SAND2020-11051R

SAND20-0724

LDRD PROJECT NUMBER: 214103

LDRD PROJECT TITLE: Multiscale modeling, high-order methods, and data-driven modeling

PROJECT TEAM MEMBERS: Eric Parish

Abstract: Projection-based reduced-order models (ROMs) comprise a promising set of data-driven approaches for accelerating the simulation of high-fidelity numerical simulations. Standard projection-based ROM approaches, however, suffer from several drawbacks when applied to the complex nonlinear dynamical systems commonly encountered in science and engineering. These limitations include a lack of stability, accuracy, and sharp a posteriori error estimators. This work addresses these limitations by leveraging multiscale modeling, least-squares principles, and machine learning to develop novel reduced-order modeling approaches, along with data-driven a posteriori error estimators, for dynamical systems. Theoretical and numerical results demonstrate that the two ROM approaches developed in this work – namely the windowed least-squares method and the Adjoint Petrov—Galerkin method – yield substantial improvements over state-of-the-art approaches. Additionally, numerical results demonstrate the capability of the *a posteriori error* models developed in this work.

Introduction and executive summary of results:

Simulating parameterized dynamical systems arises in many applications across science and engineering, including uncertainty quantification, design, and optimization. In many contexts, executing a dynamical-system simulation at a single parameter instance—which entails the numerical integration of a system of ordinary differential equations (ODEs)—incurs an extremely large computational cost. When the application is time critical or many query in nature, analysts often rely on a low-cost surrogate model that makes the application tractable.

Projection-based reduced-order models (ROMs) comprise one such surrogate modeling strategy. First, these techniques execute a computationally expensive offline stage that computes a low-dimensional trial subspace on which the dynamical-system state can be well approximated (e.g., by computing state "snapshots" at different time and parameter instances, by solving Lyapunov equations). Second, these methods execute an inexpensive online stage during which they compute approximations to the dynamical-system trajectory that reside on this trial subspace via, e.g., projection of the full-order model or residual minimization.

Model reduction for linear-time-invariant systems (and other well structured dynamical systems) is quite mature [1, 2, 3, 4], as system-theoretic properties (e.g., controllability, observability, asymptotic stability, $\mathscr{H}_2$-optimality) can be readily quantified and accounted for; this often results in certified reduced-order models that inherit such important properties. The primary challenge in developing reduced-order models for general nonlinear dynamical systems is that such properties are difficult to assess quantitatively. As a result, it is challenging to develop reduced-order models that preserve important dynamical-system properties, which often results in methods that yield trajectories that are inaccurate, unstable, or violate physical properties. Further, even if the ROM solution is stable and accurate, it is difficult to develop reliable estimates of the error incurred by the ROM.

The present work focuses on advancing the state-of-the-art in model reduction for nonlinear dynamical systems. We do this by (1) developing novel methodologies that yield robust ROM formulations for dynamical systems and (2) developing data-driven *a posteriori* error models capable of quantifying the ROM error. In what follows, we provide a literature review on the state-of-the art in model reduction methods and error modeling approaches.

To develop stable and accurate ROM formulations of nonlinear dynamical systems, researchers have pursued several directions that aim to imbue reduced-order models for nonlinear dynamical systems with properties that can improve robustness and accuracy. These efforts include residual-minimization approaches that equip the ROM solution with a notion of optimality [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]; space–time approaches that lead to error bounds that grow slowly in time [16, 17, 18, 19, 20]; "energy-based" inner products that ensure non-increasing entropy in the ROM solution [21, 22, 23]; basis-adaptation methods that improve the ROM's accuracy *a posteriori* [24, 25, 26], stabilizing subspace rotations that account for truncated modes *a priori* [27], structure-preserving methods that enforce conservation [28] or (port-)Hamiltonian/Lagrangian structure [29, 30, 31, 32, 33] in the ROM; and subgrid-scale modeling methods that aim to improve accuracy by addressing the closure problem [34, 35, 36, 37, 38, 39, 40, 41, 42]. Residual minimization and subgrid-scale modeling methods are the most relevant to the present work, and are the focus of the following review.

Residual-minimization methods in model reduction compute the solution within a low-dimensional trial subspace that minimizes the full-order-model residual.[1] Researchers have developed such residual-minimizing model-reduction methods for both static systems (i.e., systems without time-dependence) [7, 8, 9, 10, 12, 13, 14] and dynamical systems [14, 11, 13, 6, 5, 45]. In the latter category, Refs. [14, 11, 13, 6, 5] formulated the residual minimization problem for dynamical systems by sequentially minimizing the *time-discrete* full-order-model residual (i.e., the residual arising after applying time discretization) at each time instance on the time-discretization grid. This formulation is often referred to as the *least–squares Petrov–Galerkin* (LSPG) method. Numerous numerical experiments have demonstrated that LSPG often yields more accurate and stable solutions than Galerkin [14, 45, 6, 13, 46]. The common intuitive explanation for this improved performance is that, by minimizing the full-order-model residual over a finite time window (rather than time instantaneously), LSPG computes solutions that are more accurate over a larger part of the trajectory as compared to Galerkin.

However, LSPG has several notable shortcomings. First, LSPG exhibits a complex dependence on the time discretization. In particular, changing the time step (Δt) modifies both the time window over which LSPG minimizes the residual as well as the time-discretization error of the full-order model on which LSPG is based. As LSPG and Galerkin projection are equivalent in the limit of $\Delta t \rightarrow 0$, the accuracy of LSPG approaches the (sometimes poor) accuracy of Galerkin as the time step shrinks. For too-large a

[1]While we focus our review on residual-minimization approaches in the context of model reduction, we note that these approaches are intimately related to least-squares finite element methods [43, 44].

time step the accuracy of LSPG also degrades. It is unclear if this is due to the time-discretization error associated with enlarging the time step, or rather if it is due to the size of the window the residual is being minimized over. As a consequence, LSPG often yields the smallest error for an intermediate value of the time step (see, e.g., Ref. [45, Figure 9]); there is no known way to compute this optimal time step *a priori*. Second, as the LSPG approach performs sequential residual minimization in time, its *a posteriori* error bounds grow exponentially in time [45], and it is not equipped with any notion of optimality over the entire time domain of interest. As a result, LSPG is not equipped with *a priori* guarantees of accuracy or stability, even for linear time-invariant systems [14].

A second school of thought addresses stability and accuracy of ROMs from a closure modeling viewpoint. This follows from the idea that instabilities and inaccuracies in ROMs can, for the most part, be attributed to the truncated modes. While these truncated modes may not contain a significant portion of the system energy, they can play a significant role in the dynamics of the ROM [27]. This is analogous to the closure problem encountered in large eddy simulation. Research has examined the construction of mixing length [28], Smagorinsky-type [27, 29, 30, 31], and variational multiscale (VMS) closures [27, 32, 33, 34] for POD-ROMs. The VMS approach is of particular relevance to this work. Originally developed in the context of finite element methods, VMS is a formalism to derive stabilization/closure schemes for numerical simulations of multiscale problems. The VMS procedure is centered around a sum decomposition of the solution in terms of resolved/coarse-scales and unresolved/fine-scales. The impact of the fine-scales on the evolution of the coarse-scales is then accounted for by devising an approximation to the fine-scales. This approximation is often referred to as a “subgrid-scale” or “closure” model.

Finally, regardless of the type of ROM method employed, the use of a ROM introduces error; as such, it is critical to quantify this error and properly account for it in the analysis underpinning the many-query problem. For this purpose, researchers have developed a wide range of methods for *a posteriori* error quantification. These efforts have resulted in a techniques that can be categorized as (1) *a posteriori* error bounds, (2) error indicators, and (3) error models.

A posteriori error bounds place bounds on the (normed) state or quantity-of-interest (QoI) error arising from the use of an approximate solution [47, 48, 49, 50, 51, 52, 53, 45] Such bounds are appealing because they can provide guaranteed, rigorous bounds on the errors of interest, thus providing a critical tool for certified predictions. However, these bounds often suffer from lack of sharpness, i.e., they are often orders-of-magnitude larger than the approximate-solution error itself. This is exacerbated in dynamical systems, where error bounds for many types of approximate solutions grow exponentially in time (see, e.g., Ref. [45]). In addition, these bounds typically require (difficult-to-compute) estimates or bounds of operator quantities such as Lipschitz and inf–sup constants. These drawbacks often limit the utility of *a posteriori* error bounds in practical applications.

Alternatively, *error indicators* are computable quantities that are *indicative* of the approximate-solution error[54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]. Error indicators typically do not provide unbiased estimates of the error, nor do they rigorously bound the error; instead, they are often correlated with the error, correspond to a low-order approximation of the error, and/or appear as terms in error bounds. Due to their practical utility, error indicators have been largely successful in quantifying and controlling errors through mesh adaptation for static problems (i.e., problems without time evolution). However, their success has been more limited for dynamical systems due to the fact that errors for such systems exhibit dependence on *non-local* quantities, i.e., the approximate-solution error at a given time instance depends on the past time history of the system. Thus, the residual norm at the current time instance is no longer directly indicative of the approximate-solution error; the error additionally depends on non-local quantities.

Error models seek to model the state and QoI errors directly via regression techniques. The most popular such approach is the so-called "model-discrepancy" method of Kennedy and O'Hagan [66] (and related approaches [67, 68, 69, 70, 71, 72]). The model-discrepancy approach computes a Gaussian process (GP) that provides a mapping from any finite number of points in parameter space to a Gaussian random vector representing the approximate-solution error at those points. As such, this technique can be considered a regression approach, where the *regression model* is provided by a GP, the *features* are the system parameters, and the *response* is the approximate-solution error. Recently, researchers have pursued improvements over this technique by (1) considering a wider class of modern machine-learning regression methods than simply Gaussian processes (which do not scale well to high-dimensional feature spaces), and (2) considering more informative features than the system parameters alone. First, the reduced-order model error surrogates (ROMES) method [73] was proposed in the context of static problems. ROMES employs the same regression model (GP) as the model-discrepancy approach, but it employs different features; namely, it employs the aforementioned *error indicators* (i.e., residual norm, approximated dual-weighted residual) as features. Because these quantities are indicative of the error, they provide more informative features for predicting the error than the system parameters. Numerical experiments illustrated the ability of the ROMES method to produce significantly more accurate error predictions than the model-discrepancy approach. Subsequent work [74] again considered static problems, but constructed machine-learning error models that considered a wide range of candidate regression models (e.g., neural networks, support vector machines, random forests) and error-indicator-based features (e.g., residual samples) in order to predict the approximate-solution error response. This work showed the ability of machine-learning methods to predict the error with near perfect accuracy across a range of static problems and approximate-solution types.

The construction of error models for dynamical systems is significantly more challenging than doing so for static systems, as the errors exhibit dependence on non-local quantities. Relevant work on constructing error models for dynamical systems accounts for the time-dependent nature of the approximate-solution error either (1) by using both time itself and time-lagged error indicators as features [75] or (2) by constructing a different regression method for each time instance or window [76]. Unfortunately, both approaches fail to capture non-local dependencies of the error, and—as with the original model-discrepancy method—does not use particularly informative features to predict the error.

This work seeks to address several of the shortcomings described above. First, we advance the state-of-the art in residual-minimizing projections by developing the **windowed least-squares method** for nonlinear model reduction to assess the shortcomings of LSPG. The WLS approach operates by sequentially minimizing the time-continuous full-order-model residual within a low-dimensional space–time trial subspace over time windows. The approach comprises a generalization of the LSPG approach. In addition, the approach addresses key deficiencies in existing model-reduction techniques, e.g., the dependence of LSPG on the time discretization and the exponential growth in time exhibited by a posteriori error bounds for both Galerkin and LSPG projection. Numerical experiments on benchmark problems from compressible fluid dynamics demonstrate that the WLS approach yields more stable and accurate results than the LSPG and Galerkin approaches.

Second, we advance the state-of-the art in closure modeling for ROMs. We develop the **Adjoint Petrov—Galerkin method** (APG). APG is derived by decomposing the generalized coordinates of a dynamical system into a resolved coarse-scale set and an unresolved fine-scale set. A Markovian finite memory assumption within the Mori-Zwanzig formalism is then used to develop a reduced-order representation of the coarse-scales. This procedure leads to a closed reduced-order model that displays commonalities with the adjoint stabilization method used in finite elements. The formulation is shown to be equivalent to a Petrov–Galerkin method with a non-linear, time-varying test basis, thus sharing some similarities with LSPG. Numerical experiments on the compressible Navier-Stokes equations demonstrate that the proposed method can lead to improvements in numerical accuracy, robustness, and computational efficiency over the Galerkin and LSPG methods.

Lastly, we develop **time-series machine-learning error models (TMLEM)** to quantify the error incurred by an approximate solution (such as one obtained via APG or WLS). The TMLEM method constructs a regression model that maps features—which comprise error indicators that are derived from standard a posteriori error-quantification techniques—to a random variable for the approximate-solution error. The proposed framework considers a wide range of candidate features, regression methods, and additive noise models. We consider primarily recursive regression techniques developed for time-series modeling, including both classical time-series models (e.g., autoregressive models) and recurrent neural networks (RNNs), but also analyze standard non-recursive regression techniques (e.g., feed-forward neural networks) for comparative purposes. Numerical experiments conducted on multiple benchmark problems illustrate that the long short-term memory (LSTM) neural network, which is a type of RNN, outperforms other methods and yields substantial improvements in error predictions over traditional approaches.

Detailed description of research and development methodology: As discussed in the introduction, this work aims to develop novel projection-based model reduction approaches and error estimators applicable for nonlinear dynamical systems. We begin by providing the formulation for the full-order model (FOM), followed by a description of standard model-reduction methods. We then outline the windowed least-squares method, adjoint Petrov–Galerkin method, and time series machine learning error models.

Full-order model: We consider the full-order model to be a dynamical system expressed as a system of ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t),t), \qquad x(0) = \mathbf{x}_0, \qquad t \in [0,T], \tag{3.1}$$

where $x : [0,T] \to \mathbb{R}^N$ with $x : \tau \mapsto x(\tau)$ and $\dot{x} \equiv dx/d\tau$ denotes the state implicitly defined as the solution to initial value problem (4.1), $T \in \mathbb{R}_+$ denotes the final time, $\mathbf{x}_0 \in \mathbb{R}^N$ denotes the initial condition, and $f : \mathbb{R}^N \times [0,T] \to \mathbb{R}^N$ with $(\mathbf{y},\tau) \mapsto f(\mathbf{y},\tau)$ denotes the velocity, which is possibly nonlinear in its first argument. For subsequent exposition, we introduce $\mathscr{T}$ to denote the set of (sufficiently smooth) real-valued functions acting on the time domain (i.e., $\mathscr{T} = \{f \,|\, f : [0,T] \to \mathbb{R}\}$); the state can be expressed equivalently as $x \in \mathbb{R}^N \otimes \mathscr{T}$. We refer to the initial value problem defined in Eq. (4.1) as the "full-order model" (FOM) ODE. We note that although the problem of interest described in the introduction corresponds to a parameterized dynamical system, we suppress dependence of the FOM ODE (4.1) on such parameters for notational convenience, as this work focuses on devising a model-reduction approach applicable to a specific parameter instance.

Directly solving the FOM ODE (4.1) is computationally expensive if either the state-space dimension N is large, or if the time-interval length T is large relative to the time step required to numerically integrate Eq. (4.1). For time-critical or many-query applications, it is essential to replace the FOM ODE (4.1) with a strategy that enables an approximate trajectory to be computed at lower computational cost. Projection-based ROMs constitute one such promising approach.

Trial subspaces: Projection-based ROMs operate by restricting the state to belong to a low-dimensional *trial subspace*. We refer to this type of trial space as a spatial–reduction (S-reduction) subspace. At a given time instance $t \in [0,T]$, S-reduction trial subspaces approximate the FOM ODE solution as $\tilde{x}(t) \approx x(t)$, which is enforced to reside in an affine spatial trial subspace of dimension $K \ll N$ such that $\tilde{x}(t) \in \mathbf{x}_{\text{ref}} + \mathscr{V} \subseteq \mathbb{R}^N$, where $\dim(\mathscr{V}) = K$ and $\mathbf{x}_{\text{ref}} \in \mathbb{R}^N$ denotes the reference state, which is often taken to be the initial condition (i.e., $\mathbf{x}_{\text{ref}} = \mathbf{x}_0$). Here, the trial subspace $\mathscr{V}$ is spanned by an orthogonal basis such that $\mathscr{V} = \text{Ran}(\mathbf{V})$ with $\mathbf{V} \equiv \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_K \end{bmatrix} \in \mathbb{V}_K(\mathbb{R}^N)$, where $\mathbb{V}_K(\mathbb{R}^N)$ denotes the compact Stiefel manifold (i.e., $\mathbb{V}_K(\mathbb{R}^N) := \{\mathbf{X} \in \mathbb{R}^{N \times K} \,\big|\, \mathbf{X}^T\mathbf{X} = \mathbf{I}\}$). The basis vectors $\mathbf{v}_i$, $i = 1,\ldots,K$ are typically constructed using state snapshots, e.g., via proper orthogonal decomposition (POD) [77], the reduced-basis method [78, 79, 80, 81, 82]. Thus, at any time instance $t \in [0,T]$, ROMs that employ the S-reduction trial subspace approximate the FOM ODE solution as

$$x(t) \approx \tilde{x}(t) = \mathbf{V}\hat{x}(t) + \mathbf{x}_{\text{ref}}, \tag{3.2}$$

where $\hat{x} \in \mathbb{R}^K \otimes \mathscr{T}$ with $\hat{x} : \tau \mapsto \hat{x}(\tau)$ denotes the generalized coordinates. From the space–time perspective, this is equivalent to approximating the FOM ODE solution trajectory $x \in \mathbb{R}^N \otimes \mathscr{T}$ with $\tilde{x} \in \mathscr{S}\mathscr{T}_{\text{S}}$, where

$$\mathscr{S}\mathscr{T}_{\text{S}} := \mathscr{V} \otimes \mathscr{T} + \mathbf{x}_{\text{ref}} \otimes \mathscr{O} \subseteq \mathbb{R}^N \otimes \mathscr{T}, \tag{3.3}$$

with $\mathscr{O} \in \mathscr{T}$ defined as $\mathscr{O} : \tau \mapsto 1$.

Substituting the approximation (4.2) into the FOM ODE (4.1) and performing orthogonal ℓ^2-projection of the initial condition onto the trial subspace yields the overdetermined system of ODEs

$$\mathbf{V}\dot{\hat{x}}(t) = f(\mathbf{V}\hat{x}(t) + \mathbf{x}_{\text{ref}}, t), \qquad \hat{x}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{3.4}$$

where $\dot{\hat{x}} \equiv d\hat{x}/d\tau$. Because Eq. (4.4) is overdetermined, a solution may not exist. Typically, either *Galerkin* or *least-squares Petrov–Galerkin* projection is employed to reduce the number of equations such that a unique solution exists. We now describe these two methods.

Galerkin projection: The Galerkin approach reduces the number of equations in Eq. (4.4) by enforcing orthogonality of the residual to the spatial trial subspace in the (semi-)inner product induced by the positive (semi-)definite $N \times N$ matrix $\mathbf{A} \equiv [\mathbf{W}]^T\mathbf{W}$ (commonly set to $\mathbf{A} = \mathbf{I}$), i.e.,

$$\dot{\hat{x}}_{\text{G}}(t) = \mathbf{M}^{-1}\mathbf{V}^T\mathbf{A}f(\mathbf{V}\hat{x}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}, t), \qquad \hat{x}_{\text{G}}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{3.5}$$

where $\mathbf{M} \equiv \mathbf{V}^T\mathbf{A}\mathbf{V}$ denotes the $K \times K$ positive definite mass matrix. As demonstrated in Ref. [45], the Galerkin approach can be viewed alternatively as a residual-minimization method, as the Galerkin ODE (4.5) is equivalent to

$$\dot{\hat{x}}_{\text{G}}(t) = \underset{\hat{\mathbf{y}} \in \mathbb{R}^K}{\arg\min} \|\mathbf{V}\hat{\mathbf{y}} - f(\mathbf{V}\hat{x}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}, t)\|_{\mathbf{A}}^2, \qquad \hat{x}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{3.6}$$

where $\|\mathbf{x}\|_{\mathbf{A}} \equiv \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$. Thus, the computed velocity $\dot{\hat{x}}_{\text{G}}(t)$ minimizes the FOM ODE residual evaluated at the state $\mathbf{V}\hat{x}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}$ and time instance t over the spatial trial subspace $\mathscr{V}$.

Least-squares Petrov–Galerkin projection: Despite its time-instantaneous residual-minimization optimality property (4.6), the Galerkin approach can yield inaccurate solutions, particularly if the velocity is not self-adjoint or is nonlinear. Least-squares Petrov–Galerkin (LSPG) [45, 13, 6, 11, 14] was developed as an alternative method that exhibits several advantages over the Galerkin approach. Rather than minimize the (time-continuous) FOM ODE residual at a time instance (as in Galerkin), LSPG minimizes the (time-discrete) FOM OΔE residual (i.e., the residual arising after applying time discretization to the FOM ODE) over a time step. We now describe the LSPG approach in the case of linear multistep methods; Ref. [45] also presents LSPG for Runge–Kutta schemes. Without loss of generality, we introduce a uniform time grid characterized by time step Δt and time instances $t^n = n\Delta t$, $n = 0, \ldots, N_t$. Applying a linear multistep method to discretize the FOM ODE (4.1) with this time grid yields the FOM OΔE, which computes the sequence of discrete solutions $\mathbf{x}^n (\approx x(t^n))$, $n = 1, \ldots, N_t$ as the implicit solution to the system of algebraic equations

$$r^n(\mathbf{x}^n; \mathbf{x}^{n-1}, \ldots, \mathbf{x}^{n-k^n}) = 0, \qquad n = 1, \ldots, N_t, \tag{3.7}$$

with the initial condition $\mathbf{x}^0 = \mathbf{x}_0$. In the above, k^n denotes the number of steps employed by the scheme at the nth time instance and r^n denotes the FOM OΔE residual defined as

$$\begin{aligned} r^n : (\mathbf{y}^n; \mathbf{y}^{n-1}, \dots, \mathbf{y}^{n-k^n}) &\mapsto \frac{1}{\Delta t} \sum_{j=0}^{k^n} \alpha_j^n \mathbf{y}^{n-j} - \sum_{j=0}^{k^n} \beta_j^n f(\mathbf{y}^{n-j}, t^{n-j}), \\ &: \mathbb{R}^N \otimes \mathbb{R}^{k^n+1} \rightarrow \mathbb{R}^N. \end{aligned}$$

Here, $\alpha_j^n, \beta_j^n \in \mathbb{R},\ j = 0, \dots, k^n$ are coefficients that define the linear multistep method at the nth time instance.

At each time instance on the time grid, LSPG substitutes the S-reduction trial subspace approximation (4.2) into the FOM OΔE (4.7) and minimizes the residual, i.e., LSPG sequentially computes the solutions $\tilde{\mathbf{x}}_{\mathrm{L}}^n \approx \mathbf{x}^n$, $n = 1, \dots, N_t$ that satisfy

$$\tilde{\mathbf{x}}_{\mathrm{L}}^n = \underset{\mathbf{y} \in \mathscr{V} + \mathbf{x}_{\mathrm{ref}}}{\arg\min} ||\mathbf{W} r^n(\mathbf{y}; \tilde{\mathbf{x}}_{\mathrm{L}}^{n-1}, \dots, \tilde{\mathbf{x}}_{\mathrm{L}}^{n-k^n})||_2^2, \qquad n = 1, \dots, N_t,$$

where $\mathbf{W} \in \mathbb{R}^{n_s \times N}$, with $K \leq n_s \leq N$, is a weighting matrix that can be used, e.g., to enable hyper-reduction by requiring it to have a small number of nonzero columns.

As described in the introduction, although numerical experiments have demonstrated that LSPG often yields more accurate and stable solutions than Galerkin [14, 45, 6, 13, 46], LSPG still suffers from several shortcomings. In particular, LSPG suffers from its complex dependence on the time discretization, exponentially growing error bounds, and lack of optimality for the trajectory defined over the entire time domain.

Windowed-least-squares model reduction: The windowed least-squares approach seeks to overcome the limitations of existing residual-minimizing model-reduction methods, and to provide a unifying framework from which existing methods can be assessed. Specifically, we look to overcome the complex dependence of LSPG on the time discretization, and the exponential time growth of the error bounds for Galerkin and LSPG. We now describe the proposed windowed least-squares approach for this purpose.

In contrast to (1) Galerkin projection, which minimizes the (time-continuous) FOM ODE residual at a time instance, (2) LSPG projection, which minimizes the (time-discrete) FOM OΔE residual over a time step, and (3) ST-LSPG projection, which minimizes the FOM OΔE residual over the entire time interval, the proposed WLS approach sequentially minimizes the FOM ODE residual over arbitrarily defined *time windows*.

Windowed least-squares for general space–time trial subspaces: We begin by introducing a (potentially nonuniform) partition of the time domain $[0, T]$ into N_w non-overlapping windows $[t_s^n, t_f^n] \subseteq [0, T]$ of length $\Delta T^n := t_f^n - t_s^n$, $n = 1, \dots, N_w$ such that $t_s^1 = 0$, $t_f^{N_w} = T$, and $t_s^{n+1} = t_f^n$, $n = 1, \dots, N_w - 1$;

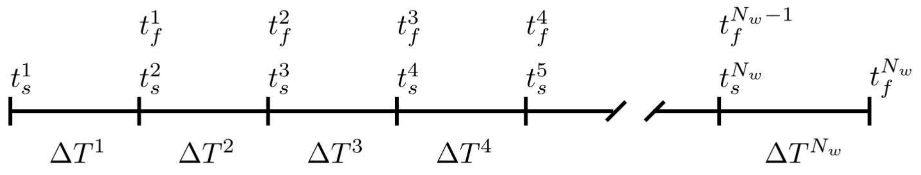


Figure 1: Partitioning of the time domain into time windows.

Fig. 1 depicts such a partitioning. Over the nth time window, we approximate the FOM ODE solution as $\tilde{x}^n(t) \approx x(t)$, $t \in [t_s^n, t_f^n]$, which is enforced to reside in the nth space–time trial subspace $\mathscr{S}\mathscr{T}^n$ such that

$$\tilde{x}^n \in \mathscr{S}\mathscr{T}^n \subseteq \mathbb{R}^N \otimes \mathscr{T}^n, \qquad n = 1, \dots, N_w, \tag{3.8}$$

where $\mathscr{T}^n$ denotes the set of (sufficiently smooth) real-valued functions acting on $[t_s^n, t_f^n]$ (i.e., $\mathscr{T}^n = \{f \,|\, f : [t_s^n, t_f^n] \to \mathbb{R}\}$). For notational purposes, we additionally define the spatial trial subspaces at the start of each window as

$$\mathscr{B}^n := \operatorname{span}(\{y(t_s^n) \,|\, y \in \mathscr{S}\mathscr{T}^n\}) \subseteq \mathbb{R}^N, \qquad n = 1, \dots, N_w, \tag{3.9}$$

such that $\tilde{x}^n(t_s^n) \in \mathscr{B}^n$. To outline WLS, we now define the objective functional over the nth window as

$$\begin{aligned} \mathscr{J}^n : y \mapsto \frac{1}{2} \int_{t_s^n}^{t_f^n} \left[\dot{y}(t) - f(y(t),t)\right]^T \mathbf{A}^n \left[\dot{y}(t) - f(y(t),t)\right] dt, \\ : \mathbb{R}^N \otimes \mathscr{T}^n \to \mathbb{R}_+, \end{aligned} \tag{3.10}$$

where $\mathbf{A}^n \equiv [\mathbf{W}^n]^T \mathbf{W}^n \in \mathbb{R}^{N \times N}$ denotes a symmetric positive semi-definite matrix that can enable hyper-reduction, for example.

The WLS approach sequentially computes approximate solutions $\tilde{x}^n \in \mathscr{S}\mathscr{T}^n$, $n = 1, \dots, N_w$, where $\tilde{x}^n$ is the solution to the minimization problem

$$\begin{aligned} &\underset{y \in \mathscr{S}\mathscr{T}^n}{\text{minimize}} \; \mathscr{J}^n(y), \\ &\text{subject to } \; y(t_s^n) = \begin{cases} \mathbb{P}^n(\tilde{x}^{n-1}(t_f^{n-1})) & n = 2, \dots, N_w, \\ \mathbb{P}^n(\mathbf{x}_0) & n = 1, \end{cases} \end{aligned} \tag{3.11}$$

where $\mathbb{P}^n : \mathbb{R}^N \to \mathscr{B}^n$ is a (spatial) projection operator onto the start of the nth trial space (e.g., ℓ^2-orthogonal projection operator).

We now define the trial subspaces $\mathscr{S}\mathscr{T}^n$ considered in this work. In particular, we introduce S-reduction trial subspaces tailored for this context. We note that space–time trial subspaces are investigated in the supporting publication for this work, [83].

S-reduction trial subspaces: In this context, the S-reduction trial subspace over the nth time window approximates the FOM ODE solution trajectory $x(t)$, $t \in [t_s^n, t_f^n]$ with $\tilde{x}^n \in \mathscr{S}\mathscr{T}_{\mathrm{S}}^n$, where

$$\mathscr{S}\mathscr{T}_{\mathrm{S}}^n := \mathscr{V}^n \otimes \mathscr{T}^n + \mathbf{x}_{\mathrm{ref}}^n \otimes \mathscr{O}^n \subseteq \mathbb{R}^N \otimes \mathscr{T}^n. \tag{3.12}$$

Here, the spatial trial subspaces $\mathscr{V}^n \subseteq \mathbb{R}^N$, $n = 1,\ldots,N_w$ satisfy $\mathscr{V}^n := \mathrm{Ran}(\mathbf{V}^n)$ with $\mathbf{V}^n \equiv [\mathbf{v}_1^n \cdots \mathbf{v}_{K^n}^n] \in \mathbb{V}_{K^n}(\mathbb{R}^N)$, the reference states $\mathbf{x}_{\mathrm{ref}}^n \in \mathbb{R}^N$, $n = 1,\ldots,N_w$ provide the affine transformation, and $\mathscr{O}^n \in \mathscr{T}^n$ is defined as $\mathscr{O}^n : \tau \mapsto 1$. Thus, at any time instance $t \in [t_s^n, t_f^n]$, the S-reduction trial subspace approximates the FOM ODE solution as

$$x(t) \approx \tilde{x}^n(t) = \mathbf{V}^n \hat{x}^n(t) + \mathbf{x}_{\mathrm{ref}}^n, \tag{3.13}$$

where $\hat{x}^n \in \mathbb{R}^{K^n} \otimes \mathscr{T}^n$ with $\hat{x}^n : \tau \mapsto \hat{x}^n(\tau)$ denotes the generalized coordinates over the nth time window.

Setting $\mathscr{S}\mathscr{T}^n \leftarrow \mathscr{S}\mathscr{T}_{\mathrm{S}}^n$ in the WLS minimization problem (4.11) and setting $\mathbb{P}^n$ to the ℓ^2-orthogonal projection operator implies that WLS with S-reduction space–time trial subspaces sequentially computes solutions $\hat{x}^n$, $n = 1,\ldots,N_w$ that satisfy

$$\begin{aligned} &\underset{\hat{y} \in \mathbb{R}^{K^n} \otimes \mathscr{T}^n}{\text{minimize}} \ \mathscr{J}^n(\mathbf{V}^n \hat{y} + \mathbf{x}_{\mathrm{ref}}^n \otimes \mathscr{O}^n), \\ &\text{subject to } \ \hat{y}(t_s^n) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{x}^{n-1}(t_f^{n-1}) + \mathbf{x}_{\mathrm{ref}}^{n-1} - \mathbf{x}_{\mathrm{ref}}^n) & n = 2,\ldots,N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}_{\mathrm{ref}}^1) & n = 1. \end{cases} \end{aligned} \tag{3.14}$$

Stationary conditions: We derive stationary conditions for optimization problem (4.14) via the Euler–Lagrange equations from the calculus of variations. To begin, we define the integrand appearing in the objective function $\mathscr{J}^n$ defined in Eq. (4.10) in terms of the generalized coordinates induced by the S-reduction subspace as

$$\begin{aligned} \mathscr{I}^n : (\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &\mapsto \frac{1}{2}\left[\mathbf{V}^n \hat{\mathbf{v}} - f(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\mathrm{ref}}^n, \tau)\right]^T \mathbf{A}^n \left[\mathbf{V}^n \hat{\mathbf{v}} - f(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\mathrm{ref}}^n, \tau)\right], \\ &: \mathbb{R}^{K^n} \times \mathbb{R}^{K^n} \times [t_s^n, t_f^n] \to \mathbb{R}_+. \end{aligned} \tag{3.15}$$

We also define the quantities[2]

$$\begin{aligned} \mathscr{I}_{\hat{\mathbf{v}}}^n : (\hat{y}, \hat{v}, \tau) &\mapsto \left[\frac{\partial \mathscr{I}^n}{\partial \hat{\mathbf{v}}}(\hat{y}(\tau), \hat{v}(\tau), \tau)\right]^T, \\ &: \mathbb{R}^{K^n} \otimes \mathscr{T}^n \times \mathbb{R}^{K^n} \otimes \mathscr{T}^n \times [t_s^n, t_f^n] \to \mathbb{R}^{K^n}, \end{aligned} \tag{3.16}$$

[2]We use numerator layout for the scalar-by-vector gradients.

$$\begin{aligned}\mathscr{I}^n_{\hat{\mathbf{y}}} : (\hat{y}, \hat{v}, \tau) &\mapsto \left[\frac{\partial \mathscr{I}^n}{\partial \hat{\mathbf{y}}}(\hat{y}(\tau), \hat{v}(\tau), \tau)\right]^T, \\ &: \mathbb{R}^{K^n} \otimes \mathscr{T}^n \times \mathbb{R}^{K^n} \otimes \mathscr{T}^n \times [t^n_s, t^n_f] \to \mathbb{R}^{K^n}.\end{aligned} \tag{3.17}$$

Using this notation, the Euler–Lagrange equations (see [83] for the derivation) over the nth time window for $t \in [t^n_s, t^n_f]$ are given by

$$\begin{aligned}&\mathscr{I}^n_{\hat{\mathbf{y}}}(\hat{x}^n, \dot{\hat{x}}^n, t) - \dot{\mathscr{I}}^n_{\hat{\mathbf{v}}}(\hat{x}^n, \dot{\hat{x}}^n, t) = 0, \\ &\hat{x}^n(t^n_s) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{x}^{n-1}(t^{n-1}_f) + \mathbf{x}^{n-1}_{\text{ref}} - \mathbf{x}^n_{\text{ref}}) & n = 2, \dots, N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}^1_{\text{ref}}) & n = 1, \end{cases} \\ &\mathscr{I}^n_{\hat{\mathbf{v}}}(\hat{x}^n, \dot{\hat{x}}^n, t^n_f) = 0.\end{aligned} \tag{3.18}$$

Solution approaches: The WLS system can be solved via two differing approaches: direct and indirect approaches. In a direct approach, the time-continuous optimization problem is transcribed into a time-discrete optimization problem, which can then be solved, e.g., via the Gauss–Newton method. The indirect approach, on the other hand, operates by solving the Euler–Lagrange equations outlined above. The Euler–Lagrange equations comprise a coupled two-point boundary value problem and can be solved, e.g., via the forward-backward sweep method. The supporting journal article for this work [83] investigates these two approaches in detail.

WLS Summary: The previous sections outlined the WLS approach for nonlinear model reduction. The approach sequentially minimizes the time-continuous full-order-model residual within a low-dimensional space–time trial subspace over time windows. We then briefly discussed two differing solution approaches: direct methods and indirect methods. In the supporting publication for this work, we additionally show that particular instances of the approach recover Galerkin, least-squares Petrov–Galerkin (LSPG), and space–time LSPG projection.

The Adjoint Petrov–Galerkin method: The WLS method described above comprises a promising approach for model reduction. However, it has a notable drawback: **solution techniques requires access to the transpose of the Jacobian of the full-order model velocity**. This is a result of WLS being a type of least-squares problem. For legacy codes, this requirement can make it very difficult to implement the WLS method (as well as LSPG) in an efficient manner.

The Adjoint Petrov–Galerkin method offers an alternative approach developing robust ROMs of nonlinear dynamical systems. APG is derived by decomposing the generalized coordinates of a dynamical system into a resolved coarse-scale set and an unresolved fine-scale set. A Markovian finite memory assumption within the Mori–Zwanzig formalism is then used to develop a reduced-order representation of the coarse-scales. This procedure leads to a closed reduced-order model that displays commonalities with the adjoint stabilization method used in finite elements. The formulation is shown to be equivalent to a Petrov–Galerkin method with a non-linear, time-varying test basis, thus sharing some similarities with the Least-Squares Petrov–Galerkin method. The following sections outline the APG approach.

Multiscale formulation: The APG method is derived via a multiscale formulation, which we now outline. We note that, in this section, we assume for simplicity that the reference state in the trial subspaces is set to zero, i.e., an affine trial subspace is not employed. First, recall that $x(t) \in \mathbb{R}^N$. We consider an orthogonal sum decompositions of this FOM space,

$$\mathbb{R}^N = \mathscr{V} \oplus \mathscr{V}', \tag{3.19}$$

with $\mathscr{V} \perp \mathscr{V}'$, $\dim(\mathscr{V}) = K$, $\dim(\mathscr{V}') = N - K$, and where

$$\begin{aligned} \mathbf{V} &\equiv \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_K \end{bmatrix}, & \mathscr{V} &:= \mathrm{Range}(\mathbf{V}), \\ \mathbf{V}' &\equiv \begin{bmatrix} \hat{\mathbf{v}}_1 \cdots & \mathbf{v}_{N-K} \end{bmatrix}, & \mathscr{V}' &:= \mathrm{Range}(\mathbf{V}'). \end{aligned}$$

The space $\mathscr{V}$ is referred to as the coarse-scale trial space, while $\mathscr{V}'$ is referred to as the fine-scale trial space. Note the following properties of the decomposition:

1. The coarse-scale space is a subspace of $\mathbb{R}^N$.
2. The fine-scale space is a subspace of $\mathbb{R}^N$.
3. The fine and coarse-scale subspaces do not overlap, i.e., $\mathscr{V} \cap \mathscr{V}' = \{\mathbf{0}\}$.

The coarse and fine-scale states are then defined as

$$\tilde{x}(t) := \sum_{i=1}^{K} \mathbf{v}_i \hat{x}_i(t) \qquad x'(t) := \sum_{i=1}^{N-K} \hat{\mathbf{v}}_i \hat{x}_i(t),$$

with $\tilde{x} : [0,T] \to \mathscr{V}$, $x' : [0,T] \to \mathscr{V}'$, $\hat{x} : [0,T] \to \mathbb{R}^K$, and $\hat{x}' : [0,T] \to \mathbb{R}^{N-K}$.

With the multiscale decomposition, the FOM ODE (**??**) can be written as the coupled system

$$\mathbf{V}^T \frac{d\tilde{x}}{dt}(t) = \mathbf{V}^T f(\tilde{x} + x', t)$$

$$\mathbf{V}'^T \frac{dx'}{dt}(t) = \mathbf{V}'^T f(\tilde{x} + x', t).$$

For simplicity, we assume $x(0) \in \mathscr{V}$ such that $\tilde{x}(0) = x(0)$ and $x'(0) = 0$. The objective of APG is to develop a closed coarse scale equation, i.e., for the resolved scales $\tilde{x}$. To this end, the optimal prediction framework/Mori–Zwanzig formalism is employed.

The Mori–Zwanzig formalism and the Adjoint Petrov–Galerkin method: The optimal prediction framework formulated by Chorin et al. [84, 85, 86], which is a (significant) reformulation of the Mori–Zwanzig (MZ) formalism of statistical mechanics, is a model order reduction tool that can be used to develop representations of the impact of the fine-scales on the coarse-scale dynamics. In this section, the optimal prediction framework is used to derive a compact approximation to the impact of the fine-scale POD modes on the evolution of the coarse-scale POD modes.

In the supporting publication for this work, [Section 3.3][46], the Mori–Zwanzig formalism is used to develop the following closed-form equation for the coarse-scales

$$\frac{d\hat{x}}{dt}(t) = \mathbf{V}^T f(\mathbf{V}\hat{x}(t),t) + \int_0^t K(\hat{x}(t-s),s)ds, \tag{3.20}$$

where $K : \mathbb{R}^K \times [0,T] \to \mathbb{R}^K$ is referred to as the **memory kernel**. The direct evaluation of the memory term in Eq. (4.20) is, in general, computationally intractable. To gain a reduction in computational cost, an approximation to the memory must be devised. A variety of such approximations exist, and here we outline the τ-model [87, 88]. The τ-model can be interpreted as the result of assuming that the memory is driven to zero in finite time and approximating the integral with a quadrature rule. This can be written as a two-step approximation,

$$\int_0^t K(\hat{x}(t-s),s)ds \approx \int_{t-\tau}^t K(\hat{x}(t-s),s)ds \approx \tau K(\hat{x}(t),0).$$

Here, $\tau \in \mathbb{R}_0^+$ is a stabilization parameter that is sometimes referred to as the "memory length." It is typically static and user-defined, though methods of dynamically calculating it have been developed in [87].

The term $K(\hat{x}(t),0)$ can be shown to be [89]

$$K(\hat{x}(t),0) = \mathbf{V}^T J[\tilde{x}(t)]\Pi' f(\tilde{x}(t),t),$$

where $J \equiv \partial f/\partial y$ and where $\Pi' : \mathscr{V} \to \mathscr{V}'$ is the "orthogonal projection operator," defined as $\Pi' \equiv \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)$. We define the corresponding coarse-scale projection operator as $\Pi : \mathscr{V} \to \tilde{\mathscr{V}}$ with $\Pi \equiv \mathbf{V}\mathbf{V}^T$. The coarse-scale equation with the τ-model reads,

$$\mathbf{V}^T\left(\frac{d}{dt}\tilde{x}(t) - f(\tilde{x}(t),t)\right) = \tau\mathbf{V}^T J[\tilde{x}(t)]\Pi' f(\tilde{x}(t)). \tag{3.21}$$

Equation (4.21) provides a closed equation for the evolution of the coarse-scales. The left-hand side of Eq. (4.21) is the standard Galerkin ROM, and the right-hand side can be viewed as a subgrid-scale model.

When compared to existing methods, the inclusion of the τ-model leads to a method that is analogous to a non-linear formulation of the *adjoint* stabilization technique developed in the finite element community. The "adjoint" terminology arises from writing Eq. (4.21) in a Petrov–Galerkin form,

$$\left[\left(\mathbf{I} + \tau\Pi'^T J^T[\tilde{x}(t)]\right)\mathbf{V}\right]^T\left(\frac{d}{dt}\tilde{x}(t) - f(\tilde{x}(t),t)\right) = \mathbf{0}. \tag{3.22}$$

It is seen that Eq. (4.22) involves taking the inner product of the coarse-scale ODE with a test-basis that contains the adjoint of the coarse-scale Jacobian. Unlike GLS stabilization, adjoint stabilization can be derived from the multiscale equations [90]. Due to the similarity of the proposed method with adjoint stabilization techniques, as well as the LSPG terminology, the complete ROM formulation will be referred to as the Adjoint Petrov–Galerkin (APG) method.

Adjoint Petrov–Galerkin method summary The previous section outlined the Adjoint Petrov–Galerkin method. As opposed to the WLS approach, which develops a stabilized ROM formulation via a *least-squares* formulation, APG formulates a stabilized ROM approach via a closure modeling viewpoint. An appealing feature of APG, as opposed to WLS, is that it does not require an evaluation of the action of the transpose of a Jacobian on a vector. This eases the computational burden associated with the APG approach. In the supporting reference [46], we investigate commonalities between APG and existing stabilization techniques. We discover that APG displays similarities with the adjoint stabilization method from stabilized finite elements. Theoretical error analysis demonstrates that the APG method can have a lower *a priori* error bound than the Galerkin method, and numerical experiments on the compressible Navier–Stokes equations demonstrate the utility of the method.

Time-series machine learning error models: Both the WLS and APG approaches generate approximate solutions, $\tilde{x}^n \approx x^n$, $n = 1, \ldots, N_t$. As a result, it is critical to quantify the error incurred by the ROM. This work developed the time-series machine learning error models (TMLEM) for this purpose [91]. A complete presentation of TMLEM is presented in the supporting publication [91], and here we highlight the important aspects.

TMLEM seeks to generate an approximation for two types of errors:

1. The normed state error: $\delta_x^N \equiv \|x^n - \tilde{x}^n\|_2^2$.
2. A quantity of interest (QoI) error: $\delta_s^N \equiv s(x^N) - s(\tilde{x}^N)$, where $s : \mathbb{R}^N \to \mathbb{R}$ is a state-to-observable map.

TMLEM generates approximations to the errors via *recursive regression functions*. Specifically, we model the error as $\hat{\delta}^n (\approx \delta^n)$ where

$$\hat{\delta}^n = \hat{f}(\rho^n, h^n) + \hat{\delta}_\varepsilon^n, \; n = 1, \ldots, N_t, \tag{3.23}$$

where $\hat{f}$ is a regression function, ρ^n are *features*, h^n are *latent variables*, and $\hat{\delta}_\varepsilon^n$ is a noise model. Critically, we enforce the latent variables to be governed by a recursion relationship,

$$h^n = g(\rho^n, h^{n-1}, f(\rho^{n-1}, h^{n-1})). \tag{3.24}$$

The system defined by Eqns (4.23) and (4.24) defines a recursive regression function. Depending on the selection of the recursion functions $\hat{f}$ and g, one can recover auto-regressive models, recurrent neural networks, etc. We refer the reader to the supporting publication [91, Section 3] for more details.

The framework proposed by TMLEM comprises the following steps:

Step 1 **Feature engineering.** Select features that are inexpensive to compute, informative of the error such that a low-noise-variance model can be constructed, and low dimensional such that less training data are required for the resulting model to generalize. To satisfy these requirements, we engineer residual-based features informed by classical error-quantification methods. A detailed description of residual based features is provided in [Section 3.1][91].

Step 2 **Data generation.** After selecting candidate features, generate both training and test data in the form of a set of response–feature pairs. Constructing these data requires computing *both* the FOM solutions and approximate solutions for the selected parameter instances.

Step 3 **Train the deterministic regression-function model.** Define candidate functional forms for the functions $\hat{f}$ and g, compute model parameters by (approximately) minimizing a loss function defined on the training data, and perform hyperparameter selection using (cross-)validation.

Step 4 **Train the stochastic noise model.** Finally, define candidate functional forms for the function, and compute model parameters via maximum likelihood estimation on a training set independent from that used to train the deterministic regression function. In this work, we employ a subset of the test set used to evaluate the deterministic regression function for this purpose.

Time-series machine learning error models summary The previous section outlined the time-series machine learning error models (TMLEM). Critically, TMLEM advances the state-of-the-art for error modeling of approximate solutions for dynamical systems by building error models capable of capturing the recursive nature of the error in approximate solutions of such systems. In the supporting manuscript for this work [91], we investigate numerous types of features, regression functions, and statistical error models. We additionally present numerical experiments for approximate solutions to several benchmark problems. We refer the reader to the following section for the high-level findings from TMLEM.

Results and discussion:

Comprehensive theoretical and numerical results for the methods proposed in this work are presented in the supporting manuscripts [83, 46, 91]. Here, we provide a high-level summary of these results.

Windowed-least squares model reduction:

In the supporting manuscript [83], both theoretical analysis and numerical experiments are performed for the WLS method. In our theoretical analysis, we make several findings:

1. [83, Theorem 5.1]: WLS with S-reduction trial subspaces recovers the Galerkin method in the limit of the window size $\Delta T \to 0$. This is consistent with previous results showing that LSPG recovers Galerkin projection in the limit of the time step going to zero.

2. [83, Theroem 5.3]: In the case that the window size comprises the entire time domain, the error incurred by the WLS approximation is bounded by the FOM residual evaluated at the WLS solution. This finding demonstrates that WLS can overcome the exponentially growing error bounds that appear in LSPG and Galerkin projection.

In the case of numerical experiments, several benchmark problems from compressible fluid dynamics are investigated [Section 6][83]. These results demonstrate the following:

1. Increasing the window size over which the residual is minimized led to more physically relevant solutions. Specifically, we observed that as the window size over which the residual was minimized grew, WLS led to less oscillatory solutions.

2. Increasing the window size over which the residual is minimized does not necessary lead to a lower space–time error as measured by the ℓ^2-norm. We observed that minimizing the residual over an intermediary window size led to the lowest space–time error in the ℓ^2-norm.

3. WLS displays time-step convergence: both the ℓ^2-error and residual norm decreased and displayed time-step convergence as the time step decreased. This is in contrast to LSPG.

4. WLS incurs a higher computational cost than LSPG. In the context of the direct solution approach, this increased cost is due to the increased expense of forming and solving the least-squares problem associated with larger window sizes. In the context of indirect methods, this increased cost is a result of the increased number of iterations of the forward backward sweep method.

Adjoint Petrov–Galerkin model reduction: Comprehensive theoretical and numerical results are presented for the APG method in the supporting manuscript [46]. The high-level findings are as follows:

1. Theoretical error analysis demonstrates that APG is equipeed with *a priori error bounds* that may be smaller than those associated with the Galerkin method.

2. Computational cost analysis shows that the APG ROM is twice as expensive as the G ROM for a given time step, for both explicit and implicit time integrators. In the implicit case, the ability of the APG ROM to make use of Jacobian–Free Newton–Krylov methods suggests that it may be more efficient than the LSPG ROM.

3. Numerical evidence from ROMs of compressible flow problems demonstrate that APG is more accurate and stable than the G ROM on problems of interest. Improvements over the LSPG ROM are observed in most cases. An analysis of the computational cost shows that the APG method can lead to lower errors than the LSPG and G ROMs for the same computational cost.

Time-series machine learning error models: Comprehensive numerical results are presented for the TMLEM method in the supporting manuscript [91]. The high-level findings are as follows:

1. The error models obtained from TMLEM yield highly accurate predictions for the error, with $r^2 > 0.999$.
2. The best overall regression functions comprised methods obtained from the T-MLEM framework with an LSTM regression-function model. These methods outperformed, e.g., autoregressive models.
3. Feature-engineering methods that employ residual-based features yielded the best performance.
4. The Laplacian stochastic noise model provided the best statistical model for the prediction noise.

Anticipated outcomes and impacts:

The present work advanced the state-of-the-art in nonlinear model reduction by developing the WLS method, APG method, and TMLEM error models. We expect for the present work to yield an enhanced ability for performing many-query analyses, such as uncertainty quantification and optimization, as follows:

1. WLS and APG provide the technology for increasingly robust model reduction methods for nonlinear dynamical systems. In particular, the WLS approach yields accurate and stable solutions in cases where existing state-of-the-art approaches experience numerical blowup.
2. The time-series machine learning error models equip analysts with the ability to *quantify* the error incurred in the ROM solution. This aspect is critical for understanding when the ROM solution is accurate and reliable.
3. The WLS approach has been implemented in Sandia's open-source reduced-order modeling code, Pressio (https://github.com/Pressio/pressio). This effort promotes longevity of the research, and allows WLS to be applied to novel computational problems encountered across national labs and academia.

Additionally, this work led to the following conference and journal publications:

1. Parish, E. J., and Carlberg, K., "Windowed least-squares model-reduction for dynamical systems", Journal of Computational Physics, July 2020, (in revision).
2. Parish, E. J., and Carlberg, K., "Time-series Machine Learning Error Models for Approximate Solutions to Dynamical Systems," Computer Methods in Applied Mechanics and Engineering, June 2020.
3. Parish, E. J., Wentland, C., and Duraisamy, K. "The Adjoint Petrov—Galerkin method for nonlinear model reduction", Computer Methods in Applied Mechanics and Engineering, June 2020.

4. Parish, E.J., and Rizzi, F., "Windowed least-squares model reduction in Pressio", Software@Sandia Series, Aug. 2020.

5. Parish, E. J., "Model reduction via time-continuous least-squares residual minimization", APS Division of Fluid Dynamics, Seattle, WA., Nov. 2019

6. Parish, E. J., and Carlberg, K., "Time-series Machine Learning Error Models for Approximate Solutions to Dynamical Systems," United States Congress of Computational Mechanics, Austin, TX., Jun. 2019.

7. Parish, E.J. "Data-informed Reduced-Order Models with Memory Effects," SIAM Computational Science and Engineering, Spokane, WA, 2019.

8. Parish, E.J., "Machine Learning Closure Modeling for Reduced-Order Models of Dynamical Systems," Bay Area Scientific Computing Day, Livermore, CA., 2018.

Conclusions: Projection-based reduced-order models (ROMs) comprise a promising set of data-driven approaches for accelerating the simulation of high-fidelity numerical simulations. When applied to nonlinear dynamical systems, however, standard projection-based ROM approaches can suffer from a lack of stability and/or accuracy. Additionally, these methods often lack reliable error indicators. This work advanced the state-of-the-art in nonlinear model reduction by developing the windowed least-squares (WLS) method and the Adjoint Petrov–Galerkin method for nonlinear model reduction, along with time-series machine learning error models to approximate the error incurred by the ROM. Numerical results demonstrated that both the windowed least-squares and Adjoint Petrov–Galerkin methods can yield improvements over state of the art methods. Specifically, windowed least-squares ROMs overcome the time step and time scheme dependence that the least-squares Petrov–Galerkin approach is subject to. Additionally, numerical experiments on the compressible Navier–Stokes equations demonstrate that the windowed-least-squares approach can yield more stable and accurate solutions than the LSPG. As the windowed least-squares method is a residual minimization approach, however, its practical utility may be limited due to the requirement that one is able to compute the action of the Jacobian transpose on a vector. For legacy and extreme-scale codes, this may not be possible. As an alternative approach, the Adjoint Petrov–Galerkin method does not require access to a Jacobian transpose. The adjoint Petrov–Galerkin method is derived via a multiscale modeling formulation, and yields substantial improvements over the least-squares Petrov–Galerkin method and Galerkin method in terms of accuracy for a given computational time. Finally, the time-series machine learning error models developed in this work enable analysts to quantify the error incurred by the ROM; this is critical for certification.

The present work led to numerous tangible outcomes. Specifically, the work has led to three peer reviewed journal publications (one is in revision at the time of writing this document), as well as four conference presentations (three additional presentations were canceled due to COVID-19). Additionally, the present work has led to the development of the windowed least-squares ROM within Sandia's open-source reduced-order modeling code, Pressio. The integration of WLS into Pressio promotes longevity of the research and enables WLS to be applied to a wide class of problems encountered across national labs.

The present work has additionally raised several questions that should be topics of future work. First, one of the challenges encountered in this work was the computational cost of residual minimization methods, such as WLS and LSPG, due to the fact that one has to compute the action of the Jacobian transpose on a vector. In practice, this is a computationally expensive task, and is the bottleneck of WLS and LSPG. In order to advance these methods, techniques for accelerating this step need to be investigated. Second, in terms of the Adjoint Petrov–Galerkin model, future work should focus on the selection of the stabilization parameter, τ. For systems with disparate scales, such as those encountered in reacting flows, it has been observed that a scalar τ may not be sufficient [40]. This is inline with results from the variational multiscale community, where vector values of τ are chosen for each equation.

References

[1] Benner, P., Gugercin, S., and Willcox, K., "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM Review*, Vol. 57, No. 4, 2015, pp. 483–531.

[2] Moore, B., "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, Vol. 26, No. 1, February 1981, pp. 17–32.

[3] Mullis, C. T. and Roberts, R. A., "Synthesis of Minimum Roundoff Noise Fixed Point Digital Filters," *IEEE Transactions on Circuits and Systems*, Vol. 23, No. 9, 1976, pp. 551–562.

[4] Gugercin, S., Antoulas, A., and Beattie, C., "$\mathscr{H}_2$ Model Reduction for Large-Scale Linear Dynamical Systems," *SIAM Journal on Matrix Analysis and Applications*, Vol. 30, No. 2, 2008, pp. 609–638.

[5] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *Int. J. Numer. Methods Eng.*, Vol. 86, 2011, pp. 155–181.

[6] 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.

[7] LeGresley, P. and Alonso, J., *Airfoil design optimization using reduced order models based on proper orthogonal decomposition*.

[8] LeGresley, P. and Alonso, J., *Investigation of non-linear projection for POD based reduced order models for Aerodynamics*.

[9] LeGresley, P. and Alonso, J., *Dynamic domain decomposition and error correction for reduced order models*.

[10] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Model Reduction for Large-Scale Systems with High-Dimensional Parametric Input Space," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 3270–3288.

[11] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Parametric Reduced-Order Models for Probabilistic Analysis of Unsteady Aerodynamic Applications," *AIAA Journal*, Vol. 46, No. 10, 2008, pp. 2520–2529.

[12] Rovas, D. V., *Reduced-basis output bound methods for parametrized partial differential equations*, Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[13] Carlberg, K., *Model reduction of nonlinear mechanical systems via optimal projection and tensor approximation*, Ph.D. thesis, Stanford University, 2011.

[14] Bui-Thanh, T., *Model-constrained optimization methods for reduction of parameterized large-scale systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[15] Abgrall, R. and Crisovan, R., "Model reduction using L1-norm minimization as an application to nonlinear hyperbolic problems," *International Journal for Numerical Methods in Fluids*, Vol. 87, No. 12, 2018, pp. 628–651.

[16] Choi, Y. and Carlberg, K., "Space–Time least-squares Petrov–Galerkin projection for nonlinear model reduction," *SIAM Journal on Scientific Computing*, Vol. 41, No. 1, 2019, pp. A26–A58.

[17] Constantine, P. G. and Wang, Q., "Residual Minimizing Model Interpolation for Parameterized Nonlinear Dynamical Systems," *SIAM J. Sci. Comput.*, 2012.

[18] Urban, K. and Patera, A. T., "A new error bound for reduced basis approximation of parabolic partial differential equations," *Comptes Rendus Mathematique*, Vol. 350, No. 3, 2012, pp. 203 – 207.

[19] Yano, M., Patera, A. T., and Urban, K., "A space-time hp-interpolation-based certified reduced basis method for Burgers equation," *Mathematical Models and Methods in Applied Sciences*, Vol. 24, No. 09, 2014, pp. 1903–1935.

[20] Baumann, M., Benner, P., and Heiland, J., "Space-time Galerkin POD with application in optimal control of semilinear partial differential equations," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. A1611–A1641.

[21] Rowley, C. W., Colonius, T., and Murray, R. M., "Model reduction for compressible flows using POD and Galerkin projection," *Physica D: Nonlinear Phenomena*, Vol. 189, No. 1-2, 2004, pp. 115–129.

[22] Kalashnikova, I., Arunajatesan, S., Barone, M. F., van Bloemen Waanders, B. G., and Fike, J. A., "Reduced Order Modeling for Prediction and Control of Large-Scale Systems," Report SAND2014-4693, Sandia, May 2014.

[23] Chan, J., "Entropy stable reduced order modeling of nonlinear conservation laws," *arXiv e-print*, , No. arXiv:1909.09103, 2019.

[24] Carlberg, K., "Adaptive *h*-refinement for reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 5, 2015, pp. 1192–1210.

[25] Peherstorfer, B. and Willcox, K., "Online adaptive model reduction for nonlinear systems via low-rank updates," *J. Sci. Comput.*, Vol. 37, 2015, pp. A2123–A2150.

[26] Etter, P. and Carlberg, K., "Online adaptive basis refinement and compression for reduced-order models via vector-space sieving," *arXiv e-print*, , No. 1902.10659, 2019.

[27] Balajewicz, M., Tezaur, I., and Dowell, E., "Minimal subspace rotation on the Stiefel manifold for stabilization and enhancement of projection-based reduced order models for the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 321, 2016, pp. 224–241.

[28] Carlberg, K., Choi, Y., and Sargsyan, S., "Conservative model reduction for finite-volume models," *Journal of Computational Physics*, Vol. 371, 2018, pp. 280–314.

[29] Lall, S., Krysl, P., and Marsden, J. E., "Structure-preserving model reduction for mechanical systems," *Physica D: Nonlinear Phenomena*, Vol. 184, No. 1, 2003, pp. 304 – 318, Complexity and Nonlinearity in Physical Systems – A Special Issue to Honor Alan Newell.

[30] Carlberg, K., Tuminaro, R., and Boggs, P., "Preserving Lagrangian structure in nonlinear model reduction with application to structural dynamics," *SIAM J. Sci. Comput.*, Vol. 37, No. 2, 2015, pp. B153—B184.

[31] Beattie, C. and Gugercin, S., "Structure-preserving model reduction for nonlinear port-Hamiltonian systems," *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, IEEE, 2011, pp. 6564–6569.

[32] Chaturantabut, S., Beattie, C., and Gugercin, S., "Structure-preserving model reduction for nonlinear port-Hamiltonian systems," *SIAM Journal on Scientific Computing*, Vol. 38, No. 5, 2016, pp. B837–B865.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

[33] Farhat, C., Avery, P., Chapman, T., and Cortial, J., "Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency," *International Journal for Numerical Methods in Engineering*, Vol. 98, No. 9, 2014, pp. 625–662.

[34] San, O. and Iliescu, T., "A stabilized proper orthogonal decomposition reduced-order model for large scale quasigeostrophic ocean circulation," *Adv Comput Math*, Vol. 41, No. 1, 2015, pp. 1289–1319.

[35] San, O. and Iliescu, T., "Proper Orthogonal Decomposition Closure Models for Fluid Flows: Burgers Equation," *Comput. Methods Appl. Mech. Engrg*, Vol. 5, No. 3, 2014, pp. 217–237.

[36] Iliescu, T. and Wang, Z., "Variational Multiscale Proper Orthogonal Decomposition: Navier–Stokes Equations," *Numerical Methods for Partial Differential Equations*, Vol. 30, No. 2, 2014, pp. 641–663.

[37] Bergmann, M., Bruneaua, C., and Iollo, A., "Enablers for robust POD models," *Journal of Computational Physics*, Vol. 228, No. 1, 2009, pp. 516–538.

[38] Caiazzo, A., Iliescu, T., John, V., and Schyschlowa, S., "A numerical investigation of velocity–pressure reduced order models for incompressible flows," *Journal of Computational Physics*, Vol. 259, 2014, pp. 598 – 616.

[39] Wang, Z., *Reduced-Order Modeling of Complex Engineering and Geophysical Flows: Analysis and Computations*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2012.

[40] Wentland, C. R., Huang, C., and Duraisamy, K., *Closure of reacting flow reduced-order models via the Adjoint Petrov-Galerkin Method*.

[41] Wang, Q., Ripamonti, N., and Hesthaven, J. S., "Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism," *Journal of Computational Physics*, 2019.

[42] San, O. and Maulik, R., "Neural network closures for nonlinear model order reduction," *Advances in Computational Mathematics*, Vol. 44, No. 6, Dec 2018, pp. 1717–1750.

[43] Bochev, P. and Gunzburger, M., "Finite Element Methods of Least-Squares Type," *SIAM Review*, Vol. 40, No. 4, 1998, pp. 789–837.

[44] Gunzburger, M. D. and Bochev, P. B., *Least-Squares Finite Element Methods*, Springer, New York, NY, 2009.

[45] Carlberg, K., Barone, M., and Antil, H., "Galerkin v. least-squares Petrov-Galerkin projection in nonlinear model reduction," *Journal of Computational Physics*, Vol. 330, 2017, pp. 693–734.

[46] Parish, E., Wentland, C., and Duraisamy, K., "The Adjoint Petrov–Galerkin method for non-linear model reduction," *Computer Methods in Applied Mechanics and Engineering*, 10 2018, pp. 50.

[47] Paraschivoiu, M., Peraire, J., and Patera, A. T., "A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 150, No. 1-4, 1997, pp. 289–312.

[48] Paraschivoiu, M. and Patera, A. T., "A hierarchical duality approach to bounds for the outputs of partial differential equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 158, No. 3-4, 1998, pp. 389–407.

[49] Maday, Y. and Patera, A. T., "Numerical analysis of *a posteriori* finite element bounds for linear functional outputs," *Mathematical Models and Methods in Applied Sciences*, Vol. 10, No. 5, 2000, pp. 785–799.

[50] C. Prud'Homme and D.V. Rovas and K. Veroy and L. Machiels and Y. Maday and A. Patera and G. Turinici, "Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations," *Proceedings SMA Symposium*, 2002.

[51] Prud'Homme, C., Rovas, D., Veroy, K., Machiels, L., Maday, Y., Patera, A., and Turinici, G., "Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods," *Journal of Fluids Engineering*, Vol. 124, No. 1, 2001, pp. 70–80.

[52] Grepl, M. and Patera, A., "A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations," *ESAIM*, Vol. 39, No. 1, 2005, pp. 157–181.

[53] Rovas, D., Machiels, L., and Maday, Y., "Reduced-basis output bound methods for parabolic problems," *IMA Journal of Numerical Analysis*, Vol. 26, No. 423-445, 2006.

[54] Bui-Thanh, T. and Wilcox, K. and Ghattas, O., "Parametric Reduced-Order Models for Probabilistic Analysis of Unsteady Aerodynamic Applications," *AIAA Journal*, Vol. 46, No. 10, October 2008, pp. 2520–2529.

[55] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Model reduction for large-scale systems with high-dimensional parametric input space," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 3270–3288.

[56] Hinze, M. and Kunkel, M., "Residual based sampling in POD model order reduction of drift–diffusion equations in parametrized electrical networks," *ZAMM-Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 92, No. 2, 2012, pp. 91–104.

[57] Wu, Y. and Hetmaniuk, U., "Adaptive training of local reduced bases for unsteady incompressible Navier–Stokes flows," *International Journal for Numerical Methods in Engineering*, Vol. 103, No. 3, 2015, pp. 183–204.

[58] Yano, M. and Patera, A. T., "An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs," *Computer Methods in Applied Mechanics and Engineering*, 2018.

[59] Zahr, M. J. and Farhat, C., "Progressive construction of a parametric reduced-order model for PDE-constrained optimization," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 5, 2015, pp. 1111–1135.

[60] Zahr, M., *Adaptive model reduction to accelerate optimization problems governed by partial differential equations*, Ph.D. thesis, Stanford University, 2016.

[61] Zahr, M., Carlberg, K., and Kouri, D., "An efficient, globally convergent method for optimization under uncertainty using adaptive model reduction and sparse grids," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 6, No. 3, 2019, pp. 877–912.

[62] Lu, J. C.-C., *An a posteriori Error Control Framework for Adaptive Precision Optimization using Discontinuous Galerkin Finite Element Method*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.

[63] Ackmann, J., Moarotzke, J., and Korn, P., "Stochastic goal-oriented error estimation with memory," *Journal of Computational Physics*, Vol. 348, No. 1, 2017, pp. 195–219.

[64] Venditti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204 – 227.

[65] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40 – 69.

[66] Kennedy, M. C. and O'Hagan, A., "Bayesian Calibration of Computer Models," *J.R. Statist. Soc. B*, Vol. 63, No. 3, 2001, pp. 425–464.

[67] Huang, D., Allen, T. T., Notz, W. I., and Miller, R. A., "Sequential kriging optimization using multiple-fidelity evaluations," *Structural and Multidisciplinary Optimization*, Vol. 32, No. 5, 2006.

[68] Eldred, M. S., Giunta, A. A., Collis, S. S., Alexandrov, N. A., and Lewis, R. M., "Second-order corrections for surrogate-based optimization with model hierarchies," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2004.

[69] March, A. and Willcox, K., "Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives," *AIAA Journal*, Vol. 50, No. 5, 2012.

[70] Gano, S. E., Renaud, J. E., and Sanders, B., "Hybrid variable fidelity optimization by using a kriging-based scaling function," *AIAA Journal*, Vol. 43, No. 11, 2005.

[71] Moosavi, A., Ştefănescu, R., and Sandu, A., "Multivariate predictions of local reduced-order-model errors and dimensions," *International Journal for Numerical Methods in Engineering*, Vol. 113, No. 3, 2018, pp. 512–533.

[72] Ştefănescu, R., Moosavi, A., and Sandu, A., "Parametric domain decomposition for accurate reduced order models: Applications of MP-LROM methodology," *Journal of Computational and Applied Mathematics*, Vol. 340, 2018, pp. 629 – 644.

[73] Drohmann, M. and Carlberg, K., "The ROMES Method for Statistical Modeling of Reduced-Order-Model Error," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 3, 2015, pp. 116–145.

[74] Freno, B. and Carlberg, K., "Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 348, 2019, pp. 250–296.

[75] Trehan, S., Carlberg, K., and Durlofsky, L. J., "Error estimation for surrogate models of dynamical systems using machine learning," *International Journal for Numerical Methods in Engineering*, Vol. 112, No. 12, 2017, pp. 1801–1827.

[76] Pagani, S., Manzoni, A., and Quarteroni, A., "Efficient State/Parameter Estimation in Nonlinear Unsteady PDEs by a Reduced Basis Ensemble Kalman Filter," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 5, 2017, pp. 890–921.

[77] Berkooz, G., Holmes, P., and Lumley, J. L., "The proper orthogonal decomposition in the analysis of turbulent flows," *Annu. Rev. Fluid Mech.*, Vol. 25, 1993, pp. 539–575.

[78] Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G., "Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods," *Journal of Fluids Engineering*, Vol. 124, No. 1, 11 2001, pp. 70–80.

[79] Veroy, K., Prud'homme, C., Rovas, D., and Patera, A., *A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations*.

[80] Veroy, K. and Patera, A. T., "Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds," *International Journal for Numerical Methods in Fluids*, Vol. 47, No. 8-9, 2005, pp. 773–788.

[81] Ngoc Cuong, N., Veroy, K., and Patera, A. T., *Certified real-time solution of parametrized partial differential equations*, Springer Netherlands, Dordrecht, 2005, pp. 1529–1564.

[82] Rozza, G., Huynh, D. B. P., and Patera, A. T., "Reduced Basis Approximation and a Posteriori Error Estimation for Affinely Parametrized Elliptic Coercive Partial Differential Equations," *Archives of Computational Methods in Engineering*, Vol. 15, No. 3, May 2008, pp. 229.

[83] Parish, E. and Carlberg, K., "Windowed least-squares model reduction for dynamical systems," *arXiv e-print*, , No. 1910.11388, 2019.

[84] Chorin, A. J., Hald, O., and Kupferman, R., "Optimal prediction and the Mori-Zwanzig representation of irreversible processes," *Proc. Natl Acad. Sci.*, Vol. 97, No. (doi:10.1073/pnas.97.7.2968), 2000, pp. 2968–2973.

[85] Chorin, A. J., Hald, O. H., and Kupferman, R., "Optimal prediction with memory," *Physica D: Nonlinear Phenomena*, Vol. 166, No. 3, 2002, pp. 239 – 257.

[86] Chorin, A. and Hald, O., *Stochastic Tools in Mathematics and Science*, Springer-Verlag, 2005.

[87] Parish, E. J. and Duraisamy, K., "A dynamic subgrid scale model for Large Eddy Simulations based on the Mori-Zwanzig formalism," *Journal of Computational Physics*, Vol. 349, 2017, pp. 154–175.

[88] Barber, J. L., *Application of Optimal Prediction to Molecular Dynamics*, Ph.D. thesis, University of California, Berkeley, CA, 2004.

[89] Parish, E. J. and Duraisamy, K., "A Unified Framework for Multiscale Modeling Using Mori-Zwanzig and the Variational Multiscale Method," *CMAME*, , No. Submitted, 2018.

[90] Hughes, T. J., "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgridscale models, bubbles and the origins of stabilized methods," *Comput. Methods Appl. Mech. Eng.*, Vol. 127, 1995, pp. 387–401.

[91] Parish, E. and Carlberg, K., "Time-series machine-learning error models for approximate solutions to parameterized dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 365, 2020, pp. 112990.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Addendum We additionally provide the arXiv prints of the three reference manuscripts for this work.

Windowed least-squares model reduction for dynamical systems

Eric J. Parish and Kevin T. Carlberg[a]

[a]*Sandia National Laboratories, Livermore, CA*

Abstract

This work proposes a windowed least-squares (WLS) approach for model-reduction of dynamical systems. The proposed approach sequentially minimizes the time-continuous full-order-model residual within a low-dimensional space–time trial subspace over time windows. The approach comprises a generalization of existing model reduction approaches, as particular instances of the methodology recover Galerkin, least-squares Petrov–Galerkin (LSPG), and space–time LSPG projection. In addition, the approach addresses key deficiencies in existing model-reduction techniques, e.g., the dependence of LSPG and space–time LSPG projection on the time discretization and the exponential growth in time exhibited by *a posteriori* error bounds for both Galerkin and LSPG projection. We consider two types of space–time trial subspaces within the proposed approach: one that reduces only the spatial dimension of the full-order model, and one that reduces both the spatial and temporal dimensions of the full-order model. For each type of trial subspace, we consider two different solution techniques: direct (i.e., discretize then optimize) and indirect (i.e., optimize then discretize). Numerical experiments conducted using trial subspaces characterized by spatial dimension reduction demonstrate that the WLS approach can yield more accurate solutions with lower space–time residuals than Galerkin and LSPG projection.

1. Introduction

Simulating parameterized dynamical systems arises in many applications across science and engineering. In many contexts, executing a dynamical-system simulation at a single parameter instance—which entails the numerical integration of a system of ordinary differential equations (ODEs)—incurs an extremely large computational cost. This occurs, for example, when the state-space dimension is large (e.g., due to fine spatial resolution when discretizing a partial differential equation) and/or when the number of time instances is large (e.g., due to time-step limitations incurred by stability or accuracy considerations). When the application is time critical or many query in nature, analysts must replace such large-scale parameterized dynamical-system models (which we refer to as the full-order model) with a low-cost approximation that makes the application tractable.

Projection-based reduced-order models (ROMs) comprise one such approximation strategy. First, these techniques execute a computationally expensive *offline* stage that computes a low-dimensional *trial subspace* on which the dynamical-system state can be well approximated (e.g., by computing state "snapshots" at different time and parameter instances, by solving Lyapunov equations). Second, these methods execute an inexpensive *online* stage during which they compute approximations to the dynamical-system trajectory that reside on this trial subspace via, e.g., projection of the full-order model or residual minimization.

Model reduction for linear-time-invariant systems (and other well structured dynamical systems) is quite mature [7, 46, 48, 33], as system-theoretic properties (e.g., controllability, observability, asymptotic stability, $\mathcal{H}_2$-optimality) can be readily quantified and accounted for; this often results in reduced-order models that inherit such important properties. The primary challenge in developing reduced-order models for general nonlinear dynamical systems is that such properties are difficult to assess quantitatively. As a result, it is challenging to develop reduced-order models that preserve important dynamical-system properties, which often results in methods that yield trajectories that are inaccurate, unstable, or violate physical properties. To address this, researchers have pursued several directions that aim to imbue reduced-order models for nonlinear dynamical systems with properties that can improve robustness and accuracy. These efforts include residual-minimization approaches that equip the ROM solution with a notion of optimality [19, 21, 41, 43, 42, 13, 14, 54, 16, 12, 1]; space–time approaches that lead to error bounds that grow slowly in time [25, 26, 62, 68, 5]; "energy-based" inner products that ensure non-increasing entropy in the ROM solution [55, 36, 23]; basis-adaptation methods that improve the ROM's accuracy *a posteriori* [17, 51, 29], stabilizing subspace rotations that account for truncated modes *a priori* [2], structure-preserving methods that enforce conservation [20] or (port-)Hamiltonian/Lagrangian structure [39, 22, 6, 24, 31] in the ROM; and subgrid-scale modeling methods that aim to improve accuracy by addressing the closure problem [59, 58, 35, 8, 15, 66, 67, 65, 60]. We note that these techniques are often not mutually exclusive. Residual-minimizing and space–time approaches are the most relevant classes of methods for the current work and comprise the focus of the following review.

Residual-minimization methods in model reduction compute the solution within a low-dimensional trial subspace that minimizes the full-order-model residual.[1] Researchers have developed such residual-minimizing model-reduction methods for both static systems (i.e., systems without time-dependence) [41, 43, 42, 13, 54, 16, 12] and dynamical systems [12, 14, 16, 21, 19, 18]. In the latter category, Refs. [12, 14, 16, 21, 19] formulated the residual minimization problem for dynamical systems by sequentially minimizing the *time-discrete* full-order-model residual (i.e., the residual arising after applying time discretization) at each time instance on the time-discretization grid. This formulation is often referred to as the *least–squares Petrov–Galerkin* (LSPG) method. Ref. [18] performed detailed analyses of this formulation and examined its connections with Galerkin projection. Critically, this work demonstrated that (1) under certain conditions, LSPG can be cast as a Petrov–Galerkin projection applied to the time-continuous full-order-model residual, and (2) LSPG and Galerkin projection are equivalent in the limit as the time step goes to zero (i.e., Galerkin projection minimizes the time-instantaneous full-order-model residual). Numerous numerical experiments have demonstrated that LSPG often yields more accurate and stable solutions than Galerkin [12, 18, 21, 16, 50]. The common intuitive explanation for this improved performance is that, by minimizing the full-order-model residual over a finite time window (rather than time instantaneously), LSPG computes solutions that are more accurate over a larger part of the trajectory as compared to Galerkin.

However, LSPG has several notable shortcomings. First, LSPG exhibits a complex dependence on the time discretization. In particular, changing the time step (Δt) modifies both the time window over which LSPG minimizes the residual as well as the time-discretization error of the full-order model on which LSPG is based. As LSPG and Galerkin projection are equivalent in the limit of $\Delta t \to 0$, the accuracy of LSPG approaches the (sometimes poor) accuracy of Galerkin as the time step shrinks. For too-large a time step the accuracy of LSPG also degrades. It is unclear if this is due to the time-discretization error associated with enlarging the time step, or rather if it is due to the size of the window the residual is being minimized over. As a consequence, LSPG often yields the smallest error for an intermediate value of the time step (see, e.g., Ref. [18, Figure 9]); there is no known way to compute this optimal time step *a priori*. Second, as the LSPG approach performs sequential residual minimization in time, its *a posteriori* error bounds grow exponentially in time [18], and it is not equipped with any notion of optimality over the entire time domain of interest. As a result, LSPG is not equipped with *a priori* guarantees of accuracy or stability, even for linear time-invariant systems [12].

Researchers have pursued the development of space–time residual-minimization approaches [25, 26, 62, 68] to address the issues incurred by sequential residual minimization in time. Existing space–time approaches differ from the classic LSPG and Galerkin approaches in (1) the definition of the space–time trial subspace and (2) the definition of the residual minimization problem. First, space–time approaches leverage a space–time trial basis that characterizes both the spatial *and* temporal dependence of the state, classic "spatial" model reduction approaches such as LSPG and Galerkin leverage only a spatial trial basis that characterizes the spatial dependence of the state. Second, space–time residual minimization approaches compute the entire space–time trajectory of the state (within the low-dimensional space–time trial subspace) that minimizes the full-order-model residual over the entire time domain; Galerkin and LSPG sequentially compute instances of the state that either minimize the full-order model instantaneously (Galerkin) or over a time step (LSPG). The result of these differences is that space–time approaches yield a system of algebraic equations defined over all space and time, whose solution comprises a vector of (space–time) generalized coordinates; on the other hand, spatial-projection-only approaches generally associate with systems of ODEs whose solutions comprise *time-dependent* vectors of (spatial) generalized coordinates.

Space–time residual minimization approaches minimize the FOM residual over all of space and time and, as a result, yield models that are equipped with a notion of space–time optimality and *a priori* error bounds that grow more slowly in time. Further, space–time approaches reduce both the spatial and

[1]While we focus our review on residual-minimization approaches in the context of model reduction, we note that these approaches are intimately related to least-squares finite element methods [10, 34].

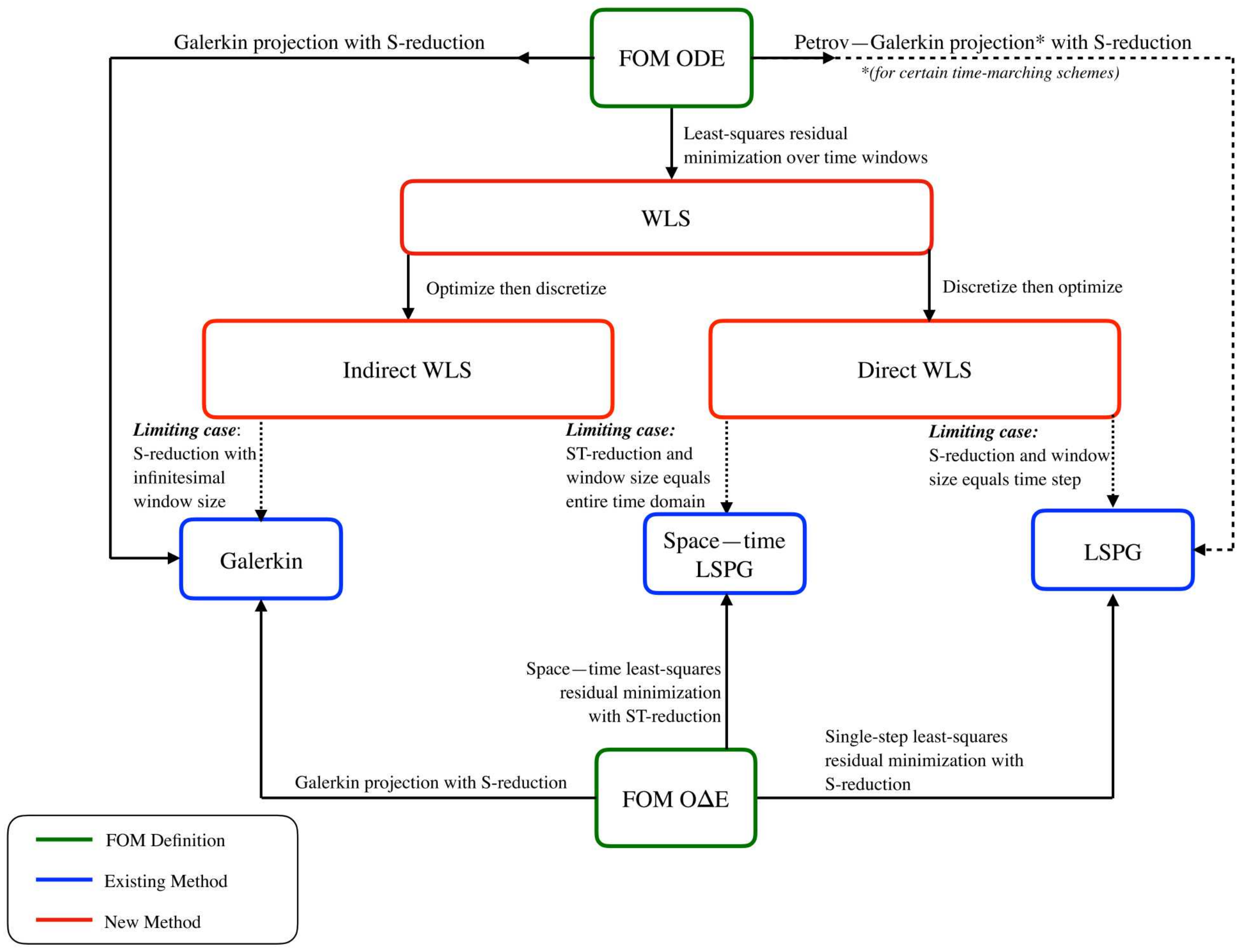
temporal dimensions of the full-order model, and thus promise cost savings over spatial-projection-only approaches. However, space–time techniques also suffer from several limitations. First, the computational cost of solving the algebraic system arising from space–time approaches scales cubically with the number of space–time degrees of freedom; in contrast, the computational cost incurred by standard spatial-projection-based ROMs is linear in the number of temporal degrees of freedom, as the attendant solvers can leverage the lower-block triangular structure of the system arising from the sequential nature of time evolution. As a result, solving the algebraic systems arising from space–time projection is generally intractable without applying hyper-reduction in time [25, 26]. Second, space–time residual minimization precludes future state prediction, as these methods employ space–time basis vectors defined over the entire time domain of interest, which must have been included in the training simulations.

The objectives of this work are to overcome the shortcomings of existing residual-minimizing methods, and to provide a unifying framework from which existing methods can be assessed. In essence, the proposed *windowed least-squares (WLS)* approach sequentially minimizes the FOM residual over a sequence of arbitrarily defined time windows. The method is characterized by three notable aspects. First, the method minimizes the *time-continuous* residual (i.e., that associated with the full-order model ODE). By adopting a time-continuous viewpoint, the formulation decouples the underlying temporal discretization scheme from the residual-minimization problem, thus addressing a key deficiency of both LSPG and space–time LSPG. Critically, time-continuous residual minimization also exposes two different solution methods: a *discretize-then-optimize* (i.e., direct) method, and an *optimize-then-discretize* (i.e., indirect) method. Second, the method sequentially minimizes the residual over arbitrarily defined *time windows* rather than sequentially minimizing the residual over time steps (as in LSPG) or over the entire time domain (as in space–time residual-minimization methods). This equips the method with additionally flexibility that enables it to explore more fine-grained tradeoffs between computational cost and error. Finally, WLS is formulated for two kinds of space–time trial subspaces: one that associates with spatial dimension reduction (as employed by traditional spatial-projection-based methods), and one that associates with space–time dimension reduction (as employed by space–time methods). The above attributes allow the WLS approach to be viewed as a generalization of existing model-reduction methods, as Galerkin, LSPG, and space–time LSPG projection correspond to specific instances of the formulation. Figure 1 depicts how the proposed WLS method provides a unifying framework from which these existing approaches can be derived.

The WLS approach can be viewed as a hybrid space–time method and displays commonalities with several related efforts. First, Ref. [12] briefly formulated a space–time least-squares ROM and connected this formulation with optimal control; specifically it mentioned the optimize-then-discretize vs. discretize-then-optimize approaches. This work did not fully develop this approach, eschewing it for sequential residual minimization in time (i.e., LSPG). The present work thus formally develops and extends several of the concepts put forth in Ref. [12]. Next, Ref. [26] developed a space–time residual minimization formulation for model interpolation. The present work distinguishes itself from Ref. [26] in that (1) this work also considers trial subspaces characterized by spatial dimension reduction only, and (2) we minimize the time-continuous FOM residual over arbitrary time windows. We note that, similar to the current work, Ref. [26] employs minimization of the time-continuous FOM residual as a starting point; as such, this work shares some thematic similarities with Ref. [26]. Lastly, Ref. [25] develops a space–time extension of LSPG projection that minimizes the time-discrete FOM residual over the entire time domain. The present work distinguishes itself from Ref. [25] in that (1) this work minimizes the time-continuous FOM residual, (2) this work minimizes this residual over arbitrary time windows, and (3) this work also considers trial subspaces associated with spatial dimension reduction only.

In summary, specific contributions of this work include:

1. The windowed least-squares (WLS) approach for dynamical-system model reduction. The approach sequentially minimizes the time-continuous full-order-model residual over arbitrary time windows.
2. Support of two space–time trial subspaces: one that associates with spatial dimension reduction and one that associates with spatial and temporal dimension reduction. The former case is of particular interest in the WLS context, as the stationary conditions are derived via the Euler–Lagrange equations and comprise a coupled two-point Hamiltonian boundary value problem containing a forward and backward system. The forward system, which is forced by an auxiliary costate, evolves the (spatial) generalized coordinates of the ROM in time. The backward system, which is forced by the time-continuous FOM residual evaluated about the ROM state, governs the dynamics of the costate.



Galerkin projection with S-reduction
FOM ODE
Petrov—Galerkin projection* with S-reduction
*(for certain time-marching schemes)
Least-squares residual minimization over time windows
WLS
Optimize then discretize
Discretize then optimize
Indirect WLS
Direct WLS
Limiting case: S-reduction with infinitesimal window size
Limiting case: ST-reduction and window size equals entire time domain
Limiting case: S-reduction and window size equals time step
Galerkin
Space—time LSPG
LSPG
Space—time least-squares residual minimization with ST-reduction
Single-step least-squares residual minimization with S-reduction
Galerkin projection with S-reduction
FOM OΔE
FOM Definition
Existing Method
New Method

Figure 1: Relationship diagram for the WLS approach for model reduction.

3. Derivation of two solution techniques: *discretize then optimize* (i.e., direct) and *optimize then discretize* (i.e., indirect).
4. Remarks and derivations of conditions under which the WLS approach recovers Galerkin, LSPG, and space–time LSPG.
5. Error analysis of the WLS approach using trial subspaces associated with spatial dimension reduction. This analysis demonstrates that, over a given window, the WLS ROMs error is bounded *a priori* by a combination of the error at the start of the window and the integrated FOM ODE residual evaluated at the FOM state projected onto the trial subspace.
6. Numerical experiments for trial subspaces associated with spatial dimension reduction, which demonstrate two key findings:
   - Minimizing the residual over a larger time window leads to more stable solutions with lower space–time residuals norms.
   - Minimizing the residual over a larger time window does not necessarily lead to a more accurate trajectory (as measured in the space–time ℓ^2-norm of the solution). Instead, minimizing the residual over an intermediate-sized time window leads to the smallest trajectory error.

The paper proceeds as follows: Section 2 outlines the mathematical setting for the full-order model, along with Galerkin, LSPG, and space–time LSPG projection. Section 3 outlines the proposed WLS approach. Section 4 outlines numerical techniques for solving WLS ROMs, including both direct and indirect methods. Section 5 provides equivalence conditions and error analysis for WLS ROMs. Section 6 presents numerical experiments. Section 7 provides conclusions and perspectives. We denote vector-valued functions with italicized bold symbols (e.g., $\boldsymbol{x}$), vectors with standard bold symbols (e.g., $\mathbf{x}$), matrices with capital bold symbols (e.g., $\mathbf{X} \equiv \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_r\end{bmatrix}$), and spaces with calligraphic symbols (e.g., $\mathcal{X}$). We additionally denote differentiation of a time-dependent function with respect to time with the $\dot{}$ operator.

2. Mathematical formulation

We begin by providing the formulation for the full-order model, followed by a description of standard model-reduction methods classified according to the type of trial subspace they employ.

2.1. Full-order model

We consider the full-order model to be a dynamical system expressed as a system of ordinary differential equations (ODEs)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), t), \qquad \boldsymbol{x}(0) = \mathbf{x}_0, \qquad t \in [0, T], \tag{2.1}$$

where $\boldsymbol{x} : [0,T] \to \mathbb{R}^N$ with $\boldsymbol{x} : \tau \mapsto \boldsymbol{x}(\tau)$ and $\dot{\boldsymbol{x}} \equiv d\boldsymbol{x}/d\tau$ denotes the state implicitly defined as the solution to initial value problem (2.1), $T \in \mathbb{R}_+$ denotes the final time, $\mathbf{x}_0 \in \mathbb{R}^N$ denotes the initial condition, and $\boldsymbol{f} : \mathbb{R}^N \times [0,T] \to \mathbb{R}^N$ with $(\mathbf{y}, \tau) \mapsto \boldsymbol{f}(\mathbf{y}, \tau)$ denotes the velocity, which is possibly nonlinear in its first argument. For subsequent exposition, we introduce $\mathcal{T}$ to denote the set of (sufficiently smooth) real-valued functions acting on the time domain (i.e., $\mathcal{T} = \{f \,|\, f : [0,T] \to \mathbb{R}\}$); the state can be expressed equivalently as $\boldsymbol{x} \in \mathbb{R}^N \otimes \mathcal{T}$. We refer to the initial value problem defined in Eq. (2.1) as the "full-order model" (FOM) ODE. We note that although the problem of interest described in the introduction corresponds to a parameterized dynamical system, we suppress dependence of the FOM ODE (2.1) on such parameters for notational convenience, as this work focuses on devising a model-reduction approach applicable to a specific parameter instance.

Directly solving the FOM ODE (2.1) is computationally expensive if either the state-space dimension N is large, or if the time-interval length T is large relative to the time step required to numerically integrate Eq. (2.1). For time-critical or many-query applications, it is essential to replace the FOM ODE (2.1) with a strategy that enables an approximate trajectory to be computed at lower computational cost. Projection-based ROMs constitute one such promising approach.

2.2. Reduced-order models

Projection-based ROMs generate approximate solutions to the FOM ODE (2.1) by approximating the state in a low-dimensional trial subspace. Two types of space–time trial subspaces are commonly used for this purpose:[2]

1. *Subspaces that reduce only the spatial dimension of the full-order model (S-reduction).* These trial subspaces are characterized by a spatial projection operator, associate with a basis that represents the spatial dependence of the state, and are employed in classic model reduction approaches, e.g., Galerkin and LSPG.
2. *Subspaces that reduce both the spatial and temporal dimensions of the full-order model (ST-reduction).* These trial subspaces are characterized by a space–time projection operator, associate with a basis that represents the spatial and temporal dependence of the state, and are employed in space–time model reduction approaches (e.g., space–time Galerkin [5], space–time LSPG [25]).

We now describe these two types of space–time trial subspaces and their application to the Galerkin, LSPG, and space–time LSPG approaches.

2.3. S-reduction trial subspaces

At a given time instance $t \in [0, T]$, S-reduction trial subspaces approximate the FOM ODE solution as $\tilde{\boldsymbol{x}}(t) \approx \boldsymbol{x}(t)$, which is enforced to reside in an affine spatial trial subspace of dimension $K \ll N$ such that $\tilde{\boldsymbol{x}}(t) \in \mathbf{x}_{\text{ref}} + \mathcal{V} \subseteq \mathbb{R}^N$, where $\dim(\mathcal{V}) = K$ and $\mathbf{x}_{\text{ref}} \in \mathbb{R}^N$ denotes the reference state, which is often taken to be the initial condition (i.e., $\mathbf{x}_{\text{ref}} = \mathbf{x}_0$). Here, the trial subspace $\mathcal{V}$ is spanned by an orthogonal basis such that $\mathcal{V} = \text{Ran}(\mathbf{V})$ with $\mathbf{V} \equiv \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_K \end{bmatrix} \in \mathbb{V}_K(\mathbb{R}^N)$, where $\mathbb{V}_K(\mathbb{R}^N)$ denotes the compact Stiefel manifold (i.e., $\mathbb{V}_K(\mathbb{R}^N) := \{\mathbf{X} \in \mathbb{R}^{N\times K} \,|\, \mathbf{X}^T\mathbf{X} = \mathbf{I}\}$). The basis vectors $\mathbf{v}_i$, $i = 1, \ldots, K$ are typically constructed using state snapshots, e.g., via proper orthogonal decomposition (POD) [9], the reduced-basis method [52, 64, 63, 49, 56]. Thus, at any time instance $t \in [0, T]$, ROMs that employ the S-reduction trial subspace approximate the FOM ODE solution as

$$\boldsymbol{x}(t) \approx \tilde{\boldsymbol{x}}(t) = \mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, \tag{2.2}$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^K \otimes \mathcal{T}$ with $\hat{\boldsymbol{x}} : \tau \mapsto \hat{\boldsymbol{x}}(\tau)$ denotes the generalized coordinates. From the space–time perspective, this is equivalent to approximating the FOM ODE solution trajectory $\boldsymbol{x} \in \mathbb{R}^N \otimes \mathcal{T}$ with $\tilde{\boldsymbol{x}} \in \mathcal{ST}_{\text{S}}$, where

$$\mathcal{ST}_{\text{S}} := \mathcal{V} \otimes \mathcal{T} + \mathbf{x}_{\text{ref}} \otimes \mathcal{O} \subseteq \mathbb{R}^N \otimes \mathcal{T}, \tag{2.3}$$

with $\mathcal{O} \in \mathcal{T}$ defined as $\mathcal{O} : \tau \mapsto 1$.

Substituting the approximation (2.2) into the FOM ODE (2.1) and performing orthogonal ℓ^2-projection of the initial condition onto the trial subspace yields the overdetermined system of ODEs

$$\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t), \qquad \hat{\boldsymbol{x}}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{2.4}$$

where $\dot{\hat{\boldsymbol{x}}} \equiv d\hat{\boldsymbol{x}}/d\tau$. Because Eq. (2.4) is overdetermined, a solution may not exist. Typically, either *Galerkin* or *least-squares Petrov–Galerkin* projection is employed to reduce the number of equations such that a unique solution exists. We now describe these two methods.

2.3.1. Galerkin projection

The Galerkin approach reduces the number of equations in Eq. (2.4) by enforcing orthogonality of the residual to the spatial trial subspace in the (semi-)inner product induced by the positive (semi-)definite $N \times N$ matrix $\mathbf{A} \equiv [\mathbf{W}]^T\mathbf{W}$ (commonly set to $\mathbf{A} = \mathbf{I}$), i.e.,

$$\dot{\hat{\boldsymbol{x}}}_{\text{G}}(t) = \mathbf{M}^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}, t), \qquad \hat{\boldsymbol{x}}_{\text{G}}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{2.5}$$

where $\mathbf{M} \equiv \mathbf{V}^T\mathbf{A}\mathbf{V}$ denotes the $K \times K$ positive definite mass matrix. As demonstrated in Ref. [18], the Galerkin approach can be viewed alternatively as a residual-minimization method, as the Galerkin ODE (2.5) is equivalent to

$$\dot{\hat{\boldsymbol{x}}}_{\text{G}}(t) = \underset{\hat{\mathbf{y}} \in \mathbb{R}^K}{\arg\min} \|\mathbf{V}\hat{\mathbf{y}} - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}, t)\|_{\mathbf{A}}^2, \qquad \hat{\boldsymbol{x}}(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}), \qquad t \in [0, T], \tag{2.6}$$

where $\|\mathbf{x}\|_{\mathbf{A}} \equiv \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$. Thus, the computed velocity $\dot{\hat{\boldsymbol{x}}}_{\text{G}}(t)$ minimizes the FOM ODE residual evaluated at the state $\mathbf{V}\hat{\boldsymbol{x}}_{\text{G}}(t) + \mathbf{x}_{\text{ref}}$ and time instance t over the spatial trial subspace $\mathcal{V}$.

[2]For both spatial and space–time ROMs of dynamical systems, all trial subspaces are, strictly speaking, space–time subspaces.

2.3.2. Least-squares Petrov–Galerkin projection

Despite its time-instantaneous residual-minimization optimality property (2.6), the Galerkin approach can yield inaccurate solutions, particularly if the velocity is not self-adjoint or is nonlinear. Least-squares Petrov–Galerkin (LSPG) [18, 16, 21, 14, 12] was developed as an alternative method that exhibits several advantages over the Galerkin approach. Rather than minimize the (time-continuous) FOM ODE residual at a time instance (as in Galerkin), LSPG minimizes the (time-discrete) FOM OΔE residual (i.e., the residual arising after applying time discretization to the FOM ODE) over a time step. We now describe the LSPG approach in the case of linear multistep methods; Ref. [18] also presents LSPG for Runge–Kutta schemes.

Without loss of generality, we introduce a uniform time grid characterized by time step Δt and time instances $t^n = n\Delta t$, $n = 0, \ldots, N_t$. Applying a linear multistep method to discretize the FOM ODE (2.1) with this time grid yields the FOM OΔE, which computes the sequence of discrete solutions $\mathbf{x}^n (\approx \boldsymbol{x}(t^n))$, $n = 1, \ldots, N_t$ as the implicit solution to the system of algebraic equations

$$\boldsymbol{r}^n(\mathbf{x}^n; \mathbf{x}^{n-1}, \ldots, \mathbf{x}^{n-k^n}) = \mathbf{0}, \qquad n = 1, \ldots, N_t, \tag{2.7}$$

with the initial condition $\mathbf{x}^0 = \mathbf{x}_0$. In the above, k^n denotes the number of steps employed by the scheme at the nth time instance and $\boldsymbol{r}^n$ denotes the FOM OΔE residual defined as

$$\begin{aligned}\boldsymbol{r}^n : (\mathbf{y}^n; \mathbf{y}^{n-1}, \ldots, \mathbf{y}^{n-k^n}) &\mapsto \frac{1}{\Delta t}\sum_{j=0}^{k^n} \alpha_j^n \mathbf{y}^{n-j} - \sum_{j=0}^{k^n} \beta_j^n \boldsymbol{f}(\mathbf{y}^{n-j}, t^{n-j}),\\ &: \mathbb{R}^N \otimes \mathbb{R}^{k^n+1} \to \mathbb{R}^N.\end{aligned}$$

Here, $\alpha_j^n, \beta_j^n \in \mathbb{R}$, $j = 0, \ldots, k^n$ are coefficients that define the linear multistep method at the nth time instance.

At each time instance on the time grid, LSPG substitutes the S-reduction trial subspace approximation (2.2) into the FOM OΔE (2.7) and minimizes the residual, i.e., LSPG sequentially computes the solutions $\tilde{\mathbf{x}}_{\mathrm{L}}^n \approx \mathbf{x}^n$, $n = 1, \ldots, N_t$ that satisfy

$$\tilde{\mathbf{x}}_{\mathrm{L}}^n = \underset{\mathbf{y}\in\mathcal{V}+\mathbf{x}_{\mathrm{ref}}}{\arg\min} ||\mathbf{W}\boldsymbol{r}^n(\mathbf{y}; \tilde{\mathbf{x}}_{\mathrm{L}}^{n-1}, \ldots, \tilde{\mathbf{x}}_{\mathrm{L}}^{n-k^n})||_2^2, \qquad n = 1, \ldots, N_t,$$

where $\mathbf{W} \in \mathbb{R}^{n_s \times N}$, with $K \le n_s \le N$, is a weighting matrix that can be used, e.g., to enable hyper-reduction by requiring it to have a small number of nonzero columns.

As described in the introduction, although numerical experiments have demonstrated that LSPG often yields more accurate and stable solutions than Galerkin [12, 18, 21, 16, 50], LSPG still suffers from several shortcomings. In particular, LSPG suffers from its complex dependence on the time discretization, exponentially growing error bounds, and lack of optimality for the trajectory defined over the entire time domain.

2.4. ST-reduction trial spaces and space–time ROMs

Space–time projection methods that employ ST-reduction trial spaces [25, 26, 62, 68, 5, 12] aim to overcome the latter two shortcomings of LSPG. Because these methods employ ST-reduction trial spaces, they reduce both the spatial and temporal dimensions of the full-order model; further, they yield error bounds that grow more slowly in time and their trajectories exhibit an optimality property over the entire time domain.

ST-reduction trial subspaces approximate the FOM ODE solution trajectory $\boldsymbol{x} \in \mathbb{R}^N \otimes \mathcal{T}$ with an approximation that resides in an affine space–time trial subspace of dimension $K_{\mathrm{ST}} \ll N$, i.e., $\tilde{\boldsymbol{x}} \in \mathcal{ST}_{\mathrm{ST}}$ with $\dim(\mathcal{ST}_{\mathrm{ST}}) = K_{\mathrm{ST}}$, where

$$\mathcal{ST}_{\mathrm{ST}} := \mathrm{Ran}(\boldsymbol{\Pi}) + \mathbf{x}_0 \otimes \mathcal{O} \subseteq \mathbb{R}^N \otimes \mathcal{T}. \tag{2.8}$$

Here $\boldsymbol{\Pi} \in \mathbb{R}^{N \times K_{\mathrm{ST}}} \otimes \mathcal{T}$, with $\boldsymbol{\Pi} : \tau \mapsto \boldsymbol{\Pi}(\tau)$ and $\boldsymbol{\Pi}(0) = \mathbf{0}$ denotes the space–time trial basis. Thus, at any time instance $t \in [0, T]$, ROMs that employ the ST-reduction trial subspace approximate the FOM ODE solution as

$$\boldsymbol{x}(t) \approx \tilde{\boldsymbol{x}}(t) = \boldsymbol{\Pi}(t)\hat{\hat{\mathbf{x}}} + \mathbf{x}_0, \tag{2.9}$$

where $\hat{\hat{\mathbf{x}}} \in \mathbb{R}^{K_{\mathrm{ST}}}$ denotes the space–time generalized coordinates. Critically, comparing the approximations arising from S-reduction and ST-reduction trial subspaces in Eqs. (2.2) and (2.9), respectively, highlights that the former approximation associates with time-dependent generalized coordinates, while the latter approximation associates with a time-dependent basis matrix.

Substituting the approximation (2.9) into the FOM ODE (2.1) yields

$$\dot{\mathbf{\Pi}}(t)\hat{\hat{\mathbf{x}}} = \boldsymbol{f}(\mathbf{\Pi}(t)\hat{\hat{\mathbf{x}}} + \mathbf{x}_0, t), \qquad t \in [0, T], \tag{2.10}$$

where $\dot{\mathbf{\Pi}} \equiv d\mathbf{\Pi}/d\tau$. We note that the initial conditions are automatically satisfied from the definition of the ST-reduction trial subspace.

Space–time methods reduce the number of equations in (2.10) to ensure a unique solution exists. We now outline one such method: space–time least-squares Petrov–Galerkin (ST-LSPG) [25]. While the space–time Galerkin method [5] is another alternative, it does not associate with any residual-minimization principle, and thus we do not discuss it further.

2.4.1. *Space–time LSPG projection*

Analogously to LSPG, space–time LSPG [25] minimizes the (time-discrete) FOM OΔE residual, but does so using the ST-reduction subspace and simultaneously minimizes this residual over all N_t time instances. We first introduce the full space–time FOM OΔE residual for linear multistep methods as

$$\begin{aligned}\overline{\boldsymbol{r}} : (\mathbf{y}^1, \dots, \mathbf{y}^{N_t}; \mathbf{x}_0) &\mapsto \begin{bmatrix} \boldsymbol{r}^1(\mathbf{y}^1; \mathbf{x}_0) \\ \vdots \\ \boldsymbol{r}^{N_t}(\mathbf{y}^{N_t}; \mathbf{y}^{N_t-1}, \dots, \mathbf{y}^{N_t - k^{N_t}}) \end{bmatrix}, \\ &: \mathbb{R}^N \otimes \mathbb{R}^{N_t+1} \rightarrow \mathbb{R}^{NN_t},\end{aligned}$$

and define the counterpart function acting on space–time generalized coordinates:

$$\begin{aligned}\hat{\hat{\boldsymbol{r}}} : (\hat{\hat{\mathbf{y}}}; \mathbf{x}_0) &\mapsto \begin{bmatrix} \boldsymbol{r}^1\left(\mathbf{\Pi}(t^1)\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\text{ref}}; \mathbf{x}_0\right) \\ \vdots \\ \boldsymbol{r}^{N_t}\left(\mathbf{\Pi}(t^{N_t})\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\text{ref}}; \mathbf{\Pi}(t^{N_t-1})\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\text{ref}}, \dots, \mathbf{\Pi}(t^{N_t-k^{N_t}})\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\text{ref}}\right) \end{bmatrix}, \\ &: \mathbb{R}^{K_{\text{ST}}} \times \mathbb{R}^N \rightarrow \mathbb{R}^{NN_t}.\end{aligned}$$

ST-LSPG computes the space–time generalized coordinates that minimize the space–time FOM OΔE residual:

$$\hat{\hat{\mathbf{x}}}_{\text{ST-LSPG}} = \underset{\hat{\hat{\mathbf{y}}} \in \mathbb{R}^{K_{\text{ST}}}}{\arg\min} ||\mathbf{W}_{\text{ST}} \hat{\hat{\boldsymbol{r}}}(\hat{\hat{\mathbf{y}}}; \mathbf{x}_0)||_2^2, \tag{2.11}$$

where $\mathbf{W}_{\text{ST}} \in \mathbb{R}^{n_{st} \times NN_t}$, with $K_{\text{ST}} \leq n_{st} \leq NN_t$, is a space–time weighting matrix that can be chosen, e.g., to enable hyper-reduction.

ST-LSPG overcomes two of the primary shortcomings of LSPG. In particular, it leads to error bounds that grow sub-quadratically in time rather than exponentially in time, and it generates entire trajectories that associate with an optimality property over the entire time domain [25]. However, it is subject to several challenges. First, the computational cost of solving Eq. (2.11) scales cubically with the number of space–time degrees of freedom K_{ST}. This cost is due to the fact that ST-LSPG yields dense systems that do not expose any natural mechanism for exploiting the sequential nature of time evolution. Second, it is unclear how these methods can be employed for future state prediction, as the space–time trial basis $\mathbf{\Pi}$ must be defined over the entire time interval of interest $[0, T]$. Third, ST-LSPG is still strongly tied to the time discretization employed for the full-order model, as it minimizes the (time-discrete) FOM OΔE residual over all time instances.

2.5. *Outstanding challenges*

This work seeks to overcome the limitations of existing residual-minimizing model-reduction methods, and to provide a unifying framework from which existing methods can be assessed. Specifically, we look to overcome the complex dependence of LSPG on the time discretization, exponential time growth of the error bounds for Galerkin and LSPG, the cubic dependence of the computational cost of ST-LSPG on the number of degrees of freedom, and the lack of ability for ST-LSPG to perform prediction in time. We now describe the proposed windowed least-squares approach for this purpose.

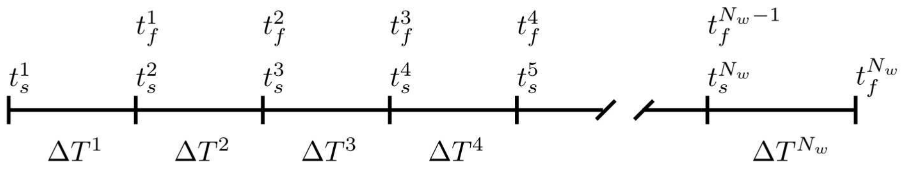


Figure 2: Partitioning of the time domain into time windows.

3. Windowed least-squares approach

This section outlines the proposed windowed least-squares (WLS) approach. In contrast to (1) Galerkin projection, which minimizes the (time-continuous) FOM ODE residual at a time instance, (2) LSPG projection, which minimizes the (time-discrete) FOM OΔE residual over a time step, and (3) ST-LSPG projection, which minimizes the FOM OΔE residual over the entire time interval, the proposed WLS approach sequentially minimizes the FOM ODE residual over arbitrarily defined *time windows*. The formulation is compatible with both S-reduction and ST-reduction trial subspaces. In this section, we start by outlining the WLS formulation for a general space–time trial subspace. We then examine the S-reduction trial subspace in this context, followed by the ST-reduction trial subspace. In each case, we derive the stationary conditions associated with the residual-minimization problems.

3.1. Windowed least-squares for general space–time trial subspaces

We begin by introducing a (potentially nonuniform) partition of the time domain $[0, T]$ into N_w non-overlapping windows $[t_s^n, t_f^n] \subseteq [0, T]$ of length $\Delta T^n := t_f^n - t_s^n$, $n = 1, \ldots, N_w$ such that $t_s^1 = 0$, $t_f^{N_w} = T$, and $t_s^{n+1} = t_f^n$, $n = 1, \ldots, N_w - 1$; Fig. 2 depicts such a partitioning. Over the nth time window, we approximate the FOM ODE solution as $\tilde{\boldsymbol{x}}^n(t) \approx \boldsymbol{x}(t)$, $t \in [t_s^n, t_f^n]$, which is enforced to reside in the nth space–time trial subspace $\mathcal{ST}^n$ such that

$$\tilde{\boldsymbol{x}}^n \in \mathcal{ST}^n \subseteq \mathbb{R}^N \otimes \mathcal{T}^n, \qquad n = 1, \ldots, N_w, \tag{3.1}$$

where $\mathcal{T}^n$ denotes the set of (sufficiently smooth) real-valued functions acting on $[t_s^n, t_f^n]$ (i.e., $\mathcal{T}^n = \{f \mid f : [t_s^n, t_f^n] \to \mathbb{R}\}$). For notational purposes, we additionally define the spatial trial subspaces at the start of each window as

$$\mathcal{B}^n := \operatorname{span}(\{\boldsymbol{y}(t_s^n) \mid \boldsymbol{y} \in \mathcal{ST}^n\}) \subseteq \mathbb{R}^N, \qquad n = 1, \ldots, N_w, \tag{3.2}$$

such that $\tilde{\boldsymbol{x}}^n(t_s^n) \in \mathcal{B}^n$. To outline WLS, we now define the objective functional over the nth window as

$$\begin{aligned} \mathcal{J}^n : \boldsymbol{y} \mapsto \frac{1}{2} \int_{t_s^n}^{t_f^n} \left[\dot{\boldsymbol{y}}(t) - \boldsymbol{f}(\boldsymbol{y}(t), t)\right]^T \mathbf{A}^n \left[\dot{\boldsymbol{y}}(t) - \boldsymbol{f}(\boldsymbol{y}(t), t)\right] dt, \\ : \mathbb{R}^N \otimes \mathcal{T}^n \to \mathbb{R}_+, \end{aligned} \tag{3.3}$$

where $\mathbf{A}^n \equiv [\mathbf{W}^n]^T \mathbf{W}^n \in \mathbb{R}^{N \times N}$ denotes a symmetric positive semi-definite matrix that can enable hyper-reduction, for example.

The WLS approach sequentially computes approximate solutions $\tilde{\boldsymbol{x}}^n \in \mathcal{ST}^n$, $n = 1, \ldots, N_w$, where $\tilde{\boldsymbol{x}}^n$ is the solution to the minimization problem

$$\begin{aligned} &\underset{\boldsymbol{y} \in \mathcal{ST}^n}{\text{minimize}}\ \mathcal{J}^n(\boldsymbol{y}), \\ &\text{subject to}\ \ \boldsymbol{y}(t_s^n) = \begin{cases} \mathbb{P}^n(\tilde{\boldsymbol{x}}^{n-1}(t_f^{n-1})) & n = 2, \ldots, N_w, \\ \mathbb{P}^n(\mathbf{x}_0) & n = 1, \end{cases} \end{aligned} \tag{3.4}$$

where $\mathbb{P}^n : \mathbb{R}^N \to \mathcal{B}^n$ is a (spatial) projection operator onto the start of the nth trial space (e.g., ℓ^2-orthogonal projection operator).

We now define the trial subspaces $\mathcal{ST}^n$ considered in this work. In particular, we introduce S-reduction trial subspaces and ST-reduction trial subspaces tailored for this context. We leave further investigation into other approaches, such as nonlinear trial manifolds [40], as a subject for future work.

3.2. S-reduction trial subspaces

In this context, the S-reduction trial subspace over the nth time window approximates the FOM ODE solution trajectory $\boldsymbol{x}(t)$, $t \in [t_s^n, t_f^n]$ with $\tilde{\boldsymbol{x}}^n \in \mathcal{ST}_{\mathrm{S}}^n$, where

$$\mathcal{ST}_{\mathrm{S}}^n := \mathcal{V}^n \otimes \mathcal{T}^n + \mathbf{x}_{\mathrm{ref}}^n \otimes \mathcal{O}^n \subseteq \mathbb{R}^N \otimes \mathcal{T}^n. \tag{3.5}$$

Here, the spatial trial subspaces $\mathcal{V}^n \subseteq \mathbb{R}^N$, $n = 1, \ldots, N_w$ satisfy $\mathcal{V}^n := \mathrm{Ran}(\mathbf{V}^n)$ with $\mathbf{V}^n \equiv [\mathbf{v}_1^n \ \cdots \ \mathbf{v}_{K^n}^n] \in \mathbb{V}_{K^n}(\mathbb{R}^N)$, the reference states $\mathbf{x}_{\mathrm{ref}}^n \in \mathbb{R}^N$, $n = 1, \ldots, N_w$ provide the affine transformation, and $\mathcal{O}^n \in \mathcal{T}^n$ is defined as $\mathcal{O}^n : \tau \mapsto 1$. Thus, at any time instance $t \in [t_s^n, t_f^n]$, the S-reduction trial subspace approximates the FOM ODE solution as

$$\boldsymbol{x}(t) \approx \tilde{\boldsymbol{x}}^n(t) = \mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\mathrm{ref}}^n, \tag{3.6}$$

where $\hat{\boldsymbol{x}}^n \in \mathbb{R}^{K^n} \otimes \mathcal{T}^n$ with $\hat{\boldsymbol{x}}^n : \tau \mapsto \hat{\boldsymbol{x}}^n(\tau)$ denotes the generalized coordinates over the nth time window.

Setting $\mathcal{ST}^n \leftarrow \mathcal{ST}_{\mathrm{S}}^n$ in the WLS minimization problem (3.4) and setting $\mathbb{P}^n$ to the ℓ^2-orthogonal projection operator implies that WLS with S-reduction space–time trial subspaces sequentially computes solutions $\hat{\boldsymbol{x}}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\begin{aligned}
&\underset{\hat{\boldsymbol{y}} \in \mathbb{R}^{K^n} \otimes \mathcal{T}^n}{\text{minimize}} \ \mathcal{J}^n(\mathbf{V}^n \hat{\boldsymbol{y}} + \mathbf{x}_{\mathrm{ref}}^n \otimes \mathcal{O}^n),\\
&\text{subject to } \ \hat{\boldsymbol{y}}(t_s^n) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{\boldsymbol{x}}^{n-1}(t_f^{n-1}) + \mathbf{x}_{\mathrm{ref}}^{n-1} - \mathbf{x}_{\mathrm{ref}}^n) & n = 2, \ldots, N_w,\\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}_{\mathrm{ref}}^1) & n = 1. \end{cases}
\end{aligned} \tag{3.7}$$

3.2.1. Stationary conditions and the Euler–Lagrange equations

We derive stationary conditions for optimization problem (3.7) via the Euler–Lagrange equations from the calculus of variations. To begin, we define the integrand appearing in the objective function $\mathcal{J}^n$ defined in Eq. (3.3) in terms of the generalized coordinates induced by the S-reduction subspace as

$$\begin{aligned}
\mathcal{I}^n : (\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &\mapsto \frac{1}{2}\big[\mathbf{V}^n\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}^n\hat{\mathbf{y}} + \mathbf{x}_{\mathrm{ref}}^n, \tau)\big]^T \mathbf{A}^n \big[\mathbf{V}^n\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}^n\hat{\mathbf{y}} + \mathbf{x}_{\mathrm{ref}}^n, \tau)\big],\\
&: \mathbb{R}^{K^n} \times \mathbb{R}^{K^n} \times [t_s^n, t_f^n] \rightarrow \mathbb{R}_+.
\end{aligned} \tag{3.8}$$

We also define the quantities[3]

$$\begin{aligned}
\mathcal{I}_{\hat{\mathbf{v}}}^n : (\hat{\boldsymbol{y}}, \hat{\boldsymbol{v}}, \tau) &\mapsto \left[\frac{\partial \mathcal{I}^n}{\partial \hat{\mathbf{v}}}(\hat{\boldsymbol{y}}(\tau), \hat{\boldsymbol{v}}(\tau), \tau)\right]^T,\\
&: \mathbb{R}^{K^n} \otimes \mathcal{T}^n \times \mathbb{R}^{K^n} \otimes \mathcal{T}^n \times [t_s^n, t_f^n] \rightarrow \mathbb{R}^{K^n},
\end{aligned} \tag{3.9}$$

$$\begin{aligned}
\mathcal{I}_{\hat{\mathbf{y}}}^n : (\hat{\boldsymbol{y}}, \hat{\boldsymbol{v}}, \tau) &\mapsto \left[\frac{\partial \mathcal{I}^n}{\partial \hat{\mathbf{y}}}(\hat{\boldsymbol{y}}(\tau), \hat{\boldsymbol{v}}(\tau), \tau)\right]^T,\\
&: \mathbb{R}^{K^n} \otimes \mathcal{T}^n \times \mathbb{R}^{K^n} \otimes \mathcal{T}^n \times [t_s^n, t_f^n] \rightarrow \mathbb{R}^{K^n}.
\end{aligned} \tag{3.10}$$

Using this notation, the Euler–Lagrange equations (see Appendix C for the derivation) over the nth time window for $t \in [t_s^n, t_f^n]$ are given by

$$\begin{aligned}
&\mathcal{I}_{\hat{\mathbf{y}}}^n(\hat{\boldsymbol{x}}^n, \dot{\hat{\boldsymbol{x}}}^n, t) - \dot{\mathcal{I}}_{\hat{\mathbf{v}}}^n(\hat{\boldsymbol{x}}^n, \dot{\hat{\boldsymbol{x}}}^n, t) = \mathbf{0},\\
&\hat{\boldsymbol{x}}^n(t_s^n) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{\boldsymbol{x}}^{n-1}(t_f^{n-1}) + \mathbf{x}_{\mathrm{ref}}^{n-1} - \mathbf{x}_{\mathrm{ref}}^n) & n = 2, \ldots, N_w,\\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}_{\mathrm{ref}}^1) & n = 1, \end{cases}\\
&\mathcal{I}_{\hat{\mathbf{v}}}^n(\hat{\boldsymbol{x}}^n, \dot{\hat{\boldsymbol{x}}}^n, t_f^n) = \mathbf{0}.
\end{aligned} \tag{3.11}$$

[3]We use numerator layout for the scalar-by-vector gradients.

Appendix D provides the steps required to evaluate the terms in system (3.11); the resulting system can be written as the following coupled forward–backward system for $t \in [t_s^n, t_f^n]$:

$$\mathbf{M}^n \dot{\hat{\boldsymbol{x}}}^n(t) - [\mathbf{V}^n]^T \mathbf{A}^n \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t) = \mathbf{M}^n \hat{\boldsymbol{\lambda}}^n(t), \tag{3.12}$$

$$\begin{aligned}\mathbf{M}^n \dot{\hat{\boldsymbol{\lambda}}}^n(t) + [\mathbf{V}^n]^T \left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\right]^T \mathbf{A}^n \mathbf{V}^n \hat{\boldsymbol{\lambda}}^n(t) = -[\mathbf{V}^n]^T \big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\big]^T \\ \mathbf{A}^n \big(\mathbf{I} - \mathbf{V}^n [\mathbf{M}^n]^{-1} [\mathbf{V}^n]^T \mathbf{A}^n\big)\big(\mathbf{V}^n \dot{\hat{\boldsymbol{x}}}^n(t) - \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\big),\end{aligned} \tag{3.13}$$

$$\hat{\boldsymbol{x}}^n(t_s^n) = \begin{cases} [\mathbf{V}^n]^T (\mathbf{V}^{n-1} \hat{\boldsymbol{x}}^{n-1}(t_f^{n-1}) + \mathbf{x}_{\text{ref}}^{n-1} - \mathbf{x}_{\text{ref}}^n) & n = 2, \dots, N_w, \\ [\mathbf{V}^1]^T (\mathbf{x}_0 - \mathbf{x}_{\text{ref}}^1) & n = 1, \end{cases} \tag{3.14}$$

$$\hat{\boldsymbol{\lambda}}^n(t_f^n) = \mathbf{0}, \tag{3.15}$$

where

$$\begin{aligned}\hat{\boldsymbol{\lambda}}^n &: \tau \mapsto \dot{\hat{\boldsymbol{x}}}^n(\tau) - [\mathbf{M}^n]^{-1} [\mathbf{V}^n]^T \mathbf{A}^n \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(\tau) + \mathbf{x}_{\text{ref}}^n, \tau), \\ &: [t_s^n, t_f^n] \to \mathbb{R}^{K^n},\end{aligned} \tag{3.16}$$

is the adjoint or "costate" variable with $\dot{\hat{\boldsymbol{\lambda}}}^n \equiv d\hat{\boldsymbol{\lambda}}^n / d\tau$, and $\mathbf{M}^n \equiv [\mathbf{V}^n]^T \mathbf{A}^n \mathbf{V}^n$ is a mass matrix. Eq. (3.12) is equivalent to a Galerkin reduced-order model forced by the costate variable $\hat{\boldsymbol{\lambda}}^n$. Eq. (3.13) is typically referred to as the adjoint equation, which is linear in the costate and is forced by the residual. We note that both ODEs (3.12) and (3.13) can be equipped with hyper-reduction, e.g., via collocation, (discrete) empirical interpolation, Gappy POD [30, 3, 28]. The state–costate coupled system (3.12)–(3.15) can be interpreted as an "optimally controlled" ROM, wherein the adjoint equation controls the forward model by enforcing the minimum residual condition over the time window.

We note that in the case $\mathbf{A}^n = \mathbf{I}$, the system simplifies to

$$\dot{\hat{\boldsymbol{x}}}^n(t) - [\mathbf{V}^n]^T \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t) = \hat{\boldsymbol{\lambda}}^n(t),$$

$$\begin{aligned}\dot{\hat{\boldsymbol{\lambda}}}^n(t) + [\mathbf{V}^n]^T \left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\right]^T \mathbf{V}^n \hat{\boldsymbol{\lambda}}^n(t) = \\ [\mathbf{V}^n]^T \left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\right]^T \big(\mathbf{I} - \mathbf{V}^n [\mathbf{V}^n]^T\big) \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t),\end{aligned}$$

$$\hat{\boldsymbol{x}}^n(t_s^n) = \begin{cases} [\mathbf{V}^n]^T (\mathbf{V}^{n-1} \hat{\boldsymbol{x}}^{n-1}(t_f^{n-1}) - \mathbf{x}_{\text{ref}}^n) & n = 2, \dots, N_w, \\ [\mathbf{V}^1]^T (\mathbf{x}_0 - \mathbf{x}_{\text{ref}}^1) & n = 1, \end{cases}$$

$$\hat{\boldsymbol{\lambda}}^n(t_f^n) = \mathbf{0}.$$

3.2.2. Formulation as an optimal-control problem of Lagrange type

The stationary conditions for WLS with S-reduction trial subspaces (3.12)–(3.15) can be alternatively formulated as a Lagrange problem from optimal control. To this end, recall the dynamics of the Galerkin ROM over $t \in [t_s^n, t_f^n]$,

$$[\mathbf{V}]^T \mathbf{A} \mathbf{V} \dot{\hat{\boldsymbol{x}}}_{\text{G}}(t) - [\mathbf{V}]^T \mathbf{A} \boldsymbol{f}(\mathbf{V} \hat{\boldsymbol{x}}_{\text{G}} + \mathbf{x}_{\text{ref}}, t) = \mathbf{0}.$$

We introduce now a controller $\hat{\boldsymbol{u}}^n \in \mathbb{R}^{K^n} \otimes \mathcal{T}^n$ and pose the problem of finding a controller that minimizes the residual over the time window and forces the dynamics as

$$[\mathbf{V}^n]^T \mathbf{A}^n \mathbf{V}^n \dot{\hat{\boldsymbol{x}}}^n(t) - [\mathbf{V}^n]^T \mathbf{A}^n \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t) = \hat{\boldsymbol{u}}^n(t). \tag{3.17}$$

We now demonstrate how to compute this controller. Before doing so, we note that (3.17) displays commonalities with *subgrid-scale* methods [58, 35, 15, 50, 67, 65, 60], which add an additional term to the reduced-order model in order to account for truncated states.

We begin by defining a Lagrangian

$$\begin{aligned}\mathcal{L}^n : (\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) \mapsto \frac{1}{2}\Big[\mathbf{V}^n\Big([\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T \mathbf{A}^n \boldsymbol{f}(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}^n, \tau) + [\mathbf{M}^n]^{-1} \hat{\mathbf{v}}\Big) - \boldsymbol{f}(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}^n, \tau)\Big]^T \mathbf{A}^n \\ \Big[\mathbf{V}^n\Big([\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T \mathbf{A}^n \boldsymbol{f}(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}^n, \tau) + [\mathbf{M}^n]^{-1} \hat{\mathbf{v}}\Big) - \boldsymbol{f}(\mathbf{V}^n \hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}^n, \tau)\Big], \\ : \mathbb{R}^{K^n} \times \mathbb{R}^{K^n} \times [t_s^n, t_f^n] \to \mathbb{R}_+,\end{aligned}$$

where we have used $\dot{\hat{\boldsymbol{x}}}^n(t) = [\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t) + [\mathbf{M}^n]^{-1}\hat{\boldsymbol{u}}^n(t)$ from Eq. (3.17). Note that this Lagrangian measures the same residual as Eq. (3.8). The WLS approach with S-reduction trial subspaces can be formulated as an optimal-control method that sequentially computes the controllers $\hat{\boldsymbol{u}}^n \in \mathbb{R}^{K^n} \otimes \mathcal{T}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\begin{aligned}
&\underset{\hat{\boldsymbol{v}} \in \mathbb{R}^{K^n}\otimes\mathcal{T}^n}{\text{minimize}} \int_{t^n_s}^{t^n_f} \mathcal{L}^n(\hat{\boldsymbol{x}}^n(t), \hat{\boldsymbol{v}}(t), t)dt, \\
&\text{subject to} \begin{cases} [\mathbf{V}^n]^T\mathbf{A}^n\mathbf{V}^n\dot{\hat{\boldsymbol{x}}}^n(t) - [\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t) = \hat{\boldsymbol{v}}(t), \quad t \in [t^n_s, t^n_f] \\ \hat{\boldsymbol{x}}^n(t^n_s) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{\boldsymbol{x}}^{n-1}(t^{n-1}_f) + \mathbf{x}^{n-1}_{\text{ref}} - \mathbf{x}^n_{\text{ref}}) & n = 2, \ldots, N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}^1_{\text{ref}}) & n = 1. \end{cases} \end{cases}
\end{aligned} \tag{3.18}$$

The solution to the system (3.18) is equivalent of that defined by (3.7). This can be demonstrated via the *Pontryagin Maximum Principle* (PMP) [37]. To this end, we introduce the Lagrange multiplier (costate) $\hat{\boldsymbol{\nu}}^n \in \mathbb{R}^{K^n} \otimes \mathcal{T}^n$ and define the Hamiltonian

$$\begin{aligned}
\mathcal{H}^n \ &: \ (\hat{\mathbf{y}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{v}}, \tau) \mapsto \hat{\boldsymbol{\mu}}^T\Big[[\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}^n\hat{\mathbf{y}} + \mathbf{x}^n_{\text{ref}}, \tau) + [\mathbf{M}^n]^{-1}\hat{\mathbf{v}}\Big] + \mathcal{L}^n(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau), \\
&: \ \mathbb{R}^{K^n} \times \mathbb{R}^{K^n} \times \mathbb{R}^{K^n} \times [t^n_s, t^n_f] \to \mathbb{R}.
\end{aligned} \tag{3.19}$$

The Pontryagin Maximum Principle states that solutions of the optimization problem (3.18) must satisfy the following conditions over the nth window,

$$\begin{aligned}
&\dot{\hat{\boldsymbol{x}}}^n(t) = \frac{\partial \mathcal{H}^n}{\partial \hat{\boldsymbol{\mu}}}(\hat{\boldsymbol{x}}^n(t), \hat{\boldsymbol{\nu}}^n(t), \hat{\boldsymbol{u}}^n(t), t), \\
&\dot{\hat{\boldsymbol{\nu}}}^n(t) = -\frac{\partial \mathcal{H}^n}{\partial \hat{\mathbf{y}}}(\hat{\boldsymbol{x}}^n(t), \hat{\boldsymbol{\nu}}^n(t), \hat{\boldsymbol{u}}^n(t), t), \\
&\frac{\partial \mathcal{H}^n}{\partial \hat{\mathbf{v}}}(\hat{\boldsymbol{x}}^n(t), \hat{\boldsymbol{\nu}}^n(t), \hat{\boldsymbol{u}}^n(t), t) = \mathbf{0}, \\
&\hat{\boldsymbol{x}}^n(t^n_s) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{\boldsymbol{x}}^{n-1}(t^{n-1}_f) + \mathbf{x}^{n-1}_{\text{ref}} - \mathbf{x}^n_{\text{ref}}) & n = 2, \ldots, N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}^1_{\text{ref}}) & n = 1, \end{cases} \\
&\hat{\boldsymbol{\nu}}^n(t^n_f) = \mathbf{0}.
\end{aligned}$$

Evaluation of the required gradients (Appendix E) yields the system to be solved over the nth window for $t \in [t^n_s, t^n_f]$,

$$\begin{aligned}
&\mathbf{M}^n\dot{\hat{\boldsymbol{x}}}^n(t) - [\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t) = \hat{\boldsymbol{u}}^n(t), \\
&\dot{\hat{\boldsymbol{\nu}}}^n(t) + [\mathbf{V}^n]^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t)\right]^T\mathbf{A}^n\mathbf{V}^n[\mathbf{M}^n]^{-1}\hat{\boldsymbol{\nu}}^n(t) = [\mathbf{V}^n]^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t)\big]^T \\
&\qquad\qquad \mathbf{A}^n\big(\mathbf{I} - \mathbf{V}^n[\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T\mathbf{A}^n\big)\big(\mathbf{V}^n\dot{\hat{\boldsymbol{x}}}^n(t) - \boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t)\big), \\
&\hat{\boldsymbol{u}}^n(t) = -\hat{\boldsymbol{\nu}}^n(t), \\
&\hat{\boldsymbol{x}}^n(t^n_s) = \begin{cases} [\mathbf{V}^n]^T(\mathbf{V}^{n-1}\hat{\boldsymbol{x}}^{n-1}(t^{n-1}_f) + \mathbf{x}^{n-1}_{\text{ref}} - \mathbf{x}^n_{\text{ref}}) & n = 2, \ldots, N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}^1_{\text{ref}}) & n = 1, \end{cases} \\
&\hat{\boldsymbol{\nu}}^n(t^n_f) = \mathbf{0}.
\end{aligned} \tag{3.20}$$

Setting $\hat{\boldsymbol{u}}^n = \mathbf{M}^n\hat{\boldsymbol{\lambda}}^n$ and $\hat{\boldsymbol{\nu}}^n = -\mathbf{M}^n\hat{\boldsymbol{\lambda}}^n$ results in equivalence between the system (3.20) and the system (3.12)–(3.15). Thus, WLS with S-reduction trial subspaces can be formulated as an optimal control problem: WLS computes a controller that modifies the Galerkin ROM to minimize the residual over the time window. The WLS method can additionally be interpreted as a subgrid-scale modeling technique that constructs a residual-minimizing subgrid-scale model.

Remark 3.1. *The Euler–Lagrange equations comprise a Hamiltonian system. This imbues WLS with S-reduction trial subspaces with certain properties; e.g., for autonomous systems the Hamiltonian* (3.19) *is conserved.*

3.3. ST-reduction trial spaces

The ST-reduction trial subspace over the nth time window approximates the FOM ODE solution trajectory $\boldsymbol{x} \in \mathbb{R}^N \otimes \mathcal{T}$ with $\tilde{\boldsymbol{x}}^n \in \mathcal{ST}^n_{\mathrm{ST}}$, where

$$\mathcal{ST}^n_{\mathrm{ST}} := \mathrm{Ran}(\boldsymbol{\Pi}^n) + \mathbf{x}^n_{\mathrm{st}} \otimes \mathcal{O}^n \subseteq \mathbb{R}^N \otimes \mathcal{T}^n. \tag{3.21}$$

Here $\boldsymbol{\Pi}^n \in \mathbb{R}^{N \times K^n_{\mathrm{ST}}} \otimes \mathcal{T}^n$, $n = 1, \ldots, N_w$, with $\boldsymbol{\Pi}^n : \tau \mapsto \boldsymbol{\Pi}^n(\tau)$ and $\boldsymbol{\Pi}^n(t^n_s) = \mathbf{0}$ is the space–time trial basis matrix function and $\mathbf{x}^n_{\mathrm{st}} \in \mathbb{R}^N$, $n = 1, \ldots, N_w$ provides the affine transformation. To enforce the initial condition and ensure solution continuity across time windows, we set $\mathbf{x}^1_{\mathrm{st}} = \mathbf{x}_0$ and $\mathbf{x}^n_{\mathrm{st}} = \tilde{\boldsymbol{x}}^{n-1}(t^{n-1}_f)$ for $n = 2, \ldots, N_w$. At any time instance $t \in [t^n_s, t^n_f]$, the ST-reduction trial subspace approximates the FOM ODE solution as

$$\boldsymbol{x}^n(t) \approx \tilde{\boldsymbol{x}}^n(t) = \boldsymbol{\Pi}^n(t)\hat{\hat{\mathbf{x}}}^n + \mathbf{x}^n_{\mathrm{st}}, \tag{3.22}$$

where $\hat{\hat{\mathbf{x}}}^n \in \mathbb{R}^{K^n_{\mathrm{ST}}}$ are the space–time generalized coordinates over the nth window. Setting $\mathcal{ST}^n \leftarrow \mathcal{ST}^n_{\mathrm{ST}}$ in the WLS minimization problem (3.4) implies that WLS with ST-reduction trial subspaces sequentially computes solutions $\hat{\hat{\mathbf{x}}}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\underset{\hat{\hat{\mathbf{y}}} \in \mathbb{R}^{K^n_{\mathrm{ST}}}}{\text{minimize}}\ \mathcal{J}^n(\boldsymbol{\Pi}^n\hat{\hat{\mathbf{y}}} + \mathbf{x}^n_{\mathrm{st}} \otimes \mathcal{O}^n). \tag{3.23}$$

3.3.1. Stationary conditions

The key difference between ST-reduction and S-reduction trial subspaces is as follows: generalized coordinates for ST-reduction trial subspaces comprise a vector in $\mathbb{R}^{K^n_{\mathrm{ST}}}$, while generalized coordinates for the S-reduction trial subspaces comprise a time-dependent vector in $\mathbb{R}^{K^n} \otimes \mathcal{T}^n$. Thus, in the ST-reduction case, the optimization problem is no longer minimizing a functional with respect to a (time-dependent) function, but is minimizing a function with respect to a vector. As such, the first-order optimality conditions can be derived using standard calculus. Differentiating the objective function with respect to the generalized coordinates and setting the result equal to zero yields

$$\int_{t^n_s}^{t^n_f} \left[\dot{\boldsymbol{\Pi}}^n(t)^T - \boldsymbol{\Pi}^n(t)^T \left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\dot{\boldsymbol{\Pi}}^n(t)\hat{\hat{\mathbf{x}}}^n + \mathbf{x}^n_{\mathrm{st}}, t) \right]^T \right] \mathbf{A}^n \Big(\dot{\boldsymbol{\Pi}}^n(t)\hat{\hat{\mathbf{x}}}^n - \boldsymbol{f}(\boldsymbol{\Pi}^n(t)\hat{\hat{\mathbf{x}}}^n + \mathbf{x}^n_{\mathrm{st}}, t) \Big) dt = \mathbf{0}. \tag{3.24}$$

Thus, for ST-reduction trial subspaces, the stationary conditions comprise a system of algebraic equations, as opposed to a system of differential equations as with S-reduction trial subspaces.

3.4. WLS summary

This section outlined the WLS approach for model reduction. In summary, WLS sequentially minimizes the time continuous FOM ODE residual within the range of a space–time trial subspace over time windows; i.e., WLS sequentially computes approximate solutions $\tilde{\boldsymbol{x}}^n \in \mathcal{ST}^n$, $n = 1, \ldots, N_w$ that comprise solutions to the optimization problem (3.4). For S-reduction trial subspaces, the stationary conditions for the residual minimization problem can be derived via the Euler–Lagrange equations and yield the system (3.12)–(3.15) over the nth window for $t \in [t^n_s, t^n_f]$. WLS with S-reduction trial subspaces can be alternatively interpreted as a Lagrange problem from optimal control: the Galerkin method is forced by a controller that enforces the residual minimization property. Section 3.3 additionally considered ST-reduction trial subspaces. For ST-reduction trial subspaces, in where the generalized coordinates are no longer functions, the stationary conditions correspond to the system of algebraic equations (3.24).

4. Numerical solution techniques

We now consider numerical-solution techniques for the WLS approach. We focus primarily on direct (i.e., discretize-then-optimize) and indirect (i.e., optimize-then-discretize) methods for S-reduction trial subspaces, and then briefly outline direct and indirect methods for ST-reduction trial subspaces.

4.1. S-reduction: direct and indirect methods

WLS with S-reduction trial subspaces can be viewed as an optimal-control problem. Numerical solution techniques for optimal-control problems can be classified as either *direct* or *indirect* methods [27]. Rather than working with the first-order optimality conditions (3.12)–(3.15), direct methods "directly" solve the optimization problem (3.4) by first numerically discretizing the objective functional and "transcribing" the infinite-dimensional problem into a finite-dimensional problem. Direct approaches thus "discretize then optimize". In contrast, indirect methods compute solutions to the Euler–Lagrange equations (3.12)–(3.15) (which comprise the first-order optimality conditions). Thus, indirect methods "optimize then discretize," and solve the optimization problem "indirectly." A variety of both discretization and solution techniques are possible for both direct and indirect methods. Collocation methods, finite-element methods, spectral methods, and shooting methods are all examples of possible solution techniques.

In the present context, we investigate both direct and indirect methods to solve WLS with S-reduction trial subspaces. In particular, we consider:

- *Direct methods (discretize then optimize)*: A direct approach that leverages linear multistep methods to directly solve the optimization problem (3.7). Section 4.2 outlines this approach.
- *Indirect methods (optimize then discretize)*: An indirect approach that leverages the forward–backward sweep algorithm to solve the Euler–Lagrange equations (3.12)–(3.15). Section 4.3 outlines this technique.

We note that a variety of other approaches exist, and their investigation comprises the subject of future work.

4.2. S-reduction trial subspaces: direct solution approach

Direct approaches solve optimization problem (3.4) by "transcribing" the infinite-dimensional optimization problem into a finite-dimensional one by discretizing the state and objective functional in time. The minimization problem is then reformulated as a (non)linear optimization problem. A variety of direct solution approaches exist, including collocation approaches, spectral methods, and genetic algorithms. In the context of S-reduction trial subspaces, the most straightforward direct solution approach consists of the following steps: (1) numerically discretize the FOM ODE (and hence the *integrand* of the objective function in problem (3.7)) and (2) select a numerical quadrature rule to evaluate the *integral* defining the objective function in (3.7). To this end, we define time grids $\{\tau^{n,i}\}_{i=0}^{N_\tau^n} \subset [t_s^n, t_f^n]$, $n = 1, \dots, N_w$ that satisfy $t_s^n = \tau^{n,0} \leq \cdots \leq \tau^{n,N_\tau^n} = t_f^n$. Figure 3 depicts such a discretization. For the purposes of indexing between different windows, we additionally define a function:

$$\theta : (n,i) \mapsto \begin{cases} (n,i) & n = 1,\ i = 0, \\ (n,i) & n \geq 1,\ i > 0, \\ \theta(n-1, N_\tau^{n-1} + i) & n > 1,\ i \leq 0. \end{cases}$$

We now outline the direct solution approach for linear-multistep schemes; the formulation for other time-integration methods (e.g., Runge–Kutta) follows closely.

4.2.1. Linear multistep schemes

Linear multistep schemes approximate the solution at time instance $\tau^{n,i}$ using the previous $k^{n,i}$ time instances, where $k^{n,i}$ denotes the number of time steps employed by the scheme at the ith time instance of the nth window. Employing such a method to discretize the FOM ODE yields the sequence of FOM OΔEs defined over the nth time window

$$\boldsymbol{r}^{n,i}(\mathbf{x}^{n,i}; \mathbf{x}^{\theta(n,i-1)}, \dots, \mathbf{x}^{\theta(n,i-k^{n,i})}) = \mathbf{0}, \qquad i = 1, \dots, N_\tau^n$$

along with the initial condition $\mathbf{x}^{1,0} = \mathbf{x}_0$. Here, $\mathbf{x}^{n,i}(\approx \boldsymbol{x}(\tau^{n,i})) \in \mathbb{R}^N$ and $\boldsymbol{r}^{n,i}$ denotes the FOM OΔE residual over the ith time instance of the nth window defined as

$$\begin{aligned} \boldsymbol{r}^{n,i} : (\mathbf{y}^i; \mathbf{y}^{i-1}, \dots, \mathbf{y}^{i-k^{n,i}}) &\mapsto \frac{1}{\Delta t^{n,i}} \sum_{j=0}^{k^{n,i}} \alpha_j^{n,i} \mathbf{y}^{i-j} - \sum_{j=0}^{k^{n,i}} \beta_j^{n,i} \boldsymbol{f}(\mathbf{y}^{i-j}, \tau^{\theta(n,i-j)}), \\ &: \mathbb{R}^N \otimes \mathbb{R}^{k^{n,i}+1} \to \mathbb{R}^N. \end{aligned}$$

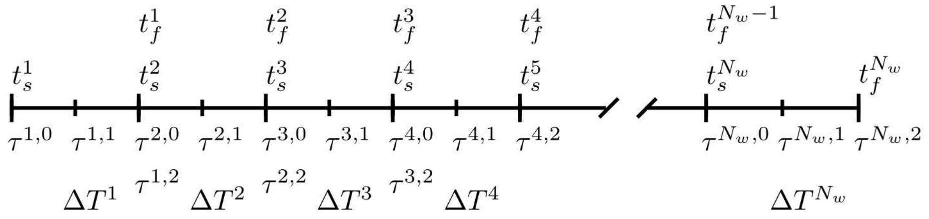

t_f^1
t_f^2
t_f^3
t_f^4
$t_f^{N_w-1}$
t_s^1
t_s^2
t_s^3
t_s^4
t_s^5
$t_s^{N_w}$
$t_f^{N_w}$
$\tau^{1,0}$ $\tau^{1,1}$ $\tau^{2,0}$ $\tau^{2,1}$ $\tau^{3,0}$ $\tau^{3,1}$ $\tau^{4,0}$ $\tau^{4,1}$ $\tau^{4,2}$
$\tau^{N_w,0}$ $\tau^{N_w,1}$ $\tau^{N_w,2}$
$\tau^{1,2}$ $\tau^{2,2}$ $\tau^{3,2}$
ΔT^1 ΔT^2 ΔT^3 ΔT^4 ΔT^{N_w}

Figure 3: Depiction of the $N_\tau^n + 1$ time instances over each window. In the figure, $N_\tau^n = 2$ for all n.

Here, $\Delta t^{n,i} := \tau^{n,i} - \tau^{n,i-1}$ denotes the time step, and $\alpha_j^{n,i}, \beta_j^{n,i} \in \mathbb{R}$ denote coefficients that define the specific type of multistep scheme at the ith time instance of the nth window. Employing a linear multistep method allows the objective *functional* (3.3) to be replaced with the objective *function*

$$J_{\mathrm{D}}^n : (\mathbf{y}^1, \ldots, \mathbf{y}^{N_\tau^n}; \mathbf{y}^0, \ldots, \mathbf{y}^{-k^{n,1}+1}) \mapsto \frac{1}{2} \sum_{i=1}^{N_\tau^n} \gamma^{n,i} [\boldsymbol{r}^{n,i}(\mathbf{y}^i; \mathbf{y}^{i-1}, \ldots, \mathbf{y}^{i-k^{n,i}})]^T \mathbf{A}^n \boldsymbol{r}^{n,i}(\mathbf{y}^i; \mathbf{y}^{i-1}, \ldots, \mathbf{y}^{i-k^{n,i}}),$$
$$: \mathbb{R}^N \otimes \mathbb{R}^{N_\tau^n + k^{n,1}} \to \mathbb{R}_+,$$

where $\gamma^{n,i} \in \mathbb{R}_+$ are quadrature weights.

WLS with the direct approach and a linear multistep method sequentially computes the solutions $\tilde{\mathbf{x}}^{n,1}, \ldots, \tilde{\mathbf{x}}^{n,N_\tau^n}$, $n = 1, \ldots, N_w$ that satisfy

$$\underset{(\mathbf{y}^1,\ldots,\mathbf{y}^{N_\tau^n}) \in (\mathcal{V} + \mathbf{x}_{\mathrm{ref}}^n) \otimes \mathbb{R}^{N_\tau^n}}{\text{minimize}} J_{\mathrm{D}}^n(\mathbf{y}^1, \ldots, \mathbf{y}^{N_\tau^n}; \tilde{\mathbf{x}}^{\theta(n,0)}, \ldots, \tilde{\mathbf{x}}^{\theta(n,-k^{n,1}+1)}),$$

with the initial condition $\tilde{\mathbf{x}}^{1,0} = \mathbf{V}^n [\mathbf{V}^n]^T (\mathbf{x}_0 - \mathbf{x}_{\mathrm{ref}}^n) + \mathbf{x}_{\mathrm{ref}}^n$.

Equivalently, WLS with the direct approach and a linear multistep method sequentially computes the generalized coordinates $\hat{\mathbf{x}}^{n,1}, \ldots, \hat{\mathbf{x}}^{n,N_\tau^n}$, $n = 1, \ldots, N_w$ with $\hat{\mathbf{x}}^{n,i} (\approx \hat{\boldsymbol{x}}^n(\tau^{n,i})) \in \mathbb{R}^{K^n}$ that satisfy

$$\underset{(\hat{\mathbf{y}}^1,\ldots,\hat{\mathbf{y}}^{N_\tau^n}) \in \mathbb{R}^{K^n} \otimes \mathbb{R}^{N_\tau^n}}{\text{minimize}} J_{\mathrm{D}}^n(\mathbf{V}^n \hat{\mathbf{y}}^1 + \mathbf{x}_{\mathrm{ref}}^n, \ldots, \mathbf{V}^n \hat{\mathbf{y}}^{N_\tau^n} + \mathbf{x}_{\mathrm{ref}}^n; \tilde{\mathbf{x}}^{\theta(n,0)}, \ldots, \tilde{\mathbf{x}}^{\theta(n,-k^{n,1}+1)}). \tag{4.1}$$

The optimization problem takes the form of a *weighted least-squares problem.* We emphasize that optimization problem (4.1) associates with an S-reduction trial subspace characterized by a reduction in spatial complexity, but no reduction in temporal complexity.

The minimization problem (4.1) requires specification of the quadrature weights (and hence the integration scheme used to discretize the objective functional). Typically, the same integration scheme used to discretize the FOM ODE is employed for consistency [53]; e.g., if a backward Euler method is used to discretize the FOM ODE, then a backward Euler method is the used to numerically integrate the objective functional.

Remark 4.1. *For the limiting case where $N_\tau^n = 1$, $n = 1, \ldots, N_w$ such that the window size is equivalent to the time step (i.e., $\Delta T^n = \tau^{n,1} - \tau^{n,0}$), uniform quadrature weights are used, a uniform trial space is employed (i.e., $\mathcal{V}^n = \mathcal{V}$ and $\mathbf{x}_{ref}^n = \mathbf{x}_{ref}, n = 1, \ldots, N_w$), the weighting matrices are taken to be $\mathbf{W}^n = \mathbf{W}, n = 1, \ldots, N_w$, and the time instances satisfy $\tau^{n,1} = t^n$, $n = 1, \ldots, N_w$, then $\tilde{\mathbf{x}}^{n,1} = \tilde{\mathbf{x}}_L^n$, $n = 1, \ldots, N_w$ and WLS with S-reduction trial subspaces solved via the direct approach recovers the LSPG approach.*

4.2.2. Solution to the least-squares problem through the Gauss–Newton method

Problem (4.1) corresponds to a discrete least-squares problem, which is nonlinear if the full-order-model velocity $\boldsymbol{f}$ is nonlinear in its first argument. A variety of algorithms exist for solving nonlinear least-squares problems, including the Gauss–Newton method, and the Levenberg–Marquardt method. The numerical experiments presented in this work consider nonlinear dynamical systems and are solved via the Gauss–Newton method; as such, we outline this approach here.

Defining a "vectorization" function

$$\begin{aligned}\boldsymbol{v} : (\mathbf{y}^1, \ldots, \mathbf{y}^m) &\mapsto \begin{bmatrix}[\mathbf{y}^1]^T & \ldots & [\mathbf{y}^m]^T\end{bmatrix}^T, \\ &: \mathbb{R}^p \otimes \mathbb{R}^m \to \mathbb{R}^{pm},\end{aligned}$$

the vectorized generalized coordinates over the nth time window are defined as

$$\hat{\bar{\bar{\mathbf{x}}}}^n := \boldsymbol{v}(\hat{\mathbf{x}}^{n,1}, \ldots, \hat{\mathbf{x}}^{n,N_\tau^n}).$$

We now define the weighted space–time residual over the entire window as

$$\bar{\bar{\boldsymbol{r}}}^n : \hat{\bar{\bar{\mathbf{y}}}} \mapsto \begin{bmatrix} \sqrt{\frac{\gamma^{n,1}}{2}}\mathbf{W}^n \boldsymbol{r}^{n,1}(\mathbf{V}^n\hat{\mathbf{y}}^1 + \mathbf{x}_{\text{ref}}^n; \tilde{\mathbf{x}}^{\theta(n,0)}, \ldots, \tilde{\mathbf{x}}^{\theta(n,1-k^{n,1})}) \\ \vdots \\ \sqrt{\frac{\gamma^{n,N_\tau^n}}{2}}\mathbf{W}^n \boldsymbol{r}^{n,N_\tau^n}(\mathbf{V}^n\hat{\mathbf{y}}^{N_\tau^n} + \mathbf{x}_{\text{ref}}^n; \mathbf{V}^n\hat{\mathbf{y}}^{N_\tau^n-1} + \mathbf{x}_{\text{ref}}^n, \ldots, \mathbf{V}^n\hat{\mathbf{y}}^{N_\tau^n - k^{n,N_\tau^n}} + \mathbf{x}_{\text{ref}}^n) \end{bmatrix},$$

where $\hat{\bar{\bar{\mathbf{y}}}} \equiv \boldsymbol{v}(\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^{N_\tau^n})$ and, for $i \leq 0$,

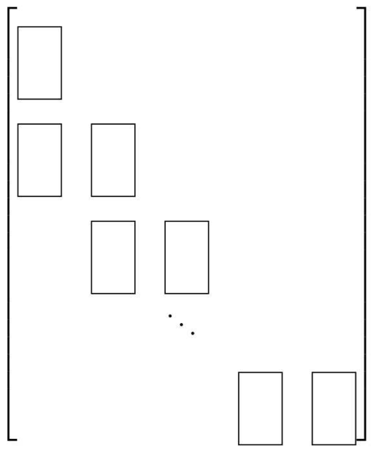
$$\hat{\mathbf{y}}^{n,i} \equiv \begin{cases} \hat{\mathbf{x}}^{\theta(n,i)} & n = 2, \ldots, N_w, \\ [\mathbf{V}^1]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}^1) & n = 1. \end{cases} \tag{4.2}$$

We note that

$$J_{\text{D}}^n\left(\tilde{\mathbf{x}}^{n,1}, \ldots, \tilde{\mathbf{x}}^{n,N_\tau^n}; \tilde{\mathbf{x}}^{\theta(n,0)}, \ldots, \tilde{\mathbf{x}}^{\theta(n,-k^{n,1}+1)}\right) = \left[\bar{\bar{\boldsymbol{r}}}^n(\hat{\bar{\bar{\mathbf{x}}}}^n)\right]^T \left[\bar{\bar{\boldsymbol{r}}}^n(\hat{\bar{\bar{\mathbf{x}}}}^n)\right].$$

Using these definitions, Algorithm 1 presents the standard Gauss–Newton method. Each Gauss–Newton iteration consists of three fundamental steps: (1) compute the FOM OΔE residual given the current guess, (2) compute the Jacobian of the residual over the time window, and (3) solve the linear least-squares problem and update the guess.

The practical implementation of the Gauss–Newton algorithm requires an efficient method for computing the Jacobian of the residual over the time window $\partial\bar{\bar{\boldsymbol{r}}}^n/\partial\hat{\bar{\bar{\mathbf{y}}}}$. For this purpose, we can leverage the fact that this Jacobian is a block lower triangular matrix with the following sparsity pattern (for $k^{n,i} = 1$, $i = 1, \ldots, N_\tau^n$):


,

where each block comprises an $N \times K^n$ dense matrix. In particular, solution techniques can leverage this structure, e.g., to efficiently compute Jacobian–vector products. Another consequence of this sparsity pattern is that the normal equations arising at each Gauss–Newton iteration comprise a banded block system that can also be exploited.

Remark 4.2. *(Acceleration of the Gauss–Newton Method)*
The principal cost of a Gauss–Newton method is often the formation of the Jacobian matrix. A variety of techniques aimed at reducing this computational burden exist; Jacobian-free Newton–Krylov methods [38], Broyden's method [11] (as explored in Ref. [16], Appendix A), and frozen Jacobian approximations are several such examples. Further, the space–time formulation introduces an extra dimension for parallelization that can be exploited to future reduce the wall time. The investigation of these additional, potentially more efficient, solution algorithms is a topic of future work.

Algorithm 1: S-reduction trial subspace: algorithm for the direct solution technique with the Gauss–Newton method and a linear multistep method over the nth window

```
Input  : tolerance, $\epsilon$; initial guess, $\hat{\mathbf{x}}_0^{n,1}, \ldots, \hat{\mathbf{x}}_0^{n,N_\tau^n}$
Output: Solution to least squares problem, $\hat{\bar{\bar{\mathbf{x}}}}^n$
Online Steps:
converged ← false                                                    ▷ Set convergence checker
$\hat{\bar{\bar{\mathbf{x}}}}_0^n \leftarrow \boldsymbol{v}(\hat{\mathbf{x}}_0^{n,1}, \ldots, \hat{\mathbf{x}}_0^{n,N_\tau^n})$        ▷ Assemble generalized coordinates over window
$k \leftarrow 0$                                                             ▷ Set counter
while not converged do
    $\mathbf{r} \leftarrow \bar{\bar{\boldsymbol{r}}}^n(\hat{\bar{\bar{\mathbf{x}}}}_k^n)$                         ▷ Compute weighted residual over window
    $\mathbf{J} \leftarrow \frac{\partial \bar{\bar{\boldsymbol{r}}}^n}{\partial \hat{\bar{\bar{\mathbf{y}}}}}(\hat{\bar{\bar{\mathbf{x}}}}_k^n)$                  ▷ Compute weighted residual-Jacobian over window
    if $\|\mathbf{J}^T\mathbf{r}\|_2 \leq \epsilon$ then
        converged ← true                 ▷ Check and set convergence based on gradient norm
        Return: $\hat{\bar{\bar{\mathbf{x}}}}^n = \hat{\bar{\bar{\mathbf{x}}}}_{k+1}^n$                              ▷ Return converged solution
    else
        Compute $\Delta\hat{\bar{\bar{\mathbf{x}}}}^n$ that minimizes $\|\mathbf{J}\Delta\hat{\bar{\bar{\mathbf{x}}}}^n + \mathbf{r}\|_2^2$      ▷ Solve the linear least-squares problem
        $\alpha \leftarrow$ linesearch($\Delta\hat{\bar{\bar{\mathbf{x}}}}^n, \hat{\bar{\bar{\mathbf{x}}}}_k^n$)        ▷ Compute $\alpha$ based on a line search, or set to 1
        $\hat{\bar{\bar{\mathbf{x}}}}_{k+1}^n \leftarrow \hat{\bar{\bar{\mathbf{x}}}}_k^n + \alpha\Delta\hat{\bar{\bar{\mathbf{x}}}}^n$                         ▷ Update guess to the state
    end
    $k \leftarrow k + 1$
end
```

4.3. S-reduction trial subspaces: indirect solution approach

In contrast to the direct approach, indirect methods "indirectly" solve the minimization problem (3.4) by solving the Euler–Lagrange equations (3.12)–(3.13) associated with stationarity. This system comprises a coupled two-point boundary value problem. Several techniques have been devised to solve such problems, including shooting methods, multiple shooting methods [47], and the forward–backward sweep method [44] (FBSM). This work explores using the FBSM.

4.3.1. Forward–backward sweep method (FBSM)

Until convergence, the FBSM alternates between solving the system (3.12) *forward* in time given a fixed value of the costate, and solving the adjoint equation (3.13) *backward* in time given a fixed value for the generalized coordinates. Typically, the system (3.12) is solved first given an initial guess for the costate. Algorithm 2 outlines the algorithm, which contains three parameters: the relaxation factor $\rho \leq 1$, the growth factor $\psi_1 \geq 1$, and the decay factor $\psi_2 \geq 1$. The relaxation factor controls the rate at which the costate seen by (3.12) is updated. The closer ρ is to unity, the faster the algorithm will converge. For large window sizes, however, a large a value of ρ can lead to an unstable iterative process. In practice, a line search is used to compute an acceptable value for the relaxation factor ρ. The line search presented in Algorithm 2 adapts the relaxation factor according to the objective. Convergence properties of the FBSM method are presented in Ref. [45], which shows that the algorithm will converge for a sufficiently small value of ρ.

4.3.2. Considerations for the numerically solving the forward and backward systems

The FBSM requires solving the forward (3.12) and backward (3.13) systems, both of which are defined at the time-continuous level. The numerical implementation of the FBSM requires two main ingredients: (1) temporal discretization schemes for the forward and backward problems and (2) an efficient method for computing terms involving the transpose of the Jacobian of the velocity.

This work employs linear multistep schemes for time discretization of the forward and backward problems. As described in Section 4.2.1, temporal discretization is achieved by introducing $N_\tau^n + 1$ time instances over each time window.

The second ingredient, namely devising an efficient method for computing terms involving the transpose of the Jacobian of the velocity, can be challenging if one does not have explicit access to this

Algorithm 2: S-reduction trial subspace: algorithm for the FBSM over the nth window.

```
Input  : tolerance, $\epsilon$; relaxation factor, $\rho \leq 1$; growth factor, $\psi_1 \geq 1$; decay factor, $\psi_2 \geq 1$; initial
         guess for state $\hat{\boldsymbol{x}}_0^n$; initial guess for costate $\hat{\boldsymbol{u}}^n$
Output: Stationary point, $\hat{\boldsymbol{x}}^n$
Online Steps:
Compute $\hat{\boldsymbol{x}}_1^n$ satisfying $\mathbf{M}^n\dot{\hat{\boldsymbol{x}}}_1^n(t) - [\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}_1^n(t) + \mathbf{x}_{\text{ref}}^n, t) = \mathbf{M}^n\hat{\boldsymbol{u}}^n(t)$    ▷ Solve (3.12)
$i \leftarrow 1$    ▷ Set counter
while $\epsilon \leq \int_{t_s^n}^{t_f^n} \|\hat{\boldsymbol{x}}_i^n(t) - \hat{\boldsymbol{x}}_{i-1}^n(t)\|_2 dt$ do
    Compute $\hat{\boldsymbol{\lambda}}^n$ satisfying $\mathbf{M}^n\dot{\hat{\boldsymbol{\lambda}}}^n(t) + [\mathbf{V}^n]^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}_i^n(t) + \mathbf{x}_{\text{ref}}^n, t)\right]^T\mathbf{A}^n\mathbf{V}^n\hat{\boldsymbol{\lambda}}^n(t) =$
      $-\left[[\mathbf{V}^n]^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}_i^n(t)+\mathbf{x}_{\text{ref}}^n, t)\right]^T\mathbf{A}^n\left(\mathbf{I}-\mathbf{V}^n[\mathbf{M}^n]^{-1}[\mathbf{V}^n]^T\mathbf{A}^n\right)\left(\mathbf{V}^n\dot{\hat{\boldsymbol{x}}}_i^n(t)-\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}_i^n(t)+\mathbf{x}_{\text{ref}}^n, t)\right)\right]$
                                                    ▷ Solve (3.13) to obtain guess to costate
    $\hat{\boldsymbol{u}}^n \leftarrow \rho\hat{\boldsymbol{u}}^n + (1-\rho)\hat{\boldsymbol{\lambda}}^n$    ▷ Weighted update to costate
    $i \leftarrow i+1$    ▷ Update counter
    Compute $\hat{\boldsymbol{x}}_i^n$ satisfying $\mathbf{M}^n\dot{\hat{\boldsymbol{x}}}_i^n(t) - [\mathbf{V}^n]^T\mathbf{A}^n\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}_i^n(t) + \mathbf{x}_{\text{ref}}^n, t) = \mathbf{M}^n\hat{\boldsymbol{u}}^n(t)$    ▷ Solve (3.12)
    if $\mathcal{J}^n(\mathbf{V}^n\hat{\boldsymbol{x}}_i^n + \mathbf{x}_{ref}^n \otimes \mathcal{O}^n) \leq \mathcal{J}^n(\mathbf{V}^n\hat{\boldsymbol{x}}_{i-1}^n + \mathbf{x}_{ref}^n \otimes \mathcal{O}^n)$ then
        $\rho \leftarrow \min(\rho\psi_1, 1)$    ▷ Grow the relaxation factor
    else
        $\rho \leftarrow \frac{\rho}{\psi_2}$    ▷ Shrink the relaxation factor
        $\hat{\boldsymbol{x}}_i^n \leftarrow \hat{\boldsymbol{x}}_{i-1}^n$    ▷ Reset state to value at previous iteration
    end
end
Return converged solution, $\hat{\boldsymbol{x}}^n = \hat{\boldsymbol{x}}_i^n$
```

Jacobian or it is too costly to compute. We discuss two methods that can be used to evaluate such terms that appear in the forward system (3.13):

1. *Jacobian-free approximation*: A non-intrusive way to evaluate these terms is to recognize that all terms including the transpose of the Jacobian of the velocity are left multiplied by the transpose of the spatial trial basis; this can be manipulated as

$$[\mathbf{V}^n]^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\right]^T = \left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t)\mathbf{V}^n\right]^T,$$

   which exposes the ability to approximate rows of this matrix via finite differences, e.g., via forward differences as

$$\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t)+\mathbf{x}_{\text{ref}}^n, t)\mathbf{v}_i^n \approx \frac{1}{\epsilon}\left(\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t)+\mathbf{x}_{\text{ref}}^n+\epsilon\mathbf{v}_i^n, t) - \boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t)+\mathbf{x}_{\text{ref}}^n, t)\right), \quad i = 1, \ldots, K^n,$$

   which requires $K^n + 1$ evaluations of the velocity.
2. *Automatic differentiation*: A more intrusive, but exact method for computing these terms is through automatic differentiation (AD), which evaluate derivatives of functions (e.g., Jacobians, vector-Jacobian products) in a numerically exact manner by recursively applying the chain rule. The numerical examples presented later in this work leverage AD. The principal drawback of this approach is its intrusiveness, which may prevent them from practical application, e.g., in legacy codes.

Remark 4.3. *(Acceleration of Indirect Methods) The FBSM is a simple iterative method for solving the coupled two-point boundary value problem. For large time windows, however, the FBSM may require many forward–backward iterations for convergence. More sophisticated solution techniques, such as a multiple FBSM method or multiple shooting methods, can reduce this cost in principle. Analyzing additional solution techniques is the subject of future work.*

4.4. ST-reduction trial subspaces: direct and indirect methods

We now consider ST-reduction trial subspaces. Because the optimization variables (i.e., the generalized coordinates) in this case are already finite dimensional, solvers for this type of trial subspace need only develop a finite-dimensional representation of the objective functional in problem (3.23). We describe two techniques for this purpose: a direct method that operates on the FOM OΔE and an indirect method that operates on the FOM ODE.

4.5. ST-reduction trial subspaces: direct solution approach

The direct solution technique seeks to minimize the fully discrete objective function associated with the FOM OΔE. We consider linear multistep methods and leverage the discretization introduced in Section 4.2. For notational simplicity, we define an index-mapping function that is equivalent to the mapping function θ, but outputs only the first argument:

$$\theta^* : (n,i) \mapsto \begin{cases} n & n = 1,\ i = 0, \\ n & n \geq 1,\ i > 0, \\ \theta^*(n-1, N_\tau^{n-1} + i) & n > 1,\ i \leq 0. \end{cases}$$

WLS with an ST-reduction trial subspace and the direct approach sequentially computes the generalized coordinates $\hat{\hat{\mathbf{x}}}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\begin{split}\underset{\hat{\hat{\mathbf{y}}} \in \mathbb{R}^{K_{\mathrm{ST}}^n}}{\text{minimize}}\ J_{\mathrm{D}}^n(\mathbf{\Pi}^n(\tau^{n,1})\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\mathrm{st}}^n, \ldots, \mathbf{\Pi}^n(\tau^{n,N_\tau^n})\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\mathrm{st}}^n; \\ \tilde{\boldsymbol{x}}^{\theta^*(n,0)}(\tau^{\theta(n,0)}), \ldots, \tilde{\boldsymbol{x}}^{\theta^*(n,-k^{n,1}+1)}(\tau^{\theta(n,-k^{n,1}+1)})).\end{split} \tag{4.3}$$

The boundary conditions are automatically satisfied through the definition of the ST-reduction trial subspace. Assuming $\mathrm{Rank}(\mathbf{W}^n)N_\tau^n \geq K_{\mathrm{ST}}^n$, the minimization problem (4.3) again yields a least-squares problem.

Remark 4.4. *Comparing optimization problems (4.1) and (4.3) reveals that WLS with the direct solution approach minimizes the same objective function in the case of both ST-reduction and S-reduction trial subspaces.*

Remark 4.5. *For the limiting case where one window comprises the entire domain (i.e., $\Delta T^1 \equiv T$), uniform quadrature weights are used, the trial subspace is set to be $\mathcal{ST}_{ST}^1 = \mathcal{ST}_{ST}$, the weighting matrix $\mathbf{W}_{ST} = \mathrm{diag}(\mathbf{W}^1)$, and $N_\tau^1 = N_t$ time instances are employed that satisfy $\tau^{1,i} = t^i$, $i = 1, \ldots, N_t$, then $\hat{\hat{\mathbf{x}}}^1 = \hat{\hat{\mathbf{x}}}_{ST\text{-}LSPG}$ and direct WLS with an ST-reduction trial subspace recovers ST-LSPG.*

Remark 4.6. *To enable equivalence in the case for a general ST-LSPG weighting matrix $\mathbf{W}_{ST}$, the weighting matrix $\mathbf{A}^1$ must be time dependent matrix-valued, which associates the objective function (4.3) with a modified space–time norm. For notational simplicity, we do not consider this case in the current manuscript.*

4.6. ST-reduction trial subspaces: indirect solution approach

As opposed to the direct approach, the indirect approach directly minimizes the continuous objective function (3.23) and sequentially computes solutions $\hat{\hat{\mathbf{x}}}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\underset{\hat{\hat{\mathbf{y}}} \in \mathbb{R}^{K_{\mathrm{ST}}^n}}{\text{minimize}}\ \mathcal{J}^n\left(\mathbf{\Pi}^n\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\mathrm{st}}^n \otimes \mathcal{O}^n\right). \tag{4.4}$$

Numerically solving the minimization problem requires the introduction of a quadrature rule for discretization of the integral. To this end, we introduce $N_{\mathrm{ST}}^n \geq \mathrm{ceil}(K_{\mathrm{ST}}^n/\mathrm{rank}(\mathbf{W}^n))$ quadrature points over the nth window, $\{\chi_{\mathrm{ST}}^{n,i}\}_{i=1}^{N_{\mathrm{ST}}^n} \subset [t_s^n, t_f^n]$, $n = 1, \ldots, N_w$. Leveraging these quadrature points, WLS with the indirect method and an ST-reduction trial subspace computes the generalized coordinates $\hat{\hat{\mathbf{x}}}^n$, $n = 1, \ldots, N_w$ that satisfy

$$\underset{\hat{\hat{\mathbf{y}}} \in \mathbb{R}^{K_{\mathrm{ST}}^n}}{\text{minimize}}\ J_{\mathrm{ST}}^n\left(\mathbf{\Pi}^n\hat{\hat{\mathbf{y}}} + \mathbf{x}_{\mathrm{st}}^n \otimes \mathcal{O}^n\right), \tag{4.5}$$

where the discrete objective function is given by

$$J^n_{\mathrm{ST}} : \boldsymbol{y} \mapsto \frac{1}{2}\sum_{i=1}^{N^n_{\mathrm{ST}}} \zeta^{n,i}\Big[\dot{\boldsymbol{y}}(\chi^{n,i}_{\mathrm{ST}}) - \boldsymbol{f}(\boldsymbol{y}(\chi^{n,i}_{\mathrm{ST}}),\chi^{n,i}_{\mathrm{ST}})\Big]^T \mathbf{A}^n \Big[\dot{\boldsymbol{y}}(\chi^{n,i}_{\mathrm{ST}}) - \boldsymbol{f}(\boldsymbol{y}(\chi^{n,i}_{\mathrm{ST}}),\chi^{n,i}_{\mathrm{ST}})\Big],$$
$$: \mathbb{R}^N \otimes \mathcal{T}^n \to \mathbb{R}_+,$$

and $\zeta^{n,i} \in \mathbb{R}_+$, $i = 1,\ldots,N^n_{\mathrm{ST}}$ are quadrature weights over the nth time window. The optimization problem (4.5) again comprises a least-squares problem.

Remark 4.7. *WLS with ST-reduction trial subspaces solved via the indirect approach naturally achieves "collocation" in time as the full-order model residual needs to be queried at only the quadrature points.*

Remark 4.8. *For the limiting case where one window comprises the entire domain (i.e., $\Delta T^1 \equiv T$), the trial subspace is set to the span of full solution trajectories, $\mathcal{ST}^1_{ST} = span\{\boldsymbol{x}_i\}_{i=1}^{K^1_{ST}}$ (where $\boldsymbol{x}_i$ are obtained, e.g., from training simulations at different parameter instances), and the weighting matrix is set to $\mathbf{A}^1 = \mathbf{I}$, WLS with ST-reduction trial subspaces and the indirect approach closely resembles the model reduction procedure proposed in Ref. [26]; the approaches differ only in that Ref. [26] imposes the constraint $\sum_{i=1}^{K^1_{ST}} \hat{\mathbf{x}}^1_i = 1$ in the associated minimization problem.*

4.7. ST-reduction trial subspaces: summary

ST-reduction trial subspaces yield a series of space–time systems of algebraic equations over each window. As a variety of work has examined space–time reduced-order models with ST-reduction trial subspaces, a detailed exposition of solution techniques for these systems is not pursued here. It is sufficient to say that WLS with ST-reduction trial subspaces yields a series of dense systems to be solved over each window.

5. Analysis

This section provides theoretical analyses of the WLS approach. First, we demonstrate equivalence conditions between WLS with (uniform) S-reduction trial subspaces and the Galerkin ROM in the limit $\Delta T^n \to 0$. Next, we derive *a priori* error bounds for autonomous systems.

5.1. Equivalence conditions

Theorem 5.1. *(Galerkin equivalence) For sequential minimization over infinitesimal time windows and uniform S-reduction trial spaces, i.e., $\mathcal{V}^n = \mathcal{V}$ and $\mathbf{x}^n_{ref} = \mathbf{x}_{ref}$, $n = 1,\ldots,N_w$, the WLS approach (weakly) recovers Galerkin projection.*

Proof. The WLS approach with uniform S-reduction subspaces comprises solving the following sequence of minimization problems for $\hat{\boldsymbol{x}}^n$, $n = 1,\ldots,N_w$,

$$\begin{aligned}&\underset{\hat{\boldsymbol{y}}\in\mathbb{R}^K\otimes\mathcal{T}^n}{\text{minimize}}\ \mathcal{J}^n(\mathbf{V}\hat{\boldsymbol{y}} + \mathbf{x}_{\mathrm{ref}}\otimes\mathcal{O}^n),\\ &\text{subject to }\ \hat{\boldsymbol{y}}(t^n_s) = \begin{cases}\hat{\boldsymbol{x}}^{n-1}(t^{n-1}_f) & n = 2,\ldots,N_w\\ [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\mathrm{ref}}) & n = 1.\end{cases}\end{aligned} \tag{5.1}$$

Following the derivation of the Euler–Lagrange equations presented in Appendix C leads to Eq. (C.5). Setting $a = t^n_s$, $b = t^n_f$, and $\mathcal{I} = \mathcal{I}^n$ in Eq. (C.5) yields the sequence of systems to be solved for $\hat{\boldsymbol{x}}^n$ (and, implicitly, $\dot{\hat{\boldsymbol{x}}}^n$) over $t \in [t^n_s, t^n_f]$:

$$\int_{t^n_s}^{t^n_f}\left(\frac{\partial\mathcal{I}^n}{\partial\hat{\mathbf{y}}}(\hat{\boldsymbol{x}}^n(t),\dot{\hat{\boldsymbol{x}}}^n(t),t)\boldsymbol{\eta}^n(t) - \frac{d}{dt}\left(\frac{\partial\mathcal{I}^n}{\partial\hat{\mathbf{v}}}(\hat{\boldsymbol{x}}^n(t),\dot{\hat{\boldsymbol{x}}}^n(t),t)\right)\boldsymbol{\eta}^n(t)\right)dt + \left(\frac{\partial\mathcal{I}^n}{\partial\hat{\mathbf{v}}}(\hat{\boldsymbol{x}}^n(t^n_f),\dot{\hat{\boldsymbol{x}}}^n(t^n_f),t^n_f)\right)\boldsymbol{\eta}^n(t^n_f) = 0, \tag{5.2}$$

for all functions $\boldsymbol{\eta}^n : [t_s^n, t_f^n] \to \mathbb{R}^K$ that satisfy $\boldsymbol{\eta}^n(t_s^n) = \mathbf{0}$, with the boundary conditions

$$\hat{\boldsymbol{x}}^n(t_s^n) = \begin{cases} \hat{\boldsymbol{x}}^{n-1}(t_f^{n-1}) & n = 2, \ldots, N_w, \\ [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}) & n = 1. \end{cases}$$

To examine what happens for infinitesimal time windows, we take a uniform window size and let $t_f^n = t_s^n + \zeta$, $n = 1, \ldots, N_w$ such that $t_s^n = \zeta(n-1)$ and $t_f^n = \zeta n$, $n = 1, \ldots, N_w$. Taking the limit $\zeta \to 0^+$ and noting that $\boldsymbol{\eta}^n$ is an arbitrary function, we obtain the following sequence of problems for $n = 1, \ldots, N_w$:

$$\hat{\boldsymbol{x}}^n(\zeta(n-1)) = \begin{cases} \hat{\boldsymbol{x}}^{n-1}(\zeta(n-1)) & n = 2, \ldots, N_w, \\ [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}}) & n = 1, \end{cases} \qquad \left[\frac{\partial \mathcal{I}^n}{\partial \hat{\mathbf{v}}}(\hat{\boldsymbol{x}}^n(\zeta n), \dot{\hat{\boldsymbol{x}}}^n(\zeta n), \zeta n)\right]^T = \mathbf{0},$$

where the first term in Eq. (5.2) has vanished, as $\lim_{\zeta \to 0^+} \int_{\zeta(n-1)}^{\zeta n} h(t)dt = 0$ for any continuous function h. Noting that the derivative evaluates to

$$\left[\frac{\partial \mathcal{I}^n}{\partial \hat{\mathbf{v}}}(\hat{\boldsymbol{x}}^n(\zeta n), \dot{\hat{\boldsymbol{x}}}^n(\zeta n), \zeta n)\right]^T = \mathbf{V}^T\mathbf{A}\mathbf{V}\dot{\hat{\boldsymbol{x}}}^n(\zeta n) - \mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}^n(\zeta n) + \mathbf{x}_{\text{ref}}, \zeta n),$$

we have

$$\mathbf{V}^T\mathbf{A}\mathbf{V}\dot{\hat{\boldsymbol{x}}}^n(\zeta n) - \mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}^n(\zeta n) + \mathbf{x}_{\text{ref}}, \zeta n) = \mathbf{0}, \qquad n = 1, \ldots, N_w,$$

with the boundary conditions $\hat{\boldsymbol{x}}^n(\zeta(n-1)) = \hat{\boldsymbol{x}}^{n-1}(\zeta(n-1))$ for $n = 2, \ldots, N_w$ and $\hat{\boldsymbol{x}}^1(0) = [\mathbf{V}]^T(\mathbf{x}_0 - \mathbf{x}_{\text{ref}})$. In the limit of $\zeta \to 0^+$ (and hence $N_w \to \infty$) this is a (weak) statement of the Galerkin ROM. □

5.2. *A priori error bounds*

We now derive *a priori* error bounds for S-reduction trial subspaces in the case that no weighting matrix is employed (i.e., $\mathbf{A} = \mathbf{I}$). We denote the error in the WLS ROM solution over the nth window as

$$\begin{aligned} \boldsymbol{e}^n :& \tau \mapsto \boldsymbol{x}(\tau) - \tilde{\boldsymbol{x}}^n(\tau), \\ :& [t_s^n, t_f^n] \to \mathbb{R}^N, \end{aligned}$$

$n = 1, \ldots, N_w$. Additionally, we denote $\tilde{\boldsymbol{x}}_{\ell^2}^n$, $n = 1, \ldots, N_w$ to be the ℓ^2-optimal solution over the nth window

$$\tilde{\boldsymbol{x}}_{\ell^2}^n = \underset{\boldsymbol{y} \in \mathcal{ST}^n}{\arg\min} \int_{t_s^n}^{t_f^n} \|\boldsymbol{y}(t) - \boldsymbol{x}(t)\|_2^2 dt.$$

We employ the following assumptions.

- **A1:** The residual is Lipshitz continuous in the first argument, i.e., there exists $\kappa > 0$ such that

  $$\|\boldsymbol{r}(\boldsymbol{w}, \tau) - \boldsymbol{r}(\boldsymbol{y}, \tau)\|_2 \leq \kappa \|\boldsymbol{w}(\tau) - \boldsymbol{y}(\tau)\|_2, \quad \forall \boldsymbol{w},\, \boldsymbol{y} \in \mathbb{R}^N \otimes \mathcal{T}, \tau \in [0, T],$$

  where

  $$\begin{aligned} \boldsymbol{r} :\ & (\boldsymbol{y}, \tau) \mapsto \dot{\boldsymbol{y}}(\tau) - \boldsymbol{f}(\boldsymbol{y}(\tau), \tau), \\ :\ & \mathbb{R}^N \otimes \mathcal{T} \times [0, T] \mapsto \mathbb{R}^N. \end{aligned}$$

- **A2:** The velocity is Lipshitz continuous in its first argument, i.e., there exists $\Gamma > 0$ such that

  $$\|\boldsymbol{f}(\mathbf{w}, \tau) - \boldsymbol{f}(\mathbf{y}, \tau)\|_2 \leq \Gamma \|\mathbf{w} - \mathbf{y}\|_2, \quad \forall \mathbf{w},\, \mathbf{y} \in \mathbb{R}^N, \tau \in [0, T].$$

- **A3:** The integrated residual is inverse Lipshitz continuous in its first argument over each time window, i.e., there exist $\alpha^n > 0$, $n = 1, \ldots, N_w$ such that

  $$\int_{t_s^n}^{t_f^n} \|\boldsymbol{w}(t) - \boldsymbol{y}(t)\|_2 dt \leq \alpha^n \int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\boldsymbol{w}, t) - \boldsymbol{r}(\boldsymbol{y}, t)\|_2 dt, \quad \forall \boldsymbol{w}, \boldsymbol{y} \in \mathcal{ST}_*^n,$$

  where $\mathcal{ST}_*^n = \{\boldsymbol{w} \in \mathbb{R}^N \otimes \mathcal{T}^n \,|\, \boldsymbol{w}(t_s^n) = \boldsymbol{x}(t_s^n)\}$.

- **A4:** The FOM solution at the start of each time window lies within the range of the trial subspace, i.e.,
$$\boldsymbol{x}^n(t_s^n) \in \mathcal{V}^n + \mathbf{x}_{\text{ref}}^n, \quad n = 1, \dots, N_w.$$

Theorem 5.2. *(A priori error bounds) Under Assumptions A1–A4, the error in the solution computed by the WLS ROM approach with S-reduction trial subspaces over the nth window is bounded as*

$$\int_{t_s^n}^{t_f^n} \|\boldsymbol{e}^n(t)\|_2 dt \le \|\boldsymbol{e}^n(t_s^n)\|_2 \bigg(\frac{e^{\Delta T^n(\kappa+\Gamma)} - 1}{\kappa + \Gamma} \bigg) + \alpha^n \int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}_{\ell^2}^n, t)\|_2 dt. \tag{5.3}$$

Proof. To obtain an error bound over the nth window, we must account for the fact that the initial conditions into the nth window can be incorrect. To this end, we define new quantities $\tilde{\boldsymbol{x}}_*^n$, $n = 1, \dots, N_w$, where $\tilde{\boldsymbol{x}}_*^n$ is the solution to the minimization problem

$$\begin{aligned} &\underset{\boldsymbol{y} \in \mathcal{ST}^n}{\text{minimize}} \ \mathcal{J}^n(\boldsymbol{y}), \\ &\text{subject to } \boldsymbol{y}(t_s^n) = \boldsymbol{x}(t_s^n). \end{aligned} \tag{5.4}$$

Note that minimization problem (5.4) is equivalent to the WLS minimization problem (3.4), but uses the FOM solution for the initial conditions. Additionally, define $\hat{\boldsymbol{\lambda}}_*^n$, $n = 1, \dots, N_w$ to be the costate solution associated with optimization problem (5.4). The error in the solution obtained by the WLS ROM over the nth window at time $t \in [t_s^n, t_f^n]$ can be written as

$$\|\tilde{\boldsymbol{x}}^n(t) - \boldsymbol{x}(t)\|_2 = \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t) + \tilde{\boldsymbol{x}}_*^n(t) - \boldsymbol{x}(t)\|_2.$$

Applying triangle inequality yields

$$\|\tilde{\boldsymbol{x}}^n(t) - \boldsymbol{x}(t)\|_2 \le \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t)\|_2 + \|\tilde{\boldsymbol{x}}_*^n(t) - \boldsymbol{x}(t)\|_2.$$

Integrating over the nth window and using the definition of the error yields

$$\int_{t_s^n}^{t_f^n} \|\boldsymbol{e}^n(t)\|_2 dt \le \int_{t_s^n}^{t_f^n} \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t)\|_2 dt + \int_{t_s^n}^{t_f^n} \|\tilde{\boldsymbol{x}}_*^n(t) - \boldsymbol{x}(t)\|_2 dt.$$

Applying Assumption A3 and $\boldsymbol{r}(\boldsymbol{x}, t) = \mathbf{0}$, $\forall t \in [0, T]$ yields

$$\int_{t_s^n}^{t_f^n} \|\boldsymbol{e}^n(t)\|_2 dt \le \int_{t_s^n}^{t_f^n} \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t)\|_2 dt + \alpha^n \int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}_*^n, t)\|_2 dt.$$

Leveraging the residual-minimization property of WLS and noting that $\tilde{\boldsymbol{x}}_{\ell^2}^n(t_s^n) = \boldsymbol{x}^n(t_s^n)$ by Assumption A4, we have

$$\int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}_*^n, t)\|_2 dt \le \int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}_{\ell^2}^n, t)\|_2 dt.$$

This leads to the following expression for the error over the nth window,

$$\int_{t_s^n}^{t_f^n} \|\boldsymbol{e}^n(t)\|_2 dt \le \int_{t_s^n}^{t_f^n} \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t)\|_2 dt + \alpha^n \int_{t_s^n}^{t_f^n} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}_{\ell^2}^n, t)\|_2 dt. \tag{5.5}$$

We now derive an upper bound for $\int_{t_s^n}^{t_f^n} \|\tilde{\boldsymbol{x}}^n(t) - \tilde{\boldsymbol{x}}_*^n(t)\|_2 dt$. Defining $\hat{\mathbf{e}}_*^n = \hat{\boldsymbol{x}}^n - \hat{\boldsymbol{x}}_*^n$, $n = 1, \dots, N_w$, where $\hat{\boldsymbol{x}}_*^n$ are the generalized coordinates of $\tilde{\boldsymbol{x}}_*^n$ (i.e., $\tilde{\boldsymbol{x}}_*^n(t) = \mathbf{V}^n \hat{\boldsymbol{x}}_*^n(t) + \mathbf{x}_{\text{ref}}^n$), the differential equation for $\hat{\mathbf{e}}_*^n$ is given by

$$\dot{\hat{\mathbf{e}}}_*^n(t) = [\mathbf{V}^n]^T [\boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t) - \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}_*^n(t) + \mathbf{x}_{\text{ref}}^n, t)] + \hat{\boldsymbol{\lambda}}^n(t) - \hat{\boldsymbol{\lambda}}_*^n(t),$$

for $t \in [t_s^n, t_f^n]$ and with the initial condition $\hat{\mathbf{e}}_*^n(t_s^n)$. We have used the notation $\dot{\hat{\mathbf{e}}}_*^n \equiv d\hat{\mathbf{e}}_*^n / d\tau$. Taking the norm of both sides and applying triangle inequality yields

$$\|\dot{\hat{\mathbf{e}}}_*^n(t)\|_2 \le \|[\boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}^n(t) + \mathbf{x}_{\text{ref}}^n, t) - \boldsymbol{f}(\mathbf{V}^n \hat{\boldsymbol{x}}_*^n(t) + \mathbf{x}_{\text{ref}}^n, t)]\|_2 + \|\hat{\boldsymbol{\lambda}}^n(t) - \hat{\boldsymbol{\lambda}}_*^n(t)\|_2,$$

with the initial condition $\|\hat{\boldsymbol{e}}^n_*(t^n_s)\|_2$. We note we have used $\|\mathbf{V}^n\|_2 = 1$. Using the definition of the costate (3.16) yields

$$\|\dot{\hat{\boldsymbol{e}}}^n_*(t)\|_2 \leq \|[\boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n(t) + \mathbf{x}^n_{\text{ref}}, t) - \boldsymbol{f}(\mathbf{V}^n\hat{\boldsymbol{x}}^n_*(t) + \mathbf{x}^n_{\text{ref}}, t)]\|_2 + \|[\mathbf{V}^n]^T\big(\boldsymbol{r}(\tilde{\boldsymbol{x}}^n, t) - \boldsymbol{r}(\tilde{\boldsymbol{x}}^n_*, t)\big)\|_2.$$

Employing assumptions A1-A2 yields the bound

$$\|\dot{\hat{\boldsymbol{e}}}^n_*(t)\|_2 \leq (\Gamma + \kappa)\|\hat{\boldsymbol{e}}^n_*(t)\|_2.$$

We use the fact that $d\|\hat{\boldsymbol{e}}^n_*\|_2/d\tau \leq \|\dot{\hat{\boldsymbol{e}}}^n_*\|_2$ to get

$$(\frac{d\|\hat{\boldsymbol{e}}^n_*\|_2}{d\tau})(t) \leq (\Gamma + \kappa)\|\hat{\boldsymbol{e}}^n_*(t)\|_2.$$

The above is a linear homogeneous equation for the bound of $\|\hat{\boldsymbol{e}}^n_*\|_2$ and has the solution for $t \in [t^n_s, t^n_f]$

$$\|\hat{\boldsymbol{e}}^n_*(t)\|_2 \leq \|\hat{\boldsymbol{e}}^n_*(t^n_s)\|_2 e^{(\kappa+\Gamma)(t-t^n_s)}.$$

Noting that $\|\hat{\boldsymbol{e}}^n_*(t^n_s)\|_2 = \|\boldsymbol{e}^n(t^n_s)\|_2$ we get the bound

$$\int_{t^n_s}^{t^n_f} \|\hat{\boldsymbol{e}}^n_*(t)\|_2 dt \leq \|\boldsymbol{e}^n(t^n_s)\|_2 \bigg(\frac{e^{\Delta T^n(\kappa+\Gamma)} - 1}{\kappa + \Gamma}\bigg). \tag{5.6}$$

Substituting bound (5.6) into bound (5.5) and noting that $\|\tilde{\boldsymbol{x}}^n - \tilde{\boldsymbol{x}}^n_*\|_2 = \|\hat{\boldsymbol{x}}^n - \hat{\boldsymbol{x}}^n_*\|_2$ gives the upper bound

$$\int_{t^n_s}^{t^n_f} \|\boldsymbol{e}^n(t)\|_2 dt \leq \|\boldsymbol{e}^n(t^n_s)\|_2 \bigg(\frac{e^{\Delta T^n(\kappa+\Gamma)} - 1}{\kappa + \Gamma}\bigg) + \alpha^n \int_{t^n_s}^{t^n_f} \|\boldsymbol{r}(\tilde{\boldsymbol{x}}^n_{\ell^2}, t)\|_2 dt.$$

□

Corollary 5.2.1. *For the case of one time window $\Delta T^1 = T$, then under Assumptions A3–A4 the error in the solution computed by the WLS ROM approach with S-reduction trial subspaces is bounded as*

$$\int_0^T \|\boldsymbol{e}^1(t)\|_2 dt \leq \alpha^1 \int_0^T \|\boldsymbol{r}(\tilde{\boldsymbol{x}}^1_{\ell^2}, t)\|_2 dt.$$

Proof. Setting $n = 1$ in (5.3) with the time intervals $t^1_s = 0$, $t^1_f = T$, noting that the initial conditions are known and employing Assumption A4 yields the desired result. □

5.3. Discussion

Theorem 5.2 provides *a priori* bounds on the integrated normed error for WLS employing S-reduction trial subspaces. We make several observations. First, it is observed WLS is subject to recursive error bounds (through the first term on the RHS in the upper bound (5.3)). As the number of time windows grows, so does the recursive growth of error. Second, we observe that when a single window spans the entire domain, the error in the WLS with S-reduction trial subspaces is bounded *a priori* by the residual of the ℓ^2-orthogonal projection of the FOM solution.

6. Numerical experiments

We now analyze the performance of WLS ROMs leveraging S-reduction trial subspaces on two benchmark problems: the Sod shock tube and compressible flow in a cavity. In each experiment, we compare WLS ROMs to the Galerkin and LSPG ROMs. The purpose of the numerical experiments is to assess the impact of minimizing the residual over an arbitrarily sized time window on the solution accuracy. We additionally assess the impact of the time step and time scheme on WLS. In both experiments, the spatial basis is equivalent for each time window, e.g., $\mathbf{V}^n \equiv \mathbf{V}$, $n = 1, \ldots, N_w$. We also note that both experiments are designed to test the *reproductive* ability of the ROMs. We do not consider future state prediction and prediction at new parameter instances as these problems introduce factors that confound the solution accuracy with the solution methodology (e.g., accuracy of the basis).

6.1. Sod shock tube

We first consider reduced-order models of the Sod shock tube problem, which is governed by the compressible Euler equations in one dimension,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial \mathsf{x}} = 0, \quad \boldsymbol{u} = \begin{Bmatrix} \rho \\ \rho u \\ \rho E \end{Bmatrix}, \quad \boldsymbol{F} = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{Bmatrix}, \tag{6.1}$$

where $\boldsymbol{u} : \Omega \times [0, T] \rightarrow \mathbb{R}^3$ comprise the density, x-momentum, and total energy, $\mathsf{x} \in \Omega := [0, 1]$ is the spatial domain, and $T = 1$ the final time. The problem setup is given by the initial conditions

$$\rho = \begin{cases} 1 & \mathsf{x} \leq 0.5 \\ 0.125 & \mathsf{x} > 0.5 \end{cases}, \qquad p = \begin{cases} 1 & \mathsf{x} \leq 0.5 \\ 0.1 & \mathsf{x} > 0.5 \end{cases}, \qquad u = \begin{cases} 0 & \mathsf{x} \leq 0.5 \\ 0 & \mathsf{x} > 0.5 \end{cases},$$

along with reflecting boundary conditions at $\mathsf{x} = 0$ and $\mathsf{x} = 1$.

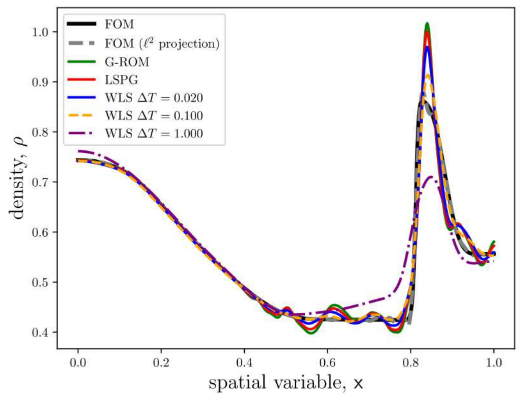
6.1.1. Description of FOM and generation of S-reduction trial subspace

We solve the 1D compressible Euler equations with a finite volume method. We partition the domain into 500 cells of uniform width and employ the Rusanov flux [57] at the cell interfaces. We employ the Crank–Nicolson (CN) scheme, which is a linear multistep method defined by the coefficients $\alpha_0 = 1, \alpha_1 = -1, \beta_0 = \beta_1 = 1/2$, for temporal integration. We evolve the FOM for $t \in [0.0, 1.0]$ at a time-step of $\Delta t = 0.002$. We construct the S-reduction trial subspace by executing Algorithm 3 with inputs $N_{\text{skip}} = 2, \mathbf{x}_{\text{ref}} = \mathbf{0}, K = 46$. The resulting trial subspace corresponds to an energy criterion of 99.99%.

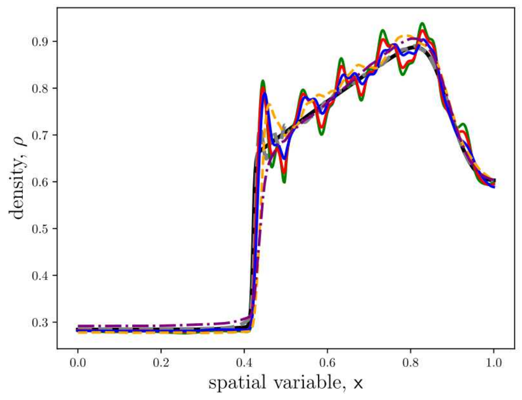
6.1.2. Description of reduced-order models

We consider reduced-order models based on Galerkin projection, LSPG projection, and the WLS approach. No hyper-reduction is considered in this example, i.e., $\mathbf{A}^n = \mathbf{I}$, $n = 1, \ldots, N_w$. Details on the implementation of the different reduced-order models is as follows:

- *Galerkin ROM*: We obtain the Galerkin ROM through Galerkin projection of the FOM and evolve the Galerkin ROM in time with the CN time scheme at a constant time step of $\Delta t = 0.002$.
- *LSPG ROM:* We construct the LSPG ROM on top of the FOM discretization leveraging the CN time scheme as previously described. Unless noted otherwise, we employ a constant time step size of $\Delta t = 0.002$ for the LSPG ROM. We solve the nonlinear least-squares problem arising at each time instance via the Gauss–Newton method, and solve the linear least-squares problems arising at each Gauss–Newton iteration via the normal equations. We deem the Gauss–Newton iteration converged when the gradient norm is less than 10^{-4}. We compute all Jacobians via automatic differentiation [61].
- *WLS ROM:* We consider WLS ROMs solved via the direct and indirect methods with two different solution techniques:
  - *Direct method*: We consider WLS ROMs solved via the direct method for both the same CN discretization employed in the FOM and LSPG, as well as for the second-order explicit Adams Bashforth (AB2) discretization using a constant time step of $\Delta t = 0.0005$. We solve the nonlinear least-squares problem arising over each window with the Gauss–Newton method, and solve the linear least-squares problems arising at each Gauss–Newton iteration via the normal equations. We again compute all Jacobians via automatic differentiation, and deem the Gauss–Newton algorithm converged when the gradient norm is less than 10^{-4} (i.e., we use the parameter $\epsilon = 10^{-4}$ in Algorithm 1). Critically, we note that we assemble the (sparse) Jacobian matrix over a window by computing local (dense) Jacobians. We store the Jacobian matrix over a window in a compressed sparse row format. We employ uniform quadrature weights for evaluating the integral in (3.7).
  - *Indirect method*: We consider two WLS ROMs solved via the indirect method. The first uses the same CN discretization at at time step of $\Delta t = 0.002$, while the second uses the AB2 discretization using a time step of $\Delta t = 0.0005$. We solve the coupled two-point boundary problem via the forward–backward sweep method, and compute the action of the Jacobian transpose on vectors via automatic differentiation. We use parameters $\epsilon = 10^{-6}$, $\psi_1 = 1.1$, and $\psi_2 = 2$ in Algorithm 2.



(a) $t = 0.5$



(b) $t = 1$

Figure 4: Density profiles at various time instances.

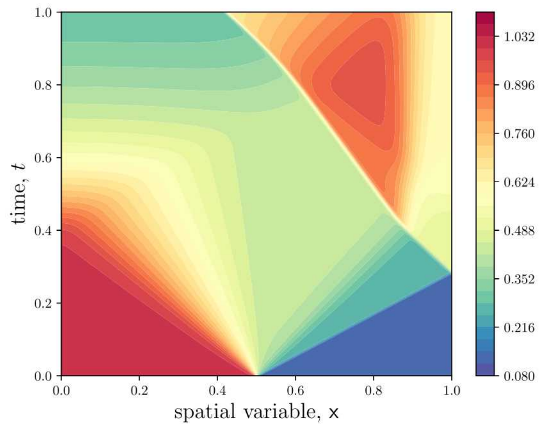
6.1.3. Numerical results

We first assess the impact of the window size on the performance of the WLS ROMs. We consider a set of WLS ROMs that minimize the residual over windows of constant size $\Delta T^n \equiv \Delta T = .002$, 0.004 ,0.008, 0.02, 0.04, 0.10, 0.20, 1.0. We additionally consider the standard Galerkin and LSPG ROMs. We first show results for WLS ROMs using the direct method with CN time discretization; a comparison of different time-marching methods and direct/indirect solution techniques will be provided later in this section. First, Figure 4 presents the density solutions produced by the various ROMs at $t = 0.5$ and 1.0. Figure 5 shows $\mathsf{x} - t$ diagrams for the same density solutions. From Figures 4 and 5, we observe that the LSPG and Galerkin ROMs accurately characterize the system: they correctly track the shock location, expansion waves, etc. We observe both predictions, however, to be highly oscillatory. These oscillations are not physical and can lead to numerical instabilities; e.g., due to negative pressure. We observe the WLS ROMs to produce less oscillatory solutions than both the Galerkin and LSPG ROMs. Critically, we see that the solution becomes less oscillatory as the window size over which the residual is minimized grows. The solution displays no oscillations when the residual is minimized over the entire space–time domain.

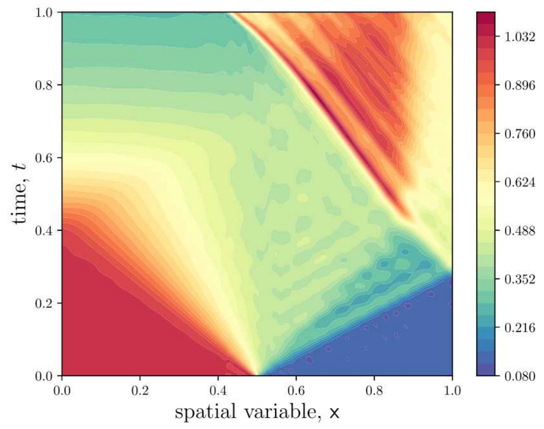
Next, Figure 6 shows space–time state errors and the objective function (3.3) (i.e., the space–time residual norm) over $t \in [0, 1]$ for the various ROMs. Most notably, we observe that increasing the window size over which the residual is minimized does *not* lead to a monotonic decrease in the space–time error as measured in the ℓ^2-norm (we refer to this as the ℓ^2-error). As expected, increasing the window size does lead to a monotonic decrease in the space–time residual, however. We additionally note that, although the space–time ℓ^2-error of the projected FOM solution is significantly lower than that of the various ROM solutions, the space–time residual norm of the projected FOM solution is *higher* than all ROMs. Thus, although the projected FOM solution is more accurate in the ℓ^2-norm, it does not satisfy the governing equations as well the ROM solutions.

We now examine the comparative performance of the direct and indirect solution techniques for the WLS ROMs using various time discretization techniques. Figure 7 shows the same space–time ℓ^2-errors and residual norms as in Figure 6, but this time results are shown for the various WLS ROMs. In both Figures 9a and 9b, we observe that the WLS method is relatively insensitive to the solution technique (direct vs indirect) and underlying discretization scheme, although some minor differences are observed. In particular, ROMs using the second order explicit AB2 scheme with a time step of $\Delta t = 0.0005$ provide similar results to the ROMs using the second-order CN scheme at a time step of $\Delta t = 0.002$. All methods display similar dependence on the window size: the optimal ℓ^2-error occurs when the window size is $\Delta T = 0.1$, and the residual decreases monotonically as the window size grows.

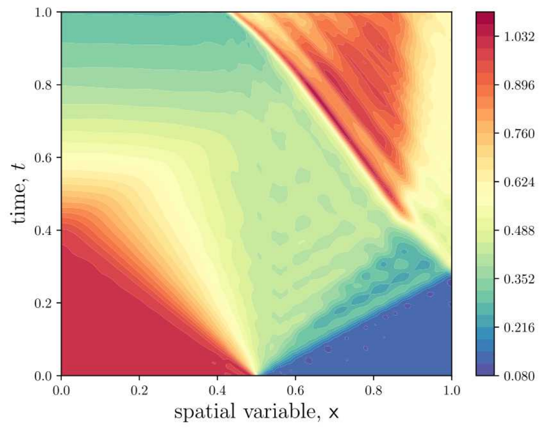
Next, Figure 8 provides the CPU wall-clock times for the various WLS ROMs. We observe that the computational cost of all methods grows as the window size is increased. For indirect methods, this increase in cost is due to the fact that, as the window size grows, more iterations of the FBSM are required for convergence. For direct methods, the increase in cost is due to (1) the cost associated with forming and solving the normal equations at each Gauss–Newton iteration and (2) the increased number of Gauss–



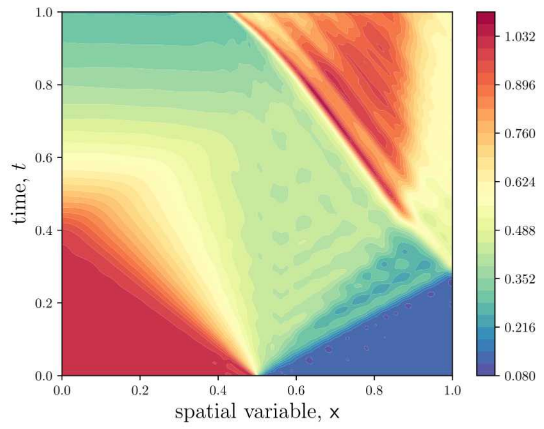
(a) Full-order model



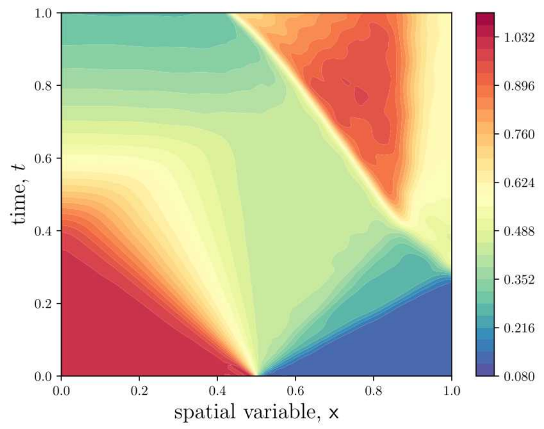
(b) G-ROM



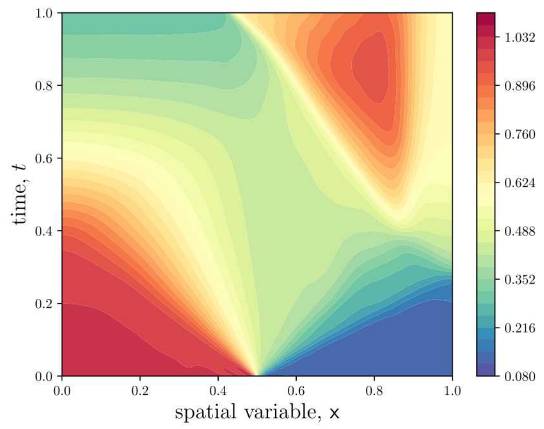
(c) LSPG



(d) WLS: $\Delta T = 0.002$

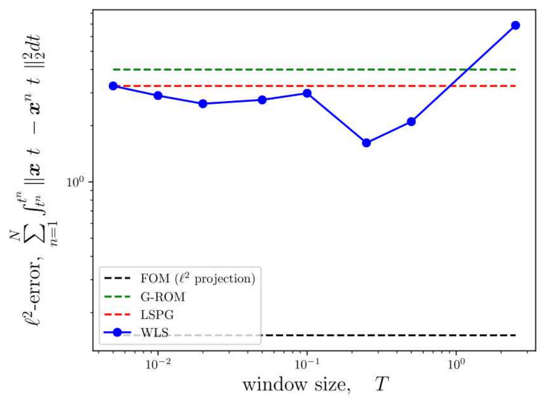


(e) WLS: $\Delta T = 0.1$

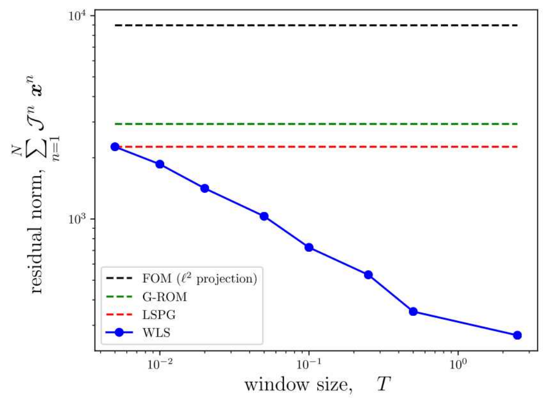


(f) WLS: $\Delta T = 1.0$

Figure 5: $\mathsf{x} - t$ diagrams for the density fields as predicted by the FOM, G-ROM, LSPG-ROM, and WLS ROMs.

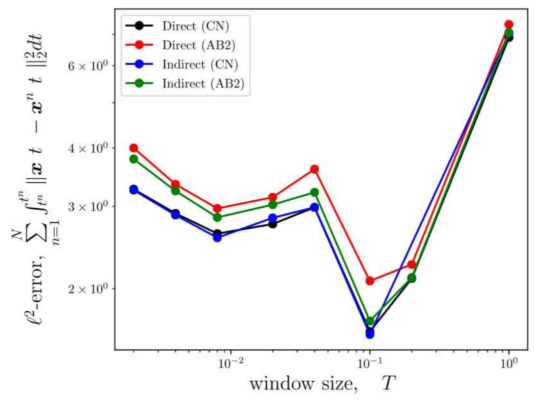


(a) Space–time ℓ^2-error of various ROMs.

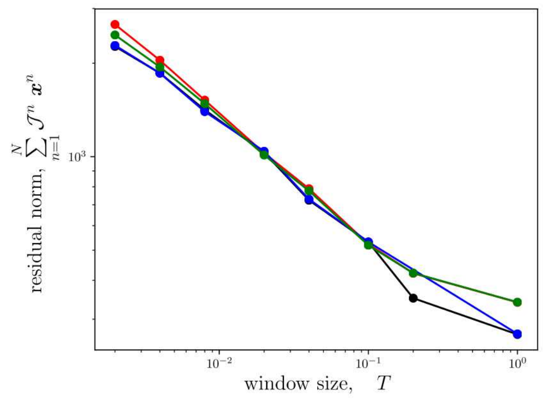


(b) Integrated residual norm of various ROMs.

Figure 6: Integrated performance metrics of the Galerkin, LSPG, and WLS ROMs. Note that the Galerkin and LSPG ROMs do not depend on the window size.



(a) Integrated ℓ^2-error of the WLS ROMs.



(b) Integrated residual norm of the WLS ROMs.

Figure 7: Integrated performance metrics of various WLS ROMs.

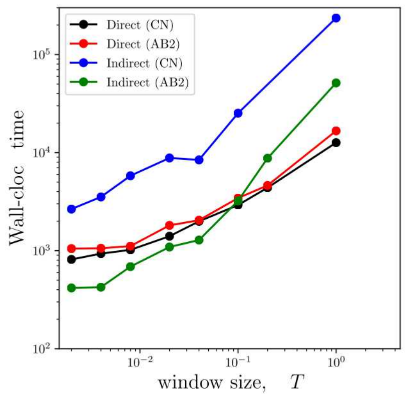


Figure 8: Comparison of wall-clock times of the direct and indirect WLS ROMs as a function of window size.

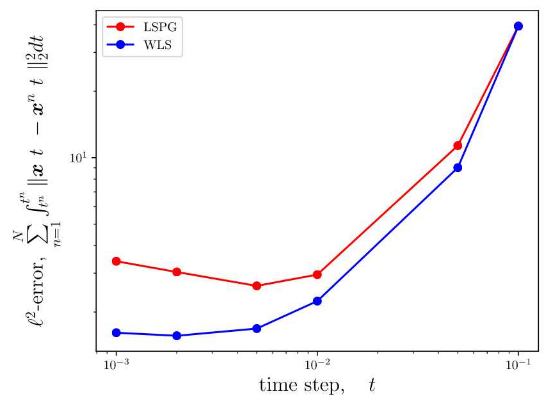
Newton iterations required for convergence. We observe the cost of the FBSM method to increase more rapidly than the direct methods. Encouragingly, WLS ROMs based on the CN discretization minimizing the full space–time residual (comprising 500 time instances) only cost approximately one order of magnitude more than the case where the residual is minimized over a single time step (i.e., LSPG). Also of interest is the fact that the direct method utilizing the AB2 time-discretization scheme costs only slightly more per a given window size than the direct method utilizing the CN time discretization. This is despite the fact that AB2 is evolved at a time step of $\Delta t = 0.0005$, while CN uses $\Delta t = 0.002$; thus AB2 contains 4 times more temporal degrees of freedom than CN. Finally, we emphasize that the results presented here are for standard algorithms (e.g., Gauss–Newton and the FBSM). As mentioned in Remarks 4.2 and 4.3, we expect that the computational cost of both indirect and direct methods can be decreased through the use and/or development of algorithms tailored to the windowed minimization problem.

Finally, we study the impact of the time step on the WLS ROM results. We examine WLS ROMs that use a window size of $\Delta T = 0.1$, with time steps $\Delta t = 0.001, 0.002, 0.005, 0.01, 0.05, 0.1$ (i.e., 100, 50, 20, 10, 2, and 1 time instances per window). We additionally consider LSPG ROMs leveraging the same set of time steps. To assess time step convergence, we compare results to a new full-order model, which is as described in Section 6.1.1 but uses a fine time step of $\Delta t = 10^{-4}$. Figure 9 shows the ℓ^2-error and residual norm of the various ROMs. We observe that the ℓ^2-error and residual norm of the WLS ROMs decrease and converge as the time step decreases. This is in direct contrast to the LSPG ROMs, in where the ℓ^2-error and residual norm display a complex dependence on the time step. This result demonstrates that WLS overcomes the time-step dependence inherit to the LSPG approach. Lastly, Figure 10 shows the relative wall-clock times of the WLS ROMs with respect to the LSPG ROMs. It is seen that for all time steps considered, the WLS ROMs are less than 6x the cost of LSPG; this is despite the window sizes comprising up to 100 time instances.

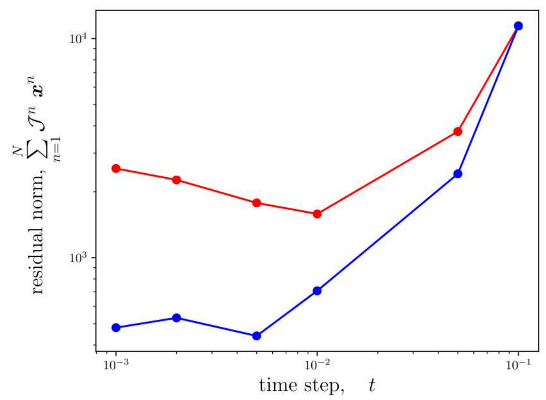
6.1.4. Summary of numerical results for the Sod shock tube

The key observations from the results of the first numerical example are:

1. Increasing the window size over which the residual is minimized led to more physically relevant solutions. Specifically, we observed that as the window size over which the residual was minimized grew, WLS led to less oscillatory solutions.
2. Increasing the window size over which the residual is minimized does not necessary lead to a lower space–time error as measured by the ℓ^2-norm. We observed that minimizing the residual over an intermediary window size led to the lowest space–time error in the ℓ^2-norm.
3. WLS displays time-step convergence: both the ℓ^2-error and residual norm decreased and displayed time-step convergence as the time step decreased. This is in contrast to LSPG.
4. For all examples considered, in where the windows comprised up to 2000 time instances, WLS with the direct method is between 1x and 10x the cost of the LSPG. The direct method was observed to be slightly more efficient and robust than the indirect method.



(a) Space–time ℓ^2-error of the WLS ROM.



(b) Space–time residual norm of the WLS ROM.

Figure 9: Performance metrics of the Galerkin, LSPG, and WLS ROMs.

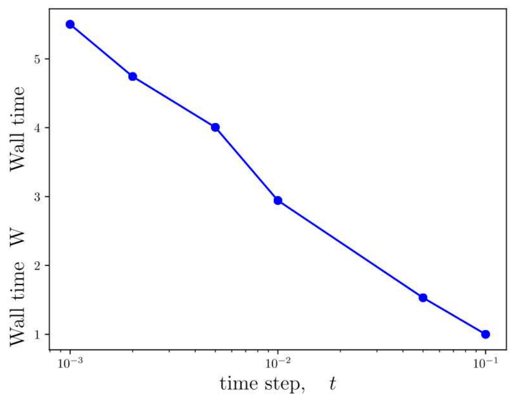


Figure 10: Comparison of wall-clock times of the WLS ROMs to LSPG ROMs as a function of the time step. WLS ROMs have a fixed window size of $\Delta T = 0.1$.

6.2. Cavity flow

The next numerical example considers collocated ROMs[4] of a viscous, compressible flow in a two-dimensional cavity. The flow is described by the two-dimensional compressible Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \big(\mathbf{F}(\mathbf{u}) - \mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u})\big) = \mathbf{0}, \tag{6.2}$$

where $\mathbf{u} : [0,T] \times \Omega \to \mathbb{R}^4$ comprise the density, x_1 and x_2 momentum, and total energy. The terms $\mathbf{F}$ and $\mathbf{F}_v$ are the inviscid and viscous fluxes, respectively. For a two-dimensional flow, the state vector and inviscid fluxes are

$$\mathbf{u} = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{Bmatrix}, \qquad \mathbf{F}_1 = \begin{Bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ u_1(E+p) \end{Bmatrix}, \qquad \mathbf{F}_2 = \begin{Bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ u_2(E+p) \end{Bmatrix}.$$

The viscous fluxes are given by

$$\mathbf{F}_{v_1} = \begin{Bmatrix} 0 \\ \tau_{11} \\ \tau_{12} \\ u_j \tau_{j1} + c_p \frac{\mu}{\mathrm{Pr}} \frac{\partial T}{\partial x_1} \end{Bmatrix}, \qquad \mathbf{F}_{v_2} = \begin{Bmatrix} 0 \\ \tau_{21} \\ \tau_{22} \\ u_j \tau_{j2} + c_p \frac{\mu}{\mathrm{Pr}} \frac{\partial T}{\partial x_2} \end{Bmatrix},$$

where $\mu \in \mathbb{R}_+$ is the dynamic viscosity, $\mathrm{Pr} = 0.72$ is the Prandtl number, $T : \Omega \to \mathbb{R}_+$ the temperature, and $c_p \in \mathbb{R}_+$ the heat capacity ratio. We assume a Newtonian fluid, which leads to a viscous stress tensor of the form

$$\tau_{ij} = 2\mu S_{ij},$$

where

$$S_{ij} = \frac{1}{2}\Big(\frac{\partial u_i}{\partial \mathsf{x}_j} + \frac{\partial u_j}{\partial \mathsf{x}_i}\Big) - \frac{1}{3}\frac{\partial u_k}{\partial \mathsf{x}_i}\delta_{ij}.$$

We close the Navier–Stokes equations with a constitutive relationship for a calorically perfect gas,

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho u_1^2 - \frac{1}{2}\rho u_2^2),$$

where $\gamma = 1.4$ is the heat-capacity ratio.

Figure 11 depicts the domain Ω and flow conditions. The Reynolds number is defined as $\mathrm{Re} = \rho_\infty \|\mathbf{v}_\infty\|_2 L/\mu$ with a characteristic length set at $L = 1$ and $\mathbf{v} = [u_1 \;\; u_2]^T$, the speed of sound is defined by $a_\infty = \sqrt{\gamma p_\infty/\rho_\infty}$, and ∞ subscripts refer to free-stream conditions. We employ free-stream boundary conditions at the inlet, outlet, and top wall of the cavity. We enforce no-slip boundary conditions on the bottom wall of the cavity.

6.2.1. Description of FOM and generation of S-reduction trial subspace

The full-order model comprises a discontinuous-Galerkin discretization. We obtain the discretization by partitioning the domain into 100 elements in the flow direction and 40 elements in the wall-normal direction. The discretization represents the solution to third order over each element using tensor product polynomials of order $p = 2$, resulting in 36000 unknowns for each conserved variable. Spatial discretization via the discontinuous Galerkin method yields a dynamical system of the form

$$\frac{d\boldsymbol{x}}{dt} = \mathbf{M}_{\mathrm{DG}}^{-1}\boldsymbol{f}_{\mathrm{DG}}(\boldsymbol{x}),$$

where $\mathbf{M}_{\mathrm{DG}} \in \mathbb{R}^{N \times N}$ is the (block diagonal) DG mass matrix and $\boldsymbol{f}_{\mathrm{DG}} : \mathbb{R}^N \to \mathbb{R}^N$ is the DG velocity operator containing surface and volume integrals. By the definition of the FOM (2.1), the dynamical system velocity is $\boldsymbol{f} = \mathbf{M}_{\mathrm{DG}}^{-1}\boldsymbol{f}_{\mathrm{DG}}$. Figure 12 shows the computational mesh. The DG method uses the Rusanov flux at the cell interfaces [57] and uses the first form of Bassi and Rebay [4] for the viscous fluxes.

[4]Collocation is a form of hyper-reduction which requires sampling the full-order model at only select grid points.

Basis #	Trial Basis Dimension (K)	Energy Criterion	Sample Points (n_S)
1	136	95.0%	1603
2	193	97.0%	1603

Table 1: Summary of the various basis sizes employed in the cavity flow example.

Time integration is performed via a third-order strong stability preserving Runge-Kutta method [32] with a time step of $\Delta t = 0.001$.

The reduced-order models leverage POD to construct the S-reduction trial basis and use q-sampling [28] based on snapshots of the FOM velocity to select the sampling points (and as a result the weighting matrix $\mathbf{A}$); we use a constant weighting matrix across all windows, e.g., $\mathbf{A}^n \equiv \mathbf{A}$, $n = 1, \ldots, N_w$. The process used to construct the initial conditions, trial subspaces, and weighting matrices for the ROMs is as follows:

1. Initialize the FOM with uniform free-stream conditions.
2. Evolve the FOM for $t \in [0, 400]$.
3. Reset the time coordinate, $0 \leftarrow t$, and execute Algorithm 3 with $\mathbf{x}_{\text{ref}} = \mathbf{0}, N_{\text{skip}} = 100$ and $K = \{136, 193\}$ over $t \in [0, 100]$ to construct two trial subspaces comprising $\approx 95\%$ and $\approx 97\%$ of the snapshot energy, respectively.
4. Execute Algorithm 4 with $N_{\text{skip}} = 100$, $n_s = 129$ to obtain the sampling point matrix of dimension $\mathbf{W} \in \{0, 1\}^{1603 \times N}$.

Figure 13 shows the resulting sample mesh used in the ROM simulations. To depict the nature of the flow, Figure 14 shows snapshots of the vorticity field generated by the FOM for several time instances used in training.

6.2.2. *Description of reduced-order models*

We consider collocated ROMs based on the Galerkin, least-squares Petrov–Galerkin, and WLS approaches. Details on the implementation of the methods is as follows:

- *Galerkin ROM with collocation*: We consider a Galerkin ROM with collocation and evolve the ROM in time with the CN time scheme at a time step of $\Delta t = 0.1$.

- *LSPG ROM with collocation*: We consider a collocated LSPG ROM, which is built on top of the FOM discretization using the CN scheme for temporal discretization. We employ a time step of $\Delta t = 0.1$. The implementation is the same as previously described.

- *WLS ROMs with collocation*: We consider WLS ROMs solved via the direct method. The ROMs use the CN time discretization with a time step of $\Delta t = 0.1$. The implementation is the same as previously described.

6.2.3. *Numerical results*

We first assess the performance of WLS ROMs with varying window sizes. To this end, we consider WLS ROMs with uniform window sizes of $\Delta T^n \equiv \Delta T = 0.2, 0.5, 1.0$, and 2.0, along with the Galerkin and LSPG ROMs. We first consider results for basis #1 as described in Table 1. For all ROMs, we evolve the solution for $t \in [0, 100]$. This comprises the same time interval used to construct the trial subspace. First, Figure 15a depicts the evolution of the pressure at the bottom wall in the midpoint of the computational domain, while Figure 15b depicts the evolution of the normalized ℓ^2-error of the various reduced-order models. Both the collocated Galerkin and LSPG ROMs blow up/fail to converge within the first several time units. The WLS ROM minimizing the residual over a window of size $\Delta T = 0.2$ also fails to converge. The WLS ROMs that minimize the residual over window sizes of $\Delta T \geq 0.5$ are seen to all be stable and accurate; the pressure response is well characterized and the normalized state errors are less than 10%. The most notable discrepancy between the WLS ROM and FOM solutions is a phase difference.

Figure 16 shows the space–time error and objective function of the stable WLS ROMs. We observe that growing the window size over which the residual is minimized leads to a lower space–time residual, but not necessarily a lower ℓ^2-error. This result is consistent with the previous numerical example. Next, Figure 21 shows the wall-clock times of the WLS ROMs for $t \in [0, 10]$ as compared to the LSPG ROM[5].

[5]We note that LSPG failed to converge at $t \approx 16.0$, so we focus on the first ten time units.

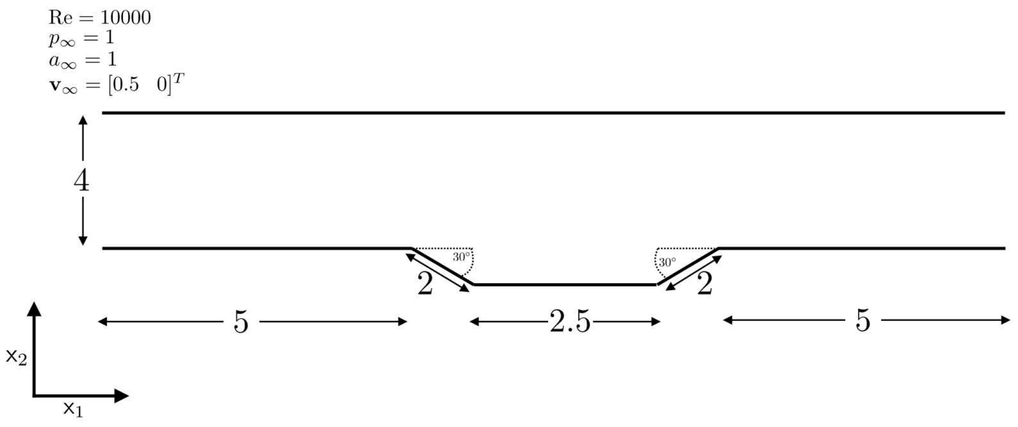


Figure 11: Figure depicting geometry and flow conditions of the cavity flow problem.

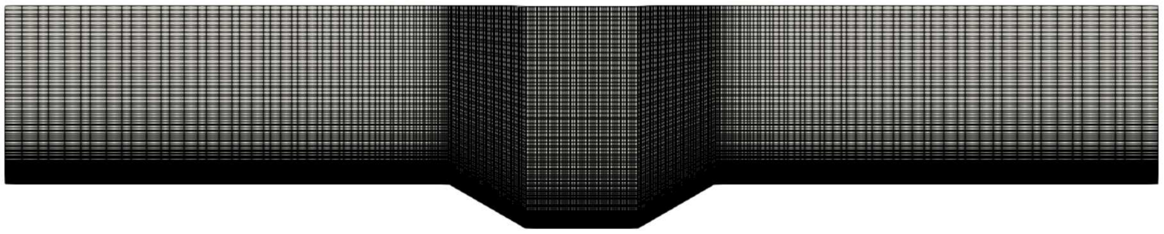


Figure 12: Computational mesh employed in cavity flow simulations.

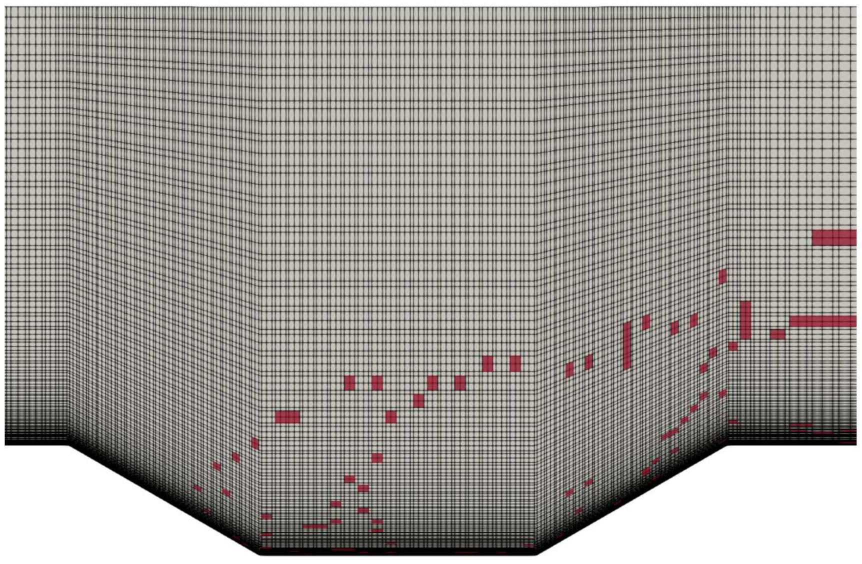
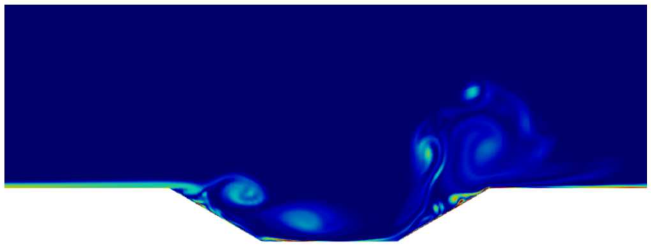
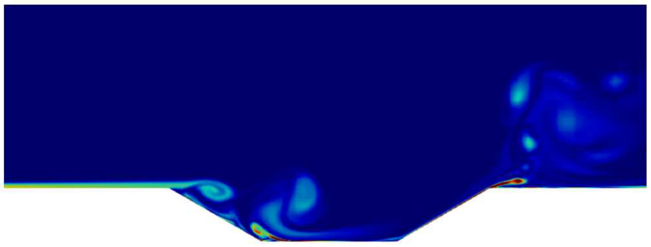
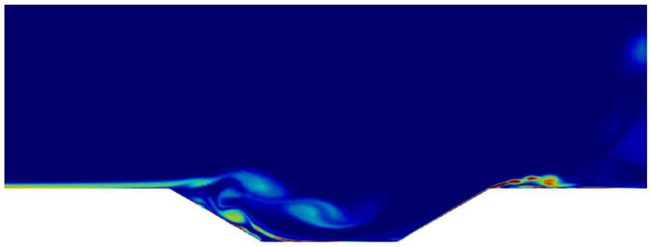
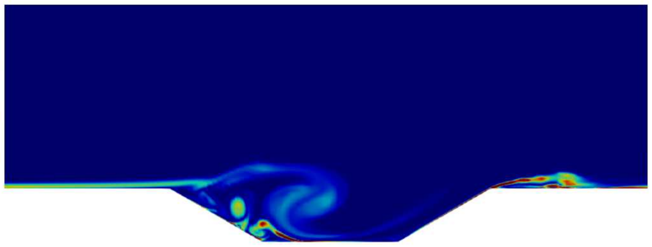


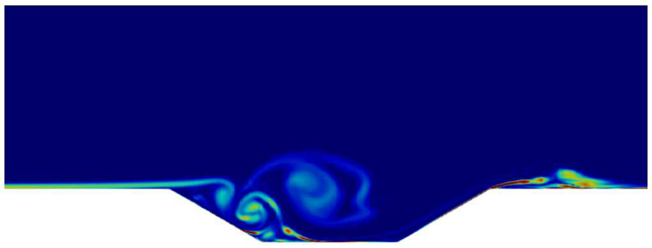
Figure 13: Close up of computational mesh with highlighted collocation cells.


(a) $t = 0.0$


(b) $t = 4.0$


(c) $t = 8.0$


(d) $t = 12.0$


(e) $t = 16.0$

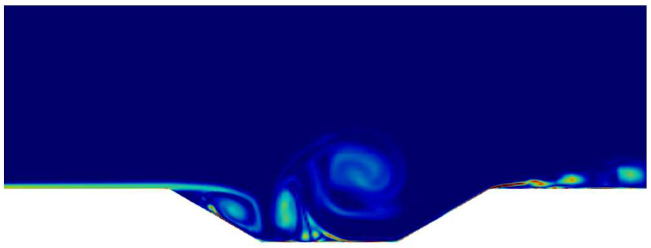
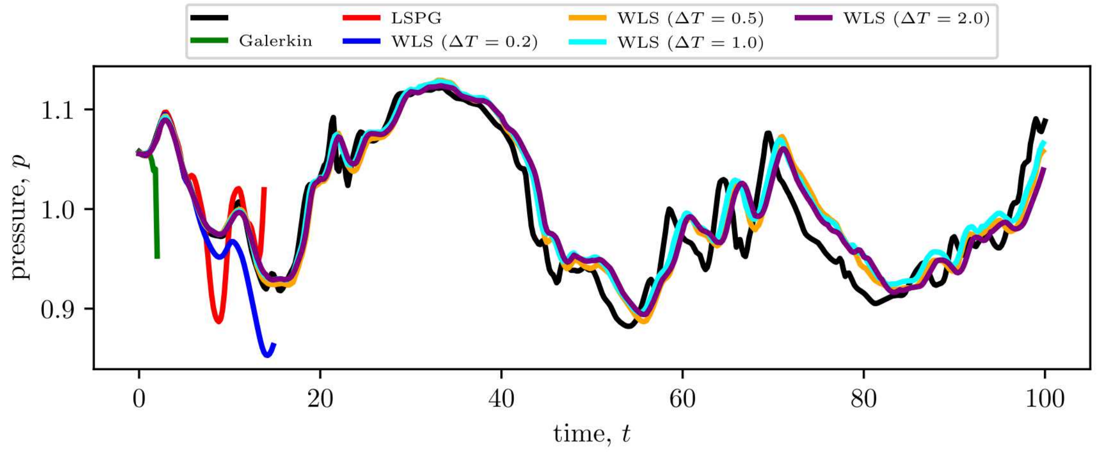
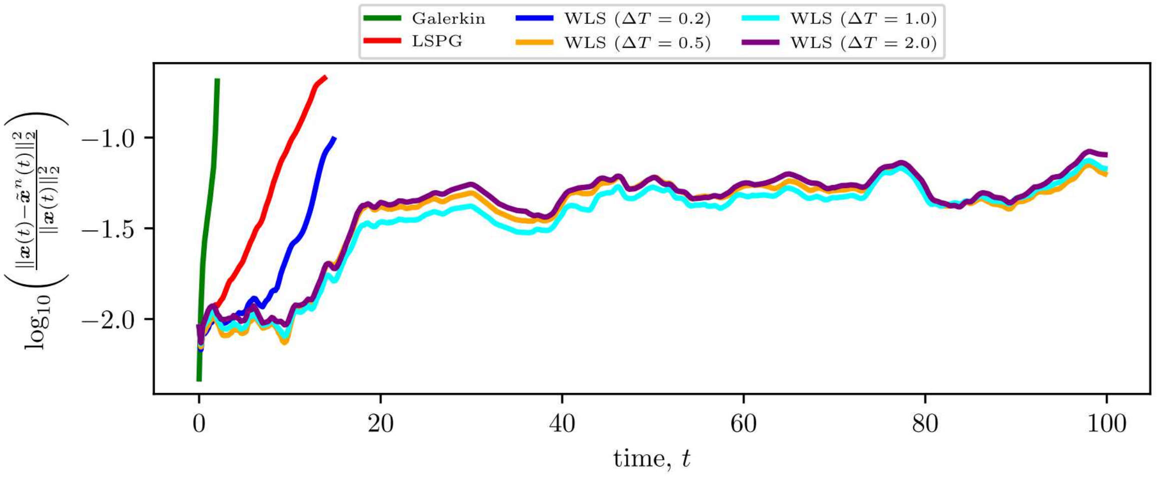

(f) $t = 20.0$

Figure 14: Vorticity snapshots from the FOM simulation at various time instances.

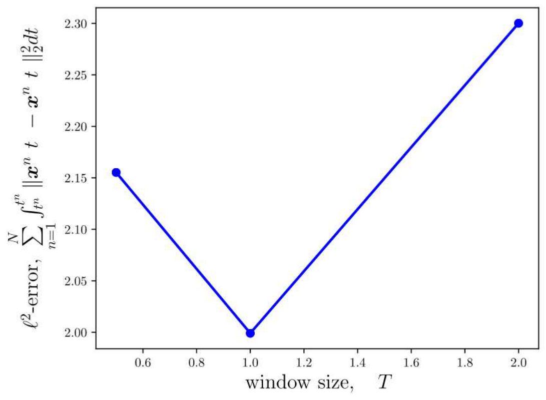


(a) Pressure



(b) Normalized ℓ^2-error

Figure 15: Comparison of the pressure profiles obtained at the midpoint of the bottom wall (top) and normalized ℓ^2-errors (bottom) of various collocated ROMs to the full-order model solution.


(a) Space–time ℓ^2-error

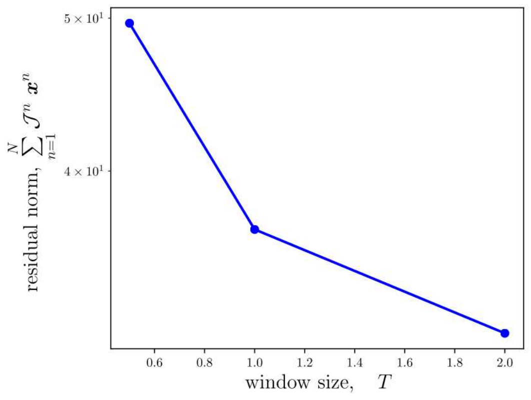

(b) Objective function

Figure 16: Space–time error (left) and objective function (right) as a function of window size.

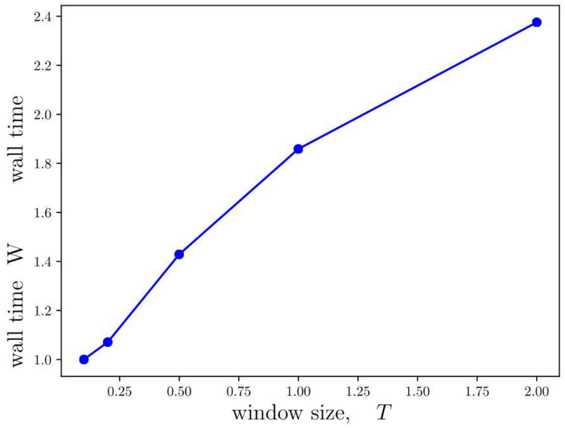


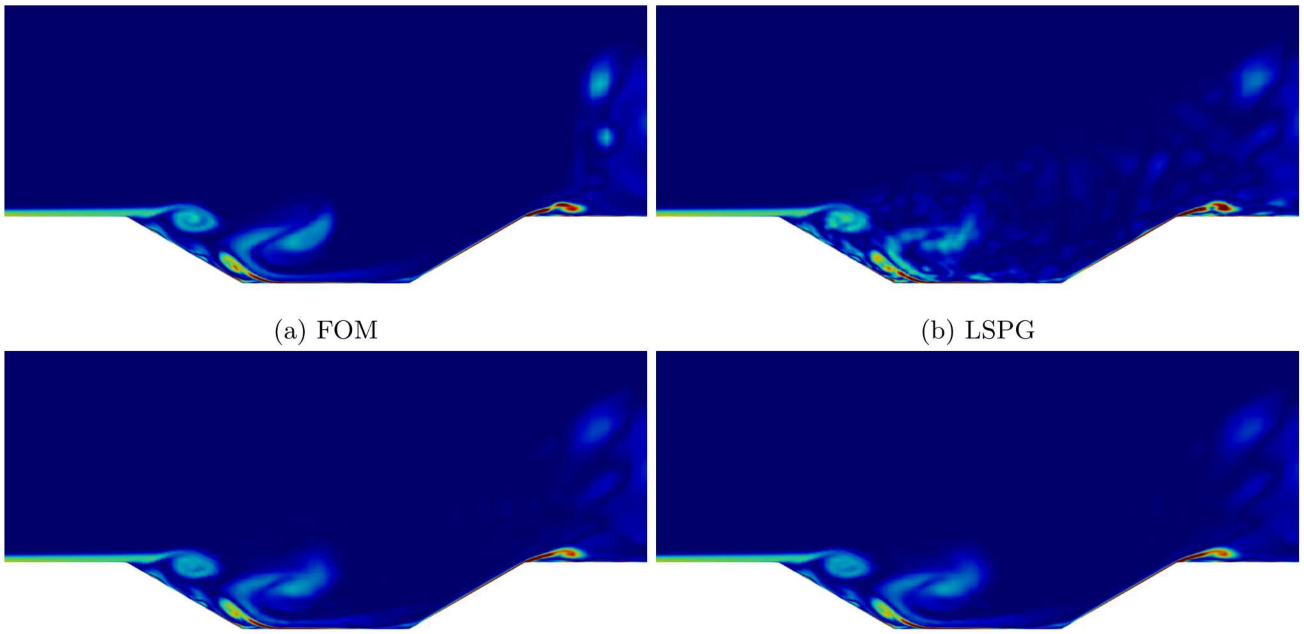
Figure 17: Wall-clock times of WLS ROMs with respect to the LSPG ROM.

As expected, increasing ΔT again leads to an increase in computational cost; minimizing the residual over a window comprising 20 time instances yields a 2.5x increase in cost over LSPG.

Figure 18 shows vorticity fields for the FOM, LSPG ROM, and WLS ROMs with $\Delta T = 0.5, 1.0$ for the time instance $t = 5.0$. LSPG is observed to exhibit artificial oscillations; the Gauss-Newton method fails to converge at $t \approx 16.0$. The WLS ROMs at $\Delta T = 0.5, 1.0$ are able to capture the important features of the flow, including the points of flow separation at the start and end of the ramp, and remain stable for the entire time interval.

Next, we assess the performance of the various ROMs for basis #2 as described in Table 1, which comprises a richer spatial basis. Figure 19 shows the same pressure and error profiles as Figure 15, but for the enriched basis. The LSPG and Galerkin ROMs blow up faster as compared to Figure 15; LSPG fails to converge around $t \approx 8$ (opposed to $t \approx 16$), while Galerkin blows up almost immediately. The WLS ROMs again yield improved performance: WLS ROMs minimizing the residual over window sizes of $\Delta T \geq 0.5$ are seen to all be stable and accurate; the pressure response is well characterized and the normalized state errors are less than 5%. The WLS ROMs employing basis #2 yield more accurate results than WLS ROMs employing basis #1.

Finally, Figure 21 shows the wall-clock times for $t \in [0, 4]$ of the WLS ROMs as compared to the LSPG ROMs for basis #2. Increasing the window size again leads to an increase in computational cost. Minimizing the residual over a window comprising 20 time instances yields a 3x increase in cost over LSPG.


(a) FOM (b) LSPG

(c) WLS, $\Delta T = 0.5$ (d) WLS, $\Delta T = 1.0$

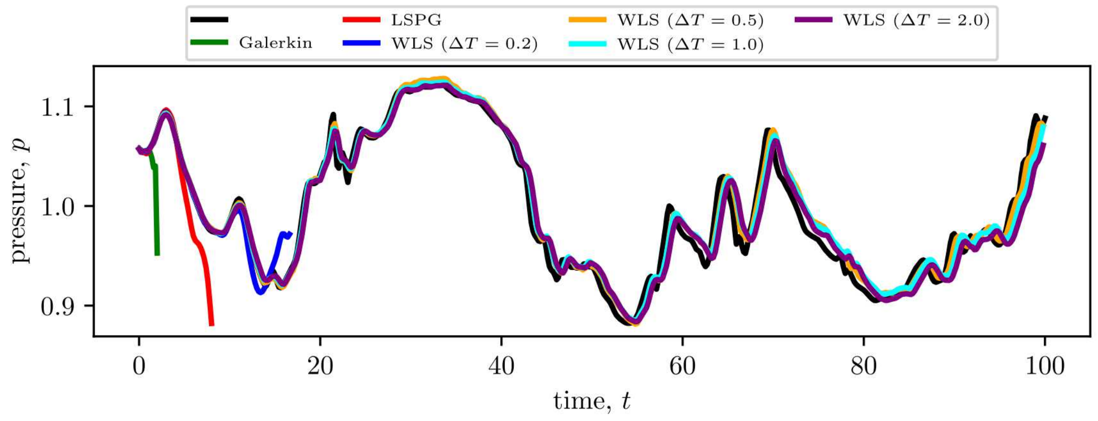
Figure 18: Vorticity snapshots from the FOM and ROM simulations at $t = 5.0$.

7. Conclusions

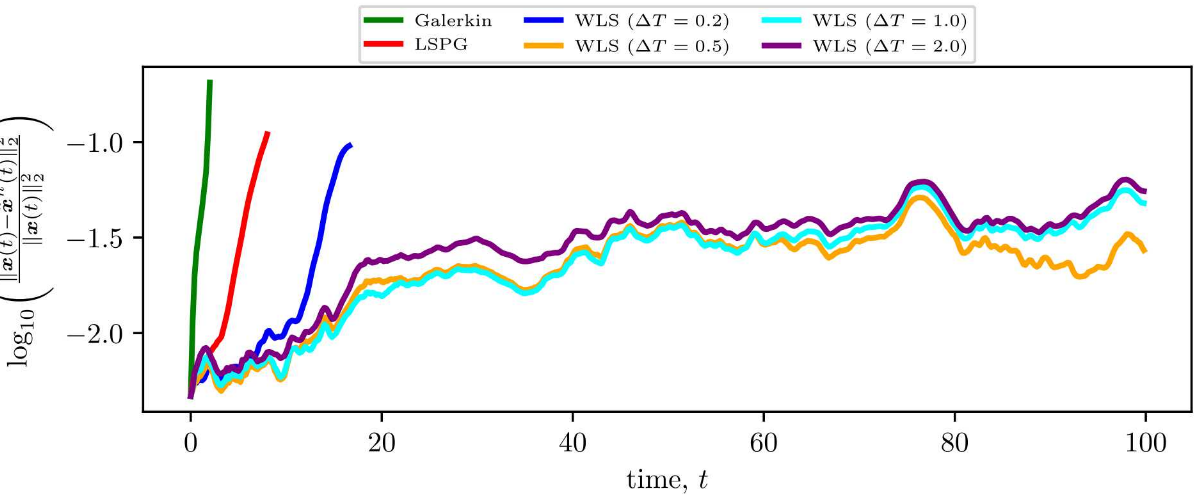
This paper proposed the windowed least-squares (WLS) approach for model reduction of dynamical systems. The approach sequentially minimizes the time-continuous full-order-model residual within a low-dimensional space–time trial subspace over time windows. The approach was formulated for two types of trial subspaces: one that reduces only the spatial dimension of the full-order model, and one that reduces both the spatial and temporal dimensions of the full-order model. For each type of trial subspace, we outlined two different solution techniques: direct (i.e., discretize then optimize) and indirect (i.e., optimize then discretize). We showed that particular instances of the approach recover Galerkin, least-squares Petrov–Galerkin (LSPG), and space–time LSPG projection. The case of S-reduction trial subspaces is of particular interest in the WLS context: it was shown that indirect methods comprise solving a coupled two-point Hamiltonian boundary value problem. The forward system, which is forced by an auxiliary costate, evolves the (spatial) generalized coordinates of the ROM in time. The backward system, which is forced by the time-continuous FOM residual evaluated about the ROM state, governs the dynamics of the costate.

Numerical experiments of the compressible Euler and Navier–Stokes equations demonstrated the utility in the proposed approach. The first numerical experiment, in where the Sod shock tube was examined, demonstrated that WLS ROMs minimizing the residual over larger time windows yielded solutions with lower space–time residuals. Increasing the window size over which the residual was minimized, however, did not necessarily decrease the solution error in the ℓ^2-norm; we observed this to occur over an intermediary window size. We additionally observed that the WLS approach overcomes the time-discretization sensitivity that LSPG is subject to. The second numerical experiment, which examined collocated ROMs of a compressible cavity flow, demonstrated the utility of the WLS formulation on a more complex flow. In this experiment, WLS ROMs yielded predictions with relative errors less than $5 - 10\%$, while the Galerkin and LSPG ROMs failed to converge/blew up. Increasing the window size over which the residual was minimized again led to a lower space–time residual, but not necessarily a lower error in the ℓ^2-norm.

The principal challenge encountered in the WLS formulation is the computational cost: increasing the window size over which the residual is minimized leads to a higher computational cost. In the context of the direct solution approach, this increased cost is due to the increased expense of forming and solving the least-squares problem associated with larger window sizes. In the context of indirect methods, this increased cost is a result of the increased number of iterations of the forward backward sweep method. While numerical experiments demonstrated that the increase in computational cost is mild, future work

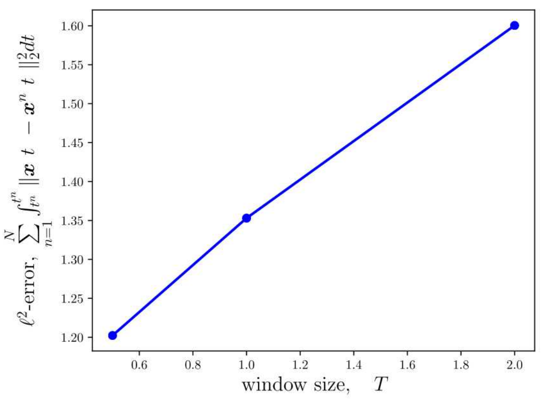


(a) Pressure

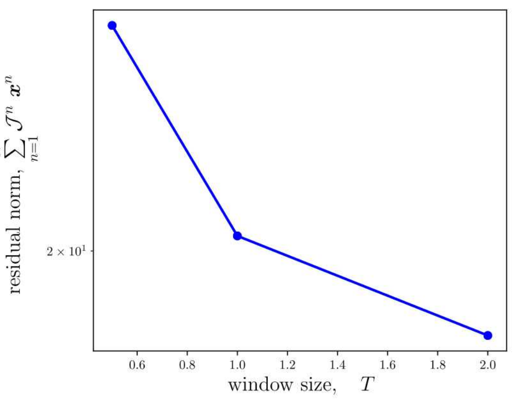


(b) Normalized ℓ^2-error

Figure 19: Comparison of the pressure profiles obtained at the midpoint of the bottom wall (top) and normalized ℓ^2-errors (bottom) of various collocated ROMs to the full-order model solution.



(a) Integrated ℓ^2-error



(b) Objective function

Figure 20: Integrated error (left) and objective function (right) as a function of window size.

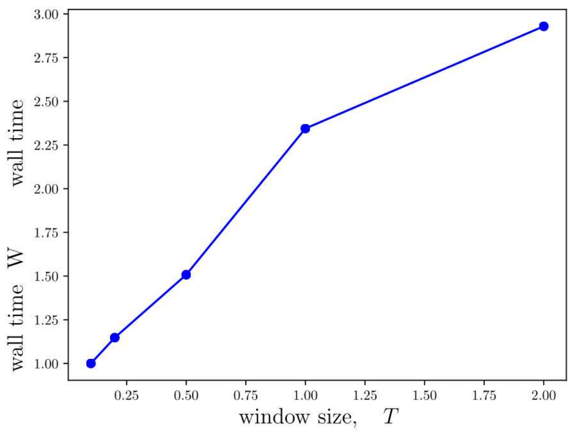


Figure 21: Wall-clock times of WLS ROMs with respect to the LSPG ROM.

will target the development of new solution techniques tailored for the WLS approach.

8. Acknowledgments

The authors thank Yukiko Shimizu and Patrick Blonigan for numerous conversations from which this work benefited. E. Parish acknowledges an appointment to the Sandia National Laboratories' John von Neumann Postdoctoral Research Fellowship in Computational Science. This work was partially sponsored by Sandia's Advanced Simulation and Computing (ASC) Verification and Validation (V&V) Project/Task #103723/05.30.02. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Appendix A Proper orthogonal decomposition

Algorithm 3 presents the algorithm for computing the trial basis via proper orthogonal decomposition.

Algorithm 3: Algorithm for generating POD Basis.

```
Input: Number of time-steps between snapshots $N_{\text{skip}}$; intercept $\mathbf{x}_{\text{ref}}$; basis dimension $K$ ;
Output: POD Basis $\mathbf{V} \in \mathbb{V}_K(\mathbb{R}^N)$ ;
Steps:
  1. Solve FOM OΔE and collect solutions into snapshot matrix
       $$\mathbf{S}(N_{\text{skip}}) := \begin{bmatrix} \mathbf{x}^0 - \mathbf{x}_{\text{ref}}^n & \mathbf{x}^{N_{\text{skip}}} - \mathbf{x}_{\text{ref}}^n & \cdots & \mathbf{x}^{\text{floor}(N_t/N_{\text{skip}})N_{\text{skip}}} - \mathbf{x}_{\text{ref}}^n \end{bmatrix}$$
  2. Compute the thin singular value decomposition,
       $$\mathbf{S}(N_{\text{skip}}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^T,$$
     where $\mathbf{U} \equiv \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_{\text{floor}(N_t/N_{\text{skip}})} \end{bmatrix}$
  3. Truncate left singular vectors and to form the basis, $\mathbf{V} \equiv \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_K \end{bmatrix}$
```

Appendix B Selection of sampling points

To construct the sampling point matrix used for hyper-reduction in the second numerical experiment, we employ q-sampling [28] and the sample mesh concept [21]. Algorithm 4 outlines the steps used in the second numerical experiment to compute the sampling points.

Algorithm 4: Algorithm for generating the sampling matrix through q-sampling.

```
Input: Number of time-steps between snapshots $N_{\text{skip}}$, number of primal sampling points, $n_s$ ;
Output: Weighting matrix $\mathbf{A} \equiv [\mathbf{W}]^T\mathbf{W} \in \{0,1\}^{N\times N}$ ;
Steps:
    1. Solve FOM OΔE and collect velocity snapshots
       $$\mathbf{F}(N_{\text{skip}}) := \begin{bmatrix}\boldsymbol{f}(\mathbf{x}^0) & \boldsymbol{f}(\mathbf{x}^{N_{\text{skip}}}) & \cdots & \boldsymbol{f}(\mathbf{x}^{\text{floor}(N_t/N_{\text{skip}})N_{\text{skip}}})\end{bmatrix}$$
    2. Compute the thin singular value decomposition,
       $$\mathbf{F}(N_{\text{skip}}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^T,$$
       where $\mathbf{U} \equiv \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_{\text{floor}(N_t/N_{\text{skip}})}\end{bmatrix}$
    3. Compute the QR factorization of $\mathbf{U}^T$ with column pivoting,
       $$\mathbf{U}^T\mathbf{P}^* = \mathbf{Q}\mathbf{R}$$
       with $\mathbf{P}^* \equiv \begin{bmatrix}\mathbf{p}_1 & \cdots & \mathbf{p}_{\text{floor}(N_t/N_{\text{skip}})}\end{bmatrix}$, $\mathbf{p}_i \in \{0,1\}^N$.
    4. Select the first $n_s$ columns of $\mathbf{P}^*$ to form the sampling point matrix $[\mathbf{W}]^T \in \{0,1\}^{N\times n_s}$.
    5. Augment the sampling point matrix, $\mathbf{W}$, with additional columns such that all unknowns are
       computed at the mesh cells selected by Step 3. For the second numerical experiment, these
       additional unknowns correspond to each conserved variable and quadrature point in the selected
       cells.
```

Appendix C Derivation of the Euler–Lagrange equations

This section details the derivation of the Euler–Lagrange equations. To this end, we consider the generic functional of the form,

$$\mathcal{J} : (\boldsymbol{y}, \boldsymbol{v}) \mapsto \int_a^b \mathcal{I}(\boldsymbol{y}(t), \boldsymbol{v}(t), t)dt, \tag{C.1}$$

where $\mathcal{I} : \mathbb{R}^M \times \mathbb{R}^M \times [a,b] \to \mathbb{R}_+$ (for arbitrary M) with $\mathcal{I} : (\mathbf{y}, \mathbf{v}, \tau) \mapsto \mathcal{I}(\mathbf{y}, \mathbf{v}, \tau)$. We now introduce the function $\boldsymbol{z} : [a,b] \to \mathbb{R}^M$ with $\boldsymbol{z} : \tau \mapsto \boldsymbol{z}(\tau)$ along with $\dot{\boldsymbol{z}} \equiv d\boldsymbol{z}/d\tau$. We define this function to be a stationary point of (C.1) (with $\boldsymbol{z}$ being the first argument and $\dot{\boldsymbol{z}}$ being the second argument) subject to the boundary condition $\boldsymbol{z}(a) = \boldsymbol{z}_a$. We additionally introduce an arbitrary function $\boldsymbol{\eta} : [a,b] \to \mathbb{R}^M$ with the boundary condition $\boldsymbol{\eta}(a) = \mathbf{0}$ and define a variation from the stationary point by

$$\begin{aligned}\overline{\boldsymbol{z}} &: (\tau, \delta) \mapsto \boldsymbol{z}(\tau) + \delta\boldsymbol{\eta}(\tau),\\ &: [a,b] \times \mathbb{R} \to \mathbb{R}^M.\end{aligned}$$

We note that $\overline{\boldsymbol{z}}$ satisfies the same boundary condition $\boldsymbol{z}$ since $\boldsymbol{\eta}(a) = \mathbf{0}$. We define a new function that is equivalent to the function (C.1) evaluated at $\overline{\boldsymbol{z}}$ in the first argument and $\dot{\overline{\boldsymbol{z}}} \equiv d\overline{\boldsymbol{z}}/d\tau$ in the second argument,

$$\mathcal{J}_\delta : \delta \mapsto \int_a^b \mathcal{I}(\overline{\boldsymbol{z}}(t,\delta), \dot{\overline{\boldsymbol{z}}}(t,\delta), t)dt.$$

The objective is to now find $\overline{\boldsymbol{z}}$ that makes $\mathcal{J}_\delta$ stationary. This can be done by differentiating with respect to δ and setting the result to zero, i.e.,

$$\frac{d}{d\delta}(\mathcal{J}_\delta) = 0. \tag{C.2}$$

Using the chain rule,

$$\frac{d}{d\delta}(\mathcal{J}_\delta)(\epsilon) = \int_a^b \left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\frac{\partial \overline{\boldsymbol{z}}}{\partial \delta}(t,\epsilon) + \frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\frac{\partial \dot{\overline{\boldsymbol{z}}}}{\partial \delta}(t,\epsilon)\right] dt.$$

Noting that

$$\frac{\partial \overline{\boldsymbol{z}}}{\partial \delta}(t,\cdot) = \boldsymbol{\eta}(t), \qquad \frac{\partial \dot{\overline{\boldsymbol{z}}}}{\partial \delta}(t,\cdot) = \dot{\boldsymbol{\eta}}(t),$$

where $\dot{\boldsymbol{\eta}} \equiv d\boldsymbol{\eta}/d\tau$, we have

$$\frac{d}{d\delta}(\mathcal{J}_\delta)(\epsilon) = \int_a^b \left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\boldsymbol{\eta}(t) + \frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\dot{\boldsymbol{\eta}}(t)\right] dt.$$

We integrate the second term by parts,

$$\frac{d}{d\delta}(\mathcal{J}_\delta)(\epsilon) = \int_a^b \left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\boldsymbol{\eta}(t) - \frac{d}{dt}\left(\frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\right)\boldsymbol{\eta}(t)\right] dt + \frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(b,\epsilon), \dot{\overline{\boldsymbol{z}}}(b,\epsilon), b\big)\boldsymbol{\eta}(b), \quad \text{(C.3)}$$

where we have used $\boldsymbol{\eta}(a) = \mathbf{0}$. Substituting Eq. (C.3) in the stationarity condition (C.2) yields

$$\int_a^b \left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\boldsymbol{\eta}(t) - \frac{d}{dt}\left(\frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(t,\epsilon), \dot{\overline{\boldsymbol{z}}}(t,\epsilon), t\big)\right)\boldsymbol{\eta}(t)\right] dt + \frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\overline{\boldsymbol{z}}(b,\epsilon), \dot{\overline{\boldsymbol{z}}}(b,\epsilon), b\big)\boldsymbol{\eta}(b) = 0. \quad \text{(C.4)}$$

By construction, $\boldsymbol{z}(\cdot) \equiv \overline{\boldsymbol{z}}(\cdot;0)$ and $\dot{\boldsymbol{z}}(\cdot) \equiv \dot{\overline{\boldsymbol{z}}}(\cdot;0)$, comprise a stationary point; thus, setting $\epsilon = 0$ in Eq. (C.4) yields

$$\int_a^b \left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), t\big)\boldsymbol{\eta}(t) - \frac{d}{dt}\left(\frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), t\big)\right)\boldsymbol{\eta}(t)\right] dt + \frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\boldsymbol{z}(b), \dot{\boldsymbol{z}}(b), b\big)\boldsymbol{\eta}(b) = 0. \quad \text{(C.5)}$$

As $\boldsymbol{\eta}$ is an arbitrary function, this equality requires

$$\begin{aligned} &\left[\frac{\partial \mathcal{I}}{\partial \mathbf{y}}\big(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), t\big)\right]^T - \left[\frac{d}{dt}\left(\frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), t\big)\right)\right]^T = \mathbf{0}, \\ &\boldsymbol{z}(a) = \boldsymbol{z}_a, \qquad \left[\frac{\partial \mathcal{I}}{\partial \mathbf{v}}\big(\boldsymbol{z}(b), \dot{\boldsymbol{z}}(b), b\big)\right]^T = \mathbf{0}. \end{aligned} \quad \text{(C.6)}$$

Equation (C.6) is known as the Euler–Lagrange equation. It states that, for ($\boldsymbol{z}$,$\dot{\boldsymbol{z}}$) to define a stationary point of $\mathcal{J}$, then they must satisfy (C.6). It is emphasized that (C.6) is a necessary condition on $(\boldsymbol{z}, \dot{\boldsymbol{z}})$ to make $\mathcal{J}$ stationary, but it is not a sufficient condition. It is additionally noted that (C.6) provides a stationary point of $\mathcal{J}$, but the resulting stationary point could be a local minima, local maxima, or saddle point.

Appendix D Evaluation of gradients in the Euler–Lagrange equations for WLS with S-reduction trial subspaces

We now derive the specific form of the Euler–Lagrange equations for the WLS formulation with S-reduction trial subspaces. Without loss of generality, we present the derivation for a single window $t \in [0, T]$ with a constant basis $\mathbf{V}$ and weighting matrix $\mathbf{A}$. To obtain the specific form of the Euler–Lagrange equations for the WLS formulation, we need to evaluate the gradients in (C.6) for the integrand

$$\begin{aligned} \mathcal{I} : (\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &\mapsto \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\big]^T \mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\big], \\ &: \mathbb{R}^K \times \mathbb{R}^K \times [0, T] \to \mathbb{R}. \end{aligned}$$

To evaluate the gradients, we first expand $\mathcal{I}$:

$$\begin{aligned}\mathcal{I}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) &= \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}} - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\\
&= \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}}\big] - \frac{1}{2}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}}\big] - \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\\
&+ \frac{1}{2}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big].\end{aligned}$$

Since $\mathbf{A}$ is symmetric,

$$\mathcal{I}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) = \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}}\big] - \big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big] + \frac{1}{2}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]. \tag{D.1}$$

For notational purposes, we write the above as,

$$\mathcal{I}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) = \mathcal{I}_1(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) + \mathcal{I}_2(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) + \mathcal{I}_3(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau),$$

where

$$\begin{aligned}\mathcal{I}_1(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) &= \frac{1}{2}\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}}\big],\\
\mathcal{I}_2(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) &= -\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big],\\
\mathcal{I}_3(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau) &= \frac{1}{2}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big].\end{aligned}$$

Constructing the Euler–Lagrange equations for this functional $\mathcal{I}$ requires evaluating the derivatives $\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{y}}}$ and $\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{v}}}$. We start by evaluating $\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{y}}}$ and go term by term.

Starting with $\mathcal{I}_1(\hat{\mathbf{y}},\hat{\mathbf{v}})$, we see,

$$\frac{\partial \mathcal{I}_1}{\partial \hat{\mathbf{y}}} = \mathbf{0},$$

where it is noted that $\mathcal{I}_1$ only depends on $\hat{\mathbf{v}}$. Working with the second term:

$$\begin{aligned}\frac{\partial \mathcal{I}_2}{\partial \hat{\mathbf{y}}} &= -\frac{\partial}{\partial \hat{\mathbf{y}}}\Big(\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big)\\
&= -\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\frac{\partial}{\partial \hat{\mathbf{y}}}\Big(\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big)\\
&= -\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \hat{\mathbf{y}}}\\
&= -\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}\\
&= -[\hat{\mathbf{v}}]^T\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\end{aligned}$$

where we have suppressed the arguments of the Jacobian for simplicity; e.g., formally

$$\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}} : (\mathbf{w},\tau) \mapsto \frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{w},\tau).$$

For $\mathcal{I}_3$,

$$\frac{\partial \mathcal{I}_3}{\partial \hat{\mathbf{y}}} = \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{y}}}\Big(\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big) \tag{D.2}$$

$$= [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \hat{\mathbf{y}}} \tag{D.3}$$

$$= [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}. \tag{D.4}$$

This gives the final expression,

$$\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{y}}} = -\big[\mathbf{V}\hat{\mathbf{v}}\big]^T \mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V} + [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)]^T \mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}.$$

We now evaluate $\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{v}}}$ and again go term by term. Starting with $\mathcal{I}_1$,

$$\begin{aligned}\frac{\partial \mathcal{I}_1}{\partial \hat{\mathbf{v}}} &= \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{v}}}\Big(\big[\mathbf{V}\hat{\mathbf{v}}\big]\mathbf{A}\big[\mathbf{V}\hat{\mathbf{v}}\big]\Big)\\ &= [\hat{\mathbf{v}}]^T\mathbf{V}^T\mathbf{A}\mathbf{V}.\end{aligned}$$

Now working with the second term:

$$\begin{aligned}\frac{\partial \mathcal{I}_2}{\partial \hat{\mathbf{v}}} &= \frac{\partial}{\partial \hat{\mathbf{v}}}\Big(\big[\mathbf{V}\hat{\mathbf{v}}\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\big]\Big)\\ &= \frac{\partial}{\partial \hat{\mathbf{v}}}\Big([\hat{\mathbf{v}}]^T\Big)\mathbf{V}^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\big]\\ &= \Big[\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\Big]^T\\ &= [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)]^T\mathbf{A}\mathbf{V}.\end{aligned}$$

Finally, for the last term,

$$\frac{\partial \mathcal{I}_3}{\partial \hat{\mathbf{v}}} = \mathbf{0},$$

where it is noted that $\mathcal{I}_3$ only depends on $\hat{\mathbf{y}}$. We thus have,

$$\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{v}}} = [\hat{\mathbf{v}}]^T\mathbf{V}^T\mathbf{A}\mathbf{V} - [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)]^T\mathbf{A}\mathbf{V}.$$

Combining all terms and evaluating at $(\hat{\boldsymbol{x}}(t), \dot{\hat{\boldsymbol{x}}}(t), t)$,

$$\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{y}}}(\hat{\boldsymbol{x}}(t), \dot{\hat{\boldsymbol{x}}}(t), t) - \frac{d}{dt}\left[\frac{\partial \mathcal{I}}{\partial \hat{\mathbf{v}}}(\hat{\boldsymbol{x}}(t), \dot{\hat{\boldsymbol{x}}}(t), t)\right] = -\big[\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t)\big]^T\mathbf{A}\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\big]\mathbf{V} +$$
$$[\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)]^T\mathbf{A}\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\big]\mathbf{V} - \frac{d}{dt}\Big[[\dot{\hat{\boldsymbol{x}}}(t)]^T\mathbf{V}^T\mathbf{A}\mathbf{V} - [\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)]^T\mathbf{A}\mathbf{V}\Big] = \mathbf{0}.$$

To put this in a more recognizable form, we can pull out the common factor in the first two terms,

$$-\Big(\big[\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t)\big]^T - [\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)]^T\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\mathbf{V} -$$
$$\frac{d}{dt}\Big[[\dot{\hat{\boldsymbol{x}}}(t)]^T\mathbf{V}^T\mathbf{A}\mathbf{V} - [\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)]^T\mathbf{A}\mathbf{V}\Big] = \mathbf{0}.$$

Taking the transpose to put into the common column major format,

$$-\mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\big]^T\mathbf{A}\Big(\big[\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t)\big] - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\Big) - \frac{d}{dt}\Big[\mathbf{V}^T\mathbf{A}\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\Big] = \mathbf{0}.$$

We now factor the second term,

$$-\mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\big]^T\mathbf{A}\Big(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\Big) - \mathbf{V}^T\mathbf{A}\frac{d}{dt}\Big[\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}, t)\Big] = \mathbf{0}.$$

Gathering terms and multiplying by negative one, the final form of the Euler–Lagrange equations are obtained,

$$\Big[\mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\big]^T\mathbf{A} + \mathbf{V}^T\mathbf{A}\frac{d}{dt}\Big]\Big(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\Big) = \mathbf{0}. \qquad \text{(D.5)}$$

This is the WLS-ROM. Note that this is a second order equation and can be written as two separate first order equations. Defining the "costate" as,

$$\hat{\boldsymbol{\lambda}} : \tau \mapsto \dot{\hat{\boldsymbol{x}}}(\tau) - \mathbf{M}^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, \tau),$$

we can manipulate Eq. (D.5) as follows: First, we add and subtract the first term multiplied by $\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}$,

$$\left[\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A} + \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right) + \mathbf{V}^T\mathbf{A}\frac{d}{dt}\right] \\ \left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) = \mathbf{0}.$$

Pulling out the term multiplied by the positive portion of $\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}$,

$$\left[\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right) + \\ \mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A} + \mathbf{V}^T\mathbf{A}\frac{d}{dt}\right]\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) = \mathbf{0}.$$

Splitting into two separate terms,

$$\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right)\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) + \\ \left[\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A} + \mathbf{V}^T\mathbf{A}\frac{d}{dt}\right]\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) = \mathbf{0}.$$

Pulling $\mathbf{M}^{-1}\mathbf{V}^T\mathbf{A}$ inside the parenthesis on the second term,

$$\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^n}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right)\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) + \\ \left[\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V} + \mathbf{M}\frac{d}{dt}\right]\left(\dot{\hat{\boldsymbol{x}}}(t) - \mathbf{M}^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) = \mathbf{0}.$$

By definition, the term inside the parenthesis of the second term is $\hat{\boldsymbol{\lambda}}(t)$,

$$\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right)\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right) + \\ \mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V}\hat{\boldsymbol{\lambda}}(t) + \mathbf{M}\frac{d}{dt}\hat{\boldsymbol{\lambda}}(t) = \mathbf{0}.$$

Re-arranging,

$$\mathbf{M}\frac{d}{dt}\hat{\boldsymbol{\lambda}}(t) + \mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V}\hat{\boldsymbol{\lambda}}(t) \\ = -\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right)\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right).$$

We thus get the splitting

$$\mathbf{M}\frac{d}{dt}\hat{\boldsymbol{x}}(t) - \mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t) = \mathbf{M}\hat{\boldsymbol{\lambda}}(t), \\ \mathbf{M}\frac{d}{dt}\hat{\boldsymbol{\lambda}}(t) + \mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right]^T\mathbf{A}\mathbf{V}\hat{\boldsymbol{\lambda}}(t) = \\ -\mathbf{V}^T[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)]^T\mathbf{A}\left(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\right)\left(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t) + \mathbf{x}_{\text{ref}}, t)\right).$$

Appendix E Evaluation of gradients for optimal control formulation

When formulated as an optimal control problem of Lagrange type, the gradients of the Hamiltonian with respect to the state, controller, and costate need to be evaluated. This section details this evaluation. The case derivation is presented for the case with one window, for notational simplicity.

The Pontryagin Maximum Principle leverages the following Hamiltonian,

$$\begin{aligned}\mathcal{H} \,:\, (\hat{\mathbf{y}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{v}}, \tau) &\mapsto \hat{\boldsymbol{\mu}}^T \left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}} \right] + \mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) \\ &: \mathbb{R}^K \times \mathbb{R}^K \times \mathbb{R}^K \times [0, T] \to \mathbb{R},\end{aligned}$$

where,

$$\begin{aligned}\mathcal{L} : (\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) \mapsto \frac{1}{2}&\left[\mathbf{V}\left([\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}\right) - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right]^T \mathbf{A} \\ &\left[\mathbf{V}\left([\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}\right) - \boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right], \\ : \mathbb{R}^K \times \mathbb{R}^K \times [0, T] \to \mathbb{R}.&\end{aligned}$$

As described in Section 3.2.2, to derive the stationary conditions of the WLS objective function, we require evaluating the following gradients, $\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{\mu}}}$, $\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{y}}}$, and $\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{v}}}$. Starting with $\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{\mu}}}$, we have,

$$\left[\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{\mu}}}\right]^T = [\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}.$$

Next, we address $\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{y}}}$.

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{y}}} &= \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}\right] + \mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right] \\ &= \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}\right]\right] + \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right] \\ &= \hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial}{\partial \hat{\mathbf{y}}}\left(\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right)\right] + \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right] \\ &= \hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \hat{\mathbf{y}}}\right] + \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right] \\ &= \hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}\right] + \frac{\partial}{\partial \hat{\mathbf{y}}}\left[\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right].\end{aligned}$$

To evaluate $\frac{\partial}{\partial \hat{\mathbf{y}}}\left[\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau)\right]$, we leverage the previous result (D.1) and insert $\hat{\mathbf{v}} = [\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + [\mathbf{M}]^{-1}\hat{\mathbf{v}}(t)$. This leads to the expression for the expanded Lagrangian,

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) = \frac{1}{2}&\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)+\mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}(t)\right]^T\mathbf{A}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)+\mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}(t)\right] \\ -\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)+\mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}(t)\right]^T&\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)\right]+\frac{1}{2}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)\right]^T\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}}, \tau)\right].\end{aligned}$$

Again for notational purposes, we split this into three terms,

$$\mathcal{L}(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) = \mathcal{L}_1(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) + \mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) + \mathcal{L}_3(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau),$$

where

$$\begin{aligned}\mathcal{L}_1(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &= \frac{1}{2}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + \mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}\right]^T\mathbf{A}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + \mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}\right], \\ \mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &= -\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau) + \mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}\right]^T\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right], \\ \mathcal{L}_3(\hat{\mathbf{y}}, \hat{\mathbf{v}}, \tau) &= \frac{1}{2}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right]^T\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}} + \mathbf{x}_{\text{ref}}, \tau)\right].\end{aligned}$$

Evaluating the first term,

$$\begin{aligned}
\frac{\partial \mathcal{L}_1}{\partial \hat{\mathbf{y}}} &= \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{y}}}\Big([\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\big[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big)+\\
&\quad \frac{\partial}{\partial \hat{\mathbf{y}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\big[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big)+\\
&\quad \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{y}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}\Big),\\
&= [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}+[\hat{\boldsymbol{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\mathbf{V},\\
&= \Big([\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}+[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\Big)\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\\
&= [\hat{\mathbf{v}}]^T\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\\
&= [\mathbf{V}\hat{\mathbf{v}}]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}.
\end{aligned}$$

Now evaluating the second term,

$$\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial \hat{\mathbf{y}}} &= -\frac{\partial}{\partial \hat{\mathbf{y}}}\big[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]-\frac{\partial}{\partial \hat{\mathbf{y}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]\Big)\\
&= -\frac{\partial}{\partial \hat{\mathbf{y}}}[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\big[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\big]-[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}\\
&= -2[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}-[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\\
&= -\Big[2[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}+[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\Big]\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\\
&= -\Big([\mathbf{V}\hat{\mathbf{v}}]^T+\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}.
\end{aligned}$$

Next, we can use the result (D.2) to have,

$$\frac{\partial \mathcal{L}_3}{\partial \hat{\mathbf{y}}} = [\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}.$$

Thus we have

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} &= [\mathbf{V}\hat{\mathbf{v}}]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}-\Big([\mathbf{V}\hat{\mathbf{v}}]^T+\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}+[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\\
&= \Big([\mathbf{V}\hat{\mathbf{v}}]^T-\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)^T\Big)\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}-\Big([\mathbf{V}\hat{\mathbf{v}}]^T-\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)^T\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},\\
&= \Big([\mathbf{V}\hat{\mathbf{v}}]^T-\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)^T\Big)\Big(\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T-\mathbf{I}\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V},
\end{aligned}$$

such that,

$$\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{y}}} = \hat{\boldsymbol{\mu}}^T\big[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}\big]+\Big([\mathbf{V}\hat{\mathbf{v}}]^T-\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)^T\Big)\Big(\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T-\mathbf{I}\Big)\mathbf{A}\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\mathbf{V}.$$

Equivalently we write

$$\left[\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{y}}}\right]^T = \mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\big]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\hat{\boldsymbol{\mu}}+\mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \mathbf{y}}\big]^T\mathbf{A}\Big(\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}-\mathbf{I}\Big)\Big(\mathbf{V}\hat{\mathbf{v}}-\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\Big).$$

Next, we evaluate $\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{v}}}$:

$$\begin{aligned}
\frac{\partial \mathcal{H}}{\partial \hat{\boldsymbol{v}}} &= \frac{\partial}{\partial \hat{\boldsymbol{v}}}\left[\hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)+[\mathbf{M}]^{-1}\hat{\boldsymbol{v}}\right]+\mathcal{L}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau)\right] \\
&= \frac{\partial}{\partial \hat{\mathbf{v}}}\left[\hat{\boldsymbol{\mu}}^T\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)+[\mathbf{M}]^{-1}\hat{\mathbf{v}}\right]\right]+\frac{\partial}{\partial \hat{\mathbf{v}}}\left[\mathcal{L}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau)\right] \\
&= [\hat{\boldsymbol{\mu}}]^T[\mathbf{M}]^{-1}+\frac{\partial}{\partial \hat{\mathbf{v}}}\left[\mathcal{L}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau)\right].
\end{aligned}$$

To evaluate $\frac{\partial}{\partial \hat{\mathbf{v}}}\left[\mathcal{L}(\hat{\mathbf{y}},\hat{\mathbf{v}},\tau)\right]$, we again go term by term. Starting with the first term,

$$\begin{aligned}
\frac{\partial \mathcal{L}_1}{\partial \hat{\mathbf{y}}} &= \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{v}}}\Big([\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]\Big)+ \\
&\quad \frac{\partial}{\partial \hat{\mathbf{v}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]\Big)+ \\
&\quad \frac{1}{2}\frac{\partial}{\partial \hat{\mathbf{v}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\hat{\mathbf{v}}\Big) \\
&= \Big([\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]\Big)^T+[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1} \\
&= \left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}+[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}.
\end{aligned}$$

Moving on to the second term,

$$\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial \hat{\mathbf{v}}} &= -\frac{\partial}{\partial \hat{\mathbf{v}}}\left[\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]^T\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]-\frac{\partial}{\partial \hat{\mathbf{v}}}\Big([\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]\Big) \\
&= -\left[[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}})\right]^T \\
&= -[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}.
\end{aligned}$$

For the third term we have simply,

$$\frac{\partial \mathcal{L}_3}{\partial \hat{\mathbf{v}}} = \mathbf{0}.$$

Thus,

$$\begin{aligned}
\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{v}}} &= [\hat{\boldsymbol{\mu}}]^T[\mathbf{M}]^{-1}+\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}+ \\
&\quad [\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}-[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}, \\
&= \Big([\hat{\boldsymbol{\mu}}]^T-[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}\Big)[\mathbf{M}]^{-1}+\Big(\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]^T\mathbf{A}\mathbf{V}+[\hat{\mathbf{v}}]^T\Big)[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1} \\
&= \Big([\hat{\boldsymbol{\mu}}]^T-[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)]^T\mathbf{A}\mathbf{V}\Big)[\mathbf{M}]^{-1}+\Big(\left[\boldsymbol{f}(\mathbf{V}\hat{\mathbf{y}}+\mathbf{x}_{\text{ref}},\tau)\right]^T\mathbf{A}\mathbf{V}+[\hat{\mathbf{v}}]^T\Big)[\mathbf{M}]^{-1} \\
&= [\hat{\boldsymbol{\mu}}]^T[\mathbf{M}]^{-1}+[\hat{\mathbf{v}}]^T[\mathbf{M}]^{-1}
\end{aligned}$$

Or, equivalently,

$$\frac{\partial \mathcal{H}}{\partial \hat{\mathbf{v}}} = [\mathbf{M}]^{-1}[\hat{\boldsymbol{\mu}}+\hat{\mathbf{v}}].$$

Evaluating at $(\hat{\boldsymbol{x}}(t),\dot{\hat{\boldsymbol{x}}}(t),t)$, the gradients in the Pontryagin Maximum Principle yield

$$\begin{aligned}
&\frac{d}{dt}\hat{\boldsymbol{x}}(t) = [\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}},t)+[\mathbf{M}]^{-1}\hat{\boldsymbol{u}}(t) \\
&\frac{d}{dt}\hat{\boldsymbol{\lambda}}(t)+\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\hat{\boldsymbol{\lambda}}(t) = -\mathbf{V}^T\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right]^T\mathbf{A}\Big(\mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}-\mathbf{I}\Big)\Big(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t)-\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}})\Big) \\
&\hat{\boldsymbol{\lambda}}(t) = -\hat{\boldsymbol{u}}(t).
\end{aligned}$$

This can be written equivalently as

$$\frac{d}{dt}\hat{\boldsymbol{x}}(t) = [\mathbf{M}]^{-1}\mathbf{V}^T\mathbf{A}\boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}}) + [\mathbf{M}]^{-1}\hat{\boldsymbol{u}}(t)$$

$$\frac{d}{dt}\hat{\boldsymbol{u}}(t) + \mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\big]^T\mathbf{A}\mathbf{V}[\mathbf{M}]^{-1}\hat{\boldsymbol{u}}(t) = -\mathbf{V}^T\big[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\big]^T\mathbf{A}\Big(\mathbf{I} - \mathbf{V}[\mathbf{M}]^{-1}\mathbf{V}^T\Big)\mathbf{A}\Big(\mathbf{V}\dot{\hat{\boldsymbol{x}}}(t) - \boldsymbol{f}(\mathbf{V}\hat{\boldsymbol{x}}(t)+\mathbf{x}_{\text{ref}})\Big).$$

References

[1] R. Abgrall and R. Crisovan, *Model reduction using L1-norm minimization as an application to nonlinear hyperbolic problems*, International Journal for Numerical Methods in Fluids, 87 (2018), pp. 628–651.

[2] M. Balajewicz, I. Tezaur, and E. Dowell, *Minimal subspace rotation on the Stiefel manifold for stabilization and enhancement of projection-based reduced order models for the compressible Navier–Stokes equations*, Journal of Computational Physics, 321 (2016), pp. 224–241.

[3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, *An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations*, C. R. Acad. Sci. Paris, 339 (2004), pp. 667–672.

[4] F. Bassi and S. Rebay, *A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations*, Journal of Computational Physics, 131 (1997), pp. 267 – 279.

[5] M. Baumann, P. Benner, and J. Heiland, *Space-time Galerkin POD with application in optimal control of semi-linear partial differential equations*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1611–A1641.

[6] C. Beattie and S. Gugercin, *Structure-preserving model reduction for nonlinear port-Hamiltonian systems*, in Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, IEEE, 2011, pp. 6564–6569.

[7] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Review, 57 (2015), pp. 483–531.

[8] M. Bergmann, C. Bruneaua, and A. Iollo, *Enablers for robust POD models*, Journal of Computational Physics, 228 (2009), pp. 516–538.

[9] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annu. Rev. Fluid Mech., 25 (1993), pp. 539–575.

[10] P. Bochev and M. Gunzburger, *Finite element methods of least-squares type*, SIAM Review, 40 (1998), pp. 789–837.

[11] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, 1965.

[12] T. Bui-Thanh, *Model-constrained optimization methods for reduction of parameterized large-scale systems*, PhD thesis, Massachusetts Institute of Technology, 2007.

[13] T. Bui-Thanh, K. Willcox, and O. Ghattas, *Model reduction for large-scale systems with high-dimensional parametric input space*, SIAM Journal on Scientific Computing, 30 (2008), pp. 3270–3288.

[14] ———, *Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications*, AIAA Journal, 46 (2008), pp. 2520–2529.

[15] A. Caiazzo, T. Iliescu, V. John, and S. Schyschlowa, *A numerical investigation of velocity–pressure reduced order models for incompressible flows*, Journal of Computational Physics, 259 (2014), pp. 598 – 616.

[16] K. Carlberg, *Model reduction of nonlinear mechanical systems via optimal projection and tensor approximation*, PhD thesis, Stanford University, 2011.

[17] K. Carlberg, *Adaptive h-refinement for reduced-order models*, International Journal for Numerical Methods in Engineering, 102 (2015), pp. 1192–1210.

[18] K. Carlberg, M. Barone, and H. Antil, *Galerkin v. least-squares Petrov-Galerkin projection in nonlinear model reduction*, Journal of Computational Physics, 330 (2017), pp. 693–734.

[19] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, Int. J. Numer. Methods Eng., 86 (2011), pp. 155–181.

[20] K. Carlberg, Y. Choi, and S. Sargsyan, *Conservative model reduction for finite-volume models*, Journal of Computational Physics, 371 (2018), pp. 280–314.

[21] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem, *The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows*, Journal of Computational Physics, 242 (2013), pp. 623–647.

[22] K. Carlberg, R. Tuminaro, and P. Boggs, *Preserving Lagrangian structure in nonlinear model reduction with application to structural dynamics*, SIAM J. Sci. Comput., 37 (2015), pp. B153—B184.

[23] J. Chan, *Entropy stable reduced order modeling of nonlinear conservation laws*, arXiv e-print, (2019).

[24] S. Chaturantabut, C. Beattie, and S. Gugercin, *Structure-preserving model reduction for nonlinear port-Hamiltonian systems*, SIAM Journal on Scientific Computing, 38 (2016), pp. B837–B865.

[25] Y. Choi and K. Carlberg, *Space–time least-squares Petrov–Galerkin projection for nonlinear model reduction*, SIAM Journal on Scientific Computing, 41 (2019), pp. A26–A58.

[26] P. G. Constantine and Q. Wang, *Residual minimizing model interpolation for parameterized nonlinear dynamical systems*, SIAM J. Sci. Comput., (2012).

[27] B. A. Conway, *A survey of methods available for the numerical optimization of continuous dynamic systems*, J. Optim. Theory Appl., 152 (2012), pp. 271–306.

[28] Z. Drmac and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, J. Sci. Comput., 38 (2016), pp. A631–A648.

[29] P. Etter and K. Carlberg, *Online adaptive basis refinement and compression for reduced-order models via vector-space sieving*, arXiv e-print, (2019).

[30] R. Everson and L. Sirovich, *Karhunen-Loève procedure for gappy data*, Journal of the Optical Society of America A, 12 (1995), pp. 1657–1644.

[31] C. Farhat, P. Avery, T. Chapman, and J. Cortial, *Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency*, International Journal for Numerical Methods in Engineering, 98 (2014), pp. 625–662.

[32] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Review, 43 (2001), pp. 89–112.

[33] S. Gugercin, A. Antoulas, and C. Beattie, *$\mathcal{H}_2$ model reduction for large-scale linear dynamical systems*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 609–638.

[34] M. D. Gunzburger and P. B. Bochev, *Least-Squares Finite Element Methods*, Springer, New York, NY, 2009.

[35] T. Iliescu and Z. Wang, *Variational multiscale proper orthogonal decomposition: Navier–Stokes equations*, Numerical Methods for Partial Differential Equations, 30 (2014), pp. 641–663.

[36] I. Kalashnikova, S. Arunajatesan, M. F. Barone, B. G. van Bloemen Waanders, and J. A. Fike, *Reduced order modeling for prediction and control of large-scale systems*, Report SAND2014-4693, Sandia, May 2014.

[37] D. E. Kirk, *Optimal Control Theory: An Introduction*, Prentice Hall, 2007.

[38] D. Knoll and D. Keyes, *Jacobian-free Newton—Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357 – 397.

[39] S. Lall, P. Krysl, and J. E. Marsden, *Structure-preserving model reduction for mechanical systems*, Physica D: Nonlinear Phenomena, 184 (2003), pp. 304 – 318. Complexity and Nonlinearity in Physical Systems – A Special Issue to Honor Alan Newell.

[40] K. Lee and K. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, arXiv e-print, (2018).

[41] P. LeGresley and J. Alonso, *Airfoil design optimization using reduced order models based on proper orthogonal decomposition*.

[42] ———, *Dynamic domain decomposition and error correction for reduced order models*.

[43] ———, *Investigation of non-linear projection for POD based reduced order models for Aerodynamics*.

[44] S. Lenhart and J. Workman, *Optimal Control Applied to Biological Models*, Chpman & Hall/CRC Press, 2007.

[45] M. McAsey, L. Mou, and W. Han, *Convergence of the forward-backward sweep method in optimal control*, Comp. Opt. and Appl., 53 (2012), pp. 207–226.

[46] B. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Transactions on Automatic Control, 26 (1981), pp. 17–32.

[47] D. Morrison, J. Riley, and J. Zancanaro, *Multiple shooting method for two-point boundary value problems*, Communications of the ACM, 5 (1962), pp. 613–614.

[48] C. T. Mullis and R. A. Roberts, *Synthesis of minimum roundoff noise fixed point digital filters*, IEEE Transactions on Circuits and Systems, 23 (1976), pp. 551–562.

[49] N. Ngoc Cuong, K. Veroy, and A. T. Patera, *Certified real-time solution of parametrized partial differential equations*, Springer Netherlands, Dordrecht, 2005, pp. 1529–1564.

[50] E. Parish, C. Wentland, and K. Duraisamy, *The Adjoint Petrov–Galerkin method for non-linear model reduction*, CMAME (Submitted), (2018), p. 50.

[51] B. Peherstorfer and K. Willcox, *Online adaptive model reduction for nonlinear systems via low-rank updates*, J. Sci. Comput., 37 (2015), pp. A2123–A2150.

[52] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici, *Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods*, Journal of Fluids Engineering, 124 (2001), pp. 70–80.

[53] A. Rac, *A survey of numerical methods for optimal control*, Advances in the Astronautical Sciences, 135 (2010).

[54] D. V. Rovas, *Reduced-basis output bound methods for parametrized partial differential equations*, PhD thesis, Massachusetts Institute of Technology, 2003.

[55] C. W. Rowley, T. Colonius, and R. M. Murray, *Model reduction for compressible flows using POD and Galerkin projection*, Physica D: Nonlinear Phenomena, 189 (2004), pp. 115–129.

[56] G. Rozza, D. B. P. Huynh, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations*, Archives of Computational Methods in Engineering, 15 (2008), p. 229.

[57] V. Rusanov, *On difference schemes of third order accuracy for nonlinear hyperbolic systems*, Journal of Computational Physics, 5 (1970), pp. 507–516.

[58] O. San and T. Iliescu, *Proper orthogonal decomposition closure models for fluid flows: Burgers equation*, Comput. Methods Appl. Mech. Engrg, 5 (2014), pp. 217–237.

[59] O. San and T. Iliescu, *A stabilized proper orthogonal decomposition reduced-order model for large scale quasi-geostrophic ocean circulation*, Adv Comput Math, 41 (2015), pp. 1289–1319.

[60] O. San and R. Maulik, *Neural network closures for nonlinear model order reduction*, Advances in Computational Mathematics, 44 (2018), pp. 1717–1750.

[61] A. W. und A. Griewank, *Getting started with ADOL-C. In Combinatorial Scientific Computing.*, Chapman-Hall CRC Computational Science, 2012.

[62] K. Urban and A. T. Patera, *A new error bound for reduced basis approximation of parabolic partial differential equations*, Comptes Rendus Mathematique, 350 (2012), pp. 203 – 207.

[63] K. Veroy and A. T. Patera, *Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: Rigorous reduced-basis a posteriori error bounds*, International Journal for Numerical Methods in Fluids, 47 (2005), pp. 773–788.

[64] K. Veroy, C. Prud'homme, D. Rovas, and A. Patera, *A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations*.

[65] Q. Wang, N. Ripamonti, and J. S. Hesthaven, *Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism*, Journal of Computational Physics, (2019).

[66] Z. Wang, *Reduced-Order Modeling of Complex Engineering and Geophysical Flows: Analysis and Computations*, PhD thesis, Virginia Polytechnic Institute and State University, 2012.

[67] C. R. Wentland, C. Huang, and K. Duraisamy, *Closure of reacting flow reduced-order models via the Adjoint Petrov-Galerkin Method*.

[68] M. Yano, A. T. Patera, and K. Urban, *A space-time hp-interpolation-based certified reduced basis method for Burgers equation*, Mathematical Models and Methods in Applied Sciences, 24 (2014), pp. 1903–1935.

The Adjoint Petrov–Galerkin Method for Non-Linear Model Reduction

Eric J. Parish[b], Christopher Wentland[a], Karthik Duraisamy[a]

[a]Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI
[b]Sandia National Laboratories, Livermore, CA

Abstract

We formulate a new projection-based reduced-ordered modeling technique for non-linear dynamical systems. The proposed technique, which we refer to as the Adjoint Petrov–Galerkin (APG) method, is derived by decomposing the generalized coordinates of a dynamical system into a resolved coarse-scale set and an unresolved fine-scale set. A Markovian finite memory assumption within the Mori-Zwanzig formalism is then used to develop a reduced-order representation of the coarse-scales. This procedure leads to a closed reduced-order model that displays commonalities with the adjoint stabilization method used in finite elements. The formulation is shown to be equivalent to a Petrov–Galerkin method with a non-linear, time-varying test basis, thus sharing some similarities with the Least-Squares Petrov–Galerkin method. Theoretical analysis examining *a priori* error bounds and computational cost is presented. Numerical experiments on the compressible Navier-Stokes equations demonstrate that the proposed method can lead to improvements in numerical accuracy, robustness, and computational efficiency over the Galerkin method on problems of practical interest. Improvements in numerical accuracy and computational efficiency over the Least-Squares Petrov–Galerkin method are observed in most cases.

1. Introduction

High-fidelity numerical simulations play a critical role in modern-day engineering and scientific investigations. The computational cost of high-fidelity or full-order models (FOMs) is, however, often prohibitively expensive. This limitation has led to the emergence of reduced-order modeling techniques. Reduced-order models (ROMs) are formulated to *approximate* solutions to a FOM on a low-dimensional manifold. Common reduced-order modeling techniques include balanced truncation [1, 2], Krylov subspace techniques [3], reduced-basis methods [4], and the proper orthogonal decomposition approach [5]. Reduced-order models based on such techniques have been implemented in a wide variety of disciplines and have been effective in reducing the computational cost associated with high-fidelity numerical simulations [6, 7, 8].

Projection-based reduced-order models constructed from proper orthogonal decomposition (POD) have proved to be an effective tool for model order reduction of complex systems. In the POD-ROM approach, snapshots from a high-fidelity simulation (or experiment) are used to construct an orthonormal basis spanning the solution space. A small, truncated set of these basis vectors forms the *trial* basis. The POD-ROM then seeks a solution within the range of the trial basis via projection. Galerkin projection, in which the FOM equations are projected onto the same trial subspace, is the simplest type of projection. The Galerkin ROM

Email addresses: ejparis@sandia.gov (Eric J. Parish), chriswen@umich.edu (Christopher Wentland), kdur@umich.edu (Karthik Duraisamy)

(G ROM) has been used successfully in a variety of problems. When applied to general non-self-adjoint and non-linear problems, however, theoretical analysis and numerical experiments have shown that Galerkin ROM lacks *a priori* guarantees of stability, accuracy, and convergence [9]. This last issue is particularly challenging as it demonstrates that enriching a ROM basis does not necessarily improve the solution [10]. The development of stable and accurate reduced-order modeling techniques for complex non-linear systems is the motivation for the current work.

A significant body of research aimed at producing accurate and stable ROMs for complex non-linear problems exists in the literature. These efforts include, but are not limited to, "energy-based" inner products [9, 11], symmetry transformations [12], basis adaptation [13, 14], L^1-norm minimization [15], projection subspace rotations [16], and least-squares residual minimization approaches [17, 18, 19, 20, 21, 22, 23, 24]. The Least-Squares Petrov–Galerkin (LSPG) [22] method comprises a particularly popular least-squares residual minimization approach and has been proven to be an effective tool for non-linear model reduction. Defined at the fully-discrete level (i.e., after spatial and temporal discretization), LSPG relies on least-squares minimization of the FOM residual at each time-step. While the method lacks *a priori* stability guarantees for general non-linear systems, it has been shown to be effective for complex problems of interest [24, 23, 25]. Additionally, as it is formulated as a minimization problem, physical constraints such as conservation can be naturally incorporated into the ROM formulation [26]. At the fully-discrete level, LSPG is sensitive to both the time integration scheme as well as the time-step. For example, in Ref. [23] it was shown that LSPG produces optimal results at an intermediate time-step. Another example of this sensitivity is that, when applied to explicit time integration schemes, the LSPG approach reverts to a Galerkin approach. This limits the scope of LSPG to implicit time integration schemes, which can in turn increase the cost of the ROM [23][1]. This is particularly relevant in the case where the optimal time-step of LSPG is small, thus requiring many time-steps of an implicit solver. Despite these challenges, the LSPG approach is arguably the most robust technique that is used for ROMs of non-linear dynamical systems.

A second school of thought addresses stability and accuracy of ROMs from a closure modeling viewpoint. This follows from the idea that instabilities and inaccuracies in ROMs can, for the most part, be attributed to the truncated modes. While these truncated modes may not contain a significant portion of the system energy, they can play a significant role in the dynamics of the ROM [27]. This is analogous to the closure problem encountered in large eddy simulation. Research has examined the construction of mixing length [28], Smagorinsky-type [27, 29, 30, 31], and variational multiscale (VMS) closures [27, 32, 33, 34] for POD-ROMs. The VMS approach is of particular relevance to this work. Originally developed in the context of finite element methods, VMS is a formalism to derive stabilization/closure schemes for numerical simulations of multiscale problems. The VMS procedure is centered around a sum decomposition of the solution u in terms of resolved/coarse-scales $\tilde{u}$ and unresolved/fine-scales u'. The impact of the fine-scales on the evolution of the coarse-scales is then accounted for by devising an approximation to the fine-scales. This approximation is often referred to as a "subgrid-scale" or "closure" model.

Research has examined the application of both phenomenological and residual-based subgrid-scale models to POD-ROMs. In Refs. [32, 35, 36], Iliescu and co-workers examine the construction of eddy-viscosity-based ROM closures via the VMS method. These eddy-viscosity methods are directly analogous to the eddy-viscocity philosophy used in turbulence modeling. While they do not guarantee stability *a priori*, these ROMs have been shown to enhance accuracy on a variety of problems in fluid dynamics. However, as eddy-viscosity methods are based on phenomenological assumptions specific to three-dimensional turbulent flows, their scope may be limited to specific types of problems. Residual-based methods, which can also be derived from VMS, constitute a more general modeling strategy. The subgrid-scale model emerging from a residual-based method typically appears as a term that is proportional to the residual of

[1]It is possible to use LSPG with an explicit time integrator by formulating the ROM for an implicit time integration scheme, and then time integrating the resulting system with an explicit integrator.

the full-order model; if the governing equations are exactly satisfied by the ROM, then the model is inactive. While residual-based methods in ROMs are not as well-developed as they are in finite element methods, they have been explored in several contexts. In Ref. [33], ROMs of the Navier-Stokes equations are stabilized using residual-based methods. This stabilization is performed by solving a ROM stabilized with a method such as streamline upwind Petrov–Galerkin (SUPG) and augmenting the POD basis with additional modes computed from the residual of the Navier-Stokes equations. In Ref. [37], residual-based stabilization is developed for velocity-pressure ROMs of the incompressible Navier-Stokes equations. Both eddy-viscosity and residual-based methods have been shown to improve ROM stability and performance. The majority of existing work on residual-based stabilization (and eddy-viscosity methods) is focused on ROMs formulated from continuous projection (i.e., projecting a continuous PDE using a continuous basis). In this instance, the ROM residual is defined at the continuous level and is directly linked to the governing partial differential equation. In many applications (arguably the majority [11]), however, the ROM is constructed through discrete projection (i.e., projecting the spatially discretized PDE using a discrete basis). In this instance, the ROM residual is defined at the semi-discrete level and is tied to the *spatially discretized* governing equations. Residual-based methods for ROMs developed through discrete projections have, to the best of the authors' knowledge, not been investigated.

Another approach that displays similarities to the variational multiscale method is the Mori-Zwanzig (MZ) formalism. Originally developed by Mori [38] and Zwanzig [39] and reformulated by Chorin and co-workers [40, 41, 42, 43], the MZ formalism is a type of model order reduction framework. The framework consists of decomposing the state variables in a dynamical system into a resolved (coarse-scale) set and an unresolved (fine-scale) set. An exact reduced-order model for the resolved scales is then derived in which the impact of the unresolved scales on the resolved scales appears as a memory term. This memory term depends on the temporal history of the resolved variables. In practice, the evaluation of this memory term is not tractable. It does, however, serve as a starting point to develop closure models. As MZ is formulated systematically in a dynamical system setting, it promises to be an effective technique for developing stable and accurate ROMs of non-linear dynamical systems. A range of research examining the MZ formalism as a multiscale modeling tool exists in the community. Most notably, Stinis and co-workers [44, 45, 46, 47, 48, 49] have developed several models for approximating the memory, including finite memory and renormalized models, and examined their application to the semi-discrete systems emerging from Fourier-Galerkin and Polynomial Chaos Expansions of Burgers' equation and the Euler equations. Application of MZ-based techniques to the classic POD-ROM approach has not been undertaken.

This manuscript leverages work that the authors have performed on the use of the MZ formalism to develop closure models of partial differential equations [50, 51, 52, 53, 54]. In addition to focusing on the development and analysis of MZ models, the authors have examined the formulation of the MZ formalism within the context of the VMS method [54]. By expressing MZ models within a VMS framework, similarities were discovered between MZ and VMS models. In particular, it was discovered that several existing MZ models are residual-based methods.

The contributions of this work include:

1. The development of a novel projection-based reduced-order modeling technique, termed the Adjoint Petrov–Galerkin (APG) method. The method leads to a ROM equation that is driven by the residual of the discretized governing equations. The approach is equivalent to a Petrov–Galerkin ROM and displays similarities to the LSPG approach. The method can be evolved in time with explicit integrators (in contrast to LSPG). This potentially lowers the cost of the ROM.
2. Theoretical error analysis examining conditions under which the *a priori* error bounds in APG may be smaller than in the Galerkin method.
3. Computational cost analysis (in FLOPS) of the proposed APG method as compared to the Galerkin and LSPG methods. This analysis shows that the APG ROM is twice as expensive as the G ROM for a given time step, for both explicit and implicit time integrators. In the implicit case, the ability of

the APG ROM to make use of Jacobian-Free Newton-Krylov methods suggests that it may be more efficient than the LSPG ROM.

4. Numerical evidence on ROMs of compressible flow problems demonstrating that the proposed method is more accurate and stable than the G ROM on problems of interest. Improvements over the LSPG ROM are observed in most cases. An analysis of the computational cost shows that the APG method can lead to lower errors than the LSPG and G ROMs for the same computational cost.
5. Theoretical results and numerical evidence that provides a relationship between the time-scale in the APG ROM and the spectral radius of the right-hand side Jacobian. Numerical evidence suggests that this relationship also applies to the selection of the optimal time-step in LSPG.

The structure of this paper is as follows: Section 2 outlines the full-order model of interest and its formulation in generalized coordinates. Section 3 outlines the reduced-order modeling approach applied at the semi-discrete level. Galerkin, Petrov–Galerkin, and VMS ROMs will be discussed. Section 3.3 details the Mori-Zwanzig formalism and the construction of the Adjoint Petrov–Galerkin ROM. Section 4 provides theoretical error analysis. Section 5 discusses the implementation and computational cost of the Adjoint Petrov–Galerkin method. Numerical results and comparisons with Galerkin and LSPG ROMs are presented in Section 6. Conclusions are provided in Section 7.

Mathematical notation in this manuscript is as follows: matrices are written as bold uppercase letters (e.g. $\mathbf{V}$), vectors as lowercase bold letters (e.g. $\mathbf{u}$), and scalars as italicized lowercase letters (e.g. a_i). Calligraphic script may denote vector spaces or special operators (e.g. $\mathcal{V}$, $\mathcal{L}$). Bold letters followed by parentheses indicate a matrix or vector function (e.g. $\mathbf{R}(\cdot)$, $\mathbf{u}(\cdot)$), and those followed by brackets indicate a linearization about the bracketed argument (e.g. $\mathbf{J}[\cdot]$).

2. Full-Order Model and Generalized Coordinates

Consider a full-order model that is described by the dynamical system,

$$\frac{d}{dt}\mathbf{u}(t) = \mathbf{R}(\mathbf{u}(t)), \qquad \mathbf{u}(0) = \mathbf{u}_0, \qquad t \in [0,T], \tag{1}$$

where $T \in \mathbb{R}^+$ denotes the final time, $\mathbf{u} : [0,T] \to \mathbb{R}^N$ denotes the state, and $\mathbf{u}_0 \in \mathbb{R}^N$ the initial conditions. The function $\mathbf{R} : \mathbb{R}^N \to \mathbb{R}^N$ with $\mathbf{y} \mapsto \mathbf{R}(\mathbf{y})$ is a (possibly non-linear) function and will be referred to as the "right-hand side" operator. Equation 1 arises in many disciplines, including the numerical discretization of partial differential equations. In this context, $\mathbf{R}(\cdot)$ may represent a spatial discretization scheme with source terms and applicable boundary conditions.

In many practical applications, the computational cost associated with solving Eq. 1 is prohibitively expensive due to the high dimension of the state. The goal of a ROM is to transform the N-dimensional dynamical system presented in Eq. 1 into a K dimensional dynamical system, with $K \ll N$. To achieve this goal, we pursue the following agenda:

1. Develop a weak form of the FOM in generalized coordinates.
2. Decompose the generalized coordinates into a K-dimensional resolved coarse-scale set and an $N-K$ dimensional unresolved fine-scale set.
3. Develop a K-dimensional ROM for the coarse-scales by making approximations to the fine-scale coordinates.

The remainder of this section will address task 1 in the above agenda.

To develop the weak form of Eq. 1, we start by defining a trial basis matrix comprising N orthonormal basis vectors,

$$\mathbf{V} \equiv \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \end{bmatrix},$$

where $\mathbf{v}_i \in \mathbb{R}^N$, $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$. The basis vectors may be generated, for example, by the POD approach. We next define the *trial space* as the range of the trial basis matrix,

$$\mathcal{V} := \text{Range}(\mathbf{V}).$$

As $\mathbf{V}$ is a full-rank $N \times N$ matrix, $\mathcal{V} \equiv \mathbb{R}^N$ and the state variable can be exactly described by a linear combination of these basis vectors,

$$\mathbf{u}(t) = \sum_{i=1}^{N} \mathbf{v}_i a_i(t). \tag{2}$$

Following [22], we collect the basis coefficients $a_i(t)$ into $\mathbf{a} : [0,T] \to \mathbb{R}^N$ and refer to $\mathbf{a}$ as the *generalized coordinates*. We similarly define the test basis matrix, $\mathbf{W}$, whose columns comprise linearly independent basis vectors that span the *test space*, $\mathcal{W}$,

$$\mathbf{W} \equiv \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_N \end{bmatrix}, \qquad \mathcal{W} := \text{Range}(\mathbf{W}),$$

with $\mathbf{w}_i \in \mathbb{R}^N$.

Equation 1 can be expressed in terms of the generalized coordinates by inserting Eq. 2 into Eq. 1,

$$\mathbf{V}\frac{d}{dt}\mathbf{a}(t) = \mathbf{R}(\mathbf{V}\mathbf{a}(t)). \tag{3}$$

The weak form of Eq. 3 is obtained by taking the L^2 inner product with $\mathbf{W}$,[2]

$$\mathbf{W}^T\mathbf{V}\frac{d}{dt}\mathbf{a}(t) = \mathbf{W}^T\mathbf{R}(\mathbf{V}\mathbf{a}(t)). \tag{4}$$

Manipulation of Eq. 4 yields the following dynamical system,

$$\frac{d}{dt}\mathbf{a}(t) = [\mathbf{W}^T\mathbf{V}]^{-1}\mathbf{W}^T\mathbf{R}(\mathbf{V}\mathbf{a}(t)), \qquad \mathbf{a}(t=0) = \mathbf{a}_0, \qquad t \in [0,T], \tag{5}$$

where $\mathbf{a}_0 \in \mathbb{R}^N$ with $\mathbf{a}_0 = [\mathbf{W}^T\mathbf{V}]^{-1}\mathbf{W}^T\mathbf{u}_0$. Note that Eq. 5 is an N-dimensional ODE system and is simply Eq. 1 expressed in a different coordinate system. It is further worth noting that, since $\mathbf{W}$ and $\mathbf{V}$ are invertible (both are square matrices with linearly independent columns), one has $[\mathbf{W}^T\mathbf{V}]^{-1}\mathbf{W}^T = \mathbf{V}^{-1}$. This will not be the case for ROMs.

3. Reduced-Order Models

3.1. Multiscale Formulation

This subsection addresses task 2 in the aforementioned agenda. Reduced-order models seek a low-dimensional representation of the original high-fidelity model. To achieve this, we examine a multiscale formulation of Eq. 5. Consider sum decompositions of the trial and test space,

$$\mathcal{V} = \tilde{\mathcal{V}} \oplus \mathcal{V}', \qquad \mathcal{W} = \tilde{\mathcal{W}} \oplus \mathcal{W}'. \tag{6}$$

The space $\tilde{\mathcal{V}}$ is referred to as the coarse-scale trial space, while $\mathcal{V}'$ is referred to as the fine-scale trial space. We refer to $\tilde{\mathcal{W}}$ and $\mathcal{W}'$ in a similar fashion. For simplicity, define $\tilde{\mathcal{V}}$ to be the column space of the first K basis vectors in $\mathbf{V}$ and $\mathcal{V}'$ to be the column space of the last $N-K$ basis vectors in $\mathbf{V}$. This approach is appropriate when the basis vectors are ordered in a hierarchical manner, as is the case with POD. Note the following properties of the decomposition:

[2]The authors recognize that many types of inner products are possible in formulating a ROM. To avoid unnecessary abstraction, we focus here on the simplest case.

1. The coarse-scale space is a subspace of $\mathcal{V}$, i.e., $\tilde{\mathcal{V}} \subset \mathcal{V}$.
2. The fine-scale space is a subspace of $\mathcal{V}$, i.e., $\mathcal{V}' \subset \mathcal{V}$.
3. The fine and coarse-scale subspaces do not overlap, i.e., $\tilde{\mathcal{V}} \cap \mathcal{V}' = \{0\}$.
4. The fine-scale and coarse-scale subspaces are orthogonal, i.e., $\tilde{\mathcal{V}} \perp \mathcal{V}'$. This is due to the fact that the basis vectors that comprise $\mathbf{V}$ are orthonormal.

For notational purposes, we make the following definitions for the trial and test spaces:

$$\mathbf{V} \equiv \left[\tilde{\mathbf{V}} \quad ; \quad \mathbf{V}'\right], \quad \mathbf{W} \equiv \left[\tilde{\mathbf{W}} \quad ; \quad \mathbf{W}'\right],$$

where $[\cdot \ ; \ \cdot]$ denotes the concatenation of two matrices and,

$$\begin{aligned} \tilde{\mathbf{V}} &\equiv \left[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_K\right], & \tilde{\mathcal{V}} &:= \text{Range}(\tilde{\mathbf{V}}), \\ \mathbf{V}' &\equiv \left[\mathbf{v}_{K+1} \quad \mathbf{v}_{K+2} \quad \cdots \quad \mathbf{v}_N\right], & \mathcal{V}' &:= \text{Range}(\mathbf{V}'). \\ \tilde{\mathbf{W}} &\equiv \left[\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_K\right], & \tilde{\mathcal{W}} &:= \text{Range}(\tilde{\mathbf{W}}), \\ \mathbf{W}' &\equiv \left[\mathbf{w}_{K+1} \quad \mathbf{w}_{K+2} \quad \cdots \quad \mathbf{w}_N\right], & \mathcal{W}' &:= \text{Range}(\mathbf{W}'). \end{aligned}$$

The coarse and fine-scale states are defined as,

$$\tilde{\mathbf{u}}(t) := \sum_{i=1}^{K} \mathbf{v}_i a_i(t) \equiv \tilde{\mathbf{V}}\tilde{\mathbf{a}}(t), \qquad \mathbf{u}'(t) := \sum_{i=K+1}^{N} \mathbf{v}_i a_i(t) \equiv \mathbf{V}'\mathbf{a}'(t),$$

with $\tilde{\mathbf{u}} : [0,T] \to \tilde{\mathcal{V}}$, $\mathbf{u}' : [0,T] \to \mathcal{V}'$, $\tilde{\mathbf{a}} : [0,T] \to \mathbb{R}^K$, and $\mathbf{a}' : [0,T] \to \mathbb{R}^{N-K}$. These decompositions allow Eq. 4 to be expressed as two linearly independent systems,

$$\tilde{\mathbf{W}}^T \tilde{\mathbf{V}} \frac{d}{dt}\tilde{\mathbf{a}}(t) + \tilde{\mathbf{W}}^T \mathbf{V}' \frac{d}{dt}\mathbf{a}'(t) = \tilde{\mathbf{W}}^T \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}(t) + \mathbf{V}'\mathbf{a}'(t)), \tag{7}$$

$$\mathbf{W}'^T \tilde{\mathbf{V}} \frac{d}{dt}\tilde{\mathbf{a}}(t) + \mathbf{W}'^T \mathbf{V}' \frac{d}{dt}\mathbf{a}'(t) = \mathbf{W}'^T \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}(t) + \mathbf{V}'\mathbf{a}'(t)). \tag{8}$$

Equation 7 is referred to as the coarse-scale equation, while Eq. 8 is referred to as the fine-scale equation. It is important to emphasize that the system formed by Eqs. 7 and 8 is still an exact representation of the original FOM.

The objective of ROMs is to solve the coarse-scale equation. The challenge encountered in this objective is that the evolution of the coarse-scales depends on the fine-scales. This is a type of "closure problem" and must be addressed to develop a closed ROM.

3.2. Reduced-Order Models

As noted above, the objective of a ROM is to solve the (unclosed) coarse-scale equation. We now develop ROMs of Eq. 1 by leveraging the multiscale decomposition presented above. This section addresses task 3 in the mathematical agenda.

The most straightforward technique to develop a ROM is to make the approximation,

$$\mathbf{u}' \approx \mathbf{0}.$$

This allows for the coarse-scale equation to be expressed as,

$$\tilde{\mathbf{W}}^T \tilde{\mathbf{V}} \frac{d}{dt}\tilde{\mathbf{a}}(t) = \tilde{\mathbf{W}}^T \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}(t)). \tag{9}$$

Equation 9 forms a K-dimensional reduced-order system (with $K \ll N$) and provides the starting point for formulating several standard ROM techniques. The Galerkin and Least-Squares Petrov–Galerkin ROMs are outlined in the subsequent subsections.

3.2.1. *The Galerkin Reduced-Order Model*

Galerkin projection is a common choice for producing a reduced set of ODEs. In Galerkin projection, the test basis is taken to be equivalent to the trial basis, i.e. $\tilde{\mathbf{W}} = \tilde{\mathbf{V}}$. The Galerkin ROM is then,

$$\tilde{\mathbf{V}}^T \frac{d}{dt}\tilde{\mathbf{u}}(t) = \tilde{\mathbf{V}}^T \mathbf{R}(\tilde{\mathbf{u}}(t)), \qquad \tilde{\mathbf{u}}(0) = \tilde{\mathbf{u}}_0, \qquad t \in [0,T]. \tag{10}$$

Galerkin projection can be shown to be optimal in the sense that it minimizes the L^2-norm of the FOM ODE residual over Range($\tilde{\mathbf{V}}$) [22]. As the columns of $\tilde{\mathbf{V}}$ no longer spans $\mathcal{V}$, it is possible that the initial state of the full system, $\mathbf{u}_0$, may differ from the initial state of the reduced system, $\tilde{\mathbf{u}}_0$. For simplicity, however, it is assumed here that the initial conditions lie fully in the coarse-scale trial space, i.e. $\mathbf{u}_0 \in \tilde{\mathcal{V}}$ such that,

$$\tilde{\mathbf{u}}_0 = \mathbf{u}_0. \tag{11}$$

Note that this issue can be formally addressed by using an affine trial space to ensure that $\tilde{\mathbf{u}}_0 = \mathbf{u}_0$.

Equation 10 can be equivalently written for the generalized coarse-scale coordinates, $\tilde{\mathbf{a}}$,

$$\frac{d}{dt}\tilde{\mathbf{a}}(t) = \tilde{\mathbf{V}}^T \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}(t)), \qquad \tilde{\mathbf{a}}(0) = \tilde{\mathbf{a}}_0, \qquad t \in [0,T], \tag{12}$$

where $\tilde{\mathbf{a}}_0 \in \mathbb{R}^K$ with $\tilde{\mathbf{a}}_0 = \tilde{\mathbf{V}}^T\mathbf{u}_0$. Equation 12 is a K-dimensional ODE system (with $K \ll N$) and is hence of lower dimension than the FOM. Note that, similar to Eq. 5, the projection via $\tilde{\mathbf{V}}^T$ would normally be $\left[\tilde{\mathbf{V}}^T\tilde{\mathbf{V}}\right]^{-1}\tilde{\mathbf{V}}$. When $\tilde{\mathbf{V}}$ is constructed via POD, its columns are orthonormal and $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \mathbf{I}$; the projector has been simplified to reflect this. Non-orthonormal basis vectors will require the full computation of $\left[\tilde{\mathbf{V}}^T\tilde{\mathbf{V}}\right]^{-1}\tilde{\mathbf{V}}$.

It is important to note that, in order to develop a computationally efficient ROM, some "hyper-reduction" method must be devised to reduce the cost associated with evaluating the matrix-vector product, $\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}(t))$. Gappy POD [55] and the (discrete) empirical interpolation method [56, 57] are two such techniques. More details on hyper-reduction are provided in Appendix A.

When applied to unsteady non-linear problems, the Galerkin ROM is often inaccurate and, at times, unstable. Examples of this are seen in Ref. [23]. These issues motivate the development of more sophisticated reduced-order modeling techniques.

3.2.2. *Petrov–Galerkin and Least-Squares Petrov–Galerkin Reduced-Order Models*

In the Petrov–Galerkin approach, the test space is different from the trial space. Petrov–Galerkin approaches have a rich history in the finite element community [58, 59] and can enhance the stability and robustness of a numerical method. In the context of reduced-order modeling for dynamical systems, the Least-Squares Petrov–Galerkin method (LSPG) [22] is a popular approach. The LSPG approach is a ROM technique that seeks to minimize the fully discrete residual (i.e., the residual after spatial and temporal discretization) at each time-step. The LSPG method can be shown to be optimal in the sense that it minimizes the L^2-norm of the *fully discrete* residual at each time-step over Range($\tilde{\mathcal{V}}$). To illustrate the LSPG method, consider the algebraic system of equations for the FOM obtained after an implicit Euler temporal discretization,

$$\frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\Delta t} - \mathbf{R}(\mathbf{u}^n) = \mathbf{0}, \tag{13}$$

where $\mathbf{u}^n \in \mathbb{R}^N$ denotes the solution at the n^{th} time-step. The FOM will exactly satisfy Eq. 13. The ROM, however, will not. The LSPG method minimizes the residual of Eq. 13 over each time-step. For notational purposes, we define the residual vector for the implicit Euler method,

$$\mathbf{r}_{\mathrm{IE}} : (\mathbf{y};\mathbf{u}^{n-1}) \mapsto \frac{\mathbf{y} - \mathbf{u}^{n-1}}{\Delta t} - \mathbf{R}(\mathbf{y}).$$

The LSPG method is defined as follows,

$$\mathbf{u}^n = \underset{\mathbf{y}\in\text{Range}(\tilde{\mathbf{V}})}{\arg\min} \; ||\mathbf{A}(\mathbf{y})\mathbf{r}_{\text{IE}}(\mathbf{y};\mathbf{u}^{n-1})||_2^2,$$

where $\mathbf{A}(\cdot) \in \mathbb{R}^{z\times n}$ with $z \leq N$ is a weighting matrix. The standard LSPG method takes $\mathbf{A} = \mathbf{I}$. For the implicit Euler time integration scheme (as well as various other implicit schemes) the LSPG approach can be shown to have an equivalent continuous representation using a Petrov–Galerkin projection [23]. For example, the LSPG method for any backward differentiation formula (BDF) time integration scheme can be written as a Petrov–Galerkin ROM with the test basis,

$$\tilde{\mathbf{W}} = \big(\mathbf{I} - \alpha\Delta t \mathbf{J}[\tilde{\mathbf{u}}(t)]\big)\tilde{\mathbf{V}},$$

where $\mathbf{J}[\tilde{\mathbf{u}}(t)] = \frac{\partial \mathbf{R}}{\partial \mathbf{y}}(\tilde{\mathbf{u}}(t))$ is the Jacobian of the right-hand side function evaluated about the coarse-scale state and α is a constant, specific to a given scheme (e.g. $\alpha = 1$ for implicit Euler, $\alpha = \frac{2}{3}$ for BDF2, $\alpha = \frac{6}{11}$ for BDF3, etc.). With this test basis, the LSPG ROM can be written as,

$$\tilde{\mathbf{V}}^T\left(\frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right) = \tilde{\mathbf{V}}^T\mathbf{J}^T[\tilde{\mathbf{u}}(t)]\alpha\Delta t\left(\frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right), \qquad \tilde{\mathbf{u}}(0) = \tilde{\mathbf{u}}_0, \qquad t \in [0,T]. \tag{14}$$

In writing Eq. 14, we have coupled all of the terms from the standard Galerkin ROM on the left-hand side, and have similarly coupled the terms introduced by the Petrov–Galerkin projection on the right-hand side. One immediately observes that the LSPG approach is a residual-based method, meaning that the stabilization added by LSPG is proportional to the residual. The LSPG method is similar to the Galerkin/Least-Squares (GLS) approach commonly employed in the finite element community [60, 61]. This can be made apparent by writing Eq. 14 as,

$$\left(\mathbf{v}_i, \frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right) = \left(\mathbf{J}[\tilde{\mathbf{u}}(t)]\mathbf{v}_i, \tau\Big[\frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\Big]\right), \qquad i = 1,2,\dots,K,$$

where $(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}$ and $\tau = \alpha\Delta t$ is the stabilization parameter. Compare the above to, say, Eq. 70 and 71 in Ref. [61]. A rich body of literature exists on residual-based methods, and viewing the LSPG approach in this light helps establish connections with other methods. We highlight several important aspects of LSPG. Remarks 1 through 3 are derived by Carlberg et al. in Ref. [23]:

1. The LSPG approach is inherently tied to the temporal discretization. For different time integration schemes, the "stabilization" added by the LSPG method will vary. For optimal accuracy, the LSPG method requires an intermediary time-step size.
2. In the limit of $\Delta t \to 0$, the LSPG approach recovers a Galerkin approach.
3. For explicit time integration schemes, the LSPG and Galerkin approach are equivalent.
4. For backwards differentiation schemes, the LSPG approach is a type of GLS stabilization for non-linear problems.
5. While commonalities exist between LSPG and multiscale approaches, the authors believe that the LSPG method should *not* be viewed as a subgrid-scale model. The reason for this is that it is unclear how Eq. 14 can be derived from Eq. 7. This is similar to the fact that, in Ref. [61], *adjoint* stabilization is viewed as a subgrid-scale model while GLS stabilization is not. The challenge in deriving Eq. 14 from Eq. 7 lies primarily in the fact that the Jacobian in Eq. 14 contains a transpose operator. We thus view LSPG as mathematical stabilization rather than a subgrid-scale model.

While the LSPG approach has enjoyed much success for constructing ROMs of non-linear problems, remarks 1, 2, 3, and 5 suggest that improvements over the LSPG method are possible. Remark 1 suggests improvements in computational speed and accuracy are possible by removing sensitivity to the time-step size. Remark 3 suggests that improvements in computational speed and flexibility are possible by formulating a method that can be used with explicit time-stepping schemes. Lastly, remark 5 suggests that improvements in accuracy are possible by formulating a method that accounts for subgrid effects.

3.3. *Mori-Zwanzig Reduced-Order Models*

The optimal prediction framework formulated by Chorin et al. [40, 41, 42], which is a (significant) reformulation of the Mori-Zwanzig (MZ) formalism of statistical mechanics, is a model order reduction tool that can be used to develop representations of the impact of the fine-scales on the coarse-scale dynamics. In this section, the optimal prediction framework is used to derive a compact approximation to the impact of the fine-scale POD modes on the evolution of the coarse-scale POD modes. For completeness, the optimal prediction framework is first derived in the context of the Galerkin POD ROM. It is emphasized that the content presented in Sections 3.3.1 and 3.3.2 is simply a formulation of Chorin's framework, with a specific projection operator, in the context of the Galerkin POD ROM.

We pursue the MZ approach on a Galerkin formulation of Eq. 5. Before describing the formalism, it is beneficial to re-write the original FOM in terms of the generalized coordinates with the solution being defined implicitly as a function of the initial conditions,

$$\frac{d}{dt}\mathbf{a}(\mathbf{a}_0,t) = \mathbf{V}^T\mathbf{R}(\mathbf{V}\mathbf{a}(\mathbf{a}_0,t)), \qquad \mathbf{a}(0) = \mathbf{a}_0, \qquad t \in [0,T], \tag{15}$$

with $\mathbf{a} : \mathbb{R}^N \times [0,T] \to \mathbb{R}^N$, $\mathbf{a} \in \mathbb{R}^N \otimes \mathcal{R}^N \otimes \mathcal{T}$ the time-dependent generalized coordinates, $\mathcal{R}^N$ the space of (sufficiently smooth) functions acting on $\mathbb{R}^N$, $\mathcal{T}$ the space of (sufficiently smooth) functions acting on $[0,T]$, and $\mathbf{a}_0 \in \mathbb{R}^N$ the initial conditions. Here, $\mathbf{a}(\mathbf{a}_0,t)$ is viewed as a function that maps from the coordinates $\mathbf{a}_0$ (i.e., the initial conditions) and time to a vector in $\mathbb{R}^N$. It is assumed that the right-hand side operator $\mathbf{R}$ is continuously differentiable on $\mathbb{R}^N$.

3.3.1. *The Liouville Equation*

The starting point of the MZ approach is to transform the non-linear FOM (Eq. 15) into a linear partial differential equation. Equation 15 can be written equivalently as the following partial differential equation in $\mathbb{R}^N \times [0,T]$ [41],

$$\frac{\partial}{\partial t}v(\mathbf{a}_0,t) = \mathcal{L}v(\mathbf{a}_0,t); \qquad v(\mathbf{a_0},0) = g(\mathbf{a}_0), \tag{16}$$

where $v : \mathbb{R}^N \times [0,T] \to \mathbb{R}^{N_v}$ with $v \in \mathbb{R}^{N_v} \otimes \mathcal{R}^N \otimes \mathcal{T}$ is a set of N_v observables and $g : \mathbb{R}^N \to \mathbb{R}^{N_v}$ is a state-to-observable map. The operator $\mathcal{L}$ is the Liouville operator, also known as the Lie derivative, and is defined by,

$$\begin{aligned} \mathcal{L} : \mathbf{q} &\mapsto \left[\frac{\partial}{\partial \mathbf{a_0}}\mathbf{q}\right]\mathbf{V}^T\mathbf{R}(\mathbf{V}\mathbf{a_0}), \\ &: \mathbb{R}^q \otimes \mathcal{R}^N \to \mathbb{R}^q \otimes \mathcal{R}^N, \end{aligned}$$

for arbitrary q. Equation 16 is referred to as the Liouville equation and is an exact statement of the original dynamics. The Liouville equation describes the solution to Eq. 15 for *all* possible initial conditions. The advantage of reformulating the system in this way is that the Liouville equation is linear, allowing for the use of superposition and aiding in the removal of the fine-scales.

The solution to Eq. 16 can be written as,

$$v(\mathbf{a_0},t) = e^{t\mathcal{L}}g(\mathbf{a_0}).$$

The operator $e^{t\mathcal{L}}$, which has been referred to as a "propagator", evolves the solution along its trajectory in phase-space [62]. The operator $e^{t\mathcal{L}}$ has several interesting properties. Most notably, the operator can be "pulled" inside of a non-linear functional [62],

$$e^{t\mathcal{L}}g(\mathbf{a}_0) = g(e^{t\mathcal{L}}\mathbf{a}_0).$$

This is similar to the composition property inherent to Koopman operators [63]. With this property, the solution to Eq. 16 may be written as,

$$v(\mathbf{a_0},t) = g(e^{t\mathcal{L}}\mathbf{a}_0).$$

The implications of $e^{t\mathcal{L}}$ are significant. It demonstrates that, given trajectories $\mathbf{a}(\mathbf{a_0},t)$, the solution v is known for any observable g.

Noting that $\mathcal{L}$ and $e^{t\mathcal{L}}$ commute, Eq. 16 may be written as,

$$\frac{\partial}{\partial t}v(\mathbf{a_0},t) = e^{t\mathcal{L}}\mathcal{L}v(\mathbf{a_0},0).$$

A set of partial differential equations for the resolved generalized coordinates can be obtained by taking $g(\mathbf{a_0}) = \tilde{\mathbf{a}}_0$,

$$\frac{\partial}{\partial t}e^{t\mathcal{L}}\tilde{\mathbf{a}}_0 = e^{t\mathcal{L}}\mathcal{L}\tilde{\mathbf{a}}_0. \tag{17}$$

The remainder of the derivation is performed for $g(\mathbf{a_0}) = \tilde{\mathbf{a}}_0$, thus $N_v = K$.

3.3.2. Projection Operators and the Generalized Langevin Equation

The objective now is to remove the dependence of Eq. 17 on the fine-scale variables. Similar to the VMS decomposition, $\mathcal{R}^N$ can be decomposed into resolved and unresolved subspaces,

$$\mathcal{R}^N = \tilde{\mathcal{H}} \oplus \mathcal{H}',$$

with $\tilde{\mathcal{H}}$ being the space of all functions of the resolved coordinates, $\tilde{\mathbf{a}}_0$, and $\mathcal{H}'$ the complementary space. The associated projection operators are defined as $\mathcal{P} : \mathcal{R}^N \rightarrow \tilde{\mathcal{H}}$ and $\mathcal{Q} = I - \mathcal{P}$. Various types of projections are possible, and here we consider,

$$\mathcal{P}f(\mathbf{a}_0) = \int_{\mathbb{R}^N} f(\mathbf{a}_0)\delta(\mathbf{a'}_0)d\mathbf{a'}_0,$$

which leads to

$$\mathcal{P}f(\mathbf{a}_0) = f([\tilde{\mathbf{a}}_0;\mathbf{0}]).$$

The projection operators can be used to split the Liouville equation,

$$\frac{\partial}{\partial t}e^{t\mathcal{L}}\tilde{\mathbf{a}}_0 = e^{t\mathcal{L}}\mathcal{P}\mathcal{L}\tilde{\mathbf{a}}_0 + e^{t\mathcal{L}}\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0. \tag{18}$$

The objective now is to remove the dependence of the right-hand side of Eq. 18 on the fine-scales, $\mathbf{a}'_0$ (i.e. $\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0$). This may be achieved by Duhamel's principle,

$$e^{t\mathcal{L}} = e^{t\mathcal{Q}\mathcal{L}} + \int_0^t e^{(t-s)\mathcal{L}}\mathcal{P}\mathcal{L}e^{s\mathcal{Q}\mathcal{L}}ds. \tag{19}$$

Inserting Eq. 19 into Eq. 18, the generalized Langevin equation is obtained,

$$\frac{\partial}{\partial t}e^{t\mathcal{L}}\tilde{\mathbf{a}}_0 = \underbrace{e^{t\mathcal{L}}\mathcal{P}\mathcal{L}\tilde{\mathbf{a}}_0}_{\text{Markovian}} + \underbrace{e^{t\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0}_{\text{Noise}} + \underbrace{\int_0^t e^{(t-s)\mathcal{L}}\mathcal{P}\mathcal{L}e^{s\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0 ds}_{\text{Memory}}. \tag{20}$$

By the definition of the initial conditions (Eq. 11), the noise-term is zero and we obtain,

$$\frac{\partial}{\partial t}e^{t\mathcal{L}}\tilde{\mathbf{a}}_0 = e^{t\mathcal{L}}\mathcal{P}\mathcal{L}\tilde{\mathbf{a}}_0 + \int_0^t e^{(t-s)\mathcal{L}}\mathcal{P}\mathcal{L}e^{s\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0 ds. \tag{21}$$

The system described in Eq. 20 is precise and not an approximation to the original ODE system. For notational purposes, define,

$$\mathbf{K}(\tilde{\mathbf{a}}_0,t) \equiv \mathcal{P}\mathcal{L}e^{t\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\tilde{\mathbf{a}}_0. \tag{22}$$

The term $\mathbf{K} : \mathbb{R}^K \times [0,T] \to \mathbb{R}^K$ with $\mathbf{K} \in \mathbb{R}^K \otimes \tilde{\mathcal{H}} \otimes \mathcal{T}$ is referred to as the memory kernel.

Using the identity $e^{t\mathcal{L}}\mathcal{P}\mathcal{L}\tilde{\mathbf{a}}_0 = \tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}(t))$ and Definition (22), Equation 21 can be written in a more transparent form,

$$\tilde{\mathbf{V}}^T\left(\frac{\partial}{\partial t}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right) = \int_0^t \mathbf{K}(\tilde{\mathbf{a}}(t-s),s)ds, \tag{23}$$

Note that the time derivative is represented as a partial derivative due to the Liouville operators embedded in the memory.

The derivation up to this point has cast the original full-order model in generalized coordinates (Eq. 15) as a linear PDE. Through the use of projection operators and Duhamel's principle, an *exact* equation (Eq. 23) for the coarse-scale dynamics *only in terms of the coarse-scale variables* was then derived. The effect of the fine-scales on the coarse-scales appeared as a memory integral. This memory integral may be thought of as the closure term that is required to exactly account for the unresolved dynamics.

3.3.3. The τ-model and the Adjoint Petrov–Galerkin Method

The direct evaluation of the memory term in Eq. 23 is, in general, computationally intractable. To gain a reduction in computational cost, an approximation to the memory must be devised. A variety of such approximations exist, and here we outline the τ-model [52, 64]. The τ-model can be interpreted as the result of assuming that the memory is driven to zero in finite time and approximating the integral with a quadrature rule. This can be written as a two-step approximation,

$$\int_0^t \mathbf{K}(\tilde{\mathbf{a}}(t-s),s)ds \approx \int_{t-\tau}^t \mathbf{K}(\tilde{\mathbf{a}}(t-s),s)ds \approx \tau\mathbf{K}(\tilde{\mathbf{a}}(t),0).$$

Here, $\tau \in \mathbb{R}$ is a stabilization parameter that is sometimes referred to as the "memory length." It is typically static and user-defined, though methods of dynamically calculating it have been developed in [52]. The *a priori* selection of τ and sensitivity of the model output to this selection are discussed later in this manuscript.

The term $\mathbf{K}(\tilde{\mathbf{a}}(t),0)$ can be shown to be [54],

$$\mathbf{K}(\tilde{\mathbf{a}}(t),0) = \tilde{\mathbf{V}}^T\mathbf{J}[\tilde{\mathbf{u}}(t)]\Pi'\mathbf{R}(\tilde{\mathbf{u}}(t)),$$

where Π' is the "orthogonal projection operator," defined as $\Pi' \equiv \left(\mathbf{I} - \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\right)$. We define the corresponding coarse-scale projection operator as $\tilde{\Pi} \equiv \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T$. The coarse-scale equation with the τ-model reads,

$$\tilde{\mathbf{V}}^T\left(\frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right) = \tau\tilde{\mathbf{V}}^T\mathbf{J}[\tilde{\mathbf{u}}(t)]\Pi'\mathbf{R}(\tilde{\mathbf{u}}(t)). \tag{24}$$

Equation 24 provides a closed equation for the evolution of the coarse-scales. The left-hand side of Eq. 24 is the standard Galerkin ROM, and the right-hand side can be viewed as a subgrid-scale model.

When compared to existing methods, the inclusion of the τ-model leads to a method that is analogous to a non-linear formulation of the *adjoint* stabilization technique developed in the finite element community. The "adjoint" terminology arises from writing Eq. 24 in a Petrov–Galerkin form,

$$\left[\left(\mathbf{I} + \tau\Pi'^T\mathbf{J}^T[\tilde{\mathbf{u}}(t)]\right)\tilde{\mathbf{V}}\right]^T\left(\frac{d}{dt}\tilde{\mathbf{u}}(t) - \mathbf{R}(\tilde{\mathbf{u}}(t))\right) = \mathbf{0}. \tag{25}$$

It is seen that Eq. 25 involves taking the inner product of the coarse-scale ODE with a test-basis that contains the adjoint of the coarse-scale Jacobian. Unlike GLS stabilization, adjoint stabilization can be derived from the multiscale equations [61]. Due to the similarity of the proposed method with adjoint stabilization techniques, as well as the LSPG terminology, the complete ROM formulation will be referred to as the Adjoint Petrov–Galerkin (APG) method.

3.3.4. Comparison of APG and LSPG

The APG method displays similarities to LSPG. From Eq. 25, it is seen that the test basis for the APG ROM is given by,

$$\tilde{\mathbf{W}}_A = \left(\mathbf{I} + \tau \Pi'^T \mathbf{J}^T[\tilde{\mathbf{u}}]\right)\tilde{\mathbf{V}}. \tag{26}$$

Recall the LSPG test basis for backward differentiation schemes,

$$\tilde{\mathbf{W}}_{LSPG} = \left(\mathbf{I} - \alpha \Delta t \mathbf{J}[\tilde{\mathbf{u}}]\right)\tilde{\mathbf{V}}. \tag{27}$$

Comparing Eq. 27 to Eq. 26, we can draw several interesting comparisons between the LSPG and APG method. Both contain a time-scale: τ for APG and $\alpha\Delta t$ for LSPG. Both include Jacobians of the non-linear function $\mathbf{R}(\tilde{\mathbf{u}})$. The two methods differ in the presence of the orthogonal projection operator in APG, a transpose on the Jacobian, and a sign discrepancy on the Jacobian. These last two differences are consistent with the discrepancies between GLS and adjoint stabilization methods used in the finite element community. See, for instance, Eqs. 71 and 73 in Ref [61].

4. Analysis

This section presents theoretical analyses of the Adjoint Petrov–Galerkin method. Specifically, error and eigenvalue analyses are undertaken for linear time-invariant (LTI) systems. Section 4.1 derives *a priori* error bounds for the Galerkin and Adjoint Petrov–Galerkin ROMs. Conditions under which the APG ROM may be more accurate than the Galerkin ROM are discussed. Section 4.2 outlines the selection of the parameter τ that appears in APG.

4.1. A Priori Error Bounds

We now derive *a priori* error bounds for the Galerkin and Adjoint Petrov–Galerkin method for LTI systems. Define $\mathbf{u}_F$ to be the solution to the FOM, $\tilde{\mathbf{u}}_G$ to be the solution to the Galerkin ROM, and $\tilde{\mathbf{u}}_A$ the solution to the Adjoint Petrov–Galerkin ROM. The full-order solution, Galerkin ROM, and Adjoint Petrov–Galerkin ROMs obey the following dynamical systems,

$$\frac{d}{dt}\mathbf{u}_F(t) = \mathbf{R}(\mathbf{u}_F(t)), \qquad \mathbf{u}_F(0) = \mathbf{u}_0, \tag{28}$$

$$\frac{d}{dt}\tilde{\mathbf{u}}_G(t) = \mathbb{P}_G\mathbf{R}(\tilde{\mathbf{u}}_G(t)), \qquad \mathbf{u}_G(0) = \mathbf{u}_0, \tag{29}$$

$$\frac{d}{dt}\tilde{\mathbf{u}}_A(t) = \mathbb{P}_A\mathbf{R}(\tilde{\mathbf{u}}_A(t)), \qquad \mathbf{u}_A(0) = \mathbf{u}_0, \tag{30}$$

where the Galerkin and Adjoint Petrov–Galerkin projections are, respectively,

$$\mathbb{P}_G = \tilde{\Pi}, \qquad \mathbb{P}_A = \tilde{\Pi}\left[\mathbf{I} + \tau\mathbf{J}[\tilde{\mathbf{u}}_A]\Pi'\right].$$

The residual of the full-order model is defined as,

$$\mathbf{r}_F : \mathbf{u} \mapsto \frac{d\mathbf{u}}{dt} - \mathbf{R}(\mathbf{u}).$$

We define the error in the Galerkin and Adjoint Petrov–Galerkin method as,

$$\mathbf{e}_G := \mathbf{u}_F - \tilde{\mathbf{u}}_G, \qquad \mathbf{e}_A := \mathbf{u}_F - \tilde{\mathbf{u}}_A.$$

Similarly, the coarse-scale error is defined as,

$$\tilde{\mathbf{e}}_G := \tilde{\Pi}\mathbf{u}_F - \tilde{\mathbf{u}}_G, \qquad \tilde{\mathbf{e}}_A := \tilde{\Pi}\mathbf{u}_F - \tilde{\mathbf{u}}_A.$$

In what follows, we assume Lipschitz continuity of the right-hand side function: there exists a constant $\kappa > 0$ such that $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,

$$\|\mathbf{R}(\mathbf{x}) - \mathbf{R}(\mathbf{y})\|_2 \leq \kappa \|\mathbf{x} - \mathbf{y}\|_2 .$$

To simplify the analysis, the Adjoint Petrov–Galerkin projection is approximated to be stationary in time. Note that the Galerkin projection is stationary in time. For clarity, we suppress the temporal argument on the states when possible in the proofs.

Theorem 1. *A priori error bounds for the Galerkin and Adjoint Petrov–Galerkin ROMs are, respectively,*

$$\|\mathbf{e}_G(t)\|_2 \leq \int_0^t e^{\|\mathbf{P}_G\|_2 \kappa s} \left\| \left[\mathbf{I} - \mathbb{P}_G\right] \mathbf{R}(\mathbf{u}_F(t-s)) \right\|_2 ds. \tag{31}$$

$$\|\mathbf{e}_A(t)\|_2 \leq \int_0^t e^{\|\mathbf{P}_A\|_2 \kappa s} \left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F(t-s)) \right\|_2 ds. \tag{32}$$

Proof. We prove only Eq. 32 as Eq. 31 is obtained through the same arguments. Following [23], start by subtracting Eq. 30 from Eq. 28, and adding and subtracting $\mathbb{P}_A \mathbf{R}(\mathbf{u}_F)$,

$$\frac{d\mathbf{e}_A}{dt} = \mathbf{R}(\mathbf{u}_F) + \mathbb{P}_A \mathbf{R}(\mathbf{u}_F) - \mathbb{P}_A \mathbf{R}(\mathbf{u}_F) - \mathbb{P}_A \mathbf{R}(\tilde{\mathbf{u}}_G), \qquad \mathbf{e}_A(0) = \mathbf{0}.$$

Taking the L^2-norm,

$$\left\| \frac{d\mathbf{e}_A}{dt} \right\|_2 = \|\mathbf{R}(\mathbf{u}_F) + \mathbb{P}_A \mathbf{R}(\mathbf{u}_F) - \mathbb{P}_A \mathbf{R}(\mathbf{u}_F) - \mathbb{P}_A \mathbf{R}(\tilde{\mathbf{u}}_G)\|_2 .$$

Applying the triangle inequality,

$$\left\| \frac{d\mathbf{e}_A}{dt} \right\|_2 \leq \left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F) \right\|_2 + \left\| \mathbb{P}_A \left(\mathbf{R}\mathbf{u}_F) - \mathbf{R}(\tilde{\mathbf{u}}_G) \right) \right\|_2 .$$

Invoking the assumption of Lipschitz continuity,

$$\left\| \frac{d\mathbf{e}_A}{dt} \right\|_2 \leq \left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F) \right\|_2 + \|\mathbb{P}_A\|_2 \kappa \|\mathbf{e}_A\|_2 .$$

Noting that $\frac{d\|\mathbf{e}_A\|_2}{dt} \leq \left\| \frac{d\mathbf{e}_A}{dt} \right\|_2$ we have [3],

$$\frac{d\,\|\mathbf{e}_A\|_2}{dt} \leq \left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F) \right\|_2 + \|\mathbb{P}_A\|_2 \kappa \|\mathbf{e}_A\|_2 . \tag{33}$$

An upper bound on the error for the Adjoint Petrov–Galerkin method is then obtained by solving Eq. 33 for $\|\mathbf{e}_A\|_2$, which yields,

$$\|\mathbf{e}_A(t)\|_2 \leq \int_0^t e^{\|\mathbb{P}_A\|_2 \kappa s} \left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F(t-s)) \right\|_2 ds.$$

□

[3]

$$\frac{d\,\|\mathbf{e}_A\|_2}{dt} = \frac{1}{\|\mathbf{e}_A\|_2} \mathbf{e}_A^T \frac{d\mathbf{e}_A}{dt} \leq \left\| \frac{1}{\|\mathbf{e}_A\|_2} \mathbf{e}_A \right\|_2 \left\| \frac{d\mathbf{e}_A}{dt} \right\|_2 \leq \left\| \frac{d\mathbf{e}_A}{dt} \right\|_2$$

Note that, for the Adjoint Petrov–Galerkin method, the error bound provided in Eq. 32 is not truly an *a priori* bound as $\mathbb{P}_A$ is a function of $\tilde{\mathbf{u}}_A$. Equation 31 (and 32) indicates an exponentially growing error and contains two distinct terms. The term $e^{\|\mathbb{P}_A\|_2 \kappa s}$ indicates the exponential growth of the error in time. The second term of interest is $\left\| \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{R}(\mathbf{u}_F(t)) \right\|_2$. This term corresponds to the error introduced at time t due to projection. It is important to note that the first term controls how the error will grow in time, while the second term controls how much error is added at a given time.

Unfortunately, for general non-linear systems, *a priori* error analysis provides minimal insight beyond what was just mentioned. To obtain a more intuitive understanding of the APG method, error analysis in the case that $\mathbf{R}(\cdot)$ is a linear time-invariant operator is now considered.

Theorem 2. *Let $\mathbf{R}(\mathbf{u}) = \mathbf{A}\mathbf{u}$ be a linear time-invariant operator. Error bounds for the coarse-scales in the Galerkin and APG ROMs, are, respectively,*

$$\|\tilde{\mathbf{e}}_G(t)\|_2 \leq \|\mathbf{S}_G\|_2 \left\|\mathbf{S}_G^{-1}\right\|_2 \int_0^t \left\| e^{\Lambda_G (t-s)} \right\|_2 \left\| \tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F(s)) \right\|_2 ds, \tag{34}$$

$$\|\tilde{\mathbf{e}}_A(t)\|_2 \leq \|\mathbf{S}_A\|_2 \left\|\mathbf{S}_A^{-1}\right\|_2 \int_0^t \left\| e^{\Lambda_A (t-s)} \right\|_2 \left\| \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{r}_F(\tilde{\mathbf{u}}_F(s)) \right\|_2 ds, \tag{35}$$

where Λ_G is a diagonal matrix of the eigenvalues of $\tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{A} \tilde{\mathbf{V}}$, while $\mathbf{S}_G$ are the eigenvectors of $\tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{A} \tilde{\mathbf{V}}$. Similar definitions hold for Λ_A and $\mathbf{S}_A$ using the APG projection.

Proof. We prove only Eq. 35 as Eq. 34 is obtained through the same arguments. Subtracting Eq. 30 from Eq. 28, and adding and subtracting $\mathbb{P}_A \mathbf{A}\mathbf{u}_F$,

$$\frac{d\mathbf{e}_A}{dt} = \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{A}\mathbf{u}_F + \mathbb{P}_A \mathbf{A} \mathbf{e}_A, \qquad \mathbf{e}_A(0) = \mathbf{0}.$$

We decompose the error into coarse and fine-scale components $\mathbf{e}_A = \tilde{\mathbf{e}}_A + \mathbf{e}'_A$. Since $\mathbf{u}'_A = \mathbf{0}$ for all time, $\mathbf{e}'_A = \mathbf{u}'_F$. An equation for the coarse-scale error is obtained by left multiplying by $\tilde{\mathbf{V}}^T \mathbb{P}_A$,

$$\tilde{\mathbf{V}}^T \mathbb{P}_A \frac{d\tilde{\mathbf{e}}_A}{dt} + \tilde{\mathbf{V}}^T \mathbb{P}_A \frac{d\mathbf{e}'_A}{dt} = \tilde{\mathbf{V}}^T \mathbb{P}_A \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{A}\mathbf{u}_F + \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbb{P}_A \mathbf{A} \tilde{\mathbf{e}}_A + \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbb{P}_A \mathbf{A} \mathbf{u}'_F.$$

Now express the coarse-scale error as,

$$\tilde{\mathbf{e}}_A = \tilde{\mathbf{V}} \tilde{\mathbf{a}}_A^{\mathbf{e}},$$

where $\tilde{\mathbf{a}}_A^{\mathbf{e}} : [0,T] \rightarrow \mathbb{R}^K$ are the coarse-scale error generalized coordinates. Rearranging gives,

$$\frac{d\tilde{\mathbf{a}}_A^{\mathbf{e}}}{dt} = \tilde{\mathbf{V}}^T \mathbb{P}_A \left[\mathbf{I} - \mathbb{P}_A\right] \mathbf{A}\mathbf{u}_F + \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbb{P}_A \mathbf{A} \tilde{\mathbf{e}}_A + \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbb{P}_A \left(\mathbf{A}\mathbf{u}'_F - \frac{d\mathbf{u}'_F}{dt} \right).$$

Noting that $\mathbb{P}_A^2 = \mathbb{P}_A$, and that the first term on the right-hand side is zero, the equation reduces to,

$$\frac{d\tilde{\mathbf{a}}_A^{\mathbf{e}}}{dt} = \tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}} \tilde{\mathbf{a}}_A^{\mathbf{e}} + \tilde{\mathbf{V}}^T \mathbb{P}_A \left(\mathbf{A}\mathbf{u}'_F - \frac{d\mathbf{u}'_F}{dt} \right). \tag{36}$$

Equation 36 is a first-order linear inhomogeneous differential equation that can be solved analytically. A standard way to do this is by performing the eigendecomposition,

$$\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}} = \mathbf{S}_\mathbf{A} \Lambda_A \mathbf{S}_\mathbf{A}^{-1}, \tag{37}$$

where $\mathbf{S}_A \in \mathbb{R}^{K \times K}$ are the eigenvectors and Λ_A is a diagonal matrix of eigenvalues. Left multiplying Eq. 36 by $\mathbf{S}_\mathbf{A}{}^{-1}$ and inserting the eigendecomposition 37, Eq. 36 becomes,

$$\frac{d\mathbf{w}_A}{dt} = \Lambda_A \mathbf{w}_A + \mathbf{S}_\mathbf{A}{}^{-1} \tilde{\mathbf{V}}^T \mathbb{P}_A \left(\mathbf{A}\mathbf{u}'_F - \frac{d\mathbf{u}'_F}{dt} \right), \tag{38}$$

where $\mathbf{w}_A = \mathbf{S}^{-1}\tilde{\mathbf{a}}^{\mathbf{e}}_A$. The above is a set of decoupled first order inhomogenous differential equation that have the solution,

$$\mathbf{w}_A(t) = \int_0^t e^{\Lambda_A s}\mathbf{S_A}^{-1}\tilde{\mathbf{V}}^T\mathbb{P}_A\left(\mathbf{A}\mathbf{u}'_F(t-s) - \frac{d}{dt}\mathbf{u}'_F(t-s)\right)ds.$$

Left multiplying by $\tilde{\mathbf{V}}\mathbf{S_A}$ to obtain an equation for the coarse-scale error,

$$\tilde{\mathbf{e}}_A(t) = \int_0^t \tilde{\mathbf{V}}\mathbf{S_A}e^{\Lambda_A(t-s)}\mathbf{S_A}^{-1}\tilde{\mathbf{V}}^T\mathbb{P}_A\left(\mathbf{A}\mathbf{u}'_F(s) - \frac{d}{dt}\mathbf{u}'_F(s)\right)ds.$$

By definition of the FOM,

$$\frac{d\tilde{\mathbf{u}}_F}{dt} + \frac{d\mathbf{u}'_F}{dt} = \mathbf{A}\tilde{\mathbf{u}}_F + \mathbf{A}\mathbf{u}'_F.$$

Thus,

$$\tilde{\mathbf{e}}_A(t) = \int_0^t \tilde{\mathbf{V}}\mathbf{S_A}e^{\Lambda_A(t-s)}\mathbf{S_A}^{-1}\tilde{\mathbf{V}}^T\mathbb{P}_A\left(\frac{d}{dt}\tilde{\mathbf{u}}_F(s) - \mathbf{A}\tilde{\mathbf{u}}_F(s)\right)ds.$$

Using the definition of the full-order residual,

$$\tilde{\mathbf{e}}_A(t) = \int_0^t \tilde{\mathbf{V}}\mathbf{S_A}e^{\Lambda_A(t-s)}\mathbf{S_A}^{-1}\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}_F(\tilde{\mathbf{u}}_F(s))ds.$$

Note that, although $\mathbf{r}_F(\mathbf{u}_F)$ is exactly zero, this is no longer guaranteed when evaluated at the projected solution $\tilde{\mathbf{u}}_F$. Taking the norm and using the sub-multiplicative property,

$$\|\tilde{\mathbf{e}}_A(t)\|_2 \leq \|\mathbf{S_A}\|_2\left\|\mathbf{S_A}^{-1}\right\|_2\int_0^t\left\|e^{\Lambda_A(t-s)}\right\|_2\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}_F(\tilde{\mathbf{u}}_F(s))\right\|_2 ds.$$

□

Similar to the non-linear case, Eqns 34 and 35 contain two distinct terms. The first term corresponds to the exponential growth in time, $e^{\Lambda_A s}$. In contrast to the non-linear case, however, the arguments in the exponential contain the eigenvalues of $\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{A}\tilde{\mathbf{V}}$. When these eigenvalues are negative, the term $e^{\Lambda_A s}$ can diminish the growth of error in time. The second term of interest is given by $\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}_F(\tilde{\mathbf{u}}(t))\right\|_2$. This term corresponds to the error introduced due to the unresolved scales at time t. This term is a statement of the error introduced due to the closure problem.

Corollary 2.1. *If* $\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbb{P}_G\mathbf{r}(\tilde{\mathbf{u}}_F(t))\right\|_2$, *then the upper bound on the error introduced due to the fine-scales is less for APG than for the Galerkin Method.*

Proof. By inspection, when $\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbb{P}_G\mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2$, the error introduced in the integrand in Eqn. 35 due to the coarse-scale residual is less than that in Eqn. 34. □

Theorem 3. *Let* $\mathbf{R}(\mathbf{u}) = \mathbf{A}\mathbf{u}$ *be a linear time-invariant operator. Upper bounds of the Galerkin and Adjoint Petrov-Galerkin projections of the full-order coarse-scale residual, are, respectively,*

$$\left\|\tilde{\mathbf{V}}^T\mathbb{P}_G\mathbf{r}(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_0^\tau e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2 + \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_\tau^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2,$$

$$\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_0^\tau e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta - \tau\tilde{\mathbf{V}}^T\mathbf{A}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t)\right\|_2 + \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_\tau^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2.$$

Proof. The result will be proved first for the Galerkin method and then for the Adjoint Petrov-Galerkin method. By definition of the residual and the Galerkin projector,

$$\mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F) = \tilde{\Pi}\frac{d\tilde{\mathbf{u}}_F}{dt} - \tilde{\Pi}\mathbf{A}\tilde{\mathbf{u}}_F.$$

By definition of the FOM,

$$\tilde{\Pi}\left[\frac{d\tilde{\mathbf{u}}_F}{dt} + \frac{d\mathbf{u}'_F}{dt} - \mathbf{A}\tilde{\mathbf{u}}_F - \mathbf{A}\mathbf{u}'_F\right] = \mathbf{0}.$$

Noting that $\Pi'\tilde{\mathbf{u}}_F = \mathbf{0}$ and rearranging,

$$\mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F) = \tilde{\Pi}\mathbf{A}\mathbf{u}'_F. \tag{39}$$

The evolution equation of the fine-scales is defined by the FOM projected onto the fine-scale space,

$$\frac{d\mathbf{u}'_F}{dt} = \Pi'\mathbf{A}\tilde{\mathbf{u}}_F + \Pi'\mathbf{A}\mathbf{u}'_F, \qquad \mathbf{u}'_F(0) = \mathbf{0}.$$

The above equation is a first-order nonhomogeneous linear system of differential equations. By the definition of the initial conditions, the solution to the fine-scales reduces to,

$$\mathbf{u}'(t) = \int_0^t e^{\Pi'\mathbf{A}(t-s)}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(s)ds,$$

where $e^{\Pi'\mathbf{A}}$ is the matrix exponential. Introducing a change of variables, $\zeta = t - s$, this becomes,

$$\mathbf{u}'(t) = \int_0^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta. \tag{40}$$

Substituting Eq. 40 into Eq. 39,

$$\mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F(t)) = \tilde{\Pi}\mathbf{A}\int_0^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta. \tag{41}$$

Left multiplying by $\tilde{\mathbf{V}}^T$, noting that $\tilde{\mathbf{V}}^T\tilde{\Pi} = \tilde{\mathbf{V}}^T$, and taking the norm,

$$\left\|\tilde{\mathbf{V}}^T\mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 = \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_0^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2.$$

Breaking the integral up into two intervals and applying the triangle inequality, the desired result for Galerkin is obtained,

$$\left\|\tilde{\mathbf{V}}^T\mathbb{P}_G \mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_0^\tau e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2 + \left\|\tilde{\mathbf{V}}^T\mathbf{A}\int_\tau^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta\right\|_2.$$

For APG,

$$\mathbb{P}_A \mathbf{r}_F(\tilde{\mathbf{u}}_F(t)) = \tilde{\Pi}\frac{d}{dt}\tilde{\mathbf{u}}_F(t) - \tilde{\Pi}\mathbf{A}\tilde{\mathbf{u}}_F(t) - \tau\tilde{\Pi}\mathbf{A}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t).$$

The first two terms on the right-hand side are the Galerkin term from Eq. 41. Substituting this into the above equation results in,

$$\mathbb{P}_A \mathbf{r}_F(\tilde{\mathbf{u}}_F(t)) = \tilde{\Pi}\mathbf{A}\int_0^t e^{\Pi'\mathbf{A}\zeta}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t-\zeta)d\zeta - \tau\tilde{\Pi}\mathbf{A}\Pi'\mathbf{A}\tilde{\mathbf{u}}_F(t).$$

Left multiplying by $\tilde{\mathbf{V}}^T$ and taking the norm,

$$\left\|\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 = \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t)\right\|_2 .$$

Breaking the integral up into two intervals, applying the triangle inequality, and noting that $\tilde{\mathbf{V}}^T\tilde{\Pi} = \tilde{\mathbf{V}}^T$, the desired result for APG is obtained,

$$\left\|\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t)\right\|_2 + \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2 .$$

□

Theorem 3 provides an upper bound on the error introduced due to the closure problem for both the Galerkin and APG ROMs. These error bounds were obtained by analytically solving for the fine-scales as a function of coarse-scales. This led to a memory integral involving the past history of the coarse-scales. The memory integral expressing the solution to the fine-scales was then split into two intervals: one from $[0,\tau]$ and one from $[\tau,t]$. The APG method can be interpreted as approximating the integral over the first interval with a quadrature rule and ignoring the second interval. The Galerkin ROM ignores both intervals. Next, Theorem 4 shows that, for sufficiently small τ, the APG approximation is more accurate than the Galerkin approximation. This result is intuitive since, for sufficiently small τ, the quadrature rule used in deriving APG is accurate.

Theorem 4. *In the limit of $\tau \to 0^+$ the error bound provided in Theorem 3 is less per unit τ in APG than in Galerkin,*

$$\lim_{\tau \to 0^+} \frac{1}{\tau}\left[\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t)\right\|_2 + \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2\right] \leq$$
$$\lim_{\tau \to 0^+} \frac{1}{\tau}\left[\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2 + \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2\right]. \quad (42)$$

Proof. Define the difference between the APG and Galerkin error incurred per unit τ as,

$$\bar{\delta}(\tau) := \frac{1}{\tau}\left[\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t)\right\|_2 + \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2\right]$$
$$- \frac{1}{\tau}\left[\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2 + \left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2\right]. \quad (43)$$

After canceling terms, we equivalently have

$$\bar{\delta}(\tau) := \frac{1}{\tau}\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t)\right\|_2 - \frac{1}{\tau}\left\|\tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta\right\|_2 . \quad (44)$$

Apply the rectangle rule to alternatively write the integrals in the above as,

$$\int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) \zeta = \tau \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t) + \frac{d}{d\zeta}\left(e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta)\right)[c]\frac{\tau^2}{2}, \quad (45)$$

for some c in the interval $[0,\tau]$. Inserting Eq. 45 into Eq. 44,

$$\bar{\delta}(\tau) = \frac{1}{\tau}\left\|\tilde{\mathbf{V}}^T \mathbf{A} \frac{d}{d\zeta}\left(e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta)\right)[c]\frac{\tau^2}{2}\right\|_2 - \frac{1}{\tau}\left\|\tau \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t) + \tilde{\mathbf{V}}^T \mathbf{A} \frac{d}{d\zeta}\left(e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta)\right)[c]\frac{\tau^2}{2}\right\|_2 .$$

Factoring out τ, which is non-negative by definition,

$$\bar{\delta}(\tau) = \left\| \tilde{\mathbf{V}}^T \mathbf{A} \frac{d}{d\zeta} \left(e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) \right) [c] \frac{\tau}{2} \right\|_2 - \left\| \tilde{\mathbf{V}}^T \mathbf{A} \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t) + \tilde{\mathbf{V}}^T \mathbf{A} \frac{d}{d\zeta} \left(e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) \right) \frac{\tau}{2} \right\|_2 .$$

Taking the limit as $\tau \to 0^+$ and noting the norm is always non-negative,

$$\lim_{\tau \to 0^+} \bar{\delta}(\tau) = - \left\| \tilde{\mathbf{V}}^T \mathbf{A} \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t) \right\|_2 \leq 0.$$

From Eq. 43 and the algebraic limit theorem, this implies that,

$$\lim_{\tau \to 0^+} \frac{1}{\tau} \left[\left\| \tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta - \tau \tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t) \right\|_2 + \left\| \tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta \right\|_2 \right] \leq$$
$$\lim_{\tau \to 0^+} \frac{1}{\tau} \left[\left\| \tilde{\mathbf{V}}^T \mathbf{A} \int_0^\tau e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta \right\|_2 + \left\| \tilde{\mathbf{V}}^T \mathbf{A} \int_\tau^t e^{\Pi' \mathbf{A} \zeta} \Pi' \mathbf{A} \tilde{\mathbf{u}}_F(t-\zeta) d\zeta \right\|_2 \right]. \quad (46)$$

□

Theorem 4 shows that, for sufficiently small τ, an upper bound on the *a priori* error introduced due to the closure problem in APG is *less* than in Galerkin. This suggests that APG may be more accurate than Galerkin. The results of Theorem 4 will be discussed more in Section 4.1.1. While Theorem 4 shows that, for sufficiently small τ, APG may be more accurate than Galerkin, it does not give insight into appropriate values of τ. Theorem 5 and Corollary 5.1 seek to provide insight into this issue.

Theorem 5. *Let* $\mathbf{R}(\mathbf{u}) = \mathbf{A}\mathbf{u}$ *be a linear time-invariant operator. If* $\mathbf{A}$ *is self-adjoint with all real negative eigenvalues, then* $\lambda_{A_i} \geq \lambda_{G_i}$*, where* $\lambda_{A_1} \geq \lambda_{A_2} \geq \ldots \geq \lambda_{A_K}$ *are the* K *eigenvalues of* $\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}}$ *and* $\lambda_{G_1} \geq \lambda_{G_2} \geq \ldots \geq \lambda_{G_K}$ *are the* K *eigenvalues of* $\tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{A} \tilde{\mathbf{V}}$.

Proof. Expanding the Adjoint Petrov-Galerkin operator,

$$\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}} = \tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \tilde{\mathbf{V}} + \tau \tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{V}}.$$

Noting that the first term on the right-hand side is just the Galerkin term,

$$\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}} = \tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{A} \tilde{\mathbf{V}} + \tau \tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{V}}.$$

As all matrices are self-adjoint, the Weyl inequalities can be used to relate the eigenvalues of $\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}}$ to $\tilde{\mathbf{V}}^T \mathbb{P}_G \mathbf{A} \tilde{\mathbf{V}}$. For $K \times K$ Hermitian positive semi-definite matrices $\mathbf{M}, \mathbf{H}$, and $\mathbf{P}$, where $\mathbf{M} = \mathbf{H} + \mathbf{P}$, the Weyl inequalities state,

$$\lambda_{M_i} \geq \lambda_{H_i},$$

where $\lambda_{M_1} \geq \lambda_{M_2} \geq \ldots \geq \lambda_{M_K}$ are the K eigenvalues of $\mathbf{M}$ and $\lambda_{H_1} \geq \lambda_{H_2} \geq \ldots \geq \lambda_{H_K}$ are the K eigenvalues of $\mathbf{H}$. In the present context, as $\tilde{\mathbf{V}}^T \tilde{\Pi} \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{V}} (= \tilde{\mathbf{V}}^T \mathbf{A} \Pi' \mathbf{A} \tilde{\mathbf{V}})$ is positive semi-definite, it follows from the Weyl inequalities that,

$$\lambda_{A_i} \geq \lambda_{G_i}.$$

□

Corollary 5.1. *Let* $\mathbf{R}(\mathbf{u}) = \mathbf{A}\mathbf{u}$ *be a linear time-invariant operator. If* $\mathbf{A}$ *is self-adjoint with all real negative eigenvalues, then* $\lambda_{A_1} \leq 0$ *for* $\tau \leq \frac{|\gamma_1|}{\gamma_N^2}$*, where* γ_1 *and* γ_N *are the largest and smallest eigenvalues of* $\mathbf{A}$*, respectively, and* λ_{A_1} *is the largest eigenvalue of* $\tilde{\mathbf{V}}^T \mathbb{P}_A \mathbf{A} \tilde{\mathbf{V}}$.

Proof. We again expand the APG projection and simplify,

$$\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{A}\tilde{\mathbf{V}} = \tilde{\mathbf{V}}^T\tilde{\Pi}(\mathbf{I}+\tau A\Pi')\mathbf{A}\tilde{\mathbf{V}} = \tilde{\mathbf{V}}^T(\mathbf{A}+\tau\mathbf{A}\Pi'\mathbf{A})\tilde{\mathbf{V}}$$

Now, let $\lambda_{A_1} \geq \lambda_{A_2} \geq \ldots \geq \lambda_{A_K}$ be the K eigenvalues of $\tilde{\mathbf{V}}^T(\mathbf{A}+\tau\mathbf{A}\Pi'\mathbf{A})\tilde{\mathbf{V}}$, and let $\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_N$ be the N eigenvalues of $\mathbf{A}+\tau\mathbf{A}\Pi'\mathbf{A}$. Recognizing that $\tilde{\mathbf{V}}$ is semi-orthogonal, we apply the Poincaré separation theorem,

$$\kappa_i \geq \lambda_{A_i} \geq \kappa_{N-K+i}, \quad \text{for } i = 1,2,\ldots,K \tag{47}$$

To ensure that $\lambda_{A_1} \leq 0$, we need only show under what conditions $\kappa_1 \leq 0$. Thus, we seek bounds on τ such that $\kappa_1 \leq 0$.

As $\mathbf{A}$ and $\tau\mathbf{A}\Pi'\mathbf{A}$ are both self-adjoint, we can use the Weyl inequalities to bound the smallest eigenvalue of $\mathbf{A}+\tau\mathbf{A}\Pi'\mathbf{A}$. Let $\chi_1 \geq \chi_2 \geq \ldots \geq \chi_N$ be the N eigenvalues of $\mathbf{A}\Pi'\mathbf{A}$, thus,

$$\kappa_1 \leq \gamma_1 + \tau\chi_1, \tag{48}$$

To ensure that $\kappa_1 \leq 0$, we specify that $\lambda_1 + \tau\chi_1 \leq 0$. Noting that $\gamma_1 \leq 0$ leads to the inequality,

$$\lambda_{A_1} \leq 0; \qquad \forall\, \tau \leq \frac{|\gamma_1|}{\chi_1} \tag{49}$$

To write the bounds on τ in Eq. 49 into a more intuitive form, the χ_i can be related to λ_i through the the Weyl inequalities. Recall that χ_i are the eigenvalues of,

$$\mathbf{A}\Pi'\mathbf{A} = \mathbf{A}^2 - \mathbf{A}\tilde{\Pi}\mathbf{A}.$$

As $\mathbf{A}\tilde{\Pi}\mathbf{A}$ is positive semi-definite, one has,

$$\chi_1 \leq \gamma_N^2.$$

Inserting the above inequality into Eq. 49 obtains the desired result,

$$\lambda_{A_1} \leq 0; \qquad \forall\, \tau \leq \frac{|\gamma_1|}{\gamma_N^2}. \tag{50}$$

This upper bound on τ is more conservative than Eq. 49, though both are equally valid.

□

4.1.1. Discussion

Theorems 4 and 5 contain three interesting results that are worth discussing. First, as discussed in Corollary 2.1, Theorem 4 shows that, in the limit $\tau \to 0^+$, the upper-bound provided in Theorem 3 on the error introduced at time t in the APG ROM due to the fine-scales is less than that introduced in the Galerkin ROM (per unit τ). This is an appealing result as APG is derived as a subgrid-scale model. Two remarks are worth making regarding this result. First, while the result was demonstrated in the limit $\tau \to 0^+$, it will hold so long as the norm of the truncation error in the quadrature approximation is less than the norm of the integral it is approximating; i.e. the approximation is doing a better job than neglecting the integral entirely. Second, the result derived in Theorem 4 does not *directly* translate to showing that,

$$\left\|\tilde{\mathbf{V}}^T\mathbb{P}_A\mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2 \leq \left\|\tilde{\mathbf{V}}^T\mathbb{P}_G\mathbf{r}_F(\tilde{\mathbf{u}}_F(t))\right\|_2. \tag{51}$$

This is a consequence of Theorem 3, in which the integral that defines the fine-scale solution was split into two intervals. The APG ROM attempts to approximate the first integral, while the Galerkin ROM ignores both terms. Theorem 4 showed that, in the limit $\tau \to 0^+$, the APG approximation to the first integral is better than in the case of Galerkin (i.e., the APG approximation is better than no approximation). The only time that the result provided in Theorem 4 will *not* translate to APG providing a better approximation to the *entire* integral (and thus proving Eq. 51), is when integration over the second interval "cancels out" the integration over the first interval. To make this idea concrete, consider the integral,

$$\int_0^{2\pi} \sin(x)dx = \int_0^{\pi} \sin(x)dx + \int_{\pi}^{2\pi} \sin(x)dx.$$

Clearly, $\int_0^{2\pi} \sin(x)dx = 0$. If the entire integral was approximated using just the interval 0 to π (which is analogous to APG), then one would end up with the approximation $\int_0^{2\pi} \sin(x)dx \approx \int_0^{\pi} \sin(x)dx = 2$. Alternatively, if one were to ignore the integral entirely (which is analogous to Galerkin) and make the approximation $\int_0^{2\pi} \sin(x)dx \approx 0$ (which in this example is exact), a better approximation would be obtained.

The next interesting result is presented in Theorem 5, where it is shown that for a self-adjoint system with negative eigenvalues, the eigenvalues associated with the APG ROM error equation are *greater* than the Galerkin ROM. This implies that APG is *less* dissipative than Galerkin, and means that errors may be slower to decay in time. Thus, while Theorem 4 shows that the *a priori* contributions to the error due to the closure problem may be smaller in the APG ROM than in the Galerkin ROM, the errors that *are* incurred may be slower to decay. Finally, Corollary 5.1 shows that, for self-adjoint systems with negative eigenvalues, the bounds on the parameter τ such that all eigenvalues associated with the evolution of the error in APG remain negative depends on the spectral content of the Jacobian of $\mathbf{A}$. This observation has been made heuristically in Ref [51]. Although the upper bound on τ in Eq. 50 is very conservative due to the repeated use of inequalities, it provides insight into the selection and behaviour of τ.

4.2. *Selection of Memory Length* τ

The APG method requires the specification of the parameter τ. Theorem 5 showed that, for a self-adjoint linear system, bounds on the value of τ are related to the eigenvalues of the Jacobian of the full-dimensional right-hand side operator. While such bounds provide intuition into the behavior of τ, they are not particularly useful in the selection of an optimal value of τ as they 1.) are conservative due to repeated use of inequalities and 2.) require the eigenvalues of the full right-hand side operator, which one does not have access to in a ROM. Further, the bounds were derived for a self-adjoint linear system, and the extension to non-linear systems is unclear.

In practice, it is desirable to obtain an expression for τ using only the coarse-scale Jacobian, $\tilde{\mathbf{V}}^T \mathbf{J}[\tilde{\mathbf{u}}]\tilde{\mathbf{V}}$. In Ref. [51], numerical evidence showed a strong correlation between the optimal value of τ and this coarse-scale Jacobian. Based on this numerical evidence and the analysis in the previous section, the following heuristic for selecting τ is used:

$$\tau = \frac{C}{\rho(\tilde{\mathbf{V}}^T \mathbf{J}[\tilde{\mathbf{u}}]\tilde{\mathbf{V}})}, \tag{52}$$

where C is a model parameter and $\rho(\cdot)$ indicates the spectral radius. In Ref. [51], C was reported to be 0.2. In the numerical experiments presented later in this manuscript, the sensitivity of APG to the value of τ and the validity of Eq. 52 are examined.

Similar to the selection of τ in the APG method, the LSPG method requires the selection of an appropriate time-step [23]. In practice, this fact can be problematic as finding an optimal time-step for LSPG which minimizes error may result in a small time-step and, hence, an expensive simulation. The selection of the parameter τ, on the other hand, does not impact the computational cost of the APG ROM.

5. Implementation and Computational Cost of the Adjoint Petrov–Galerkin Method

This section details the implementation of the Adjoint Petrov–Galerkin ROM for simple time integration schemes. Algorithms for explicit and implicit time integration schemes are provided, and the approximate

Step in Algorithm 1	Approximate FLOPs
1	$2NK - N$
2	ωN
3	$4NK - N - K$
4	N
5	$(\omega + 4)N$
6	$2N$
7	$2NK - K$
8	$2K$
Total	$8NK + (2\omega + 5)N$
Total for Galerkin Method	$4NK + (\omega - 1)N + K$

Table 1: Approximate floating-point operations for an explicit Euler update to the Adjoint Petrov–Galerkin method reported in Algorithm 1. The total FLOP count for the Galerkin ROM with an explicit Euler update is additionally reported for comparison. A full description of the Galerkin update is provided in Appendix C.

cost of each method in floating-point operations (FLOPs) is analyzed. Here, a FLOP refers to any floating-point addition or multiplication; no distinction is made between the computational cost of either operation. The notation used here is as follows: N is the full-order number of degrees of freedom, K is the number of modes retained in the POD basis, and ωN is the number of FLOPs required for one evaluation of the right-hand side, $\mathbf{R}(\tilde{\mathbf{u}}(t))$. For sufficiently complex problems, ω is usually on the order of $O(10) < \omega < O(1000)$. The analysis presented in this section does not consider hyper-reduction. The analysis can be approximately extended to hyper-reduction by replacing the full-order degrees of freedom with the dimension of the hyper-reduced right-hand side.[4]

5.1. Explicit Time Integration Schemes

This section explores the cost of the APG method within the scope of explicit time integration schemes. For simplicity, the analysis is carried out only for the explicit Euler scheme. The computational cost of more sophisticated time integration methods, such as Runge-Kutta and multistep schemes, is generally a proportional scaling of the cost of the explicit Euler scheme. Algorithm 1 provides the step-by-step procedure for performing an explicit Euler update to the Adjoint Petrov–Galerkin ROM. Table 1 provides the approximate floating-point operations for the steps reported in Algorithm 1. The algorithm for an explicit update to the Galerkin ROM, along with the associated FLOP counts, is provided in Algorithm 7 and Table 8 in Appendix C. As noted previously, LSPG reverts to the Galerkin method for explicit schemes, and so is not detailed in this section. Table 1 shows that, in the case that $K \ll N$ (standard for a ROM) and $\omega \gg 1$ (sufficiently complex right-hand side), the Adjoint Petrov–Galerkin ROM is approximately twice as expensive as the Galerkin ROM.

5.2. Implicit Time Integration Schemes

This section evaluates the computational cost of the Galerkin, Adjoint Petrov–Galerkin, and Least-Squares Petrov–Galerkin methods for implicit time integration schemes. For non-linear systems, implicit time integration schemes require the solution of a non-linear algebraic system at each time-step. Newton's method, along with a preferred linear solver, is typically employed to solve the system. For simplicity, the analysis provided in this section is carried out for the implicit Euler time integration scheme along

[4]An accurate cost-analysis for hyper-reduced ROMs should consider over sampling of the right-hand side and the FOM stencil.

Algorithm 1 Algorithm for an explicit Euler update for the APG ROM

```
Input: $\tilde{\mathbf{a}}^n$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
  1. Compute the state from the generalized coordinates, $\tilde{\mathbf{u}}^n = \tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1}$
  2. Compute the right-hand side from the state, $\mathbf{R}(\tilde{\mathbf{u}}^n)$
  3. Compute the projection of the right-hand side, $\tilde{\Pi}\mathbf{R}(\tilde{\mathbf{u}}^n) = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}^n)$
  4. Compute the orthogonal projection of the right-hand side, $\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n) = \mathbf{R}(\tilde{\mathbf{u}}^n) - \tilde{\Pi}\mathbf{R}(\tilde{\mathbf{u}}^n)$
  5. Compute the action of the Jacobian on $\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)$ using either of the two following strategies:
     (a) Finite difference approximation:
         $$\mathbf{J}[\tilde{\mathbf{u}}^n]\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n) \approx \frac{1}{\varepsilon}\Big[\mathbf{R}\big(\tilde{\mathbf{u}}^n + \varepsilon\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)\big) - \mathbf{R}(\tilde{\mathbf{u}}^n)\Big],$$
         where $\varepsilon$ is a small constant value, usually $\sim O(10^{-5})$.
     (b) Exact linearization:
         $$\mathbf{J}[\tilde{\mathbf{u}}^n]\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n) = \mathbf{R}'[\tilde{\mathbf{u}}^n](\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)),$$
         where $\mathbf{R}'[\tilde{\mathbf{u}}^n]$ is right-hand side operator linearized about $\tilde{\mathbf{u}}^n$.
  6. Compute the full right-hand side: $\mathbf{R}(\tilde{\mathbf{u}}^n) + \tau\mathbf{J}[\tilde{\mathbf{u}}^n]\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)$
  7. Project: $\tilde{\mathbf{V}}^T\Big[\mathbf{R}(\tilde{\mathbf{u}}^n) + \tau\mathbf{J}[\tilde{\mathbf{u}}^n]\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)\Big]$
  8. Update the state $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}^n + \Delta t\tilde{\mathbf{V}}^T\Big[\mathbf{R}(\tilde{\mathbf{u}}^n) + \tau\mathbf{J}[\tilde{\mathbf{u}}^n]\Pi'\mathbf{R}(\tilde{\mathbf{u}}^n)\Big]$
```

with Newton's method to solve the non-linear system. Before proceeding, the full-order residual, Galerkin residual, and APG residual at time-step $(n+1)$ are denoted as,

$$\mathbf{r}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1}) = \tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1} - \tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n} - \Delta t \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1}),$$
$$\mathbf{r}_G(\tilde{\mathbf{a}}^{n+1}) = \tilde{\mathbf{a}}^{n+1} - \tilde{\mathbf{a}}^{n} - \Delta t \tilde{\mathbf{V}}^T \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1}),$$
$$\mathbf{r}_A(\tilde{\mathbf{a}}^{n+1}) = \tilde{\mathbf{a}}^{n+1} - \tilde{\mathbf{a}}^{n} - \Delta t \tilde{\mathbf{V}}^T \left[\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1}) + \tau \mathbf{J}[\tilde{\mathbf{u}}]\Pi' \mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}^{n+1})\right].$$

As the future state, $\tilde{\mathbf{a}}^{n+1}$, is unknown, we denote an intermediate state, $\tilde{\mathbf{a}}_k$, that is updated after every Newton iteration until some convergence criterion is met. Newton's method is defined by the iteration,

$$\frac{\partial \mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial \tilde{\mathbf{a}}_k}\left[\tilde{\mathbf{a}}_{k+1} - \tilde{\mathbf{a}}_k\right] = -\mathbf{r}(\tilde{\mathbf{a}}_k). \tag{53}$$

Newton's method solves Eq. 53 for the change in the state, $\tilde{\mathbf{a}}_{k+1} - \tilde{\mathbf{a}}_k$, for $k = 1, 2, \ldots$, until the residual converges to a sufficiently-small number. For a ROM, both the assembly and solution of this linear system is the dominant cost of an implicit method.

Two methods are considered for the solution to the non-linear algebraic system arising from implicit time discretizations of the G and APG ROMs: Newton's method with direct Gaussian elimination and Jacobian-Free Newton-Krylov GMRES. The Gauss-Newton method with Gaussian elimination is considered for the solution to the least-squares problem arising in LSPG.

Algorithm 2 provides the step-by-step procedures for performing an implicit Euler update to the Adjoint Petrov–Galerkin ROM with the use of Newton's method and Gaussian elimination. Table 2 provides the approximate floating-point operations for the steps reported in these algorithms. Analogous results for the Galerkin and LSPG ROMs are reported in Algorithms 8 and 9, and Tables 9 and 10 in Appendix C. In the limit that $K \ll N$ and $\omega \gg 1$, the total FLOP counts reported show that APG is twice as expensive as both the LSPG and Galerkin ROMs. It is observed that the dominant cost for all three methods lies in the computation of the low-dimensional residual Jacobian. Computation of the low-dimensional Jacobian requires K evaluations of the unsteady residual. Depending on values of ω, N, and K, this step can consist of over 50% $(K \ll N)$ of the CPU time.[5]

To avoid the cost of computing the low-dimensional Jacobian required in the linear solve at each Netwon step, the Galerkin and APG ROMs can make use of Jacobian-Free Netwon-Krylov (JFNK) methods to solve the linear system, opposed to direct methods such as Gaussian elimination. JFNK methods are iterative methods that allow one to circumvent the expense associated with computing the full low-dimensional Jacobian. Instead, JFNK methods only compute the *action* of the Jacobian on a vector at each iteration of the linear solve. This can drastically decrease the cost of the implicit solve. JFNK utilizing the Generalized Minimal Residual (GMRES) method [65], for example, is guaranteed to converge to the solution $\tilde{\mathbf{a}}_k$ in at most K iterations. It takes K residual evaluations just to form the Jacobian required for direct methods.

The LSPG method is formulated as a non-linear least-squares problem. The use of Jacobian-free methods to solve non-linear least-squares problems is significantly more challenging. The principle issue encountered in attempting to use Jacobian-free methods for such applications it that one requires the action of the *transpose* of the residual Jacobian on a vector. This quantity cannot be computed via a standard finite difference approximation or linearization. It is only recently that true Jacobian-free methods have been utilized for solving non-linear least-squares problems. In Ref [66], for example, automatic differentiation is utilized to compute the action of the transposed Jacobian on a vector. Due to the challenges associated with Jacobian-free methods for non-linear least-squares problems, this method is not considered here as a solution technique for LSPG.

[5]It is noted that the low-dimensional Jacobian can be computed in parallel.

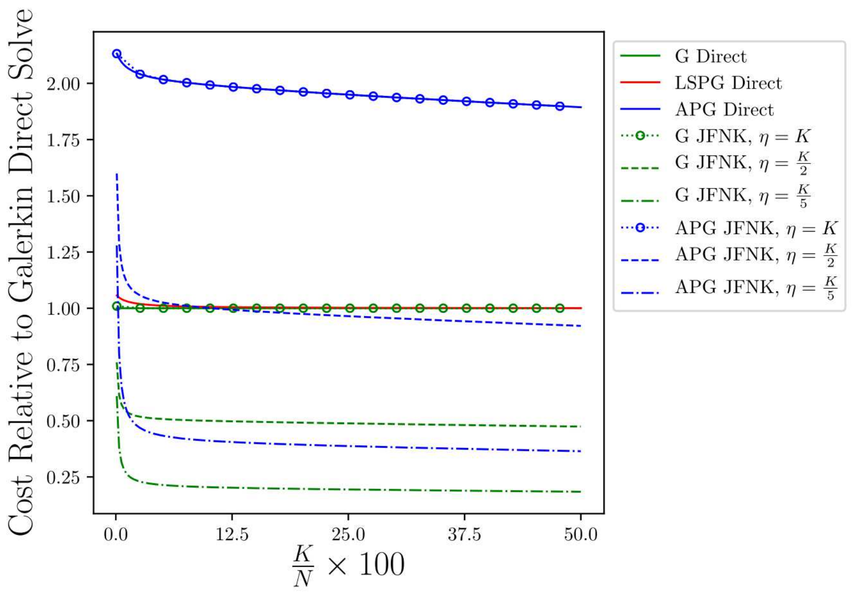


Figure 1: Estimates of the G, APG, and LSPG reduced-order models for an implicit Euler update. This plot is generated for values of $N = 1000$ and $\omega = 50$, and $\eta = \{K, K/2, K/5\}$, where η is the total number of iterations required for the GMRES solver at each Newton step.

Algorithm 3 and Table 3 report the algorithm and FLOPs required for an implicit Euler update to APG using JFNK GMRES. The term $\eta \leq K$ is the number of iterations needed for convergence of the GMRES solver at each Newton iteration. For a concise presentation, the same update for the Galerkin ROM is not presented. Figure 1 shows the ratio of the cost of the various implicit ROMs as compared to the Galerkin ROM solved with Gaussian elimination. The standard LSPG method is seen to be approximately the same cost of Galerkin, while APG is seen to be approximately 2x the cost of Galerkin. The success of the JFNK methods depends on the number of GMRES iterations required for convergence. If $\eta = K$, which is the maximum number of iterations required for GMRES, the cost of JFNK methods is seen to be the same as their direct-solve counterparts. For cases where JFNK converges at a rate of $\eta < K$, the iterative methods out-perform their direct-solve counterparts.

The analysis presented here shows that, for a given basis dimension, the Adjoint Petrov–Galerkin ROM is approximately twice the cost of the Galerkin ROM for both implicit and explicit solvers. In the implicit case, the APG ROM utilizing a direct linear solver is approximately 2x the cost of LSPG. It was highlighted, however, that APG can be solved via JFNK methods. For cases where one either doesn't have access to the full Jacobian, or the full Jacobian can't be stored, JFNK methods can significantly decrease the ROM cost. The use of JFNK methods within the LSPG approach is more challenging due to the presence of the transpose of the residual Jacobian. Lastly it is noted that, although hyper-reduction can decrease the cost of a residual evaluation, it does not entirely alleviate the cost of forming the Jacobian.

6. Numerical Examples

Applications of the APG method are presented for ROMs of compressible flows: the 1D Sod shock tube problem and 2D viscous flow over a cylinder. In both problems, the test bases are chosen via POD. The shock tube problem highlights the improved stability and accuracy of the APG method over the standard Galerkin ROM, as well as improved performance over the LSPG method. The impact of the choice of τ (APG) and Δt (LSPG) time-scales are also explored. The cylinder flow experiment examines a more

Algorithm 2 Algorithm for an implicit Euler update for the APG ROM using Newton's Method with Gaussian Elimination

```
Input: $\tilde{\mathbf{a}}^n$, residual tolerance $\xi$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
  1. Set initial guess, $\tilde{\mathbf{a}}_k$
  2. Loop while $\mathbf{r}^k > \xi$
     (a) Compute the state from the generalized coordinates, $\tilde{\mathbf{u}}_k = \tilde{\mathbf{V}}\tilde{\mathbf{a}}_k$
     (b) Compute the right-hand side from the full state, $\mathbf{R}(\tilde{\mathbf{u}}_k)$
     (c) Compute the projection of the right-hand side, $\tilde{\Pi}\mathbf{R}(\tilde{\mathbf{u}}^n) = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}^n)$
     (d) Compute the orthogonal projection of the right-hand side, $\Pi'\mathbf{R}(\tilde{\mathbf{u}}_k) = \mathbf{R}(\tilde{\mathbf{u}}_k) - \tilde{\Pi}\mathbf{R}(\tilde{\mathbf{u}}_k)$
     (e) Compute the action of the right-hand side Jacobian on $\Pi'\mathbf{R}(\tilde{\mathbf{u}}_k)$, as in Alg. 1.
     (f) Compute the modified right-hand side, $\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k) + \tau\mathbf{J}[\tilde{\mathbf{u}}]\Pi'\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k)$
     (g) Project the modified right-hand side, $\tilde{\mathbf{V}}^T\left[\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k) + \tau\mathbf{J}[\tilde{\mathbf{u}}]\Pi'\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k)\right]$
     (h) Compute the APG residual, $\mathbf{r}_A(\tilde{\mathbf{a}}_k) = \tilde{\mathbf{a}}_k - \tilde{\mathbf{a}}^n - \Delta t\tilde{\mathbf{V}}^T\left[\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k) + \tau\mathbf{J}[\tilde{\mathbf{u}}]\Pi'\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k)\right]$
     (i) Compute the residual Jacobian, $\frac{\partial\mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial\tilde{\mathbf{a}}_k}$
     (j) Solve the linear system via Gaussian Elimination: $\frac{\partial\mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial\tilde{\mathbf{a}}_k}\Delta\tilde{\mathbf{a}} = -\mathbf{r}(\tilde{\mathbf{a}}_k)$
     (k) Update the state: $\tilde{\mathbf{a}}_{k+1} = \tilde{\mathbf{a}}_k + \Delta\tilde{\mathbf{a}}$
     (l) $k = k+1$
  3. Set final state, $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}_k$
```

Step in Algorithm 2	Approximate FLOPs
2a	$2NK - N$
2b	ωN
2c	$4NK - N - K$
2d	N
2e	$(\omega+4)N$
2f	$2N$
2g	$2NK - K$
2h	$3K$
2i	$(2\omega+5)NK + K^2 + 8NK^2$
2j	K^3
2k	K
Total	$(2\omega+5)N + 2K + (2\omega+13)NK + K^2 + 8NK^2 + K^3$
Galerkin ROM FLOP count	$(\omega-1)N + 3K + (\omega+3)NK + 2K^2 + 4NK^2 + K^3$
LSPG ROM FLOP count	$(\omega+2)N + (\omega+6)NK - K^2 + 4NK^2 + K^3$

Table 2: Approximate floating-point operations for one Newton iteration for the implicit Euler update to the Adjoint Petrov–Galerkin method reported in Algorithm 2. FLOP counts for the Galerkin ROM and LSPG ROM with an implicit Euler update are additionally reported for comparison. A full description of the Galerkin and LSPG ROM updates are provided in Appendix C

Algorithm 3 Algorithm for an implicit Euler update for the APG ROM using JFNK GMRES

```
Input: $\tilde{\mathbf{a}}^n$, residual tolerance $\xi$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
   1. Set initial guess, $\tilde{\mathbf{a}}_k$
   2. Loop while $\mathbf{r}^k > \xi$
     (a–h) Compute steps 2a through 2h in Algorithm 2
        (i) Solve the linear system, $\frac{\partial \mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial \tilde{\mathbf{a}}_k}\Delta\tilde{\mathbf{a}}_k = \mathbf{r}_A(\tilde{\mathbf{a}}_k)$ using Jacobian-Free GMRES
        (j) Update the state: $\tilde{\mathbf{a}}_{k+1} = \tilde{\mathbf{a}}_k + \Delta\tilde{\mathbf{a}}$
        (k) $k = k+1$
   3. Set final state, $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}_k$
```

Step in Algorithm 3	Approximate FLOPs
2a	$2NK - N$
2b	ωN
2c	$4NK - N - K$
2d	N
2e	$(\omega + 4)N$
2f	$2N$
2g	$2NK - K$
2h	$3K$
2i	$(2\omega + 5)N\eta + K\eta + 8NK\eta + \eta^2 K$
2k	K
Total	$\big((2\eta+2)\omega + 5\eta + 5\big)N + (\eta^2 + \eta + 2)K + (8\eta + 8)NK$

Table 3: Approximate floating-point operations for one Newton iteration for the implicit Euler update to the Adjoint Petrov–Galerkin method using Jacobian-Free GMRES reported in Algorithm 3.

complex problem and assesses the predictive capability of APG in comparison with Galerkin and LSPG ROMs. The effect of the choice of τ on simulation accuracy is further explored.

6.1. Example 1: Sod Shock Tube with reflection

The first case considered is the Sod shock tube, described in more detail in [67]. The experiment simulates the instantaneous bursting of a diaphragm separating a closed chamber of high-density, high-pressure gas from a closed chamber of low-density, low pressure gas. This generates a strong shock, a contact discontinuity, and an expansion wave, which reflect off the shock tube walls at either end and interact with each other in complex ways. The system is described by the one-dimensional compressible Euler equations with the initial conditions,

$$\rho = \begin{cases} 1 & x \leq 0.5 \\ 0.125 & x > 0.5 \end{cases}, \qquad p = \begin{cases} 1 & x \leq 0.5 \\ 0.1 & x > 0.5 \end{cases}, \qquad u = \begin{cases} 0 & x \leq 0.5 \\ 0 & x > 0.5 \end{cases},$$

with $x \in [0,1]$. Impermeable wall boundary conditions are enforced at x = 0 and x = 1.

6.1.1. Full-Order Model

The 1D compressible Euler equations are solved using a finite volume method and explicit time integration. The domain is partitioned into 1,000 cells of uniform width. The finite volume method uses the first-order Roe flux [68] at the cell interfaces. A strong stability-preserving RK3 scheme [69] is used for time integration. The solution is evolved for $t \in [0.0, 1.0]$ with a time-step of $\Delta t = 0.0005$, ensuring CFL≤ 0.75 for the duration of the simulation. The solution is saved every other time-step, resulting in 1,000 solutions snapshots for each conserved variable.

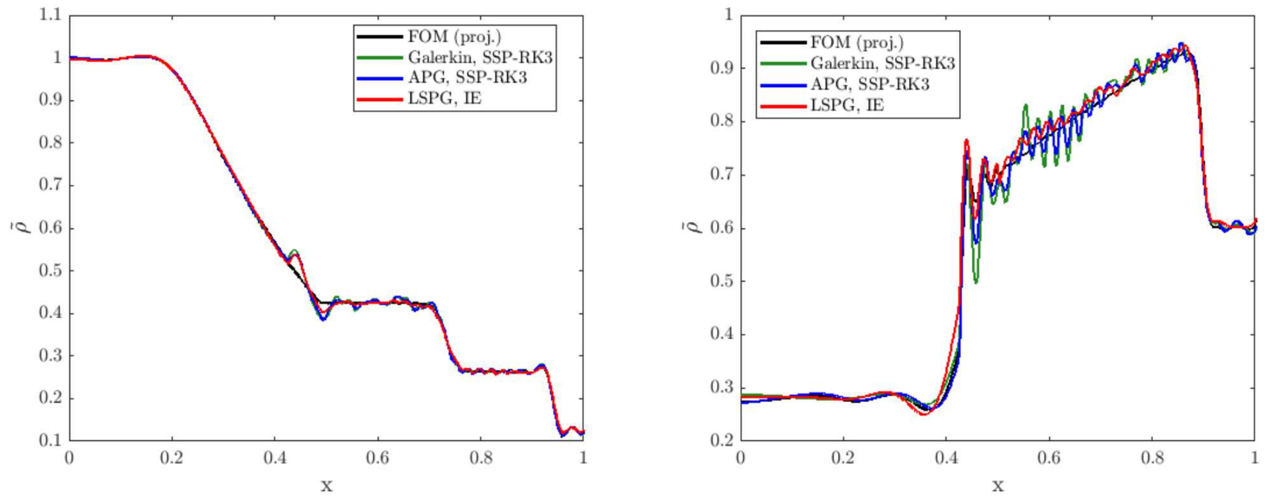
6.1.2. Solution of the Reduced-Order Model

Using the FOM data snapshots, trial bases for the ROMs are constructed via the proper orthogonal decomposition (POD) approach. A separate basis is constructed for each conserved variable. The complete basis construction procedure is detailed in Appendix B. Once a coarse-scale trial basis $\tilde{\mathbf{V}}$ is built, a variety of ROMs are evaluated according to the following formulations:

1. Galerkin ROM:
$$\tilde{\mathbf{V}}^T \left(\frac{d\tilde{\mathbf{u}}}{dt} - \mathbf{R}(\tilde{\mathbf{u}}) \right) = 0, \qquad t \in [0,1].$$
2. Adjoint Petrov–Galerkin ROM:
$$\tilde{\mathbf{V}}^T \left(\mathbf{I} + \tau \mathbf{J}[\tilde{\mathbf{u}}]\Pi' \right) \left(\frac{d\tilde{\mathbf{u}}}{dt} - \mathbf{R}(\tilde{\mathbf{u}}) \right) = 0, \qquad t \in [0,1].$$
Remark: The Adjoint Petrov–Galerkin ROM requires specification of τ.
3. Least-Squares Petrov–Galerkin ROM (Implicit Euler Time Integration):
$$\mathbf{u}^n = \underset{\mathbf{y} \in \text{Range}(\tilde{\mathbf{V}})}{\arg\min} \left\| \frac{\mathbf{y} - \tilde{\mathbf{u}}^{n-1}}{\Delta t} - \mathbf{R}(\mathbf{y}) \right\|_2^2, \qquad \text{for } n = 1, 2, \ldots, \text{ceil}\big(\frac{1}{\Delta t}\big).$$
Remark: The LSPG approach is strictly coupled to the time integration scheme and time-step.

ROM Type	Time Scheme	Δt	τ	$\int \|\|e\|\|_2 dt$
Galerkin	SSP-RK3	0.0005	N/A	1.5752
Galerkin	Imp. Euler	0.0005	N/A	1.0344
APG	SSP-RK3	0.0005	0.00043	1.0637
APG	Imp. Euler	0.0005	0.00043	0.8057
APG	Imp. Euler	0.001	0.00043	0.7983
LSPG	Imp. Euler	0.0005	N/A	1.1668
LSPG	Imp. Euler	0.001	N/A	1.4917

Table 4: Computational details for Sod shock tube ROM cases, $K = 150$



(a) $t = 0.25$.

(b) $t = 1$.

Figure 2: Density profiles for the Sod shock tube with $K = 150$, $\Delta t = 0.0005$.

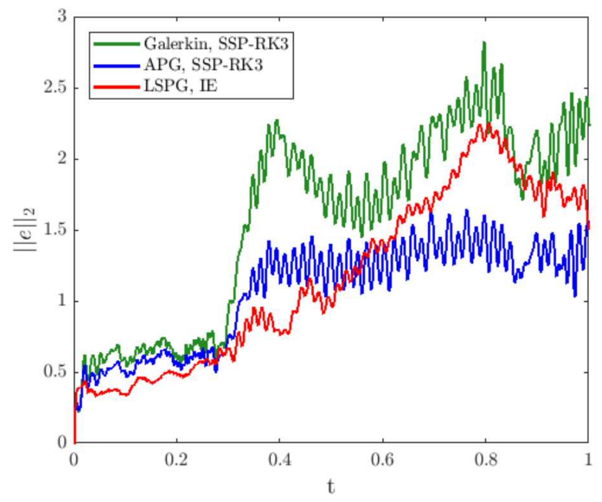
6.1.3. Numerical Results

The first case considered uses 50 basis vectors each for the conserved variables $\rho, \rho u$, and ρE. The total dimension of the reduced model is thus $K = 150$. Roughly 99.9–99.99% of the POD energy is captured by this 150-mode basis. In fact, 99% of the energy is contained in the first 5-12 modes of each conserved variable.

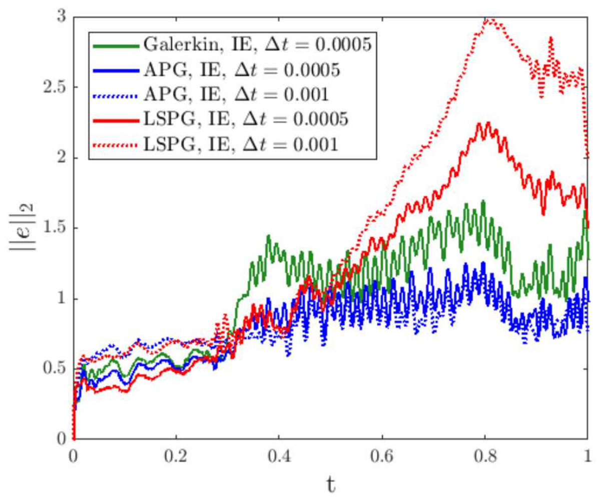
The Adjoint Petrov–Galerkin ROM requires specification of the memory length τ. Similarly, LSPG requires the selection of an appropriate time-step. The sensitivity of both methods to this selection will be discussed later in this section. The simulation parameters are provided in Table 4.

Density profiles at $t = 0.25$ and $t = 1.0$ for explicit Galerkin and APG ROMs, along with an implicit LSPG ROM, are displayed in Fig. 2. All three ROMs are capable of reproducing the shock tube density profile in Fig. 2a; a normal shock propagates to the right and is followed closely behind by a contact discontinuity, while an expansion wave propagates to the left. All three methods exhibit oscillations at $x = 0.5$, the location of the imaginary burst diaphragm, and near the shock at $x = 0.95$. At $t = 1.0$, when the shock has reflected from the right wall and interacted with the contact discontinuity, much stronger oscillations are present, particularly near the reflected shock at $x = 0.45$. These oscillations are reminiscent of Gibbs phenomenon, and are an indicator of the inability to accurately reconstruct sharp gradients. The Galerkin ROM exhibits the largest oscillations of the ROMs considered, while LSPG exhibits the smallest.

Figure 3 shows the evolution of the error for all of the ROMs listed in Table 4. The L^2-norm of the error



(a) Explicit Galerkin/APG, implicit LSPG, $\Delta t = 0.0005$



(b) All implicit, various Δt

Figure 3: L^2-norm error profiles for the Sod shock tube with 150 basis vectors.

is computed as,

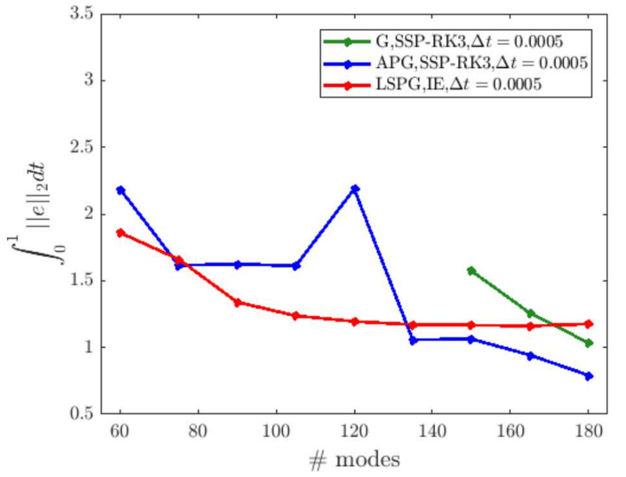
$$||e||_2 = \sqrt{\sum_{i=1}^{1000} \left[(\tilde{\rho}_{i,ROM} - \tilde{\rho}_{i,FOM})^2 + (\widetilde{\rho u}_{i,ROM} - \widetilde{\rho u}_{i,FOM})^2 + (\widetilde{\rho E}_{i,ROM} - \widetilde{\rho E}_{i,FOM})^2 \right]}.$$

Here, the subscript i denotes each finite volume cell. The FOM values used for error calculations are projections of the FOM data onto $\tilde{\mathcal{V}}$, e.g. $\tilde{\rho}_{FOM} = \tilde{\Pi}\rho_{FOM}$. This error measure provides a fair upper bound on the accuracy of the ROMs, as the quality of the ROM is generally dictated by the richness of the trial basis and the projection of the FOM data is the maximum accuracy that can be reasonably hoped for.

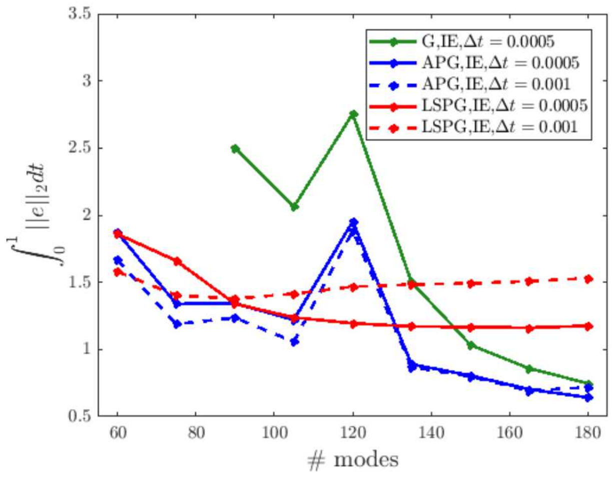
In Figure 3, it is seen that the APG ROM exhibits improved accuracy over the Galerkin ROM. The LSPG ROM for $\Delta t = 0.0005$ performs slightly better than the explicit Galerkin ROM, and worse than the implicit Galerkin ROM. Increasing the time-step to $\Delta t = 0.001$ results in a significant increase in error for the LSPG ROM. This is due to the fact that the performance of LSPG is influenced by the time-step. For a trial basis containing much of the residual POD energy, LSPG will generally require a very small time-step to improve accuracy; this sensitivity will be explored later. Lastly, it is observed that the APG ROM is *not* significantly affected by the time-step. The APG ROM with $\Delta t = 0.001$ shows moderately increased error prior to $t = 0.3$ and similar error afterwards when compared against the $\Delta t = 0.0005$ APG ROM case.

Figure 4 studies the effect of the number of modes retained in the trial basis on the stability and accuracy, over the range $K = 60, 75, 90, \ldots, 180$. Missing data points indicate an unstable solution. Values of τ for the APG ROMs are again selected by user choice. The most striking feature of these plots is the fact that even though the explicit Galerkin ROM is unstable for $K \leq 135$ and the implicit Galerkin ROM is unstable for $K \leq 75$, the APG and LSPG ROMs are stable for all cases. Furthermore, the APG and LSPG ROMs are capable of achieving stability with a time-step twice as large as that of the Galerkin ROM. The cost of the APG and LSPG ROMs are effectively halved, but they are still able to stabilize the simulation. Interestingly, the Galerkin and APG ROMs both exhibit abrupt peaks in error at $K = 120$, while the LSPG ROMs do not. The exact cause of this is unknown, but displays that a monotonic decrease in error with enrichment of the trial space is not guaranteed.

Several interesting comparisons between APG and LSPG arise from Figure 4. First, with the exception of the $K = 120$ case, Fig. 3a shows that the APG ROM with explicit time integration exhibits accuracy comparable to that of the LSPG ROM with implicit time integration. As can be seen in comparing Tables 1



(a) Explicit Galerkin/APG, implicit LSPG, fixed Δt



(b) All implicit, various Δt

Figure 4: Integrated error mode sensitivity study for the Sod shock tube.

and 10, the cost of APG with explicit time integration is significantly lower than the cost of LSPG. This is an attractive feature of APG, as it is able to use inexpensive explicit time integration while LSPG is restricted to implicit methods. Additionally, we draw attention to the poor performance of LSPG at high K for a moderate time-step in Fig. 4b. Increasing the time-step to $\Delta t = 0.001$ to decrease simulation cost only exacerbates this issue; as the trial space is enriched, LSPG requires a smaller time-step to yield accurate results. If we wish to improve the LSPG solution for $K = 150$, we must decrease the time-step below that of the FOM. The accuracy of the APG ROM does not change when the time-step is doubled from $\Delta t = 0.0005$ to $\Delta t = 0.001$. This halves the cost of the APG ROM with no significant drawbacks.

6.1.4. Optimal Memory Length Investigations

As mentioned previously, the success of LSPG is tied to the physical time-step and the time integration scheme, while the parameter τ in the APG method may be chosen independently from these factors. In minimizing ROM error, finding an optimal value of τ for the APG ROM may permit the choice of a much larger time-step than the optimal LSPG time-step. Further, the APG method may be applied with explicit time integration schemes, which are generally much less expensive than the implicit methods which LSPG is restricted to. To demonstrate this, the APG ROM and LSPG ROM with $K = 150$ are simulated for a variety of time scales (τ for APG and Δt for LSPG).

Figure 5 shows the integrated error of the ROMs versus the relevant time scale. For this case, the optimal value of Δt for LSPG is less than 0.0001 and is not shown. The optimal value of τ is not greatly affected by the choice of time integration scheme (implicit or explicit) or time-step. Furthermore, because τ can be chosen independently from Δt for APG, the APG ROM can produce low error at a much larger time-step ($\Delta t = 0.001$) than the optimal time-step for the LSPG ROM. This highlights the fact that the "optimal" LSPG ROM may be computationally expensive due to a small time-step, whereas the "optimal" APG ROM requires only the specification of τ and can use, potentially, much larger time-steps than LSPG. It has to be mentioned, however, that the choice of τ has an impact on performance — selections of τ larger than those plotted in Fig. 5 caused the ROM to lose stability.

Also discussed previously, computing an optimal value of τ *a priori* may be linked to the spectral radius of the coarse-scale Jacobian, (i.e. $\rho\big(\tilde{\mathbf{V}}^T \mathbf{J}[\tilde{\mathbf{u}}_0]\tilde{\mathbf{V}}\big)$, not to be confused with the physical density ρ). We consider the APG ROM for basis sizes of $K = 30, 60, 90, 150, 180, 240$. For each case, an "optimal" τ is found by minimizing the misfit between the ROM solution and the projected FOM solution. The misfit is

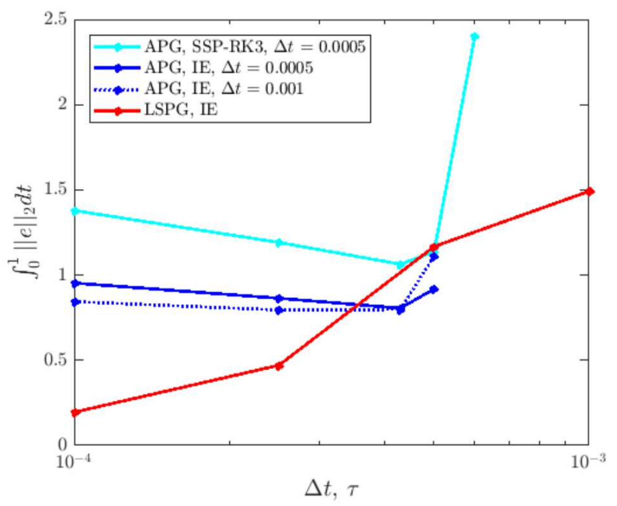


Figure 5: Error as a function of time scale, $K = 150$.

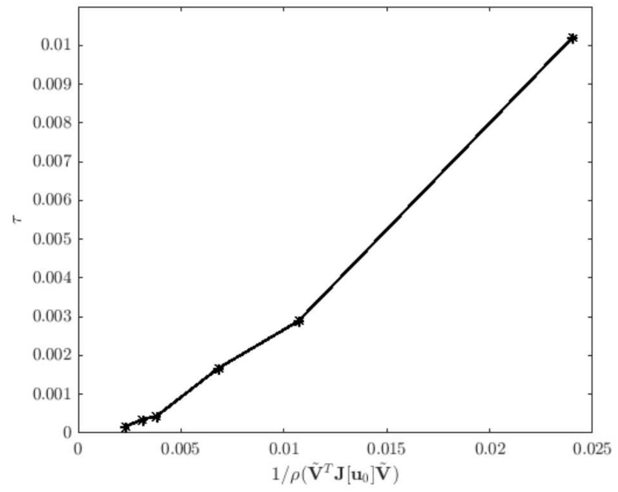


Figure 6: Optimal τ as a function of the spectral radius evaluated at $t = 0$.

defined as follows,

$$\mathcal{J}(\tau) = \sum_{i=1}^{200} ||e(\tau, t = i10\Delta t)||_2. \tag{54}$$

Equation 54 corresponds to summing the L^2-norm of the error at every 10th time-step. Figure 6 shows the resulting optimal τ for each case plotted against the inverse of the spectral radius of the coarse-scale Jacobian evaluated at $t = 0$. The strong linear correlation suggests that a near-optimal value of τ may be chosen by evaluating the spectral radius, $\rho\left(\tilde{\mathbf{V}}^T \mathbf{J}[\tilde{\mathbf{u}}_0]\tilde{\mathbf{V}}\right)$ and using the above linear relationship to τ.

Two points are emphasized here:

1. The spectral radius plays an important role in both implicit and explicit time integrators and is often the determining factor in the choice of the time-step. Theoretical analysis on the stability of explicit methods (and convergence of implicit methods) shows a similar dependence to the spectral radius. Choosing the memory length to be $\tau = \Delta t$ is one simple heuristic that may be used.
2. While a linear relationship between τ and the spectral radius of the coarse-scale Jacobian has been observed in every problem the authors have examined, the slope of the fit is somewhat problem dependent. For the purpose of reduced-order modeling, however, this is only a minor inconvenience as an appropriate value of τ can be selected by assessing the performance of the ROM on the training set, i.e., on the simulation used to construct the POD basis.
3. Finally, more complex methods may be used to compute τ. A method to dynamically compute τ based on Germano's identity, for instance, was proposed in [52] in the context of the simulation of turbulent flows with Fourier-Galerkin methods. Extension of this technique to projection-based ROMs and the development of additional techniques to select τ will be the subject of future work.

6.2. Example 2: Flow Over Cylinder

The second case considered is viscous compressible flow over a circular cylinder. The flow is described by the two-dimensional compressible Navier-Stokes equations. A Newtonian fluid and a calorically perfect gas are assumed.

r_{max}	N_r	N_θ	p_r	p_θ	Δt	Mach	a_∞	p_∞	T_∞
60	80	80	3	3	$5e-3$	0.2	1.0	1.0	γ^{-1}

Table 5: Details used for flow over cylinder problem. In the above table, N_r and N_θ are the number of cells in the radial and θ direction, respectively. Similarly, p_r and p_θ are the polynomial orders in the radial and θ direction. Lastly, a_∞, p_∞, and T_∞ are the free-stream speed of sound, pressure, and temperature.

6.2.1. Full-Order Model

The compressible Navier-Stokes equations are solved using a discontinuous Galerkin (DG) method and explicit time integration. Spatial discretization with the discontinuous Galerkin method leads to a semi-discrete system of the form,

$$\frac{d\mathbf{u}}{dt} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{u}), \qquad \mathbf{u}(t=0) = \mathbf{u}_0,$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is a block diagonal mass matrix and $\mathbf{f}(\mathbf{u}) \in \mathbb{R}^N$ is a vector containing surface and volume integrals. Thus, using the notation defined in Eq. 1, the right-hand side operator for the DG discretization is defined as,

$$\mathbf{R}(\mathbf{u}) = \mathbf{M}^{-1}\mathbf{f}(\tilde{\mathbf{u}}).$$

For the flow over cylinder problem considered in this section, a single block domain is constructed in polar coordinates by uniformly discretizing in θ and by discretizing in the radial direction by,

$$r_{i+1} = r_i + r_i(R_g - 1),$$

where R_g is a stretching factor and is defined by,

$$R_g = r_{max}^{1/N_r}.$$

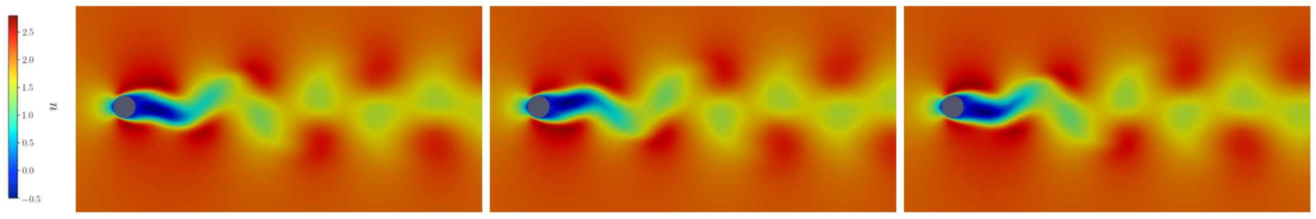
The DG method utilizes the Roe flux at the cell interfaces and uses the first form of Bassi and Rebay [70] for the viscous fluxes. Temporal integration is again performed using a strong stability preserving RK3 method. Far-field boundary conditions and linear elements are used. Details of the FOM are presented in Table 5.

6.2.2. Solution of the Full-Order Model and Construction of the ROM Trial Space

Flow over a cylinder at Re=$100, 200$, and 300, where Re=$\rho_\infty U_\infty D/\mu$ is the Reynolds number, are considered. These Reynolds numbers give rise to the well studied von Kármán vortex street. Figure 7 shows the FOM solution at Re=100 for several time instances to illustrate the vortex street.

The FOM is used to construct the trial spaces used in the ROM simulations. The process used to construct these trial spaces is as follows:

1. Initialize FOM simulations at Reynold's numbers of Re=$100, 200$, and 300. The Reynold's number is controlled by raising or lowering the viscosity.
2. Time-integrate the FOM at each Reynolds number until a statistically steady-state is reached.
3. Once the flow has statistically converged to a steady state, reset the time coordinate to be $t = 0$, and solve the FOM for $t \in [0, 100]$.
4. Take snapshots of the FOM solution obtained from Step 3 at every $t = 0.5$ time units over a time-window of $t \in [0, 100]$ time units, for a total of 200 snapshots at each Reynolds number. This time window corresponds to roughly two cycles of the vortex street, with 100 snapshots per cycle.
5. Assemble the snapshots from each case into one global snapshot matrix of dimension $N \times 600$. This snapshot matrix is used to construct the trial subspace through POD. Note that only one set of basis functions for all conserved variables is constructed.



(a) x-velocity at $t = 0.0$ **(b)** x-velocity at $t = 25.0$ **(c)** x-velocity at $t = 50.0$

Figure 7: Evolution of the von Kármán vortex street at Re=100.

6. Construct trial spaces of dimension $N \times 11$, $N \times 43$, and $N \times 87$. These subspace dimensions correspond to an energy criterion of $99, 99.9$, and 99.99%. The different trial spaces are summarized in Table 6.

6.2.3. *Solution of the Reduced-Order Models*

The G ROM, APG ROM, and LSPG ROMs are considered. Details on their implementation are as follows:

1. Galerkin ROM: The Galerkin ROM is evolved in time using both explicit and implicit time integrators. In the explicit case, a strong stability RK3 method is used. In the implicit case, Crank-Nicolson time integration is used. The non-linear algebraic system is solved using SciPy's Jacobian-Free Netwon-Krylov solver. LGMRES is employed as the linear solver. The convergence tolerance for the max-norm of the residual is set at the default ftol=6e-6.
2. Adjoint Petrov-Galerkin ROM: The APG ROM is evolved in time using the same time integrators as the Galerkin ROM. The extra term appearing in the APG model is computed via finite difference[6] with a step size of $\varepsilon =$1e-5. Unless noted otherwise, the memory length τ is selected to be $\tau = \frac{0.2}{\rho(\tilde{\mathbf{V}}^T \mathbf{J}[\tilde{\mathbf{u}}(0)]\tilde{\mathbf{V}})}$. The impact of τ on the numerical results is considered in the subsequent sections.
3. LSPG ROM: The LSPG ROM is formulated from an implicit Crank-Nicolson temporal discretization. The resulting non-linear least-squares problem is solved using SciPy's least-squares solver with the 'dogbox' method [71]. The tolerance on the change to the cost function is set at ftol=1e-8. The tolerance on the change to the generalized coordinates is set at xtol=1e-8. The SciPy least-squares solver is comparable in speed to our own least-squares solver that utilizes the Gauss-Newton method with a thin QR factorization to solve the least-squares problem. The SciPy solver, however, was observed to be more robust in driving down the residual than the basic Gauss-Newton method with QR factorization, presumably due to SciPy's inclusion of trust-regions, and hence results are reported with the SciPy solver.

All ROMs are initialized with the solution of the Re=100 FOM at time $t = 0$, the x-velocity of which is shown in Figure 7a.

6.2.4. *Reconstruction of Re=100 Case*

Reduced-order models of the Re=100 case are first considered. This case was explicitly used in the construction of the POD basis and tests the ability of the ROM to reconstruct previously "seen" dynamics.

[6]It is noted that, when used in conjunction with a JFNK solver that utilizes finite difference to approximate the action of the Jacobian on a vector, approximating the extra RHS term in APG via finite difference leads to computing the finite difference approximation of a finite difference approximation. While not observed in the examples presented here, this can have a detrimental effect on accuracy and/or convergence.

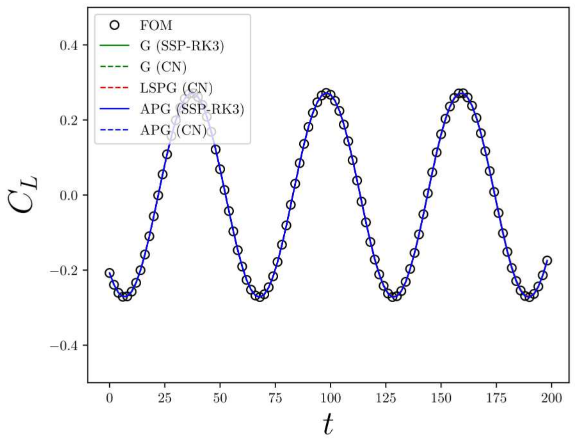
Basis #	Trial Basis Dimension (K)	Energy Criteria	τ (Adjoint Petrov-Galerkin)
1	11	99%	1.0
2	42	99.9%	0.3
3	86	99.99%	0.1

Table 6: Summary of the various basis dimensions used for Example 2

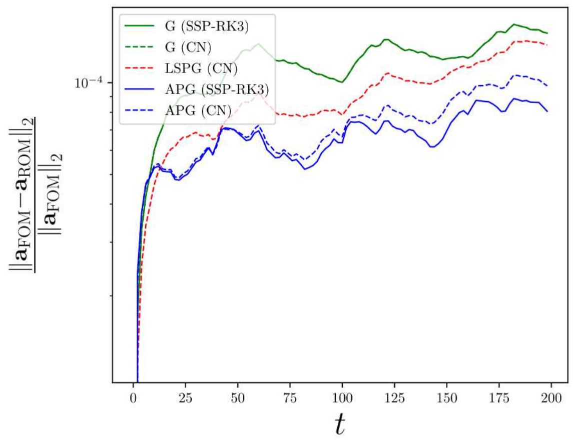
Unless otherwise noted, the default time-step for all ROMs is taken to be $\Delta t = 0.5$. The values of τ used in the APG ROMs are selected from the spectral radius heuristic and are given in Table 6. Figures 8a and 8b show the lift coefficient as well as the mean squared error (MSE) of the full-field ROM solutions for the G ROM, APG ROM, and LSPG ROMs for Basis #2, while Figure 8c shows the integrated MSE for $t \in [0, 200]$ for Basis #1, 2, and 3. Figure 8d shows the integrated error as a function of relative CPU time for the various ROMs. The relative CPU time is defined with respect to the FOM, which is integrated with an explicit time-step 100 times lower than the ROMs. The lift coefficients predicted by all three ROMs are seen to overlay the FOM results. The mean squared error shows that, for a given trial basis dimension, the APG ROM is more accurate than both the Galerkin and LSPG ROMs. This is the case for both explicit and implicit time integrators. As shown in Figure 8c, the APG ROM converges at a similar rate to the G ROM as the dimension of the trial space grows. For Basis #2 and #3, the implicit time-marching schemes are slightly less accurate than the explicit time-marching schemes. Finally, Figure 8d shows that, for a given CPU time, the G ROM with explicit time-marching produces the least error. The APG ROM with explicit time-marching is the second-best performing method. In the implicit case, both the G and APG ROMs lead to lower error at a given CPU time than LSPG. This decrease in cost is due to the fact that the G and APG ROMs utilize Jacobian-Free Netwon-Krylov solvers. As discussed in Section 5, it is much more challenging for LSPG to utilize Jacobian-Free methods. Due to the increased cost associated with implicit solvers, only explicit time integration is used for the G ROM and APG ROM beyond this point.

Next, we investigate the sensitivity of the different ROMs to the time-step size. Reduced-order models of the Re=100 case using Basis #2 are solved using time-steps of $\Delta t = \left[0.1, 0.2, 0.5, 1\right]$. Note that the largest time-step considered is 200 times larger than the FOM time-step, thus reducing the temporal dimensionality of the problem by 200 times. The mean-squared error of each ROM solution is shown in Figure 9. The G and APG ROMs are stable for all time-steps considered. Further, it is seen that varying the time-step has a minimal effect on the accuracy of the G and APG ROMs. In contrast, the accuracy of LSPG deteriorates if the time-step grows too large. This is due to the fact that, as shown in Ref. [23], the stabilization added by LSPG depends on the time-step size. Optimal accuracy of the LSPG method requires an intermediate time-step. The ability of the APG and G ROMs to take large time-steps without a significant degradation in accuracy allows for significant computational savings. This advantage is further amplified when large time-steps can be taken with an explicit solver, as is the case here. This will be discussed in more detail in Section 6.2.6.

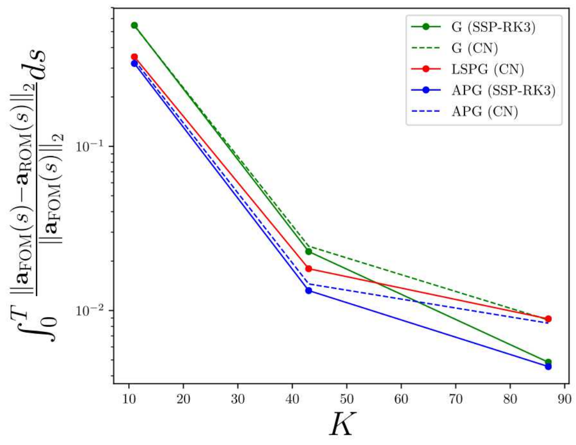
Lastly, we numerically investigate the sensitivity of APG to the parameter τ by running simulations for $\tau = [0.001, 0.01, 0.1, 0.3, 0.5, 1.]$. All simulations are run at $\Delta t = 0.5$. The results of the simulations are shown in Figure 10. It is seen that, for all values of τ, the APG ROM produces a better solution than the G ROM. The lowest error is observed for an intermediate value of τ, in which case the APG ROM leads to over a 50% reduction in error from the G ROM. As τ approaches zero, the APG ROM solution approaches the Galerkin ROM solution. Convergence plots for LSPG as a function of Δt are additionally shown in Figure 10b. It is seen that the optimal time-step in LSPG is similar to the optimal value of τ in APG.



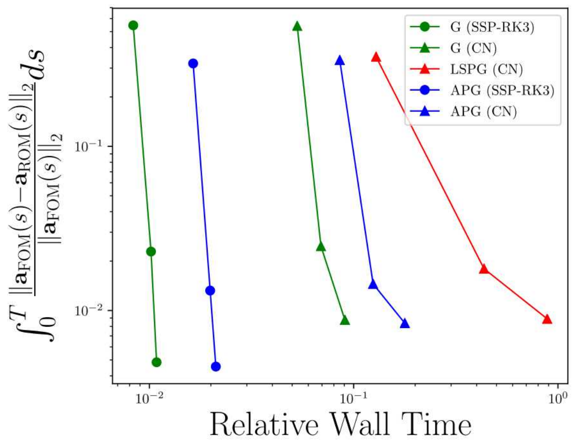
(a) Lift Coefficient as a function of time for Basis # 2.



(b) Normalized error as a function of time for Basis # 2.

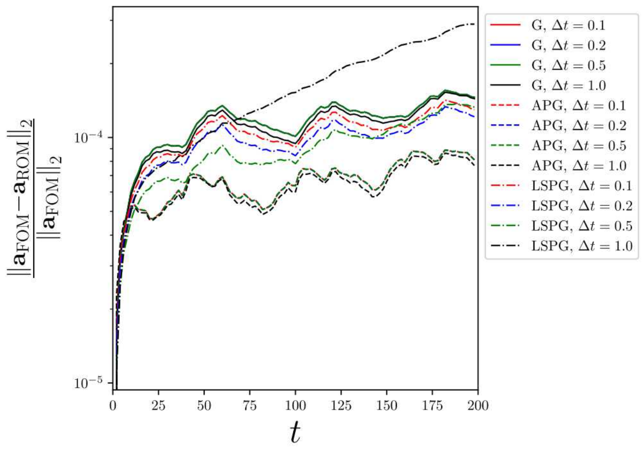


(c) Integrated normalized error vs trial subspace dimension.

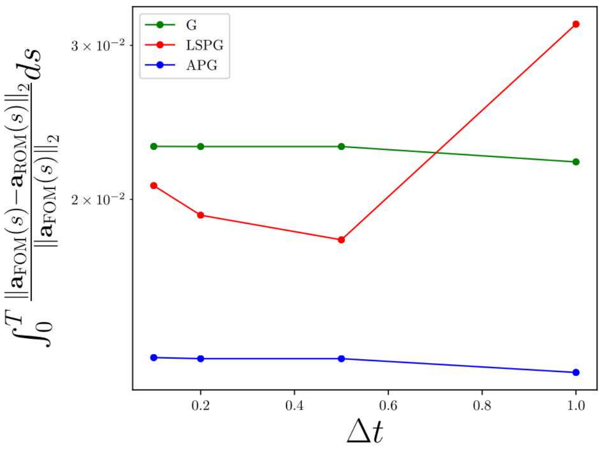


(d) Integrated normalized error vs relative CPU time.

Figure 8: ROM results for flow over cylinder at Re=100. The lift coefficient is defined as $C_L = \frac{2L}{\rho U_\infty^2 D}$, with L being the integrated force on the cylinder perpendicular to the free-stream velocity vector.

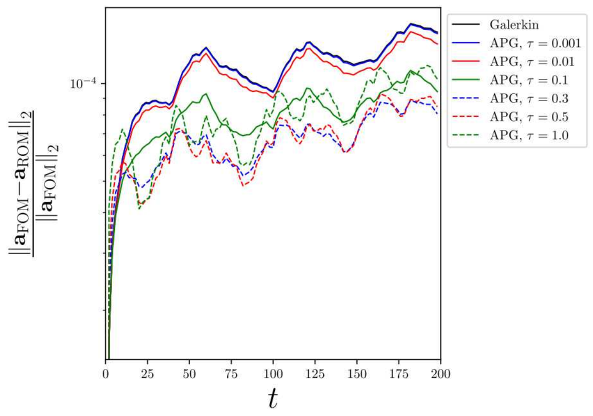


(a) Normalized error as a function of time.

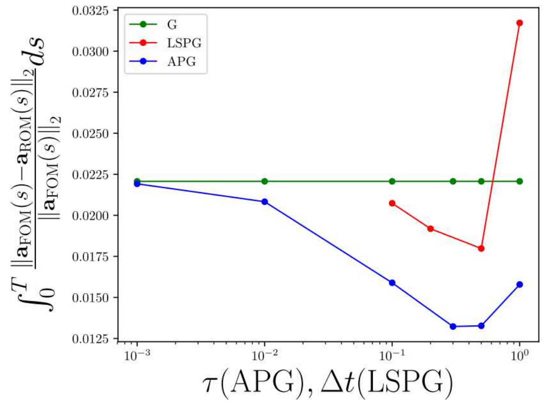


(b) Integrated normalized error as a function of the time-step.

Figure 9: Results for time-step study of flow over cylinder at Re= 100.



(a) Normalized error as a function of time



(b) Integrated normalized error as a function of τ. Results for how the time-step Δt impacts LSPG are included for reference.

Figure 10: Summary of numerical results investigating the impact of the parameter τ on the performance of the Adjoint Petrov-Galerkin method.

6.2.5. Parametric Study of Reynolds Number Dependence

Next, the ability of the different ROMs to interpolate between between different Reynolds numbers is studied. Simulations at Reynolds numbers of Re=100,150,200,250, and 300 with Basis #2 and #3 are performed. All cases are initialized from the Re=100 simulation. Note that the trial spaces in the ROMs were constructed from statistically steady-state FOM simulations of Re=100,200,300. The Reynolds number is modified by changing the viscosity.

Figure 11 summarizes the amplitude of the lift coefficient signal as well as the shedding frequency for the various methods. The values reported in Figure 11 are computed from the last 150s of the simulations[7]. The Galerkin ROM is seen to do poorly in predicting the lift coefficient amplitude for both Basis #2 and Basis #3. Unlike in the Re=100 case, enhancing the basis dimension does not improve the performance of the ROMs. Both the LSPG and APG ROMs are seen to offer much improved predictions over the Galerkin and ROM. This result is promising, as the ultimate goal of reduced-order modeling is to provide predictions in new regimes.

The results presented in this example highlight the shortcomings of the Galerkin ROM. To obtain results that are even qualitatively correct for the Re=$\{150,200,250,300\}$ cases, either APG or LSPG must be used. As reported in Figure 8d, explicit APG is over an order of magnitude faster than LSPG, and implicit APG with a JFNK solver is anywhere from 2x to 5x faster than LSPG. Therefore, APG is the best-performing method for this example.

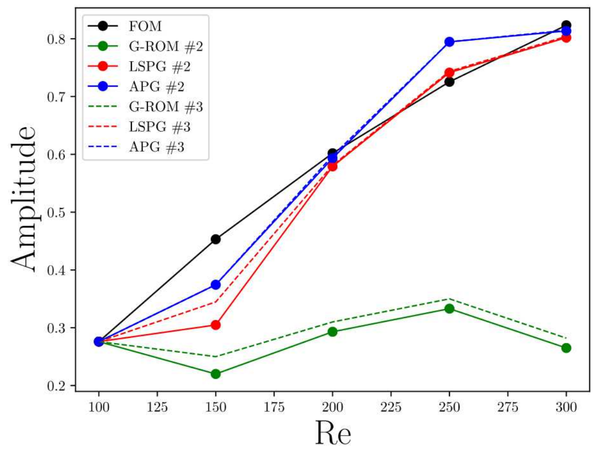
6.2.6. Flow Over Cylinder at Re=100 *with Hyper-Reduction*

The last example considered is again flow over a cylinder, but this time the reduced-order models are augmented with hyper-reduction. The purpose of this example is to examine the performance of the different reduced-order models when fully equipped with state-of-the-art reduction techniques. For hyper-reduction, an additional snapshot matrix of the right-hand side is generated. This additional snapshot matrix is generated by following steps one through five provided in Section 6.2.2. Hyper-reduction for the G ROM and the APG ROM is achieved through the Gappy POD method [55]. Hyper-reduction for LSPG is achieved through collocation using the same sampling points.[8] When augmented with hyper-reduction, the trial basis dimension (K), right-hand side basis dimension (r), and number of sample points (N_s) can impact the performance of the ROMs. Table 7 summarizes the various permutations of K, r, and N_s considered in this example. The sample points are selected through a QR factorization of the right-hand side snapshot matrix [72]. These sample points are then augmented such that they contain every conserved variable and quadrature point at the selected cells. The sample mesh corresponding to Basis numbers 4,5, and 6 in Table 7 is shown in Figure 12. Details on hyper-reduction and its implementation in our discontinuous Galerkin code are provided in Appendix A.

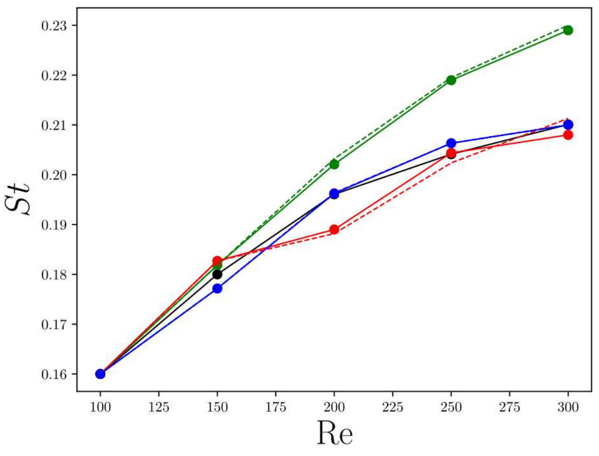
Flow at Re=100 is considered. All simulations are performed at the maximum stable time-step for a given basis dimension, as summarized in Table 7. Figure 13 shows the integrated error as a function of relative wall time for the various ROM techniques and basis numbers. All methods show significant computational speedup, with the G and APG ROMs producing wall-times up to 5000 times faster than the FOM while retaining a reasonable MSE. It is noted that the majority of this speed up is attributed to the increase in time-step size. For a given level of accuracy, LSPG is significantly more expensive than the G and APG ROMs. The reason for this expense is three-fold. First, LSPG is inherently implicit. For a given time-step size, an implicit step is more expensive than an explicit step. Second, LSPG requires an intermediate time-step for optimal accuracy. In this example, this intermediate time-step is small enough that the computational gains that could be obtained with an implicit method are negated. The third reason is that, for each Gauss-Newton iteration, LSPG requires the computation of the action of the Jacobian on

[7]Not all G ROMs reached a statistically steady state over the time window considered

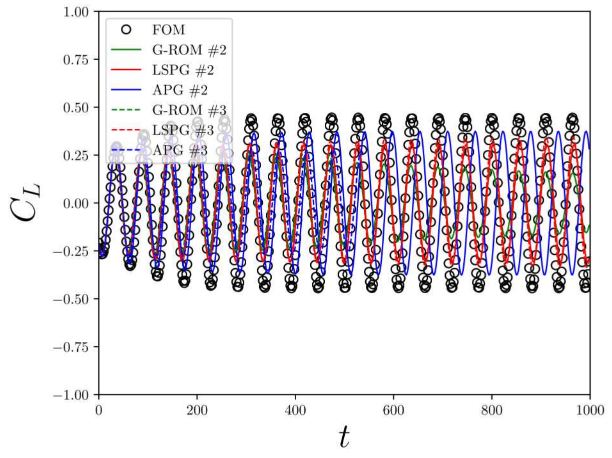
[8]It is noted that collocated LSPG out-performed the GNAT method for this example, and thus GNAT is not considered.



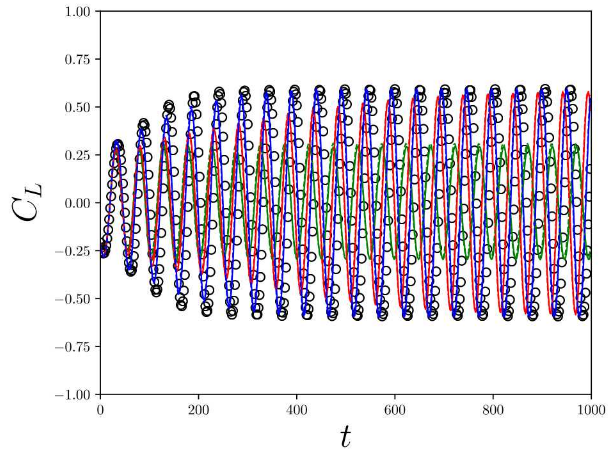
(a) Prediction for lift coefficient amplitudes



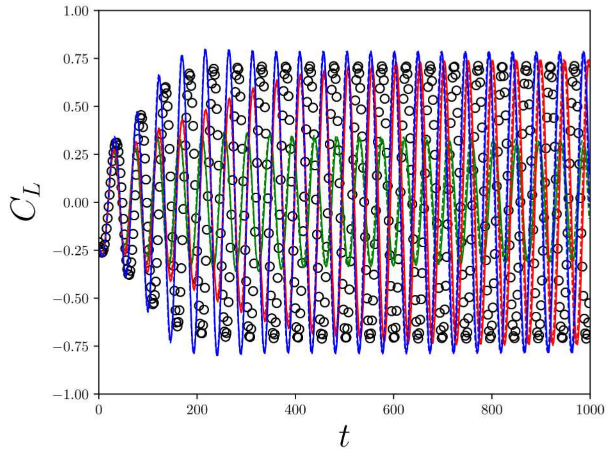
(b) Prediction for shedding frequency



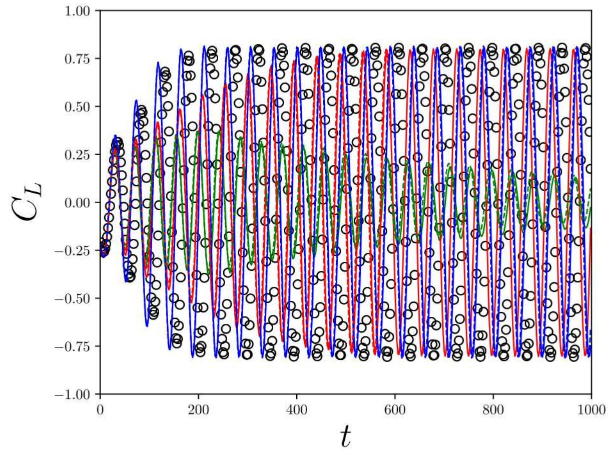
(c) Re=150



(d) Re=200



(e) Re=250

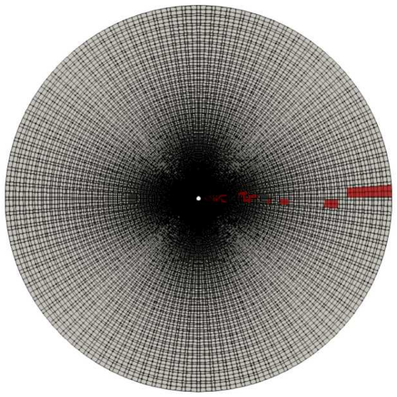


(f) Re=300

Figure 11: Summary of ROM simulations

Basis #	Trial Basis Dimension (K)	RHS Basis Dimension (r)	Sample Points (N_S)	Maximum Stable Δt
1	11	103	4230	4.0
2	42	103	4230	2.0
3	86	103	4230	1.0
4	11	268	8460	4.0
5	42	268	8460	2.0
6	86	268	8460	1.0

Table 7: Summary of the various reduced-order models evaluated on the flow over cylinder problem. The maximum stable Δt for each basis is reported for the SSP-RK3 explicit time-marching scheme and was empirically determined.


(a) Full sample mesh

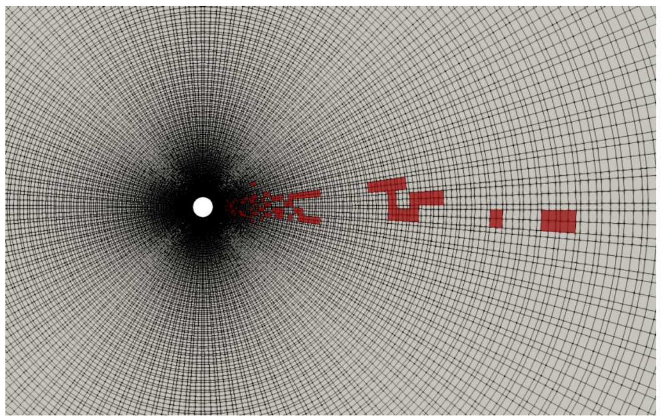
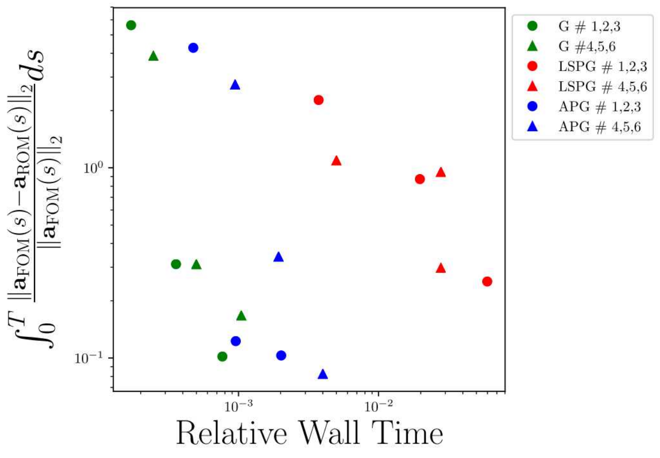

(b) Close up of wake

Figure 12: Mesh used for hyper-reduction. Cells colored in red are the sampled cells. Note that all conserved variables and quadrature points are computed within a cell.

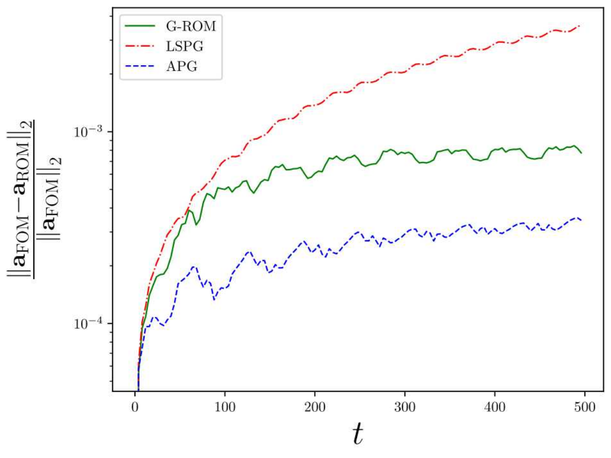
the trial space basis, $\tilde{\mathbf{V}}$. This expense can become significant for large basis dimensions as it requires the computation of a dense Jacobian. It is noted that it may be possible to achieve computational speed-ups in our implementation of LSPG through the development of a least-squares solver more tailored to LSPG, sparse Jacobian updates, or Jacobian approximation techniques.

7. Conclusion

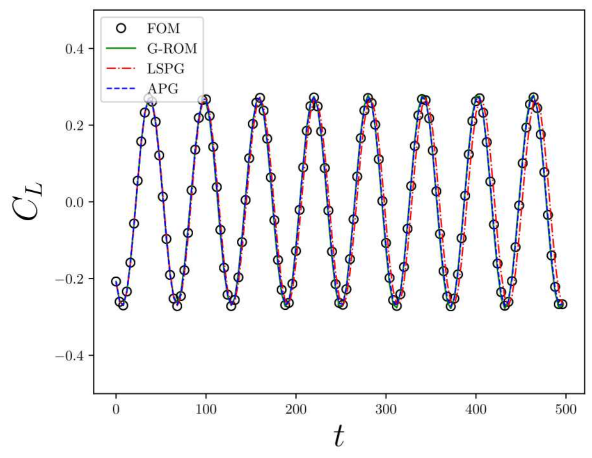
This work introduced the Adjoint Petrov–Galerkin method for non-linear model reduction. Derived from the variational multiscale method and Mori-Zwanzig formalism, the Adjoint Petrov–Galerkin method is a Petrov–Galerkin projection technique with a non-linear time-varying test basis. The method is designed to be applied at the semi-discrete level, i.e., after spatial discretization of a partial differential equation, and is compatible with both implicit and explicit time integration schemes. The method displays commonalities with the adjoint-stabilization method used in the finite element community as well as the Least-Squares Petrov–Galerkin approach used in non-linear model-order reduction. Theoretical error analysis was presented that showed conditions under which the Adjoint Petrov–Galerkin ROM may have lower *a priori* error bounds than the Galerkin ROM. The theoretical cost of the Adjoint Petrov–Galerkin method was considered for both explicit and implicit schemes, where it was shown to be approximately twice that of the Galerkin method. In the case of implicit time integration schemes, the Adjoint Petrov–Galerkin ROM was shown to be capable of being more efficient than Least-Squares Petrov–Galerkin when the non-linear system is solved via Jacobian-Free Newton-Krylov methods.



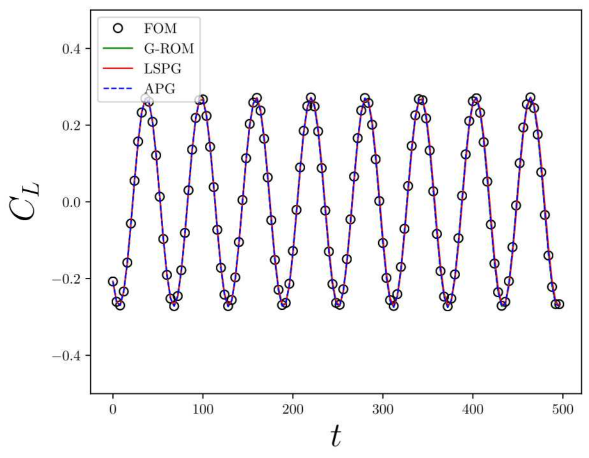
(a) Wall time vs normalized error for hyper-reduced ROMs.



(b) Normalized error as a function of time for Basis #5



(c) Lift coefficient as a function of time for Basis # 5.



(d) Lift coefficient as a function of time for Basis # 6.

Figure 13: Results for hyper-reduced ROMs at Re= 100.

Numerical experiments with the Adjoint Petrov–Galerkin, Galerkin, and Least-Squares Petrov–Galerkin method were presented for the Sod shock tube problem and viscous compressible flow over a cylinder parameterized by the Reynolds number. In all examples, the Adjoint Petrov–Galerkin method provided more accurate predictions than the Galerkin method for a fixed basis dimension. Improvements over the Least-Squares Petrov–Galerkin method were observed in most cases. In particular, the Adjoint Petrov–Galerkin method was shown to provide relatively accurate predictions for the cylinder flow at Reynolds numbers outside of the training set used to construct the POD basis. The Galerkin method, with both an equivalent and an enriched trial space, failed to produce accurate results in these cases. Additionally, numerical evidence showed a correlation between the spectral radius of the reduced Jacobian and the optimal value of the stabilization parameter appearing in the Adjoint Petrov–Galerkin method.

When augmented with hyper-reduction, the Adjoint Petrov–Galerkin ROM was shown to be capable of producing accurate predictions within the POD training set with computational speedups up to 5000 times compared to the full-order models. This speed-up is a result of hyper-reduction of the right-hand side, as well as the ability to use explicit time integration schemes at large time-steps. A study of the Pareto front for simulation error versus relative wall time showed that, for the compressible cylinder problem, the Adjoint Petrov–Galerkin ROM is competitive with the Galerkin ROM, and more efficient than the LSPG ROM for the problems considered.

8. Acknowledgements

The authors acknowledge support from the US Air Force Office of Scientific Research through the Center of Excellence Grant FA9550-17-1-0195 (Tech. Monitors: Mitat Birkan & Fariba Fahroo) and the project LES Modeling of Non-local effects using Statistical Coarse-graining (Tech. Monitors: Jean-Luc Cambier & Fariba Fahroo). E. Parish acknowledges an appointment to the Sandia National Laboratories John von Neumann fellowship. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA-0003525.

Appendix A Hyper-reduction for the Adjoint Petrov–Galerkin Reduced-Order Model

In the numerical solution of non-linear dynamical systems, the evaluation of the non-linear right-hand side term usually accounts for a large portion (if not the majority) of the computational cost. Equation 12 shows that standard projection-based ROMs are incapable of reducing this cost, as the evaluation of $\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}})$ still scales with the number of degrees of freedom N. If this issue is not addressed, and there is no reduction in temporal dimensionality, the ROM will typically be *more* expensive than the FOM due to additional matrix-vector products from projection onto the reduced-order space. Techniques for overcoming this bottleneck are typically referred to as hyper-reduction methods. The thesis of hyper-reduction methods is that, instead of computing the entire right-hand side vector, only a few entries are calculated. The missing entries can either be ignored (as done in collocation methods) or reconstructed (as done in the discrete interpolation and gappy POD methods). This section outlines the Gappy POD method, selection of the sampling points through QR factorization, and the algorithm for the hyper-reduced Adjoint Petrov–Galerkin ROM.

A.1 Gappy POD

The gappy POD method [55] seeks to find an approximation for $\mathbf{R}(\cdot)$ that evaluates the right-hand side term at a reduced number of spatial points $r \ll N$. This is achieved through the construction of a trial space for the right-hand side and least-squares reconstruction of a sampled signal. The offline steps required in the Gappy POD method are given in Algorithm 4.

Algorithm 4 Algorithm for the offline steps required for hyper-reduction via gappy POD.

Offline Steps:

1. Compute the full-order solution, storing n_t time snapshots in the following matrices,

$$\text{Right-hand side snapshots: } \mathbf{F} = [\mathbf{R}(\mathbf{u}_1) \quad \mathbf{R}(\mathbf{u}_2) \quad \dots \quad \mathbf{R}(\mathbf{u}_{n_t})] \in \mathbb{R}^{N \times n_t}$$

2. Compute the right-hand side POD basis $\mathbf{U} \in \mathbb{R}^{N \times r}$, from $\mathbf{F}$.
3. Compute the sampling point matrix $\mathbf{P} = [\mathbf{e}_{p_1} \quad \mathbf{e}_{p2} \quad \dots \mathbf{e}_{p_r}] \in \mathbb{R}^{N_p \times N}$, where $\mathbf{e}_i$ is the ith cannonical unit vector and N_p is the number of sampling points.
4. Compute the stencil matrix $\mathbf{P}_s = [\mathbf{e}_{s_1} \quad \mathbf{e}_{s2} \quad \dots \mathbf{e}_{s_r}] \in \mathbb{R}^{N_s \times N}$, where $\mathbf{e}_i$ is the ith cannonical unit vector and N_s is the number of stencil points required to reconstruct the residual at the sample points. Note $N_s \geq N_p$.
5. Compute the least-squares reconstruction matrix: $\left[\mathbf{P}^T\mathbf{U}\right]^+$, where the superscript $+$ denotes the pseudo-inverse. Note that this matrix corresponds to the solution of the least-squares problem for a gappy signal $\mathbf{f} \in \mathbb{R}^N$:

$$\mathbf{a_f} = \underset{\mathbf{b} \in \mathbb{R}^r}{\operatorname{argmin}} ||\mathbf{P}^T\mathbf{U}\mathbf{b} - \mathbf{P}^T\mathbf{f}||,$$

which has the solution,

$$\mathbf{a_f} = \left[\mathbf{P}^T\mathbf{U}\right]^+ \mathbf{f}.$$

Note that, because the gappy POD approximation of the right-hand side only samples the right-hand side term at N_p spatial points, the cost of the ROM no longer scales with the full-order degrees of freedom N, but instead with the number of POD basis modes K and the number of sample points N_p. Thus, the cost of evaluating the full right-hand side may be drastically reduced at the price of storing another snapshot matrix $\mathbf{F}$ and computing another POD basis $\mathbf{U}$ in the offline stage of computation. Furthermore, the product $\tilde{\mathbf{V}}^T\mathbf{U}\left[\mathbf{P}^T\mathbf{U}\right]^+$ may be precomputed during the offline stage if both $\mathbf{P}$ and $\mathbf{U}$ remain static throughout the simulation. This results in an relatively small $K \times N_p$ matrix. As such, an increase in offline computational cost may produce a significant decrease in online computational cost.

A.2 Selection of Sampling Points

Step 3 in Algorithm 4 requires the construction of the sampling point matrix, for which several methods exist. In the discrete interpolation method proposed by Chaturantabut and Sorensen [57], the sample points are selected inductively from the basis $\mathbf{U}$, based on an error measure between the basis vectors and approximations of the basis vectors via interpolation. The method proposed by Drmac and Gugercin [72] leverages the rank-revealing QR factorization to compute $\mathbf{P}$. Dynamic updates of the $\mathbf{U}$ and $\mathbf{P}$ via periodic sampling of the full-order right-hand side term is even possible via the methods developed by Peherstorfer and Willcox [14].

The 2D compressible cylinder simulations presented in this manuscript uses a modified version of the rank-revealing QR factorization proposed in [72] to obtain the sampling points. The modifications are added to enhance the stability and accuracy of the hyper-reduced ROM within the discontinuous Galerkin method. Algorithm 5 outlines the steps used in this manuscript to compute the sampling points.

Algorithm 5 Algorithm for QR-factorization-based selection of sampling matrix.

Input: Right-hand side POD basis $\mathbf{U} \in \mathbb{R}^{N\times r}$
Output: Sampling matrix $\mathbf{P}$
Steps:

1. Transpose the right-hand side POD basis, $\mathbf{U}' = \mathbf{U}^T$
2. Compute the rank-revealing QR factorization of $\mathbf{U}'$, generating a permutation matrix $\Gamma \in \mathbb{R}^{N\times N}$, unitary $\mathbf{Q} \in \mathbb{R}^{r\times r}$, and orthonormal $\mathbf{R} \in \mathbb{R}^{r\times N}$ such that

$$\mathbf{U}'\Gamma = \mathbf{QR}$$

   Details on computing rank-revealing QR decompositions can be found in [73], though many math libraries include optimized routines for this operation.
3. From the permutation matrix Γ, select the first r columns to form the interpolation point matrix $\mathbf{P}$.
4. When applied to systems of equations, sampling approaches that select only specific indices of the residual can be inaccurate due to the fact that, at a given cell, the residual of all unknowns at that cell may not be calculated. This issue is further exacerbated in the discontinuous Galerkin method, where each cell has a number of quadrature points. As such, we perform the additional step:
   (a) Augment the sampling point matrix, $\mathbf{P}$, with additional columns such that all unknowns are computed at the mesh cells selected by Step 3. In the present context, these additional unknowns correspond to each conserved variable and quadrature point in the selected cells. This step leads to $N_p > r$.

A.3 Hyper-Reduction of the Adjoint Petrov–Galerkin ROM

Lastly, the online steps required for an explicit Euler update to the Adjoint Petrov–Galerkin ROM with Gappy POD hyper-reduction is provided in Algorithm 6. It is worth noting that Step 3 in Algorithm 6 requires one to reconstruct the right-hand side at the stencil points. Hyper-reduction via a standard collocation method, which provides no means to reconstruct the right-hand side, is thus not compatable with the Adjoint Petrov–Galerkin ROM.

Appendix B POD Basis Construction

For the 1D Euler case detailed in this manuscript, the procedure for constructing separate POD bases for each conserved variables is as follows:

1. Run the full-order model for $t \in (0,1)$ at a time-step of $\Delta t = 0.0005$. The state vector is saved at every other time step to create 1000 state snapshots.
2. Collect the snapshots for each state into three state snapshot matrices:

$$\mathbf{S}_\rho = \begin{bmatrix}\rho_1 & \rho_2 & \dots & \rho_{1000}\end{bmatrix}, \mathbf{S}_{\rho u} = \begin{bmatrix}\rho\mathbf{u}_1 & \rho\mathbf{u}_2 & \dots & \rho\mathbf{u}_{1000}\end{bmatrix}, \mathbf{S}_{\rho E} = \begin{bmatrix}\rho\mathbf{E}_1 & \rho\mathbf{E}_2 & \dots & \rho\mathbf{E}_{1000}\end{bmatrix},$$

   where $\rho_i, \rho\mathbf{u}_i, \rho\mathbf{E}_i \in \mathbb{R}^{1000}$.
3. Compute the singular-value decomposition (SVD) of each snapshot matrix, e.g. for $\mathbf{S}_\rho$,

$$\mathbf{S}_\rho \overset{\text{SVD}}{=} \mathbf{V}_\rho \Sigma_\rho \mathbf{U}_\rho^T.$$

   The columns of $\mathbf{V}_\rho$ and $\mathbf{U}_\rho$ are the left and right singular vectors of $\mathbf{S}_\rho$, respectively. Σ_ρ is a diagonal matrix of the singular values of $\mathbf{S}_\rho$. The columns of $\mathbf{V}_\rho$ form a basis for the solution space of ρ_i.
   Remarks

Algorithm 6 Algorithm for an explicit Euler update for the Adjoint Petrov–Galerkin ROM with gappy POD hyper-reduction.

Input: $\tilde{\mathbf{a}}^n$
Output: $\tilde{\mathbf{a}}^{n+1}$
Online steps at time-step n:

1. Compute the state at the stencil points: $\tilde{\mathbf{u}}_s^n = \mathbf{P}_s^T \tilde{\mathbf{V}} \tilde{\mathbf{a}}^n$, with $\tilde{\mathbf{u}}_s^n \in \mathbb{R}^{N_s}$
2. Compute the generalized coordinates to the right-hand side evaluation via,
$$\mathbf{a}_{\mathbf{R}}^n = \left[\mathbf{P}^T \mathbf{U}\right]^+ \mathbf{P}^T \mathbf{R}(\tilde{\mathbf{u}}_s^n),$$
with $\mathbf{a}_{\mathbf{R}}^n \in \mathbb{R}^r$. Note that the product $\mathbf{P}^T \mathbf{R}(\tilde{\mathbf{u}}_s^n)$ requires computing $\mathbf{R}(\tilde{\mathbf{u}}_s^n)$ *only at the sample points*, as given by the unit vectors stored in $\mathbf{P}$.
3. Reconstruct the right-hand side at the stencil points: $\overline{\mathbf{R}_s(\tilde{\mathbf{u}}_s^n)} = \mathbf{P}_s^T \mathbf{U} \mathbf{a}_{\mathbf{R}}^n$
4. Compute the orthogonal projection of the approximated right-hand side at the stencil points:
$$\Pi' \overline{\mathbf{R}_s(\tilde{\mathbf{u}}^n)} = \overline{\mathbf{R}_s(\tilde{\mathbf{u}}^n)} - \tilde{\mathbf{V}} \tilde{\mathbf{V}}^T \overline{\mathbf{R}_s(\tilde{\mathbf{u}}^n)}$$
5. Compute the generalized coordintes for the action of the Jacobian on $\Pi' \overline{\mathbf{R}_s(\tilde{\mathbf{u}}^n)}$ at the sample points using either finite difference or exact linearization. For finite difference:
$$\mathbf{a}_{\mathbf{J}}^n \approx \frac{1}{\varepsilon} \left[\mathbf{P}^T \mathbf{U}\right]^+ \mathbf{P}^T \left[\mathbf{R}_s\left(\tilde{\mathbf{u}}_s^n + \varepsilon \Pi' \overline{\mathbf{R}_s}(\tilde{\mathbf{u}}_s^n)\right) - \mathbf{R}_s(\tilde{\mathbf{u}}_s^n)\right],$$
with $\mathbf{a}_{\mathbf{J}}^n \in \mathbb{R}^r$. Note ε is a small constant value, usually $\sim O(10^{-5})$.
6. Compute the combined sampled right-hand side: $\tilde{\mathbf{V}}^T \mathbf{U} \left[\mathbf{a}_{\mathbf{R}}^n + \tau \mathbf{a}_{\mathbf{J}}^n\right]$
7. Update the state: $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}^n + \Delta t \tilde{\mathbf{V}}^T \mathbf{U} \left[\mathbf{a}_{\mathbf{R}}^n + \tau \mathbf{a}_{\mathbf{J}}^n\right]$

   (a) In this example, a separate basis is computed for each conserved quantity. It is also possible to construct a global basis by stacking $\mathbf{S}_\rho$, $\mathbf{S}_{\rho\mathbf{u}}$, and $\mathbf{S}_{\rho\mathbf{E}}$ into one snapshot matrix and computing one "global" SVD.

4. Decompose each basis into bases for the resolved and unresolved scales by selecting the first K columns and last $1000-K$ columns, respectively, e.g.

$$\mathbf{V}_\rho = \left[\tilde{\mathbf{V}}_\rho \quad ; \quad \mathbf{V}'_\rho\right],$$

   where $\tilde{\mathbf{V}}_\rho \in \mathbb{R}^{1000\times K}$ and $\mathbf{V}' \in \mathbb{R}^{1000\times 1000-K}$.
   Remarks
   (a) Each basis vector is orthogonal to the others, hence the coarse and fine-scales are orthogonal.
   (b) In this example, we have selected 1000 snapshots such that the column space of $\mathbf{V}$ spans $\mathcal{V}$. In general, this is not the case. As the APG method requires no processing of the fine-scale basis functions, however, this is not an issue.

5. Construct a global coarse-scale basis,

$$\tilde{\mathbf{V}} = \begin{bmatrix} \tilde{\mathbf{V}}_\rho & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{V}}_{\rho u} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{V}}_{\rho\mathbf{E}} \end{bmatrix}.$$

Appendix C Algorithms for the Galerkin and LSPG ROMs

Section 5 presented an analysis on the computational cost of the Adjoint Petrov–Galerkin ROM. This appendix presents similar algorithms and FLOP counts for the Galerkin and LSPG ROMs. The following algorithms and FLOP counts are reported:

1. An explicit Euler update to the Galerkin ROM (Algorithm 7, Table 8).
2. An implicit Euler update to the Galerkin ROM using Newton's method with Gaussian elimination (Algorithm 8, Table 9).
3. An implicit Euler update to the Least-Squares Petrov–Galerkin ROM using the Gauss-Newton method with Gaussian elimination (Algorithm 9, Table 10).

Algorithm 7 Algorithm for an explicit Euler update for the Galerkin ROM.

```
Input: $\tilde{\mathbf{a}}^n$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
    1. Compute the state from the generalized coordinates, $\tilde{\mathbf{u}}^n = \tilde{\mathbf{V}}\tilde{\mathbf{a}}^n$
    2. Compute the right-hand side from the state, $\mathbf{R}(\tilde{\mathbf{u}}^n)$
    3. Project the right-hand side, $\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}^n)$
    4. Update the state $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}^n + \Delta t\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}^n)$
```

Step in Algorithm 7	Approximate FLOPs
1	$2NK-N$
2	ωN
3	$2NK-K$
4	$2K$
Total	$4NK+(\omega-1)N+K$

Table 8: Approximate floating-point operations for an explicit Euler update to the Galerkin method reported in Algorithm 7.

Algorithm 8 Algorithm for an implicit Euler update for the Galerkin ROM using Newton's Method with Gaussian Elimination

```
Input: $\tilde{\mathbf{a}}^n$, residual tolerance $\xi$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
  1. Set initial guess, $\tilde{\mathbf{a}}_k$
  2. Loop while $\mathbf{r}^k > \xi$
      (a) Compute the state from the generalized coordinates, $\tilde{\mathbf{u}}_k = \tilde{\mathbf{V}}\tilde{\mathbf{a}}_k$
      (b) Compute the right-hand side from the full state, $\mathbf{R}(\tilde{\mathbf{u}}_k)$
      (c) Project the right-hand side, $\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{u}}_k)$
      (d) Compute the Galerkin residual, $\mathbf{r}_G(\tilde{\mathbf{a}}_k) = \tilde{\mathbf{a}}_k - \tilde{\mathbf{a}}^n - \Delta t\tilde{\mathbf{V}}^T\mathbf{R}(\tilde{\mathbf{V}}\tilde{\mathbf{a}}_k)$
      (e) Compute the residual Jacobian, $\frac{\partial\mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial\tilde{\mathbf{a}}_k}$
      (f) Solve the linear system via Gaussian Elimination: $\frac{\partial\mathbf{r}(\tilde{\mathbf{a}}_k)}{\partial\tilde{\mathbf{a}}_k}\Delta\tilde{\mathbf{a}} = -\mathbf{r}(\tilde{\mathbf{a}}_k)$
      (g) Update the state: $\tilde{\mathbf{a}}_{k+1} = \tilde{\mathbf{a}}_k + \Delta\tilde{\mathbf{a}}$
      (h) $k = k+1$
  3. Set final state, $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}_k$
```

Step in Algorithm 2	Approximate FLOPs
2a	$2NK-N$
2b	ωN
2c	$2NK-K$
2d	$3K$
2e	$4NK^2+(\omega-1)NK+2K^2$
2f	K^3
2g	K
Total	$(\omega-1)N+3K+(\omega+3)NK+2K^2+4NK^2+K^3$

Table 9: Approximate floating-point operations for one Newton iteration for the implicit Euler update to the Galerkin method reported in Algorithm 8.

Algorithm 9 Algorithm for an implicit Euler update for the LSPG ROM using a Gauss-Newton method with Gaussian Elimination

```
Input: $\tilde{\mathbf{a}}^n$, residual tolerance $\xi$
Output: $\tilde{\mathbf{a}}^{n+1}$
Steps:
  1. Set initial guess, $\tilde{\mathbf{a}}_k$
  2. Loop while $\mathbf{r}_k > \xi$
     (a) Compute the state from the generalized coordinates, $\tilde{\mathbf{u}}_k = \tilde{\mathbf{V}}\tilde{\mathbf{a}}_k$
     (b) Compute the right-hand side from the full state, $\mathbf{R}(\tilde{\mathbf{u}}_k)$
     (c) Compute the residual, $\mathbf{r}(\tilde{\mathbf{u}}_k) = \tilde{\mathbf{u}}_k - \tilde{\mathbf{u}}^n - \Delta t \mathbf{R}(\tilde{\mathbf{u}}_k)$
     (d) Compute the test basis, $\mathbf{W}_k = \frac{\partial \mathbf{r}(\tilde{\mathbf{u}}_k)}{\partial \tilde{\mathbf{u}}_k}\tilde{\mathbf{V}} = \frac{\partial \mathbf{r}(\tilde{\mathbf{u}}_k)}{\partial \tilde{\mathbf{a}}_k}$
     (e) Compute the product, $\tilde{\mathbf{W}}_k^T \tilde{\mathbf{W}}_k$
     (f) Project the residual onto the test space, $\tilde{\mathbf{W}}^T \mathbf{r}(\tilde{\mathbf{a}}_k)$
     (g) Solve $\tilde{\mathbf{W}}^T \tilde{\mathbf{W}} \Delta\tilde{\mathbf{a}} = -\tilde{\mathbf{W}}^T \mathbf{r}(\tilde{\mathbf{a}}_k)$ for $\Delta\tilde{\mathbf{a}}$ via Gaussian elimination
     (h) Update solution, $\tilde{\mathbf{a}}_{k+1} = \tilde{\mathbf{a}}_k + \Delta\tilde{\mathbf{a}}$
     (i) k = k + 1
  3. Set final state, $\tilde{\mathbf{a}}^{n+1} = \tilde{\mathbf{a}}_k$
```

Step in Algorithm 9	Approximate FLOPs
2a	$2NK - N$
2b	ωN
2c	$3N$
2d	$(\omega+2)NK + 2NK^2$
2e	$2NK^2 - K^2$
2f	$2NK - K$
2g	K^3
2h	K
Newton Iteration Total	$(\omega+2)N + (\omega+6)NK - K^2 + 4NK^2 + K^3$

Table 10: Approximate floating-point operations for one Newton iteration for the implicit Euler update to the LSPG method reported in Algorithm 9.

References

[1] Moore, B., "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, Vol. 26, No. 1, February 1981, pp. 17–32.

[2] Mullis, C. T. and Roberts, R. A., "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Transactions on Circuits and Systems*, Vol. 23, 1976, pp. 551–562.

[3] Pillage, L. T., Huang, X., and Rohrer, R. A., "Asymptotic Waveform Evaluation for Timing Analysis," *Proceedings of the 26th ACM/IEEE Design Automation Conference*, DAC '89, ACM, New York, NY, USA, 1989, pp. 634–637.

[4] Hesthaven, J. S., Rozza, G., and Stamm, B., *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer International Publishing, Cham, 2016.

[5] Chatterjee, A., "An introduction to the proper orthogonal decomposition," *Current Science*, Vol. 78, No. 7, 2000, pp. 808–817.

[6] Kerschen, G., Golinval, J.-C., Vakakis, A. F., and Bergman, L. A., "The method of proper orthogonal decomposition for dynamical characterization and order reduction in mechanical systems: An overview," *Nonlinear Dynamics*, Vol. 41, 2005, pp. 147–169.

[7] Padhi, R. and Balakrishnan, S., "Proper orthogonal decomposition based optimal neurocontrol synthesis of a chemical reactor process using approximate dynamic programming," *Neural Networks*, Vol. 16, 2003, pp. 719–728.

[8] Cao, Y., Zhu, J., Navon, I., and Zhengdong, L., "A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition," *Int. J. Numer. Meth. Fluids*, Vol. 53, 2007, pp. 1571–1583.

[9] Rowley, C. W., Colonius, T., and Murray, R. M., "Model reduction for compressible flows using POD and Galerkin projection," *Physica D: Nonlinear Phenomena*, Vol. 189, No. 1-2, 2004, pp. 115–129.

[10] Huang, C., Duraisamy, K., and Merkle, C. L., "Challenges in Reduced Order Modeling of Reacing Flow," *2018 Joint Propulsion Conference, AIAA Propulsion and Energy Forum*, Cincinnati, Ohio, 2018.

[11] Kalashnikova, I., Arunajatesan, S., Barone, M. F., van Bloemen Waanders, B. G., and Fike, J. A., "Reduced Order Modeling for Prediction and Control of Large-Scale Systems," Report SAND2014-4693, Sandia, May 2014.

[12] Sirovich, L., "Turbulence and the dynamics of coherent structures. II: Symmetries and transformations," *Quarterly of Applied Mathematics*, Vol. 45, No. 3, 1987, pp. 573–582.

[13] Carlberg, K., "Adaptive h-refinement for reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 5, 2015, pp. 1192–1210.

[14] Peherstorfer, B. and Willcox, K., "Online adaptive model reduction for nonlinear systems via low-rank updates," *J. Sci. Comput.*, Vol. 37, 2015, pp. A2123–A2150.

[15] Abgrall, R. and Crisovan, R., "Model reduction using L1-norm minimization as an application to nonlinear hyperbolic problems," *International Journal for Numerical Methods in Fluids*, Vol. 87, No. 12, 2018, pp. 628–651.

[16] Balajewicz, M., Tezaur, I., and Dowell, E., "Minimal subspace rotation on the Stiefel manifold for stabilization and enhancement of projection-based reduced order models for the compressible NavierStokes equations," *Journal of Computational Physics*, Vol. 321, 2016, pp. 224 – 241.

[17] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Model Reduction for Large-Scale Systems with High-Dimensional Parametric Input Space," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 3270–3288.

[18] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Parametric Reduced-Order Models for Probabilistic Analysis of Unsteady Aerodynamic Applications," *AIAA Journal*, Vol. 46, No. 10, 2008, pp. 2520–2529.

[19] Rovas, D. V., *Reduced-basis output bound methods for parametrized partial differential equations*, Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[20] Carlberg, K., *Model Reduction of Nonlinear Mechanical Systems via Pptimal Projection and Tensor Approximation*, Ph.D. thesis, Stanford University, 2011.

[21] Bui-Thanh, T., *Model-constrained optimization methods for reduction of parameterized large-scale systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[22] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations," *Int. J. Numer. Methods Eng.*, Vol. 86, 2011, pp. 155–181.

[23] Carlberg, K., Barone, M., and Antil, H., "Galerkin v. least-squares Petrov-Galerkin projection in nonlinear model reduction," *Journal of Computational Physics*, Vol. 330, 2017, pp. 693–734.

[24] 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.

[25] Huang, C., Duraisamy, K., and Merkle, C., *Investigations and Improvement of Robustness of Reduced-Order Models of Reacting Flow*.

[26] Carlberg, K., Choi, Y., and Sargsyan, S., "Conservative model reduction for finite-volume models," *Journal of Computational Physics*, Vol. 371, 2018, pp. 280–314.

[27] Wang, Z., *Reduced-Order Modeling of Complex Engineering and Geophysical Flows: Analysis and Computations*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2012.

[28] Aubry, N., Holmes, P., Lumley, J. L., and Stone, E., "The dynamics of coherent structures in the wall region of a turbulent boundary layer," *J. Fluid Mech.*, Vol. 192, 1988, pp. 115–173.

[29] Ullmann, S. and Lang, J., "A POD-Galerkin reduced model with updated coefficients for Smagorinsky LES," *J.C.F. Pereira, A. Sequeira (Eds.), V European Conference on Computational Fluid Dynamics, ECCOMAS CFD*, Lisbon, Portugal, 2010.

[30] Wang, Z., Akhtar, I., Borggaard, J., and Iliescu, T., "Two-level discretizations of nonlinear closure models for proper orthogonal decomposition," *Journal of Computational Physics*, Vol. 230, No. 1, 2011, pp. 126–146.

[31] Noack, B., Papas, P., and Monkewitz, P., "Low-dimensional Galerkin model of a laminar shear-layer," Report 2002-01, École Polytechnique Fédérale de Lausanne, 2002.

[32] San, O. and Iliescu, T., "A stabilized proper orthogonal decomposition reduced-order model for large scale quasigeostrophic ocean circulation," *Adv Comput Math*, Vol. 41, No. 1, 2015, pp. 1289–1319.

[33] Bergmann, M., Bruneaua, C., and Iollo, A., "Enablers for robust POD models," *Journal of Computational Physics*, Vol. 228, No. 1, 2009, pp. 516–538.

[34] Stabile, G., Ballarin, F., Zuccarino, G., and Rozza, G., "A reduced order variational multiscale approach for turbulent flows," *Advances in Computational Mathematics*, Jun 2019.

[35] San, O. and Iliescu, T., "Proper Orthogonal Decomposition Closure Models for Fluid Flows: Burgers Equation," *Comput. Methods Appl. Mech. Engrg*, Vol. 5, No. 3, 2014, pp. 217–237.

[36] Iliescu, T. and Wang, Z., "Variational Multiscale Proper Orthogonal Decomposition: Navier-Stokes Equations," *Numerical Methods for Partial Differential Equations*, Vol. 30, No. 2, 2014, pp. 641–663.

[37] Caiazzo, A., Iliescu, T., John, V., and Schyschlowa, S., "A numerical investigation of velocity-pressure reduced models for incompressible flows," *Journal of Computational Physics*, Vol. 259, No. 1, 2014, pp. 598–616.

[38] Mori, H., "Transport, collective motion, and Brownian motion," *Prog. Theoret. Phys*, Vol. 33, No. 3, 1965, pp. 423–455.

[39] Zwanzig, R., "Nonlinear generalized Langevin equations," *Journal of Statistical Physics*, 1973, pp. 215–220.

[40] Chorin, A. J., Hald, O., and Kupferman, R., "Optimal prediction and the Mori-Zwanzig representation of irreversible processes," *Proc. Natl Acad. Sci.*, Vol. 97, No. (doi:10.1073/pnas.97.7.2968), 2000, pp. 2968–2973.

[41] Chorin, A. J., Hald, O. H., and Kupferman, R., "Optimal prediction with memory," *Physica D: Nonlinear Phenomena*, Vol. 166, No. 3, 2002, pp. 239 – 257.

[42] Chorin, A. and Hald, O., *Stochastic Tools in Mathematics and Science*, Springer-Verlag, 2005.

[43] Chorin, A. and Stinis, P., "Problem reduction, renormalization, and memory," *Commun. Appl. Math. Comput. Sci.*, 2006, pp. 239–257.

[44] Hald, O. H. and Stinis, P., "Optimal prediction and the rate of decay for solutions of the Euler equations in two and three dimensions," *Proceedings of the National Academy of Sciences*, Vol. 104, No. 16, 2007, pp. 6527–6532.

[45] Stinis, P., "Higher order Mori-Zwanzig models for the Euler equations," *Multiscale Model. Simul.*, Vol. 6, No. 3, 2007, pp. 741–760.

[46] Stinis, P., "Renormalized reduced models for singular PDEs," *Commun. Appl. Math. Comput. Sci.*, Vol. 8, No. 1, 2013, pp. 39–66.

[47] Stinis, P., "Mori-Zwanzig reduced models for uncertainty quantification I: Parametric uncertainty," *arXiv:1211.4285*, 2012.

[48] Price, J. and Stinis, P., "Renormalized Reduced Order Models with Memory for Long Time Prediction," *Multiscale Modeling & Simulation*, Vol. 17, No. 1, 2019, pp. 68–91.

[49] Price, J. and Stinis, P., "Renormalization and blow-up for the 3D Euler equations," *arXiv:1805.08766v1*, 2018.

[50] Parish, E. and Duraisamy, K., "Reduced Order Modeling of Turbulent Flows Using Statistical Coarse-graining," *46th AIAA Fluid Dynamics Conference, AIAA AVIATION Forum*, Washington, D.C., June 2016.

[51] Parish, E. J. and Duraisamy, K., "Non-Markovian Closure Models for Large Eddy Simulation using the Mori-Zwanzig Formalism," *Physical Review Fluids*, Vol. 2, No. 1, 2017.

[52] Parish, E. J. and Duraisamy, K., "A dynamic subgrid scale model for Large Eddy Simulations based on the Mori-Zwanzig formalism," *Journal of Computational Physics*, Vol. 349, 2017, pp. 154–175.

[53] Gouasmi, A., Parish, E. J., and Duraisamy, K., "A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori-Zwanzig formalism," *Proceedings of The Royal Society A*, Vol. 473, No. 20170385, 2017.

[54] Parish, E. J. and Duraisamy, K., "A Unified Framework for Multiscale Modeling Using Mori-Zwanzig and the Variational Multiscale Method," *arXiv Preprint*, 2018.

[55] Everson, R. and Sirovich, L., "Karhunen-Lo'eve procedure for gappy data," *Journal of the Optical Society of America A*, Vol. 12, 1995, pp. 1657–1644.

[56] 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," *C. R. Acad. Sci. Paris*, Vol. 339, No. 9, 2004, pp. 667–672.

[57] Chaturantabut, S. and Sorensen, D. C., "Nonlinear Model Reduction via Discrete Empirical Interpolation," *SIAM J. Sci. Comput.*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[58] Brooks, A. N. and Hughes, T. J., "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Comput. Method Appl. M.*, Vol. 32, 1982, pp. 199–259.

[59] Hughes, T. J. and Tezduyar, T. E., "Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations," *Comput Method Appl. M.*, Vol. 45, 1984, pp. 217–284.

[60] Hughes, T. J., Franca, L. P., and Hulbert, G. M., “A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 73, No. 2, 1989, pp. 173 – 189.

[61] Hughes, T. J., “Multiscale phenomena: Green’s functions, the Dirichlet-to-Neumann formulation, subgridscale models, bubbles and the origins of stabilized methods,” *Comput. Methods Appl. Mech. Eng.*, Vol. 127, 1995, pp. 387–401.

[62] Zwanzig, R., *Nonequilibrium Statistical Mechanics*, Oxford University Press, 2001.

[63] Koopman, B., “Hamiltonian systems and transformations in Hilbert space,” *Proceedings of the National Academy of Sciences of the US A*, Vol. 17, No. 5, 1931, pp. 315–318.

[64] Barber, J. L., *Application of Optimal Prediction to Molecular Dynamics*, Ph.D. thesis, University of California, Berkeley, CA, 2004.

[65] Saad, Y. and Schultz, M., “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,” *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[66] Xu, W., Zheng, N., and Hayami, K., “Jacobian-Free Implicit Inner-Iteration Preconditioner for Nonlinear Least Squares Problems,” *Journal of Scientific Computing*, Vol. 68, No. 3, Sep 2016, pp. 1055–1081.

[67] Sod, G. A., “Survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws,” *Journal of Computational Physics*, Vol. 27, 1978, pp. 1–31.

[68] Roe, P., “Approximate Riemann solvers, parameter vectors and difference schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[69] Gottlieb, S., Shu, C.-W., and Tadmor, E., “Strong Stability-Preserving High-Order Time Discretization Methods,” *SIAM Review*, Vol. 43, No. 1, 2001, pp. 89–112.

[70] Bassi, F. and Rebay, S., “A High Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations,” *Journal of Computational Physics*, Vol. 131, 1997, pp. 267–279.

[71] Voglis, C. and Lagaris, I. E., “A Rectangular Trust Region Dogleg Approach for Unconstrained and Bound Constrained Nonlinear Optimization,” *WSEAS International Conference on Applied Mathematics.*, 2004.

[72] Drmac, Z. and Gugercin, S., “A new selection operator for the discrete empirical Interpolation method—Improved a priori error bound and extensions,” *J. Sci. Comput.*, Vol. 38, 2016, pp. A631–A648.

[73] Gu, M. and Eisenstat, S. C., “Efficient algorithms for computing a strong rank-revealing QR factorization,” *J. Sci. Comput.*, Vol. 17, 1996, pp. 848–869.

Time-series machine-learning error models for approximate solutions to parameterized dynamical systems

Eric J. Parish[a], Kevin T. Carlberg[a]

[a]*Sandia National Laboratories, 7011 East Ave, Livermore, CA 94550*

Abstract

This work proposes a machine-learning framework for modeling the error incurred by approximate solutions to parameterized dynamical systems. In particular, we extend the machine-learning error models (MLEM) framework proposed in Ref. [15] to dynamical systems. The proposed Time-Series Machine-Learning Error Modeling (T-MLEM) method constructs a regression model that maps features—which comprise error indicators that are derived from standard *a posteriori* error-quantification techniques—to a random variable for the approximate-solution error at each time instance. The proposed framework considers a wide range of candidate features, regression methods, and additive noise models. We consider primarily recursive regression techniques developed for time-series modeling, including both classical time-series models (e.g., autoregressive models) and recurrent neural networks (RNNs), but also analyze standard non-recursive regression techniques (e.g., feed-forward neural networks) for comparative purposes. Numerical experiments conducted on multiple benchmark problems illustrate that the long short-term memory (LSTM) neural network, which is a type of RNN, outperforms other methods and yields substantial improvements in error predictions over traditional approaches.

Keywords: error modeling, time-series modeling, recurrent neural networks, long short-term memory networks, parameterized dynamical systems, model reduction

1. Introduction

Myriad applications in engineering and science require many simulations of a (parameterized) dynamical-system model. Examples of such "many-query" problems include uncertainty propagation, design optimization, and statistical inference. When the cost of a single dynamical-system simulation is computationally expensive—as is the case for partial-differential-equation (PDE) models characterized by a fine spatiotemporal discretization—such many-query problems are intractable. In these cases, analysts often replace the original (high-fidelity) dynamical-system model with a surrogate model that can generate low-cost *approximate solutions*, which can in turn make the many-query problem computationally tractable. Examples of surrogate models include *inexact solutions* arising from early termination of an iterative method, *lower-fidelity models* characterized by simplifying physical assumptions (e.g., linearized dynamics, inviscid flow, elastic deformation) or a coarse discretization, and projection-based *reduced-order models (ROMs)* (e.g., Galerkin projection with a proper orthogonal decomposition basis).

The use of an approximate solution introduces error; as such, it is critical to quantify this error and properly account for it in the analysis underpinning the many-query problem. For this purpose, researchers have developed a wide range of methods for *a posteriori* error quantification. These efforts, which date back to the pioneering work of Babuška and Rheinboldt for finite element analysis [24, 3, 2], have resulted in a techniques that can be categorized as (1) *a posteriori* error bounds, (2) error indicators, and (3) error models.

Email addresses: `ejparis@sandia.gov` (Eric J. Parish), `ktcarlb@sandia.gov` (Kevin T. Carlberg)

A posteriori error bounds place bounds on the (normed) state or quantity-of-interest (QoI) error arising from the use of an approximate solution. These methods were originally developed in the finite-element [37, 36, 29] and model-reduction communities. In the reduced-basis community, Refs. [8, 39] derived *a posteriori* QoI error bounds for the reduced-basis method applied to linear elliptic and parabolic PDEs. Refs. [18, 42] extended these results to time-dependent linear parabolic PDEs. In the context of model reduction for nonlinear dynamical systems, Ref. [21] derived error bounds for the linearized Galerkin ROM, while Ref. [9] derived error bounds for Galerkin and least-squares Petrov-Galerkin (LSPG) ROMs. These error bounds are *residual-based*, meaning that they depend on the (dual) norm of the high-fidelity-model residual evaluated at the approximate solution. Such bounds are appealing because they can provide guaranteed, rigorous bounds on the errors of interest, thus providing a critical tool for certified predictions. However, these bounds often suffer from lack of sharpness, i.e., they are often orders-of-magnitude larger than the approximate-solution error itself. This is exacerbated in dynamical systems, where error bounds for many types of approximate solutions grow exponentially in time (see, e.g., Ref. [9]). In addition, these bounds typically require (difficult-to-compute) estimates or bounds of operator quantities such as Lipschitz and inf–sup constants. These drawbacks often limit the utility of *a posteriori* error bounds in practical applications.

Alternatively, *error indicators* are computable quantities that are *indicative* of the approximate-solution error. Error indicators typically do not provide unbiased estimates of the error, nor do they rigorously bound the error; instead, they are often correlated with the error, correspond to a low-order approximation of the error, and/or appear as terms in error bounds. The two most common error indicators are the (dual) norm of the high-fidelity-model residual evaluated at the approximate solution (i.e., the residual norm) and the dual-weighted residual. The residual norm is informative of the error, as it typically appears in *a posteriori* error bounds and is usually strongly correlated with the error; it is often used within greedy methods for snapshot collection [7, 6, 19, 48, 49] and when using ROMs for PDE-constrained optimization within a trust-region framework [52, 50, 51]. In contrast, dual-weighted residuals are derived from a Taylor-series approximation of the error, and are typically used for goal-oriented error estimation and mesh adaptation [28, 1, 46, 47]. Due to their practical utility, error indicators have been largely successful in quantifying and controlling errors through mesh adaptation for static problems (i.e., problems without time evolution). However, their success has been more limited for dynamical systems due to the fact that errors for such systems exhibit dependence on *non-local* quantities, i.e., the approximate-solution error at a given time instance depends on the past time history of the system. Thus, the residual norm at the current time instance is no longer directly indicative of the approximate-solution error; the error additionally depends on non-local quantities. This can be clearly seen from *a posteriori* error bounds, as the bound for the error at a given time instance depends on the residual norm at previous time instances. Similarly, dual-weighted-residual error estimation requires computing solutions to either the forward-in-time linearized sensitivity equations or the backward-in-time linearized adjoint equations. Not only does this incur significant computational and storage costs, it can also be linearly unstable and produce exploding gradients when applied to chaotic systems (although this can be mitigated by approaches such as least-squares shadowing [5, 44]). Further, it is often practically challenging to implement adjoint-based methods, e.g., due to the requirement of residual-Jacobian transposes, which are not always easily exposed.

Error models seek to model the state and QoI errors directly via regression techniques. The most popular such approach is the so-called "model-discrepancy" method of Kennedy and O'Hagan [25] (and related approaches [22, 13, 30, 16, 31, 53]). The model-discrepancy approach computes a Gaussian process (GP) that provides a mapping from any finite number of points in parameter space to a Gaussian random vector representing the approximate-solution error at those points. As such, this technique can be considered a regression approach, where the *regression model* is provided by a GP, the *features* are the system parameters, and the *response* is the approximate-solution error. Recently, researchers have pursued improvements over this technique by (1) considering a wider class of modern machine-learning regression methods than simply Gaussian processes (which do not scale well to high-dimensional feature spaces), and (2) considering more informative features than the system parameters alone. First, the reduced-order model error surrogates (ROMES) method [12] was proposed in the context of static problems. ROMES employs the same regression model (GP) as the model-discrepancy approach, but it employs different features; namely, it employs the aforementioned *error indicators* (i.e., residual norm, approximated dual-weighted residual) as features. Because these quantities are indicative of the error, they provide more informative features for predicting the error than the system parameters. Numerical experiments illustrated the ability of the ROMES method to produce significantly more accurate error predictions than the model-discrepancy approach. More recently, the ROMES method was applied to construct error models for the in-plane and out-of-plane state error incurred by reduced-order models for static problems [34]. Subsequent work [15] again considered static problems, but constructed machine-learning error models that considered a wide range of candidate regression models (e.g., neural networks, support vector machines, random forests) and error-indicator-based features (e.g., residual samples) in order to predict the approximate-solution error response. This work showed the ability of machine-learning methods to predict the error with near perfect accuracy across a range of static problems and approximate-solution types.

The construction of error models for dynamical systems is significantly more challenging than doing so for static systems, as the errors exhibit dependence on non-local quantities. Relevant work on constructing error models for dynamical systems accounts for the time-dependent nature of the approximate-solution error either (1) by using both time itself and time-lagged error indicators as features [45] or (2) by constructing a different regression method for each time instance or window [35]. In the first case, Ref. [45] proposes the error modeling via machine learning (EMML) method. EMML is a regression approach, where the regression model is provided by random forests or LASSO, the features correspond to application-specific quantities that include time itself and time-lagged quantities, and the response is the (relative) QoI or state error at a given time instance. While this work generated promising results on an oil–water subsurface flow problem, it suffers from a noticeable limitation: it treats error modeling as a classical (non-recursive) regression problem as opposed to a time-series-prediction problem. As such, it does not capture the intrinsic recursive structure of the error. In the second case, Ref. [35] proposes a regression model that is tantamount to applying the model-discrepancy method [25] to model the approximate-solution error at each time instance or window, as the regression model corresponds to a GP, the features are system parameters, and the response is the QoI error at a given time instance/window. Unfortunately, this approach also fails to capture non-local dependencies of the error, and—as with the original model-discrepancy method—does not use particularly informative features to predict the error.

The objective of this work is to construct accurate error models for dynamical systems by (1) applying the same machine-learning error modeling framework of Ref. [15], but (2) considering recursive regression techniques developed for time-series modeling, as this naturally enables the method to capture the non-local dependencies of the error. The proposed Time-Series Machine-Learning Error Modeling (T-MLEM) method, which views the error modeling problem from the time-series-prediction perspective, is characterized by a *regression model* provided by a recursive time-series-prediction model (e.g., autoregressive model, recurrent neural network [43]), *features* corresponding to time-local error indicators (e.g., residual samples), and a *response* corresponding to the normed state or QoI error at a given time instance. For comparative purposes, we also consider classical non-recursive regression methods (e.g., feed-forward neural networks) within the T-MLEM framework, which is analogous to the approach proposed in Ref. [45]. To produce a statistical model of the approximate-solution error, we also consider several noise models (e.g., Gaussian, Laplacian). The net result is a random variable for the error whose statistics can be used to assess the uncertainty in the error prediction.

The paper proceeds as follows. Section 2 sets the mathematical context by introducing parameterized dynamical systems, approximate solutions, and classical methods for error quantification. Section 3 outlines the proposed T-MLEM method, including the structure of the regression model, feature engineering, data generation, and training of the regression and noise models. Section 4 provides numerical experiments that apply the proposed framework to 1) the parameterized advection–diffusion equation, where the approximate solution is provided by a proper orthogonal decomposition (POD) ROM with Galerkin projection, 2) the parameterized shallow-water equations, where the approximate solution is again provided by a POD–Galerkin ROM, and 3) the parameterized Burgers' equation, where the approximate solution is provided by a coarse-mesh model. Finally, Section 5 provides conclusions.

2. Parametrized nonlinear dynamical systems

This work considers the high-fidelity model to be a parameterized dynamical system characterized by a parameterized system of nonlinear ordinary differential equations (ODEs) of the form

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t; \boldsymbol{\mu}), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0(\boldsymbol{\mu}), \qquad t \in [0, T], \tag{1}$$

where $T \in \mathbb{R}^+$ denotes the final time; $\boldsymbol{x} \equiv \boldsymbol{x}(t, \boldsymbol{\mu})$ with $\boldsymbol{x} : [0, T] \times \mathcal{D} \to \mathbb{R}^N$ denotes the state implicitly defined as the solution to the initial-value problem (1); $\boldsymbol{x}_0 : \mathcal{D} \to \mathbb{R}^N$ denotes the parameterized initial condition; $\boldsymbol{\mu} \in \mathcal{D} \subseteq \mathbb{R}^{N_\mu}$ denote parametric inputs to the system; and $\boldsymbol{f} : \mathbb{R}^N \times [0, T] \times \mathcal{D} \to \mathbb{R}^N$ denotes the parameterized velocity, which may be linear or nonlinear in its first argument. We refer to Eq. (1) as the full-order-model system of ordinary differential equations (FOM ODE), which may arise, for example, from the spatial discretization of a system of PDEs.

Numerically computing a solution to Eq. (1) requires applying a time-discretization method. In this work, we assume the use of a linear multistep method for this purpose. For notational simplicity, we assume a uniform time discretization characterized by N_t time instances $t^n = n\Delta t$, $n = 0, \ldots, N_t$ with time step $\Delta t = T/N_t$; however, the proposed methodology does not rely on this restriction. Applying a linear multistep method to discretize the FOM ODE (1) yields the following system of algebraic equations arising at the nth time instance, which we refer to as the FOM OΔE:

$$\boldsymbol{r}^n(\boldsymbol{x}^n; \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-k^n}, \boldsymbol{\mu}) = \boldsymbol{0}, \qquad n = 1, \ldots, N_t. \tag{2}$$

The discrete state $\boldsymbol{x}^n$ comprises the numerical approximation to $\boldsymbol{x}(t^n)$ and is implicitly defined as the solution to Eq. (2) given the discrete state at the previous k^n time instances and the parameters $\boldsymbol{\mu}$. Here, the discrete residual $\boldsymbol{r}^n : \mathbb{R}^N \otimes \mathbb{R}^{k^n+1} \times \mathcal{D} \to \mathbb{R}^N$ is defined as

$$\boldsymbol{r}^n : (\boldsymbol{w}; \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-k^n}, \boldsymbol{\mu}) \mapsto \alpha_0 \boldsymbol{w} - \Delta t \beta_0 \boldsymbol{f}(\boldsymbol{w}, t^n; \boldsymbol{\mu}) + \sum_{i=1}^{k^n} \alpha_i \boldsymbol{x}^{n-i} - \Delta t \sum_{i=1}^{k^n} \beta_i \boldsymbol{f}(\boldsymbol{x}^{n-i}, t^{n-i}; \boldsymbol{\mu}). \tag{3}$$

Given the initial state $\boldsymbol{x}^0(\boldsymbol{\mu}) = \boldsymbol{x}_0(\boldsymbol{\mu})$, solving the FOM O$\Delta$E (2) for $n = 1, \ldots, N_t$ yields the sequence of (discrete) FOM solutions $\boldsymbol{x}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$.

In many cases, the analyst is primarily interested in computing a scalar-valued quantity of interest (QoI) at each time instance, which we define as

$$s^n : \boldsymbol{\mu} \mapsto g(\boldsymbol{x}^n(\boldsymbol{\mu}); t^n, \boldsymbol{\mu}), \quad n = 0, \ldots, N_t, \tag{4}$$

where $s^n : \mathcal{D} \to \mathbb{R}$ and $g : \mathbb{R}^N \times [0, T] \times \mathcal{D} \to \mathbb{R}$ denotes the QoI functional.

If the FOM state-space dimension N or the number of time instances N_t is large, then computing the sequence of FOM solutions $\boldsymbol{x}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$ and associated QoIs $s^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$ can be computationally expensive. If additionally these solutions are sought at many parameter instances, then the resulting *many-query* problem is computationally intractable. In such scenarios, analysts typically aim to overcome this computational barrier by replacing the FOM with a surrogate model that can generate low-cost *approximate solutions*.

2.1. Approximate solutions

When a surrogate model is employed to mitigate the aforementioned computational bottleneck, it generates a sequence of *approximate solutions* $\tilde{\boldsymbol{x}}^n : \mathcal{D} \to \mathbb{R}^N$, $n = 0, \ldots, N_t$ with $\tilde{\boldsymbol{x}}^0(\boldsymbol{\mu}) = \tilde{\boldsymbol{x}}_0(\boldsymbol{\mu})$ given and attendant approximate QoIs

$$\tilde{s}^n : \boldsymbol{\mu} \mapsto g(\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}); t^n, \boldsymbol{\mu}), \quad n = 0, \ldots, N_t, \tag{5}$$

where $\tilde{s}^n : \mathcal{D} \to \mathbb{R}$. As in Ref. [15], we now discuss three types of surrogate models that are often employed to generate the approximate solutions $\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$.

2.1.1. *Inexact solutions*

If one applies an iterative method (e.g., Newton's method) to solve the FOM OΔE system (2) arising at each time instance, then inexact tolerances $\varepsilon^n > 0$, $n = 1, \ldots, N_t$ can be employed such that the approximate solutions satisfy

$$\|\boldsymbol{r}^n(\tilde{\boldsymbol{x}}^n; \tilde{\boldsymbol{x}}^{n-1}, \ldots, \tilde{\boldsymbol{x}}^{n-k^n}, \boldsymbol{\mu})\|_2 \leq \varepsilon^n, \qquad n = 1, \ldots, N_t. \tag{6}$$

In such cases, the surrogate model merely corresponds to early termination of the associated iterative method.

2.1.2. *Lower-fidelity models*

Approximate solutions can also be obtained by employing a computational model of lower fidelity, which can be achieved, for example, by introducing simplifying physical assumptions or coarsening the spatial discretization. In such cases, the corresponding low-fidelity-model (LFM) ODE takes the form

$$\frac{d\boldsymbol{x}_{\mathrm{LF}}}{dt} = \boldsymbol{f}_{\mathrm{LF}}(\boldsymbol{x}_{\mathrm{LF}}, t; \boldsymbol{\mu}), \qquad \boldsymbol{x}_{\mathrm{LF}}(0) = \boldsymbol{x}_{\mathrm{LF},0}(\boldsymbol{\mu}), \qquad t \in [0, T], \tag{7}$$

where $\boldsymbol{x}_{\mathrm{LF}} \equiv \boldsymbol{x}_{\mathrm{LF}}(t, \boldsymbol{\mu})$ with $\boldsymbol{x}_{\mathrm{LF}} : [0, T] \times \mathcal{D} \to \mathbb{R}^{N_{\mathrm{LF}}}$ denotes the LFM state of dimension $N_{\mathrm{LF}}(\ll N)$; $\boldsymbol{x}_{\mathrm{LF},0} : \mathcal{D} \to \mathbb{R}^{N_{\mathrm{LF}}}$ denotes the parameterized initial conditions; and $\boldsymbol{f}_{\mathrm{LF}} : \mathbb{R}^{N_{\mathrm{LF}}} \times [0, T] \times \mathcal{D} \to \mathbb{R}^{N_{\mathrm{LF}}}$ denotes the parameterized LFM velocity.

Applying a linear multistep method (with the same uniform time step Δt as the FOM) to the discretized LFM ODE (7) yields the LFM OΔE arising at the nth time instance

$$\boldsymbol{r}^n_{\mathrm{LF}}(\boldsymbol{x}^n_{\mathrm{LF}}; \boldsymbol{x}^{n-1}_{\mathrm{LF}}, \ldots, \boldsymbol{x}^{n-k^n}_{\mathrm{LF}}, \boldsymbol{\mu}) = \mathbf{0}, \qquad n = 1, \ldots, N_t, \tag{8}$$

where the discrete state $\boldsymbol{x}^n_{\mathrm{LF}}$ is the numerical approximation to $\boldsymbol{x}_{\mathrm{LF}}(t^n)$ and is implicitly defined as the solution to Eq. (8) given the LFM state at the previous k^n time instances and the parameters $\boldsymbol{\mu}$. Analogously to definition (3), the LFM OΔE residual $\boldsymbol{r}^n : \mathbb{R}^{N_{\mathrm{LF}}} \otimes \mathbb{R}^{k^n+1} \times \mathcal{D} \to \mathbb{R}^N$ is defined as

$$\boldsymbol{r}^n_{\mathrm{LF}} : (\boldsymbol{w}_{\mathrm{LF}}; \boldsymbol{x}^{n-1}_{\mathrm{LF}}, \ldots, \boldsymbol{x}^{n-k^n}_{\mathrm{LF}}, \boldsymbol{\mu}) \mapsto \alpha_{\mathrm{LF},0}\boldsymbol{w}_{\mathrm{LF}} - \Delta t \beta_{\mathrm{LF},0}\boldsymbol{f}_{\mathrm{LF}}(\boldsymbol{w}_{\mathrm{LF}}, t^n; \boldsymbol{\mu}) + \sum_{i=1}^{k^n} \alpha_{\mathrm{LF},i}\boldsymbol{x}^{n-i}_{\mathrm{LF}} - \Delta t \sum_{i=1}^{k^n} \beta_{\mathrm{LF},i}\boldsymbol{f}_{\mathrm{LF}}(\boldsymbol{x}^{n-i}_{\mathrm{LF}}, t^{n-i}; \boldsymbol{\mu}). \tag{9}$$

We emphasize that the time integrators for the FOM and LFM can be different, as denoted by the LF subscripts on the coefficients in definition (9). As with the FOM, given the initial LFM state $\boldsymbol{x}^0_{\mathrm{LF}}(\boldsymbol{\mu}) = \boldsymbol{x}_{\mathrm{LF},0}(\boldsymbol{\mu})$, solving the LFM OΔE (8) yields the sequence of (discrete) LFM solutions $\boldsymbol{x}^n_{\mathrm{LF}}$, $n = 0, \ldots, N_t$.

Critically, to recover the approximate solution in the FOM state space $\mathbb{R}^N$, we assume the existence of a prolongation operator $\mathbf{p} : \mathbb{R}^{N_{\mathrm{LF}}} \times \mathcal{D} \to \mathbb{R}^N$ such that

$$\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}) = \mathbf{p}(\boldsymbol{x}^n_{\mathrm{LF}}(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad n = 0, \ldots, N_t. \tag{10}$$

EP: Various prolongation operators exist depending on the type of low-fidelity model. In the case that the LFM is a coarse discretization of a partial differential equation, the prolongation operator typically comprises an interpolation scheme that maps the coarse-grid solution to a fine grid. Another example of a prolongation operator arises when the LFM is a projection-based reduced-order model, which we outline in the following section.

2.1.3. *Projection-based reduced-order models*

Projection-based reduced-order models (ROMs) comprise a popular technique for generating approximate solutions. Such approaches can be derived by applying projection to either the FOM ODE (1) or OΔE (2). In the former case, ROMs correspond to a particular type of low-fidelity model as described in Section 2.1.2. In particular, introducing a trial-basis matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times K}_\star$ (e.g., computed via proper orthogonal decomposition [4]) and a test-basis matrix $\boldsymbol{\Psi} \in \mathbb{R}^{N \times K}_\star$ (e.g., $\boldsymbol{\Psi} = \boldsymbol{\Phi}$ in the case of Galerkin projection), where $K \ll N$ and $\mathbb{R}^{m \times n}_\star$ denotes the set of $m \times n$ full-column-rank matrices (i.e., the non-compact Stiefel manifold), ROMs seek an approximate solution in a K-dimensional affine trial subspace, i.e.,

$$\boldsymbol{x}(t; \boldsymbol{\mu}) \approx \tilde{\boldsymbol{x}}(t; \boldsymbol{\mu}) = \boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}(t; \boldsymbol{\mu}), \tag{11}$$

where $\hat{\boldsymbol{x}} : [0,T] \times \mathcal{D} \to \mathbb{R}^K$ are the generalized coordinates and $\boldsymbol{x}_{\text{ref}} : \mathcal{D} \to \mathbb{R}^K$ denotes the reference state (typically chosen to be either $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \boldsymbol{x}_0(\boldsymbol{\mu})$ or $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \mathbf{0}$). The generalized coordinates are computed by performing a projection process performed on the FOM ODE (1). Namely, the approximation (11) is substituted in the FOM ODE (1), and the resulting residual is enforced to be orthogonal to range of the test-basis matrix. This projection process yields

$$\frac{d\hat{\boldsymbol{x}}}{dt} = [\boldsymbol{\Psi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Psi}^T\boldsymbol{f}(\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}, t; \boldsymbol{\mu}), \qquad \hat{\boldsymbol{x}}(0;\boldsymbol{\mu}) = \boldsymbol{\Phi}^+(\boldsymbol{x}_0(\boldsymbol{\mu}) - \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu})), \tag{12}$$

where the superscript + denotes the Moore–Penrose pseudo-inverse. This approach to model reduction is equivalent to a low-fidelity model of the form (7) with $\boldsymbol{x}_{\text{LF}} = \hat{\boldsymbol{x}}$, $\boldsymbol{f}_{\text{LF}} : (\hat{\boldsymbol{x}}, t; \boldsymbol{\mu}) \mapsto [\boldsymbol{\Psi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Psi}^T\boldsymbol{f}(\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}, t; \boldsymbol{\mu})$, and $\mathbf{p} : (\hat{\boldsymbol{x}}, \boldsymbol{\mu}) \mapsto \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}$.

The most common method that associates with applying projection to the FOM OΔE (2) is the least-squares Petrov–Galerkin (LSPG) method [9], which substitutes the approximation (11) into the FOM OΔE (2) and minimizes the resulting residual in the ℓ^2-norm. In this case, the approximate solutions satisfy

$$\tilde{\boldsymbol{x}}^n \in \underset{\tilde{\boldsymbol{x}} \in \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \text{Ran}(\boldsymbol{\Phi})}{\arg\min} \|\boldsymbol{r}^n(\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{x}}^{n-1}, \ldots, \tilde{\boldsymbol{x}}^{n-k^n}, \boldsymbol{\mu})\|_2. \tag{13}$$

It can be shown [9] that this LSPG approach can be expressed as a time-continuous projection (12) with $\boldsymbol{\Psi} = \boldsymbol{\Psi}(\hat{\boldsymbol{x}}, t; \boldsymbol{\mu}) = (\alpha_0\mathbf{I} - \Delta t\beta_0 \partial\boldsymbol{f}/\partial\boldsymbol{x}(\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}, t; \boldsymbol{\mu}))\boldsymbol{\Phi}$ if $\beta_j = 0$ for $j \geq 1$ (i.e., a single-step method is used), the velocity $\boldsymbol{f}$ is linear in its first argument, or if $\beta_0 = 0$ (i.e., an explicit scheme is used).

2.2. Approximate-solution errors

Employing an approximate solution in lieu of the FOM solution incurs an error that must be accounted for in the ultimate analysis. This work focuses on quantifying two types of such errors, namely

1. the quantity-of-interest error $\delta_s^n(\boldsymbol{\mu}) := s^n(\boldsymbol{\mu}) - \tilde{s}^n(\boldsymbol{\mu})$, $n = 1, \ldots, N_t$ and
2. the normed state error $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu}) := \|\boldsymbol{x}^n(\boldsymbol{\mu}) - \tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})\|_2$, $n = 1, \ldots, N_t$.

Because $\tilde{\boldsymbol{x}}^0(\boldsymbol{\mu})$ and $\boldsymbol{x}^0(\boldsymbol{\mu})$ are given, we can compute $\delta_s^0(\boldsymbol{\mu})$ and $\delta_{\boldsymbol{x}}^0(\boldsymbol{\mu})$ explicitly.

2.3. Classical approaches for error quantification

Classical approaches for quantifying approximate-solution errors for parameterized dynamical systems correspond to *a posteriori error bounds* and *error indicators*. We proceed by briefly reviewing these methods, as they serve as a starting point for our work and will be used to engineer features for the proposed regression framework. We focus in particular on the dual-weighted-residual error indicator.

2.3.1. A posteriori error bounds

We now state *a posteriori* error bounds for both the errors $|\delta_s^n(\boldsymbol{\mu})|$ and $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 1, \ldots, N_t$, which arise from *any* arbitrary sequence of approximate solutions $\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$. The first result was presented in Ref. [27, Theorem 4.3].

Proposition 2.1 (*A posteriori* error bounds [27])**.** For a given parameter instance $\boldsymbol{\mu} \in \mathcal{D}$, if the velocity $\boldsymbol{f}$ is Lipschitz continuous, i.e., there exists a constant $\kappa > 0$ such that $\|\boldsymbol{f}(\mathbf{x}, t; \boldsymbol{\mu}) - \boldsymbol{f}(\mathbf{y}, t; \boldsymbol{\mu})\|_2 \leq \kappa \|\mathbf{x} - \mathbf{y}\|_2$ for all $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^N$ and $t \in \{t^n\}_{i=1}^{N_t}$, and the time step is sufficiently small such that $\Delta t < |\alpha_0|/|\beta_0|\,\kappa$, then

$$\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu}) \leq \frac{1}{h}\left\|\boldsymbol{r}^n\left(\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}); \tilde{\boldsymbol{x}}^{n-1}(\boldsymbol{\mu}), \ldots, \tilde{\boldsymbol{x}}^{n-k^n}(\boldsymbol{\mu}), \boldsymbol{\mu}\right)\right\|_2 + \sum_{j=1}^{k^n} \gamma_j \delta_{\boldsymbol{x}}^{n-j}(\boldsymbol{\mu}) \tag{14}$$

for all $n = 1, \ldots, N_t$. Here, $h := |\alpha_0| - |\beta_0| \kappa \Delta t$ and $\gamma_j := (|\alpha_j| + |\beta_j| \kappa \Delta t)/h$. If additionally the QoI functional g is Lipschitz continuous, i.e., there exists a constant $\kappa_g > 0$ such that $|g(\mathbf{x}; t, \boldsymbol{\mu}) - g(\mathbf{y}; t, \boldsymbol{\mu})| \leq \kappa_g \|\mathbf{x} - \mathbf{y}\|_2$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ and $t \in \{t^n\}_{i=1}^{N_t}$, then

$$|\delta_s^n(\boldsymbol{\mu})| \leq \frac{\kappa_g}{h} \left\| \boldsymbol{r}^n \left(\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}); \tilde{\boldsymbol{x}}^{n-1}(\boldsymbol{\mu}), \ldots, \tilde{\boldsymbol{x}}^{n-k^n}(\boldsymbol{\mu}), \boldsymbol{\mu} \right) \right\|_2 + \kappa_g \sum_{j=1}^{k^n} \gamma_j \delta_{\boldsymbol{x}}^{n-j}(\boldsymbol{\mu}) \tag{15}$$

for all $n = 1, \ldots, N_t$.

Proof. Bound (14) was derived in Ref. [27, Theorem 4.3]. Bound (15) follows directly from Lipschitz continuity of the QoI functional and the bound (14). □

We make three critical observations from inequalities (14) and (15):

1. The error bounds are *residual based*, i.e., the magnitudes of the error bounds are driven by the FOM residual operator $\boldsymbol{r}^n$ evaluated at the sequence of approximate solutions $\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$. Note that the approximate solution incurs no error if the approximate solutions exactly satisfy the FOM OΔE (2) such that the residual is zero at all time instances. This also illustrates why the residual norm is often viewed as a useful *error indicator* for guiding greedy methods for snapshot collection [7, 6, 19, 48, 49] or trust-region optimization algorithms [52, 50, 51].

2. The error bounds exhibit dependence on *non-local* quantities, i.e., the error at the nth time instance depends on the error at the previous k^n time instances through recursion via the rightmost term in inequalities (14) and (15). This comprises a fundamental difference from error bounds for static problems.

3. The error bounds often *lack sharpness*, i.e., they are often significant larger than the magnitudes of the actual errors. This overestimation is exacerbated for ill-conditioned problems (in which case the Lipschitz constants κ and κ_g are large) and long time horizons (as the error bounds grow exponentially in time). As such, their practical utility for error estimation itself is often limited.

4. The error bounds are *difficult to compute*, i.e., they require computing Lipschitz constants κ and κ_g, which is challenging for general nonlinear dynamical systems. While bounding these constants is possible in some contexts [23], this often incurs substantial additional computational costs.

2.3.2. Error indicator: dual-weighted residuals

Besides the residual norm, the dual-weighted residual is perhaps the most popular error indicator. The dual-weighted residual is derived from a low-order approximation of the QoI error and, as such, is often strongly correlated with the error [12]. Because deriving dual-weighted residuals for dynamical systems requires considering the problem from the monolithic "space–time" perspective, we now introduce the required space–time formulation of the problem.

We begin by defining a "vectorization" function that assembles the full space–time solution from a tuple of time-local solutions as

$$\begin{aligned} \operatorname{vec} &: (\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{N_t}) \mapsto \begin{bmatrix} [\boldsymbol{y}^1]^T & \cdots & [\boldsymbol{y}^{N_t}]^T \end{bmatrix}^T \\ &: \mathbb{R}^p \otimes \mathbb{R}^{N_t} \to \mathbb{R}^{pN_t}, \end{aligned}$$

for arbitrary p. Then, we define the space–time FOM state and space–time approximate solution as $\bar{\boldsymbol{x}}(\boldsymbol{\mu}) := \operatorname{vec}(\boldsymbol{x}^1(\boldsymbol{\mu}), \ldots, \boldsymbol{x}^{N_t}(\boldsymbol{\mu}))$ and $\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}) := \operatorname{vec}(\tilde{\boldsymbol{x}}^1(\boldsymbol{\mu}), \ldots, \tilde{\boldsymbol{x}}^{N_t}(\boldsymbol{\mu}))$, respectively, where $\bar{\boldsymbol{x}}, \bar{\tilde{\boldsymbol{x}}} : \mathcal{D} \to \mathbb{R}^{NN_t}$. We also define the space–time residual as

$$\begin{aligned} \bar{\boldsymbol{r}} &: (\bar{\boldsymbol{w}}; \boldsymbol{\mu}) \mapsto \operatorname{vec}(\boldsymbol{r}^1(\boldsymbol{w}^1; \boldsymbol{\mu}), \ldots, \boldsymbol{r}^{N_t}(\boldsymbol{w}^{N_t}; \boldsymbol{w}^{N_t-1}, \ldots, \boldsymbol{w}^{N_t-k^{N_t}}, \boldsymbol{\mu})), \\ &: \mathbb{R}^{NN_t} \times \mathcal{D} \to \mathbb{R}^{NN_t}, \end{aligned}$$

where $\bar{\boldsymbol{w}} \equiv \mathrm{vec}(\boldsymbol{w}^1, \ldots, \boldsymbol{w}^{N_t})$. Lastly, we define the space–time QoI functional as

$$\begin{aligned} \bar{g} &: \big(\bar{\boldsymbol{w}}; \boldsymbol{\mu}\big) \mapsto \mathrm{vec}\big(g(\boldsymbol{w}^1; t^1, \boldsymbol{\mu}), \ldots, g(\boldsymbol{w}^{N_t}; t^{N_t}, \boldsymbol{\mu})\big), \\ &: \mathbb{R}^{NN_t} \times \mathcal{D} \to \mathbb{R}^{N_t}, \end{aligned}$$

such that the space–time QoIs computed from the FOM and approximate solutions are given by

$$\bar{s} : \boldsymbol{\mu} \mapsto \bar{g}(\bar{\boldsymbol{x}}(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad \bar{\tilde{s}} : \boldsymbol{\mu} \mapsto \bar{g}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu}),$$

respectively, where $\bar{s}, \bar{\tilde{s}} : \mathcal{D} \to \mathbb{R}^{N_t}$.

Assuming the residual is twice continuously differentiable in its first argument and noting that $\bar{\boldsymbol{r}}(\bar{\boldsymbol{x}}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \boldsymbol{0}$, we can write

$$-\bar{\boldsymbol{r}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \left[\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]\Big(\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\Big) + \mathcal{O}\Big(\|\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\|_2^2\Big). \tag{16}$$

Assuming the space–time residual Jacobian $\left[\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]$ is invertible, Eq. (16) can be solved for a first-order approximation of the solution error

$$\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}) = -\left[\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]^{-1} \bar{\boldsymbol{r}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu}) + \mathcal{O}\Big(\|\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\|_2^2\Big). \tag{17}$$

If we additionally assume the space–time QoI functional $\bar{g}$ is twice continuously differentiable in its first argument, the error in the space–time QoI can be represented to first order as,

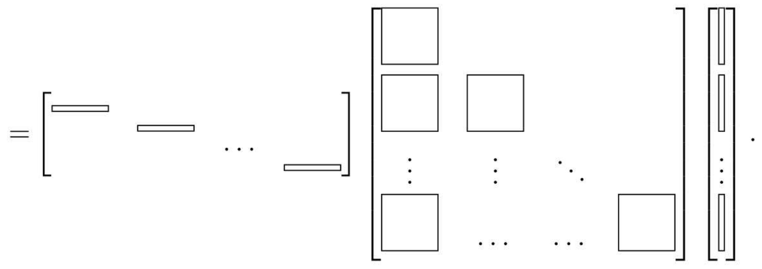
$$\bar{s}(\boldsymbol{\mu}) - \bar{\tilde{s}}(\boldsymbol{\mu}) = \left[\frac{\partial \bar{g}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]\Big(\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\Big) + \mathcal{O}\Big(\|\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\|_2^2\Big). \tag{18}$$

Substituting Eq. (17) into Eq. (18) yields,

$$\bar{s}(\boldsymbol{\mu}) - \bar{\tilde{s}}(\boldsymbol{\mu}) = d(\boldsymbol{\mu}) + \mathcal{O}\Big(\|\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\|_2^2\Big), \tag{19}$$

where the dual-weighted residual comprises a first-order approximation of the QoI error and is defined as

$$d(\boldsymbol{\mu}) := -\underbrace{\left[\frac{\partial \bar{g}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]}_{N_t \times NN_t} \underbrace{\left[\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]^{-1}}_{NN_t \times NN_t} \underbrace{\bar{\boldsymbol{r}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})}_{NN_t \times 1}, \tag{20}$$


(21)

Computing the dual-weighted residual (20) involves applying the inverse of the $NN_t \times NN_t$ space–time residual Jacobian and can be computed in one of two ways: (1) the direct (forward) method, or (2) the adjoint (backward) method. The direct method solves the (linear) *sensitivity equations*

$$\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\bar{\boldsymbol{q}} = \bar{\boldsymbol{r}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu}), \tag{22}$$

where $\bar{\boldsymbol{q}} \equiv \bar{\boldsymbol{q}}(\boldsymbol{\mu}) \equiv \text{vec}(\boldsymbol{q}^1(\boldsymbol{\mu}), \ldots, \boldsymbol{q}^{N_t}(\boldsymbol{\mu}))$ with $\boldsymbol{q}^n : \mathcal{D} \to \mathbb{R}^N$, $n = 1, \ldots, N_t$ is implicitly defined as the solution to Eq. (22) and denotes the space–time *sensitivity vector*. The direct method subsequently computes the dual-weighted residual as

$$d(\boldsymbol{\mu}) = -\big[\frac{\partial \bar{g}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\big]\bar{\boldsymbol{q}}(\boldsymbol{\mu}). \tag{23}$$

Due to the sparsity pattern of the space–time residual Jacobian $\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})$, this approach is equivalent to performing a linear forward-in-time solve, which can be executed simultaneously with the forward-in-time solve used to generate the approximate solutions, i.e., the sensitivities $\boldsymbol{q}^n(\boldsymbol{\mu})$ can be computed contemporaneously with $\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})$. In contrast, the adjoint method solves the multiple-right-hand-side (linear) *adjoint equations*

$$\left[\frac{\partial \bar{\boldsymbol{r}}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]^T \bar{\boldsymbol{y}} = \left[\frac{\partial \bar{g}}{\partial \bar{\boldsymbol{w}}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu})\right]^T, \tag{24}$$

where

$$\bar{\boldsymbol{y}} \equiv \bar{\boldsymbol{y}}(\boldsymbol{\mu}) \equiv \left[\text{vec}(\boldsymbol{y}_1^1(\boldsymbol{\mu}), \ldots, \boldsymbol{y}_1^{N_t}(\boldsymbol{\mu})) \quad \cdots \quad \text{vec}(\boldsymbol{y}_{N_t}^1(\boldsymbol{\mu}), \ldots, \boldsymbol{y}_{N_t}^{N_t}(\boldsymbol{\mu}))\right] \in \mathbb{R}^{NN_t \times N_t},$$

with $\boldsymbol{y}_i^n : \mathcal{D} \to \mathbb{R}^N$, $i, n = 1, \ldots, N_t$ is implicitly defined as the solution to Eq. (24) and denotes N_t space–time *dual vectors*. The adjoint method subsequently computes the dual-weighted residual as

$$d(\boldsymbol{\mu}) = -\bar{\boldsymbol{y}}(\boldsymbol{\mu})^T \bar{\boldsymbol{r}}(\bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu}); \boldsymbol{\mu}). \tag{25}$$

Due to the sparsity pattern of the space–time residual Jacobian transpose, this approach is tantamount to performing a linear backward-in-time solve, and must be computed *after* the approximate-solution sequence has been fully computed, as this sequence defines the argument at which the space–time residual Jacobian is evaluated.

Analogously to the observations made about error bounds in the Section 2.3.1, we now make several observations about the dual-weighted residual:

1. The dual-weighted residual is *residual-based*, i.e., it corresponds to a linear combination of elements of the space–time residual, where the weights in the linear combination are provided by the dual according to Eq. (25).

2. The dual-weighted residual exhibits dependence on *non-local* quantities, i.e., computing the dual-weighted residual using either the direct method (23) or the adjoint method (25) requires solving a linear system either forward-in-time or backward-in-time over the entire time domain. Further, the sparsity pattern of the operators in Eq. (20) illustrates that the QoI error at the nth time instance depends on quantities computed at all previous time instances.

3. The dual-weighted residual may be an *inaccurate* approximation of the error, i.e., it is a first-order approximation valid in the limit $\|\bar{\boldsymbol{x}}(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{x}}}(\boldsymbol{\mu})\|_2 = \sqrt{\sum_{n=1}^{N_t}(\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu}))^2} \to 0$, and thus can be inaccurate for cases where the normed state error is not negligible. This is precisely the case we are interested in, as we aim to model this error.

4. Dual-weighted residuals can be *difficult to compute*, i.e., they require solving either the forward sensitivity (22) or adjoint equations (24), each of which is characterized by a block triangular system matrix of dimension $NN_t \times NN_t$, with NN_t the space–time dimension of the original full-order-model problem. The use of approximate solutions aims to avoid any computations that scale with the dimension N, and thus computing the dual-weighted residual in this manner is often impractical. To mitigate this computational burden, analysts often adopt the adjoint method, solve the adjoint equations (24) using a coarse model (e.g., a coarse mesh), and prolongate the computed (coarse) dual vector to a representation compatible with the (fine) full-order model in order to compute the dual-weighted residual [46]. In addition to these computational-cost issues, adopting the adjoint method introduces other challenges, as it requires access to the residual-Jacobian transpose to solve Eq. (24), which may not be available.

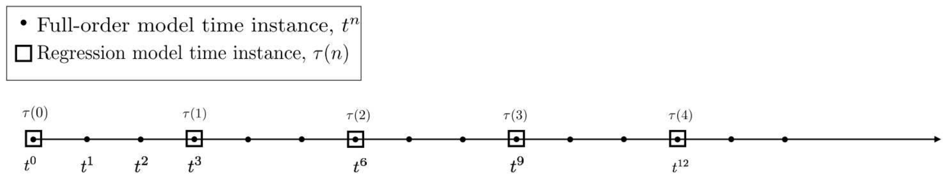


Figure 1: Example of time discretizations for the full-order model and machine learning model. In this case, the regression model is solved on a temporal grid that is three times as coarse as the approximate and full-order models such that $\mathbb{T} = \{3, 6, 9, 12, \ldots\}$.

2.3.3. Discussion

This work aims to overcome the most prominent shortcomings of *a posteriori* error bounds and dual-weighted residuals: (1) their lack of sharpness or inaccuracy, and (2) the difficulty in their computation. We aim to achieve this by applying modern machine-learning methods within a regression setting, wherein the *regression model* is provided by a recursive time-series-prediction model that can capture dependencies on non-local quantities, the *features* correspond to residual-based time-local error indicators that appear in the definitions of the *a posteriori* error bounds and dual-weighted residuals[1], and the *response* corresponds to the time-local normed state or QoI error. Thus, we employ the classical methods for error quantification to guide the design of the both the regression model and features. We also aim to construct a statistical model of the response error by considering several noise models. The next section describes the proposed methodology, which adopts this strategy.

3. Time-Series Machine-Learning Error Modeling

We now present the proposed Time-Series Machine-Learning Error Modeling (T-MLEM) method, which constructs a mapping from residual-based features to a random variable for the error using regression techniques that can capture dependencies on non-local quantities. The method comprises an extension of the machine-learning error modeling (MLEM) framework presented in Ref. [15]—which was applicable to parameterized static systems—to parameterized dynamical systems. Because the method is applicable to modeling both the QoI error $\delta_s^n(\boldsymbol{\mu})$ and normed-state error $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$, we denote the error of interest generically as $\delta^n(\boldsymbol{\mu})$ in this section.

3.1. Machine-learning framework

The proposed method aims to predict errors on a *subset* of the time grid $t^n = n\Delta t$, $n = 0, \ldots, N_t$ used to define the FOM OΔE (2). We make this choice because the time step Δt is typically chosen to control time-discretization errors, while we may be interested in modeling the error at only a (relatively small) subset of these grid points; furthermore, constructing a time-series model of these errors on a coarser time grid reduces the computational cost of training the regression model. We define this coarse time grid as $t^n = n\Delta t$, $n \in \mathbb{T}$, where $\mathbb{T} \subseteq \{1, \ldots, N_t\}$ denotes the coarse time-index set comprising $\bar{N}_t := |\mathbb{T}|(\leq N_t)$ time instances and equipped with a mapping $\tau : \{0, \ldots, \bar{N}_t\} \to \{0, \ldots, N_t\}$ from a coarse time index to the corresponding fine time index, which satisfies $\tau(0) = 0$.

We assume that we can compute *features* $\boldsymbol{\rho}^n(\boldsymbol{\mu}) \in \mathbb{R}^{N_\rho}$ that are informative of the *response* $\delta^n(\boldsymbol{\mu}) \in \mathbb{R}$—which is the time-local error of interest—at time instances $n \in \mathbb{T}$. Section 3.1.1 provides candidate features,

[1]We emphasize that we use terms *appearing* in, e.g., error bounds as features, not the error bounds themselves. This is an important distinction as the error bound can be a poor indicator of the actual error due to a lack of sharpness.

which are inspired by the classical error-quantification methods described in Section 2.3. Given the ability to compute these features, we propose to construct error models $\hat{\delta}^n(\boldsymbol{\mu})(\approx \delta^n(\boldsymbol{\mu}))$ of the form

$$\hat{\delta}^n(\boldsymbol{\mu}) = \hat{\delta}^n_f(\boldsymbol{\mu}) + \hat{\delta}^n_\epsilon(\boldsymbol{\mu}), \quad n \in \mathbb{T}, \tag{26}$$

where $\hat{\delta}^n_f : \mathcal{D} \to \mathbb{R}$, $n \in \{0\} \cup \mathbb{T}$ denotes the sequence of *deterministic regression-function models* and $\hat{\delta}^n_\epsilon : \mathcal{D} \to \mathbb{V}$, $n \in \{0\} \cup \mathbb{T}$ denotes a sequence of mean-zero random variables that serve as the *stochastic noise models* such that $\hat{\delta}^n : \mathcal{D} \to \mathbb{V}$, $n \in \mathbb{T}$ with $\mathrm{E}[\hat{\delta}^n(\boldsymbol{\mu})] = \hat{\delta}^n_f(\boldsymbol{\mu})$; we denote the set of real-valued random variables by $\mathbb{V}$.

As previously described, we propose to construct the regression model using a recursive time-series-prediction model that can capture dependencies on non-local quantities. In particular, we impose the following structure on the regression-function model:

$$\hat{\delta}^n_f(\boldsymbol{\mu}) = \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}), \boldsymbol{h}^n(\boldsymbol{\mu})), \quad n \in \mathbb{T}. \tag{27}$$

Here, $\hat{f} : \mathbb{R}^{N_\rho} \times \mathbb{R}^{N_{\boldsymbol{h}}} \to \mathbb{R}$, while $\boldsymbol{h}^n : \mathcal{D} \to \mathbb{R}^{N_{\boldsymbol{h}}}$, $n \in \{0\} \cup \mathbb{T}$ denotes the sequence of *latent variables*. Critically, the inclusion of these latent variables enables the proposed method to capture dependencies on non-local quantities, as we enforce the latent variables to be governed by the recursion

$$\boldsymbol{h}^{\tau(n)}(\boldsymbol{\mu}) = \mathbf{g}(\boldsymbol{\rho}^{\tau(n)}(\boldsymbol{\mu}), \boldsymbol{h}^{\tau(n-1)}(\boldsymbol{\mu}), \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu})), \quad n = 1, \ldots, \bar{N}_t, \tag{28}$$

where $\mathbf{g} : \mathbb{R}^{N_\rho} \times \mathbb{R}^{N_{\boldsymbol{h}}} \times \mathbb{R} \to \mathbb{R}^{N_{\boldsymbol{h}}}$; $\hat{\delta}^0_f(\boldsymbol{\mu}) = \delta^0(\boldsymbol{\mu})$ is given; and $\boldsymbol{h}^0(\boldsymbol{\mu}) = \mathbf{0}$. Similarly, we impose the following structure on the noise model:

$$\hat{\delta}^{\tau(n)}_\epsilon(\boldsymbol{\mu}) = \hat{\epsilon}(\hat{\delta}^{\tau(n-1)}_\epsilon(\boldsymbol{\mu}), \epsilon^{\tau(n)}), \quad n = 1, \ldots, \bar{N}_t. \tag{29}$$

Here, $\hat{\epsilon} : \mathbb{V} \times \mathbb{V} \to \mathbb{V}$, $\hat{\delta}^0_\epsilon(\boldsymbol{\mu}) = 0$, and $\epsilon^n \in \mathbb{V}$, $n \in \mathbb{T}$ denotes a sequence of independent and identically distributed mean-zero random variables.

Remark 3.1 (Classical non-recursive regression)**.** We note that classical non-recursive regression methods can be described by omitting latent variables from the regression function approximation such that $\hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}), \boldsymbol{h}^n(\boldsymbol{\mu})) \equiv \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}))$, omitting latent-variable dynamics (28), and employing a simple non-recursive noise model such that $\hat{\delta}^n_\epsilon(\boldsymbol{\mu}) = \epsilon^n$, $n \in \mathbb{T}$.

Remark 3.2 (Noise models)**.** We note that more sophisticated functional forms of $\hat{\epsilon}$ could be considered, e.g., by allowing parameters that define the probability distribution of ϵ^n for $n \in \mathbb{T}$ to depend on features, or by including latent variables as inputs to the function $\hat{\epsilon}$.

The primary distinction between the proposed T-MLEM method and the methods proposed in Refs. [15, 45] is the consideration of recursive regression models that employ latent variables. The MLEM method [15] considered only static problems and thus did not require the use of such variables. The EMML framework [45] accounted for dependencies on non-local quantities by explicitly considering features corresponding to quantities computed at previous time instances. This introduces difficulties, as it requires knowledge about the memory of the system (i.e., how many previous time instances to consider in constructing the features) and can lead to a prohibitively large set of features. In contrast, the use of latent variables and the specific functional forms (27)–(29) allow the proposed T-MLEM method to capture non-local effects naturally.

As in Ref. [15], we aim to construct regression models that exhibit the following properties:

Objective 1 **Low cost**. The error model $\hat{\delta}^n(\boldsymbol{\mu})$ should be computationally inexpensive to deploy, i.e., its evaluation should not dominate the computational cost of computing the approximate solutions $\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu})$, $n = 0, \ldots, N_t$. This requirement implies that the features $\boldsymbol{\rho}^n(\boldsymbol{\mu})$ should be cheap to compute, the latent-variable dimension $N_{\boldsymbol{h}}$ should be relatively small, and it should be inexpensive to evaluate the functions $\hat{f}$, $\mathbf{g}$, and $\hat{\epsilon}$.

Objective 2 **Low noise variance**. Because the variance of the noise model $\hat{\delta}^n_\epsilon(\boldsymbol{\mu})$ reflects the magnitude of epistemic uncertainty introduced by the use of an approximate solution, reducing this variance in turn reduces the uncertainty introduced into the analysis. A model with low noise variance has not been underfit to training data.

Objective 3 **Generalizability**. For the regression model to be trustworthy, the distributions of the true error $\delta^n(\boldsymbol{\mu})$ and the error model $\hat{\delta}^n(\boldsymbol{\mu})$ should closely match on an independent test set, which was not used to train the error model. A model exhibiting generalizability has not been overfit to training data.

We adopt the machine-learning framework proposed in Ref. [15] to satisfy these objectives. This framework comprises the following steps:

Step 1 **Feature engineering.** Select features that are inexpensive to compute (Objective 1), informative of the error such that a low-noise-variance model can be constructed (Objective 2), and low dimensional (i.e., N_ρ small) such that less training data are required for the resulting model to generalize (Objective 3). To satisfy these requirements, we engineer residual-based features informed by classical error-quantification methods as described in Section 2.3. We ensure these features are low dimensional and inexpensive to compute through the use of sampling-based approximations (e.g., gappy principal component analysis). Section 3.1.1 describes this step.

Step 2 **Data generation.** After selecting candidate features, generate both training and test data in the form of a set of response–feature pairs on the coarse time-index set $\mathbb{T}$ for parameter instances sampled according to the imposed probability distribution on the parameter domain $\mathcal{D}$. Constructing these data requires computing *both* the FOM solutions and approximate solutions for the selected parameter instances. Sections 3.1.2–3.1.3 describe this step.

Step 3 **Train the deterministic regression-function model $\hat{\delta}^n_f(\boldsymbol{\mu})$.** Define candidate functional forms for the functions $\hat{f}$ and $\mathbf{g}$, compute model parameters by (approximately) minimizing a loss function defined on the training data, and perform hyperparameter selection using (cross-)validation. Sections 3.1.4–3.1.5 describes this step.

Step 4 **Train the stochastic noise model $\hat{\delta}^n_\epsilon(\boldsymbol{\mu})$.** Finally, define candidate functional forms for the function $\hat{\epsilon}$, and compute model parameters via maximum likelihood estimation on a training set independent from that used to train the deterministic regression function. In this work, we employ a subset of the test set used to evaluate the deterministic regression function for this purpose. Sections 3.1.6–3.1.7 describe this step.

3.1.1. Feature engineering

We now describe Step 1 of the proposed framework: feature engineering. Namely, we describe the candidate features $\boldsymbol{\rho}^n$ that we consider, which are inspired by those proposed in Ref. [15]. Several considerations drive feature engineering, which balance Objective 1–Objective 3:

Consideration 1 **Feature cost.** To satisfy Objective 1, we aim to develop features that are inexpensive to compute. Satisfying this objective can compromise quality, as high-quality features are often costly to compute.

Consideration 2 **Feature quality.** To satisfy Objective 2, we aim to develop features that are as informative of the error as possible, because employing high-quality explanatory features can lead to a low-noise-variance regression model.

Consideration 3 **Number of features.** Employing a large number of features often leads to a high-capacity model that can yield low noise variance with sufficient training data (i.e., Objective 2 is bolstered). However, computing a large number of features can be computationally expensive (i.e., Objective 1 suffers), and often requires a large amount of training data to generalize well and avoid overfitting (i.e., Objective 3 suffers). Regularization strategies can be employed when training a regression model with a large number of features to mitigate overfitting.

The remainder of this section details a variety of candidate features that differ in terms of the above considerations.

Feature 1 **Parameters**: The system's parametric inputs $\boldsymbol{\mu}$ correspond to free-to-compute but low-quality features, where the number of features is equal to the number of parameters $N_{\boldsymbol{\mu}}$. Simply setting $\boldsymbol{\rho}^n(\boldsymbol{\mu}) = \boldsymbol{\mu}$ comprises the same feature choice as that employed by the classical model-discrepancy approach [25], which uses Gaussian-process regression to construct the regression model. These features are time independent.

Feature 2 **Time**: The time variable t^n is a single, free-to-compute feature, which is likely to be low quality in most scenarios. This low quality arises from the fact that—in most applications—the error is not driven by the time variable itself. Instead, the classical error-quantification methods discussed in Section 2.3 suggest that the error is instead determined by the residual, Lipschitz constants, and dual vectors. Intuitively, employing time as a feature can cause the resulting regression model to be "tied" to the particular time coordinates associated with the behavior observed during training. This implies that the resulting regression method will lack invariance with respect to the time grid, i.e., the model will not generalize well for problems characterized by phase shifts or for prediction beyond the training time interval. The EMML framework considered time as a feature [45, Table I].

Feature 3 **Residual norm**: As discussed in Section 2.3, classical error-quantification methods are residual based. In particular, the first term on the right-hand side of *a posteriori* error bound (14) demonstrates that the residual norm $\|\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\|_2$, where $\tilde{\boldsymbol{r}}^n : \boldsymbol{\mu} \mapsto \boldsymbol{r}^n\left(\tilde{\boldsymbol{x}}^n(\boldsymbol{\mu}); \tilde{\boldsymbol{x}}^{n-1}(\boldsymbol{\mu}), \ldots, \tilde{\boldsymbol{x}}^{n-k^n}(\boldsymbol{\mu}), \boldsymbol{\mu}\right)$, is informative of the state error $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$; bound (15) shows that the residual norm $\|\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\|_2$ is also informative of the magnitude of the QoI error $|\delta_s^n(\boldsymbol{\mu})|$. This feature is computationally expensive to compute, as it requires computing all N elements of the vector $\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$ before computing its norm. However, this (single) feature is likely of higher quality than the both Feature 1 and Feature 2, as it appears explicitly in classical *a posteriori* error bounds (14)–(15). Yet, this feature will likely be of lower quality for predicting the (signed) QoI error $\delta_s^n(\boldsymbol{\mu})$ because it is not informative of the error's sign. We note that—in contrast to static systems—the residual norm $\|\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\|_2$ (along with stability constants) is not sufficient to bound the errors $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$ and $|\delta_s^n(\boldsymbol{\mu})|$. Instead, inequalities (14)–(15) illustrate that these bounds depend additionally on non-local quantities, namely $\delta_{\boldsymbol{x}}^{n-j}(\boldsymbol{\mu})$, $j = 1, \ldots, k^n$. Thus, we expect this feature choice—as well as all subsequent residual-based features—to perform best when used with recursive regression methods that can capture dependencies on non-local quantities. We note that the ROMES method [12] employed features $\boldsymbol{\rho} = \|\tilde{\boldsymbol{r}}(\boldsymbol{\mu})\|_2$ with Gaussian-process regression to construct a regression model for the normed state error.

Feature 4 **Residual**: To increase the number of features relative to Feature 3 yet maintain high feature quality, one may employ all N elements of the residual vector $\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$ as features. This has the effect of changing the number of features from one extreme to another, as we assume the full-order-model dimension N is large. Employing N features can enable very high-capacity regression models, which can lead to low noise variance given sufficient training data; however, the amount of training data needed to ensure generalizability may be excessive. Further, these features are costly to compute. These features are of high quality, as the residual not only appears in the *a posteriori* error bounds (14)–(15), but it also appears in the dual-weighted residual discussed in Section 2.3.2. Namely, inserting Eq. (25) into Eq. (19) shows that the QoI error can be approximated to first order by a linear combination of space–time residual elements.

Feature 5 **Residual principal components** $\hat{\boldsymbol{r}}^n(\boldsymbol{\mu})$. As proposed in Ref. [15], to retain the high quality of Feature 4, but reduce the number of features such that less training data is needed for generalization, we can employ *principal components* of the residual as features. In particular, one can use

$$\hat{\boldsymbol{r}}^n(\boldsymbol{\mu}) := \boldsymbol{\Phi}_{\boldsymbol{r}}^T(\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{r}}})$$

as features, where the columns of $\boldsymbol{\Phi}_{\boldsymbol{r}} \in \mathbb{R}_\star^{N\times n_r}$ comprise the first $n_r(\ll N)$ principal components of the training set $\mathcal{T}_{\boldsymbol{r},\text{train}} \subset \mathbb{R}^N$ (which will be described in Section 3.1.3), and $\bar{\tilde{\boldsymbol{r}}} := \frac{1}{|\mathcal{T}_{\boldsymbol{r},\text{train}}|}\sum_{\tilde{\boldsymbol{r}}\in\mathcal{T}_{\boldsymbol{r},\text{train}}} \tilde{\boldsymbol{r}}$ denotes the training-set mean. These features are costly to compute, as they require computing all N elements of the residual $\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$ before applying projection with the principal-component matrix $\boldsymbol{\Phi}_{\boldsymbol{r}}$. However, these features retain the high quality of the residual vector if the truncation level n_r is set such that the principal components explain most of the variance of the data in the training set $\mathcal{T}_{\boldsymbol{r},\text{train}}$. Critically, this approach associates with using n_r features and thus constitutes a compromise between Feature 3 and Feature 4: it will lead to a higher-capacity model than can be obtained using Feature 3, but will likely generalize with significantly less training data than would be the case with Feature 4.

Feature 6 **Residual gappy principal components**: This approach aims to address the primary shortcoming of Feature 5: the high computational cost incurred by the need to compute all N elements of the residual $\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$. In particular, this feature choice aims to approximate the residual principal components using the gappy POD method [14], which replaces orthogonal projection with a sampling-based linear least-squares approximation to realize computational-cost savings. Specifically, introducing a sampling matrix $\mathbf{P} \in \{0,1\}^{n_s\times N}$ comprising n_s selected rows of the identity matrix (with $n_r \le n_s \ll N$), the residual gappy principal components are

$$\hat{\boldsymbol{r}}_g^n(\boldsymbol{\mu}) := [\mathbf{P}\boldsymbol{\Phi}_{\boldsymbol{r}}]^+\mathbf{P}(\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}) - \bar{\tilde{\boldsymbol{r}}}).$$

In this work, we construct the sampling matrix $\mathbf{P}$ according to the q-sampling method [11]; Appendix C presents the associated algorithm. We note that the number of features for this method is the same as that for Feature 5, and the feature quality is similar to that of Feature 5 if least-squares reconstruction yields similar results to orthogonal projection. However, computing these features requires computing only $n_s(\ll N)$ elements of the residual vector $\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$; thus, this feature choice is substantially less costly to compute than Feature 5.

Feature 7 **Sampled residual**: Feature 6 constitutes an affine transformation applied to the sampled residual $\mathbf{P}\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$. Thus, it is sensible to consider employing only the underlying sampled residual $\mathbf{P}\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})$ as features, given that many regression techniques are likely to discover this affine transformation if it yields superior prediction performance. Further, employing the sampled residual does not require computing the residual principal components $\boldsymbol{\Phi}_{\boldsymbol{r}}$ (although q-sampling employs this matrix to construct the sampling matrix $\mathbf{P}$). We note that the number of features in this case is $n_s(\ge n_r)$.

In the numerical experiments, we consider feature-engineering methods that employ different combinations of the above features. However, all considered methods use at most one of the residual-based features, i.e., at most one of Feature 3–Feature 7. Table 1 summarizes the feature-engineering methods considered in the numerical experiments.

3.1.2. Data generation

We now describe Step 2 of the proposed framework: data generation. After selecting the features, the method generates response–feature pairs that comprise the training, validation, and test sets used to select the deterministic regression-function model and stochastic noise model, respectively. Specifically, we first define

$$\mathcal{T}(\overline{\mathcal{D}}, \overline{\mathbb{T}}) := \{(\delta^n(\boldsymbol{\mu}), \boldsymbol{\rho}^n(\boldsymbol{\mu}))\,|\,\boldsymbol{\mu}\in\overline{\mathcal{D}}, n\in\overline{\mathbb{T}}\}$$

as the set of response–feature pairs generated by computing the full-order and approximate states for $\boldsymbol{\mu}\in\overline{\mathcal{D}}$ and computing the features and errors at time instances t^n with $n\in\overline{\mathbb{T}}\subseteq\{1,\ldots,N_t\}$. We then define the training, validation, and test data as

$$\mathcal{T}_{\text{train}} := \mathcal{T}(\mathcal{D}_{\text{train}}, \mathbb{T}_{\text{train}}), \qquad \mathcal{T}_{\text{val}} := \mathcal{T}(\mathcal{D}_{\text{val}}, \mathbb{T}_{\text{val}}), \qquad \mathcal{T}_{\text{test}} := \mathcal{T}(\mathcal{D}_{\text{test}}, \mathbb{T}_{\text{test}}), \tag{30}$$

Description	Features $\boldsymbol{\rho}^n$	Number of features N_ρ	Number of residual elements required
Parameters	$\boldsymbol{\mu}$	$N_{\boldsymbol{\mu}}$	0
Parameters and time	$[\boldsymbol{\mu}; t^n]$	$N_{\boldsymbol{\mu}} + 1$	0
Residual norm	$\lVert\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\rVert_2$	1	N
Parameters and residual norm	$[\boldsymbol{\mu}; \lVert\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\rVert_2]$	$N_{\boldsymbol{\mu}} + 1$	N
Parameters, residual norm, and time	$[\boldsymbol{\mu}; \lVert\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})\rVert_2; t^n]$	$N_{\boldsymbol{\mu}} + 2$	N
Parameters and residual	$[\boldsymbol{\mu}; \tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})]$	$N_{\boldsymbol{\mu}} + N$	N
Parameters, residual, and time	$[\boldsymbol{\mu}; \tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}); t^n]$	$N_{\boldsymbol{\mu}} + N + 1$	N
Parameters and residual principal components	$[\boldsymbol{\mu}; \hat{\boldsymbol{r}}^n(\boldsymbol{\mu})]$	$N_{\boldsymbol{\mu}} + n_r$	N
Parameters, residual principal components, and time	$[\boldsymbol{\mu}; \hat{\boldsymbol{r}}^n(\boldsymbol{\mu}); t^n]$	$N_{\boldsymbol{\mu}} + n_r + 1$	N
Parameters and gappy residual principal components	$[\boldsymbol{\mu}; \hat{\boldsymbol{r}}_g^n(\boldsymbol{\mu})]$	$N_{\boldsymbol{\mu}} + n_r$	n_s
Parameters, gappy residual principal components, and time	$[\boldsymbol{\mu}; \hat{\boldsymbol{r}}_g^n(\boldsymbol{\mu}); t^n]$	$N_{\boldsymbol{\mu}} + n_r + 1$	n_s
Parameters and sampled residual	$[\boldsymbol{\mu}; \mathbf{P}\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu})]$	$N_{\boldsymbol{\mu}} + n_s$	n_s
Parameters, sampled residual, and time	$[\boldsymbol{\mu}; \mathbf{P}\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}); t^n]$	$N_{\boldsymbol{\mu}} + n_s + 1$	n_s

Table 1: Summary of considered feature-engineering methods.

respectively, where $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}} \subset \mathcal{D}$; and $\mathbb{T}_{\text{train}} := \{\tau(1), \ldots, \tau(|\mathbb{T}_{\text{train}}|)\} \subseteq \mathbb{T}$, $\mathbb{T}_{\text{val}} := \{\tau(1), \ldots, \tau(|\mathbb{T}_{\text{val}}|)\} \subseteq \mathbb{T}$, and $\mathbb{T}_{\text{test}} := \{\tau(1), \ldots, \tau(|\mathbb{T}_{\text{test}}|)\} \subseteq \mathbb{T}$. For example, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$ can be generated by drawing samples from a probability distribution defined on the parameter domain $\mathcal{D}$, and the time indices can be set to $\mathbb{T}_{\text{train}} = \mathbb{T}_{\text{val}} = \mathbb{T}_{\text{test}} = \mathbb{T}$ (which we do in the numerical experiments).

3.1.3. Training data for principal component analysis and q-sampling

Feature 5 and Feature 6 require computing principal components of the residual; Feature 6 and Feature 7 require constructing the sampling matrix $\mathbf{P}$, which we do via q-sampling [11] (which in turn requires residual principal components). In these cases, computing residual principal components requires the additional training set

$$\mathcal{T}_{\boldsymbol{r},\text{train}} := \{\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}) \,|\, \boldsymbol{\mu} \in \mathcal{D}_{\text{train}}, n \in \mathbb{T}_{\text{train}}\}. \tag{31}$$

Because principal component analysis is an unsupervised machine learning method, it does not require validation or test sets.

3.1.4. Regression function

We now describe Step 3 of the proposed framework: training the deterministic regression-function model $\hat{\delta}_f^n(\boldsymbol{\mu})$ of the form (27)–(28). In particular, this step constructs the mappings $\hat{f}$ and $\mathbf{g}$. To achieve this, we consider a variety of both recursive and non-recursive regression methods from classical time-series analysis and supervised machine learning. In particular, we consider three categories of regression methods:

Category 1 **Non-recursive.** Non-recursive regression methods do not include latent variables or latent-variable dynamics. In this case, the latent variables can be omitted from the regression function arguments, i.e., $\hat{f}(\boldsymbol{\rho}, \boldsymbol{h}) \equiv \hat{f}(\boldsymbol{\rho})$, and the mapping $\mathbf{g}$ is not employed. EP: We note that this category of models are *physically-inconsistent* in the sense that they do not capture the non-local nature of the error as outlined in Sections 2.3.1 and 2.3.2; we include this category of regression models for benchmark purposes.

Category 2 **Recursive with latent state equal to previous prediction.** This category of methods employs latent variables, but sets them equal to the predicted response at the previous time instance, i.e., $\boldsymbol{h}^{\tau(n)} = \hat{\delta}_f^{\tau(n-1)}$, $n = 1, \ldots, \bar{N}_t$. This is equivalent to prescribing the recursion function as simply

$$\mathbf{g} : (\boldsymbol{\rho}, \boldsymbol{h}, \hat{\delta}_f) \mapsto \hat{\delta}_f. \tag{32}$$

These methods place no particular restriction on the mapping $\hat{f}$. EP: It is noted that we specifically consider Category 2 models as they correspond to the recursive form of, e.g., standard autoregressive models and *integrated* methods. We describe these approaches in more detail later in this section.

Category 3 **Recursive.** This category of methods follows the general framework outlined in Section 3.1 with no particular restrictions on the forms of the mappings $\hat{f}$ or $\mathbf{g}$.

Table 2 summarizes the considered methods and their associated categories.

- ***k*-nearest neighbors (kNN)** (Category 1): kNN is a simple regression method wherein the prediction comprises a weighted average of the nearest neighbors in feature space such that

$$\hat{f} : \boldsymbol{\rho} \mapsto \frac{1}{k} \sum_{(\bar{\delta}, \bar{\boldsymbol{\rho}}) \in \mathcal{T}_{\text{kNN}}(\boldsymbol{\rho})} w(\boldsymbol{\rho}, \bar{\boldsymbol{\rho}}) \bar{\delta} \tag{33}$$

where $\mathcal{T}_{\text{kNN}}(\boldsymbol{\rho}) \subseteq \mathcal{T}_{\text{train}}$ with $|\mathcal{T}_{\text{kNN}}(\boldsymbol{\rho})| = k$ denotes the k response–feature pairs of the training set $\mathcal{T}_{\text{train}}$ whose features are closest to the vector $\boldsymbol{\rho}$ in feature space. Common choices for the weights are uniform weights, $w : (\boldsymbol{\rho}, \bar{\boldsymbol{\rho}}) \mapsto 1/k$ and weights based on Euclidean distance, i.e.,

$$w : (\boldsymbol{\rho}, \bar{\boldsymbol{\rho}}) \mapsto \frac{(\sum_{(\tilde{\delta}, \tilde{\boldsymbol{\rho}}) \in \mathcal{T}_{\text{kNN}}(\boldsymbol{\rho})} \|\boldsymbol{\rho} - \tilde{\boldsymbol{\rho}}\|_2^{-1})^{-1}}{\|\boldsymbol{\rho} - \bar{\boldsymbol{\rho}}\|_2}. \tag{34}$$

Description	Description	Classification	Latent dynamics	Latent dimension $N_{\boldsymbol{h}}$
k-nearest neighbors	kNN	Category 1	None	N/A
Artificial neural network	ANN	Category 1	None	N/A
Autoregressive w/ exogenous inputs	ARX	Category 2	Linear	1
Integrated artificial neural network	ANN-I	Category 2	Linear	1
Latent autoregressive w/ exogenous inputs	LARX	Category 3	Linear	Arbitrary
Recurrent neural network	RNN	Category 3	Nonlinear	Arbitrary
Long short-term memory network	LSTM	Category 3	Nonlinear	Arbitrary

Table 2: Summary of considered regression methods.

The hyperparameters associated with this method correspond to the number of neighbors k, and the definition of the weights w. We note that all parameters characterizing this model are hyperparameters; there are no parameters that are subject to training via optimization.

- **Artificial neural network (ANN)** (Category 1): Feed-forward ANNs (i.e., multilayer perceptrons) employ function composition to create nonlinear input–output maps. In the context of error modeling, Ref. [15] observed ANNs to yield the best performance for static systems. In the present context, feed-forward ANNs can be used to construct the (parameterized) non-recursive regression function that satisfies $\hat{f}(\boldsymbol{\rho}, \boldsymbol{h}) \equiv \hat{f}(\boldsymbol{\rho})$ and is given by

$$\hat{f} : (\boldsymbol{\rho}; \boldsymbol{\theta}_f) \mapsto \boldsymbol{z}_d(\cdot, \boldsymbol{\theta}_d) \circ \cdots \circ \boldsymbol{z}_1(\boldsymbol{\rho}; \boldsymbol{\theta}_1),$$

where $\boldsymbol{\theta}_f \equiv (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d)$ with $\boldsymbol{\theta}_i \equiv (\boldsymbol{W}_i, \boldsymbol{b}_i) \in \mathbb{R}^{p_i \times p_{i-1}} \times \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ denotes neural-network weights and biases; $\boldsymbol{z}_i(\cdot; \boldsymbol{\theta}_i) : \mathbb{R}^{p_{i-1}} \to \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ denotes the activation function employed at the ith layer; d denotes the depth of the network; and p_i, $i = 0, \ldots, d$ (with $p_0 = N_\rho$ and $p_d = 1$) denotes the number of neurons at layer i. The activation function takes the form

$$\boldsymbol{z}_i : (\boldsymbol{w}; \boldsymbol{\theta}_i) \mapsto \boldsymbol{\zeta}_i(\boldsymbol{W}_i \boldsymbol{w} + \boldsymbol{b}_i), \quad i = 1, \ldots, d,$$

where $\boldsymbol{\zeta}_i : \mathbb{R}^{p_i} \to \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ denote activations (e.g., rectified linear unit (ReLU), hyperbolic tangent) applied elementwise.

In this work, we consider only fully connected feed-forward ANNs with ReLU activation functions. EP: We choose ReLU activation functions due to their wide-spread use in the machine-learning community; we note that in preliminary numerical experiments we found the specific choice of activation function to have minimal impact. As such, we consider the number of layers d and the number of neurons at each layer p_i, $i = 1, \ldots, d-1$ to be hyperparameters.

- **Autoregressive with exogenous inputs (ARX)** (Category 2): Autoregressive methods are linear models that have been widely adopted for time-series analysis. The ARX model is an extension of the autoregressive (AR) model that allows for exogenous inputs, which we treat as the features. ARX(p, r) methods regress the response at given time index as a linear combination of the value of the response at the previous p time instances and the exogenous inputs at both the current time instance and the previous $r-1$ time instances. This work considers the ARX$(1,1)$ model, in which case the (parameterized) regression function is given by

$$\hat{f} : (\boldsymbol{\rho}, \boldsymbol{h}; \boldsymbol{\theta}_f) \mapsto \boldsymbol{\theta}_1^T \boldsymbol{\rho} + \theta_2 \boldsymbol{h} + b, \tag{35}$$

where $\boldsymbol{\theta}_f \equiv (\boldsymbol{\theta}_1, \theta_2, b) \in \mathbb{R}^{N_\rho} \times \mathbb{R} \times \mathbb{R}$ denote the regression-function weights and bias, and the latent variable (of dimension $N_{\boldsymbol{h}} = 1$) corresponds to the predicted response at the previous time instance such that $\boldsymbol{h}^{\tau(n)} = \hat{\delta}_f^{\tau(n-1)}$, $n = 1, \ldots, \bar{N}_t$ and the latent-variables dynamics are prescribed via (32).

We remark that classical autoregressive models typically include the noise term directly within the recursion, i.e., the latent variables $\boldsymbol{h}^{\tau(n)}$ correspond to the regression-function prediction perturbed by stochastic noise. As a result, generating statistical predictions requires simulating an ensemble of model realizations. To avoid this complication, this work constructs the stochastic noise model *separately* from the deterministic regression-function model; for the stochastic noise models considered in this work, the statistical distribution of the response is given analytically. Sections 3.1.6–3.1.7 describe this step. We also note that it is possible to construct higher-order autoregressive models (i.e., AR(p, q) with $p > 1$ and/or $q > 1$) that include additional lagged states. While we do not consider such approaches in this work, the ARX formulation presented above could be modified to address this case by changing the definition of the latent variables and recursion function.

- **Integrated artificial neural network (ANN-I)** (Category 2): To facilitate time-series modeling, it is common to apply transformations to the training data in order to make the associated processes weakly stationary. The most commonly adopted transformation is time differencing. This process leads to *integrated* methods. Here, we investigate the use of an "integrated" artificial neural network (ANN-I) model, wherein a neural network is used to predict the time-differenced error. This leads to a modification of the ANN model wherein the (parameterized) recursive regression function takes the form
$$\hat{f} : (\boldsymbol{\rho}, \boldsymbol{h}; \boldsymbol{\theta}_f) \mapsto \boldsymbol{h} + \boldsymbol{z}_d(\cdot, \boldsymbol{\theta}_d) \circ \cdots \circ \boldsymbol{z}_1(\boldsymbol{\rho}, \boldsymbol{\theta}_1),$$
where the parameters $\boldsymbol{\theta}_f \equiv (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d)$ and activations $\boldsymbol{z}_i$, $i = 1, \ldots, d$ are defined as in the ANN model and the latent variable (of dimension $N_{\boldsymbol{h}} = 1$) corresponds to the predicted response at the previous time instance such that $\boldsymbol{h}^{\tau(n)} = \hat{\delta}_f^{\tau(n-1)}$, $n = 1, \ldots, \bar{N}_t$ and (32) holds. As with ANN, we consider only fully connected feed-forward networks with ReLU activation functions such that the hyperparameters correspond to the number of layers d and the number of neurons at each layer p_i, $i = 1, \ldots, d-1$.

- **Latent autoregressive with exogenous inputs (LARX)** (Category 3): Because ARX falls into Category 2, it employs the predicted response at the previous time instance as the latent variables according to Eq. (32). The LARX method relaxes this assumption and considers higher-dimensional latent variables governed by general linear dynamics. This equips the model with a higher capacity than ARX and thus enables it to capture more complex non-local dependencies. In particular, LARX defines the (parameterized) regression function as a linear mapping from the latent variables to the prediction that satisfies $\hat{f}(\boldsymbol{\rho}, \boldsymbol{h}) \equiv \hat{f}(\boldsymbol{h})$ and is given by
$$\hat{f} : (\boldsymbol{h}; \boldsymbol{\theta}_f) \mapsto \boldsymbol{\theta}_1^T \boldsymbol{h} + b$$
where $\boldsymbol{\theta}_f \equiv (\boldsymbol{\theta}_1, b) \in \mathbb{R}^{N_{\boldsymbol{h}}} \times \mathbb{R}$. The latent-variable recursion operator is linear in its first two arguments such that $\mathbf{g}(\boldsymbol{\rho}, \boldsymbol{h}, \hat{\delta}_f) \equiv \mathbf{g}(\boldsymbol{\rho}, \boldsymbol{h})$ with
$$\mathbf{g} : (\boldsymbol{\rho}, \boldsymbol{h}; \boldsymbol{\theta}_{\boldsymbol{g}}) \mapsto \boldsymbol{\Theta}_\rho \boldsymbol{\rho} + \boldsymbol{\Theta}_{\boldsymbol{h}} \boldsymbol{h} + \boldsymbol{b}_{\boldsymbol{h}},$$
where $\boldsymbol{\theta}_{\boldsymbol{g}} \equiv (\boldsymbol{\Theta}_\rho, \boldsymbol{\Theta}_{\boldsymbol{h}}, \boldsymbol{b}_{\boldsymbol{h}}) \in \mathbb{R}^{N_{\boldsymbol{h}} \times N_\rho} \times \mathbb{R}^{N_{\boldsymbol{h}} \times N_{\boldsymbol{h}}} \times \mathbb{R}^{N_{\boldsymbol{h}}}$.

  LARX is characterized by one hyperparameter: the number of latent variables $N_{\boldsymbol{h}}$.

- **Recurrent neural network (RNN)** (Category 3): RNNs [43] comprise a particular neural-network architecture that have been widely adopted for modeling time sequences. The main idea of an RNN is to employ the same feed-forward neural network for prediction at each point in a sequence, but to treat the hidden layer of the neural network as inputs to the neural network at the next point in the sequence; these hidden layers associate with the latent variables. RNNs comprise a generalization of LARX models: LARX methods develop a linear recursion for the latent-variable dynamics, while RNNs enable nonlinear latent-variable dynamics. Figure 2 illustrates the computational graph for an "unrolled" RNN. An RNN of depth d employs latent variables $\boldsymbol{h} \equiv (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d)$, with $\boldsymbol{h}_i \in \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ and $N_{\boldsymbol{h}} = \sum_{i=1}^{d} p_i$. For RNNs, the deterministic regression-function model is given by an output activation function acting on the latent variables at the final layer; it satisfies $\hat{f}(\boldsymbol{\rho}, \boldsymbol{h}) \equiv \hat{f}(\boldsymbol{h})$ and is given by

$$\hat{f} : (\boldsymbol{h}; \boldsymbol{\theta}_f) \mapsto \zeta(\boldsymbol{w}_f^T \boldsymbol{h}_d + b_f) \tag{36}$$

where $\boldsymbol{\theta}_f \equiv (\boldsymbol{w}_f, b_f) \in \mathbb{R}^{p_d} \times \mathbb{R}$ and $\zeta(\cdot)$ is typically the identity operator.

The latent variables satisfy the recursive relationships

$$\boldsymbol{h}_i^{\tau(n)} = \boldsymbol{r}_i(\boldsymbol{h}_{i-1}^{\tau(n)}, \boldsymbol{h}_i^{\tau(n-1)}; \boldsymbol{\theta}_i), \quad i = 1, \ldots, d, \quad n = 1, \ldots, \bar{N}_t, \tag{37}$$

where $\boldsymbol{r}_i(\cdot, \cdot, \boldsymbol{\theta}_i) : \mathbb{R}^{p_{i-1}} \times \mathbb{R}^{p_i} \to \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ denote activation functions and the latent state at layer zero is defined from the input features such that $\boldsymbol{h}_0^{\tau(n)} = \boldsymbol{h}_0(\boldsymbol{\rho}^{\tau(n)})$ with $\boldsymbol{h}_0 : \mathbb{R}^{N_\rho} \to \mathbb{R}^{p_0}$. These recursive relationships can be expressed by a latent-variable recursion operator, which satisfies $\mathbf{g}(\boldsymbol{\rho}, \boldsymbol{h}, \hat{\delta}_f) \equiv \mathbf{g}(\boldsymbol{\rho}, \boldsymbol{h})$ and is given by

$$\mathbf{g} : (\boldsymbol{\rho}, \boldsymbol{h}; \boldsymbol{\theta}_{\mathbf{g}}) \mapsto \begin{bmatrix} \boldsymbol{r}_1(\boldsymbol{h}_0(\boldsymbol{\rho}), \boldsymbol{h}_1; \boldsymbol{\theta}_1) \\ \boldsymbol{r}_2(\cdot, \boldsymbol{h}_2; \boldsymbol{\theta}_2) \circ \boldsymbol{r}_1(\boldsymbol{h}_0(\boldsymbol{\rho}), \boldsymbol{h}_1; \boldsymbol{\theta}_1) \\ \vdots \\ \boldsymbol{r}_d(\cdot, \boldsymbol{h}_d; \boldsymbol{\theta}_d) \circ \cdots \circ \boldsymbol{r}_1(\boldsymbol{h}_0(\boldsymbol{\rho}), \boldsymbol{h}_1; \boldsymbol{\theta}_1) \end{bmatrix}, \tag{38}$$

where $\boldsymbol{\theta}_{\mathbf{g}} \equiv (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d)$.

For standard RNNs, the function applied at the ith layer takes the form

$$\boldsymbol{r}_i : (\boldsymbol{h}_{i-1}, \boldsymbol{h}_i; \boldsymbol{\theta}_i) \mapsto \boldsymbol{\zeta}_i(\boldsymbol{W}_{i,1}\boldsymbol{h}_{i-1} + \boldsymbol{W}_{i,2}\boldsymbol{h}_i + \boldsymbol{b}_i), \tag{39}$$

where $\boldsymbol{\theta}_i \equiv (\boldsymbol{W}_{i,1}, \boldsymbol{W}_{i,2}, \boldsymbol{b}_i) \in \mathbb{R}^{p_i \times p_{i-1}} \times \mathbb{R}^{p_i \times p_i} \times \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ denote the recursive weights and biases, $\boldsymbol{\zeta}_i : \mathbb{R}^{p_i} \to \mathbb{R}^{p_i}$, $i = 1, \ldots, d$ again denote activations (e.g., ReLU) applied elementwise, and $\boldsymbol{h}_0$ is the identity operator such that $\boldsymbol{h}_0^{\tau(n)} = \boldsymbol{\rho}^{\tau(n)}$, $n = 1, \ldots, \bar{N}_t$ and $p_0 = N_\rho$.

Different RNN architectures can be obtained, for example, by modifying the activation functions, the number of latent variables, or the network depth. The LARX method can be recovered from RNN by setting the depth $d = 1$ and setting $\zeta(\cdot)$ and $\boldsymbol{\zeta}_1(\cdot)$ to be the identity mappings. We consider hyperparameters corresponding to the number of neurons at each layer and the depth of the network.

- **Long short-term memory network (LSTM)** (Category 3): The LSTM network [20] is a particular type of RNN designed to capture dependencies on long-term non-local quantities. Figure 3 provides a diagram of an "unrolled" LSTM. LSTMs are able to capture long-term non-local dependencies through the use of two latent variables: the *hidden state* and the *cell state*. The cell state, which is depicted as the topmost horizontal line running through the cells in Figure 3, undergoes only minor interactions [33]: elementwise multiplication and elementwise addition. Due to the minor changes experienced by the cell state, it can (in principle) more easily capture long-term dependencies. The LSTM architecture is arguably the most widely adopted RNN architecture as of this writing.

  Formally, an LSTM decomposes latent variables at the ith layer as $\boldsymbol{h}_i \equiv \left(\tilde{\boldsymbol{h}}_i, \boldsymbol{c}_i\right)$, where $\tilde{\boldsymbol{h}}_i \in \mathbb{R}^{p_i/2}$ and $\boldsymbol{c}_i \in \mathbb{R}^{p_i/2}$ denote the hidden state and cell state, respectively. The deterministic regression function for LSTMs is defined as in (36), but with elements $p_i/2 + 1$ to p_i of the weighting vector $\boldsymbol{w}_f$ set to zero so that the output depends only on $\tilde{\boldsymbol{h}}_d$.

  The latent-variable recursion is also defined as in (37)–(38); however, the activations take a special form based on the LSTM "cell". In particular, the LSTM cell corresponds to defining the ith layer's

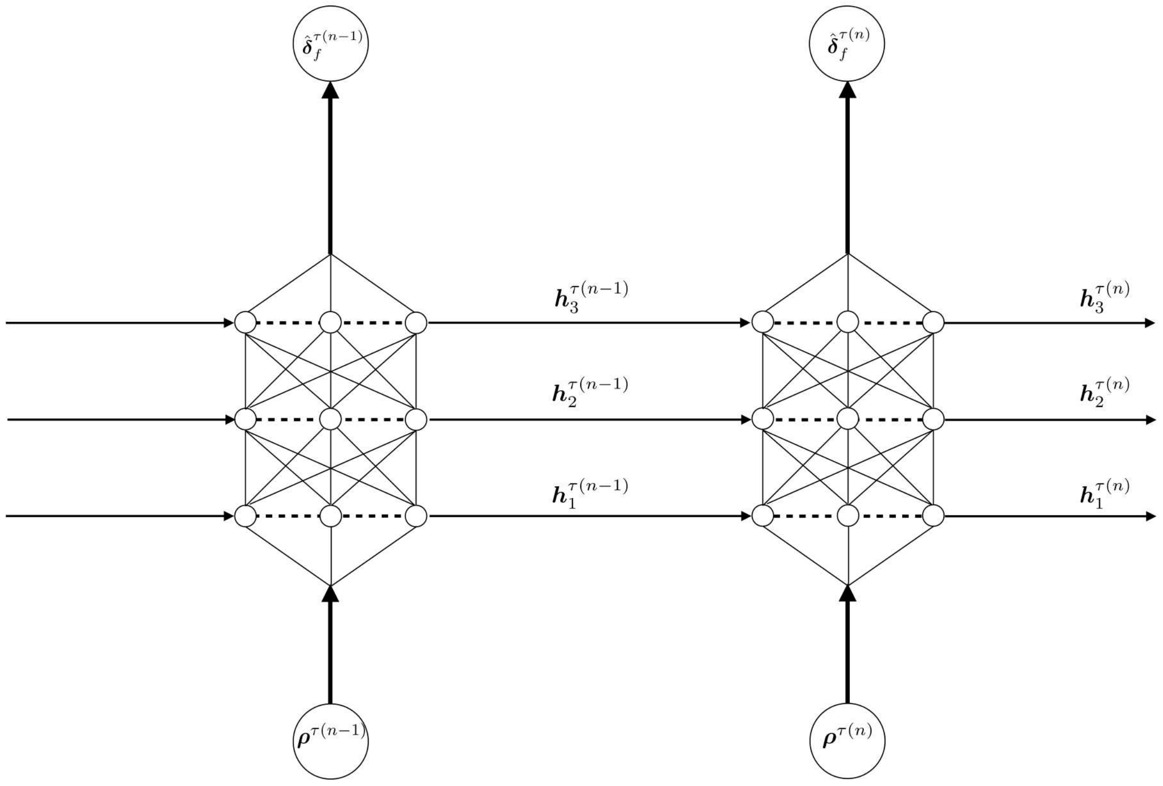

$\hat{\boldsymbol{\delta}}_f^{\tau(n-1)}$
$\hat{\boldsymbol{\delta}}_f^{\tau(n)}$
$\boldsymbol{h}_3^{\tau(n-1)}$
$\boldsymbol{h}_3^{\tau(n)}$
$\boldsymbol{h}_2^{\tau(n-1)}$
$\boldsymbol{h}_2^{\tau(n)}$
$\boldsymbol{h}_1^{\tau(n-1)}$
$\boldsymbol{h}_1^{\tau(n)}$
$\boldsymbol{\rho}^{\tau(n-1)}$
$\boldsymbol{\rho}^{\tau(n)}$

Figure 2: Diagram of an “unrolled” recurrent neural network with depth $d = 3$.

function as

$$\boldsymbol{r}_i : (\boldsymbol{h}_{i-1}, \boldsymbol{h}_i; \boldsymbol{\theta}_i) \mapsto \begin{bmatrix} \boldsymbol{r}_{\sigma,i}(\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,1}) \odot \tanh(\boldsymbol{r}_{\text{cell},i}(\tilde{\boldsymbol{h}}_{i-1}, \boldsymbol{h}_i; \boldsymbol{\theta}_i)) \\ \boldsymbol{r}_{\text{cell},i}(\tilde{\boldsymbol{h}}_{i-1}, \boldsymbol{h}_i; \boldsymbol{\theta}_i) \end{bmatrix}, \tag{40}$$

where $\boldsymbol{h}_i \equiv (\tilde{\boldsymbol{h}}_i, \boldsymbol{c}_i)$, $\boldsymbol{\theta}_i \equiv (\boldsymbol{\theta}_{i,1}, \ldots, \boldsymbol{\theta}_{i,4})$, and $\odot$ denotes the elementwise (Hadamard) product, and

$$\boldsymbol{r}_{\text{cell},i} : (\tilde{\boldsymbol{h}}_{i-1}, \boldsymbol{h}_i; \boldsymbol{\theta}_i) \mapsto \boldsymbol{r}_{\sigma,i}(\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,2}) \odot \boldsymbol{c}_i + \boldsymbol{r}_{\sigma,i}(\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,3}) \odot \boldsymbol{r}_{\tanh,i}(\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,4})$$

denotes the cell activation. Here,

$$\begin{aligned} \boldsymbol{r}_{\sigma,i} &: (\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,j}) \mapsto \sigma(\boldsymbol{W}_{i,j,1}\tilde{\boldsymbol{h}}_{i-1} + \boldsymbol{W}_{i,j,2}\tilde{\boldsymbol{h}}_i + \boldsymbol{b}_{i,j}), \quad j = 1, \ldots, 3 \\ \boldsymbol{r}_{\tanh,i} &: (\tilde{\boldsymbol{h}}_{i-1}, \tilde{\boldsymbol{h}}_i; \boldsymbol{\theta}_{i,4}) \mapsto \tanh(\boldsymbol{W}_{i,4,1}\tilde{\boldsymbol{h}}_{i-1} + \boldsymbol{W}_{i,4,2}\tilde{\boldsymbol{h}}_i + \boldsymbol{b}_{i,4}), \end{aligned}$$

where $\boldsymbol{\theta}_{i,j} \equiv (\boldsymbol{W}_{i,j,1}, \boldsymbol{W}_{i,j,2}, \boldsymbol{b}_{i,j}) \in \mathbb{R}^{p_i/2 \times p_{i-1}/2} \times \mathbb{R}^{p_i/2 \times p_i/2} \times \mathbb{R}^{p_i/2}$, $i = 1, \ldots, d$, $j = 1, \ldots, 4$ denote the sigmoid and hyperbolic tangent operators, respectively. For LSTM, the latent state at layer zero is defined via $\boldsymbol{h}_0 : \boldsymbol{\rho} \mapsto [\boldsymbol{\rho}^T\ \boldsymbol{0}^T]^T$ such that such that $\tilde{\boldsymbol{h}}_0^{\tau(n)} = \boldsymbol{\rho}^{\tau(n)}$, $n = 1, \ldots, \bar{N}_t$ and $p_0 = 2N_\rho$.

Different types of LSTM networks can be constructed, e.g., by modifying the definition of the LSTM cell. For instance, the "peephole" LSTM network can be obtained by replacing the hyperbolic tangent function in Eq. (40) with the identity function. We consider hyperparameters corresponding to the number of neurons p_i, $i = 1, \ldots, d$ and the network depth d.

To summarize, this section outlined a variety of non-recursive and recursive regression-function models. This exposition started with a description of the kNN and ANN regression-function models. These regression-function models are non-recurrent and employ no latent variables; non-local effects cannot be captured with kNN and ANN. To capture non-local effects, we introduced the ARX and ANN-I regression-function models. Both of these models employ a single latent variable corresponding to the prediction at the previous time instance; both methods are thus characterized by linear latent dynamics with a latent dimension of $N_{\boldsymbol{h}} = 1$. To increase the capacity of the latent-dynamics recursion, we introduced the LARX regression-function model. Unlike ARX and ANN-I, LARX employs an arbitrary number of latent variables governed by general linear dynamics. Finally, to further increase the capacity of the latent dynamics recursion, we introduced the RNN and LSTM regression-function models. These methods comprise an extension of LARX as, in addition to supporting an arbitrary number of latent variables, they allow for nonlinear latent-variable dynamics. The regression-function models described in this section are thus characterized by the following hierarchy of models with increasing capacity: no latent variables (kNN, ANN), one latent variable with linear latent-variable dynamics (ARX, ANN-I), an arbitrary number of latent variables with linear latent-variable dynamics (LARX), and an arbitrary number of latent variables with nonlinear latent-variable dynamics (RNN, LSTM).

3.1.5. Regression-function training

To train a regression model, we employ a validation procedure consisting of two loops: the outer loop modifies hyperparameters characterizing the model, while the inner loop computes optimal model parameters $\boldsymbol{\theta}_f^\star$ and $\boldsymbol{\theta}_{\boldsymbol{g}}^\star$ via optimization executing on the training set $\mathcal{T}_{\text{train}}$ given fixed hyperparameter values. The final model is chosen based on hyperparameter values that yield the smallest MSE on the validation set $\mathcal{T}_{\text{val}}$. In the case of limited data, this process can be replaced by executing cross-validation within the training set $\mathcal{T}_{\text{train}}$; this obviates the need for an independent validation set $\mathcal{T}_{\text{val}}$.

The inner-loop optimization problem depends on the category within which the regression model falls.[2]

[2] We note that kNN is characterized by only hyperparameters and thus the inner loop does not require solving an optimization problem.

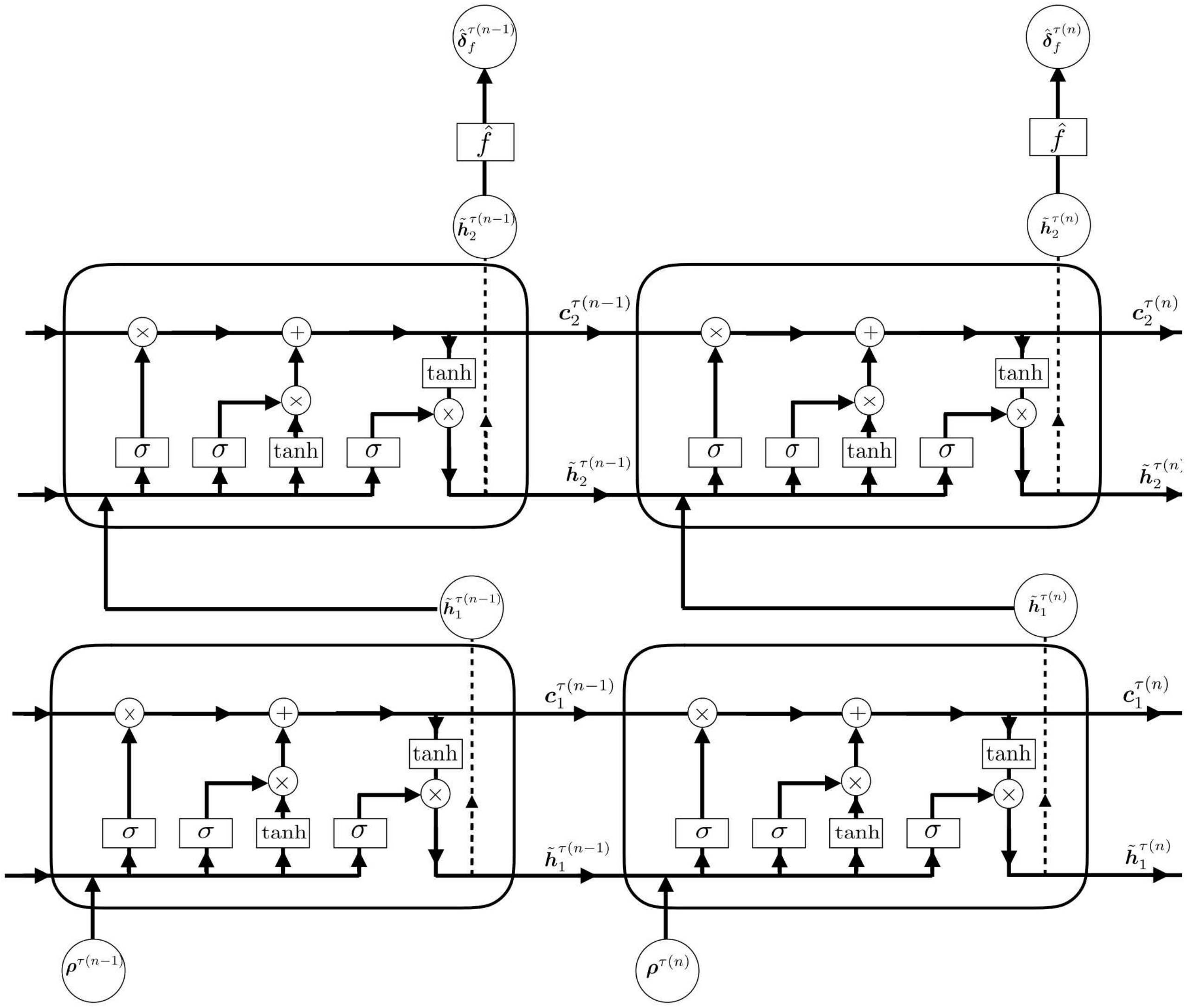

$\hat{\boldsymbol{\delta}}_f^{\tau(n-1)}$
$\hat{\boldsymbol{\delta}}_f^{\tau(n)}$
$\hat{f}$
$\hat{f}$
$\tilde{\boldsymbol{h}}_2^{\tau(n-1)}$
$\tilde{\boldsymbol{h}}_2^{\tau(n)}$
$\boldsymbol{c}_2^{\tau(n-1)}$
$\boldsymbol{c}_2^{\tau(n)}$
$\tilde{\boldsymbol{h}}_2^{\tau(n-1)}$
$\tilde{\boldsymbol{h}}_2^{\tau(n)}$
$\tilde{\boldsymbol{h}}_1^{\tau(n-1)}$
$\tilde{\boldsymbol{h}}_1^{\tau(n)}$
$\boldsymbol{c}_1^{\tau(n-1)}$
$\boldsymbol{c}_1^{\tau(n)}$
$\tilde{\boldsymbol{h}}_1^{\tau(n-1)}$
$\tilde{\boldsymbol{h}}_1^{\tau(n)}$
$\boldsymbol{\rho}^{\tau(n-1)}$
$\boldsymbol{\rho}^{\tau(n)}$
σ
tanh
$\times$
$+$

Figure 3: Diagram for a deterministic regression-function model utilizing the LSTM architecture with a depth of $d = 2$. Figure concept adopted from Ref. [33].

1. For regression models in Category 1, we compute optimal model parameters $\boldsymbol{\theta}_f^\star$ as the solution to the minimization problem

$$\underset{\boldsymbol{\theta}_f}{\text{minimize}} \sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{train}}} \sum_{n\in\mathbb{T}_{\text{train}}} \left\| \delta^n(\boldsymbol{\mu}) - \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}); \boldsymbol{\theta}_f) \right\|_2^2 + \kappa_f(\boldsymbol{\theta}_f), \tag{41}$$

where $\kappa_f(\cdot) \in \mathbb{R}_+$ is a regularization operator, e.g., LASSO, ridge, elastic–net. We note that optimization problem (41) is nonrecursive, as the prediction at one time instance does not affect the objective-function contribution associated with the prediction at the next time instance.

2. For regression models in Category 2, we can train the models in one of two ways. The first approach is non-recursive in nature and computes the optimal model parameters $\boldsymbol{\theta}_f^\star$ as the solution to the minimization problem

$$\underset{\boldsymbol{\theta}_f}{\text{minimize}} \sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{train}}} \sum_{n=1}^{|\mathbb{T}_{\text{train}}|} \left\| \delta^{\tau(n)}(\boldsymbol{\mu}) - \hat{f}(\boldsymbol{\rho}^{\tau(n)}(\boldsymbol{\mu}), \delta^{\tau(n-1)}(\boldsymbol{\mu}); \boldsymbol{\theta}_f) \right\|_2^2 + \kappa_f(\boldsymbol{\theta}_f). \tag{42}$$

Because the actual error $\delta^{\tau(n-1)}(\boldsymbol{\mu})$ from the training data is employed as the second argument in $\hat{f}$ rather than the predicted value $\hat{\delta}_f^{\tau(n-1)}$ (which is the actual value of the hidden state), the optimization problem (42) is non-recursive. We refer to this approach as non-recursive training (NRT).

3. Finally, regression models in either Category 2 and Category 3 can be trained via a dynamics-constrained recursive training strategy. In this approach, the optimal weights $\boldsymbol{\theta}_f^\star$ and $\boldsymbol{\theta}_{\boldsymbol{g}}^\star$ are the solution to the problem

$$\begin{aligned} \underset{\boldsymbol{\theta}_f, \boldsymbol{\theta}_{\boldsymbol{g}}}{\text{minimize}} & \sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{train}}} \sum_{n\in\mathbb{T}_{\text{train}}} \left\| \delta^n(\boldsymbol{\mu}) - \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}), \boldsymbol{h}^n(\boldsymbol{\mu}); \boldsymbol{\theta}_f) \right\|_2^2 + \kappa_f(\boldsymbol{\theta}_f) + \kappa_{\boldsymbol{g}}(\boldsymbol{\theta}_{\boldsymbol{g}}) \\ \text{subject to } & \boldsymbol{h}^{\tau(n)}(\boldsymbol{\mu}) = \mathbf{g}(\boldsymbol{\rho}^{\tau(n)}(\boldsymbol{\mu}), \boldsymbol{h}^{\tau(n-1)}(\boldsymbol{\mu}), \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu}); \boldsymbol{\theta}_{\boldsymbol{g}}), \quad \boldsymbol{\mu} \in \mathcal{D}_{\text{train}},\ n = 1, \ldots, |\mathbb{T}_{\text{train}}|, \end{aligned} \tag{43}$$

where $\kappa_{\boldsymbol{g}}(\cdot) \in \mathbb{R}_+$ is a regularization operator and the latent-variable dynamics appear explicitly as constraints. In this approach, the prediction $\hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu})$ *does* affect the contribution to the objective function at time instance $\tau(n)$ and parameters $\boldsymbol{\mu}$ through the constraints. Thus, this formulation reflects the recursive nature of the regression model and can enable more accurate long-time predictions at the expense of more computationally expensive training. We refer to this approach as recursive training (RT). We note that the "backpropagation through time" (BPTT) method [32, 41, 17] is a popular approach for solving the minimization problem (43). We additionally note that for models in Category 2, problem (43) can be derived by substituting $\delta^{\tau(n-1)}(\boldsymbol{\mu}) \leftarrow \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu})$ in problem (42).

In the outer loop, for each candidate value of the hyperparameters, optimization problem (41), (42), or (43) EP: is solved for either $\boldsymbol{\theta}_f^\star$ (optimization problems (41) and (42)) or $\boldsymbol{\theta}_f^\star$ and $\boldsymbol{\theta}_{\boldsymbol{g}}^\star$ (optimization problem (43)), and the resulting model is evaluated on the validation set $\mathcal{T}_{\text{val}}$. For regression models in Category 1, we evaluate the validation criterion

$$\sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{val}}} \sum_{n\in\mathbb{T}_{\text{val}}} \left\| \delta^n(\boldsymbol{\mu}) - \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}); \boldsymbol{\theta}_f^\star) \right\|_2^2, \tag{44}$$

while for methods in Category 2 or Category 3, we evaluate the validation criterion

$$\sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{val}}} \sum_{n\in\mathbb{T}_{\text{val}}} \left\| \delta^n(\boldsymbol{\mu}) - \hat{f}(\boldsymbol{\rho}^n(\boldsymbol{\mu}), \boldsymbol{h}^n(\boldsymbol{\mu}); \boldsymbol{\theta}_f^\star) \right\|_2^2, \tag{45}$$

where the dynamics of the latent variables are defined as $\boldsymbol{h}^{\tau(n)}(\boldsymbol{\mu}) = \mathbf{g}(\boldsymbol{\rho}^{\tau(n)}(\boldsymbol{\mu}), \boldsymbol{h}^{\tau(n-1)}(\boldsymbol{\mu}), \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu}); \boldsymbol{\theta}_{\boldsymbol{g}}^\star)$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{val}}$, $n = 1, \ldots, |\mathbb{T}_{\text{val}}|$. For each regression model, the hyperparameter values yielding the smallest validation-criterion value are selected.

Description	Noise Distribution	Time-dependent
Stationary Gaussian	Gaussian	No
Stationary Laplacian	Laplacian	No
Autoregressive 1 (AR1)	Gaussian	Yes

Table 3: Summary of considered noise models.

3.1.6. Noise model

We now describe Step 4 of the proposed framework: training the stochastic noise model $\hat{\delta}_{\epsilon}^{n}(\boldsymbol{\mu})$ of the form (29). While complex noise models could be considered in principle, for the sake of simplicity, we consider here only three types of noise models: a homoscedastic Gaussian model, a homoscedastic Laplacian model, and an autoregressive model.

- **Stationary Gaussian homoscedastic noise model**: Arguably, the simplest way to quantify the discrepancy between the error model and the true error is to assume the errors to be uncorrelated, Gaussian, stationary, and homoscedastic. The resulting error model is independent of the prediction at the previous time instance such that $\hat{\epsilon}(\hat{\delta}_{\epsilon}, \epsilon) \equiv \hat{\epsilon}(\epsilon)$ with

$$\hat{\epsilon} : \epsilon \mapsto \epsilon, \tag{46}$$

  and defining $\epsilon^n \in \mathbb{V}$, $n \in \mathbb{T}$ as a sequence of independent and identically distributed mean-zero Gaussian random variables, i.e.,

$$\epsilon^n \sim \mathcal{N}(0, \sigma^2), \quad n \in \mathbb{T}. \tag{47}$$

  This model does not have any latent variables and requires estimating only the variance parameter σ^2.

- **Stationary Laplacian homoscedastic noise model**: This approach is similar to the Gaussian model, but instead models the random variables as Laplacian, i.e., (46) holds with

$$\epsilon^n \sim \mathcal{L}(0, b), \quad n \in \mathbb{T},$$

  where $b > 0$ denotes the diversity, which must be estimated. Again, this error model does not require any input features or latent variables.

- **Autoregressive Gaussian homoscedastic discrepancy model:** Finally, we consider an autoregressive noise model, as we expect errors to display time correlations. We refer to this model as AR1. The model is given by

$$\hat{\epsilon} : (\hat{\delta}_{\epsilon}, \epsilon) \mapsto c\hat{\delta}_{\epsilon} + \epsilon,$$

  where $c \in \mathbb{R}$ is a model constant, and defining $\epsilon^n \in \mathbb{V}$, $n \in \mathbb{T}$ as a sequence of independent and identically distributed mean-zero Gaussian random variables such that (47) holds. This method requires estimating the parameters c and σ^2. The auto-regressive model produces a series of (correlated) mean-zero Gaussian distributions where the variance at time-instance $\tau(n)$ is defined recursively as

$$\sigma_{\epsilon}^{\tau(n)} := \sqrt{\sigma^2 + \sum_{i=1}^{n-1} (c\sigma^{\tau(i)})^2}, \qquad n = 1, \ldots, \bar{N}_t. \tag{48}$$

3.1.7. Noise-model training

The stationary Gaussian, stationary Laplacian, and AR1 noise models require specification of the parameters σ^2, b, and (c, σ^2), respectively. We compute these parameters via maximum likelihood estimation (MLE) on a training set $\mathcal{T}_{\text{train},\epsilon} \subseteq \mathcal{T}_{\text{test}}$, and evaluate the performance of the noise model on a test set $\mathcal{T}_{\text{test},\epsilon} \subseteq \mathcal{T}_{\text{test}}$, where $\mathcal{T}_{\text{train},\epsilon} \cap \mathcal{T}_{\text{test},\epsilon} = \emptyset$ and

$$\mathcal{T}_{\text{train},\epsilon} := \mathcal{T}(\mathcal{D}_{\text{train},\epsilon}, \mathbb{T}_{\text{train},\epsilon}), \qquad \mathcal{T}_{\text{test},\epsilon} := \mathcal{T}(\mathcal{D}_{\text{test},\epsilon}, \mathbb{T}_{\text{test},\epsilon}). \tag{49}$$

In this work, we employ $\mathcal{D}_{\text{train},\epsilon}, \mathcal{D}_{\text{test},\epsilon} \subseteq \mathcal{D}_{\text{test}}$ with $\mathcal{D}_{\text{train},\epsilon} \cap \mathcal{D}_{\text{test},\epsilon} = \emptyset$ and $\mathcal{D}_{\text{train},\epsilon} \cup \mathcal{D}_{\text{test},\epsilon} = \mathcal{D}_{\text{test}}$ with $\mathbb{T}_{\text{train},\epsilon} = \mathbb{T}_{\text{test},\epsilon} = \mathbb{T}$. For the stationary Gaussian model, the MLE estimate for σ^2 is given by

$$\sigma^2_{\text{MLE}} = \frac{1}{|\mathcal{T}_{\text{train},\epsilon}|} \sum_{\boldsymbol{\mu} \in \mathcal{D}_{\text{train},\epsilon}} \sum_{n \in \mathbb{T}_{\text{train},\epsilon}} (\delta^n(\boldsymbol{\mu}) - \hat{\delta}^n_f(\boldsymbol{\mu}))^2. \tag{50}$$

For the stationary Laplacian model, the MLE estimate for b is given by

$$b_{\text{MLE}} = \frac{1}{|\mathcal{T}_{\text{train},\epsilon}|} \sum_{\boldsymbol{\mu} \in \mathcal{D}_{\text{train},\epsilon}} \sum_{n \in \mathbb{T}_{\text{train},\epsilon}} |\delta^n(\boldsymbol{\mu}) - \hat{\delta}^n_f(\boldsymbol{\mu})|. \tag{51}$$

Lastly, for the AR1 model, the MLE estimates for the parameters are given by

$$\begin{aligned} c_{\text{MLE}} &= \frac{\sum_{\boldsymbol{\mu} \in \mathcal{D}_{\text{train},\epsilon}} \sum_{n=1}^{|\mathbb{T}_{\text{train},\epsilon}|} (\delta^{\tau(n-1)}(\boldsymbol{\mu}) - \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu}))(\delta^{\tau(n)}(\boldsymbol{\mu}) - \hat{\delta}_f^{\tau(n)}(\boldsymbol{\mu}))}{\sum_{\boldsymbol{\mu} \in \mathcal{D}_{\text{train},\epsilon}} \sum_{n=1}^{|\mathbb{T}_{\text{train},\epsilon}|} (\delta^{\tau(n-1)}(\boldsymbol{\mu}) - \hat{\delta}_f^{\tau(n-1)}(\boldsymbol{\mu}))^2}, \\ \sigma^2_{\text{MLE}} &= \frac{1}{|\mathcal{T}_{\text{train},\epsilon}|} \sum_{\boldsymbol{\mu} \in \mathcal{D}_{\text{train},\epsilon}} \sum_{n=1}^{|\mathbb{T}_{\text{train},\epsilon}|} (c_{\text{MLE}} \hat{\delta}_\epsilon^{\tau(n-1)}(\boldsymbol{\mu}) - (\delta^{\tau(n)}(\boldsymbol{\mu}) - \hat{\delta}_f^{\tau(n)}(\boldsymbol{\mu})))^2. \end{aligned} \tag{52}$$

4. Numerical experiments

This section investigates the performance of the proposed T-MLEM methods. In particular, we consider all combinations of the 13 feature-engineering methods reported in Table 1 with the 7 deterministic regression function models listed in Table 2. For the regression methods in Category 2 (i.e., ARX and ANN-I), we construct two separate regression-function models: one using non-recursive training (NRT) and one using recursive training (RT) as described in Section 3.1.5. In total, we consider 117 candidate T-MLEM methods. We also compare the performance of the T-MLEM methods with that achieved by time-local Gaussian-process (GP) regression in parameter space [35]; Appendix D describes this method in detail.

We consider three examples. The first two examples employ approximate solutions generated by a ROM constructed via Galerkin projection (as discussed in Section 2.1.3), while the third example considers approximate solutions generated by a coarse-mesh lower-fidelity model (as discussed in Section 2.1.2).

4.1. Implementation and model evaluation details

We now provide details on the implementation of the machine-learning regression methods, training algorithms, hyperparameter grids, and evaluation metrics.

4.1.1. Training and model selection of machine learning algorithms

We implement k-nearest neighbors and GP regression algorithms with Scikit-learn [38], and all other machine-learning regression methods with Keras [10]. We employ the following workflow to train and test all machine-learning regression models:

1. Define the coarse time grid $\mathbb{T}$ on which regression methods will operate.
2. Define the training set $\mathcal{T}_{\text{train}}$, validation set $\mathcal{T}_{\text{val}}$, and test set $\mathcal{T}_{\text{test}}$ according to (30) with $\mathbb{T}_{\text{train}} = \mathbb{T}_{\text{val}} = \mathbb{T}_{\text{test}} = \mathbb{T}$ and $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$ generated by drawing independent samples of the parameters $\boldsymbol{\mu} \sim \mathcal{U}(\mathcal{D})$, where $\mathcal{U}(\cdot)$ denotes the uniform distribution. We ensure that $|\mathcal{T}_{\text{train}}| = 4|\mathcal{T}_{\text{val}}|$ (i.e., we employ an 80/20 split between training and validation).
    - For Feature 5, Feature 6, and Feature 7, additionally extract the residual training set $\mathcal{T}_{\boldsymbol{r},\text{train}}$ from $\mathcal{D}_{\text{train}}$ and $\mathbb{T}_{\text{train}}$ according to (31).
3. Standardize features and responses according to their respective means and variances on the training set $\mathcal{T}_{\text{train}}$.

4. Define the hyperparameter grid for each considered regression model.

5. For each value of the hyperparameters, train the regression model by solving optimization problem (41), (42), or (43), depending on the category of the method. For the ARX, LARX, ANN, ANN-I, RNN, and LSTM models, we employ the Adam algorithm [26] with early stopping (assessed on a 20% holdout set) for this purpose. For regression methods employing the recursive training (RT) approach, as described in Section 3.1.5, BPTT is used to solve the optimization problem. Within the optimization problems, we employ ridge regularization such that $\kappa_f(\cdot) \equiv \alpha\|\cdot\|_2^2$ and $\kappa_{\boldsymbol{g}}(\cdot) \equiv \alpha\|\cdot\|_2^2$, where $\alpha \in \mathbb{R}_+$ is a regularization hyperparameter. As the Adam algorithm is stochastic, each model is trained 20 times. For the interested reader, we note that Appendix A presents results for all 20 training runs for the first numerical experiment.

6. For each regression method, select the model—which corresponds to one value of the hyperparameters and one execution of the Adam algorithm—by evaluating validation criterion (44) or (45), depending on the category of the method.

7. Define the noise training set $\mathcal{T}_{\text{train},\epsilon}$ and test set $\mathcal{T}_{\text{test},\epsilon}$ according to (49) with $\mathbb{T}_{\text{train},\epsilon} = \mathbb{T}_{\text{test},\epsilon} = \mathbb{T}$, $\mathcal{D}_{\text{train},\epsilon}$ drawn randomly from $\mathcal{D}_{\text{test}}$, and $\mathcal{D}_{\text{test},\epsilon} = \mathcal{D}_{\text{test}} \setminus \mathcal{D}_{\text{train},\epsilon}$.

8. Train stochastic noise models corresponding to a stationary Gaussian, stationary Laplacian, or AR1 via (50), (51), and (52), respectively.

9. Evaluate the trained noise models using statistical validation criteria (e.g., empirical prediction intervals, Komolgorov–Smirnov test) on the test set $\mathcal{T}_{\text{test},\epsilon}$.

As discussed in Section 3.1.5, the validation procedure described in steps 5–6 could be replaced with a cross-validation process.

4.1.2. Hyperparameter grid and optimization

We now describe the specific hyperparameter grid employed by each considered regression method as defined in step 4.

- ***k*-nearest neighbors (kNN)**: The hyperparameters for kNN correspond to the number of neighbors k, and the definition of the weights w. We employ the hyperparameter grid

$$(k, \text{weighting function}) \in \{1, \ldots, 5\} \times \{\text{uniform, Euclidean distance}\}.$$

- **Artificial neural network (ANN)**: The hyperparameters for ANN correspond to the network depth d and the number of neurons per layer p, which we set to be constant such that $p_i = p$, $i = 1, \ldots, d$. We also consider the regularization parameter α that appears in ridge regularization terms $\kappa_f(\cdot) \equiv \alpha\|\cdot\|_2^2$ and $\kappa_{\boldsymbol{g}}(\cdot) \equiv \alpha\|\cdot\|_2^2$ as a hyperparameter. We employ the hyperparameter grid

$$(d, p, \alpha) \in \{1, 2\} \times \{10, 25, 50, 100\} \times \{10^{-i}\}_{i=1}^{5}.$$

  Listing 1 provides the Python code for the Keras implementation of this method.

- **Autoregressive with exogenous inputs (ARX)**: Because we consider only ARX(1,1) methods, the only hyperparameter for this technique is the regularization parameter α; we employ the hyperparameter grid

$$\alpha \in \{10^{-i}\}_{i=1}^{5}.$$

  We construct ARX models using both the NRT and RT approaches (see Section 3.1.5). We refer to the resulting models as ARX (NRT) and ARX (RT). Listing 2 provides the Python code for the Keras implementation of ARX (RT).

- **Integrated artificial neural network (ANN-I)**: For ANN-I, we employ the same hyperparameter grid as for ANN. As with ARX, we construct two different ANN-I models: one with NRT and one with RT. We refer to the resulting models as ANN-I (NRT) and ANN-I (RT). Listing 3 provides the Python code for the Keras implementation of ANN-I (RT).
- **Latent autoregressive with exogenous inputs (LARX)**: The hyperparameter for LARX is the number of latent variables $N_{\boldsymbol{h}}$. We also consider the regularization term α and employ the hyperparameter grid
$$(N_{\boldsymbol{h}}, \alpha) \in \{10, 25, 50, 100\} \times \{10^{-i}\}_{i=1}^{5}.$$
Listing 4 provides the Python code for the Keras implementation of LARX.
- **Recurrent neural network (RNN)**: For RNN, we employ the same hyperparameter grid as for ANN. Listing 5 reports the Python code used to develop the RNNs used in this work; we employ Keras' `SimpleRNN` cell.
- **Long short-term memory network (LSTM)**: For LSTM, we employ the same hyper-parameter grid as for ANN. Listing 6 gives the Python code used to develop the LSTM networks in this work; we employ Keras' `LSTM` cell.
- **GP regression**: GP regression is characterized by one hyperparameter: the noise magnitude λ. We employ the hyperparameter grid
$$\lambda \in \{10^{-8+\frac{8(i-1)}{19}}\}_{i=1}^{20}.$$

Additionally, for feature methods employing the principal components of the residual or residual samples, the number of principal components n_r and number of residual samples n_s can be viewed as hyperparameters. Here, we employ $n_r = n_s$, where n_r is determined such that the retained singular values associated with the truncated residual principal components contain 99% of the total statistical energy (see, e.g., Ref. [9, Appendix A]).

4.1.3. Evaluation metrics

To evaluate the performance of the deterministic regression-function models, we employ the fraction of variance unexplained (FVU) computed on the test set $\mathcal{T}_{\text{test}}$, which is given by

$$\text{FVU} := 1 - r^2,$$

where the coefficient of determination is defined as

$$r^2 := 1 - \frac{\sum_{n\in\mathbb{T}_{\text{test}},\boldsymbol{\mu}\in\mathcal{D}_{\text{test}}} (\hat{\delta}_f^n(\boldsymbol{\mu}) - \delta^n(\boldsymbol{\mu}))^2}{\sum_{n\in\mathbb{T}_{\text{test}},\boldsymbol{\mu}\in\mathcal{D}_{\text{test}}} \left(\delta^n(\boldsymbol{\mu}) - \overline{\delta}\right)^2},$$

where the mean response is $\overline{\delta} := 1/|\mathcal{T}_{\text{test}}| \sum_{n\in\mathbb{T}_{\text{test}},\boldsymbol{\mu}\in\mathcal{D}_{\text{test}}} \delta^n(\boldsymbol{\mu})$.

4.2. Example 1: Advection–diffusion equation with projection-based reduced-order model

The first example considers the advection–diffusion equation and approximate solutions generated by a POD–Galerkin reduced-order model. The governing PDE is

$$\frac{\partial}{\partial t}u(\mathsf{x},t) + \mu_1\frac{\partial}{\partial \mathsf{x}}u(\mathsf{x},t) = \mu_2\frac{\partial^2}{\partial \mathsf{x}^2}u(\mathsf{x},t), \quad u(0,t) = u(2,t) = 0, \quad u(\mathsf{x},0) = \mathsf{x}(2-\mathsf{x})\exp(2\mathsf{x}), \tag{53}$$

on the domain $\mathsf{x} \in \Omega = [0, 2]$ with $t \in [0, T]$ and $T = 0.3$ and $u : \Omega \times [0, T] \to \mathbb{R}$. We employ a parameter domain of $(\mu_1, \mu_2) \in \mathcal{D} = [-2, -0.1] \times [0.1, 1]$, where the parameter μ_1 is the wave speed and the parameter μ_2 is the diffusion coefficient.

To derive a parameterized dynamical system of the form (1) for the FOM, we apply a finite-difference spatial discretization to the governing PDE (53), which uses central differencing for the diffusion term and upwind differencing for the advection term. This discretization is obtained by partitioning the spatial domain into 101 cells of equal width, yielding a FOM with $N = 100$ degrees of freedom of the form

$$\frac{dx_k}{dt} + \mu_1 \frac{x_{k+1} - x_k}{\Delta \mathsf{x}} = \mu_2 \frac{x_{k+1} - 2x_k + x_{k-1}}{\Delta \mathsf{x}^2}, \quad k = 1, \ldots, N,$$

where the state vector is $\boldsymbol{x}(t, \boldsymbol{\mu}) \equiv [x_1(t, \boldsymbol{\mu}) \ \cdots \ x_N(t, \boldsymbol{\mu})]^T$ and $x_k(t, \boldsymbol{\mu})$ is the finite-difference approximation to $u(\mathsf{x}_k, t, \boldsymbol{\mu})$ with $\mathsf{x}_k := 2k/(N+1)$ the kth point in the finite-difference discretization. The initial condition is characterized by $\boldsymbol{x}_0(\boldsymbol{\mu}) \equiv \boldsymbol{x}_0 = [\mathsf{x}_1(2 - \mathsf{x}_1)\exp(2\mathsf{x}_1) \ \cdots \ \mathsf{x}_N(2 - \mathsf{x}_N)\exp(2\mathsf{x}_N)]^T$.

To define the FOM OΔE of the form (2), we employ a time step $\Delta t = 3 \times 10^{-4}$ with the implicit Crank–Nicolson scheme, which is characterized by multistep coefficients $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = \beta_1 = 1/2$ in (3), which yields $N_t = 10^3$ time instances.

To construct approximate solutions, we employ projection-based reduced-order models as described in Section 2.1.3. For this purpose, we construct the trial-basis matrix $\boldsymbol{\Phi} \in \mathbb{R}_\star^{N \times K}$ using proper orthogonal decomposition (POD) by executing Algorithm 1 in Appendix B with inputs $\mathcal{D}_{\text{ROM}} = \{0.1, 1.05, 2.0\} \times \{0.1, 0.55, 1.0\}$, $N_{\text{skip}} = 10$, reference state $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \boldsymbol{x}_0(\boldsymbol{\mu})$, and $K = 5$. For the ROM, we employ Galerkin projection such that the ROM ODE corresponds to (12) with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$. We employ the same time discretization for the ROM as was employed for the FOM, i.e., the Crank–Nicolson scheme with time step $\Delta t = 3 \times 10^{-4}$.

We model both the QoI error $\delta_s^n(\boldsymbol{\mu})$, where the QoI corresponds to the solution at the midpoint of the domain such that the associated functional is $g(\boldsymbol{x}; t, \boldsymbol{\mu}) \mapsto \boldsymbol{e}_{51}^T \boldsymbol{x}$ with $\boldsymbol{e}_i$ the ith canonical unit vector, and the normed state error $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$.

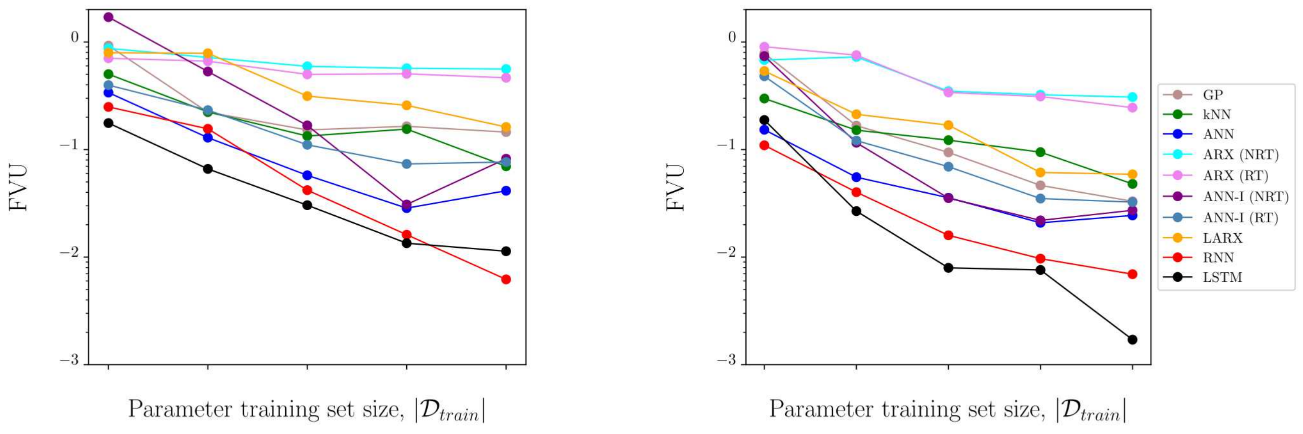
4.2.1. Coarse time grid, training, validation, and test sets

We now describe how we execute Steps 1–2 and 7 described in Section 4.1.1. For Step 1, we set $\mathbb{T} = \{20n\}_{n=1}^{50}$; the coarse grid is selected such that the error responses on the training set are well represented. For Step 2, we employ set sizes of $|\mathcal{D}_{\text{train}}| = 40$, $|\mathcal{D}_{\text{val}}| = 10$, and $|\mathcal{D}_{\text{test}}| = 50$. To assess the impact of the training set size on the performance of the deterministic regression-function model, we consider four smaller training and validation sets with $(|\mathcal{D}_{\text{train}}|, |\mathcal{D}_{\text{val}}|) \in \{(8,2), (16,4), (24,6), (32,8)\}$, which are constructed by sampling the original sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. In Step 7, we set $|\mathcal{D}_{\text{train},\epsilon}| = 20$.

4.2.2. Regression results

Figures 4–8 summarize the performance of the considered error models. First, Figure 4 reports the dependence of the performance of each deterministic regression-function model (with its best performing feature-engineering method) on the size of the training set $|\mathcal{D}_{\text{train}}|$. This figure illustrates several trends. First, we observe that traditional time-series-modeling methods, i.e., ARX and LARX, nearly always yield the worst performance. Here, ARX yields the worst performance, likely due to the fact that it is a low-capacity model and contains only a single latent variable. LARX outperforms ARX; this is likely due to its inclusion of additional latent variables. We observe that the performance of ANN-I is slightly better than ARX or LARX, but not as good as the (non-integrated) ANN. This indicates that, in this case, differencing the data provides no particular benefit. We also observe that recursive-neural-network models, i.e., RNN and LSTM, yield the best performance nearly uniformly, and generate FVU values at least one order of magnitude smaller than both traditional time-series-modeling methods. This highlights that there is substantial benefit to modeling nonlinear latent dynamics as is done with RNN and LSTM models. Of these two, LSTM tends to yield best performance in most cases. Finally, LSTM and RNN outperform the GP regression-function model. This is likely due to the fact that the time-local GP method employs only the parameters $\boldsymbol{\mu}$ as features, which are low quality compared to the residual-based quantities employed by the T-MLEM methods.

Figure 5 shows a histogram that, for regression methods in Category 2, compares the ratio of the fraction of variance unexplained using RT for training to fraction of variance unexplained using NRT for training. This figure shows that RT training generally outperforms NRT training; this is likely due to the fact that RT training properly accounts for the latent dynamics while training the model. For ARX, regression functions trained through RT outperform their NRT counterparts 90.8% of the time; for ANN-I the regression functions trained through RT outperform their NRT counterparts 72.3% of the time.



(a) Normed state error $\delta_{\boldsymbol{x}}$ **(b)** QoI error δ_s

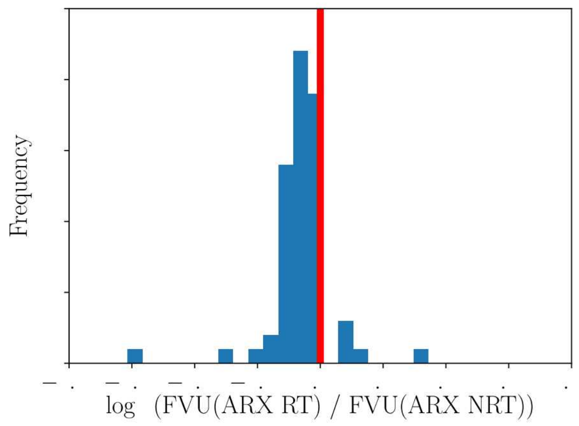
Figure 4: *Advection–diffusion equation.* Fraction of variance unexplained for all considered deterministic regression-function models, with each employing their best respective feature-engineering method.

Next, Table 4 summarizes the percentage of cases that a given deterministic regression-function model produces results with the lowest FVU; each case is defined by one of the 13 feature-engineering methods and one of the 5 training-set sizes, yielding 65 total cases. We do not include the GP method in these results, as it employs only the parameters as features. Results are presented for the cases where (1) all feature methods are considered and (2) only feature methods that don't include time are considered. The table shows that LSTM is the best overall performing regression-function model, leading to the lowest FVUs approximately 60% of the time. RNN is the next best performing regression-function model, followed by ANN. For the normed state error $\delta_{\boldsymbol{x}}$, recursive regression methods comprise the best performing regression-function models in 87.7% of cases when all feature-engineering methods are considered, and in 97.1% of cases when only feature-engineering engineering methods that omit time are considered; this shows that including time can improve the relative performance of non-recursive regression methods. For the QoI error δ_s, recurrent architectures comprise the best performing methods in over 97% of cases whether or not time is considered as a feature.

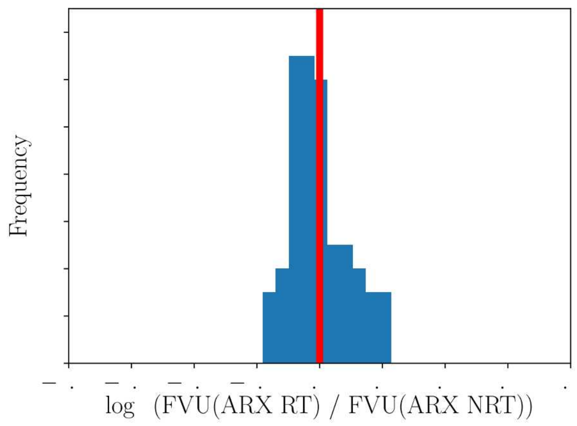
error	include feat. eng. w/ time?	kNN	ANN	ARX (NRT)	ARX (RT)	ANN-I (NRT)	ANN-I (RT)	LARX	RNN	LSTM	Recursive total
$\delta_{\boldsymbol{x}}$	Yes	0.0	12.3	0.0	0.0	1.5	0.0	0.0	27.7	58.5	87.7
$\delta_{\boldsymbol{x}}$	No	0.0	2.9	0.0	0.0	2.9	0.0	0.0	31.4	60.0	97.1
δ_s	Yes	0.0	1.5	0.0	0.0	0.0	1.5	1.5	32.3	63.1	98.5
δ_s	No	0.0	2.9	0.0	0.0	0.0	2.9	2.9	40.0	51.4	97.1

Table 4: *Advection–diffusion equation.* Percentage of cases having the lowest predicted FVU for each regression method. Results are summarized over all values $|\mathcal{D}_{\text{train}}| \in \{8, 16, 24, 32, 40\}$ and all feature-engineering methods. Recursive total reports the sum of all regression methods in Category 2 and Category 3.

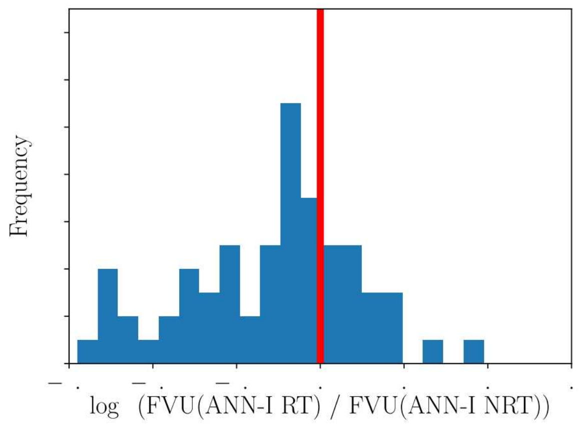
Figure 6 summarizes the performance of each combination of regression method and feature-engineering method for the case $|\mathcal{D}_{\text{train}}| = 40$. These results clearly show two critical trends. First, as hypothesized, using residual-based features can improve performance significantly; this is why they were considered high-quality features. This trend was also observed in Ref. [15]. Further, we note that—in this case—excellent results were obtained by computing only $7(\ll N)$ samples of the residual. Second, as hypothesized, the recurrent regression methods yield the best performance (even for low-quality features), with LSTM performing best



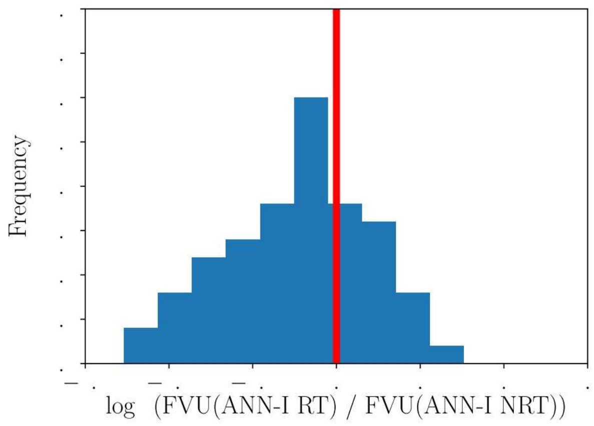
(a) Normed state error $\delta_{\boldsymbol{x}}$



(b) QoI error δ_s

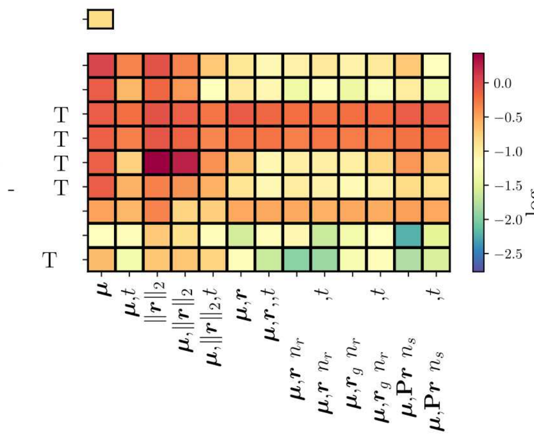


(c) Normed state error $\delta_{\boldsymbol{x}}$

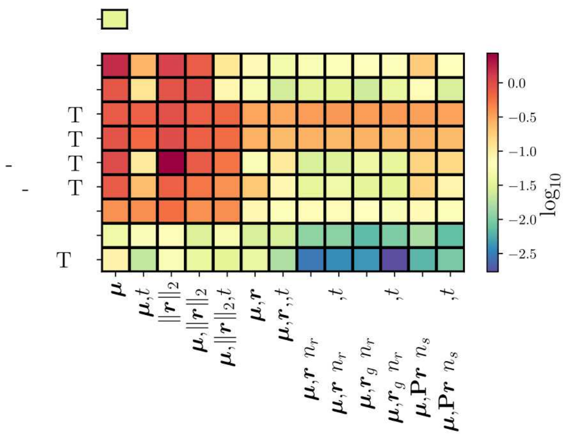


(d) QoI error δ_s

Figure 5: *Advection–diffusion equation.* Histogram of the ratio of the fraction of variance unexplained using RT for training to fraction of variance unexplained using NRT for training. Results are shown for ARX (top) and ANN-I (bottom). For ARX, RT outperforms NRT 90.8% of the time. For ANN-I, RT outperforms NRT 72.3% of the time.



(a) Normed state error $\delta_{\boldsymbol{x}}$



(b) QoI error δ_s

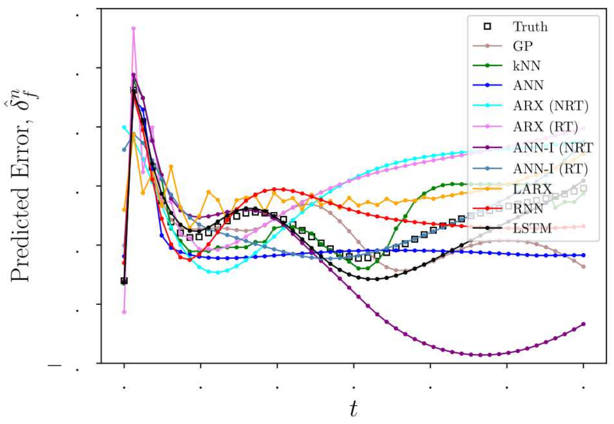
Figure 6: *Advection–diffusion equation.* Summary of the performance of each combination of regression method and feature-engineering method for the case $|\mathcal{D}_{\text{train}}| = 40$.

of all. In fact, LSTM and RNN outperform the time-local GP by approximately one order of magnitude. We also observe that including time slightly improves performance for these methods. Finally, we observe that for the deterministic regression-function models in Category 2, both training methods yield comparable results.

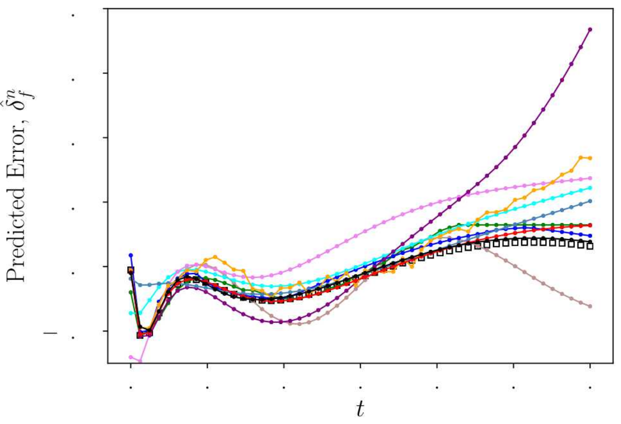
Figure 7 shows the predictions made by the different considered regression methods (for their best performing respective feature-engineering method) as a function of time for a randomly drawn element of $\mathcal{D}_{\text{test}}$ for cases $|\mathcal{D}_{\text{train}}| = 8$ and $|\mathcal{D}_{\text{train}}| = 40$. First, we note that the case $|\mathcal{D}_{\text{train}}| = 8$ yields significant errors for all considered models, with LSTM yielding the best performance. For the case $|\mathcal{D}_{\text{train}}| = 40$, all regression methods display substantially improved performance; LSTM, RNN, and ANN yield particularly accurate results.

Figure 8 shows the regression errors (i.e., the response error of the deterministic regression-function models) for the best performing regression method (i.e., LSTM) employing its best performing feature-engineering method. The figure compares the regression-error distribution with the distributions predicted by the Gaussian, Laplacian, and AR1 stochastic noise models. Table 5 reports validation frequencies of the approximated distributions, where $\omega(C)$ corresponds to the fraction of test data that lie within the C-prediction interval generated by the stochastic noise model. We also report the Kolmogorov–Smirnov (K-S) statistic. We observe that the Laplacian noise model yields the most accurate prediction-interval frequencies, with AR1 yielding similar performance to that of the Laplacian model.

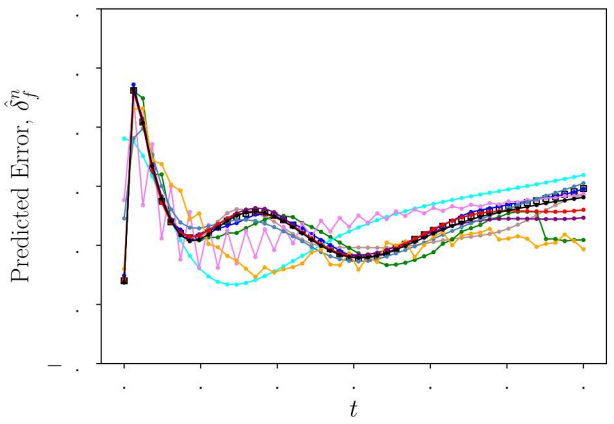
In summary, for the advection–diffusion example, the LSTM model yielded best performance, followed by RNN and ANN. We observed the residual-based features to yield best performance as well. Finally, the Laplacian model yielded the most accurate stochastic noise model.



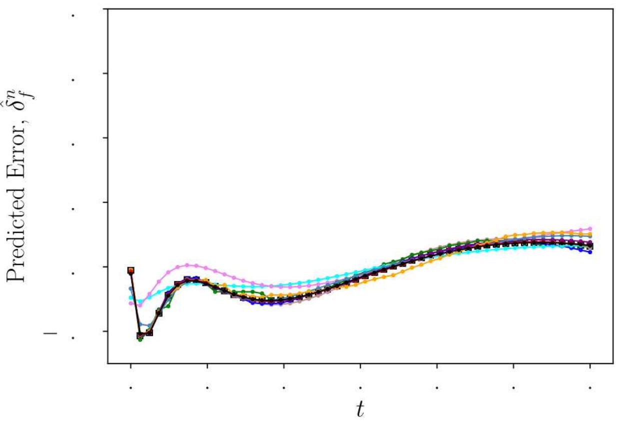
(a) Normed state error $\delta_{\boldsymbol{x}}$, $|\mathcal{D}_{\text{train}}| = 8$



(b) QoI error δ_s, $|\mathcal{D}_{\text{train}}| = 8$

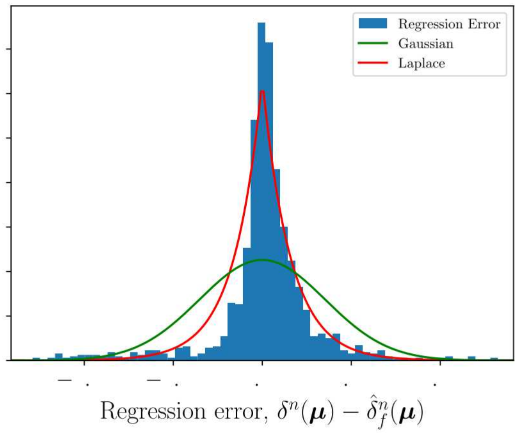


(c) Normed state error $\delta_{\boldsymbol{x}}$, $|\mathcal{D}_{\text{train}}| = 40$

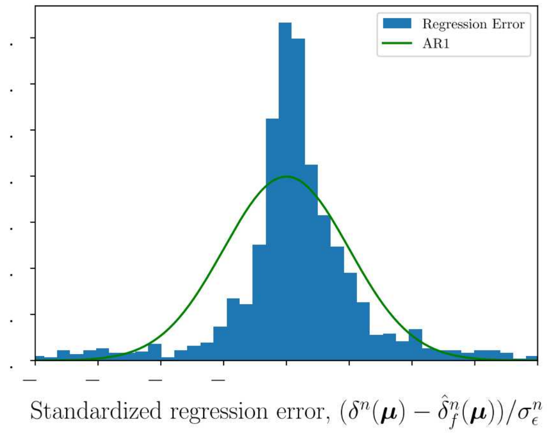


(d) QoI error δ_s, $|\mathcal{D}_{\text{train}}| = 40$

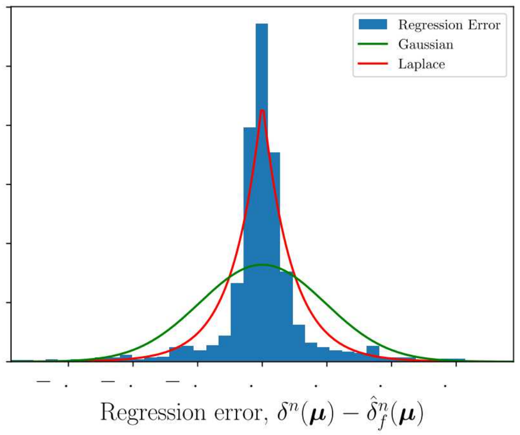
Figure 7: *Advection–diffusion equation.* Error response $\delta_{\boldsymbol{x}}$ (left) and δ_s (right) predicted by each regression method as a function of time. Results are shown for the best performing feature engineering method on the $|\mathcal{D}_{\text{train}}| = 8$ case (top) and the $|\mathcal{D}_{\text{train}}| = 40$ case (bottom).



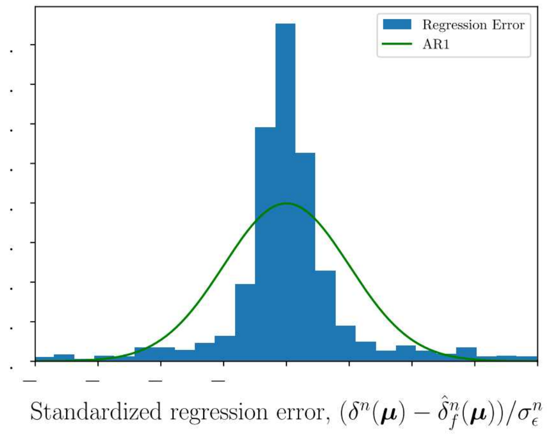
(a) Normed State Error



(b) Standardized Normed State Error



(c) QoI Error



(d) Standardized QoI Error

Figure 8: *Advection–diffusion equation.* Histogram of regression errors on the noise test set, $\mathcal{T}_{\text{test},\epsilon}$, for the normed state error (top) and QoI error (bottom). On the left, we show absolute errors $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. On the right, we show standardized errors, $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))/\sigma_\epsilon^n$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. The true regression errors are compared to the distributions predicted by the Gaussian, Laplacian, and AR1 error models. Regression errors are shown for the LSTM deterministic regression function model with the best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case.

	Gaussian		Laplacian		AR1	
Prediction interval	$\delta_{\boldsymbol{x}}$	δ_s	$\delta_{\boldsymbol{x}}$	δ_s	$\delta_{\boldsymbol{x}}$	δ_s
$\omega(0.68)$	0.842	0.805	0.724	0.673	0.714	0.725
$\omega(0.95)$	0.924	0.932	0.883	0.913	0.834	0.880
$\omega(0.99)$	0.943	0.967	0.868	0.967	0.868	0.913
K-S Statistic	0.216	0.216	0.114	0.088	0.146	0.131

Table 5: *Advection–diffusion equation.* Prediction intervals for the various stochastic noise models. Results correspond to those generated by the LSTM regression function model with its best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case, with samples collected over $\mathcal{T}_{\text{test},\epsilon}$.

4.3. Example 2: Shallow-water equations with projection-based reduced-order model

The second example considers the shallow water equations and approximate solutions generated by a POD–Galerkin reduced-order model. The governing system of PDEs is

$$\begin{aligned}
&\frac{\partial h}{\partial t} + \frac{\partial}{\partial \mathsf{x}}(hu) + \frac{\partial}{\partial \mathsf{y}}(hv) = 0\\
&\frac{\partial hu}{\partial t} + \frac{\partial}{\partial \mathsf{x}}(hu^2 + \frac{1}{2}\mu_1 h^2) + \frac{\partial}{\partial \mathsf{y}}(huv) = 0\\
&\frac{\partial hv}{\partial t} + \frac{\partial}{\partial \mathsf{x}}(huv) + \frac{\partial}{\partial \mathsf{y}}(hv^2 + \frac{1}{2}\mu_1 h^2) = 0
\end{aligned}$$

on the domain $(\mathsf{x}, \mathsf{y}) \in \Omega = [0,5] \times [0,5]$, $t \in [0,T]$ with $T = 10$, and where $h, u, v : \Omega \times [0,T] \to \mathbb{R}$, with h denoting the height of the water surface, u denoting the x-velocity, and v denoting the y-velocity. The parameter μ_1 is the "gravity" parameter.

The parameterized initial conditions are given by

$$h(\mathsf{x},\mathsf{y},0) = 1 + \mu_2 e^{(\mathsf{x}-1)^2+(\mathsf{y}-1)^2}, \quad u(\mathsf{x},\mathsf{y},0) = v(\mathsf{x},\mathsf{y},0) = 0, \quad (\mathsf{x},\mathsf{y}) \in \Omega,$$

which corresponds to a Gaussian pulse centered at $(1,1)$ with a magnitude of μ_2. We consider a parameter domain $(\mu_1, \mu_2) \in \mathcal{D} = [2,9] \times [0.05, 2]$.

To derive a parameterized dynamical system of the form (1) for the FOM, we apply a third-order discontinuous-Galerkin spatial discretization, which partitions the domain into a 16×16 grid of uniform square elements and employs tensor-product polynomials of order 2 to represent the solution over each element, yielding a FOM of dimension $N = 1.23 \times 10^4$. We employ slip-wall boundary conditions at the edge of the domain.

To define the FOM OΔE of the form (2), we employ a time step $\Delta t = 1.25 \times 10^{-2}$ with the implicit Crank–Nicolson scheme, which is characterized by multistep coefficients $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = \beta_1 = 1/2$ in (3), which yields $N_t = 800$ time instances. Figure 9 shows elevation surfaces obtained from the solution to the FOM OΔE at one randomly chosen parameter instance.

As in the first example, we employ projection-based reduced-order models to construct approximate solutions. We again construct a POD trial-basis matrix $\boldsymbol{\Phi} \in \mathbb{R}_\star^{N \times K}$ by executing Algorithm 1 in Appendix B with inputs $\mathcal{D}_{\text{ROM}} = \{3,6,9\} \times \{0.05, 0.1, 0.2\}$, $N_{\text{skip}} = 1$, reference state $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) = \mathbf{0}$, and $K = 78$. For the ROM, we employ Galerkin projection such that the ROM ODE corresponds to (12) with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$. We employ the same time discretization for the ROM as was employed for the FOM.

We model both the QoI error $\delta_s^n(\boldsymbol{\mu})$, where the QoI corresponds to the water-surface height in the middle of the domain, and the normed state error $\delta_{\boldsymbol{x}}^n(\boldsymbol{\mu})$.

4.3.1. Coarse time grid, training, validation, and test sets

We now describe how we execute Steps 1–2 and 7 introduced Section 4.1.1. For Step 1, we set $\mathbb{T} = \{4n\}_{n=1}^{200}$; this coarse grid is again selected such that the error responses on the training set are well represented. For Step 2, we employ set sizes of $|\mathcal{D}_{\text{train}}| = 40$, $|\mathcal{D}_{\text{val}}| = 10$, and $|\mathcal{D}_{\text{test}}| = 20$. Again, to assess the

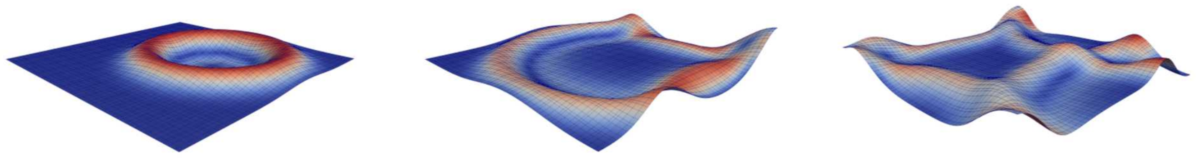

(a) $t = 1$ **(b)** $t = 2$ **(c)** $t = 3$

Figure 9: *Shallow-water equations.* Elevation surfaces of h for a randomly chosen parameter instance. Solutions are colored by velocity magnitude.

impact of the training set size on the performance of the deterministic regression-function model, we consider four smaller training and validation sets with $(|\mathcal{D}_{\text{train}}|, |\mathcal{D}_{\text{val}}|) \in \{(8, 2), (16, 4), (24, 6), (32, 8)\}$, which are constructed by sampling the original sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. In Step 7, we set $|\mathcal{D}_{\text{train},\epsilon}| = 10$.

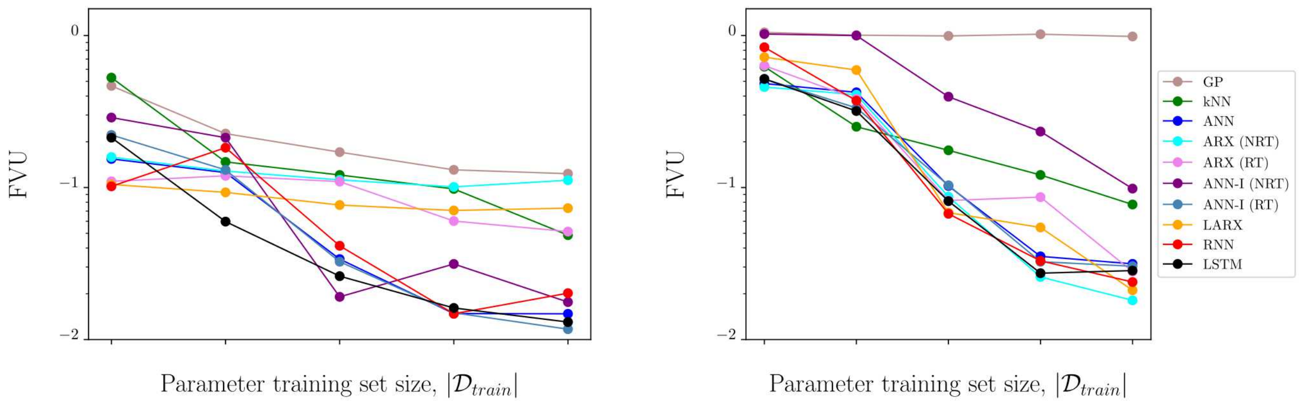
Lastly, we note that we do not consider any feature-engineering methods that employ Feature 4, as the dimension of the residual $N = 1.23 \times 10^4$ is prohibitively large in this example.

4.3.2. Regression results

Figures 10–13 summarize the performance of the deterministic regression-function models and stochastic noise models. First, Figure 10 reports the dependence of the performance of each deterministic regression-function model (with its best performing feature-engineering method) on the size of the training set $|\mathcal{D}_{\text{train}}|$. Figures 10a–10b considers all feature-engineering methods, while Figures 10c–10d limit consideration to feature-engineering methods that omit time from the set of features. When time is included as a feature, LSTM and RNN yield the best overall performance, followed closely by ANN; ANN slightly outperforms RNN in predicting the normed state error when time is included as a feature. ARX and LARX do not perform well for predicting the normed state error $\delta_{\boldsymbol{x}}$, but do perform very well in predicting the QoI error δ_s. We additionally observe that both methods for training ARX yield similar trends. ARX (RT) outperforms ARX (NRT) in predicting the normed state error, and ARX (NRT) outperforms ARX (RT) in predicting the QoI error. Next, we again observe that ANN-I performs worse than the standard ANN. We also observe that, in general, ANN-I (RT) performs slightly better than ANN-I (NRT) and yields similar results to ANN. Next, in comparing Figures 10a and 10b to Figures 10c and 10d, it is observed that the non-recursive regression methods are more reliant upon time to yield good performance than the recursive regression methods. This indicates that the non-recursive methods will likely not generalize well for problems characterized by phase shifts or for prediction beyond the training time interval. Lastly, the time-local GP benchmark is the worst performing method, and is particularly poor at predicting the QoI errors. This is thought to be due to the fact that the parameters are low quality features.

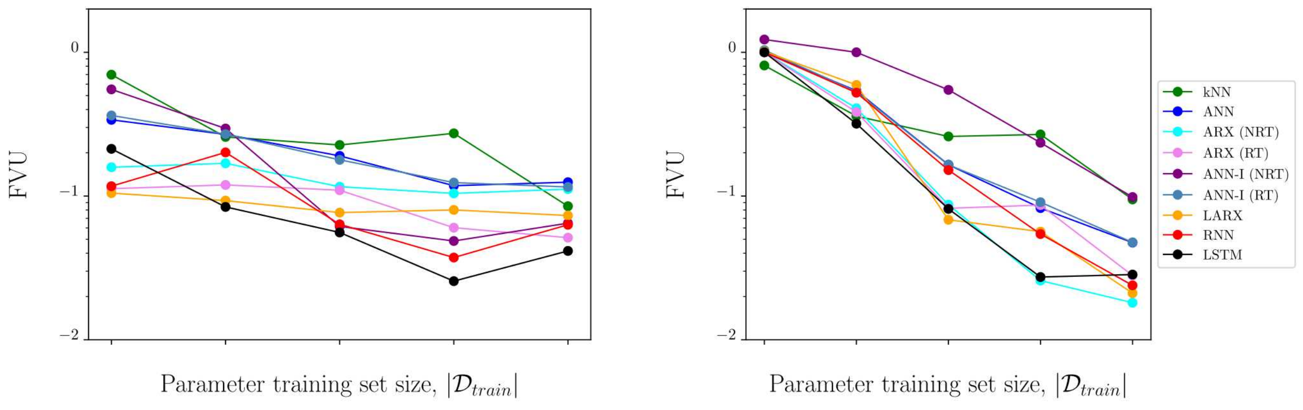
Table 6 summarizes the percentage of cases that a deterministic regression-function model produces results with the lowest FVU, where each case is defined by one of the 11 feature-engineering methods (recall that the residual, i.e., Feature 4 is not considered in this example) and one of the 5 training-set sizes, yielding 55 total cases. We again present results for the cases where (1) all feature methods are considered and (2) only feature methods that don't include time are considered. Again, the results indicate that LSTM yields the best overall performance, followed by LARX and RNN. Recursive methods yield the best performance for predicting the normed state error in 81.8% of cases when all feature engineering methods are considered, and in 86.7% of cases when only feature engineering methods that do not include time are considered. For the QoI prediction, recursive methods yield the best performance in 85.5% and 86.7% of cases, respectively. As before, this shows that including time as a feature can (slightly) improve the relative performance of non-recursive regression methods. Also as observed previously, including time as an input feature is observed to be more beneficial in predicting the normed state error than the QoI error.

Figure 11 summarizes the performance of each combination of regression method and feature-engineering method for the cases $|\mathcal{D}_{\text{train}}| = 8$ (top) and $|\mathcal{D}_{\text{train}}| = 40$ (bottom). For the case $|\mathcal{D}_{\text{train}}| = 40$, the recursive



(a) Normed state error $\delta_{\boldsymbol{x}}$ (All feature-engineering methods)

(b) QoI error δ_s (All feature-engineering methods)



(c) Normed state error $\delta_{\boldsymbol{x}}$ (Feature-engineering methods without time)

(d) QoI error δ_s (Feature-engineering methods without time)

Figure 10: *Shallow-water equations.* Fraction of variance unexplained for all considered deterministic regression-function models, with each employing their best respective feature-engineering method.

error	include feat. eng. w/ time?	kNN	ANN	ARX (NRT)	ARX (RT)	ANN-I (NRT)	ANN-I (RT)	LARX	RNN	LSTM	Recursive total
$\delta_{\boldsymbol{x}}$	Yes	9.1	9.1	0.0	5.5	9.1	9.1	5.5	7.3	45.5	81.8
$\delta_{\boldsymbol{x}}$	No	13.3	0.0	0.0	10.0	6.7	0.0	10.0	3.3	56.7	86.7
δ_s	Yes	14.5	0.0	12.7	5.5	5.5	7.3	20.0	16.4	18.2	85.5
δ_s	No	13.3	0.0	16.7	3.3	3.3	3.3	16.7	13.3	30.0	86.7

Table 6: *Shallow-water equations.* Percentage of cases having the lowest predicted FVU for each regression method. Results are summarized over all values $|\mathcal{D}_{\text{train}}| \in \{10, 20, 30, 40, 50\}$ and all feature-engineering methods. Recursive total reports the sum of all regression methods in Category 2 and Category 3.

regression methods generally display better overall performance than the non-recursive regression methods in predicting the normed state error $\delta_{\boldsymbol{x}}$. The low-dimensional feature engineering methods are additionally seen to lead to better overall performance than the high-dimensional feature engineering methods in this case, as using a small number of features typically leads to lower-capacity models that generalize with limited training data (see Consideration 3). For the QoI error, all regression methods perform poorly for the case $|\mathcal{D}_{\text{train}}| = 10$, with the lowest observed FVU value around FVU = 0.5. For the case $|\mathcal{D}_{\text{train}}| = 40$, all methods perform substantially better; here, it is clear that residual-based features are essential for obtaining good performance. LSTM, RNN, ANN, and ANN-I (RT) generally yield the best performance for the normed state error $\delta_{\boldsymbol{x}}$; however, ANN and ANN-I (RT) perform well only when time is included as a feature. For the QoI prediction, LSTM, RNN, LARX, and ARX (NRT) generally perform best.

Figure 12 shows the predictions made by the different considered regression methods (for their best performing respective feature-engineering method) as a function of time for a randomly drawn element of $\mathcal{D}_{\text{test}}$ for case $|\mathcal{D}_{\text{train}}| = 40$. The figure shows that all approaches except GP provide reasonably accurate predictions for both the normed state and QoI errors.

Figure 13 and Table 7 summarize the performance of the stochastic noise models, where results are shown for the LSTM regression method employing its best performing feature-engineering method for the case $|\mathcal{D}_{\text{train}}| = 40$. Similar to the previous case, the Laplacian noise model yields the best performance.

	Gaussian		Laplacian		AR1	
Prediction interval	$\delta_{\boldsymbol{x}}$	δ_s	$\delta_{\boldsymbol{x}}$	δ_s	$\delta_{\boldsymbol{x}}$	δ_s
$\omega(0.68)$	0.956	0.971	0.801	0.848	0.927	0.972
$\omega(0.95)$	0.991	0.995	0.966	0.960	0.984	0.990
$\omega(0.99)$	0.998	0.996	0.987	0.988	0.987	0.991
K-S Statistic	0.265	0.323	0.125	0.146	0.231	0.350

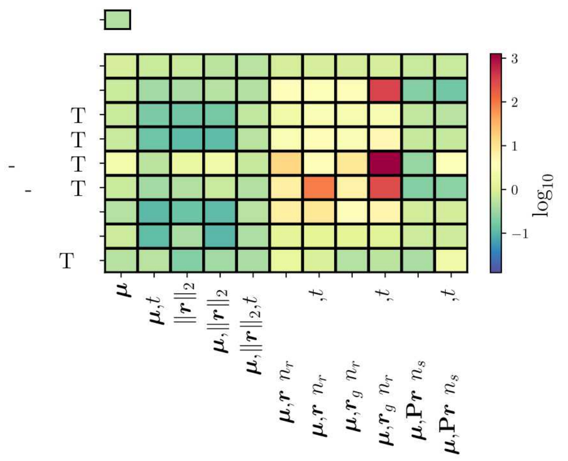
Table 7: *Shallow-water equations.* Prediction intervals for the various stochastic noise models. Results correspond to those generated by the best LSTM regression function model with its best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case, with samples collected over $\mathcal{T}_{\text{test},\epsilon}$.

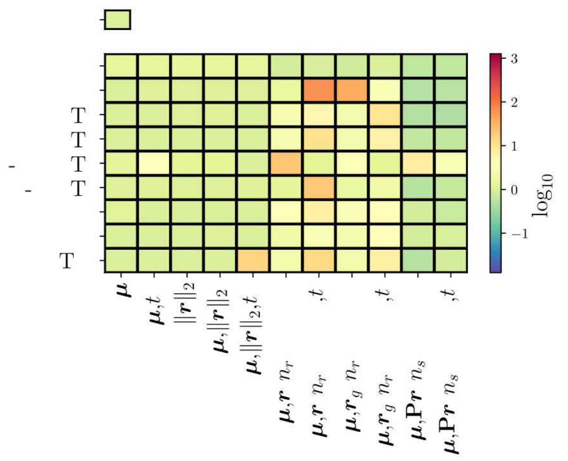
4.4. Example 3: Inviscid Burgers' equation with coarse-mesh low-fidelity model

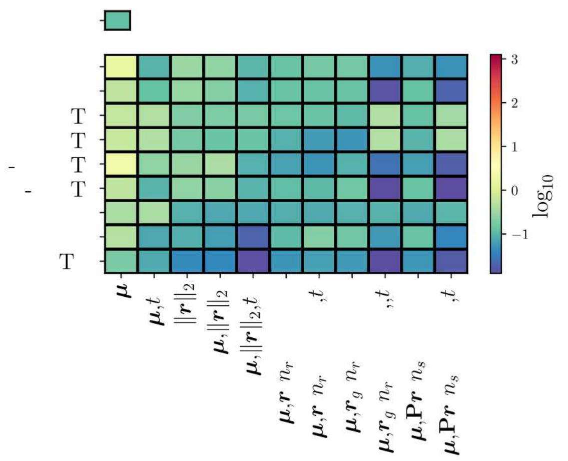
The final example considers the one-dimensional inviscid Burgers' equation and approximate solutions generated by a low-fidelity model provided by a coarse spatial discretization. The governing PDE is

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial \mathsf{x}} = \mu_1 e^{\mu_2 \mathsf{x}}, \quad u(0,t) = \mu_3, \quad u(\mathsf{x},0) = \mu_4 \text{ for } \mathsf{x} \in (0,100] \tag{54}$$

on the domain $\mathsf{x} \in \Omega = [0,100]$ with $t \in [0,T]$ and $T = 40$, and $u : \Omega \times [0,T] \to \mathbb{R}$. We employ a parameter domain of $(\mu_1, \mu_2, \mu_3, \mu_4) \in \mathcal{D} = [0.005, 0.05] \times [0.005, 0.05] \times [3, 5] \times [0.5, 2.5]$.


(a) Normed state error $\delta_{\boldsymbol{x}}, |\mathcal{D}_{\text{train}}| = 8$


(b) QoI error $\delta_s, |\mathcal{D}_{\text{train}}| = 8$


(c) Normed state error $\delta_{\boldsymbol{x}}, |\mathcal{D}_{\text{train}}| = 40$

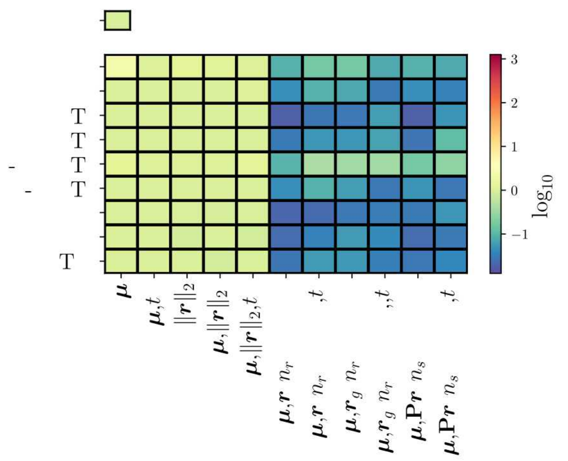
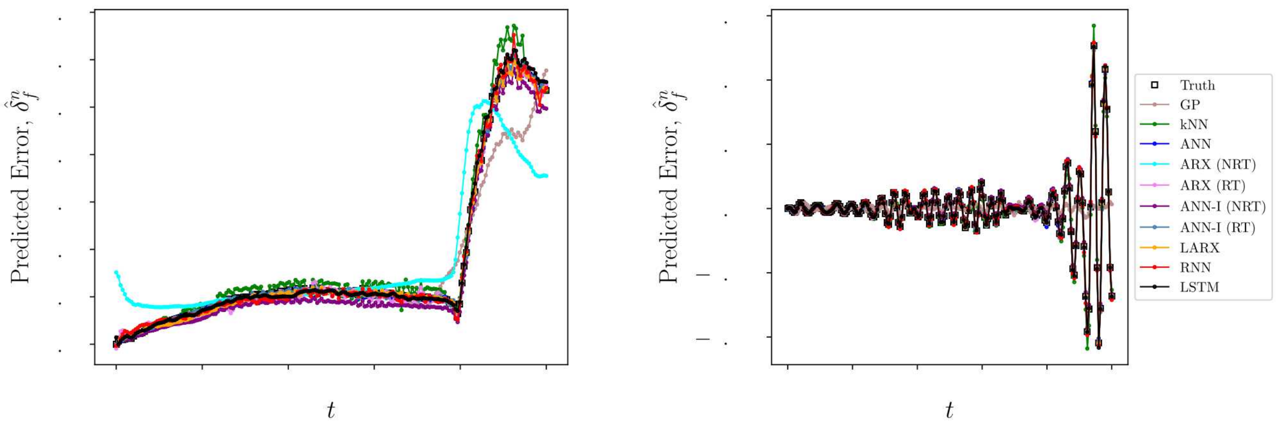

(d) QoI error $\delta_s, |\mathcal{D}_{\text{train}}| = 40$

Figure 11: *Shallow-water equations.* Summary of the performance of each combination of regression method and feature-engineering method for the cases $|\mathcal{D}_{\text{train}}| = 8$ (top) and $|\mathcal{D}_{\text{train}}| = 40$ (bottom).



(a) Normed state error $\delta_{\boldsymbol{x}}$ **(b)** QoI error δ_s

Figure 12: *Shallow-water equations.* Error response $\delta_{\boldsymbol{x}}$ (left) and δ_s (right) predicted by each regression method as a function of time. Results are shown for the best performing feature engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case.

To derive a parameterized dynamical system of the form (1) for the FOM, we apply a finite-volume spatial discretization to the governing equations (54), which uses an upwind flux at the cell interfaces and a uniform control-volume width of $\Delta\mathsf{x} = 0.1$, yielding a FOM with $N = 1000$ degrees of freedom.

To define the FOM OΔE of the form (2), we employ a time step $\Delta t = 0.05$ with the implicit Euler scheme, which is characterized by multistep coefficients $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = 1$ in (3), which yields $N_t = 800$ time instances.

To construct approximate solutions, we employ coarse-mesh low-fidelity models as described in Section 2.1.2. In particular, to define the LFM ODE of the form (7), we employ the same finite-volume scheme as that employed by the FOM, but use a uniform control-volume width of $\Delta\mathsf{x} = 2$ such that $N_{\text{LF}} = 50$. The prolongation operator $\mathbf{p}$ consists of a linear interpolation of the coarse-grid solution onto the fine grid.

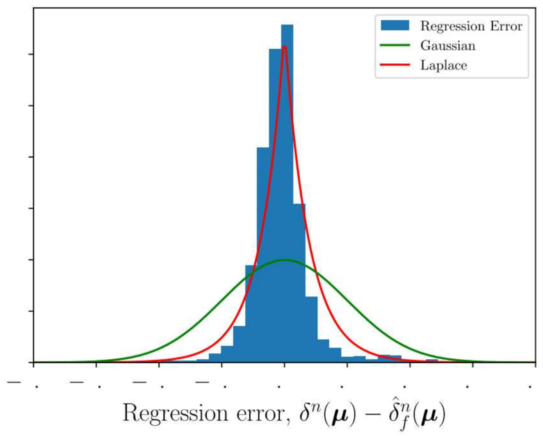
We model the QoI error $\delta_s^n(\boldsymbol{\mu})$, where the QoI corresponds to the solution at the midpoint of the domain such that the associated functional is $g(\boldsymbol{x}; t, \boldsymbol{\mu}) \mapsto \boldsymbol{e}_{501}^T \boldsymbol{x}$.

4.4.1. Coarse time grid, training, validation, and test sets

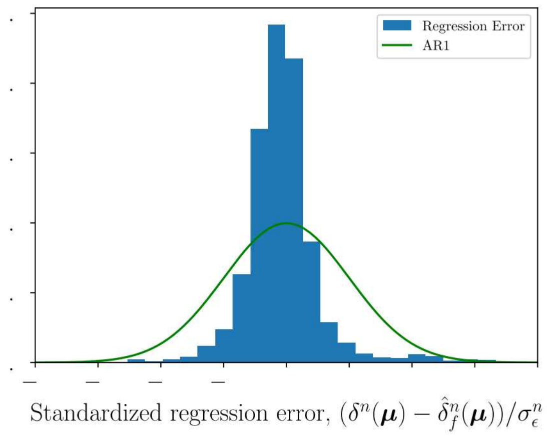
We now describe how we execute Steps 1–2 and 7 described in Section 4.1.1. For Step 1, we set $\mathbb{T} = \{8n\}_{n=1}^{100}$. For Step 2, we employ the same approach as in the previous section. Namely, we employ set sizes of $|\mathcal{D}_{\text{train}}| = 40$, $|\mathcal{D}_{\text{val}}| = 10$, and $|\mathcal{D}_{\text{test}}| = 50$. To assess the impact of the training set size on the performance of the deterministic regression-function model, we consider four smaller training and validation sets with $(|\mathcal{D}_{\text{train}}|, |\mathcal{D}_{\text{val}}|) \in \{(8, 2), (16, 4), (24, 6), (32, 8)\}$, which are constructed by sampling the original sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$. In Step 7, we set $|\mathcal{D}_{\text{train},\epsilon}| = 20$.

4.4.2. Regression results

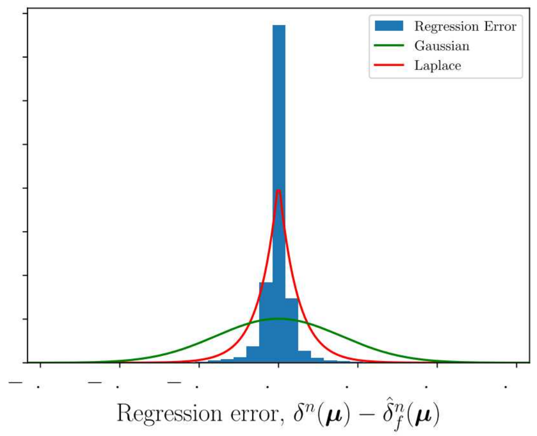
Figures 14 through 17 summarize the performance of the deterministic regression-function models and stochastic noise models. First, Figure 14 reports the dependence of the performance of each deterministic regression-function model (with its best performing feature-engineering method) on the size of the training set $|\mathcal{D}_{\text{train}}|$. Figure 14a considers all feature-engineering methods, while Figure 14b limits consideration to feature-engineering methods that omit time from the set of features. We observe that LSTM and RNN again yield the best performance, although their improvement over ANN is less significant than what was observed in the previous two examples. We observe that all other methods—including the time-local GP benchmark—perform substantially worse than these three techniques.



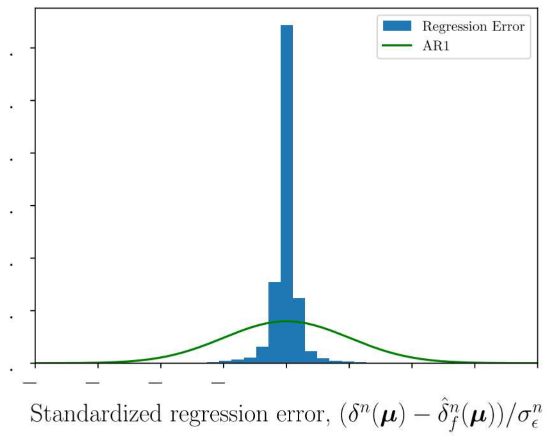
(a) Normed State Error



(b) Standardized Normed State Error

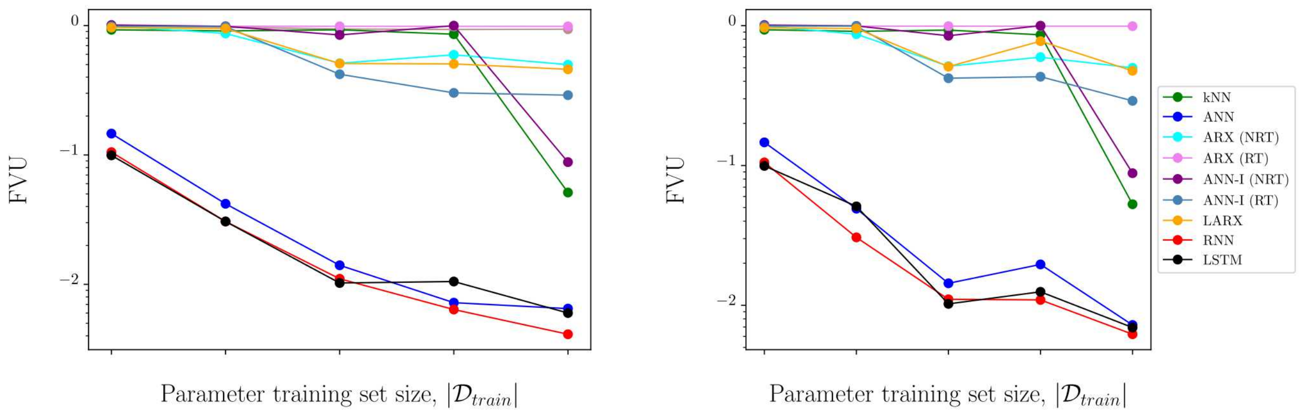


(c) QoI Error



(d) Standardized QoI Error

Figure 13: *Shallow-water equations.* Histogram of regression errors on the noise test set, $\mathcal{T}_{\text{test},\epsilon}$, for the normed state error (top) and QoI error (bottom). On the left, we show absolute errors $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. On the right, we show standardized errors, $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))/\sigma_\epsilon^n$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. The true regression errors are compared to the distributions predicted by the Gaussian, Laplacian, and AR1 error models. Regression errors are shown for the LSTM deterministic regression function model with the best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case.



(a) QoI error δ_s (All feature-engineering methods) **(b)** QoI error δ_s (Feature-engineering methods without time)

Figure 14: *Inviscid Burgers' equation.* Fraction of variance unexplained for all considered deterministic regression-function models, with each employing their best respective feature-engineering method.

Next, Table 8 summarizes the percentage of cases that a deterministic regression-function model produces results with the lowest FVU, where each case is defined by one of the 13 feature-engineering methods and one of the 5 training-set sizes, yielding 65 total cases. Results are presented for the cases where (1) all feature methods are considered and (2) only feature methods that don't include time are considered. The table shows that LSTM is the best overall performing regression-function model, leading to the lowest FVUs in over half of cases. RNN yields the next best performance, followed by ANN. We again observe that recursive regression methods yield the best performance in the vast majority of cases; we also again observe that including time can improve the relative performance of non-recursive regression methods.

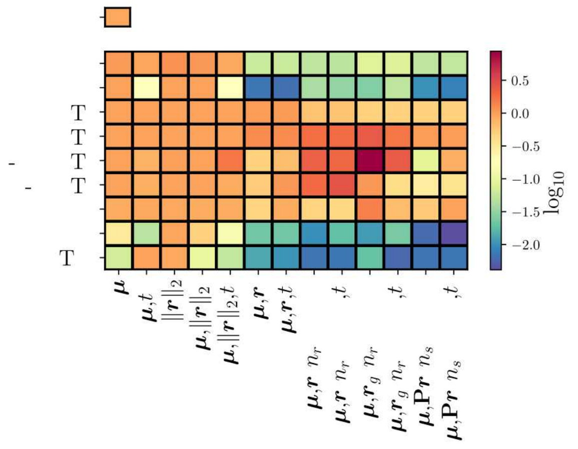
error	include feat. eng. w/ time?	kNN	ANN	ARX (NRT)	ARX (RT)	ANN-I (NRT)	ANN-I (RT)	LARX	RNN	LSTM	Recursive total
δ_s	Yes	0.0	12.3	0.0	0.0	0.0	0.0	1.5	35.4	50.8	87.7
δ_s	No	0.0	5.7	0.0	0.0	0.0	0.0	2.9	34.3	57.1	94.3

Table 8: *Inviscid Burgers' equation.* Percentage of cases having the lowest predicted FVU for each regression method. Results are summarized over all values $|\mathcal{D}_{\text{train}}| \in \{8, 16, 24, 32, 40\}$ and all feature-engineering methods. Recursive total reports the sum of all regression methods in Category 2 and Category 3.

Figure 15 summarizes the performance of each combination of regression method and feature-engineering method for the case $|\mathcal{D}_{\text{train}}| = 40$. We once again observe that feature-engineering methods employing elements of the residual lead to the best performance. We also observe that the LSTM and RNN methods yield the best performance, and substantially outperform the time-local GP method; indeed, they generate FVU values around 0.1 when they use only the parameters $\boldsymbol{\mu}$ as features. These methods also generate FVU values lower than 0.01 when using only 46 residual samples in the set of features.

Figure 16 shows the predictions made by the different considered regression methods (for their best performing respective feature-engineering method) as a function of time for a randomly drawn element of $\mathcal{D}_{\text{test}}$ for cases $|\mathcal{D}_{\text{train}}| = 8$ and $|\mathcal{D}_{\text{train}}| = 40$. Here, we observe that LSTM, RNN, and ANN provide the most accurate predictions; indeed, their predictions are already very accurate for only $|\mathcal{D}_{\text{train}}| = 8$. The time-local GP benchmark and ARX models perform the worst.

Lastly, Figure 17 and Table 9 summarize the performance of each stochastic noise model for the LSTM regression method employing its best performing feature-engineering method. Similar to the previous cases,



(a) QoI error δ_s

Figure 15: *Inviscid Burgers' equation.* Summary of the performance of each combination of regression method and feature-engineering method for the case $|\mathcal{D}_{\text{train}}| = 40$.

the Laplacian stochastic noise model provides the best stochastic noise model.

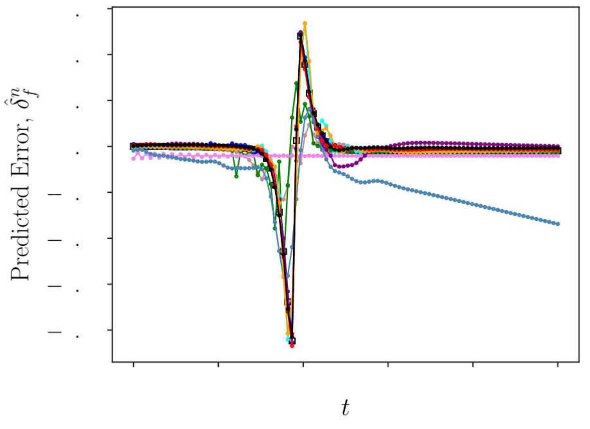
	Gaussian	Laplacian	AR1
Prediction interval	δ_s	δ_s	δ_s
$\omega(0.68)$	0.943	0.810	0.933
$\omega(0.95)$	0.970	0.946	0.959
$\omega(0.99)$	0.976	0.975	0.965
K-S Statistic	0.273	0.111	0.274

Table 9: *Inviscid Burgers' equation.* Prediction intervals for the various stochastic noise models. Results correspond to those generated by the best performing regression method (i.e., LSTM) with its best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case, with samples collected over $\mathcal{T}_{\text{test}}$.

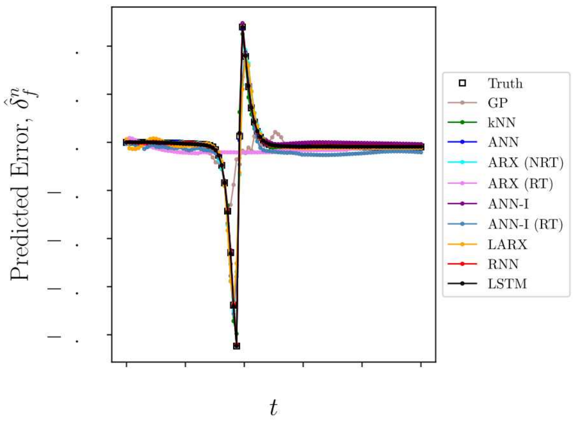
5. Conclusions

This work proposed the Time-Series Machine-Learning Error Modeling (T-MLEM) method for constructing error models for approximate solutions to parameterized dynamical systems. The T-MLEM method extends the machine learning error modeling (MLEM) [15] method to dynamical systems. The key contribution of this work as compared to previous works on error modeling for dynamical systems is the use of residual-based features and recursive regression methods (e.g., recurrent neural networks, long short-term memory networks) that employ latent variables. The use of these latent variables enables these regression models to model dependence on temporally non-local quantities, which is essential for modeling errors in this context (as illustrated by classical approaches for error quantification).

Similar to the MLEM method, the T-MLEM method entails four steps: feature engineering, data generation, training a deterministic regression-function model that maps from features to the conditional expectation of the error, and training a stochastic noise model. For feature engineering, we considered a variety of techniques based on those proposed in Ref. [15], along with several new techniques that include the time variable as a feature. The majority of these feature-engineering methods employ residual-based quantities inspired by classic error-quantification approaches. Next, we considered a hierarchy of regression methods

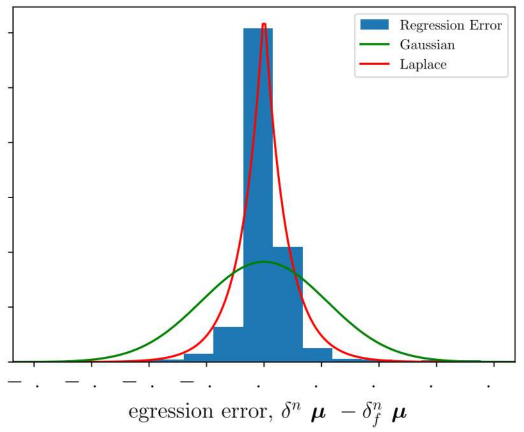


(a) $|\mathcal{D}_{\text{train}}| = 8$

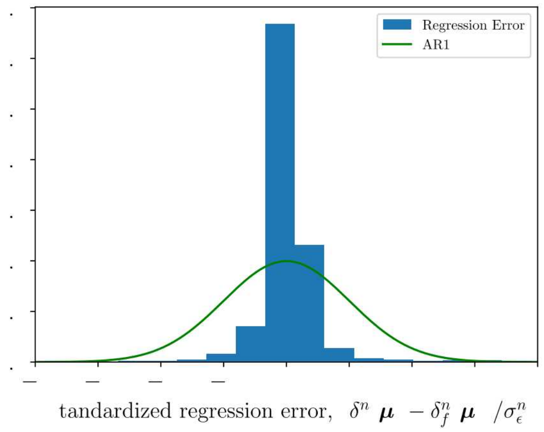


(b) $|\mathcal{D}_{\text{train}}| = 40$

Figure 16: *Inviscid Burgers' equation.* Error response δ_s predicted by each regression method as a function of time. Results are shown for the best performing feature engineering method on the $|\mathcal{D}_{\text{train}}| = 8$ case (left) and the $|\mathcal{D}_{\text{train}}| = 40$ case (right).



(a) QoI Error



(b) Standardized QoI Error

Figure 17: *Inviscid Burgers' equation.* Histogram of regression errors on the noise test set, $\mathcal{T}_{\text{test},\epsilon}$, for the QoI error. On the left, we show absolute errors $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. On the right, we show standardized errors, $(\delta^n(\boldsymbol{\mu}) - \hat{\delta}_f^n(\boldsymbol{\mu}))/\sigma_\epsilon^n$, $\boldsymbol{\mu} \in \mathcal{D}_{\text{test},\epsilon}$, $n \in \mathbb{T}$. The true regression errors are compared to the distributions predicted by the Gaussian, Laplacian, and AR1 error models. Regression errors are shown for the LSTM deterministic regression function model with the best performing feature-engineering method on the $|\mathcal{D}_{\text{train}}| = 40$ case.

that employ both non-recurrent and recurrent regression functions as well as linear and nonlinear latent-variable dynamics. These techniques include classic time-series modeling techniques (e.g., ARX), as well as time-series modeling techniques emerging from the deep-learning community (e.g., LSTM). Lastly, we examined three different stochastic noise models: Gaussian, Laplacian, and autoregressive Gaussian.

Across three benchmark problems and two different types of approximate solutions, the numerical experiments illustrated the best overall performance was obtained using the T-MLEM framework with an LSTM regression-function model. We attribute this favorable performance to LSTM's ability to capture dependencies on long-term non-local quantities through its use of a "cell state", as well as its amenability to training. While the best-performing feature-engineering method varied across the considered problems, generally feature-engineering methods that employ residual-based features yielded the best performance. Lastly, we observed the Laplacian stochastic noise model to provide the best statistical model for the prediction noise. In particular, we emphasize that the T-MLEM method with an LSTM regression-function model and residual-based features significantly outperformed both (1) the classical model-discrepancy method [25], which constructs an error model by applying Gaussian-process regression in parameter space, and (2) error models constructed using non-recursive regression models as was proposed in Ref. [45].

Future work involves the application of the T-MLEM framework on truly large-scale problems, as well as in more difficult scenarios, e.g., predicting errors beyond the training time interval.

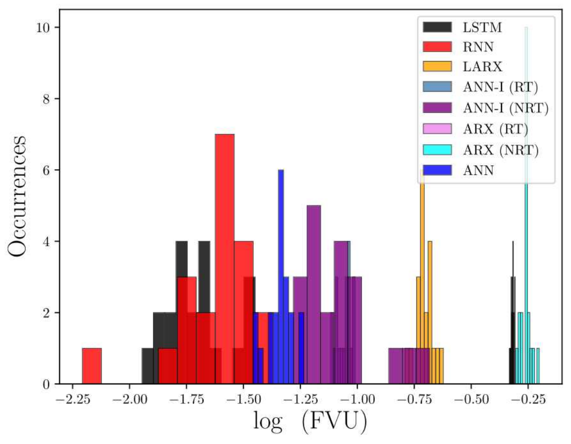
6. Acknowledgements

E. Parish acknowledges an appointment to the Sandia National Laboratories' John von Neumann Postdoctoral Research Fellowship in Computational Science. The authors thank Brian Freno for providing the data that was used in the third numerical experiment. This work was partially sponsored by Sandia's Advanced Simulation and Computing (ASC) Verification and Validation (V&V) Project/Task #103723/05.30.02. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

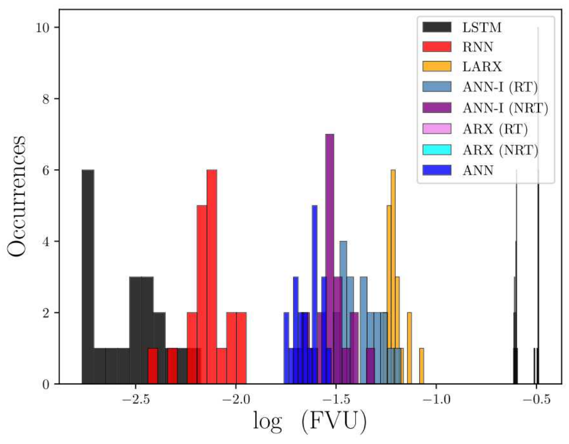
Appendix A Distributions of trained models

In Section 4.1.1, we outlined the training and model selection process used in the numerical experiments. In Step 5, we train the regression-function model for each value of hyperparameters 20 times using the Adam algorithm; the models are trained 20 times because the Adam algorithm is stochastic. In Step 6, we then select the model – which corresponds to one value of the hyperparameters and one execution of the Adam algorithm – by evaluating validation criterion (44) or (45), depending on the category of the method. This "best" model is presented in the numerical results.

The reader may be interested in the distribution of models obtained through the 20 training runs. Here, we display these results for the $|\mathcal{D}_{\text{train}}| = 40$ case in the first numerical experiment. Figure 18 presents a histogram of the fraction of variance unexplained for all twenty training runs of the regression-function models. Each count on the histogram in Figure 18 corresponds to a trained regression-function model employing its best respective feature-engineering method. We observe that Figure 18 displays the same trends as Figure 4. LSTM and RNN yield the best distribution of regression-function models, followed closely by ANN and ANN-I. One feature worth noting for the normed state error is that LSTM yields a more accurate *distribution* of regression-function models than RNN; RNN, however, yields one regression-function model that has a particularly low FVU. This provides insight into why RNN slightly outperforms LSTM in predicting the normed state error in Figure 4.



(a) Normed state error $\delta_{\boldsymbol{x}}$



(b) QoI error δ_s

Figure 18: *Advection–diffusion equation.* Histogram of fraction of variance unexplained for all twenty training runs of the regression-function models. Each entry corresponds to a trained regression-function model employing its best respective feature-engineering method.

Appendix B Proper orthogonal decomposition

Algorithm 1 provides the algorithm for computing the POD basis used in this work. We note that the basis dimension K can be determined from the decay of the singular values; for simplicity, we treat it as an algorithm input.

Appendix C Q-sampling

Feature engineering methods explored in this work that employ either the gappy principal components (Feature 6) or sampled residual (Feature 7) require the specification of the sampling matrix $\mathbf{P}$. In this work, we compute this sampling matrix via q-sampling [11]. Algorithm 2 provides the associated algorithm.

Algorithm 1 Proper orthogonal decomposition (POD)

Input: Training parameter instances $\mathcal{D}_{\text{ROM}} \equiv \{\boldsymbol{\mu}_{\text{ROM}}^{i}\} \subseteq \mathcal{D}$; number of time steps between collected snapshots $N_{\text{skip}} \leq N_t$; reference state $\boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu})$; desired basis dimension K.
Output: POD basis $\mathbf{\Phi} \equiv [\boldsymbol{\phi}_1 \ \cdots \ \boldsymbol{\phi}_K] \in \mathbb{R}_{\star}^{N\times K}$.
Steps:

1. Solve the FOM OΔE (2) for $\boldsymbol{\mu} \in \mathcal{D}_{\text{ROM}}$ and collect the snapshot matrix

$$\boldsymbol{S}(\boldsymbol{\mu}, N_{\text{skip}}) := \left[\boldsymbol{x}^{N_{\text{skip}}}(\boldsymbol{\mu}) - \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu}) \ \cdots \ \boldsymbol{x}^{\text{floor}(N_t/N_{\text{skip}})N_{\text{skip}}}(\boldsymbol{\mu}) - \boldsymbol{x}_{\text{ref}}(\boldsymbol{\mu})\right]. \tag{55}$$

2. Compute the (thin) singular value decomposition

$$\left[\boldsymbol{S}(\boldsymbol{\mu}_{\text{ROM}}^{1}, N_{\text{skip}}) \ \cdots \ \boldsymbol{S}(\boldsymbol{\mu}_{\text{ROM}}^{N_{\text{train,ROM}}}, N_{\text{skip}})\right] = \boldsymbol{U}\mathbf{\Sigma}\boldsymbol{V}^T, \tag{56}$$

   where $\boldsymbol{U} \equiv \left[\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_{N_{\text{train,ROM}}N_t/N_{\text{skip}}}\right]$.

3. Truncate the left singular vectors such that $\boldsymbol{\phi}_i = \boldsymbol{u}_i$, $i = 1, \ldots, K$.

Algorithm 2 Algorithm for generation of sampling matrix through q-sampling.

Input: Residual training set $\mathcal{T}_{\boldsymbol{r},\text{train}} \equiv \{\tilde{\boldsymbol{r}}^n(\boldsymbol{\mu}) \,|\, \boldsymbol{\mu} \in \mathcal{D}_{\text{train}} \equiv \{\boldsymbol{\mu}_i\}_{i=1}^{|\mathcal{D}_{\text{train}}|}, n \in \mathbb{T}_{\text{train}}\}$
Output: Sampling matrix $\mathbf{P} \in \{0,1\}^{n_s \times N}$
Steps:

1. Compute the residual snapshot matrix:

$$\boldsymbol{S}_{\boldsymbol{r}} := \left[\tilde{\boldsymbol{r}}^{\tau(1)}(\boldsymbol{\mu}_1) \quad \cdots \quad \tilde{\boldsymbol{r}}^{\tau(\bar{N}_t)}(\boldsymbol{\mu}_1) \quad \cdots \quad \tilde{\boldsymbol{r}}^{\tau(1)}(|\boldsymbol{\mu}_{\mathcal{D}_{\text{train}}}|) \quad \cdots \quad \tilde{\boldsymbol{r}}^{\tau(\bar{N}_t)}(\boldsymbol{\mu}_{|\mathcal{D}_{\text{train}}|})\right].$$

2. Compute the column mean of $\boldsymbol{S}_{\boldsymbol{r}}$

$$\bar{\tilde{\boldsymbol{r}}} := \frac{1}{|\mathcal{T}_{\boldsymbol{r},\text{train}}|} \sum_{\tilde{\boldsymbol{r}} \in \mathcal{T}_{\boldsymbol{r},\text{train}}} \tilde{\boldsymbol{r}}$$

3. Compute the (thin) singular value decomposition:

$$\boldsymbol{S}_{\boldsymbol{r}} - \bar{\tilde{\boldsymbol{r}}}\mathbf{e}^T = \boldsymbol{U}\mathbf{\Sigma}\boldsymbol{V}^T,$$

   where $\mathbf{e} \in \{1\}^{|\mathcal{T}_{\boldsymbol{r},\text{train}}|}$.

4. Compute the QR factorization of $\boldsymbol{U}^T$ with column pivoting

$$\boldsymbol{U}^T[\mathbf{P}^*]^T = \boldsymbol{Q}\boldsymbol{R},$$

   where $\mathbf{P}^* \equiv \left[\mathbf{p}_1 \quad \cdots \quad \mathbf{p}_{|\mathcal{T}_{\boldsymbol{r},\text{train}}|}\right]^T$, $\mathbf{p}_i \in \{0,1\}^N$.

5. From the permutation matrix $\mathbf{P}^*$, select the first $n_s \leq |\mathcal{T}_{\boldsymbol{r},\text{train}}|$ rows to form the sampling matrix

$$\mathbf{P} := \left[\mathbf{p}_1 \quad \cdots \quad \mathbf{p}_{n_s}\right]^T.$$

Appendix D Gaussian-process regression

The numerical experiments consider time-local Gaussian-process regression models as a benchmark error-modeling method. This method is analogous to employing the "model-discrepancy" method of Kennedy and O'Hagan [25] at each time instance. This approach constructs a separate GP model to predict the error for each time instnace of interest. As such, the approach can only be used to make predictions at time instances for which training data exist, i.e., prediction beyond the training time interval is not possible. We note that the use of time-local GP regression was considered in Ref. [35], wherein GP models were constructed in parameter space over time windows.

We begin by setting $\mathcal{D}_{\text{train}} \equiv \{\boldsymbol{\mu}_i\}_{i=1}^{N_s}$, where $N_s := |\mathcal{D}_{\text{train}}|$, and consider an arbitrary test point $\boldsymbol{\mu} \in \mathcal{D}$. We define the vector of training responses and the *model* of the test response at time instance $n \in \mathbb{T}$ as

$$\bar{\boldsymbol{\delta}}^n(\boldsymbol{\mu}) := \begin{bmatrix} \boldsymbol{\delta}^n_{\text{train}} \\ \hat{\delta}^n(\boldsymbol{\mu}) \end{bmatrix},$$

where $\boldsymbol{\delta}^n_{\text{train}} := [\delta^n(\boldsymbol{\mu}_1) \ \cdots \ \delta^n(\boldsymbol{\mu}_{N_s})]^T$. We then assume the data are drawn from the prior distribution

$$\bar{\boldsymbol{\delta}}^n(\boldsymbol{\mu}) \sim \mathcal{N}(\mathbf{0}, \bar{\boldsymbol{\Sigma}}^n(\boldsymbol{\mu}) + \lambda \mathbf{I}),$$

where $\lambda \in \mathbb{R}_+$ is an additive noise hyper-parameter and $\bar{\boldsymbol{\Sigma}}^n(\boldsymbol{\mu})$ is a positive definite $(N_s+1) \times (N_s+1)$ covariance matrix of the form

$$\bar{\boldsymbol{\Sigma}}^n(\boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{\Sigma}^n_{\text{train}} & \boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})^T \\ \boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu}) & 1 \end{bmatrix},$$

where we employ radial-basis-function kernels such that

$$\begin{aligned} [\boldsymbol{\Sigma}^n_{\text{train}}]_{ij} &:= \exp \frac{\left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2^2}{h^2}, \qquad i, j = 1, \ldots, N_s, \\ [\boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})]_{1i} &= \exp \frac{\left\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\right\|_2^2}{h^2}, \quad i = 1, \ldots, N_s, \end{aligned} \tag{57}$$

Given the training data, the posterior distribution of the model of the test response is

$$\hat{\delta}^n(\boldsymbol{\mu}) \sim \mathcal{N}\bigg(\boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})(\boldsymbol{\Sigma}^n_{\text{train}} + \lambda \mathbf{I})^{-1}\boldsymbol{\delta}^n_{\text{train}}, 1 + \lambda - \boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})(\boldsymbol{\Sigma}^n_{\text{train}} + \lambda \mathbf{I})^{-1}(\boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu}))^T\bigg),$$

Comparing with Eq. (26), we see that this model corresponds to the following quantities in the present T-MLEM framework:

$$\begin{aligned} \hat{\delta}^n_f(\boldsymbol{\mu}) &= \boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})(\boldsymbol{\Sigma}^n_{\text{train}} + \lambda \mathbf{I})^{-1}\boldsymbol{\delta}^n_{\text{train}} \\ \hat{\delta}^n_\epsilon(\boldsymbol{\mu}) &\sim \mathcal{N}\bigg(0, 1 + \lambda - \boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu})(\boldsymbol{\Sigma}^n_{\text{train}} + \lambda \mathbf{I})^{-1}(\boldsymbol{\Sigma}^n_{\text{test}}(\boldsymbol{\mu}))^T\bigg). \end{aligned} \tag{58}$$

This model requires specifying parameters λ and h. We compute the length-scale parameter h by maximizing the log-marginal likelihood of the posterior distribution (see Ref. [40]), while we treat the noise parameter λ as a hyper-parameter that is selected using a validation procedure with a grid search.

Appendix E Keras algorithms

We implement the ANN, ARX, ANN-I, LARX, RNN, and LSTM models using Keras. This appendix provides the Python code that was used to construct these networks. For ARX and ANN-I, only networks corresponding to the RT training formulation are presented.

E.1 Artificial neural network (ANN)

```python
def build_nn(X,y,hyperparams):
  neurons,depth,reg = hyperparams[0],np.int(hyperparams[1]),hyperparams[2]
  model = Sequential()
  model.add(Dense(neurons,input_shape=(None,np.shape(X)[-1]),activation='relu',use_bias=True
    ,kernel_regularizer=regularizers.l2(10.**reg)))
  for i in range(1,depth):
    model.add(Dense(neurons,activation='relu',use_bias=True,
      kernel_regularizer=regularizers.l2(10**reg)))
  model.add(Dense(np.shape(y)[-1],activation='linear',use_bias=True))
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 1: Python code for the initialization of ANN.

E.2 Autoregressive w/ Exogenous Inputs (ARX)

```python
def build_model_arx(X,y,hyperparams):
  neurons,reg = int(hyperparams[0]),np.int(hyperparams[1])
  model = Sequential()
  model.add(SimpleRNN(np.shape(y)[-1],input_shape=(None,np.shape(X)[-1]),return_sequences=
    True,activation='linear',kernel_regularizer=regularizers.l2(10**reg),
    recurrent_regularizer=regularizers.l2(10**reg)) )
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 2: Python code for the initialization of ARX.

E.3 Integrated Artificial Neural Network (ANN-I)

```python
def build_model_nni(X,y,hyperparams):
  neurons,depth,reg = hyperparams[0],np.int(hyperparams[1]),hyperparams[2]
  model = Sequential()
  model.add(Dense(neurons,input_shape=(None,np.shape(X)[-1]),activation='relu',use_bias=True
    ,kernel_regularizer=regularizers.l2(10.**reg)))
  for i in range(1,depth):
    model.add(Dense(neurons,activation='relu',use_bias=True,kernel_regularizer=regularizers.
    l2(10**reg)))
  model.add(Dense(np.shape(y)[-1],activation='linear',use_bias=True))
  model.add(SimpleRNN(np.shape(y)[-1],return_sequences=True ,activation='linear',use_bias=
    False,recurrent_initializer='ones',kernel_initializer='ones',name='RNN'))
  model.get_layer('RNN').trainable = False
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 3: Python code for the initialization of ANN-I.

E.4 Latent Auto-Regressive w/ Exogenous Inputs (LARX)

```python
def build_model_rnn_linear(X,y,hyperparams):
  neurons,reg = int(hyperparams[0]),hyperparams[1]
  model = Sequential()
  model.add(SimpleRNN(neurons,input_shape=(None,np.shape(X)[-1]),return_sequences=True,
    activation='linear',kernel_regularizer=regularizers.l2(10**reg),recurrent_regularizer=
    regularizers.l2(10**reg)) )
  model.add(Dense(np.shape(y)[-1],activation='linear',use_bias=True))
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 4: Python code for the initialization of LARX.

E.5 Recurrent neural network (RNN)

```
def build_rnn(X,y,hyperparams):
  neurons,depth,reg = int(hyperparams[0]),np.int(hyperparams[1]),hyperparams[2]
  model = Sequential()
  model.add(SimpleRNN(neurons,input_shape=(None,np.shape(X)[-1]),
   return_sequences=True , kernel_regularizer=regularizers.l2(10**reg),
   recurrent_regularizer=regularizers.l2(10**reg)) )
  for i in range(1,depth):
    model.add(SimpleRNN(neurons,
     return_sequences=True,
     kernel_regularizer=regularizers.l2(10**reg) ,
     recurrent_regularizer=regularizers.l2(10**reg)) )
  model.add(Dense(np.shape(y)[-1],activation='linear',use_bias=True))
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 5: Python code for the initialization of the RNN.

E.6 Long short-term memory network (LSTM)

```
def build_lstm(X,y,hyperparams):
  neurons,depth,reg = int(hyperparams[0]),np.int(hyperparams[1]),hyperparams[2]
  model = Sequential()
  model.add(LSTM(neurons,input_shape=(None,np.shape(X)[-1]), return_sequences=True ,
    kernel_regularizer=regularizers.l2(10**reg) , recurrent_regularizer=regularizers.l2(10**
    reg)) )
  for i in range(1,depth):
    model.add(LSTM(neurons,return_sequences=True,kernel_regularizer=regularizers.l2(10**reg)
    ,recurrent_regularizer=regularizers.l2(10**reg)) )
  model.add(Dense(np.shape(y)[-1],activation='linear',use_bias=True))
  model.compile(loss='mean_squared_error', optimizer='Adam')
  return model
```

Listing 6: Python code for the initialization of LSTM.

References

[1] J. Ackmann, J. Moarotzke, and P. Korn, *Stochastic goal-oriented error estimation with memory*, Journal of Computational Physics, 348 (2017), pp. 195–219.

[2] M. Ainsworth and J. T. Oden, *A posteriori error estimation in finite element analysis*, Computer methods in applied mechanics and engineering, 142 (1997), pp. 1–88.

[3] I. Babuška and W. Rheinboldt, *A-posteriori error estimates for the finite element method*, International Journal for Numerical Methods in Engineering, 12 (1978), pp. 1597–1615.

[4] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annu. Rev. Fluid Mech., 25 (1993), pp. 539–575.

[5] P. J. Blonigan, *Least Squares Shadowing for sensitivity analysis of large chaotic systems and fluid flows*, PhD thesis, Massachusetts Institute of Technology, 2016.

[6] T. Bui-Thanh, K. Willcox, and O. Ghattas, *Model reduction for large-scale systems with high-dimensional parametric input space*, SIAM Journal on Scientific Computing, 30 (2008), pp. 3270–3288.

[7] Bui-Thanh, T. and Wilcox, K. and Ghattas, O., *Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications*, AIAA Journal, 46 (2008), pp. 2520–2529.

[8] C. Prud'Homme and D.V. Rovas and K. Veroy and L. Machiels and Y. Maday and A. Patera and G. Turinici, *Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations*, Proceedings SMA Symposium, (2002).

[9] K. Carlberg, M. Barone, and H. Antil, *Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction*, Journal of Computational Physics, 330 (2017), pp. 693–734.

[10] F. Chollet et al., *Keras*. `https://keras.io`, 2015.

[11] Z. Drmac and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, SIAM Journal on Scientific Computing, 38 (2016), pp. A631–A648.

[12] M. Drohmann and K. Carlberg, *The ROMES method for statistical modeling of reduced-order-model error*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 116–145.

[13] M. S. Eldred, A. A. Giunta, S. S. Collis, N. A. Alexandrov, and R. M. Lewis, *Second-order corrections for surrogate-based optimization with model hierarchies*, in 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, 2004.

[14] R. Everson and L. Sirovich, *Karhunen–Loève procedure for gappy data*, Journal of the Optical Society of America A, 12 (1995), pp. 1657–1644.

[15] B. Freno and K. Carlberg, *Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations*, Computer Methods in Applied Mechanics and Engineering, 348 (2019), pp. 250–296.

[16] S. E. Gano, J. E. Renaud, and B. Sanders, *Hybrid variable fidelity optimization by using a kriging-based scaling function*, AIAA Journal, 43 (2005).

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] M. Grepl and A. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, ESAIM, 39 (2005), pp. 157–181.

[19] M. Hinze and M. Kunkel, *Residual based sampling in POD model order reduction of drift–diffusion equations in parametrized electrical networks*, ZAMM-Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, 92 (2012), pp. 91–104.

[20] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Comput., 9 (1997), pp. 1735–1780.

[21] C. Homescu, L. R. Petzold, and R. Serban, *Error estimation for reduced-order models of dynamical systems*, SIAM Review, 49 (2007).

[22] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller, *Sequential kriging optimization using multiple-fidelity evaluations*, Structural and Multidisciplinary Optimization, 32 (2006).

[23] D. Huynh, D. Knezevic, Y. Chen, J. S. Hesthaven, and A. Patera, *A natural-norm successive constraint method for inf-sup lower bounds*, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 1963–1975.

[24] I. Babuška and W.C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM Journal on Numerical Analysis, 15 (1978), pp. 736–754.

[25] M. C. Kennedy and A. O'Hagan, *Bayesian Calibration of Computer Models*, J.R. Statist. Soc. B, 63 (2001), pp. 425–464.

[26] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, CoRR, abs/1412.6980 (2015).

[27] K. Lee and K. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, arXiv e-print, (2018).

[28] J. C.-C. Lu, *An a posteriori Error Control Framework for Adaptive Precision Optimization using Discontinuous Galerkin Finite Element Method*, PhD thesis, Massachusetts Institute of Technology, 2005.

[29] Y. Maday and A. T. Patera, *Numerical analysis of a posteriori finite element bounds for linear functional outputs*, Mathematical Models and Methods in Applied Sciences, 10 (2000), pp. 785–799.

[30] A. March and K. Willcox, *Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives*, AIAA Journal, 50 (2012).

[31] A. Moosavi, R. Ştefănescu, and A. Sandu, *Multivariate predictions of local reduced-order-model errors and dimensions*, International Journal for Numerical Methods in Engineering, 113 (2018), pp. 512–533.

[32] M. C. Mozer, *A focused backpropagation algorithm for temporal pattern recognition*, Complex Systems, 3 (1989).

[33] C. Olah, *Understanding LSTM networks*, August 2015.

[34] S. Pagani, A. Manzoni, and K. Carlberg, *Statistical closure modeling for reduced-order models of stationary systems by the ROMES method*, arXiv:1901.02792, (2019).

[35] S. Pagani, A. Manzoni, and A. Quarteroni, *Efficient state/parameter estimation in nonlinear unsteady pdes by a reduced basis ensemble kalman filter*, SIAM/ASA Journal on Uncertainty Quantification, 5 (2017), pp. 890–921.

[36] M. Paraschivoiu and A. T. Patera, *A hierarchical duality approach to bounds for the outputs of partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 158 (1998), pp. 389–407.

[37] M. Paraschivoiu, J. Peraire, and A. T. Patera, *A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 150 (1997), pp. 289–312.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[39] C. Prud'Homme, D. Rovas, K. Veroy, L. Machiels, Y. Maday, A. Patera, and G. Turinici, *Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods*, Journal of Fluids Engineering, 124 (2001), pp. 70–80.

[40] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[41] A. J. Robinson and F. Fallside, *The utility driven dynamic error propagation network*, Tech. Rep. CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, Cambridge, UK, 1987.

[42] D. Rovas, L. Machiels, and Y. Maday, *Reduced-basis output bound methods for parabolic problems*, IMA Journal of Numerical Analysis, 26 (2006).

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536.

[44] Y. S. Shimizu, *Output-based error estimation for chaotic flows using reduced-order modeling*, in AIAA SciTech Forum, (AIAA 2018-0826), 2018.

[45] S. Trehan, K. Carlberg, and L. J. Durlofsky, *Error estimation for surrogate models of dynamical systems using machine learning*, International Journal for Numerical Methods in Engineering, 112 (2017), pp. 1801–1827.

[46] D. A. Venditti and D. L. Darmofal, *Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow*, Journal of Computational Physics, 164 (2000), pp. 204 – 227.

[47] D. A. Venditti and D. L. Darmofal, *Grid adaptation for functional outputs: Application to two-dimensional inviscid flows*, Journal of Computational Physics, 176 (2002), pp. 40 – 69.

[48] Y. Wu and U. Hetmaniuk, *Adaptive training of local reduced bases for unsteady incompressible Navier–Stokes flows*, International Journal for Numerical Methods in Engineering, 103 (2015), pp. 183–204.

[49] M. Yano and A. T. Patera, *An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs*, Computer Methods in Applied Mechanics and Engineering, (2018).

[50] M. Zahr, *Adaptive model reduction to accelerate optimization problems governed by partial differential equations*, PhD thesis, Stanford University, 2016.

[51] M. Zahr, K. Carlberg, and D. Kouri, *An efficient, globally convergent method for optimization under uncertainty using adaptive model reduction and sparse grids*, SIAM/ASA Journal on Uncertainty Quantification, 6 (2019), pp. 877–912.

[52] M. J. Zahr and C. Farhat, *Progressive construction of a parametric reduced-order model for PDE-constrained optimization*, International Journal for Numerical Methods in Engineering, 102 (2015), pp. 1111–1135.

[53] R. Ştefănescu, A. Moosavi, and A. Sandu, *Parametric domain decomposition for accurate reduced order models: Applications of MP-LROM methodology*, Journal of Computational and Applied Mathematics, 340 (2018), pp. 629 – 644.