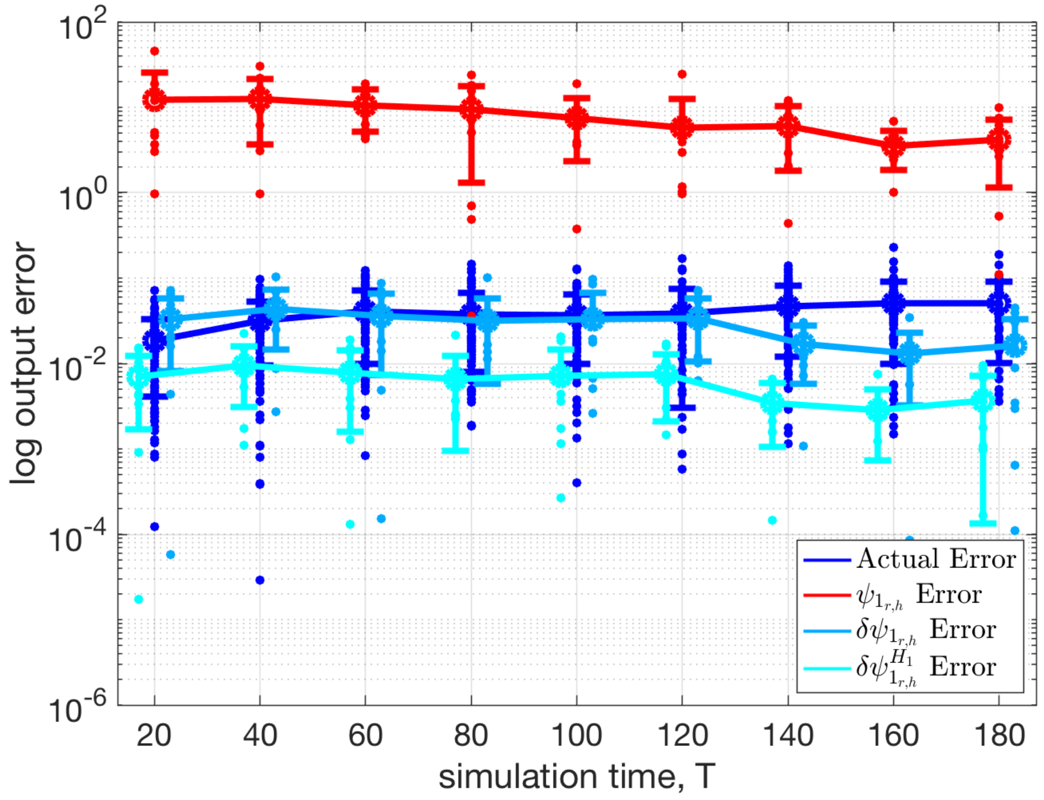
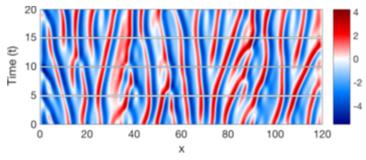
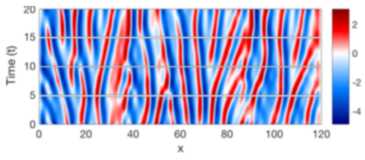
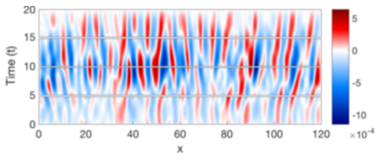
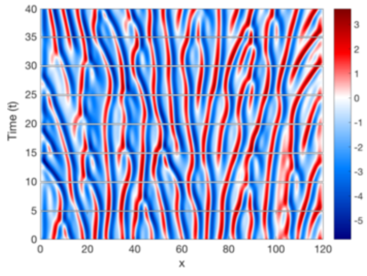


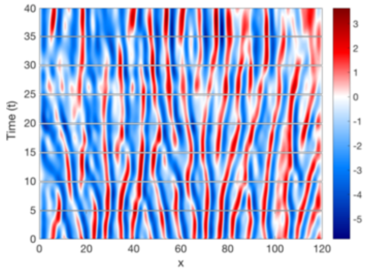
Figure 7.5: KS: LSS results for full adjoint error estimation, projected adjoint error estimation, and H_1 projected error estimation for $n_i = 200$ nodes. The full adjoint error estimate over predicts the actual error for $T = [20, 180]$. The projected adjoint error estimation performs better when the influence of the coarse adjoint is taken away from the fine space adjoint. However, by $T = 140$, the projected adjoint error estimates become worse as the $n_i = 480$ case, but still within reasonable values. Note that for the KS equation, the system does not become chaotic until $T \approx 100$. Even with fewer sample nodes, the error estimates are still recoverable compared to the $n_i = 480$ case. However, these results are less computationally expensive, making these results ideal. Note that this plot lacks results for $T = 200$ case. This is due to that for the $T = 200$, more sample nodes are required due to $T = 200$ needing more basis functions, n_r, with smaller number of sample nodes n_i.

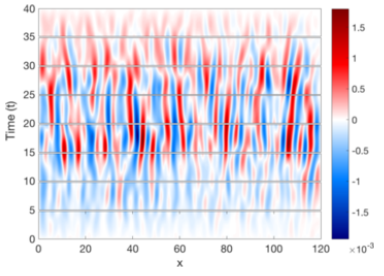

(a) $T = 20$ primal

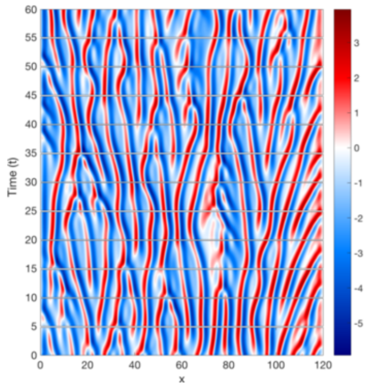

(b) $T = 20$ HROM

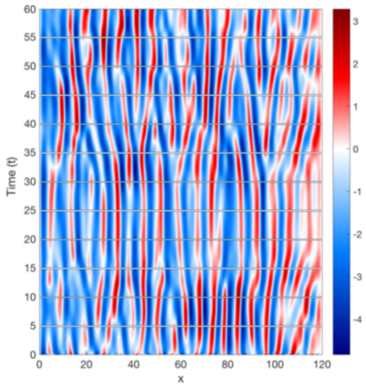

(c) $T = 20$ adjoint

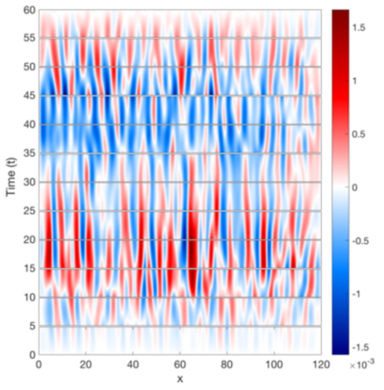

(d) $T = 40$ primal

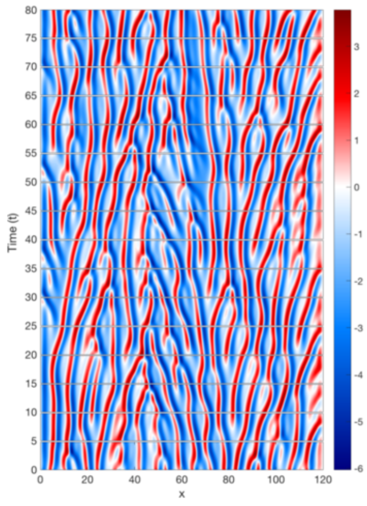

(e) $T = 40$ HROM

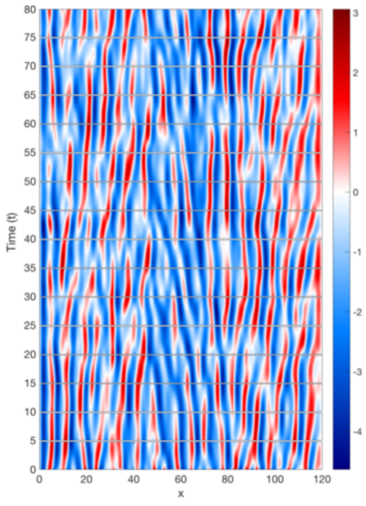

(f) $T = 40$ adjoint


(g) $T = 60$ primal


(h) $T = 60$ HROM


(i) $T = 60$ adjoint


(j) $T = 80$ primal


(k) $T = 80$ HROM

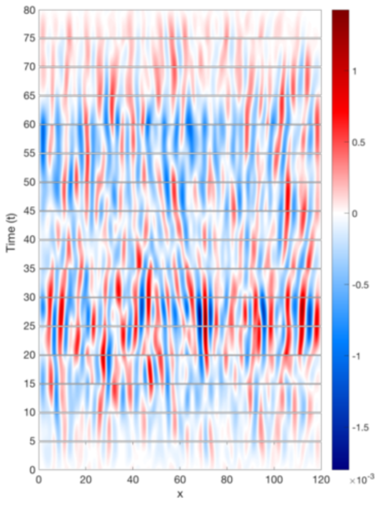
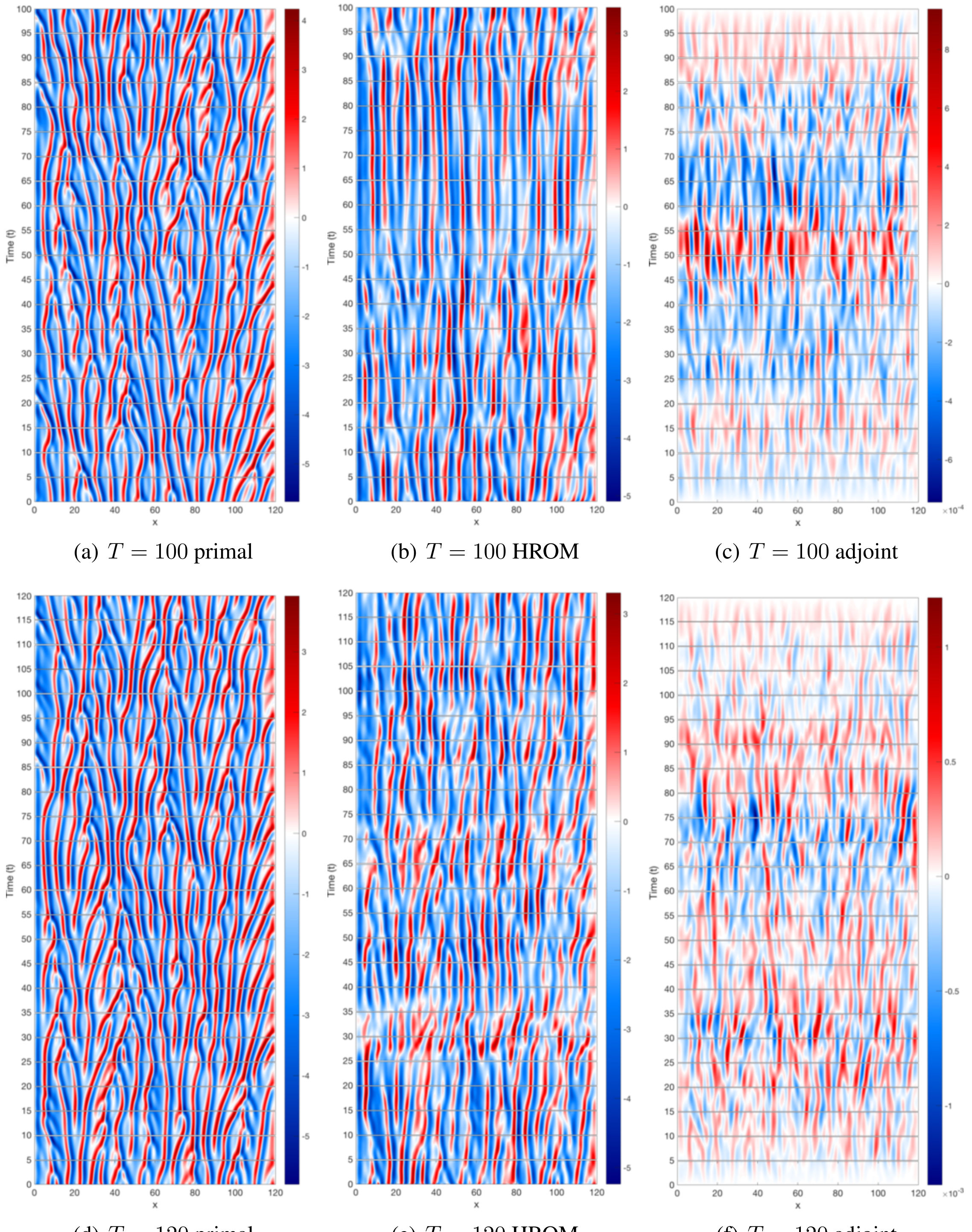

(l) $T = 80$ adjoint

Figure 7.6: KS: Example of primal, HROM, and approximate adjoint, $\boldsymbol{\psi}_1$, trajectories for one test case with $n_i = 200$ sample nodes for $T = 20, 40, 60, 80$.


(a) $T = 100$ primal (b) $T = 100$ HROM (c) $T = 100$ adjoint

(d) $T = 120$ primal (e) $T = 120$ HROM (f) $T = 120$ adjoint

Figure 7.7: KS: Example of primal, HROM, and adjoint, approximate $\boldsymbol{\psi}_1$, trajectories for one test case with $n_i = 200$ sample nodes for $T = 100, 120$.

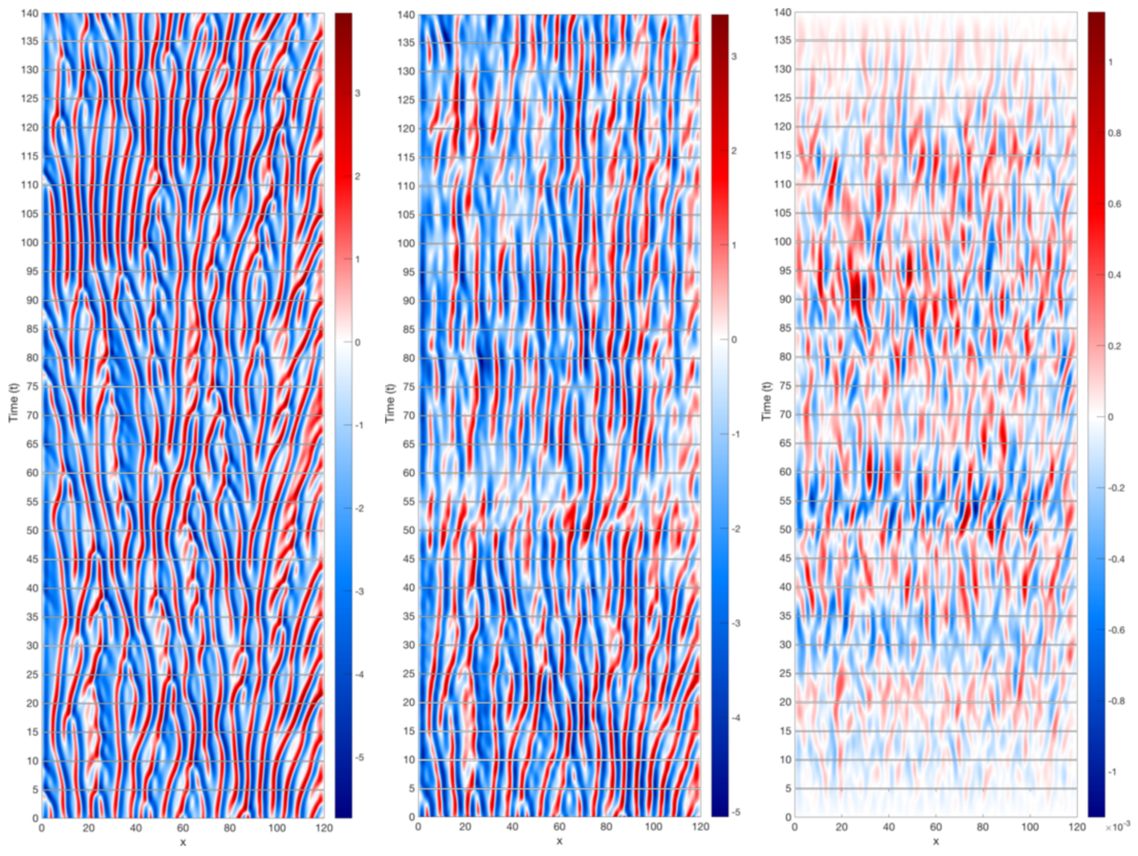
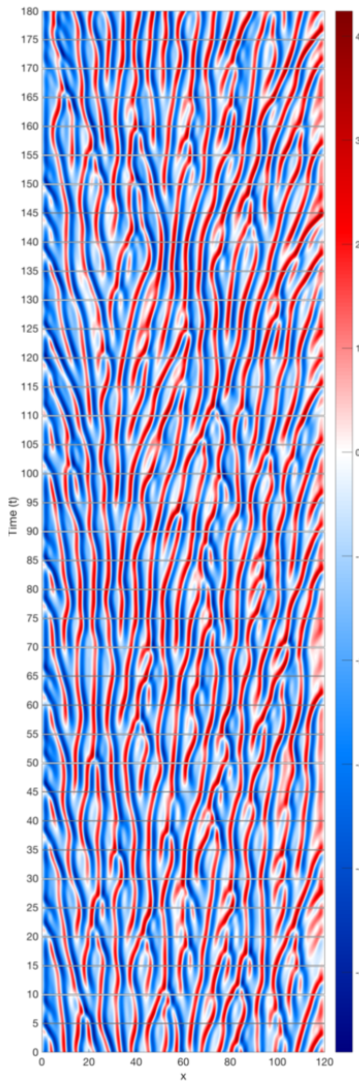
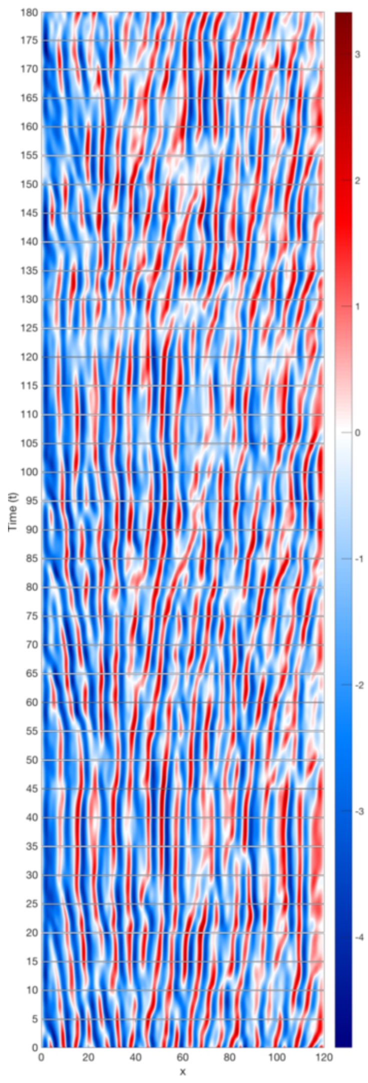

(a) $T = 140$ primal (b) $T = 140$ HROM (c) $T = 140$ adjoint

Figure 7.8: KS: Example of primal, HROM, and approximate adjoint, $\boldsymbol{\psi}_1$, trajectories for one test case with $n_i = 200$ sample nodes for $T = 140$.


(a) $T = 180$ primal


(b) $T = 180$ HROM

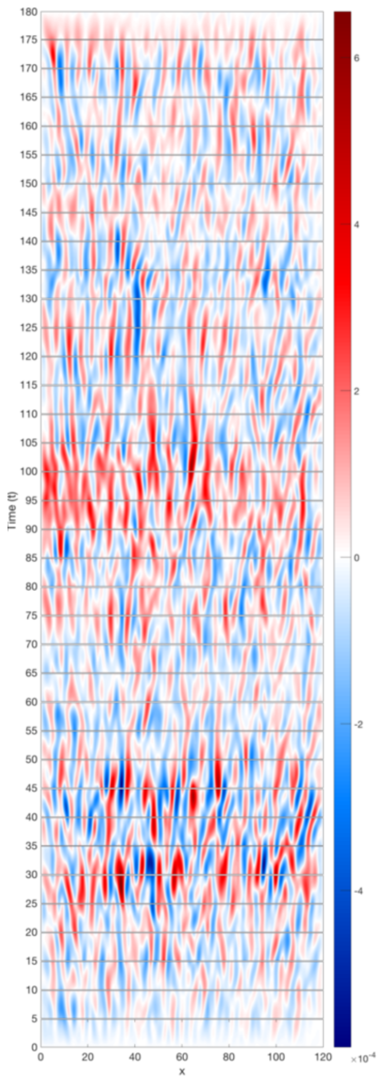

(c) $T = 180$ adjoint

Figure 7.9: KS: Example of primal, HROM, and approximate adjoint, $\boldsymbol{\psi}_1$, trajectories for one test case with $n_i = 200$ sample nodes for $T = 160$.

the residuals have no effect on the time-average output. Areas of non-zero adjoint indicate areas where residuals do have an effect on the adjoint. For mesh adaptation, these non-zero adjoint areas refer to regions where mesh refinement should take place. The adjoint results show that regions where there is non-zero influence of the residual on the time-average output, occur more during the middle of the time simulation. The adjoints for $T = [20, 180]$, are close to zero at the initial and final time of the simulation. The adjoint solutions share a few qualities with their corresponding HROM solution, such as the slightly unresolved presence of coherent structures. Note that with an increasing number of basis functions, n_r, the coherent structures in the adjoint solution will become more apparent with higher resolution.

7.3.3 Comparison of HROM-LSS with LSS for Kuramoto-Sivashinsky

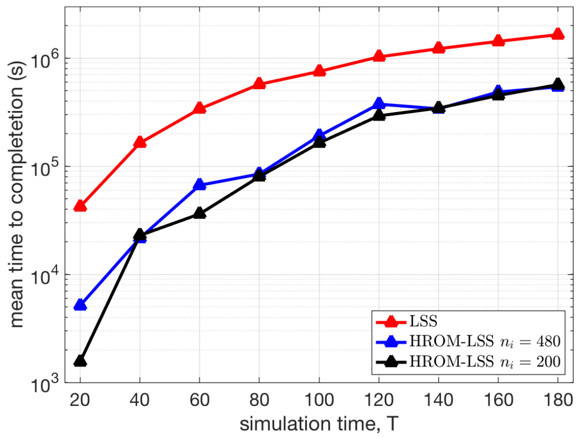
From Chapter V, it was seen that the full adjoint, the projected adjoint, and the H_1 projected adjoint gave error estimates that did not give as accurate results compared to that of the Lorenz attractor. The projected adjoint actually contributed to error estimates that were far worse compared to the simpler full adjoint. It was revealed that the projection process itself was inaccurate and difficult to fine possibly due to non-smooth/chaotic nature of the adjoint itself, increasing the overall error estimate instead of decreasing it. Another possible reason why the error estimates became worse, was that the coarse space adjoint is different enough than the find space adjoint. This behavior would be reflected in the error estimate. In order to remedy this problem, the H_1 projection was used as well, which projected the fine space adjoint given additional constraints on the slopes of the adjoint at the basis nodes for each element. This improved the error estimate, but gave error estimates that were on average similar to the error estimate of the full adjoint; with longer time periods, the H_1 projected adjoint performed better than the other error estimate. This behavior is due to the possibility that the coarse adjoint is relatively close to zero already. The results of LSS for the KS equations were not ideal. This is most likely due to the KS equations having a larger number of positive Lyapunov exponents compared to the Lorenz attractor case, making it more difficult to quantify the effect of discretization errors on the output for error estimation and mesh adaptation.

The next step that was taken was to reduce the LSS equations and use the HROM found from the GNAT method for error estimation with the goal of reducing the cost of LSS, while still obtaining the same level of accuracy from the KS results from Chapter V. The main difference between the results of HROM-LSS and the results of the LSS method is the computational cost associated with the GMRES linear solver.

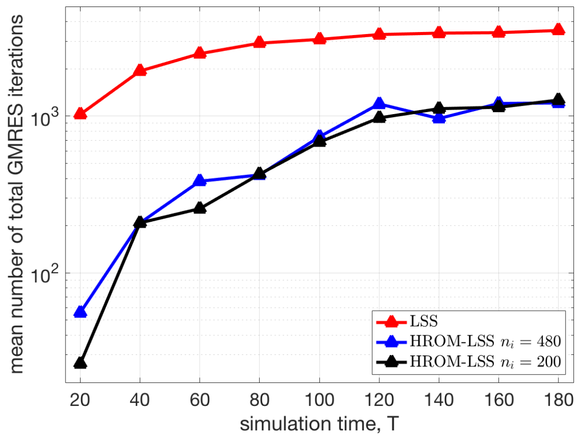
The full adjoint error estimates for LSS and HROM-LSS methods are similar. The error estimates for HROM-LSS and LSS with $T_k = 5$ are approximately on the order of 10^1, while the error estimates for HROM-LSS and LSS with $T_k = 4$ hover at approximately on the order of 10^0. It is promising that one is able to obtain the same results with fewer GMRES iterations, with less time needed to complete each GMRES iteration. The projected adjoint error estimates are plotted as well for HROM-LSS, which show some stark differences in behavior compared to that of the full LSS. Instead of increasing in value by almost an order of magnitude, the projected error estimates actually decrease by just under three orders of magnitude. Their accuracy improved greatly, especially at earlier time simulations. This is very different from the LSS results which showed that projecting the fine chaotic space adjoint on the coarse mesh actually increases the error estimate. This leads to the possibility that model reduction does more than reduce the states needed to solve the governing equations and the computational costs associated with calculating for its adjoints; model reduction for KS acts as a smoother such that the fine space adjoint is itself less oscillatory, allowing for the projection of the fine space adjoint onto the coarse mesh to be more accurate than before. This leads to a significant decrease in the error estimate and more accuracy. Hence, model reduction for the calculation of chaotic adjoints has two purposes: to reduce the computational cost of a large chaotic system to solve the adjoint equations and to produce smoother adjoints on the affine space that leads to increase accuracy of the adjoint and error estimation.

The H_1 projected adjoint produced results similar to the original projected adjoint for KS; however, it underestimated the actual error, but still produced more accurate results than that of the full adjoint. Hence, it can be concluded that the best error estimates for a chaotic system are found via model reduction and with a projected adjoint, $\boldsymbol{\psi}_{1_{r,h}}$.

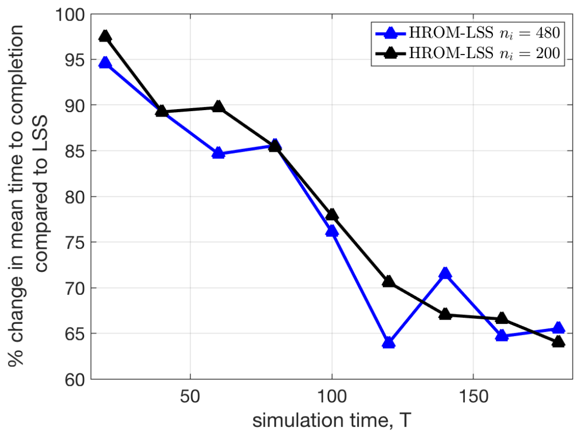
Compared to the reduced LSS case with $n_i = 480$, the results for the error estimates were found to be very similar. However, the main interest of reducing n_i is the effect it has on the time it takes to complete the simulation to find the adjoint. Figure 7.10 shows the average CPU time in seconds it takes to complete the adjoint calculation for LSS and HROM-LSS for each time simulation and the average number of total GMRES iterations it takes to complete the same adjoint calculation for LSS and HROM-LSS. The red data refers to the full LSS calculation, the blue line refers to the reduced LSS calculation with $n_i = 480$ sample nodes, and the black line refers to the reduced LSS calculation with $n_i = 200$ sample nodes. These results show that the HROM-LSS requires fewer GMRES iterations and less time to find the adjoint for a chaotic system, which happens in this case to be more accurate as well. However, on average the time it takes to complete the simulation



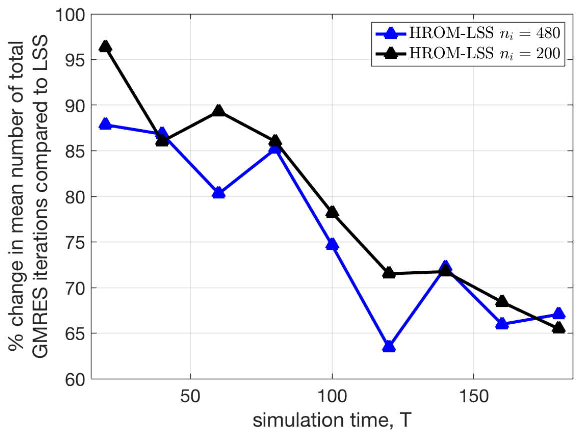
(a) Time to complete simulation for T



(b) Number of GMRES iterations for T



(c) Percentage change in time for HROM-LSS to complete simulation compared to LSS



(d) Percentage change in the number of GMRES iterations for HROM-LSS compared to LSS

Figure 7.10: KS: Comparison of the LSS method for $T_k = 5$ to the HROM-LSS method based on the results for $n_i = 480, 200$ for number average time to complete simulation and average number of GMRES iterations needed. Ran on Intel (R) Xeon (R) CPU X5450 @ 2.67 GHz

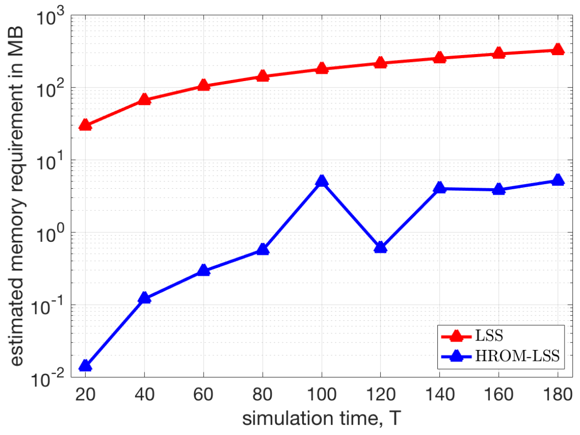
and the average number of GMRES iterations required for $n_i = 200$ is only slightly less and/or the same as the $n_i = 480$ cases, which seems counter intuitive. If less than half of the nodes are retained, the time it takes to complete the simulation should be significantly less; however, this is not necessarily the case especially and is mostly due to KS not being a large enough system to see a significant difference when decreasing the number of sample nodes.

Additionally, Figure 7.10 shows the percentage reduction in CPU time and the percentage reduction in number of GMRES iterations for HROM-LSS in comparison to LSS. Note that for $n_i = 480$ and $n_i = 200$, the percentage reduction in CPU time and number of GMRES iterations are similar. For $T = 20$, HROM-LSS calculated the error estimates 97.44% faster than LSS. For $T = 60$, HROM-LSS calculated the error estimates 84.6% faster than LSS. For $T = 120$, HROM-LSS calculated the error estimates 70.6% faster than LSS. For Kuramoto-Sivashinsky equation, this percentage reduction in CPU time for longer time simulations follows the following quadratic trend,

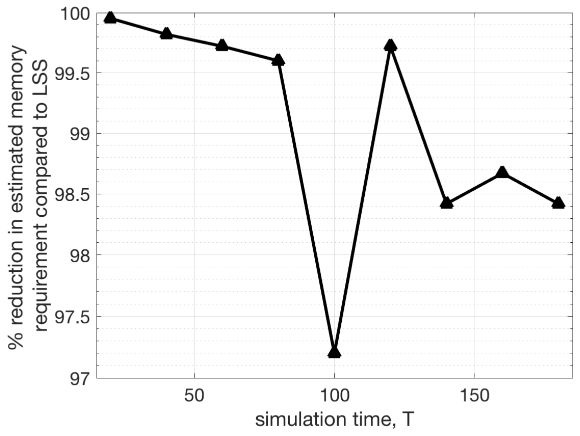
$$y = 0.00067T^2 - 0.33T + 100, \tag{7.31}$$

which was found found by fitting a quadratic equation to the HROM-LSS results for $n_i = 200$ in Figure 7.10(c). For longer simulation times beyond $T = 180$, the percentage reduction in time to completion for HROM-LSS and percentage reduction in the number of GMRES converge to a approximately 65%.

To further show that HROM-LSS is a more economical method than LSS, the estimated



(a) Percentage change in the number of GMRES iterations for HROM-LSS compared to LSS



(b) Percentage change in the number of GMRES iterations for HROM-LSS compared to LSS

Figure 7.11: KS: The estimated memory requirement for the LSS and the general HROM-LSS method.

memory requirement in MB is shown in Figure 7.11(a) for time simulations, $T = [20, 180]$. The estimated memory required to execute these techniques were found by comparing the size of the linear system of LSS. Since HROM-LSS creates overall a smaller system based on the number of basis functions, n_r, where $n_r < N$, the memory requirement for HROM-LSS is less than that of LSS. This can be seen in Figure 7.11(b), where by $T = 180$, the memory requirement for HROM-LSS is reduced by approximately 98.5% compared to LSS.

Overall for KS, HROM-LSS is computationally cheaper and more accurate when using the projected adjoint error estimate compared to LSS. In the next section, the HROM-LSS results for the Navier-Stokes equations are presented.

7.4 HROM-LSS results for Navier-Stokes Equations

Chapter IV introduced the Navier-Stokes equations, which are the governing equations of aerodynamics. It was shown that the Navier-Stokes equations for laminar flow at Reynolds number of $Re = 10^4$, produce interesting results for the adjoint when applying the traditional adjoint equation. Even though this flow is not chaotic, this simulation produces adjoints that suffer the same behavior as that of chaotic flows: the adjoint for the Navier-Stokes equations increases exponentially backwards in time, making it difficult to perform output-based error estimation and mesh adaption for high Reynolds number flows. Hence the GNAT method was used to find a HROM of same test case in Chapter IV and used with reduced LSS to calculate usable chaotic adjoints. DIRK3 was used advance the solution forward in time. The initial conditions for these each state components were perturbed by $\delta \boldsymbol{u} = \pm[0.1, 0.1, 0.1, 0.1]$ before the burn time was executed. The coarse space interpolation order was set to $p_H = 1$ and the fine space interpolation order was set to $p_h = 2$. The number of sample nodes was set to $n_i = 500$ out of the possible $N = 3198$ spatial nodes, which is approximately 15%, significantly lower than the KS case. The error estimate results for the Navier-Stokes equation are shown in Figure 7.12 The results from Figure 7.12 show only the error estimates for the full adjoint and the projected adjoint. (The H_1 projection error estimate was not calculated for the Navier-Stokes equations test case). Note that each test case for a simulation time, T, has a unique number of reduced basis functions and corresponding reduced basis functions for the residuals and the Jacobians. Unlike the KS case, the Navier-Stokes equations GNAT parameters are more sensitive to the perturbed conditions.

The actual error estimates are shown in blue, which compared to that of the KS case, show very low standard deviations. This is mostly likely due to Navier-Stokes case not

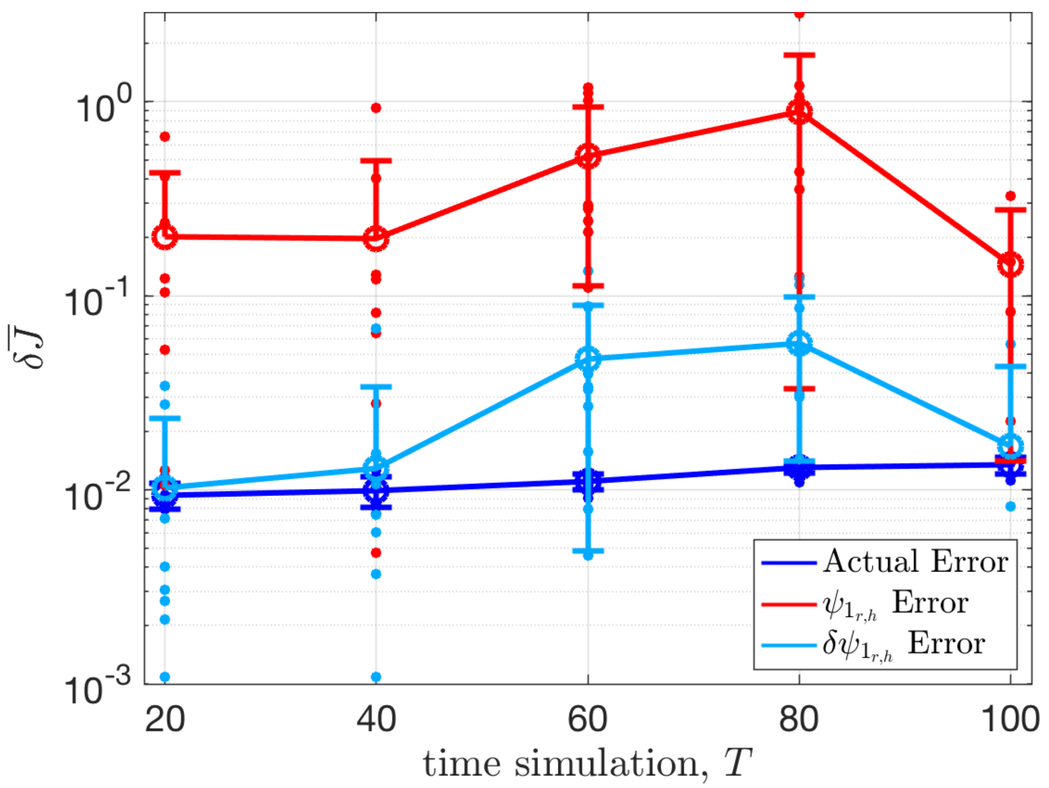


Figure 7.12: 2D Navier-Stokes error estimate and actual error results for $T = [20, 100]$

being chaotic. The full adjoint error estimates are shown in red and overestimate the actual error by approximately an order to two orders of magnitude, which is about two orders of magnitude less than that of the KS case. The projected error estimate is shown in light blue, which exhibits improvement in its estimates compared to that of the full adjoint. For $T = 20, 40$, the projected error estimate stays relatively close to the actual error estimate; however, the error estimates for $T = 60$ and $T = 80$ exhibit greater discrepancy within reason between the actual and estimated error. For these particular cases, more reduced basis functions and reduced basis functions for the residuals and Jacobians are needed. However, by $T = 100$, the error estimate for the full adjoint is more accurate.

These results were obtained with far fewer number of GMRES iterations compared to if error estimates were found with LSS. Note that the number of spatial nodes for the Navier-Stokes equations case is $N = 3198$.

7.5 Summary

In this chapter, the HROM-LSS method was presented, which combines the LSS technique and the GNAT method in order to calculate usable and accurate adjoints, efficiently.

HROM-LSS first reduces the LSS adjoint equations using model reduction techniques and the GNAT method presented in Chapter VI. Once the LSS adjoint equations were reduced, the HROM found from the GNAT technique was used instead of the primal solution to calculate adjoints for chaotic systems. The HROM-LSS technique is a function of four main parameters, the number of time windows, n_{seg}, the number of basis functions, n_r, the number of basis functions for the residuals and the Jacobians, n_{RJ}, and the number of sample nodes n_i. In order to reduce the computational costs resulting from the adjoint calculation, it is ideal to keep these parameters as low as possible without sacrificing the fidelity of the problem.

This chapter presented results for the HROM-LSS technique applied to the Kuramoto-Sivashinsky equation. From Chapter V, it was found that the LSS method over- predicted the actual error estimates by two orders of magnitude, which was not as accurate as that of the Lorenz attractor. The computational cost of the system was dictated by the size of the LSS linear solver system, $N(2n_{seg} - 1)$, which is quite high. On the contrary, it was found that the HROM-LSS method not only reduced the high computational cost of calculating the error estimates, but increased their accuracy as well. The error estimates that produced the most accurate results were based on the projected adjoint calculation. This increased accuracy is most likely due to the smoothing of the full adjoint as a result of the model reduction. As a result of the smoothing of the full adjoint, the projection of the fine adjoint to the coarse space was done more accurately than without the addition of the model reduction techniques. This regularization of the full adjoint is another possible reason why model reduction is a good addition to the adjoint calculation process for chaotic flows.

To reduce the size of the system further, the number of sample nodes was increased from $n_i = 480$ to $n_i = 200$, which produced results that were very similar to those of the $n_r = 480$ case, showing promise in the inclusion of the GNAT method in this combined process. The full adjoint trajectories were presented as well with their corresponding primal and HROM solutions in order to show the regions in space and time, where the residuals will affect the time-average output the most. In addition, the adjoint trajectories show where the residual has the most influence on the output, useful for mesh adaptation. Overall, the HROM-LSS method is able to retain the accuracy and even improve the error estimates while reducing the high computational costs associated with LSS, making it a more practical method to use for high fidelity methods such as LES.

To further see the effect of HROM-LSS, the method was implemented for the Navier-Stokes equations. Even though this Navier-Stokes case is not chaotic, its high Reynolds number and high angle of attack produces adjoints using the traditional adjoint sensitivity technique that increase exponentially backwards in time. Thus HROM-LSS would be a good technique to try for the Navier-Stokes equations. The results of this implementation were shown to be promising. The statistics of the error estimates for the Navier-Stokes equations behave differently as a result of the governing equations not being chaotic, but the error estimates based on the projected adjoint were able to predict the actual adjoint fairly well, within one order of magnitude. The number of sample nodes used was $n_i = 500$, which is proportionally much less than that of the KS equation. The number of sample nodes was about 15% of the total spatial nodes for the Navier-Stokes case, while the most reduced version of the KS results had about 42% of the total spatial nodes. Still, the HROM-LSS technique was able to produce accurate error estimates with many spatial nodes not being used in the calculations.

Overall, by using HROM-LSS, which reduces the LSS adjoint equations and uses model reduction techniques to reduce the primal solutions to n_r spatial nodes, one is able to overcome the butterfly effect and calculate usable and accurate adjoints compared to the traditional unsteady adjoint method and at lower computational costs compared to LSS.

CHAPTER VIII

Conclusions

8.1 Summary

This dissertation provides a new combined approach that allows one to quantify the effects of discretization errors on outputs of chaotic flows. Previous research in the field of computational fluid dynamics has shown success of the prediction of discretization errors through the use of output-based error estimation for steady and unsteady flows. In this research, the governing equations are solved using the discontinuous Galerkin finite element method. The output-based error estimation technique has been successful in predicting where the discretization errors occur in time and space, allowing for the use of mesh adaptation. As a result, output-based error estimation and mesh adaptation address some of the obstacles faced in computational fluid dynamics and high fidelity simulations. The success of output-based error estimation comes from the successful calculation of the adjoint, which is a sensitivity of the output of interest in terms of the discretized residual. The sensitivity of the output in terms of the residual, makes the adjoint a useful quantity to understand the errors associated with discretizations of governing equations. To calculate the adjoint, the linearization of the output and the residual are used to produced the dual, adjoint equation. This, combined with vast research in output-error estimation, has become a reliable technique. However, this is not the case for chaotic systems.

It was shown in Chapter IV that chaotic systems are unique compared to general unsteady flows. These systems are characterized by their Lyapunov exponents and corresponding covariant Lyapunov vectors which dictate the degree of divergence and separation of two close trajectories. This effect is commonly known as the butterfly effect and makes it difficult to use the traditional technique of adjoint calculations to find the dual equation that will work for chaotic systems. As a result, the adjoint increases backwards in time. Using these exponentially large adjoint values leads to unusable error estimates for mesh adaption. In computational fluid dynamics, providing this ability for output-based error estimation to work for chaotic systems is important, because it has many different applications that would be helpful. One major area that this research can benefit is in Large Eddy Simulation research, which is characterized by high computational costs and chaotic behaviors in its variables of interest. LES is known for high simulation costs and thus output-based error estimation would be greatly advantageous in reducing the computational time.

Chapter V introduced the Least Squares Shadowing method, which is an alternative to the traditional adjoint calculation. It was shown that by relaxing the initial conditions, a shadow trajectory could be found that stays close to the reference trajectory, allowing for one ideas from the traditional adjoint method to be used. The results of this method showed promise and was able to produce usable adjoints for error estimation. However, LSS has high computational costs due to its linear iterative solver used in its optimization process.

Chapter VI introduced model reduction techniques in an attempt to reduce the computational costs of LSS. The results of this chapter showed that is is possible to find a reduced order model and a hyper reduced order model when using the Gauss-Newton with Approximated Tensors technique to approximate the residual and Jacobian for a chaotic system.

Lastly, Chapter VII showed how the LSS adjoint equations are reduced further with hyper-reduction in the HROM-LSS method. The results of HROM-LSS for the KS equation and the Navier-Stokes equations showed that the method was able to reduce the computational cost of calculating the adjoint, while still retaining/improving the accuracy of the error estimate. For a time simulation of $T = 20$, the HROM-LSS method reduced the CPU time by 97.44% compared to LSS. For a time simulation of $T = 60$, the HROM-LSS method reduced the CPU time by 84.6% compared to LSS. By $T = 180$, the HROM-LSS method reduced the CPU time by 64% compared to LSS. As $T \to \infty$, the percentage reduction in the CPU time plateaus to approximately 60%. HROM-LSS, with the projected adjoint error estimation, was able to predict the discretization output error well, within an order of magnitude of the actual output error between interpolation orders, $p_H = 2$ and $p_h = 3$.

8.2 Research Contributions

The major contributions of this dissertation are:

- Implementation of the Least Squares Shadowing Method with the discontinuous Galerkin method.

- Extension of the Least-Squares Petrov-Galerkin method and the Gauss-Newton with

Approximated Tensors technique to chaotic systems, specifically, the Kuramoto-Sivashinsky equation and the two-dimensional Navier-Stokes equations.

- Development of the Hyper-Reduced-Order Modeling-Least Squares Shadowing technique which performed as well as the Least Squares Shadowing method at calculating adjoints for chaotic flows, but with less computational resources.

8.3 Future Work

The HROM-LSS method has shown promise for output-based error estimation of chaotic flows. However, based on the results of this method, additional research in areas of mesh adaptation, model reduction, and Least Squares Shadowing will only improve the method further.

- **Space-Time Least-Squares Petrov-Galerkin**

  Least-Squares Petrov-Galerkin and GNAT reduces the size of the system in space. However, in this dissertation, temporal reduction has not been mentioned. For unsteady problems, many time discretization methods are expensive, especially DG in time. To reduce the computational costs further, the LSPG in time technique can be implemented as well in time, reducing the overall costs further. (This method temporal reduction method is referred to the ST-LSPG method [88]).

- **Mesh Adaptation**

  Along with estimating the output error, the results of this dissertation can be used for mesh adaptation. However, work is needed to figure out how to successfully incorporate the ideas of reduced-order modeling with mesh adaption. For example, it is unclear if the same reduced-order basis function matrix, $\mathbf{\Phi}$, can be used after one iteration of adaptation. Overall, incorporation of mesh adaptation with HROM-LSS will further improve the capabilities of the chaotic simulations.

- **Finding a Preconditioner with HROM-LSS Properties for LSS**

  The results from Chapter VII showed that the projected adjoint error estimates accurately predicted the effect of the discretization errors on the output compared to the full adjoint error estimate. It was shown that the projection of the fine space adjoint in the coarse space performed better for HROM-LSS than for LSS. This leads to the possibility that the adjoint from HROM-LSS is regularized so that the adjoint itself has more favorable properties than before. Applying model reduction techniques in general may improve the distribution of the Lyapunov exponents, resulting in a more manageable system. This similar behavior can be seen with preconditioners for systems with high condition numbers which are used to transform the system to have better spectral properties. In general, by using a preconditioner, the iterative solver can converge quickly than without it. It may be possible to obtain similar and/or improved results compared to that of Chapter VI by finding a preconditioner, $\boldsymbol{M}$, that has the same properties as HROM-LSS,

$$\boldsymbol{M}^{-1}\boldsymbol{A}x = \boldsymbol{M}^{-1}\boldsymbol{b}. \tag{8.1}$$

Solving 8.1 will theoretically give the same answer as the full order LSS equation in Eqn. 5.38, but at a higher convergence rate. Finding and using a preconditioner $\boldsymbol{M}$ that exhibits the same properties of the model reduction performed in this dissertation would provide an alternative, non-intrusive/offline way to reduce the computational costs of LSS. This would simplify the HROM-LSS technique and make it easier to apply output-based error estimation to more complicated problems.

- **Application of Model Reduction Techniques to the Full Trajectory Design of the Least Squares Shadowing Problem**

  It was shown in Chapter V that there are two different ways to solve for the LSS adjoint equations: the full trajectory design method and the checkpoint design. In this dissertation, model reduction techniques were applied to the checkpoint design to solve for the LSS adjoint equations. Originally the LSS checkpoint design was used to solve for the full LSS adjoints equation due to the checkpoint design having fewer variables to solve than the full trajectory design. However, applying model reduction techniques directly to the adjoint equations of the full trajectory design could be potentially simpler to implement than the checkpoint design.

  To apply model reduction techniques to the full trajectory design, the adjoint equations would have to rederived to take into account of the time transformation. Before the time transformation for the checkpoint design was taken into account by using projection operators. The KKT conditions from Chapter V with the time transformation term form the following system,

$$\frac{d\boldsymbol{v}(t)}{dt} = \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} + \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})\eta(t), \tag{8.2}$$

$$\frac{d\boldsymbol{w}(t)}{dt} = -\left(\frac{\partial \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\mu});\boldsymbol{\mu})}{\partial \boldsymbol{u}}\right)^{*} \boldsymbol{w}(t) + \boldsymbol{v}(t), \tag{8.3}$$

$$\alpha^2 \eta = -\langle \boldsymbol{f}, \boldsymbol{w} \rangle. \tag{8.4}$$

These adjoint equations can be directly discretized with the discontinuous Galerkin to find the linear system,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\psi}_{2,0} \\ \boldsymbol{\psi}_{2,1} \\ \boldsymbol{\psi}_{2,m} \\ \eta_1 \\ \eta_m \\ \boldsymbol{\psi}_{1,1} \\ \boldsymbol{\psi}_{1,m} \end{bmatrix}, \tag{8.5}$$

where the reduced $\boldsymbol{x}$ can be found from the affine subspace of $\boldsymbol{A}$ such that

$$\boldsymbol{x} \approx \Phi_x \boldsymbol{x}_r. \tag{8.6}$$

The linear model reduction techniques from section 6.3 can be used to reduce the linear system by looking at the left nullspace,

$$\boldsymbol{\varphi}^T \left[\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\right]. \tag{8.7}$$

Since this is a linear problem, Galerkin projection can be used. However, it is unclear what $\boldsymbol{\Phi}_x$ should be for the full trajectory design. Based on the relationship of the full adjoint to its reduced adjoints, $\boldsymbol{\Phi}_x$ may be a function of $\boldsymbol{\Phi}$. Thus, the full forward problem simulation is still needed to find to find $\boldsymbol{\Phi}$. More research into what can be used for $\boldsymbol{\Phi}_x$ is need.

- **Output-Based Error Estimation for Turbulent Flows**

  The Navier-Stokes equations in this dissertation are laminar and not chaotic. The next step would be to implement the HROM-LSS method for a turbulent chaotic case to see if the method can be translated from the laminar to a truly turbulent case.

- **Output-Based Error Estimation for 3D Chaotic Flows**

  Due to the promising results for the two-dimensional Navier-Stokes results, the

HROM-LSS can be applied to the laminar three-dimensional Navier-Stokes equations as well.

BIBLIOGRAPHY

[1] "Number 1 (Lavender Mist), 1950," Available at `www.wikiart.org/en/jackson-pollock/number-1-lavender-mist-1950-1`.

[2] Fidkowski, K., "Output-based error estimation and mesh adaptation for steady and unsteady flow problems," *38th Advanced VKI CFD Lecture Series. von Karman Institute*, 2015.

[3] Cockburn, B. and Shu, C.-W., "Runge–Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of scientific computing*, Vol. 16, No. 3, 2001, pp. 173–261.

[4] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113.

[5] Bey, K. S. and Oden, J. T., "hp-version discontinuous Galerkin methods for hyperbolic conservation laws," *Computer Methods in Applied Mechanics and Engineering*, Vol. 133, No. 3-4, 1996, pp. 259–286.

[6] Hartmann, R. and Houston, P., "Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations," *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532.

[7] Fidkowski, K. J., *A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[8] Houston, P., Senior, B., and Süli, E., "hp-Discontinuous Galerkin finite element methods for hyperbolic problems: error analysis and adaptivity," *International Journal for Numerical Methods in Fluids*, Vol. 40, No. 1-2, 2002, pp. 153–169.

[9] Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods*, Springer, 2000, pp. 197–208.

[10] Bassi, F. and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier–Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, No. 1-2, 2002, pp. 197–207.

[11] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., "Unified analysis of discontinuous Galerkin methods for elliptic problems," *SIAM journal on numerical analysis*, Vol. 39, No. 5, 2002, pp. 1749–1779.

[12] Brezzi, F., Marini, L. D., and Süli, E., "Discontinuous Galerkin methods for first-order hyperbolic problems," *Mathematical models and methods in applied sciences*, Vol. 14, No. 12, 2004, pp. 1893–1903.

[13] Peraire, J. and Persson, P.-O., "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM Journal on Scientific Computing*, Vol. 30, No. 4, 2008, pp. 1806–1824.

[14] Bar-Yoseph, P. and Elata, D., "An efficient L2 Galerkin finite element method for multi-dimensional non-linear hyperbolic systems," *International Journal for Numerical Methods in Engineering*, Vol. 29, No. 6, 1990, pp. 1229–1245.

[15] Lowrie, R. B., Roe, P. L., and van Leer, B., "Properties of space-time discontinuous Galerkin," Tech. rep., 1998.

[16] van der Vegt, J., "Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows," Tech. rep., National Aerospace Laboratory NLR-TP-98239, 1998.

[17] Van der Ven, H. and van der Vegt, J. J., "Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: II. Efficient flux quadrature," *Computer methods in applied mechanics and engineering*, Vol. 191, No. 41-42, 2002, pp. 4747–4780.

[18] Klaij, C. M., van der Vegt, J. J., and van der Ven, H., "Space–time discontinuous Galerkin method for the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 217, No. 2, 2006, pp. 589–611.

[19] Barth, T. J., "Space-time error representation and estimation in Navier-Stokes calculations," *Complex Effects in Large Eddy Simulations*, Springer, 2007, pp. 29–48.

[20] Meidner, D. and Vexler, B., "Adaptive Space-Time Finite Element Methods for Parabolic Optimization Problems," *SIAM Journal on Control Optimization*, Vol. 46, No. 1, 2007, pp. 116–142.

[21] Schmich, M. and Vexler, B., "Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 1, 2008, pp. 369–393.

[22] Fidkowski, K. J. and Luo, Y., "Output-based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[23] Reed, W. H. and Hill, T., "Triangular mesh methods for the neutron transport equation," Tech. rep., Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[24] Cockburn, B., Karniadakis, G. E., and Shu, C.-W., "The development of discontinuous Galerkin methods," *Discontinuous Galerkin Methods*, Springer, 2000, pp. 3–50.

[25] Fidkowski, K. J., "Aero 623 Course Notes: Advanced Computational Fluid Dynamics," 2017, p. 144.

[26] LeVeque, R., *Numerical Methods for Conservation Laws*, Lectures in Mathematics ETH Zürich, Department of Mathematics Research Institute of Mathematics, Springer, 1992.

[27] MIT, *Lecture 12: Numerical Schemes for Scalar One-Dimensional Conservation Laws*, MIT 16.920 Class Notes, 2011.

[28] Georgoulis, E. H., Houston, P., and Virtanen, J., "An a Posteriori Error Indicator for Discontinuous Galerkin Approximations of Fourth-Order Elliptic Problems," *IMA journal of numerical analysis*, 2009, pp. drp023.

[29] Fidkowski, K. J., "A Hybridized Discontinuous Galerkin Method on Mapped Deforming Domains," *Computers and Fluids*, Vol. 139, No. 5, November 2016, pp. 80–91.

[30] Kennedy, C. A. and Carpenter, M. H., "Diagonally implicit runge-kutta methods for ordinary differential equations. a review," 2016.

[31] Pierce, N. A. and Giles, M. B., "Adjoint recovery of superconvergent functionals from PDE approximations," *SIAM review*, Vol. 42, No. 2, 2000, pp. 247–264.

[32] Becker, R. and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta numerica*, Vol. 10, 2001, pp. 1–102.

[33] Hartmann, R. and Houston, P., "Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations," *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532.

[34] Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.

[35] Nemec, M. and Aftosmis, M., "Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes," *18th AIAA Computational Fluid Dynamics Conference*, 2007, p. 4187.

[36] Fidkowski, K. J. and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *AIAA journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[37] Mani, K. and Mavriplis, D., "Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems," *18th AIAA Computational Fluid Dynamics Conference*, 2007, p. 3944.

[38] Mani, K. and Mavriplis, D. J., "Error estimation and adaptation for functional outputs in time-dependent flow problems," *Journal of Computational Physics*, Vol. 229, No. 2, 2010, pp. 415–440.

[39] Fidkowski, K. J., "An Output-Based Dynamic Order Refinement Strategy for Unsteady Aerodynamics," AIAA Paper 2012-77, 2012.

[40] Fidkowski, K. J., "Output-Based Space-Time Mesh Optimization for Unsteady Flows Using Continuous-in-Time Adjoints," *Journal of Computational Physics*, Vol. 341, No. 15, July 2017, pp. 258–277.

[41] Krakos, J. A., *Unsteady Adjoint Analysis for Output Sensitivity and Mesh Adaptation*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[42] Belme, A., Dervieux, A., and Alauzet, F., "Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows," *Journal of Computational Physics*, Vol. 231, No. 19, 2012, pp. 6323–6348.

[43] Flynt, B. T. and Mavriplis, D. J., "Discrete Adjoint Based Adaptive Error Control in Unsteady Flow Problems," AIAA Paper 2012-0078, 2012.

[44] Schmich, M. and Vexler, B., "Adaptivity with Dynamic Meshes for Space-Time Finite Element Discretizations of Parabolic Equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 1, 2008, pp. 369–393.

[45] Fidkowski, K. J., "An Output-Based Adaptive Hybridized Discontinuous Galerkin Method on Deforming Domains," AIAA Paper 2015–2602, 2015.

[46] Nguyen, N., Peraire, J., and Cockburn, B., "An Implicit High-Order Hybridizable Discontinuous Galerkin Method for Linear Convection-Diffusion Equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 3232–3254.

[47] Cockburn, B., Gopalakrishnan, J., and Lazarov, R., "Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems," *SIAM Journal on Numerical Analysis*, Vol. 47, No. 2, 2009, pp. 1319–1365.

[48] Nguyen, N. C., Peraire, J., and Cockburn, B., "Hybridizable Discontinuous Galerkin Methods," *Spectral and High Order Methods for Partial Differential Equations*, edited by J. S. Hesthaven, E. M. Rønquist, T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, Vol. 76 of *Lecture Notes in Computational Science and Engineering*, Springer Berlin Heidelberg, 2011, pp. 63–84.

[49] Woopen, M., Balan, A., May, G., and Schütz, J., "A Comparison of Hybridized and Standard DG Methods for Target-Based hp-adaptive Simulation of Compressible Flow," *Computers & Fluids*, Vol. 98, 2014, pp. 3–16.

[50] Dahm, J. P. and Fidkowski, K. J., "Error Estimation and Adaptation in Hybridized Discontinous Galerkin Methods," AIAA Paper 2014–0078, 2014.

[51] Fidkowski, K. J., "Output Error Estimation Strategies for Discontinuous Galerkin Discretizations of Unsteady Convection-Dominated Flows," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 12, 2011, pp. 1297–1322.

[52] Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation," *Computers & Fluids*, Vol. 28, No. 4-5, 1999, pp. 443–480.

[53] Giles, M. B. and Pierce, N. A., "Analytic adjoint solutions for the quasi-one-dimensional Euler equations," *Journal of Fluid Mechanics*, Vol. 426, 2001, pp. 327–345.

[54] Fidkowski, K., "AE 623 Classnotes: Adjoint Sensitivity Analysis," 2013, pp. 1–3.

[55] Oliver, T. A. and Darmofal, D. L., "Analysis of dual consistency for discontinuous Galerkin discretizations of source terms," *SIAM Journal on Numerical Analysis*, Vol. 47, No. 5, 2009, pp. 3507–3525.

[56] Lu, J., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.

[57] Chen, G. and Fidkowski, K., "Output-based mesh adaptation for multifidelity PDE-constrained optimization," *Journal of Computational Physics*, 2018.

[58] Dahm, J., "Toward Accurate, Efficient, and Robust Hybridized Discontinuous Galerkin Methods," 2017.

[59] Dolnick, E., *The Clockwork Universe: Isaac Newton, the Royal Society, and the Birth of the Modern World*, P.S., HarperCollins, 2012.

[60] Laplace, P. and Dale, A., *Pierre-Simon Laplace Philosophical Essay on Probabilities: Translated from the fifth French edition of 1825 With Notes by the Translator*, Sources in the History of Mathematics and Physical Sciences, Springer New York, 1998.

[61] Lorenz, E. N., "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, Vol. 20, No. 2, 1963, pp. 130–141.

[62] Strogatz, S. H., *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*, CRC Press, 2018.

[63] Gleick, J., *Chaos: Making a New Science*, A Penguin Book: Science, Penguin, 1988.

[64] Ginelli, F., Chaté, H., Livi, R., and Politi, A., "Covariant lyapunov vectors," *Journal of Physics A: Mathematical and Theoretical*, Vol. 46, No. 25, 2013, pp. 254005.

[65] Eckmann, J.-P. and Ruelle, D., “Ergodic theory of chaos and strange attractors,” *The Theory of Chaotic Attractors*, Springer, 1985, pp. 273–312.

[66] Gallavotti, G. and Cohen, E. G. D., “Dynamical ensembles in stationary states,” *Journal of Statistical Physics*, Vol. 80, No. 5-6, 1995, pp. 931–970.

[67] Richter, T., “Discontinuous Galerkin as time-stepping scheme for the Navier–Stokes equations,” *Modeling, Simulation and Optimization of Complex Processes*, Springer, 2012, pp. 271–281.

[68] Hyman, J. and Nicolaenko, B., “The Kuramoto-Sivashinsky Equation: A Bridge Between PDE’s and Dynamical Systems,” *Physica D: Nonlinear Phenomena*, Vol. 18, 1986, pp. 113–126.

[69] Kuramoto, Y. and Tsuzuki, T., “On the Formation of Dissipative Structures in Reaction-Diffusion Systems Reductive Perturbation Approach,” *Progress of Theoretical Physics*, Vol. 54, No. 3, 1975, pp. 687–699.

[70] Blonigan, P. J., Gomez, S. A., and Wang, Q., “Least squares shadowing for sensitivity analysis of turbulent fluid flows,” *52nd Aerospace Sciences Meeting*, 2014, p. 1426.

[71] Wang, Q., “Forward and Adjoint Sensitivity Computation of Chaotic Dynamical Systems,” *Journal of Computational Physics*, Vol. 235, 2013, pp. 1–13.

[72] Wang, Q., Hu, R., and Blonigan, P., “Least Squares Shadowing Sensitivity Analysis of Chaotic Limit Cycle Oscillations,” *Journal of Computational Physics*, Vol. 267, 2014, pp. 210–224.

[73] Wang, Q., “Convergence of the Least Squares Shadowing Method for Computing Derivative of Ergodic Averages,” *SIAM Journal on Numerical Analysis*, Vol. 52, No. 1, 2014, pp. 156–170.

[74] Pilyugin, S. Y., *Shadowing in dynamical systems*, Springer, 2006.

[75] Amsallem, D., Zahr, M. J., and Farhat, C., “Nonlinear Model Order Reduction Based on Local Reduced-order bases,” *International Journal for Numerical Methods in Engineering*, Vol. 92, No. 10, 2012, pp. 891–916.

[76] Carlberg, K., Bou-Mosleh, C., and Farhat, C., “Efficient Non-Linear Model Reduction Via a Least-Squares Petrov–Galerkin Projection and Compressive Tensor Approximations,” *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, 2011, pp. 155–181.

[77] Carlberg, K., Cortial, J., Amsallem, D., Zahr, M., and Farhat, C., “The GNAT Nonlinear Model Reduction Method and its Application to Fluid Dynamics Problems,” *6th AIAA Theoretical Fluid Mechanics Conference*, Vol. 2730, 2011, pp. 2011–3112.

[78] Rewienski, M. and White, J., "Improving trajectory piecewise-linear approach to nonlinear model order reduction for micromachined devices using an aggregated projection basis," *Proceedings of the 5th International Conference on Modeling and Simulation of Microsystems*, 2002, pp. 128–131.

[79] Nahvi, S. A., Nabi, M.-u., and Janardhanan, S., "Trajectory piece-wise quasi-linear approximation of large non-linear dynamic systems," *International Journal of Modelling, Identification and Control*, Vol. 19, No. 4, 2013, pp. 369–377.

[80] Schmid, P. J., "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, Vol. 656, 2010, pp. 5–28.

[81] Alla, A. and Kutz, J. N., "Nonlinear model order reduction via dynamic mode decomposition," *SIAM Journal on Scientific Computing*, Vol. 39, No. 5, 2017, pp. B778–B796.

[82] Zhang, W. and Wei, M., "Model order reduction using DMD modes and adjoint DMD modes," *8th AIAA Theoretical Fluid Mechanics Conference*, 2017, p. 3482.

[83] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D., "The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows," *Journal of Computational Physics*, Vol. 242, 2013, pp. 623–647.

[84] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T., "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathematique*, Vol. 339, No. 9, 2004, pp. 667–672.

[85] Chaturantabut, S., "Dimension reduction for unsteady nonlinear partial differential equations via empirical interpolation methods," Tech. rep., 2009.

[86] Chaturantabut, S. and Sorensen, D. C., "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[87] Everson, R. and Sirovich, L., "Karhunen–Loève Procedure for Gappy Data," *J. Opt. Soc. Am. A*, Vol. 12, No. 8, Aug 1995, pp. 1657–1664.

[88] Choi, Y. and Carlberg, K., "Space-time least-squares Petrov-Galerkin projection for nonlinear model reduction," *arXiv preprint arXiv:1703.04560*, 2017.