Output-Based Error Estimation for Chaotic Flows Using Reduced-Order Modeling

Yukiko S. Shimizu[*] and Krzysztof J. Fidkowski[†]

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

A direct application of the unsteady output adjoint method for error estimation is not possible for chaotic flows due to their inherent sensitivity to the initial conditions and the exponential growth of their corresponding adjoint solutions. A method that has proven to provide accurate and usable output adjoints for chaotic flows is the least squares shadowing method (LSS), which was originally developed for sensitivity analysis. However, this approach has been shown to be costly, especially for larger simulations. Another method, the approximate time-windowing approach, has been shown to be cheaper to implement and execute. However, this method produces results that are not as accurate as that of LSS. In this paper, we choose to concentrate on reducing the computational cost associated with LSS by using reduced-order modeling (ROM) and hyper-reduced-order modeling (HROM). We first present a study on how accurate ROM and HROM can be for a chaotic system. We then introduce a combined reduced-order model, least squares shadowing method (HROM-LSS) for approximating output adjoints more accurately and more economically compared to previous approaches. Lastly, we present preliminary HROM-LSS results for the 1D chaotic Kuramoto-Sivashinsky (KS) problem.

I. Introduction

Turbulent flows are characterized by chaotic variations in state variables such as velocity and pressure. These variations are a result of mixing in the flow, which is found in many different aerospace applications such as jet engine mixing and flow over bluff bodies. Large-eddy simulations (LES) of these chaotic turbulent flows provide useful information in the design process. However, LES is resource- and time-intensive, paving the way for opportunities to investigate ideas that would make LES and other kinds of turbulent solvers run more efficiently. One way to decrease the computational costs of these simulations is to quantify the numerical errors in these chaotic systems, so as to enable mesh adaptation. By reducing costs without sacrificing accuracy, output-based error quantification and localization can provide more confidence in output values computed from chaotic simulations and increase the efficiency of meshes through adaptation. Such adaptive capability would therefore make LES simulations more practical for analysis and design.

Chaotic systems have many features at various scales that are difficult to resolve numerically. This makes methods that can accurately pick out important areas for resolution crucial for efficiency. Several methods already exist for the estimation of numerical errors in outputs for general steady and unsteady flows.[1] Of the numerous techniques that have been investigated in the past, output-based methods have proven to be robust and accurate, making them a desirable technique to use to quantify errors associated with chaotic flows.

Output-based methods have already been demonstrated to improve solution accuracy and decrease computational costs of deterministic unsteady systems. However, these techniques, which rely on the solution of a linear adjoint problem, have been shown to be unsuccessful when applied to chaotic systems. The non-deterministic behavior of chaotic flows results in linearly-unstable adjoint systems that cannot be solved via standard procedures. There is therefore a new challenge to discover methods of calculating output-based error estimates and adaptive indicators for chaotic flows. In previous works, LSS[2] has been introduced as a stable and accurate method for calculating output sensitivities and the requisite adjoint solutions. However,

[*]Graduate Research Assistant, AIAA Member
[†]Associate Professor, AIAA Senior Member

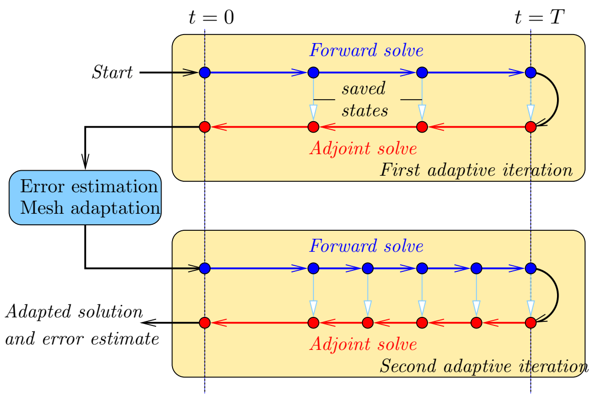
the success of LSS has been mitigated by the additional conclusion that LSS utilizes computationally expensive routines in its optimization process. This paper shows that it is possible to calculate adjoint-based error estimates for output-based methods that are sufficiently accurate, robust, and cheap through the use of reduced-order models.

In the following section, we briefly review output-based error estimation. Next, we describe aspects of chaotic systems that make these problems less amenable to standard error estimation techniques. We introduce the Kuramoto-Sivashinsky (KS) equations, a prototypical chaotic partial differential equation (PDE) that is used for error estimation, and describe the method used to discretize this PDE. Next, we briefly go over the standard LSS technique and then explain our chosen class of model reduction techniques. In order to further reduce the computational cost, we present an HROM procedure based on the Gauss-Newton method with approximated tensors (GNAT). Finally, we introduce HROM-LSS and explain how the hyper-reduced-order states solutions are used to calculate adjoints for error estimation. Results for ROM, HROM, and HROM-LSS are presented for the KS equation.

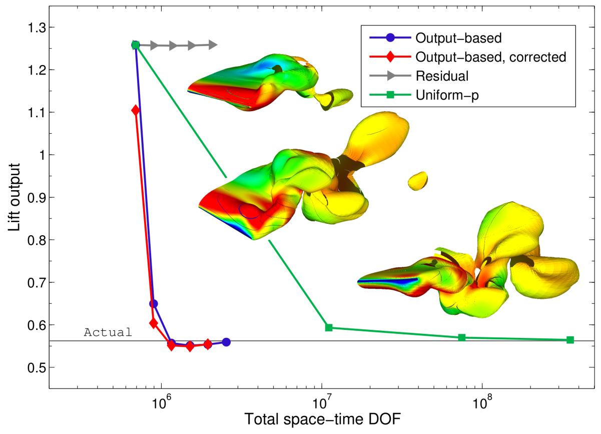
II. Output-Based Error Estimation

Output-based adaptive methods have been shown to be successful for steady problems;[1,3–7] however, their application to unsteady simulations has been limited[8–14] due to challenges in implementation and computational cost associated with the fine-space adjoint solutions. Success of these methods for unsteady problems include: temporal-only error estimation and adaptation;[8,9] spatial-only error estimation and adaptation;[10,15,16] combined temporal and spatial mesh refinement with a static geometry and mesh;[12,14,17] combined temporal and dynamic spatial refinement on static geometries;[13,18,19] combined temporal and dynamic-order spatial refinement on deformable domains.[20–22] In our previous work we have used space-time discontinuous Galerkin (DG) and hybridized discontinuous Galerkin (HDG)[23–28] finite element discretizations using time slabs and an approximate space-time solver.[12,29,30]

Output-based methods use adjoint solutions, which are calculated for unsteady problems by reverse time-integration and linearization about the primal solution. Details of how the adjoint calculation is implemented in output-based methods are shown in Figure 1. However, the usual unsteady adjoint calculation fails to produce useful adjoints for chaotic flows, due to the high sensitivity of chaotic problems to initial conditions. It is crucial to work on improving adjoint calculations for unsteady chaotic systems, or to seek alternative methods, in order to take advantage of the accuracy and robustness improvements of output-based error methods. By way of motivation, Figure 1 shows how output-based methods produce better output convergence results than that of residual-based or uniform refinement techniques for deterministic problems. The goal of the present work is to obtain similar improvements for chaotic problems.



(a) Unsteady adaptive iterations



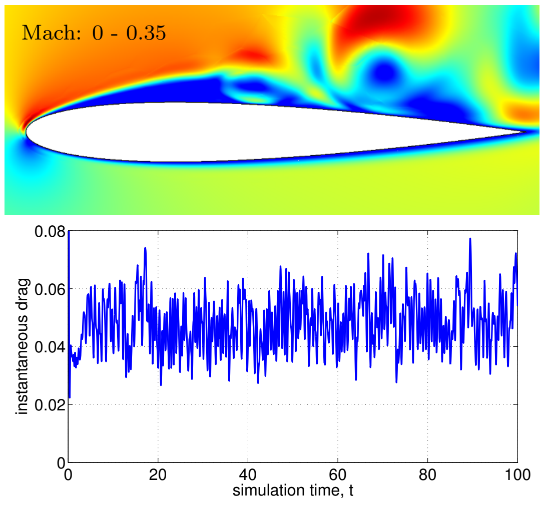
(b) Representative output-based solution

Figure 1. Schematic of an adaptive primal and adjoint solution procedure for output-based unsteady simulations and output convergence result for a representative flapping-wing Navier-Stokes simulation on a deforming domain.

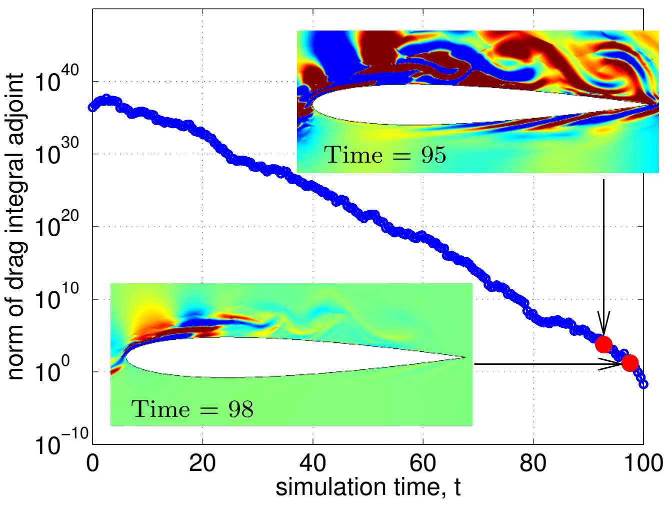
III. Chaotic Systems

For high Reynolds numbers and long simulation times, outputs of Navier-Stokes simulations can become chaotic, making the application of unsteady output-based error estimates challenging due to the failure of adjoint calculations. The chaotic nature is seen in the adjoint solution, which increases in magnitude exponentially when solved backward in time, as shown in Figure 2. This figure shows a Navier-Stokes simulation of a NACA 0012 at moderate Reynolds number. The adjoint field deteriorates quickly and is not usable for error estimation due to the high sensitivity of chaotic systems to perturbations in the initial conditions. Instead of looking at the instantaneous output, a more practical approach is to study statistical outputs in order to take advantage of the ergodic characteristic of the system, which is defined as the lack of influence of the initial conditions on the long term output. This type of behavior can also be seen when considering perturbations in parameter values, investigated in detail by Wang.[31] Just as outputs of deterministic simulations, statistical outputs are also polluted by discretization errors[32,33] due to insufficient spatial or temporal mesh resolution.

One aspect of chaotic systems of interest in the current work is the sensitivity of statistical outputs to changes in the discretization, such as the approximation order. We have studied this in our previous work for the Lorenz attractor,[34] and we found that the output statistics of the Lorenz attractor improve as the simulation time (T) increases; however, there exists a persistent discrepancy among different temporal accuracy orders resulting in the simulations converging to different statistical time-averaged output values. We note that an accurate distinction between temporal and spatial discretization errors is important for efficiency and convergence of adaptation.



(a) Primal snapshot and output history



(b) Drag integral adjoint history and snapshots

Figure 2. NACA 0012, $M = 0.2$, $Re = 10000$, $\alpha = 8°$: ill-conditioning of average-drag prediction manifests itself through an unstable adjoint; i.e. the output is highly-sensitive to initial conditions.

IV. 1D Kuramoto-Sivashinsky Equation

Instead of using the Navier-Stokes equations for the prototype equation in this paper, we use the simpler Kuramoto-Sivashinsky equation (KS), which was derived to model the Belousov-Zabotinskii reactions in three space dimensions.[35] In addition, KS was derived to model small thermal diffusive instabilities in laminar flame fronts by 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. For this paper, we have chosen to solve KS using Dirichlet boundary conditions.[31] The KS equation for a primal scalar state $u(x, t)$ reads

$$\frac{\partial u}{\partial t} = -\alpha\frac{\partial u}{\partial x} - u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \beta\frac{\partial^4 u}{\partial x^4}, \tag{1}$$

where the role of the advection term is to ensure that the KS solution exhibits ergodic behavior. The initial condition consists of a delta distribution,

$$u(x, t = 0) = u_0(x), \tag{2}$$

$$u_0(x) = \begin{cases} 1, & x = \frac{X_f}{2} \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

and a burn time is used to verify that the true initial conditions that we use for the simulations are actually on the attractor. The boundary conditions are

$$u(x = X_0, t) = u(x = X_f, t) = \frac{\partial u}{\partial x}(x = X_0, t) = \frac{\partial u}{\partial x}(x = X_f, t) = 0. \tag{4}$$

The output of interest is a spatial and temporal average of the state,

$$\overline{J} = \frac{1}{TX} \int_{T_0}^{T_f} \int_{X_0}^{X_f} u(x,t)\, dx\, dt, \tag{5}$$

where $T = T_f - T_0$ and $X_f - X_0$. This integral is evaluated using quadrature in space and trapezoidal time integration.

The trajectories of KS exhibit temporal chaos and can develop strange attractors with positive Lyapounov exponents, which is the driving force behind the reason why unsteady calculations of adjoints fail for chaotic systems. The burning procedure ensures that the time-averaged statistics are not heavily affected by the initial conditions of the system. In the results section, a study of how the α and β coefficients affect the trajectories is shown and used to determine how long the burn time should be for each set of parameters. These results inform later studies into least-squares shadowing (LSS), and reduced-order modeling (ROM) coupled with LSS. The next section covers the discretization of Eqn. 1.

V. Discretization

The spatially-discretized version of Eqn. 1 reads

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{M}\frac{d\boldsymbol{u}}{dt} + \boldsymbol{R}_s(\boldsymbol{u}) = \boldsymbol{0} \tag{6}$$

where $\boldsymbol{u} \in \mathbb{R}^N$ is the discretized primal state vector, $\boldsymbol{R}_s \in \mathbb{R}^N$ is the spatial residual, $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ is the mass matrix, and $\boldsymbol{R} \in \mathbb{R}^N$ is the total unsteady residual driven to zero. Here, N denotes the number of degrees of freedom in the high-fidelity (full-order) state. The primal solution $u(x,t)$ is approximated using finite elements with discontinuous basis/test functions,

$$u(x,t) \approx \sum_{k=1}^{N_e} \sum_{j=1}^{p+1} u_{k,j}(t)\phi_{k,j}(x), \tag{7}$$

where N_e is the number of elements and p is the order of spatial approximation for each element. The solution is mapped from reference to global space, where the reference element in 1D extends from -1 to 1. The discrete solution is solved using a discontinuous Galerkin finite-element method. For the KS equations, the numerical flux is most easily defined separately for each part of the PDE. The discretization of the advection and nonlinear terms is done using a standard Godunov flux. The discretization of the diffusion term is calculated using the interior penalty (IP) method. The discretization of the fourth-order diffusion term is more complicated and is done using a modified interior penalty method, as presented in Georgoulis et. al.[36] Second-order backwards differencing is used to discretize the system in time, and Newton's method is used to solve the implicit system of equations at each time step.

VI. Least Squares Shadowing

LSS has been used successfully to compute sensitivities for chaotic systems. LSS implementations have been categorized into three different versions.[2, 31, 37, 38] In this paper, we use LSS Type III checkpoint design to calculate adjoints. In this section, we briefly review the implementation procedure for LSS and its extension to output-based error estimation. Given a discrete dynamical system,

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}, s), \tag{8}$$

with s as a parameter, the tangent LSS solution $\boldsymbol{u}(t)$ is defined by the minimization statement,

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \frac{1}{2} \int_{T_0}^{T_f} \|\boldsymbol{v}\|^2 dt, \quad \text{s.t.} \quad \frac{d\boldsymbol{v}}{dt} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \boldsymbol{f}}{\partial s} + \eta \boldsymbol{f}, \tag{9}$$

where $\eta = \frac{d\tau}{dt} - 1$ is a time-dilation term. The shadow trajectory is $\boldsymbol{u}(t) + \boldsymbol{v}(t)\delta s$. The Lagrangian is formed from this optimization problem as

$$\mathcal{L}_{\text{LSS}} = \int_{T_0}^{T_f} \left[\frac{1}{2}\|\boldsymbol{v}\|^2 + \boldsymbol{w}^\top \left(\frac{\partial \boldsymbol{v}}{\partial t} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} - \frac{\partial \boldsymbol{f}}{\partial s} - \eta \boldsymbol{f} \right) \right] dt. \tag{10}$$

The equations that define the Lagrangian as stationary with respect to variations in $\boldsymbol{w}$ and $\boldsymbol{v}$ give the tangent and the Lagragian equations or the Karush-Kuhn-Tucker conditions,

$$\frac{d\boldsymbol{v}}{dt} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \boldsymbol{f}}{\partial s} + \boldsymbol{f}(\boldsymbol{u}, s)\eta \tag{11}$$

$$\frac{d\boldsymbol{w}}{dt} = -\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}^\top \boldsymbol{w} + \boldsymbol{v} \tag{12}$$

Given the Lagragnge adjoint function

$$\mathcal{L}_{\text{adjoint}} = \bar{J}' - \int_{T_0}^{T_f} \boldsymbol{\psi}^\top \boldsymbol{R}(\boldsymbol{v}, \boldsymbol{w})\, dt \tag{13}$$

and the KKT conditions, the adjoint equations to solve are

$$\frac{d\boldsymbol{\psi}_2}{dt} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \right) \boldsymbol{\psi}_2, \tag{14}$$

$$\frac{d\boldsymbol{\psi}_1}{dt} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \right)^\top \boldsymbol{\psi}_1 - \boldsymbol{\psi}_2 - \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}}^\top. \tag{15}$$

One way to solve these adjoint equations is by using an iterative checkpoint process outlined in LSS Type III,[2] which requires only solving for the tangent and the adjoint on short time segments. $\boldsymbol{\psi}_2$ refers to the tangent solution and $\boldsymbol{\psi}_1$ refers to the adjoint solution. The iterative solver first requires the primal solution for all time segments. Once this is found, 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 done for all time segments and repeated until the tangent and adjoint solution converges via an iterative matrix-free solver, GMRES, to a prescribed tolerance.

VII. Reduced-Order Modeling

Reduced-order modeling (ROM) works well when the state solution can be approximated accurately as a member of an affine subspace whose dimension is significantly smaller than that of the primal solution.[39] The hypothesis is that by finding a ROM for a chaotic system whose size is significantly smaller than that of the original primal solution, we can reduce the overall computational costs associated with LSS. Before using a ROM for LSS, we need to verify that an accurate ROM can indeed be found for chaotic systems.

When calculating the reduced solution, we need to capture the information associated with the affine subspace of the state as best as possible. The approximate solution that is found in this subspace for time node i is written as

$$\boldsymbol{u}_i \approx \Phi \boldsymbol{u}_{r_i}, \tag{16}$$

where $\Phi \in \mathbb{R}^{N \times n_r}$ is a matrix whose columns store the discrete reduced-order basis functions that describes the subspace. The reduced solution is referred here as $\boldsymbol{u}_r \in \mathbb{R}^{n_r}$. Before going further into the ROM technique, we describe how the reduced-order basis matrix is calculated using the proper orthogonal decomposition (POD) method.[39]

VII.A. Calculation of Φ using Proper Orthogonal Decomposition (POD)

When calculating the POD of a snapshot matrix, 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 (columns) in the snapshot matrix, $\boldsymbol{S} \in \mathbb{R}^{N \times n_s}$ and $n_r \leq n_s$ is the number of reduced-order basis functions, i.e. columns in the reduced-order basis matrix, Φ.

We begin by generating the snapshot matrix, $\boldsymbol{S}$, which consists of n_s primal solutions. Some experimentation is required to determine which pre-computed primal solutions to use in the snapshot matrix. Typically these primal solutions are taken from every few time steps in an unsteady calculation. We note that a poor set of snapshots will lead to an ill-conditioned matrix. $\boldsymbol{S}$ takes the form

$$\boldsymbol{S} = [\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_{n_s}] \in \mathbb{R}^{N \times n_s}, \tag{17}$$

where $\boldsymbol{u}_i$ is the primal state snapshots at time t_i.

In POD, we perform a singular-value decomposition of the snapshot matrix, $\boldsymbol{S} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$, and then we construct the reduced-order basis matrix from the first n_r columns of $\boldsymbol{U}$,

$$\Phi = \boldsymbol{U}(:, 1 : n_r) \in \mathbb{R}^{N \times n_r}. \tag{18}$$

Again, experimentation is required to determine the optimal value of n_r in order to obtain an economical and accurate ROM.

VII.B. Solving for the Reduced State

There are two methods for solving for the reduced state: orthogonal projection and minimization of the residual. Both procedures result in a least-squares Petrov-Galerkin projection and use variations of Newton solves. We use Algorithm 1 to solve for the reduced states using the Gauss-Newton solver.

VII.B.1. Orthogonal Projection with Newton's Method

After projecting the solution to a smaller affine space to find Φ, we follow the approach taken by Amsallem et. al. to find the reduced solution of the nonlinear problem. This approach uses the Gauss-Newton method based on a variation of Newton's method.[39]

We constrain the residual seen in Eqn. 6 to be orthogonal to test functions, which are columns of $\Psi \in \mathbb{R}^{N \times n_r}$, giving

$$\boldsymbol{\Gamma} \equiv \boldsymbol{\Psi}^\top \boldsymbol{R}(\boldsymbol{u}(\boldsymbol{u}_r)) = \boldsymbol{\Psi}^T \left[\boldsymbol{M}\frac{d\boldsymbol{u}}{dt} + \boldsymbol{R}_s(\boldsymbol{u})\right] = \boldsymbol{0}. \tag{19}$$

The least-squares Petrov-Galerkin (LSPG) projection, in which $\Psi = \boldsymbol{J}\Phi$ and $\boldsymbol{J}(\boldsymbol{u}) = \frac{\partial \boldsymbol{R}(\boldsymbol{u})}{\partial \boldsymbol{u}}$, is chosen over the Galerkin projection, $\Psi = \Phi$, because LSPG is more accurate at capturing non-linear effects and is more stable for unsteady non-linear model reduction performed at the discrete level.[40]

The resulting equation for the state update, $\boldsymbol{p}$ is

$$\boldsymbol{p} = \boldsymbol{u}_t - \boldsymbol{u}_{t-1} = -\left[\nabla\boldsymbol{\Gamma}(\boldsymbol{u}_{t-1})\right]^{-1} \boldsymbol{\Gamma}(\boldsymbol{u}_{t-1}). \tag{20}$$

Setting $\Psi = \boldsymbol{J}\Phi$ and simplyfing Eqn. 20 gives

$$\Phi^\top \boldsymbol{J}^\top \boldsymbol{J}\Phi\boldsymbol{p} = -\Phi^\top \boldsymbol{J}^\top \boldsymbol{R}, \tag{21}$$

To avoid computing $\boldsymbol{J}^T\boldsymbol{J}$, we note that Eqn. 21 is the normal equation form of the least-squares problem

$$\boldsymbol{p} = \arg\min_{a\in\mathbb{R}^{n_r}} \|\boldsymbol{J}\Phi a + \boldsymbol{R}\|_2, \tag{22}$$

which can be solved using QR factorization ($\boldsymbol{J}\Phi = \boldsymbol{Q}_{\boldsymbol{J}\Phi}\boldsymbol{R}_{\boldsymbol{J}\Phi}$). We solve

$$\boldsymbol{R}_{\boldsymbol{J}\Phi}\boldsymbol{p} = -\boldsymbol{Q}_{\boldsymbol{J}\Phi}^{\top}\boldsymbol{R} \tag{23}$$

to find the direction of the update $\boldsymbol{p}$. At Gauss-Newton iteration k, the update for the reduced solution is

$$\boldsymbol{u}_r^{k+1} = \boldsymbol{u}_r^k + \alpha^k\boldsymbol{p}_i^k \tag{24}$$

$$\boldsymbol{u}^{k+1} = \Phi\boldsymbol{u}_r^{k+1} \tag{25}$$

where α is found from a line search. Eqns. 22–24 are part of an iterative process that terminates when the update $\boldsymbol{p}$ convergences to a certain tolerance established by the user for each time node.

VII.B.2. Solving the Minimization Problem using Newton's Method

An alternate approach to solving for the reduced solution uses a different form of the Gauss-Newton method. We define the minimum residual and the search direction as

$$\boldsymbol{\Gamma} = \min\|\boldsymbol{R}(\boldsymbol{u}(\boldsymbol{u}_r))\|_2 \tag{26}$$

$$\boldsymbol{p} = \boldsymbol{u}_t - \boldsymbol{u}_{t-1} = -[\nabla^2\boldsymbol{\Gamma}(\boldsymbol{u}_{t-1})]^{-1}\nabla\boldsymbol{\Gamma}(\boldsymbol{u}_{t-1}), \tag{27}$$

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

$$\boldsymbol{J}\Phi\boldsymbol{p} = -\boldsymbol{R}. \tag{28}$$

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

$$\boldsymbol{R}_{\boldsymbol{J}\Phi}\boldsymbol{p} = -\boldsymbol{Q}_{\boldsymbol{J}\Phi}^{\top}\boldsymbol{R}. \tag{29}$$

Note that this equation is the same as Eqn. 23. We once again perform a line-search procedure, find the direction magnitude α, and update the reduced solution. Once again, k refers to the the current Gauss-Newton iteration, respectively. Eqns. 27–29 are part of an iterative process that is repeated until $\boldsymbol{p}^k$ convergences to a certain tolerance for each time node. The online calculation of the Gauss-Newton method for ROM can be found in Algorithm 1.

Algorithm 1 Reduced-Order Modeling

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

VIII. Hyper Reduced Order Modeling

The previous section presents a projection of the primal solution to a smaller space of dimension n_r. However, the online solution of the reduced model still requires working with vectors that are of the same size as the primal system, N. It is necessary to reduce the size of this system in order to truly lower the computational cost associated with using the reduced-order model. This approach for alleviating this computational cost is referred to as hyper-reduction, which produces a hyper-reduced-order model (HROM). One version of such an approach is Gauss-Newton with approximated tensor quantities (GNAT), which introduces additional steps and approximations into the model reduction procedure.[39] For this section, we use GNAT to approximate the residuals and Jacobians by projecting them on reduced-order basis matrices, Φ_R and Φ_J. We then further expand and manipulate the orthogonal projection method from the previous section to obtain the HROM. The basic principles of HROM starts with the approximate solution defined in Eqn. 16. We reduce the computational cost by reducing the number of states to be calculated,

$$\overline{\boldsymbol{u}} \approx \overline{\Phi}\boldsymbol{u}_r \tag{30}$$

$$\overline{\boldsymbol{u}} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n_j} \end{bmatrix} \quad \overline{\Phi} = \begin{bmatrix} \phi_{0,0} & \phi_{0,1} & \cdots & \phi_{0,n_r} \\ \phi_{1,0} & \phi_{1,1} & \cdots & \phi_{1,n_r} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n_j,0} & \phi_{0,0} & \cdots & \phi_{n_j,n_r} \end{bmatrix}, \tag{31}$$

where $n_j \leq N$ is the total number of restricted states and refers to each of the specific state vector indices required to reconstruct and approximate the residual and Jacobian accurately. Another parameter of importance is the number of sample nodes n_i, which is the total number of nodes at which to calculate the residual and the Jacobian. n_i is determined via a Greedy algorithm, which indirectly determines n_j. 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 of the $N - n_i$ 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.

VIII.A. Greedy Algorithm for Computing Sample Nodes

Once the states have been defined mathematically, we need to find an efficient method to determine at which entries (nodes) to calculate quantities such as the residual and the Jacobian. In total there are n_i of these nodes. By not sampling at all nodes, we can further reduce the computational cost of calculating a reduced-order model. Note that the number of sample nodes required for the residual and the Jacobian is different from 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 compact. 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 to find the greedy version of any vector and of any matrix. An example of a greedy residual and a Jacobian is

$$\hat{\boldsymbol{R}} = \begin{bmatrix} R_0 \\ \vdots \\ R_{n_i} \end{bmatrix}, \quad \hat{\boldsymbol{J}} = \begin{bmatrix} J_{0,0} & \cdots & J_{0,n_j} \\ \vdots & \ddots & \vdots \\ J_{n_i,0} & \cdots & J_{n_i,n_j}. \end{bmatrix}. \tag{32}$$

The greedy algorithm 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. $n_j \geq n_i$. A thorough code for this algorithm can be found at Carlberg et. al.[41]

VIII.B. 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),the data needs to be reconstructed as a result of the missing values or "gaps". This reconstruction process is called the gappy data reconstruction, which was first introduced for image reconstruction.[42] In particular, we need 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.[40] The gappy data reconstruction method computes an approximation of a desired quantity as

$$\tilde{\mathbf{\Lambda}} = \mathbf{\Lambda} \quad \tilde{\mathbf{\Lambda}} = \Phi_{\mathbf{\Lambda}} \mathbf{\Lambda}_g, \tag{33}$$

where

$$\mathbf{\Lambda}_g = \arg \min_{a \in \mathbb{R}^{n_{\mathbf{\Lambda}}}} \|\widehat{\mathbf{\Phi}}_{\mathbf{\Lambda}} a - \widehat{\mathbf{\Lambda}}\|_2 \tag{34}$$

The solution to Eqn. 34 is

$$\mathbf{\Lambda}_g = \widehat{\mathbf{\Phi}}^+_{\mathbf{\Lambda}} \widehat{\mathbf{\Lambda}}. \tag{35}$$

where $(\cdot)^+$ refers to the Moore-Penrose pseudo-inverse of a matrix. Before applying the method to approximate the residuals and Jacobians, we say that the non-linear residual and Jacobian is exactly approximated as

$$\widetilde{\boldsymbol{R}} = \boldsymbol{R}, \quad \widetilde{\boldsymbol{J}\Phi} = \boldsymbol{J}\Phi, \tag{36}$$

allowing the use of the gappy reconstruction method to approximate these nonlinear terms as

$$\widetilde{\boldsymbol{R}} \approx \Phi_R \boldsymbol{R}_g, \tag{37}$$

$$\widetilde{\boldsymbol{J}\Phi} \approx \Phi_J \boldsymbol{J}_g, \tag{38}$$

where Φ_R and Φ_J are new reduced-order basis function matrices for the residual and Jacobian, which consist of accurately approximating the full residual and Jacobian values given the sample node solutions. The overall new approximations for these values are,

$$\widetilde{\boldsymbol{R}} \approx \Phi_R \widehat{\Phi}^+_R \widehat{\boldsymbol{R}} \tag{39}$$

$$\widetilde{\boldsymbol{J}\Phi} \approx \Phi_J \widehat{\Phi}^+_J \widehat{\boldsymbol{J}\Phi}. \tag{40}$$

VIII.C. Calculation of Φ_R and Φ_J using POD

Unlike the calculation of Φ, there is only one critical parameter to set for the calculation of the residual-reduced-order basis matrix Φ_R and the Jacobian-reduced-order basis matrix Φ_J: the number of reduced-order basis functions (columns) n_{RJ}, which is different than n_s. The calculation of Φ_R and Φ_J begins with the generation of the residual and the Jacobian snapshot metrices, $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$. To find the state snapshots, the user chooses which states at a particular time t to add; however, this is not the case for Φ_R and Φ_J. Instead, the snapshot matrix is generated during the ROM stage where, during each Gauss-Newton iteration, the information on 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, n_{ST}. Due to large number of Gauss-Newton iterations, the residual and Jacobian data is saved to an a file on disk. $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$ take the form,

$$\boldsymbol{S}_R = [\boldsymbol{R}_0,\ \boldsymbol{R}_1,\ \ldots,\ \boldsymbol{R}_{n_{ST-1}}] \in \mathbb{R}^{N \times n_{ST-1}}, \tag{41}$$

$$\boldsymbol{S}_J = [\boldsymbol{J}_0 \Phi \boldsymbol{p}_0,\ \boldsymbol{J}_1 \Phi \boldsymbol{p}_1,\ \ldots,\ \boldsymbol{J}_{n_{ST-1}} \Phi \boldsymbol{p}_{n_{ST-1}}] \in \mathbb{R}^{N \times n_{ST-1}}. \tag{42}$$

This and other ways to establish the snapshot matrices $\boldsymbol{S}_R$ and $\boldsymbol{S}_J$ can be found in Carlberg et. al.[40] Once the snapshot matrices are defined, POD is performed on these matrices. After performing singular value decompositions of the snapshot matrices, $\boldsymbol{S}_{\boldsymbol{R}=\boldsymbol{U}_{\boldsymbol{R}}\Sigma \boldsymbol{V}_{\boldsymbol{R}}{}^\top}$ an $\boldsymbol{S}_{\boldsymbol{J}=\boldsymbol{U}_{\boldsymbol{J}}\Sigma \boldsymbol{V}_{\boldsymbol{J}}{}^\top}$, the residual-reduced-order basis matrix and the Jacobian-reduced-order basis matrix are chosen as

$$\mathbf{\Phi}_R = \boldsymbol{U}_{\boldsymbol{R}}(:, 1 : n_{RJ}) \quad \mathbf{\Phi}_J = \boldsymbol{U}_{\boldsymbol{J}}(:, 1 : n_{RJ}). \tag{43}$$

VIII.D. Modified Least-Squares Petrov-Galerkin Projection for HROM

HROM requires some 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, we create offline matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ that 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. 37 and Eqn. 38 into Eqn. 22 to get a modified minimization problem,

$$\boldsymbol{p} = \arg\min_{\boldsymbol{a}\in\mathbb{R}^{n_r}} \|\Phi_J \widehat{\Phi}_J^+ \widehat{\boldsymbol{J}\Phi}\boldsymbol{a} + \Phi_R \widehat{\Phi}_R^+ \widehat{\boldsymbol{R}}\|_2. \tag{44}$$

We define $\boldsymbol{C}$ as

$$\boldsymbol{C} = \Phi_J \widehat{\Phi}_J^+ \widehat{\boldsymbol{J}\Phi} \tag{45}$$

and then perform a QR factorization

$$\boldsymbol{C} = \boldsymbol{Q}_c \boldsymbol{R}_c, \quad \boldsymbol{A} = \boldsymbol{R}_c, \quad \boldsymbol{B} = \boldsymbol{Q}_c^\top \Phi_R \widehat{\Phi}_R^+. \tag{46}$$

We rearrange Eqn. 44 to obtain

$$\boldsymbol{p} = \arg\min_{\boldsymbol{a}\in\mathbb{R}^{n_r}} \|\boldsymbol{A}\widehat{\Phi}_J \boldsymbol{a} + \boldsymbol{B}\widehat{\boldsymbol{R}}\|_2. \tag{47}$$

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

$$\boldsymbol{R}_{A\hat{J}\Phi}\boldsymbol{p} = -\boldsymbol{Q}_{A\widehat{J\Phi}}^\top \boldsymbol{B}\widehat{\boldsymbol{R}} \tag{48}$$

Once again, we perform a line-search procedure to find the update magnitude α and then update the reduced state via

$$\overline{\boldsymbol{u}}_r^{k+1} = \overline{\boldsymbol{u}}_r^k + \alpha^k \boldsymbol{p}^k, \tag{49}$$

$$\boldsymbol{u}^{k+1} = \overline{\Phi}\boldsymbol{u}_r^{k+1}. \tag{50}$$

Here, k refers to the Gauss-Newton iteration number. The online calculation of the Gauss-Newton method for HROM can be found in Algorithm 2, which is a modified version of Algorithm 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 approximated substitutions are made, the iteration process can proceed to find the HROM model of KS, which then can also be used for LSS.

Algorithm 2 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 $t = 1, \dots n_T - 1$ 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 $\boldsymbol{R}_E \boldsymbol{p}_t^k = -\boldsymbol{Q}_E^\top \boldsymbol{D}_t^k$
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:     $\overline{\boldsymbol{u}}_t = \overline{\boldsymbol{u}}_t^{k+1}$
17:     $\boldsymbol{u}_{r,t} = \boldsymbol{u}_{r,t}^{k+1}$
18: end for
```

IX. Hyper-Reduced-Order Model-Least Squares Shadowing

To apply adjoint-based error estimation to HROM-LSS, the original LSS equations need to be converted into the reduced form. In ROM, the approximate solution for time t is calculated from the reduced solution u_r via Eqn. 16. This relationship is used to derive the reduced tangent ($\boldsymbol{\psi}_2$) and reduced Lagrange ($\boldsymbol{\psi}_1$) equations. We first derive these reduced adjoint equations, starting from Eqn. 8. Substituting Eqn. 16 into Eqn. 8 and performing a projection using test functions Ψ,

$$\Psi^\top \left[\frac{d(\Phi \boldsymbol{u}_r)}{dt} = \boldsymbol{f}\left(\Phi \boldsymbol{u}_r\right) \right], \tag{51}$$

we obtain the reduced problem

$$\frac{d\boldsymbol{u}_r}{dt} = \boldsymbol{f}_r, \tag{52}$$

where

$$\boldsymbol{f}_r = \left[\Psi^\top \Phi\right]^{-1} \Psi^\top \boldsymbol{f}\left(\Phi \boldsymbol{u}_r\right). \tag{53}$$

Using this system, we can rewrite the LSS equations as

$$\frac{d\boldsymbol{\psi}_{2,r}}{dt} = \left(\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}\right) \boldsymbol{\psi}_{2,r}, \tag{54}$$

$$\frac{d\boldsymbol{\psi}_{1,r}}{dt} = -\left(\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}\right)^T \boldsymbol{\psi}_{1,r} - \boldsymbol{\psi}_{2,r} - \frac{1}{T}\frac{\partial J}{\partial \boldsymbol{u}_r}^\top. \tag{55}$$

$\boldsymbol{\psi}_{2,r}$ refers to the reduced tangent solution and $\boldsymbol{\phi}_{1,r}$ refers to the reduced adjoint solution. The goal is to write the tangent equation and Lagrange equation in terms of known quantities. We begin with Eqn. 53. Calculating $\boldsymbol{f}(\Phi \boldsymbol{u}_r)$ is expensive and can be approximated using the gappy reconstruction procedure from GNAT,

$$\widetilde{\boldsymbol{f}} = \boldsymbol{f}, \tag{56}$$

$$\widetilde{\boldsymbol{f}} \approx \Phi_f \widehat{\Phi}_f^+ \widehat{\boldsymbol{f}}, \tag{57}$$

where

$$\widehat{\boldsymbol{f}} = -\widehat{\boldsymbol{M}^{-1}\boldsymbol{R}}_s. \tag{58}$$

We set $\Phi_f = \Phi_R$, which we already have from HROM, to obtain the final form of $\boldsymbol{f}_r$ used in HROM-LSS,

$$\boldsymbol{f}_r = \boldsymbol{A}\widehat{\boldsymbol{M}^{-1}\boldsymbol{R}} \quad \boldsymbol{A} = -\left[\Psi^\top \Phi\right]^{-1} \Psi^\top \Phi_R \widehat{\Phi}_R^+. \tag{59}$$

Taking the derivative of Eqn. 53, we find

$$\frac{\partial f_r}{\partial \boldsymbol{u}_r} = \left[\Psi^\top \Phi\right]^{-1} \Psi^\top \frac{\partial \boldsymbol{f}\left(\Phi \boldsymbol{u}_r\right)}{\partial \boldsymbol{u}_r}. \tag{60}$$

A similar procedure is used to find $\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}$ using a gappy algorithm.

$$\widetilde{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_r}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_r}, \tag{61}$$

$$\widetilde{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_r}} \approx \Phi_{\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}} \widehat{\Phi}^+_{\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}} \widehat{\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}}, \tag{62}$$

where

$$\widehat{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_r}} = -\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}}\Phi. \tag{63}$$

Setting $\widehat{\Phi}_{\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}} = \Phi_J$, which we already have from HROM to get the final form of $\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r}$ used in HROM-LSS,

$$\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{u}_r} = \boldsymbol{B}\widehat{\boldsymbol{M}^{-1}\boldsymbol{J}}\Phi \quad \boldsymbol{B} = -\left[\Psi^\top \Phi\right]^{-1} \Psi^\top \Phi_J \widehat{\Phi}_J^+. \tag{64}$$

For the HROM-LSS equations, we use a Galerkin projection, $\Psi = \Phi$ and

$$\frac{\partial J}{\partial \boldsymbol{u}_r}^\top = \left(\frac{\partial J}{\partial \boldsymbol{u}}\Phi\right)^\top. \tag{65}$$

Substituting Eqn. 64 into the reduced LSS equations gives the final modified LSS equations,

$$\frac{d\boldsymbol{\psi}_{2,r}}{dt} = \left(\widehat{\boldsymbol{B}\boldsymbol{M}^{-1}\boldsymbol{J}}\Phi\right)\boldsymbol{\psi}_{2,r}, \tag{66}$$

$$\frac{d\boldsymbol{\psi}_{1,r}}{dt} = -\left(\widehat{\boldsymbol{B}\boldsymbol{M}^{-1}\boldsymbol{J}}\Phi\right)^T\boldsymbol{\psi}_{1,r} - \boldsymbol{\psi}_{2,r} - \frac{1}{T}\Phi^\top\frac{\partial J}{\partial \boldsymbol{u}}^\top, \tag{67}$$

We solve these modified HROM-LSS equations with the same iterative checkpoint process outlined in the LSS Type III algorithm.[2] Similarly as in LSS, a guess for the reduced tangent solution and reduced adjoint solution for each check point is made. Once again, GMRES is used to find the desired reduced adjoint solution.

IX.A. Output Error Estimation via the Adjoint-Weighted Residual

After solving the reduced-adjoint LSS equations via the LSS Type III algorithm, output sensitivies can be calcuated from $\boldsymbol{\psi}_1$, since this is the adjoint that weights the residual term with $\frac{\partial \boldsymbol{f}}{\partial s}$. Specifically,

$$\frac{\partial \bar{J}}{\partial s} = \int_0^T \boldsymbol{\psi}_{1_r}^\top \frac{\partial \boldsymbol{f}_r}{\partial s}dt. \tag{68}$$

To estimate the output error using the LSS adjoint, we apply Eqn. 68, using a residual perturbation computed from two different discretization spaces: a coarse one with temporal order r_H, and a fine one with temporal order $r_h = r_H + 1$. We solve the primal problem with order r_H and then inject the solution into the finer space. Doing so gives us a perturbation in the residual, which we weight by the fine-space adjoint to obtain the error estimate:

$$\delta\bar{J} = -\int_{T_0}^{T_f} \boldsymbol{\psi}_{1,h}^T\left[\frac{d\boldsymbol{u}_{r,H}}{dt} - \boldsymbol{f}_r(\boldsymbol{u}_{r,H})\right]_h dt = \boldsymbol{\Psi}_{1,h}^T\left[\Phi^\top\Phi_R\widehat{\Phi}_R^+\widehat{\boldsymbol{R}}_h\left(\overline{\boldsymbol{u}}_H\right)\right], \tag{69}$$

where $\boldsymbol{\Psi}_h$ is the vector of all of the $\boldsymbol{\psi}_h$ unknowns and $\boldsymbol{R}_h(\overline{\boldsymbol{u}}_H)$ is the order r_h residual vector evaluated with the order r_H injected solution.

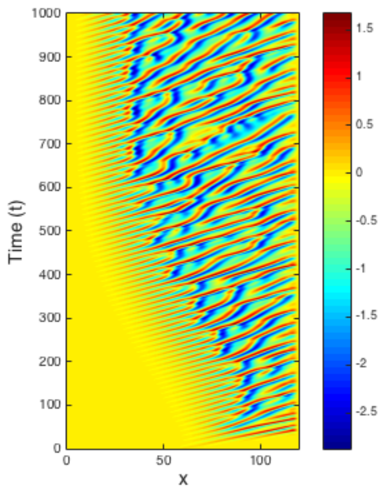
X. Results

X.A. Effect of Kuramoto-Sivashinsky Parameters on Trajectories and Burn Time

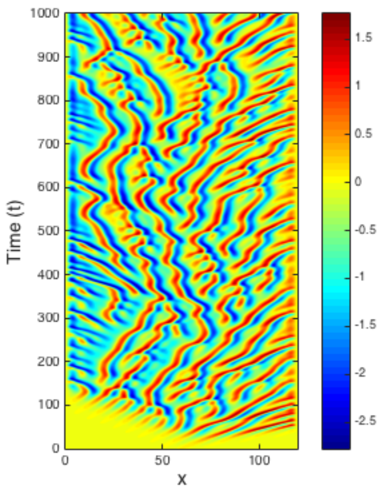
KS exhibits different trajectories depending on the advection speed and the "super viscosity", (α, β), which can be seen in $x-t$ contour plots in Figure 3 for six different cases with zero burn time. These results provide some preliminary insight into which trajectory would make an adequate prototype for LSS and HROM-LSS. In addition, these results indicate what the burn time (t_{burn}) should be for each set of parameters. The burn time should be large 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 Figure 3. The time-averaged output $\overline{\boldsymbol{J}}$ is plotted in Figure 4, showing the effect of advection speed and the super viscosity.

Figure 3(a) 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. This reveals that the trajectory is possibly not ergodic, making it a poor choice as a prototypical trajectory for LSS, ROM-LSS, and HROM-LSS calculations.

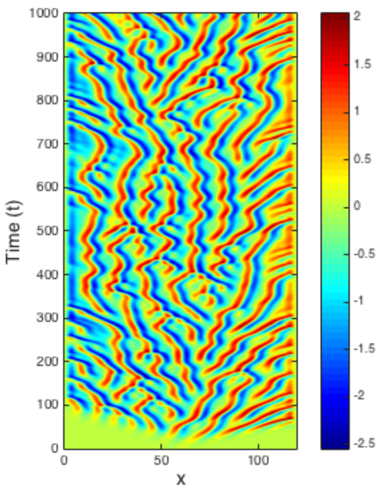
Figure 3(b) shows that by decreasing α by half, we obtain a trajectory in which the initial conditions become "washed out" in a reasonable amount of time. The lower advection speeds do not overcome the



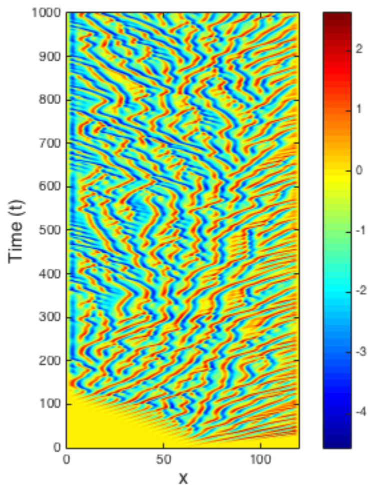
(a) $\alpha = 1,\ \beta = 1$



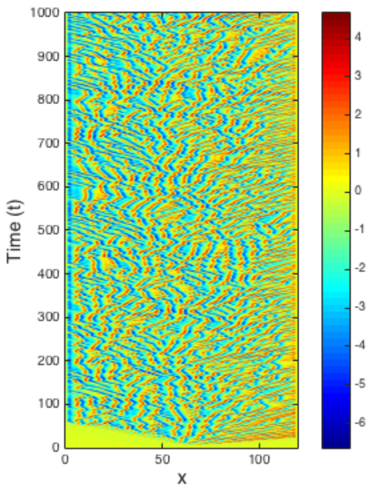
(b) $\alpha = 0.5,\ \beta = 1$



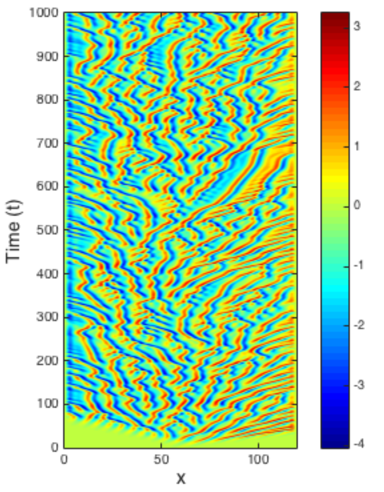
(c) $\alpha = 0.25,\ \beta = 1$



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



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



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

Figure 3. KS solutions for different parameters

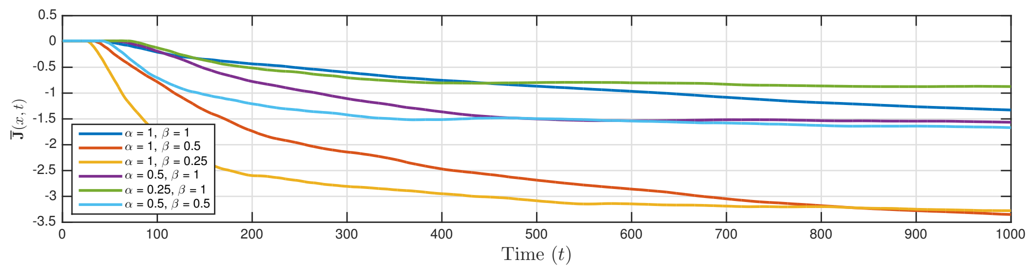


Figure 4. Time-averaged outputs

super viscosity. Figure 4 shows that this trajectory reaches a steady time-averaged output by about $t = 500$, a result that we did not see for $\alpha = 1, \beta = 1$. For this trajectory, given how long the zero regions exist in the $x - t$ contour plots, the minimum burn time should be $t_{burn} \approx 125$. From this results, decreasing the advection speed, gives an ergodic system compared to when $\alpha = 1$. This makes it a possible choice as a prototypical trajectory for further testing.

Figure 3(c) shows that by further decreasing α to $\alpha = 0.25$, we have similarly looking trajectories compared to when $\alpha = 0.5$, except that t_{burn} can be set at a minimum to $t_{burn} \approx 100$ allowing some savings in the computational costs in creating the primal solution. Figure 4 shows that compared to the $\alpha = 0.5$ case, the $\alpha = 0.25$ trajectory reaches a steady state time-averaged solution earlier, meaning that the influence of the initial conditions on the time-averaged output disappears earlier, and making it more ergodic than the $\alpha = 0.25$ case.

Now, while keeping $\alpha = 1$, we investigate the effect of the super viscosity on the trajectories. Figure 3(d) shows the trajectory 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 3(a), the trajectory is able to reach a situation where the initial conditions have little influence on the time-averaged output. From this we see that the lower β leads to lower super viscosity, which dilutes the high advection speed, avoiding dealing with a non ergodic case. However, the advection speed does influence the solution more than in previous cases, where we see that the minimum t_{burn} should be set as $t_{burn} \approx 150$.

We further decrease the super viscosity to $\beta = 0.25$ to see the effect on the trajectory. Figure 3(e) shows the solution for $\alpha = 1, \beta = 0.25$. The thickness of the coherent structures has decreased further (showing stronger chaotic behavior) and the 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 statistically steady-state time-averaged outputs. This case shows that we have a stronger ergodic behavior compared to when $\beta = 0.5$.

Lastly, we present results for $\alpha = 0.5, \beta = 0.5$ in Figure 3(f), which shows that both the magnitudes of α and β, and the ratio of these parameters affects the solution behavior. Even though the ratio is 1, the solution is ergodic and is able to give steady state time-averaged outputs that are influenced little by the initial conditions. However, this case shows that the advection speed heavily influences the value of the time-averaged output since this case reaches approximately the same time-averaged output as $\alpha = 0.5, \beta = 1$. The minimum burn time here should be $t_{burn} \approx 75$.

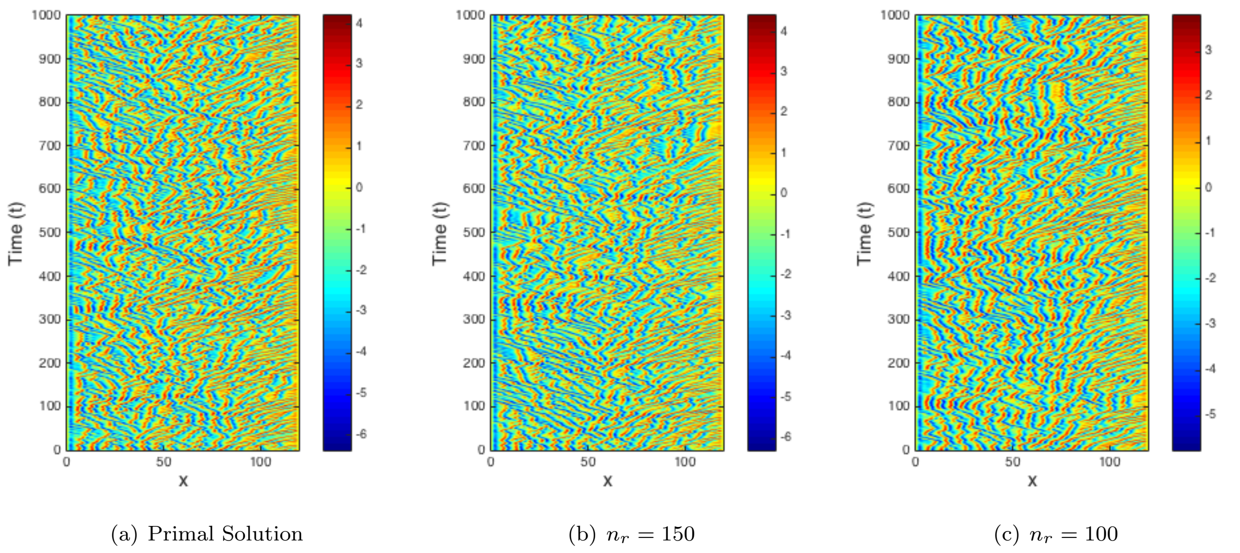
A good prototypical chaotic system to test LSS, ROM, and HROM is one that is heavily chaotic, has minimal required burn time, reaches statistically-converged time-averaged outputs as quickly as possible, and is ergodic. The case that best fulfills all of these requirements is the one presented in Figure 3(e). Another possible choice is the parameters in Figure 3(f). For the rest of this paper, all results are for $\alpha = 1, \beta = 0.25$.

X.B. Reduced-Order Modeling of the Kuramoto-Sivashinsky equation

Before demonstrating the effectiveness of ROM and HROM for KS with $\alpha = 1, \beta = 0.25$, we confirm that the ROM of KS produces approximately the same time-averaged outputs as that of the original primal solution. Figure 5 shows $x - t$ contour results for $n_r = 150, 100, 50, 25, 10$ and for the primal solution. The burn time is set to $t_{\text{burn}} = 100$ and the start and final times of the simulation are set to $T_0 = 0$ to $T_f = 1000$. The simulation contains $T_n = 50005$ time nodes and $n_s = 25002$ snapshots. The spatial domain corresponds to $X_0 = 0$ to $X_f = 120$ and contains $x_n = 360$ spatial nodes. BDF2 is used for the time discretization method and the spatial approximation order is set to $p = 2$. Figure 6 shows how n_r affects the long-term time-averaged output relative to the full-order solution.

In Figure 5, we see that each reduced-order solution produces significantly different trajectories with time. This is to be expected due to the fact that any small perturbations in space will produce a different trajectory that diverges away from the original trajectory. Results in Figures 5(b) and 5(c) show that the trajectories for $n_r = 150$ and $n_r = 100$ look relatively similar compared to the primal solution (full-order model) in terms of the solution coherent structures. This similarity is reflected in Figure 6, where we see that the time-averaged outputs for $n_r = 150$ and $n_r = 100$ estimate the time-averaged output of the primal solution relatively closely. As n_r decreases, we see from Figure 6, that the time-averaged solutions do become more inaccurate, which can be seen for $n_r = 50$, $n_r = 25$, and $n_r = 10$. To understand why this is the case, we look at the trajectories of these cases in Figure 5.

In Figures 5(d)- 5(f), we see that the reduced-order solutions begin to act less chaotic and almost reach steady like behaviors, especially for $n_r = 10$. This increase in non-chaotic nature with decreasing n_r affects the accuracy of the estimated time-averaged solution. It is evident here that one requirement to estimate the


(a) Primal Solution (b) $n_r = 150$ (c) $n_r = 100$

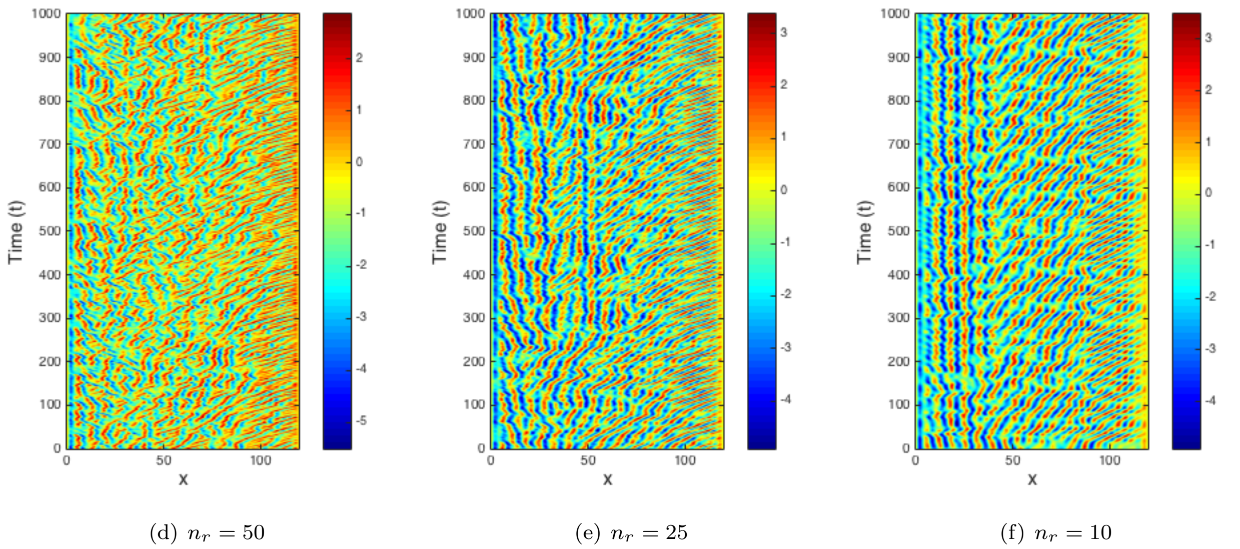

(d) $n_r = 50$ (e) $n_r = 25$ (f) $n_r = 10$

Figure 5. ROM results for KS equations: Each x-t contour plot shows a ROM for a specific number of basis functions, n_r. The number of basis functions for each case is not the same number of snapshots n_s. With more basis functions, the ROM is still exhibits chaotic behaviors, whereas with fewer basis functions e.g. $n_r = 10$, the ROM does not exhibit chaotic behaviors.

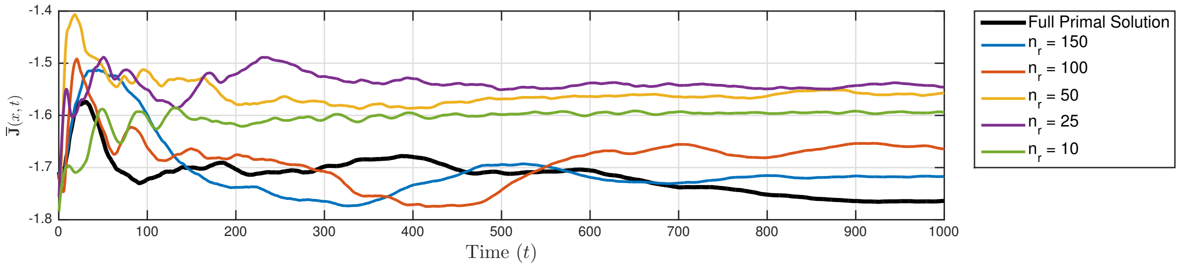


Figure 6. Time-averaged output for $n_r = 150$, $n_r = 100$, $n_r = 50$, $n_r = 25$, and $n_r = 10$ in comparison to the full-order primal solution.

time-averaged solution accurately is that the ROM should also exhibit chaotic behavior. One reason why the trajectory for $n_r = 10$ does 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. This behavior can also be seen more gradually for $n_r = 50$ and $n_r = 25$.

The most effective ROM to use for ROM-LSS is one in which n_s is small enough without compromising the chaotic nature of the original problem. Based on the results of Figures 5 and 6, the next step is to run the LSS procedure on a ROM that contains $n_s = 1001$ and $n_r = 100$. The $n_r = 100$ case still approximates the time-averaged output with the fewest number of basis functions n_r. By using this ROM, we reduce the number of spatial variables of ROM by 160, from the total number of spatial nodes of $x_n = 360$. Next, we perform LSS on this ROM and see if we can calculate usable error estimates.

X.C. Hyper-Reduced-Order Modeling of the Kuramoto-Sivashinsky Equation

ROM is the first step to reducing the computational cost associated with the calculation of adjoints using LSS. However, further approximations need to be made for the residuals and the Jacobians by implementing GNAT for HROM. The main tuning parameter is the number of sample nodes, n_i, calculated from the greedy algorithm. Decreasing n_i decreases the size of the states needed to calculate the residuals and Jacobians. However, with decreasing n_i, errors from the approximation increase. If the number of sample nodes, n_i, is too low, HROM will fail. Figure 7 shows the results on how decreasing the number of sample nodes, n_i, ranging from $n_i = [180, 360]$, affects the time-averaged output on u. Figure 8 shows the full model solution, the corresponding ROM solution, and the corresponding HROM solutions for $n_i = [360, 180]$. The burn time was set to $t_{\text{burn}} = 100$ and the start and final times of the simulation are set to $T_0 = 0$ to $T_f = 1000$. The simulation contains $T_n = 50005$ nodes and $n_s = 12501$ snapshots (fewer snapshots than the ROM case). The spatial domain also corresponds to $X_0 = 0$ to $X_f = 120$ and contains $x_n = 360$ spatial nodes. $p = 2$ and second-order backwards differencing in time are used in the discretization. For the HROM routine, the Gauss-Newton tolerance was set to 10^{-9}.

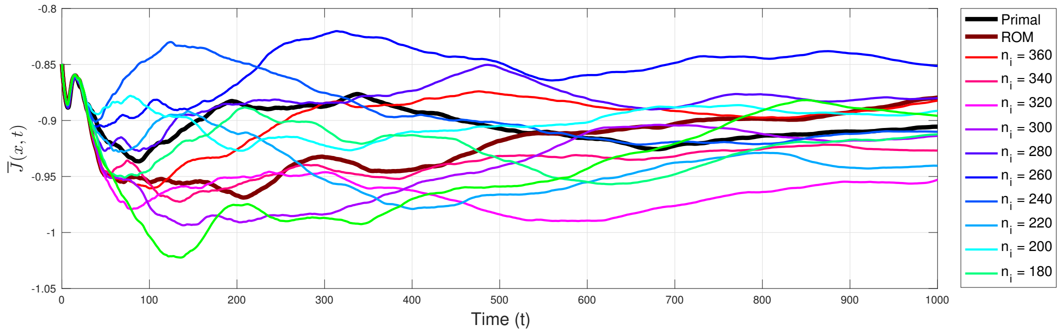
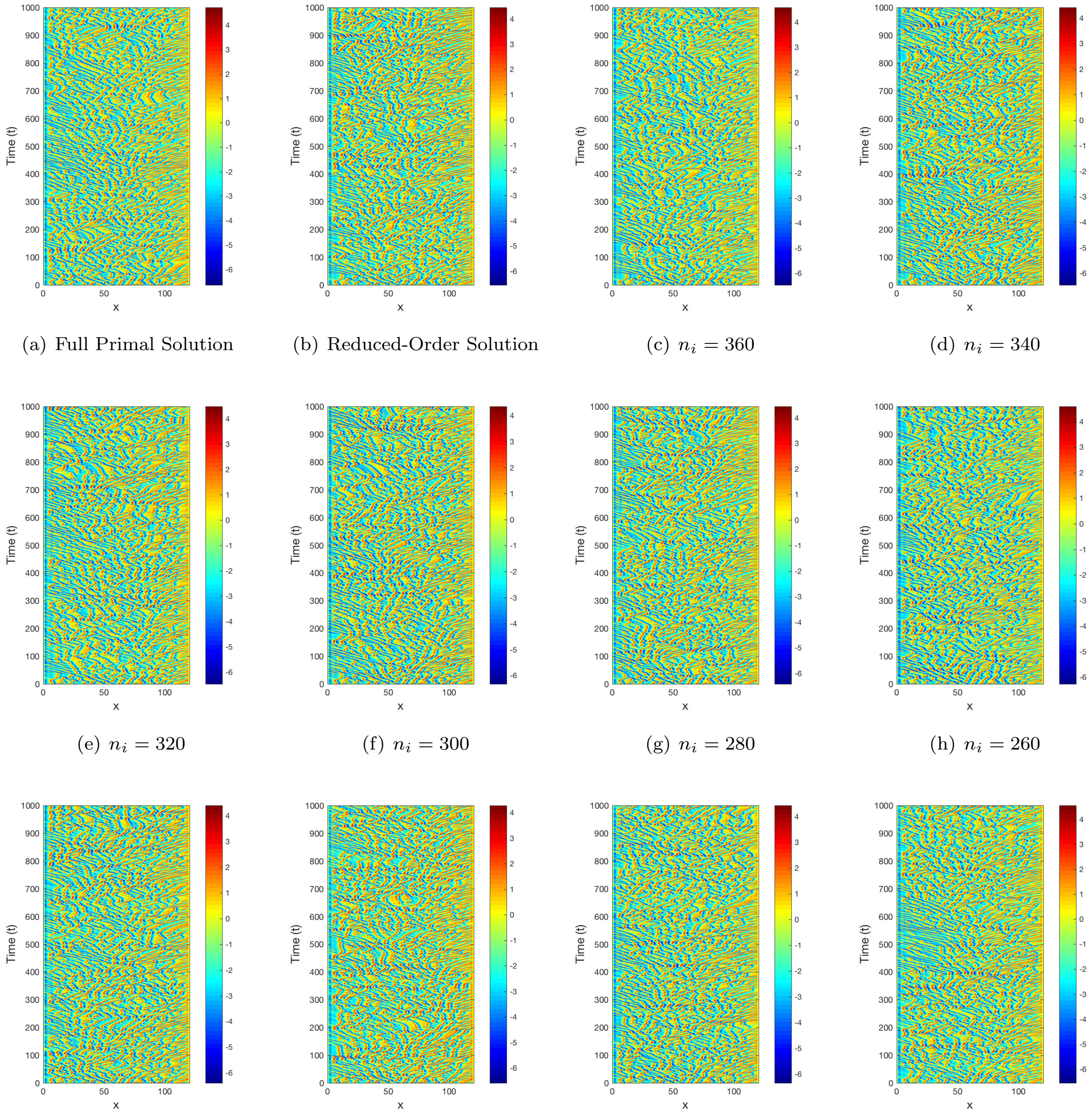


Figure 7. Time-averaged HROM output for various n_i in comparison to the full-order primal solution and ROM.

In Figure 7, we see that hyper-reduced-order solutions predict the time-averaged output for the primal solution within 5%. Additional runs were made for $n_i < 160$, but the HROM produced time-averaged outputs that increased exponentially. This shows that n_i must be sufficiently large, in this case greater than $n_i = 160$, in order to accurately predict the time-averaged output. In Figure 8, we see that over a short period of time, which is approximately $T \approx 100$, decreasing n_i, does little to affect the KS trajectories. However, by the time we get to $T = 1000$, the trajectories diverge away from the full primal solution, the ROM solution, and the other HROM solutions with decreasing n_i. Again, this is due to the chaotic system being very sensitive to the initial conditions. With decreasing n_i, the system still exhibits chaotic behaviors. These results show that it is possible to find a smaller HROM for a chaotic solution that can be used for LSS.



(a) Full Primal Solution (b) Reduced-Order Solution (c) $n_i = 360$ (d) $n_i = 340$

(e) $n_i = 320$ (f) $n_i = 300$ (g) $n_i = 280$ (h) $n_i = 260$

(i) $n_i = 240$ (j) $n_i = 220$ (k) $n_i = 200$ (l) $n_i = 180$

Figure 8. HROM trajectory results for different number of sample nodes n_i

X.D. Preliminary Error Estimation Results Using HROM-LSS

It has been shown that we can acquire ROMs and HROMs that emulate the original full order solution accurately given that enough reduced-order basis functions and reduced-order basis residual/Jacobian functions are used. This was shown by studying how the time average outputs of u behaved with different numbers of basis functions (n_r) and samples nodes (n_i). The HROMs can now be used in conjunction with LSS to find error estimates. To this end, an HROM-LSS method was implemented using KS to obtain some preliminary results. The coarse and fine spatial orders were set to $p_H = 2$ and $p_h = 3$, respectively. To start, the number of sample nodes were set to $n_i = 350$ out of the possible 480 nodes and the time step was set to $\Delta t = 0.2$. The burn time was set to $t_{\text{burn}} = 100$. Second-order backwards differencing was used for the time discretization. Note that the $p = 1$ solution for KS does not exhibit chaotic behavior, making it not suitable for these error estimate test cases. The Gauss-Newton tolerance was set to 10^{-9} in the HROM portion of the HROM-LSS method. The GMRES tolerance was set to 10^{-8} in the LSS portion of the HROM-LSS method.

Figure 9 shows error estimates obtained using the adjoint-weighted residual for four different simulation times, $T = 20$, $T = 40$, $T = 60$, and $T = 80$. For each of these four time simulations, we ran ten different test cases. After the burn time, each simulation's initial conditions were perturbed by a spatially random value in the range of $\delta u_0 = [-0.2, 0.2]$. Multiple runs are necessary because we are interested in the statistical output and behavior of the error estimate. The error bars in Figure 9 refer to the first standard deviation of the data and the box markers refer to the statistical average of all the error estimates. More data was taken for the actual errors separately in order to obtain a better understanding of how these errors change with the perturbations to the initial conditions. In Figure 9, we see that for $T = 20$ and $T = 40$, the

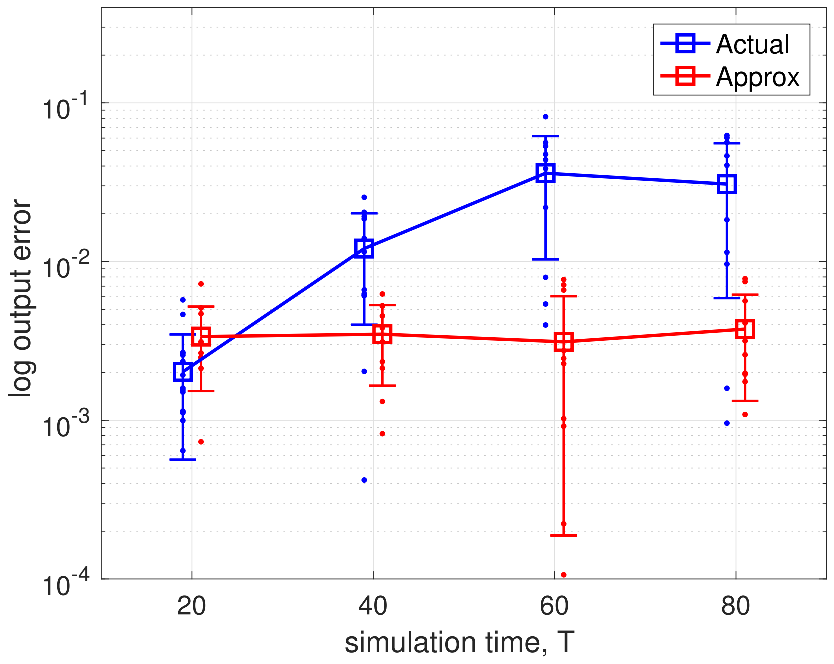
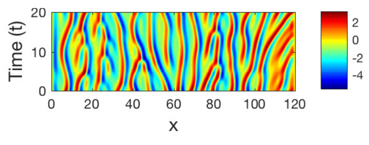
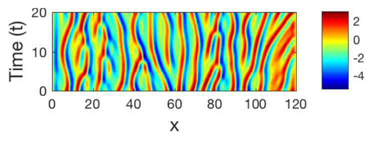


Figure 9. HROM-LSS results for $T = 20$ and $T = 40$ for different initial conditions. Blue data refers to the actual error estimate and the red data refers to the approximate error estimate found from HROM-LSS.

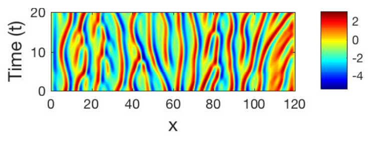
distribution of the actual error estimates in blue are only slightly larger than the distribution of the HROM-LSS approximated error estimates in red. These results for $T = 20$ and $T = 40$ are somewhat unexpected; when we perturb the initial conditions of a chaotic system, we would expect a larger spread in the actual errors, due to the outputs not being statistically converged. After studying how the KS equations behave, we conclude that the smaller spread may be due to the KS trajectories not yet diverging sufficiently by $T = 20$ and $T = 40$. This is further supported by the results of $T = 40$, where we see that the differences in the mean between the actual and approximate error estimates are larger than that at $T = 20$. This may point to the likelihood that by $T = 40$, the trajectories have diverged just enough to exhibit the beginnings of some chaotic behavior. The actual error estimate spread for $T = 60$ is only slightly larger than that of



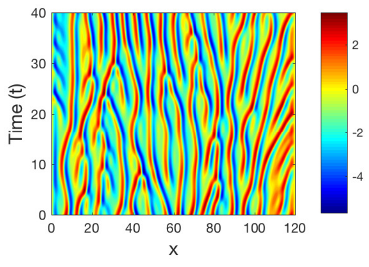
(a) FOM: $T = 20, n_{seg} = 4, n_r = 20$



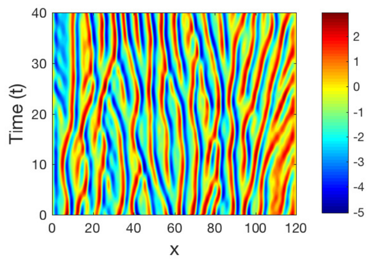
(b) ROM: $T = 20, n_{seg} = 4, n_r = 20$



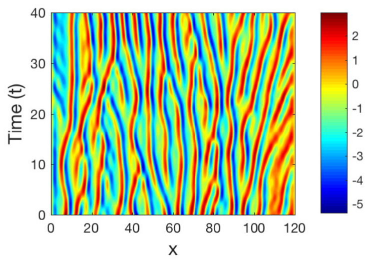
(c) HROM: $T = 20, n_{seg} = 4, n_r = 20$



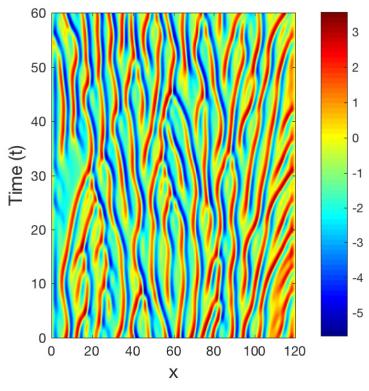
(d) FOM: $T = 40, n_{seg} = 8, n_r = 20$



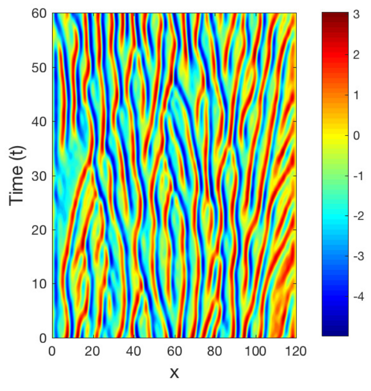
(e) ROM: $T = 40, n_{seg} = 8, n_r = 20$



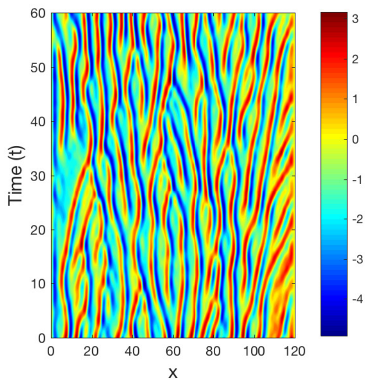
(f) HROM: $T = 40, n_{seg} = 8, n_r = 20$



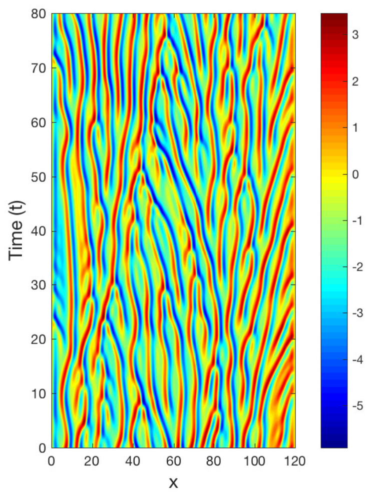
(g) FOM: $T = 60, n_{seg} = 12, n_r = 30$



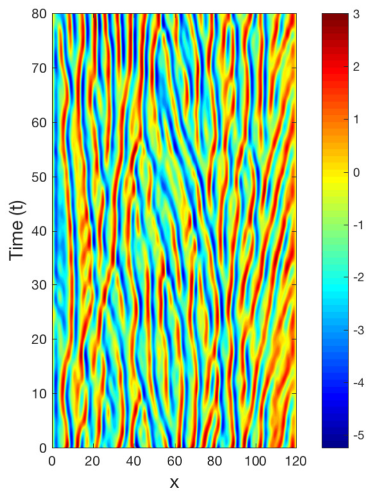
(h) ROM: $T = 60, n_{seg} = 12, n_r = 30$



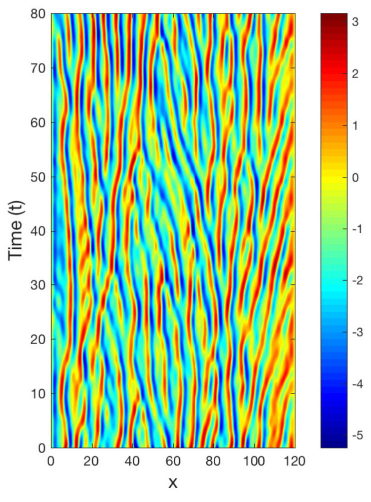
(i) HROM: $T = 60, n_{seg} = 12, n_r = 30$



(j) FOM: $T = 80, n_{seg} = 16, n_r = 30$

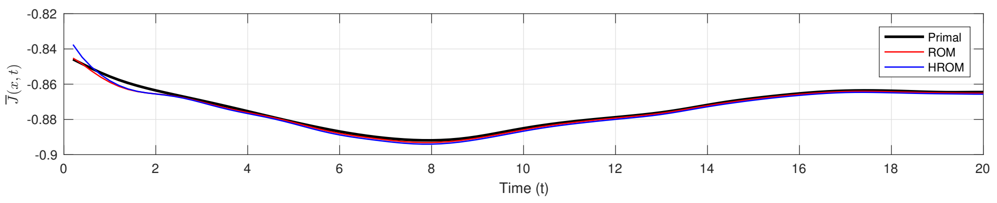


(k) ROM: $T = 80, n_{seg} = 16, n_r = 30$

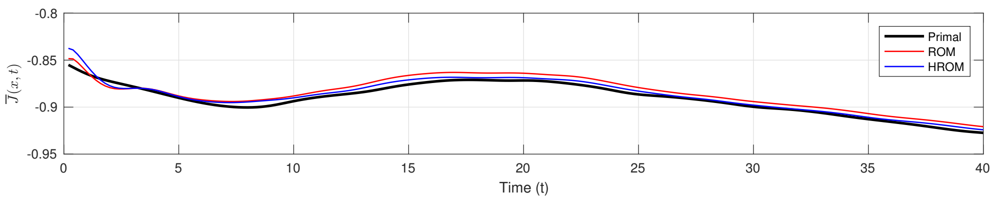


(l) HROM: $T = 80, n_{seg} = 16, n_r = 30$

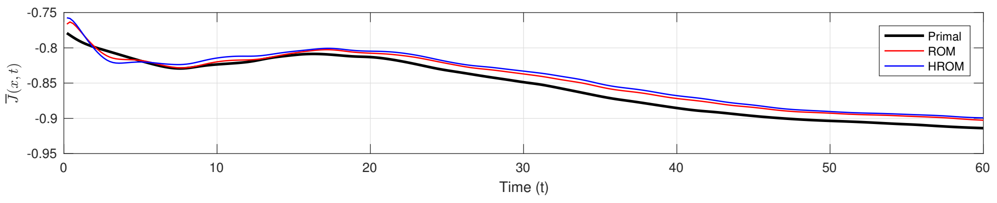
Figure 10. HROM-LSS trajectory results for $T = 20$, $T = 40$, $T = 60$, and $T = 80$ for one of ten cases.



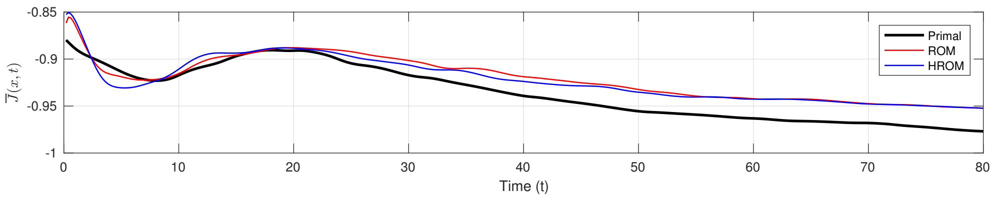
(a) $T = 20, n_{seg} = 4, n_r = 20$



(b) $T = 40, n_{seg} = 8, n_r = 20$



(c) $T = 60, n_{seg} = 12, n_r = 30$

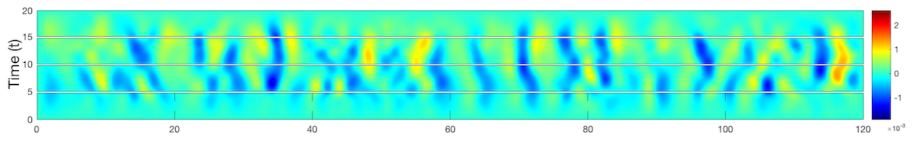


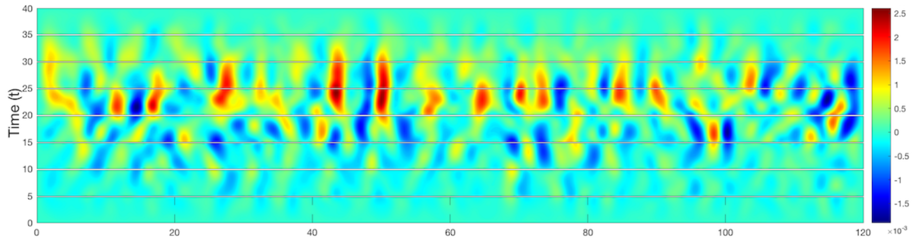
(d) $T = 80, n_{seg} = 16, n_r = 30$

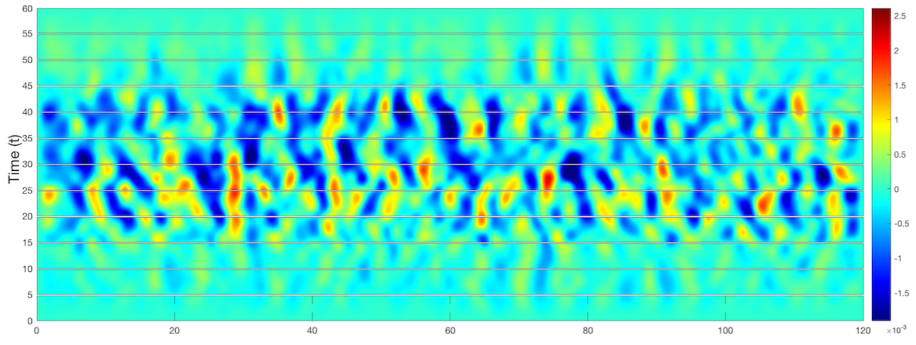
Figure 11. HROM-LSS time average output results for $T = 20$, $T = 40$, $T = 60$, and $T = 80$ for one of ten cases.

$T = 40$ while the approximate error estimate spread for $T = 60$ is significantly larger than previous time simulations, consistent with expectations for a chaotic system. However, we still would expect the spread for the actual error estimate for $T = 60$ to be larger. When we look at the trajectories for KS, we see that the trajectories still have not diverged enough to fully exhibit chaotic behavior, giving possible reasons for this smaller spread of the actual error estimate data. We see that the difference between the mean actual error estimate and the mean approximate error estimate continues to increase from $T = 20$ to $T = 60$, showing that the KS trajectories are becoming more chaotic with time. Lastly, we look at the results for $T = 80$. The spread of the actual error estimate is larger in comparison to the previous simulation time results. The difference between the mean actual error estimate and the mean approximate error estimate decreased compared to that of $T = 60$, behavior that we saw with LSS results for the Lorenz attractor.[34] We look at the KS trajectories to further see how chaotic the trajectories are and find them more chaotic than that of $T = 60$, but not exhibiting full chaotic behavior. Note that by $T = 80$, the traditional adjoint calculation for KS has already started diverging and is useless for error estimates. Further runs for longer time spans, $T = [100, 1000]$ are needed in order to fully understand the relationship between the chaotic system and the error estimates.

In Figure 10, we show for each time simulation, one test case's solutions from Figure 9 for the full-order model (FOM), the ROM, and the HROM. The number of time segments n_{seg} is used in the checkpoint design of HROM-LSS and is designated for each time span T. The number of reduced-order basis functions n_r is designated for each trajectory plot as well. Figure 11 shows the time average output of the solutions from Figure 10. A working ROM and HROM must be able to produce trajectories that emulate that of the FOM


(a) $T = 20, n_{seg} = 4, n_r = 20$


(b) $T = 40, n_{seg} = 8, n_r = 20$


(c) $T = 60, n_{seg} = 12, n_r = 30$

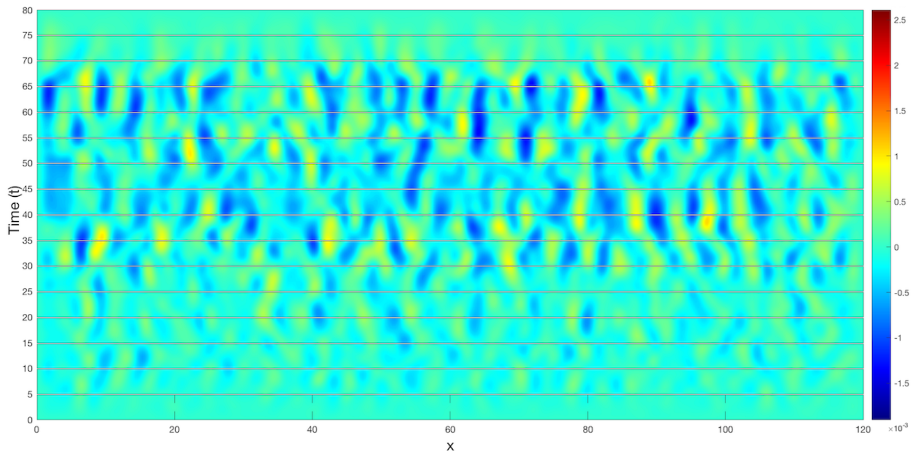

(d) $T = 80, n_{seg} = 16, n_r = 30$

Figure 12. HROM-LSS adjoint $\phi_{1,r}$ results for $T = 20$, $T = 40$, $T = 60$, and $T = 80$ for one of ten cases.

as much as possible and must be able to emulate the time average output. From Figure 10 and Figure 11, we can conclude that we were able to obtain a working hyper-reduced order solution. This is important to verify before using it as a reduced solution for the HROM-LSS method.

Another important result to look at is the adjoint associated with the trajectories from Figure 10 and Figure 11, $\boldsymbol{\psi}_1$. This shows in which regions along space and time the output is most sensitive to residual perturbations. $\boldsymbol{\psi}_{1,r}$ was calculated exactly on the fine space, $p = 3$. Figure 12 shows the adjoint for each time length, T. The contours are constructed separately for each time segment, $\Delta T = 5$. The breaks in the adjoint solution between each time segment are characteristic of the checkpoint method from LSS.[2] The adjoint, $\boldsymbol{\psi}_1$, was found from $\boldsymbol{\psi}_{1,r}$ by expanding the reduced adjoint solution with the basis vectors,

$$\boldsymbol{\psi}_1 = \Phi \boldsymbol{\psi}_{1,r}. \tag{70}$$

Based of the results of Figure 9, HROM-LSS is able to calculate useful and accurate adjoints for error estimates unlike the traditional adjoint method. However, a natural question is whether or not HROM-LSS can calculate adjoints for error estimates with lower computational costs than the original LSS method. To answer this we look at and compare the expected number of GMRES iterations, n_G, from the optimization process required to solve the original LSS equations and the expected number of GMRES iterations from the optimization process required to solve the HROM-LSS equations. The summary of the number of GMRES iterations needed to solve for the adjoints using LSS and HROM-LSS is found in Table 1. We see that HROM-LSS requires fewer GMRES iterations compared to LSS to solve for the adjoint. Furthermore, each iteration is cheaper due to fewer unknowns in the reduced system compared to the full system. This leads to the conclusion that HROM-LSS allows for the calculation of accurate adjoints with fewer computational resources compared to that of LSS and shows that reduced-order modeling of the chaotic system is a reasonable step to take for error estimation.

Table 1. Number of GMRES Iteration to Solve for $\boldsymbol{\psi}_{1,r}$

	Expected n_G		Average Actual n_G
	LSS	HROM-LSS	HROM-LSS
$T = 20$	840	140	120
$T = 40$	1800	300	267.7
$T = 60$	2760	690	605.1
$T = 80$	3720	930	834.3

XI. Conclusion

ROM and HROM can decrease the size and preserve the inherent characteristics of the chaotic system, which makes the application of LSS more feasible for the estimation of numerical errors in outputs. The ROMs help increase accuracy in comparison to time-windowing approaches, while decreasing the computational costs required by LSS applied to the full-order system. This is possible due to ROM's and HROM's ability to reduce the size of the system without significantly reducing the accuracy of the predicted outputs. By using ROM and HROM instead of the full-order primal solution, we can obtain usable adjoints more economically. The outputs of interest are statistical quantities and are used in this study to determine which ROMs and HROMs are suitable to use for LSS. In ROM, the residuals and Jacobians are calculated exactly, whereas in HROM, the residuals and Jacobians are approximately calculated using GNAT, further reducing the size of the overall solution. In this paper, this modified LSS method is referred to as HROM-LSS.

The prototypical equation for this work is the 1D Kuramoto-Sivashinky problem. The parameters for this governing equation are set to $\alpha = 1$ and $\beta = 0.25$. After testing several different types of ROMs and HROMs on KS, we found that the best ROM and HROM that preserves the nature of the original problem consists of a large number of snapshots (n_s) and the fewest possible number of basis functions (n_r). We also found that HROM does produce usable reduced states that still accurately predict the time-averaged output when compared to the primal solution. This is promising and leads to HROM-LSS. Preliminary results for HROM-LSS have shown that fewer GMRES iterations are needed compared to that of LSS in order to still produce accurate error approximations. Based on these results, the main area of future work is to improve the HROM-LSS algorithm's robustness in more complicated cases and to extend this method to 2D chaotic problems.

Acknowledgments

The authors acknowledge the support of the Boeing Company, with technical monitor Dr. Mori Mani.

References

[1]Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *American Institute of Aeronautics and Astronautics Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[2]Blonigan, P. J., Gomez, S. A., and Wang, Q., "Least Squares Shadowing for Sensitivity Analysis of Turbulent Fluid Flows," AIAA Paper 2014–1426, 2014.

[3]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.

[4]Becker, R. and Rannacher, R., "An Optimal Control Approach to a Posteriori Error Estimation in Finite Element Methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[5]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.

[6]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.

[7]Nemec, M. and Aftosmis, M. J., "Error Estimation and Adpative Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 2007.

[8]Mani, K. and Mavriplis, D. J., "Discrete Adjoint Based Time-Step Adaptation and Error Reduction in Unsteady Flow Problems," AIAA Paper 2007-3944, 2007.

[9]Mani, K. and Mavriplis, D. J., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *Journal of Computational Physics*, Vol. 229, 2010, pp. 415–440.

[10]Barth, T. J., "Space-Time Error Representation and Estimation in Navier-Stokes Calculations," *Complex Effects in Large Eddy Simulations*, edited by S. C. Kassinos, C. A. Langer, G. Iaccarino, and P. Moin, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.

[11]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.

[12]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.

[13]Fidkowski, K. J., "An Output-Based Dynamic Order Refinement Strategy for Unsteady Aerodynamics," AIAA Paper 2012-77, 2012.

[14]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.

[15]Krakos, J. A., *Unsteady Adjoint Analysis for Output Sensitivity and Mesh Adaptation*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[16]Belme, A., Dervieux, A., and Alauzet, F., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *Journal of Computational Physics*, Vol. 231, 2012, pp. 6323–6348.

[17]Flynt, B. T. and Mavriplis, D. J., "Discrete Adjoint Based Adaptive Error Control in Unsteady Flow Problems," AIAA Paper 2012-0078, 2012.

[18]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.

[19]Kast, S. M., Fidkowski, K. J., and Roe, P. L., "An Unsteady Entropy Adjoint Approach for Adaptive Solution of the Shallow-Water Equations," AIAA Paper 2011-3694, 2011.

[20]Kast, S. M., Ceze, M. A., and Fidkowski, K. J., "Output-Adaptive Solution Strategies for Unsteady Aerodynamics on Deformable Domains," Seventh International Conference on Computational Fluid Dynamics ICCFD7-3802, 2012.

[21]Kast, S. M. and Fidkowski, K. J., "Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494.

[22]Fidkowski, K. J., "An Output-Based Adaptive Hybridized Discontinuous Galerkin Method on Deforming Domains," AIAA Paper 2015–2602, 2015.

[23]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.

[24]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.

[25]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. Rnquist, 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, 10.1007/978-3-642-15337-2_4.

[26]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.

[27]Dahm, J. P. and Fidkowski, K. J., "Error Estimation and Adaptation in Hybridized Discontinous Galerkin Methods," AIAA Paper 2014–0078, 2014.

[28]Kast, S. M., Dahm, J. P., and Fidkowski, K. J., "Optimal Test Functions for Boundary Accuracy in Discontinuous Finite Element Methods," *Journal of Computational Physics*, Vol. 298, No. 1, 2015, pp. 360 – 386.

[29]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.

[30]Fidkowski, K. J., "A Hybridized Discontinuous Galerkin Method on Mapped Deforming Domains," *Computers and Fluids*, Vol. 139, No. 5, November 2016, pp. 80–91.

[31] Wang, Q., "Forward and Adjoint Sensitivity Computation of Chaotic Dynamical Systems," *Journal of Computational Physics*, Vol. 235, 2013, pp. 1–13.

[32] Ghosal, S., "An Analysis of Numerical Errors in Large-Eddy Simulations of Turbulence," *Journal of Computational Physics*, Vol. 125, No. 1, 1996, pp. 187–206.

[33] Kravchenko, A. and Moin, P., "On the Effect of Numerical Errors in Large Eddy Simulations of Turbulent Flows," *Journal of Computational Physics*, Vol. 131, No. 2, 1997, pp. 310–322.

[34] Shimizu, Y. S. and Fidkowski, K., "Output Error Estimation for Chaotic Flows," *46th AIAA Fluid Dynamics Conference*, 2016, p. 3806.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.