SANDIA REPORT

SAND2014-19406
Unlimited Release
Printed October 2014

Breaking Computational Barriers:
Real-time Analysis and Optimization with Large-scale Nonlinear Models via Model Reduction

Kevin Carlberg, Martin Drohmann, Ray Tuminaro, Paul Boggs, Jaideep Ray, Bart van Bloemen Waanders

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2014-19406
Unlimited Release
Printed October 2014

Breaking Computational Barriers:
Real-time Analysis and Optimization with Large-scale Nonlinear Models via Model Reduction

Kevin Carlberg
Quantitative Modeling & Analysis
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
carlberg@sandia.gov

Martin Drohmann
Quantitative Modeling & Analysis
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
mdrohmann@gmail.com

Ray Tuminaro
Computational Mathematics
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
rstumin@sandia.gov

Paul Boggs
Quantitative Modeling & Analysis
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
ptboggs@sandia.gov

Jaideep Ray
Quantitative Modeling & Analysis
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
jairay@sandia.gov

Bart van Bloemen Waanders
Optimization & Uncertainty Estimation
Sandia National Laboratories
7011 East Avenue, MS 9159
Livermore, CA 94550
bartv@sandia.gov

Abstract

Model reduction for dynamical systems is a promising approach for reducing the computational cost of large-scale physics-based simulations to enable high-fidelity models to be used in many-query (e.g., Bayesian inference) and near-real-time (e.g., fast-turnaround simulation) contexts. While model reduction works well for specialized problems such as linear time-invariant systems, it is much more difficult to obtain accurate, stable, and efficient reduced-order models (ROMs) for systems with general nonlinearities. This report describes several advances that enable nonlinear reduced-order models (ROMs) to be deployed in a variety of time-critical settings.

First, we present an error bound for the Gauss–Newton with Approximated Tensors (GNAT) nonlinear model reduction technique. This bound allows the state-space error for the GNAT method to be quantified when applied with the backward Euler time-integration scheme. Second, we present a methodology for preserving classical Lagrangian structure in nonlinear model reduction. This technique guarantees that important properties—such as energy conservation and symplectic time-evolution maps—are preserved when performing model reduction for models described by a Lagrangian formalism (e.g., molecular dynamics, structural dynamics). Third, we present a novel technique for decreasing the *temporal complexity*—defined as the number of Newton-like iterations performed over the course of the simulation—by exploiting time-domain data. Fourth, we describe a novel method for refining projection-based reduced-order models *a posteriori* using a goal-oriented framework similar to mesh-adaptive *h*-refinement in finite elements. The technique allows the ROM to generate *arbitrarily accurate* solutions, thereby providing the ROM with a 'failsafe' mechanism in the event of insufficient training data. Finally, we present the reduced-order model error surrogate (ROMES) method for statistically quantifying reduced-order-model errors. This enables ROMs to be rigorously incorporated in uncertainty-quantification settings, as the error model can be treated as a source of epistemic uncertainty.

This work was completed as part of a Truman Fellowship appointment. We note that much additional work was performed as part of the Fellowship. One salient project is the development of the Trilinos-based model-reduction software module *Razor*, which is currently bundled with the Albany PDE code and currently allows nonlinear reduced-order models to be constructed for any application supported in Albany. Other important projects include the following:

1. ROMES-equipped ROMs for Bayesian inference: K. Carlberg, M. Drohmann, F. Lu (Lawrence Berkeley National Laboratory), M. Morzfeld (Lawrence Berkeley National Laboratory).
2. ROM-enabled Krylov-subspace recycling: K. Carlberg, V. Forstall (University of Maryland), P. Tsuji, R. Tuminaro.
3. A pseudo balanced POD method using only dual snapshots: K. Carlberg, M. Sarovar.
4. An analysis of discrete v. continuous optimality in nonlinear model reduction: K. Carlberg, M. Barone, H. Antil (George Mason University).

Journal articles for these projects are in progress at the time of this writing.

Acknowledgment

Kevin Carlberg gratefully acknowledges the Truman selection committee for granting him the unique opportunity and resources to lead a three-year research project. Kevin also acknowledges Julien Cortial for performing excellent work in developing the Razor software module, and Martin Drohmann for his great work and patience in developing the ROMES method. Kevin also thanks Paul Boggs and Jaideep Ray for providing three years of invaluable mentorship. Kevin also acknowledges Jerry McNeish, Jim Costa, and Len Napolitano for their managerial support during his tenure as a Truman Fellow.

This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000. The authors also acknowledge support by the Department of Energy Office of Advanced Scientific Computing Research under contract 10-014804.

Contents

Chapter 1

Preserving Lagrangian structure in nonlinear model reduction with application to structural dynamics

This chapter presents a methodology for preserving classical Lagrangian structure when performing model reduction on nonlinear dynamical systems described by a Lagrangian formalism. The method guarantees that intrinsic properties such as energy conservation are preserved by the reduced-order model. Important applications include molecular dynamics and structural dynamics; numerical experiments are executed on a structural-dynamics example. This work has been submitted to the SIAM Journal on Scientific Computing and is past the first round of revisions at the time of this writing.

PRESERVING LAGRANGIAN STRUCTURE IN NONLINEAR MODEL REDUCTION WITH APPLICATION TO STRUCTURAL DYNAMICS

KEVIN CARLBERG*, RAY TUMINARO†, AND PAUL BOGGS‡

Abstract. This work proposes a model-reduction methodology that preserves Lagrangian structure and achieves computational efficiency in the presence of high-order nonlinearities and arbitrary parameter dependence. As such, the resulting reduced-order model retains key properties such as energy conservation and symplectic time-evolution maps. We focus on parameterized simple mechanical systems subjected to Rayleigh damping and external forces, and consider an application to nonlinear structural dynamics. To preserve structure, the method first approximates the system's 'Lagrangian ingredients'—the Riemannian metric, the potential-energy function, the dissipation function, and the external force—and subsequently derives reduced-order equations of motion by applying the (forced) Euler–Lagrange equation with these quantities. From the algebraic perspective, key contributions include two efficient techniques for approximating parameterized reduced matrices while preserving symmetry and positive definiteness: matrix gappy POD and reduced-basis sparsification (RBS). Results for a parameterized truss-structure problem demonstrate the practical importance of preserving Lagrangian structure and illustrate the proposed method's merits: it reduces computation time while maintaining high accuracy and stability, in contrast to existing nonlinear model-reduction techniques that do not preserve structure.

Key words. nonlinear model reduction, structure preservation, Lagrangian dynamics, Hamiltonian dynamics, structural dynamics, positive definiteness, matrix symmetry

1. Introduction. Computational modeling and simulation for parameterized simple mechanical systems characterized by a Lagrangian formalism has become indispensable across a variety of industries. For example, computational structural dynamics tools have become widely used in applications ranging from aerospace to biomedical-device design; molecular-dynamics simulations have gained popularity in materials science and biology. However, the high computational cost incurred by simulating large-scale simple mechanical systems can result in simulation times on the order of weeks. As a result, these simulation tools are impractical for time-critical applications such as nondestructive evaluation for structural health monitoring, multiscale modeling, design optimization, and uncertainty quantification.

Model-reduction methods present a promising approach for addressing time-critical problems. During the *offline stage*, these methods perform computationally expensive 'training' tasks, which may include evaluating the high-fidelity model for several instances of the system parameters and computing a representative low-dimensional subspace for the configuration variables. Then, during the inexpensive *online stage*, these methods quickly compute approximate solutions for arbitrary parameter values via a projection process of the high-fidelity-model equations onto this low-dimensional subspace; they also introduce other approximations when nonlinearities are present. This offline/online strategy is effective in two scenarios: 'many query' problems (e.g., Bayesian inference), where the high offline cost is amortized over many online evaluations, and real-time problems (e.g., control) characterized by stringent constraints on online evaluation time.

Generating a reduced-order model that preserves the Lagrangian structure intrinsic to mechanical systems is not a trivial task. Such structure is critical to preserve, as it leads to fundamental properties such as energy conservation, conservation of quantities associated with symmetries in the system, and symplectic time-evolution maps. In fact, the class of structure-preserving time integrators (e.g., geometric integrators [17], variational integrators [21]) has been developed to ensure that the discrete solution to the high-fidelity model associates with the time-evolution map of a (modified) Lagrangian system.

Lall et al. [20] show that performing a Galerkin projection on the Euler–Lagrange equation—as opposed to the first-order state-space form—leads to a reduced-order model that preserves Lagrangian structure. However, the computational cost of assembling the associated low-dimensional equations of motion scales with the dimension of the high-fidelity model. For this reason, this approach is efficient only when the low-dimensional operators can be assembled *a priori*; this occurs only in very limited cases e.g., when operators have a low-order polynomial dependence on the state and are affine in functions of the parameters [24].

Several methods have been developed in the context of nonlinear-ODE model reduction that can reduce

*Harry S. Truman Fellow, Quantitative Modeling & Analysis Department ktcarlb@sandia.gov
†Numerical Analysis and Applications Department, rstumin@sandia.gov
‡Quantitative Modeling & Analysis Department (retired), ptboggs@sandia.gov

the computational cost of assembling the low-dimensional equations of motion. However, these methods destroy Lagrangian structure when applied to simple mechanical systems. For example, collocation approaches [3, 25] perform a Galerkin projection on only a small subset of the full-order equations characterizing the high-fidelity model. The discrete empirical interpolation method (DEIM) [9, 15, 11] and gappy proper orthogonal decomposition (POD) method [13, 6, 7] compute a few entries of the vector-valued nonlinear functions, and then approximate the uncomputed entries by interpolation or least-squares regression with an empirically derived basis. Galerkin projection can then be performed with the approximated functions.

The goal of this work is to devise a reduced-order model for nonlinear simple mechanical systems with general parameter dependence that leads to computationally inexpensive online solutions and preserves Lagrangian structure. We focus particularly on parameterized structural-dynamics models under Rayleigh damping and external forces. The methodology we propose constructs a reduced-order model by first approximating the 'Lagrangian ingredients' (i.e., quantities defining the problem's Lagrangian structure) and subsequently deriving the equations of motion by applying the Euler–Lagrange equation to these ingredients. The method approximates the Lagrangian ingredients as follows:

I. *Configuration space.* The low-dimensional configuration space is derived using standard dimension-reduction techniques, e.g., proper orthogonal decomposition, modal decomposition.

II. *Riemannian metric.* The Riemannian metric is defined by a (parameterized) low-dimensional symmetric positive-definite matrix. We propose two efficient methods for approximating this low-dimensional matrix that preserve symmetry and positive definiteness.

III. *Potential-energy function.* The potential energy function is approximated by employing the original potential-energy function, but with the low-dimensional reduced-basis matrix replaced by a low-dimensional *sparse* matrix with only a few nonzero rows. This sparse matrix is computed online by matching the gradient of the reduced potential to first order about equilibrium.

IV. *Dissipation function.* The damping matrix associated with the Rayleigh dissipation function is a linear combination of the mass matrix (which defines the Riemannian metric) and the Hessian of the potential. Thus, we form the approximated Rayleigh dissipation function in the same fashion, but employ the approximated mass matrix from ingredient II and approximated potential from ingredient III.

V. *External force.* The external force is derived by applying the Lagrange–D'Alembert principle with variations in the configuration space. We approximate this by applying gappy POD reconstruction to the external force as expressed in the original coordinates. As a result, the external force appearing in the reduced-order equations of motion can be derived by applying the Lagrange–D'Alembert principle to this modified external force with variations restricted to the low-order configuration space.

We note that a structure-preserving method [5] has been recently proposed for nonlinear port-Hamiltonian systems, which are generalizations of Hamiltonian systems. While this technique guarantees that properties such as stability and passivity are preserved, it does not in fact preserve Lagrangian or classical Hamiltonian structure.[1]

The remainder of the paper is organized as follows. Section 2 introduces the Lagrangian-mechanics formulation. Section 3 outlines existing model-reduction techniques and highlights the need for an efficient, structure-preserving method. Section 4 presents the proposed method. As hinted above, preserving structure for Lagrangian ingredient II is equivalent to efficiently approximating a low-dimensional reduced matrix while preserving symmetry and positive definiteness. This algebraic task is relevant to a broad scope of applications, e.g., approximating extreme eigenvalues/eigenvectors of a parameterized matrix, preserving Hessian positive definiteness in optimization algorithms. For this reason, Section 5 presents approximation techniques for Lagrangian ingredient II in a stand-alone algebraic setting that does not rely on the Lagrangian formalism. Similarly, Section 6 considers Lagrangian ingredient III in a purely algebraic context. Section 7 presents numerical experiments applied to a simple mechanical system from structural dynamics. Finally, Section 8 summarizes the contributions and suggests further research.

We note that the model-reduction methods proposed in this work also preserve Hamiltonian structure when the Hamiltonian formulation of classical mechanics is used. Finally, the label prefix 'S' indicates content included as supplementary material.

[1] In particular, the resulting reduced-order equations of motion cannot be derived from approximated ingredients such as those enumerated above; as a consequence, the approach does not ensure symplecticity or energy conservation for conservative systems, for example.

2. Lagrangian dynamics formulation. We consider nonlinear *simple mechanical systems* defined by a parameterized triple (Q, g, V); system parameters $\mu \in \mathcal{D}$ with parameter domain $\mathcal{D}$ may describe, for example, variations in material properties. The triple is composed of:

- A differentiable configuration manifold Q. We take $Q = \mathbb{R}^N$ where N denotes the number of degrees of freedom in the model, considered to be 'large' in this work.
- A parameterized Riemannian metric $g(\mathbf{v}, \mathbf{w}; \mu)$, where $\mathbf{v}$ and $\mathbf{w}$ belong to the tangent bundle of Q. We take $g(\mathbf{v}, \mathbf{w}; \mu) = \mathbf{v}^T \mathbf{M}(\mu) \mathbf{w}$, where $\mathbf{M}(\mu)$ denotes the $N \times N$ parameterized symmetric positive-definite mass matrix.
- A parameterized potential-energy function $V : Q \times \mathcal{D} \to \mathbb{R}$, where the mapping $(\mathbf{q}; \mu) \mapsto V$ is nonlinear in both arguments.

The kinetic energy of a simple mechanical system can be expressed as $T(\dot{\mathbf{q}}; \mu) = \frac{1}{2} g(\dot{\mathbf{q}}, \dot{\mathbf{q}}; \mu) = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M}(\mu) \dot{\mathbf{q}}$, where $\mathbf{q} : [0, \mathsf{T}] \to Q$ denotes the time-dependent configuration variables and T denotes the final time. This leads to the following expression for the Lagrangian, which represents the difference between the kinetic and potential energies:

$$L(\mathbf{q}, \dot{\mathbf{q}}; \mu) = \frac{1}{2} g(\dot{\mathbf{q}}, \dot{\mathbf{q}}; \mu) - V(\mathbf{q}; \mu) = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M}(\mu) \dot{\mathbf{q}} - V(\mathbf{q}; \mu). \tag{2.1}$$

The non-conservative forces[2] often consist of an applied external force and a dissipative force arising from Rayleigh viscous damping. This dissipative force derives from a positive-semidefinite dissipation function[3]

$$\mathcal{F}(\dot{\mathbf{q}}; \mu) \equiv \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{C}(\mu) \dot{\mathbf{q}}, \tag{2.2}$$

where $\mathbf{C}(\mu) = \alpha \mathbf{M}(\mu) + \beta \nabla_{\mathbf{qq}} V(\mathbf{q}_0(\mu); \mu)$ denotes a parameterized $N \times N$ symmetric positive-semidefinite matrix with $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$. Here, $\mathbf{q}_0 : \mathcal{D} \to \mathbb{R}^N$ denotes the (parameterized) equilibrium configuration such that $\nabla_{\mathbf{q}} V(\mathbf{q}_0(\mu); \mu) = 0$. So, we consider non-conservative forces of the form $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t; \mu) - \nabla_{\dot{\mathbf{q}}} \mathcal{F}(\dot{\mathbf{q}}; \mu)$, where $\mathbf{f}$ denotes the external force that is derived from the Lagrange–D'Alembert variational principle.

Given the Lagrangian (2.1), one can derive the equations of motion for a simple mechanical system subject to an external force and Rayleigh viscous damping from the forced Euler–Lagrange equation

$$\frac{d}{dt} \nabla_{\dot{\mathbf{q}}} L(\mathbf{q}, \dot{\mathbf{q}}; \mu) - \nabla_{\mathbf{q}} L(\mathbf{q}, \dot{\mathbf{q}}; \mu) = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t; \mu) - \nabla_{\dot{\mathbf{q}}} \mathcal{F}(\dot{\mathbf{q}}; \mu). \tag{2.3}$$

Substituting Eqs. (2.1) and (2.2) into Eq. (2.3) leads to the familiar equations of motion

$$\mathbf{M}(\mu) \ddot{\mathbf{q}} + \mathbf{C}(\mu) \dot{\mathbf{q}} + \nabla_{\mathbf{q}} V(\mathbf{q}; \mu) = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t; \mu). \tag{2.4}$$

Conservative mechanical systems, where $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t; \mu) = 0$ and $\mathbf{C}(\mu) = 0$, exhibit important properties and can be characterized using the (conservative) Hamiltonian formulation of classical mechanics. For example, these systems conserve energy and quantities associated with symmetry, and their time-evolution maps are symplectic. Because these properties are intrinsic characteristics of the mechanical systems, it is desirable for numerical methods and approximations to preserve these properties. For this reason, we aim to develop an efficient reduced-order model that preserves Lagrangian structure. The specific properties we seek to preserve were enumerated in Section 1: a configuration space, a parameterized Riemannian metric, a parameterized potential-energy function, a parameterized positive-semidefinite dissipation function, and an external force derived from the Lagrange–D'Alembert principle. The first three properties constitute the parameterized triple defining a simple mechanical system; the last two characterize non-conservative forces.

3. Existing model-reduction techniques. Model-reduction techniques aim to generate a low-dimensional model that is inexpensive to evaluate, yet captures the essential features of the high-fidelity model. These methods first conduct a computationally expensive offline stage during which they perform analyses (e.g., solving equations of motion, modal analyses) for a training set $\{\mu^i\}_{i=1}^p \subset \mathcal{D}$. The training set can be obtained by any design of experiments approach for sampling the parameter domain $\mathcal{D}$, e.g., Latin

[2] Conservative forces can be handled by directly including them in the Lagrangian.

[3] Non-viscously damped systems can also often be derived by a positive-semidefinite dissipation function [1].

hypercube sampling [22], greedy sampling [16]. Then, they employ data generated during these analyses to define a low-dimensional configuration manifold, and other approximations to achieve efficiency in the presence of nonlinearities. The resulting low-dimensional model can then be used to perform inexpensive analyses for any specified point $\mu^\star \in \mathcal{D}$ during the online stage.

When the configuration space is Euclidean (as is the case for the models considered herein), the configuration space of reduced dimension $n \ll N$ can be expressed as

$$\mathsf{Q}_r \equiv \{\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r \mid \mathbf{q}_r \in Q_r\}, \tag{3.1}$$

where $\bar{\mathbf{q}}(\mu) : \mathcal{D} \to \mathbb{R}^N$ denotes the (parameterized) reference configuration about which the affine reduced subspace is centered, $Q_r = \mathbb{R}^n$, and $\mathbf{V} \in \mathbb{R}^{N\times n}_*$ defines a (typically dense) parameter-independent matrix whose columns can be interpreted as a *reduced basis* spanning an n-dimensional subspace of $\mathbb{R}^N$. Here, $\mathbb{R}^{m\times n}_*$ denotes the noncompact Stiefel manifold: the set of full-column-rank $m \times n$ matrices. This leads to the following expression for the generalized coordinates and their derivatives:

$$\mathbf{q} = \bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r, \quad \dot{\mathbf{q}} = \mathbf{V}\dot{\mathbf{q}}_r, \quad \ddot{\mathbf{q}} = \mathbf{V}\ddot{\mathbf{q}}_r. \tag{3.2}$$

Thus, the low-dimensional configuration space can be described in terms of low-dimensional generalized coordinates $\mathbf{q}_r \in Q_r$ or in terms of original coordinates by Eq. (3.2). The basis $\mathbf{V}$ can be determined by a variety of techniques, including proper orthogonal decomposition and modal decomposition.

3.1. Galerkin projection. Model reduction based on Galerkin projection preserves Lagrangian structure. As pointed out by Lall et al. [20], the Galerkin projection must be carried out on the Euler–Lagrange equation (2.3)—not the first-order state-space form—in order to preserve this structure. Following their approach, Galerkin-projection-based methods replace the original configuration space Q by the reduced-order configuration space Q_r and subsequently derive equations of motion in the usual way using a set of lower-dimensional coordinates. The model then has structure identical to that of the original problem.

For simple mechanical systems subject to non-conservative forces, this amounts to defining the Lagrangian as

$$L_r(\mathbf{q}_r, \dot{\mathbf{q}}_r; \mu) \equiv L(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r, \mathbf{V}\dot{\mathbf{q}}_r; \mu) = \frac{1}{2}\dot{\mathbf{q}}_r{}^T\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}\dot{\mathbf{q}}_r - V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r; \mu) \tag{3.3}$$

and the dissipation function as

$$\mathcal{F}_r(\dot{\mathbf{q}}_r; \mu) \equiv \mathcal{F}(\mathbf{V}\dot{\mathbf{q}}_r; \mu) = \frac{1}{2}\dot{\mathbf{q}}_r{}^T\mathbf{V}^T\mathbf{C}(\mu)\mathbf{V}\dot{\mathbf{q}}_r.$$

The external force derived from the Lagrange–D'Alembert variational principle is transformed by (3.2) into

$$\mathbf{f}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r, t; \mu) \equiv \mathbf{V}^T\mathbf{f}(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r, \mathbf{V}\dot{\mathbf{q}}_r, t; \mu).$$

Following Section 2, the forced Euler–Lagrange equation applied to the Lagrangian L_r, the dissipation function $\mathcal{F}_r$, and the external force $\mathbf{f}_r$ leads to the reduced-order equations of motion

$$\frac{d}{dt}\nabla_{\dot{\mathbf{q}}_r}L_r(\mathbf{q}_r, \dot{\mathbf{q}}_r; \mu) - \nabla_{\mathbf{q}_r}L_r(\mathbf{q}_r, \dot{\mathbf{q}}_r; \mu) + \nabla_{\dot{\mathbf{q}}_r}\mathcal{F}_r(\dot{\mathbf{q}}_r; \mu) = \mathbf{f}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r, t; \mu). \tag{3.4}$$

This can be rewritten as

$$\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}\ddot{\mathbf{q}}_r + \mathbf{V}^T\mathbf{C}(\mu)\mathbf{V}\dot{\mathbf{q}}_r + \mathbf{V}^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r; \mu) = \mathbf{V}^T\mathbf{f}(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r, \mathbf{V}\dot{\mathbf{q}}_r, t; \mu). \tag{3.5}$$

Note that Eq. (3.5) could have been derived by applying Galerkin projection to the original Euler–Lagrange equation (2.4), i.e., making substitutions (3.2) and pre-multiplying the system of equations by $\mathbf{V}^T$.

Thus, the Galerkin reduced-order model preserves Lagrangian structure as it preserves all five properties:

I. a configuration space $Q_r = \mathbb{R}^n$, which relates to the original configuration space by Eq. (3.1),
II. a parameterized Riemannian metric $g_r(\mathbf{v}_r, \mathbf{w}_r; \mu) = \mathbf{v}_r^T\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}\mathbf{w}_r$,
III. a parameterized potential-energy function $V_r(\mathbf{q}_r; \mu) = V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r; \mu)$,
IV. a parameterized positive-semidefinite dissipation function $\mathcal{F}_r(\dot{\mathbf{q}}_r; \mu) = \frac{1}{2}\dot{\mathbf{q}}_r{}^T\mathbf{V}^T\mathbf{C}(\mu)\mathbf{V}\dot{\mathbf{q}}_r$, and
V. an external force $\mathbf{f}_r$ derived from applying the Lagrange–D'Alembert principle to the original external force $\mathbf{f}$, but restricted to variations in the configuration space Q_r.

3.1.1. Computational bottleneck. Although the equations of motion (3.5) are low dimensional, they remain computationally expensive to solve when the operators exhibit arbitrary parameter dependence and the potential is nonlinear. The reason is simple: computing the low-dimensional components of Eq. (3.5) incurs large-scale operations due to the density of $\mathbf{V}$. For example, computing $\mathbf{V}^T\mathbf{M}(\mu^\star)\mathbf{V}$ online incurs $\mathcal{O}(N\omega n + Nn^2)$ flops, where ω denotes the average number of nonzeros per row of the matrix $\mathbf{M}(\mu^\star)$. If the potential energy V exhibits a (general) nonlinear dependence on coordinates $\mathbf{q}$, the vector $\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r;\mu^\star)$ and product $\mathbf{V}^T\nabla_{\mathbf{q}}V$ must be computed for every instance of $\mathbf{q}_r$.

REMARK 1. If the mass matrix is affine in functions of the parameters $\mathbf{M}(\mu) = \sum_i^{m_\mathbf{M}} \alpha_i(\mu)\mathbf{M}_i$ with $\alpha_i : \mathcal{D} \to \mathbb{R}$, $\mathbf{M}_i \in \mathbb{R}^{N\times N}$, and $m_\mathbf{M} \ll N$, then products $\mathbf{V}^T\mathbf{M}_i\mathbf{V}$ can be assembled offline, and $\mathbf{V}^T\mathbf{M}(\mu^\star)\mathbf{V} = \sum_i^{m_\mathbf{M}} \alpha_i(\mu^\star)\left[\mathbf{V}^T\mathbf{M}_i\mathbf{V}\right]$ can be computed inexpensively, i.e., in $\mathcal{O}(m_\mathbf{M}n^2)$ floating-point operations (flops) during the online stage [19, 23]. Similar low-complexity results can be obtained for the other terms if they can be similarly expressed in separable form. However, affine parameter dependence is a quite limiting scenario and does not generally hold.

3.2. Complexity reduction. Several techniques have been developed to mitigate the computational bottleneck described in Section 3.1.1. Before applying projection, these methods compute (or sample) only a few entries of the vector-valued functions. In effect, this complexity-reduction strategy is equivalent to employing a sparse left-projection test basis, which leads to N-independent operation counts if the vector-valued functions exhibit H-independence (i.e., vector-valued functions have sparse Jacobians). Such methods have been successfully applied to ODEs that do not exhibit particular structure. However, when applied to mechanical systems described by Lagrangian mechanics, these techniques destroy Lagrangian structure.

3.2.1. Collocation. Collocation approaches [3, 25] compute only a subset of the full-order equations of motion (2.4) before applying Galerkin projection. That is, they approximate Eq. (3.5) as

$$\begin{aligned}\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{M}(\mu)\mathbf{V}\ddot{\mathbf{q}}_r + \mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{C}(\mu)\mathbf{V}\dot{\mathbf{q}}_r + \mathbf{V}^T\mathbf{P}\mathbf{P}^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r;\mu)\\ = \mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{f}(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r, \mathbf{V}\dot{\mathbf{q}}_r, t;\mu).\end{aligned} \tag{3.6}$$

Here, $\mathbf{P} \in \{0,1\}^{N\times m}$ is a (full-column-rank) 'sampling matrix' consisting of m selected columns of the $N \times N$ identity matrix.[4] If one considers the matrix $\mathbf{P}\mathbf{P}^T\mathbf{V}$ as defining a basis for a test space, Eq. (3.6) can be viewed as a Petrov–Galerkin projection.

Computing the components of Eq. (3.6) is inexpensive in the case of H-independence, i.e., when the matrices $\mathbf{M}$, $\mathbf{C}$, $\nabla_{\mathbf{qq}}V$, $\nabla_{\mathbf{q}}\mathbf{f}$, and $\nabla_{\dot{\mathbf{q}}}\mathbf{f}$ are sparse. To see this, consider the first term in Eq. (3.6): computing $\left(\mathbf{V}^T\mathbf{P}\right)\left(\mathbf{P}^T\mathbf{M}(\mu^\star)\right)\mathbf{V}$ for specific online point $\mu^\star \in \mathcal{D}$ incurs $\mathcal{O}(m\omega n + mn^2)$ flops. This cost is small if the sparsity measure of $\mathbf{M}$ is small, i.e., $\omega \ll N$.

However, this cost-reduction approach destroys the problem's structure, as it does not preserve the following Lagrangian properties described in Section 2:

II. The mass matrix $\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{M}(\mu)\mathbf{V}$ is not symmetric, so it does not define a metric.

III. The term $\mathbf{V}^T\mathbf{P}\mathbf{P}^T\nabla_{\mathbf{qq}}V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r;\mu)\mathbf{V}$ is not symmetric, so it cannot be the Hessian of a potential-energy function.

IV. The damping matrix $\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{C}(\mu)\mathbf{V}$ is not symmetric, so it does not derive from a dissipation function.

Note that Property I is trivially satisfied, as the configuration space can be described as $Q_r = \mathbb{R}^n$ and relates to the original configuration space by Eq. (3.1). Further, Property V is satisfied, because the non-conservative forces can be derived by applying the Lagrange–D'Alembert variational principle to a modified external force $\mathbf{P}\mathbf{P}^T\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu)$, but restricted to variations in the (true) configuration space Q_r.

3.2.2. DEIM/gappy POD. Discrete empirical interpolation [9, 15, 11] and gappy POD [13, 6, 7] approximate via least-squares regression the nonlinear or non-affine vector-valued functions appearing in Eq. (2.4); these include $\mathbf{M}(\mu)\ddot{\mathbf{q}}$, $\mathbf{C}(\mu)\dot{\mathbf{q}}$, $\nabla_{\mathbf{q}}V(\mathbf{q};\mu)$, and $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu)$. Because these approaches approximate each term in the governing equations separately, they often achieve higher accuracy than collocation.

During the offline stage, these methods construct an orthogonal basis $\mathbf{W}_{\boldsymbol{\theta}} \in \mathbb{R}^{N\times n_{\boldsymbol{\theta}}}$ with $n_{\boldsymbol{\theta}} \leq m$ for each nonlinear function $\boldsymbol{\theta}(t;\mu)$ appearing in the equations of motion; typically, the basis $\mathbf{W}_{\boldsymbol{\theta}}$ is computed

[4] Note that the sampling matrix—when acting on a vector—returns a subset (or sample) of its entries. Here, the term 'sampling' does not imply snapshot collection.

empirically via proper orthogonal decomposition (POD). This consists of two steps: 1) collect snapshots $\mathcal{X}_{\boldsymbol{\theta}} = \{\boldsymbol{\theta}(t;\mu) \mid t \in \mathsf{T}_{\text{sample}}(\mu),\ \mu \in \{\mu^i\}\}$, where $\mathsf{T}_{\text{sample}}(\mu) \subset [0,\mathsf{T}]$ designates the time instances taken by the time-integration method for the training simulation; and 2) compute $\mathbf{W}_{\boldsymbol{\theta}}$ by Algorithm S1 using $\mathcal{X}_{\boldsymbol{\theta}}$ and an energy criterion $\eta_{\boldsymbol{\theta}} \in [0,1]$ as inputs.

During the online stage, these methods approximate the nonlinear function as

$$\boldsymbol{\theta}(t;\mu) \approx \mathbf{W}_{\boldsymbol{\theta}}[\mathbf{P}^T\mathbf{W}_{\boldsymbol{\theta}}]^+\mathbf{P}^T\boldsymbol{\theta}(t;\mu), \tag{3.7}$$

where a superscript + denotes the Moore–Penrose pseudoinverse.

As with collocation, this approximation technique leads to computational-cost savings during the online stage if computing $\mathbf{P}^T\boldsymbol{\theta}(t;\mu)$ incurs a flop count independent of N, i.e., $\boldsymbol{\theta}(t;\mu)$ exhibits H-independence. Substituting least-squares approximations for the nonlinear functions into Eq. (3.5) yields the approximated reduced-order equations of motion

$$\mathbf{Y}_{\mathbf{M}\ddot{\mathbf{q}}_r}\mathbf{M}(\mu)\mathbf{V}\ddot{\mathbf{q}}_r + \mathbf{Y}_{\mathbf{C}\dot{\mathbf{q}}_r}\mathbf{C}(\mu)\mathbf{V}\dot{\mathbf{q}}_r + \mathbf{Y}_{\nabla_{\mathbf{q}}V}\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu)+\mathbf{V}\mathbf{q}_r;\mu) = \mathbf{Y}_{\mathbf{f}}\mathbf{f}\left(\mathbf{q}_0(\mu)+\mathbf{V}\mathbf{q}_r, \mathbf{V}\dot{\mathbf{q}}_r, t;\mu\right).$$

Here, we have used the notation $\mathbf{Y}_{\boldsymbol{\theta}} \equiv \mathbf{V}^T\mathbf{W}_{\boldsymbol{\theta}}\left[\mathbf{P}^T\mathbf{W}_{\boldsymbol{\theta}}\right]^+\mathbf{P}^T$, and the subscript of $\mathbf{Y}$ and $\mathbf{W}$ denotes the function for which the approximation has been constructed.

Unfortunately, this approximation method also destroys the Lagrangian structure. As before, Lagrangian properties II–IV are lost because the reduced mass, stiffness, and damping matrices are not symmetric.

4. Efficient, structure-preserving model reduction. The main idea of the proposed approach is to directly approximate the quantities defining the Lagrangian structure of the Galerkin-projection reduced-order model, and subsequently derive the equations of motion. Section 3.1 enumerates these quantities for the simple mechanical systems considered herein. Approximations to these ingredients should 1) preserve salient properties, 2) lead to computationally inexpensive reduced-order-model simulations, and 3) incur minimal approximation error. To this end, we propose a model defined by

I. a configuration space $Q_r = \mathbb{R}^n$, which relates to the original coordinates by Eq. (3.1),
II. an approximated Riemannian metric $\tilde{g}_r$,
III. an approximated potential-energy function $\tilde{V}_r$,
IV. an approximated positive-semidefinite dissipation function $\tilde{\mathcal{F}}_r$, and
V. an approximated external force $\tilde{\mathbf{f}}_r$ derived from applying the Lagrange–D'Alembert principle to an approximated force $\tilde{\mathbf{f}}$ represented in the original coordinates, but limited to variations in the reduced configuration space Q_r.

We derive the equations of motion by applying the forced Euler–Lagrange equation with these approximations:

$$\frac{d}{dt}\nabla_{\dot{\mathbf{q}}_r}\tilde{L}_r(\mathbf{q}_r,\dot{\mathbf{q}}_r;\mu) - \nabla_{\mathbf{q}_r}\tilde{L}_r(\mathbf{q}_r,\dot{\mathbf{q}}_r;\mu) + \nabla_{\dot{\mathbf{q}}_r}\tilde{\mathcal{F}}_r(\dot{\mathbf{q}}_r;\mu) = \tilde{\mathbf{f}}_r(\mathbf{q}_r,\dot{\mathbf{q}}_r,t;\mu), \tag{4.1}$$

where the approximated Lagrangian is defined as

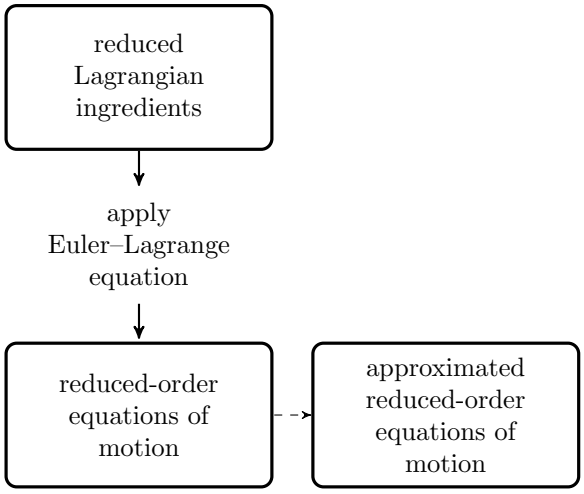
$$\tilde{L}_r(\mathbf{q}_r,\dot{\mathbf{q}}_r;\mu) \equiv \frac{1}{2}\tilde{g}_r(\dot{\mathbf{q}}_r,\dot{\mathbf{q}}_r;\mu) - \tilde{V}_r(\mathbf{q}_r;\mu). \tag{4.2}$$

Note that Eq. (4.1) approximates Eq. (3.4), while Eq. (4.2) approximates Eq. (3.3).

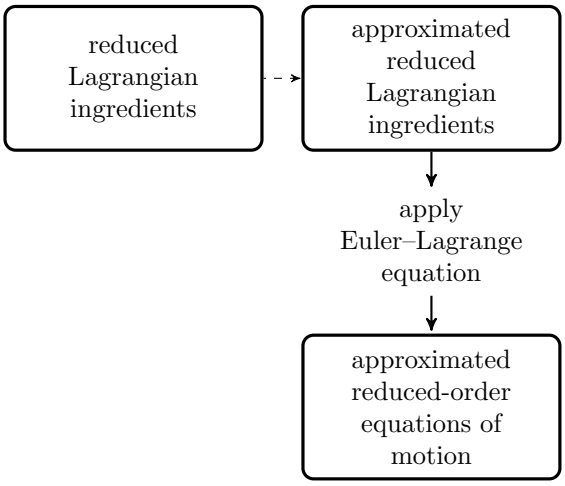
Figure 4.1 depicts the strategy graphically. Note that the proposed method is *not* equivalent to carrying out a Galerkin projection on the original equations of motion (2.4), as it employs a different technique to approximate each quantity in the Galerkin reduced-order equations of motion (3.5). The next sections describe two proposed methods that align with this strategy for structure preservation. For reference, Table 4.1 reports components of the equations of motion for these methods.

4.1. Riemannian-metric approximation $\tilde{g}_r$. The function $g_r : (\mathbf{v}_r,\mathbf{w}_r;\mu) \mapsto \mathbf{v}_r^T\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}\mathbf{w}_r$ is defined by a low-dimensional symmetric positive-definite matrix $\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}$. To ensure structure preservation, we propose computing an approximated Riemannian metric $\tilde{g}_r : \mathbb{R}^n \times \mathbb{R}^n \times \mathcal{D} \to \mathbb{R}$ as

$$\tilde{g}_r(\mathbf{v}_1,\mathbf{v}_2;\mu) \equiv \mathbf{v}_1^T\tilde{\mathbf{M}}(\mu)\,\mathbf{v}_2, \tag{4.3}$$



(a) **Existing complexity-reduction methods** (see Section 3.2). By approximating the equations of motion, such methods destroy Lagrangian structure.



(b) **Proposed approach**. By approximating Lagrangian ingredients before deriving the equations of motion, the approach preserves Lagrangian structure.

FIG. 4.1. *Comparing existing complexity-reduction approaches with the proposed approach. A dashed arrow implies a complexity-reduction approximation.*

method	mass matrix	damping matrix	potential-energy gradient	external force	str. pres.	low cost
Galerkin	$\mathbf{V}^T\mathbf{M}(\mu)\mathbf{V}$	$\mathbf{V}^T\mathbf{C}(\mu)\mathbf{V}$	$\mathbf{V}^T\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu)+\mathbf{V}\mathbf{q}_r;\mu)$	$\mathbf{V}^T\mathbf{f}$	yes	no
collocation	$\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{M}(\mu)\mathbf{V}$	$\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{C}(\mu)\mathbf{V}$	$\mathbf{V}^T\mathbf{P}\mathbf{P}^T\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu)+\mathbf{V}\mathbf{q}_r;\mu)$	$\mathbf{V}^T\mathbf{P}\mathbf{P}^T\mathbf{f}$	no	yes
gappy POD	$\mathbf{Y}_{\mathbf{M}\dot{\mathbf{q}}_r}\mathbf{M}(\mu)\mathbf{V}$	$\mathbf{Y}_{\mathbf{C}\dot{\mathbf{q}}_r}\mathbf{C}(\mu)\mathbf{V}$	$\mathbf{Y}_{\nabla_{\mathbf{q}}V}\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu)+\mathbf{V}\mathbf{q}_r;\mu)$	$\mathbf{Y}_{\mathbf{f}}\mathbf{f}$	no	yes
proposal 1	${\mathbf{U}_M}^T\mathbf{M}(\mu)\mathbf{U}_M$	$\alpha{\mathbf{U}_M}^T\mathbf{M}(\mu)\mathbf{U}_M+$ $\beta{\mathbf{U}_V}^T\nabla_{\mathbf{q}\mathbf{q}}V(\mathbf{q}_0(\mu);\mu)\mathbf{U}_V$	${\mathbf{U}_V}^T\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu)+\mathbf{U}_V\mathbf{q}_r;\mu)$	$\mathbf{Y}_{\mathbf{f}}\mathbf{f}$	yes	yes
proposal 2	$\sum_{i=1}^{n_M}\xi^i_{\mathbf{M}}(\mu)\mathbf{V}^T\underline{\mathbf{M}}^i\mathbf{V}$	$\alpha\sum_{i=1}^{n_M}\xi^i_{\mathbf{M}}(\mu)\mathbf{V}^T\underline{\mathbf{M}}^i\mathbf{V}+$ $\beta{\mathbf{U}_V}^T\nabla_{\mathbf{q}\mathbf{q}}V(\mathbf{q}_0(\mu);\mu)\mathbf{U}_V$	${\mathbf{U}_V}^T\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu)+\mathbf{U}_V\mathbf{q}_r;\mu)$	$\mathbf{Y}_{\mathbf{f}}\mathbf{f}$	yes	yes

TABLE 4.1

Terms appearing in the equations of motion for various model-reduction techniques. Here, $\mathbf{Y}_{\boldsymbol{\theta}}\equiv\mathbf{V}^T\mathbf{W}_{\boldsymbol{\theta}}\left[\mathbf{P}^T\mathbf{W}_{\boldsymbol{\theta}}\right]^+\mathbf{P}^T$ *and* $\mathbf{U}_{\boldsymbol{\theta}}\equiv\mathbf{P}\underline{\mathbf{U}}_{\boldsymbol{\theta}}$*. In principle, a different sampling matrix* $\mathbf{P}$ *could be used for each approximation; however, such an approach would complicate the online implementation, as each term would require a different 'sample mesh' [7].*

where $\tilde{\mathbf{M}}(\mu)$ is an $n\times n$ matrix that must be symmetric and positive definite.

Section 5 describes the algebraic problem of approximating this matrix and proposes two structure-preserving techniques for computing the approximation $\tilde{\mathbf{M}}(\mu)$. The first method (proposal 1 in Table 4.1) employs the reduced-basis sparsification of Section 5.1 and approximates $\tilde{\mathbf{M}}(\mu)$ via Eq. (5.1). The second method (proposal 2 in Table 4.1) employs the matrix gappy POD of Section 5.2 and approximates this matrix by Eq. (5.8). Procedures 2 (Section 5.1) and 3 (Section 5.2) provide the offline/online steps.

4.2. Potential-energy-function approximation $\tilde{V}_r$. Because only $\nabla_{\mathbf{q}_r}\tilde{V}_r$ appears in the reduced-order equations of motion (see Eqs. (4.1)–(4.2)), the task of approximating the potential energy should focus on accurately representing its gradient. Section 6 describes this algebraic task and proposes approximating the reduced potential energy according to Eq. (6.1). We adopt this approximation, and also set the reference configuration to equilibrium $\bar{\mathbf{q}}=\mathbf{q}_0$ to avoid the limitations associated with other choices (see the discussion in Section 6). Procedure 4 of Section 6.2 describes the offline/online steps for this approximation.

4.3. Dissipation-function approximation $\mathcal{F}_r$. To maintain the Rayleigh-damping structure, we simply approximate the damping matrix as a linear combination of the approximated mass matrix and

Hessian of the potential at equilibrium

$$\tilde{\mathcal{F}}_r(\mathbf{v};\mu) = \frac{\alpha}{2}\mathbf{v}^T\tilde{\mathbf{M}}(\mu)\,\mathbf{v} + \frac{\beta}{2}\mathbf{v}^T\nabla_{\mathbf{q}_r\mathbf{q}_r}\tilde{V}_r(0;\mu)\mathbf{v},$$

where α and β are the Rayleigh damping coefficients defined in Section 2.

4.4. External-force approximation $\tilde{\mathbf{f}}_r$. The following form of the approximated external force preserves structure, i.e., ensures it is derived from applying the Lagrange–D'Alembert principle to an approximated force $\tilde{\mathbf{f}}$ limited to variations in the reduced configuration space Q_r:

$$\tilde{\mathbf{f}}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r, t;\mu) \equiv \mathbf{V}^T\tilde{\mathbf{f}}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu). \tag{4.4}$$

Thus, this approximation amounts to computing $\tilde{\mathbf{f}}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu)$—an approximation to the vector-valued function $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu)$. That is, we assign no special mathematical properties to $\mathbf{f}$ aside from the fact that it is a vector. One way to accomplish this is by the DEIM/gappy POD approach described in Section 3.2.2.

The error in this approximation can be bounded using a result derived from the error in the gappy POD approximation of $\mathbf{f}$ (e.g., see [7, Appendix D]). We obtain

$$\|\tilde{\mathbf{f}}_r - \mathbf{f}_r\|_2 \leq \|\left(\mathbf{I} - \mathbf{W}_\mathbf{f}\left[\mathbf{P}^T\mathbf{W}_\mathbf{f}\right]^+\mathbf{P}^T\right)\mathbf{f}\|_2 \leq \|\mathbf{R}^{-1}\|_2\|\left(\mathbf{I} - \mathbf{W}_\mathbf{f}\mathbf{W}_\mathbf{f}^T\right)\mathbf{f}\|_2,$$

where $\mathbf{W}_\mathbf{f}$ is an orthogonal basis used to represent the external force, and $\mathbf{P}^T\mathbf{W}_\mathbf{f} = \mathbf{QR}$ is the thin QR matrix factorization. This result assumes that $\mathbf{P}^T\mathbf{W}_\mathbf{f}$ has full rank. Thus, the accuracy of this approximation relies both on the sampling matrix $\mathbf{P}^T$ and how close $\mathbf{f}$ is to the range of $\mathbf{W}_\mathbf{f}$. To achieve accuracy, we compute $\mathbf{W}_\mathbf{f}$ via POD, which minimizes the average value of $\|\left(\mathbf{I} - \mathbf{W}_\mathbf{f}\mathbf{W}_\mathbf{f}^T\right)\mathbf{f}\|_2^2$ over the training data. Procedure 1 provides the offline and online steps for implementing the external-force approximation.

Procedure 1: External-force approximation via gappy POD

Offline stage

1. Collect snapshots of the external force $\mathcal{X}_\mathbf{f} \equiv \{\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu) \mid \mu \in \{\mu^i\},\ t \in \mathsf{T}_{\text{sample}}(\mu)\}$
2. Compute a POD basis $\mathbf{W}_\mathbf{f}$ using Algorithm S1 with inputs $\mathcal{X}_\mathbf{f}$ and $\eta_\mathbf{f} \in [0,1]$.
3. Determine the sampling matrix $\mathbf{P}$.
4. Compute the low-dimensional matrix $\mathbf{Y}_\mathbf{f} = \mathbf{V}^T\mathbf{W}_\mathbf{f}\left[\mathbf{P}^T\mathbf{W}_\mathbf{f}\right]^+$.

Online stage (given $\mu^\star$)

1. Compute $m \ll N$ entries of the external force $\mathbf{P}^T\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu^\star)$.
2. Compute the low-dimensional matrix–vector product $\mathbf{Y}_\mathbf{f}\left[\mathbf{P}^T\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t;\mu^\star)\right]$.

4.4.1. Exactness conditions. Exactness conditions are similar to those that will be described in Section 5.2.3 for the matrix gappy POD approximation. In the general case where $m < N$, if $\mathbf{f} = 0$, then the approximation is exact, i.e., $\tilde{\mathbf{f}}_r = \mathbf{f}_r$. If instead $\mathbf{f}$ has at least one non-zero entry, then sufficient conditions for an exact approximation are 1) $\mathbf{f} \in \text{range}(\mathbf{W}_\mathbf{f})$ and 2) $\mathbf{P}^T\mathbf{W}_\mathbf{f}$ has full column rank. The first of these conditions holds, for example, when $\mathbf{W}_\mathbf{f}$ is computed via POD, the POD basis is not truncated, $\mathbf{f}$ is independent of $\mathbf{q}$ and $\dot{\mathbf{q}}$, $\mu^\star \in \{\mu^i\}$, and if a snapshot of the external force was collected at the considered time instance. The second of these can be enforced by the method for choosing $\mathbf{P}$. In the full-sampling case where $m = N$, condition 2 holds automatically, so we only require condition 1 in this case.

5. Preserving matrix symmetry and positive definiteness. This section presents approximation techniques for Lagrangian ingredient II (see Section 4.1) in an algebraic setting. Let $\mathbf{A}(\mu)$ denote an $N \times N$ parameterized symmetric positive-definite (possibly dense) matrix. We consider the following *online problem*:

(P1) At a cost independent of N, compute a symmetric positive-definite matrix $\tilde{\mathbf{A}}(\mu^\star)$ that is appropriately close to the matrix $\mathbf{V}^T\mathbf{A}(\mu^\star)\mathbf{V}$ for any online point $\mu^\star \in \mathcal{D}$.

Directly computing $\mathbf{V}^T\mathbf{A}(\mu^\star)\mathbf{V}$ is not a viable solution to this problem, as it requires computing all $\mathcal{O}(N^2)$ entries of the matrix $\mathbf{A}(\mu^\star)$ due to the density of $\mathbf{V}$. We assume only that computing a single entry of $\mathbf{A}(\mu^\star)$ for any online point $\mu^\star \in \mathcal{D}$ is inexpensive, i.e., the number of floating-point operations (flops) is independent of N. We do not assume affine parametric dependence, and we view $\mu \mapsto \mathbf{A}(\mu)$ simply as a mechanism for generating instances of the matrix $\mathbf{A}$.

We now present two methods for solving online problem (P1). Both rely on computing offline p 'snapshots' of the matrix $\mathbf{A}(\mu^i)$, $i = 1, \ldots, p$, where $\mu^i \in \mathcal{D}$ denotes the ith instance of the training set. Method 1 approximates the reduced matrix by projecting the full matrix onto a sparse basis, while Method 2 approximates the reduced matrix as a linear combination of pre-computed reduced matrices.

5.1. Reduced-basis sparsification (RBS). We first consider a strategy that 'injects sparseness' into the matrix $\mathbf{V}$. That is, we replace $\mathbf{V}$ by $\mathbf{U_A} \in \mathbb{R}_*^{N\times n}$, which has only m rows ($n \le m \ll N$) with nonzero entries:

$$\tilde{\mathbf{A}}(\mu) = \mathbf{U_A}^T\mathbf{A}(\mu)\mathbf{U_A}. \tag{5.1}$$

This sparse matrix may be expressed as $\mathbf{U_A} \equiv \mathbf{P}\underline{\mathbf{U_A}}$, where $\underline{\mathbf{U_A}} \in \mathbb{R}_*^{m\times n}$ is dense. Clearly, $\tilde{\mathbf{A}}(\mu)$ is symmetric positive definite if $\mathbf{A}(\mu)$ is symmetric positive definite, as $\mathbf{U_A}$ has full column rank (it is the product of full-column-rank matrices); thus, the approximation defined by (5.1) preserves the requisite structure. Note that this approximation will also preserve structure in cases where $\mathbf{A}(\mu)$ is symmetric positive semidefinite or simply symmetric. Further, the (online) operation count for computing $\tilde{\mathbf{A}}(\mu^\star)$ for online point $\mu^\star \in \mathcal{D}$ is independent of N. Computing $\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P}$ requires computing only m^2 (symmetric) entries of $\mathbf{A}(\mu^\star)$ and entails $\mathcal{O}(m^2)$ flops; subsequently computing $\tilde{\mathbf{A}}(\mu^\star) = \underline{\mathbf{U_A}}^T\left[\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P}\right]\underline{\mathbf{U_A}}$ entails $\mathcal{O}(m^2n + mn^2)$ flops.

Given a sampling matrix $\mathbf{P}$, the matrix $\underline{\mathbf{U_A}}$ can be computed offline to minimize the average approximation error over the snapshots, i.e., according to the following optimization problem:

$$\underline{\mathbf{U_A}} = \arg\min_{\mathbf{X}\in\mathbb{R}_*^{m\times n}} \sum_{i=1}^{p} \left\|\mathbf{X}^T\mathbf{P}^T\mathbf{A}(\mu^i)\mathbf{P}\mathbf{X} - \mathbf{V}^T\mathbf{A}(\mu^i)\mathbf{V}\right\|_F^2, \tag{5.2}$$

where the subscript F denotes the Frobenius norm. To handle the fact that $\mathbb{R}_*^{m\times n}$ is an open set, optimization problem (5.2) can first be solved over $\mathbb{R}^{m\times n}$ and the solution can be subsequently projected onto $\mathbb{R}_*^{m\times n}$, which is analogous to the approach taken by Vandereycken [27, Algorithm 6]. Note that problem (5.2) is a small-scale optimization problem that can be solved at a cost independent of N during the offline stage after the matrix snapshots $\mathbf{A}(\mu^i)$, $i = 1, \ldots, p$ and their reduced counterparts $\mathbf{V}^T\mathbf{A}(\mu^i)\mathbf{V}$, $i = 1, \ldots, p$ have been computed. Procedure 2 provides the offline and online steps required to implement the RBS approximation.

Procedure 2: Reduced-basis sparsification for symmetric matrices

Offline stage

1. Collect matrix snapshots $\mathbf{A}(\mu^i)$, $i = 1, \ldots, p$.
2. Form reduced the matrices $\mathbf{V}^T\mathbf{A}(\mu^i)\mathbf{V}$, $i = 1, \ldots, p$.
3. Choose the sample matrix $\mathbf{P}$.
4. Determine $\underline{\mathbf{U_A}}$ as the solution to problem (5.2).

Online stage (given $\mu^\star$)

1. Compute $\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P}$.
2. Form $\tilde{\mathbf{A}}(\mu^\star) = \underline{\mathbf{U_A}}^T\left[\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P}\right]\underline{\mathbf{U_A}}$.

REMARK 2. This work does not focus on methods for selecting the sampling matrix $\mathbf{P}$ for three reasons. First, the standard greedy approach [4, 9] to constructing the sample matrix works well for the proposed approximations. This is evidenced by the numerical experiments in Section 7, which employ the GNAT sample-mesh-based adaptation [7, Algorithm 3]. Second, this standard approach leads to a single

sample matrix, which is valid for all approximations. Using a unique sample matrix for each approximated term implies constructing a unique sample mesh for each term, which would complicate the implementation. Finally, the GNAT approach constructs the sample mesh using snapshots of the numerical residual across time steps and Newton iterations. This ensures that algorithm accounts for *all* terms in the equations of motion, as the residual is composed of contributions from every term. Future work entails tailoring the sampling matrix to the particular structure-preserving approximations.

5.1.1. Exactness conditions. In the full-sampling case where $m = N$, the approximation is exact if problem (5.2) is solved via a gradient-based method with an initial guess of $\mathbf{X}^{(0)} = \mathbf{P}^T\mathbf{V}$; we do this in practice. Under these conditions, $\mathbf{U}_\mathbf{A} = \mathbf{V}$ and so $\tilde{\mathbf{A}}(\mu) = \mathbf{V}^T\mathbf{A}(\mu)\mathbf{V}$.

In the general case where $m < N$, one can show that the approximation is exact if the matrix is parameter-independent (i.e., $\mathbf{A}(\mu) = \mathbf{A}$) and $m \geq n$. This situation is considered in the discussion that follows Theorem 5.1 below. One can also prove a more general exactness result when $\mathbf{A}(\mu)$ exhibits the following simple parametric dependence:

$$\mathbf{A}(\mu) = h_1(\mu)\ \mathbf{A}_1 + h_2(\mu)\ \mathbf{A}_2. \tag{5.3}$$

Here, $\mathbf{A}_1$ and $\mathbf{A}_2$ are $N \times N$ symmetric positive-definite matrices and $h_1, h_2 : \mathcal{D} \to \mathbb{R}$. It can then be shown that a sparse reduced basis exists that exactly captures $\mathbf{V}^T\mathbf{A}(\mu)\mathbf{V}$ under conditions related to how well the eigenvalues of the sampled matrix $\mathbf{P}^T\mathbf{A}(\mu)\mathbf{P}$ *encompass* those of the reduced matrix $\mathbf{V}^T\mathbf{A}(\mu)\mathbf{V}$. The following theorem uses eigenvalue interlacing ideas to make this notion precise.

THEOREM 5.1. *Let $\mathbf{A}(\mu)$ have the form given by Eq. (5.3). Then,*

$$\exists\ \underline{\mathbf{U}_\mathbf{A}} \in \mathbb{R}^{m\times n} \quad \textit{such that} \quad \underline{\mathbf{U}_\mathbf{A}}^T\,\mathbf{P}^T\,\mathbf{A}(\mu)\,\mathbf{P}\,\underline{\mathbf{U}_\mathbf{A}} = \mathbf{V}^T\,\mathbf{A}(\mu)\,\mathbf{V}, \quad \forall \mu \in \mathcal{D} \tag{5.4}$$

if and only if the generalized eigenvalues of $(\mathbf{V}^T\mathbf{A}_2\mathbf{V}, \mathbf{V}^T\mathbf{A}_1\mathbf{V})$ interlace the generalized eigenvalues of $(\mathbf{P}^T\mathbf{A}_2\mathbf{P}, \mathbf{P}^T\mathbf{A}_1\mathbf{P})$, i.e.,

$$\lambda_i^{(s)} \leq \lambda_i^{(r)} \leq \lambda_{i+m-n}^{(s)}, \quad i = 1, ..., n, \quad \textit{with} \tag{5.5}$$

$$\left[\mathbf{V}^T\mathbf{A}_2\mathbf{V}\right]\mathbf{x}_i^{(r)} = \lambda_i^{(r)}\left[\mathbf{V}^T\mathbf{A}_1\mathbf{V}\right]\mathbf{x}_i^{(r)}, \quad i = 1, \ldots n, \tag{5.6}$$

$$\left[\mathbf{P}^T\mathbf{A}_2\mathbf{P}\right]\mathbf{x}_i^{(s)} = \lambda_i^{(s)}\left[\mathbf{P}^T\mathbf{A}_1\mathbf{P}\right]\mathbf{x}_i^{(s)}, \quad i = 1, \ldots, m, \tag{5.7}$$

where the eigenvalues are sorted in order of increasing magnitude. Section S1.1 contains the proof, which relies on a generalization of the Cauchy interlacing theorem; the starting point is Theorem 4.3.10 of Ref. [18].

We now discuss the theorem's implications. When $\mathbf{A}$ is independent of μ, we can choose $h_1 = h_2 = 1$ and $\mathbf{A}_1 = \mathbf{A}_2$. The interlacing property is then trivially satisfied for $m = n$ with $\lambda_i^{(s)} = \lambda_i^{(r)} = 1$, and so the equality in (5.4) always holds. When $\mathbf{A}_1 \neq \mathbf{A}_2$ and m=~~n~~+1, the interlacing definition is restrictive, as it implies $\lambda_k^{(s)} \leq \lambda_k^{(r)} \leq \lambda_{k+1}^{(s)}$. We would not generally expect the eigenvalues of the sampled and reduced matrices to have this property. However, when $m \gg n+1$, each interval width is (much) larger and the condition is less restrictive. For example, for $n = 100$ and $m = 300$, interlacing implies that $\lambda_i^{(s)} \leq \lambda_i^{(r)} \leq \lambda_{i+200}^{(s)}$. Thus, we expect the conditions of the theorem to be satisfied for sufficiently large m, although this is not guaranteed. Future work includes extending Theorem 5.1 to a broader class of matrix parameterizations.

5.2. Matrix gappy POD. An alternative approximation applicable to online problem (P1) is

$$\tilde{\mathbf{A}}(\mu) = \sum_{i=1}^{n_\mathbf{A}} \xi_\mathbf{A}^i(\mu)\mathbf{V}^T\underline{\mathbf{A}}_i\mathbf{V}. \tag{5.8}$$

Here, the $N \times N$ symmetric matrices $\underline{\mathbf{A}}_i$, $i = 1, \ldots, n_\mathbf{A}$ are computed offline and define a basis for the matrix $\mathbf{A}(\mu)$. Due to the symmetry of $\underline{\mathbf{A}}_i$, $i = 1, \ldots, n_\mathbf{A}$, the approximation $\tilde{\mathbf{A}}(\mu)$ will always be symmetric. The parameter-dependent coefficients $\xi_\mathbf{A} \equiv \left(\xi_\mathbf{A}^1, \ldots, \xi_\mathbf{A}^{n_\mathbf{A}}\right)$ can be computed online in an efficient manner that ensures $\tilde{\mathbf{A}}(\mu)$ is positive definite and thereby preserves requisite structure; the next sections describe this.

We refer to this method as 'matrix gappy POD', as it amounts to the gappy POD procedure [13] applied to matrix data with modifications to preserve positive definiteness. The approach, which we originally proposed in Ref. [8], is a more general formulation of the 'matrix DEIM' approach [28] (or 'multi-component EIM' [26] in the context of the reduced-basis method), as it permits least-squares reconstruction (not simply interpolation). Further, it is equipped with a mechanism to maintain positive definiteness.

5.2.1. Offline computation: matrix basis. To obtain the matrix basis, we propose applying a vectorized POD method, wherein the basis can be considered a set of 'principal matrices' that optimally represent[5] the matrix $\mathbf{A}$ over the training set $\{\mu^i\}$. The (offline) steps for this method are as follows:

1. Collect matrix snapshots $\mathbf{A}\left(\mu^i\right)$, $i = 1, \ldots, p$.
2. Vectorize snapshots $\mathbf{a}^i \equiv \text{vec}\left(\mathbf{A}\left(\mu^i\right)\right) \in \mathbb{R}^{N^2}$, $i = 1, \ldots, p$; here, $\text{vec} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N^2}$ vectorizes a matrix by stacking its columns.
3. Compute an $n_{\mathbf{A}}$-dimensional (with $n_{\mathbf{A}} \leq p$) POD basis of the vectorized snapshots

$$\mathbf{W}_{\mathbf{a}} \equiv \left[\underline{\mathbf{a}}^1 \ \cdots \ \underline{\mathbf{a}}^{n_{\mathbf{A}}}\right] \in \mathbb{R}^{N^2\times n_{\mathbf{A}}} \tag{5.9}$$

   using vectorized snapshots $\{\mathbf{a}^i\}_{i=1}^p$ and an 'energy criterion' $\eta_{\mathbf{A}} \in [0, 1]$ as inputs to Algorithm S1.
4. Transform these POD basis vectors into their matrix counterparts:

$$\underline{\mathbf{A}}_i = \text{vec}^{-1}\left(\underline{\mathbf{a}}^i\right), \quad i = 1, \ldots, n_{\mathbf{A}}.$$

Each matrix $\underline{\mathbf{A}}_i$, $i = 1, \ldots, n_{\mathbf{A}}$ is guaranteed to be symmetric, as Algorithm S1 forms this basis by taking linear combinations of symmetric matrices.

5.2.2. Online computation: coefficients. The approximation error can be bounded as follows:

$$\|\mathbf{V}^T\mathbf{A}(\mu)\mathbf{V} - \sum_{i=1}^{n_{\mathbf{A}}} \xi^i_{\mathbf{A}}(\mu)\mathbf{V}^T\underline{\mathbf{A}}_i\mathbf{V}\|_F \leq \|\mathbf{V}\|_F^2 \left\|\mathbf{A}(\mu) - \sum_{i=1}^{n_{\mathbf{A}}} \xi^i_{\mathbf{A}}(\mu)\underline{\mathbf{A}}_i\right\|_F \tag{5.10}$$

where $\|\mathbf{V}\|_F^2 = n$ if $\mathbf{V}$ is orthogonal. This leads to a natural choice for the scalar coefficients based on minimizing the upper bound (5.10). In particular, we compute coefficients $\xi_{\mathbf{A}}(\mu^\star)$ online as the solution to

$$\begin{aligned} \underset{(x_1,\ldots,x_{n_{\mathbf{A}}})}{\text{minimize}} \quad & \|\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P} - \sum_{i=1}^{n_{\mathbf{A}}} x_i\mathbf{P}^T\underline{\mathbf{A}}_i\mathbf{P}\|_F^2 \\ \text{subject to} \quad & \sum_{i=1}^{n_{\mathbf{A}}} x_i\mathbf{V}^T\underline{\mathbf{A}}_i\mathbf{V} > 0. \end{aligned} \tag{5.11}$$

Note that the coefficients are computed to match (as closely as possible) the full matrix and the linear combination of pre-computed full matrices at a few entries. The constraints amount to a strict linear-matrix-inequality, where $\mathbf{A} > 0$ denotes a generalized inequality that indicates $\mathbf{A}$ is a positive-definite matrix. This constraint ensures that structure is preserved. Note that the constraint can be modified (resp. dropped) in cases where positive semidefiniteness (resp. symmetry) aims to be preserved.

Problem (5.11) is equivalent to a linear least-squares problem with nonlinear constraints; this can be seen from a vectorized form of the objective function:

$$\|\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P} - \sum_{i=1}^{n_{\mathbf{A}}} x_i\mathbf{P}^T\underline{\mathbf{A}}_i\mathbf{P}\|_F^2 = \|\left(\mathbf{P}^T\otimes\mathbf{P}^T\right)\text{vec}\left(\mathbf{A}(\mu^\star)\right) - \left(\mathbf{P}^T\otimes\mathbf{P}^T\right)\mathbf{W}_{\mathbf{a}}\mathbf{x}\|_2^2.$$

The objective function is equivalent to that of the gappy POD method [13]—which was discussed in Section 3.2.2—applied to matrix data. Note that this optimization problem is solved *online* using the online-sampled data $\mathbf{P}^T\mathbf{A}(\mu^\star)\mathbf{P}$; Section S2 describes a method for solving this optimization problem. In practice, we usually observe the constraints to be inactive at the unconstrained solution. Therefore, typically the constraints need not be handled directly, and solving problem (5.11) amounts to solving a small-scale linear least-squares problem characterized by an $(m^2+m)/2 \times n_{\mathbf{A}}$ matrix. To ensure a unique solution to problem (5.11), the matrix $\left(\mathbf{P}^T\otimes\mathbf{P}^T\right)\mathbf{W}_{\mathbf{a}}$ must have full column rank. This can be achieved by enforcing $(m^2+m)/2 \geq n_{\mathbf{A}}$ as well as mild conditions on the sampling matrix $\mathbf{P}$. Procedure 3 describes the offline and online stages for implementing the matrix gappy POD approximation.

[5]These matrices are optimal in the sense that they minimize the average projection error (as measured in the Frobenius norm) of the matrix snapshots.

Procedure 3: Matrix gappy POD

Offline stage

1 Compute the basis matrices $\underline{\mathbf{A}}_i$, $i = 1, \ldots, n_{\mathbf{A}}$ using the vectorized POD approach described in Section 5.2.1.
2 Determine the sampling matrix $\mathbf{P}$, which gives rise to a full-column-rank matrix $\left(\mathbf{P}^T \otimes \mathbf{P}^T\right) \mathbf{W}_{\mathbf{a}}$ with m chosen so that $(m^2 + m)/2 \geq n_{\mathbf{A}}$.
3 Compute low-dimensional matrices $\mathbf{V}^T \underline{\mathbf{A}}_i \mathbf{V}$, $i = 1, \ldots, n_{\mathbf{A}}$.
4 Retain the sampled entries of the matrix basis $\mathbf{P}^T \underline{\mathbf{A}}_i \mathbf{P}$, $i = 1, \ldots, n_{\mathbf{A}}$; discard other entries.

Online stage (given $\mu^\star$)

1 Compute $\mathbf{P}^T \mathbf{A}(\mu^\star) \mathbf{P}$.
2 Solve the small-scale optimization problem (5.11) for coefficients $\xi_{\mathbf{A}}(\mu^\star)$.
3 Assemble the low-dimensional matrix $\tilde{\mathbf{A}}(\mu^\star)$ by Eq. (5.8).

5.2.3. Exactness conditions. THEOREM 5.2. *The matrix gappy POD approximation is exact for any specified online parameters $\mu^\star \in \mathcal{D}$ if vec$(\mathbf{A}(\mu^\star)) \in$ range$(\mathbf{W}_{\mathbf{a}})$ and $\left(\mathbf{P}^T \otimes \mathbf{P}^T\right) \mathbf{W}_{\mathbf{a}}$ has full column rank.* See Section S1.2 for the proof. Condition 1 holds, e.g., when $\mu^\star \in \{\mu^i\}$ and $n_{\mathbf{A}} = p$. Condition 2 can be enforced by the choice of $\mathbf{P}$ and automatically holds in the case of full sampling, i.e., $m = N$.

6. Preserving potential-energy structure. This section presents a technique for approximating Lagrangian ingredient III (see Section 4.2) within an algebraic setting. Unlike the matrix approximations of the previous section, the nonlinear dependence on the potential energy on configuration variables $\mathbf{q}$ introduces additional challenges that must be considered carefully.

We aim to devise an *offline* method—which may entail expensive operations—for constructing a scalar-valued function $\tilde{V}_r : \mathbb{R}^n \times \mathcal{D} \to \mathbb{R}$. This function will be used *online* and should satisfy problem (P2):

(P2) Compute the gradient vector $\nabla_{\mathbf{q}_r} \tilde{V}_r(\mathbf{q}_r^\star; \mu^\star)$ at a cost independent of N. Given any online parameters $\mu^\star \in \mathcal{D}$, this vector should be appropriately close to $\mathbf{V}^T \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star) + \mathbf{V}\mathbf{q}_r^\star; \mu^\star)$ for all coordinates $\mathbf{q}_r^\star \in \mathbb{R}^n$.

Note that this problem aims to approximate the *gradient* of the function as opposed to the function itself.

In specialized cases, the above approximation can be simplified considerably. For example, when $\bar{\mathbf{q}}(\mu) = 0$, $\forall \mu \in \mathcal{D}$ and the function $V(\mathbf{q}; \mu)$ is purely quadratic in its first argument, then $\mathbf{V}^T \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r; \mu) = \mathbf{V}^T \mathbf{A}(\mu) \mathbf{V}\mathbf{q}_r$, where $\mathbf{A}(\mu)$ is a symmetric Hessian matrix; in this case, one of the approximation techniques described in Section 5 can be straightforwardly applied. Alternatively, if the potential energy is defined by the integral over a domain (i.e., $V(\mathbf{q}; \mu) = \int_\Omega \mathsf{V}(X, \mathbf{q}; \mu) d_{\Omega_X}$), a sparse cubature method '[2, 14] can be used to achieve computational efficiency and structure preservation. In more general cases, however, another approach is needed. In the following, we develop a method that makes no simplifying assumptions about the dependence of the potential V on the configuration variables or parameters.

Due to the density of the matrix $\mathbf{V}$, the most straightforward approach of setting $\tilde{V}_r(\mathbf{q}_r; \mu) = V(\bar{\mathbf{q}}(\mu) + \mathbf{V}\mathbf{q}_r; \mu)$ leads to expensive online operations: computing the gradient $\nabla_{\mathbf{q}_r} \tilde{V}_r(\mathbf{q}_r^\star; \mu^\star) = \mathbf{V}^T \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star) + \mathbf{V}\mathbf{q}_r^\star; \mu^\star)$ requires first computing all N entries of the gradient vector $\nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star) + \mathbf{V}\mathbf{q}_r^\star; \mu^\star)$. To rectify this, we revisit the RBS technique proposed in Section 5.1 and introduce some modifications. In particular, we replace $\mathbf{V}$ by a sparse *parameter-dependent* matrix $\mathbf{U}_V(\mu) \equiv \mathbf{P}\underline{\mathbf{U}_V}(\mu) \in \mathbb{R}_*^{N \times n}$ with only $m \ll N$ nonzero rows, where $\underline{\mathbf{U}_V}(\mu) \in \mathbb{R}_*^{m \times n}$ is a dense matrix. That is, we approximate the potential energy as

$$\tilde{V}_r(\mathbf{q}_r; \mu) \equiv V(\bar{\mathbf{q}}(\mu) + \mathbf{U}_V(\mu)\mathbf{q}_r; \mu). \tag{6.1}$$

This approximation preserves structure, as $\tilde{V}_r$ remains a parameterized scalar-valued function. Now, we wish to compute $\mathbf{U}_V$ such that $\nabla_{\mathbf{q}_r} \tilde{V}_r(\mathbf{q}_r^\star; \mu^\star) = \mathbf{U}_V(\mu^\star)^T \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star) + \mathbf{U}_V(\mu^\star)\mathbf{q}_r^\star; \mu^\star)$ is as close as possible to $\mathbf{V}^T \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star) + \mathbf{V}\mathbf{q}_r^\star; \mu^\star)$ for any online point $\mu^\star \in \mathcal{D}$ and any $\mathbf{q}_r^\star \in \mathbb{R}^n$. One can imagine a variety of methods for computing $\mathbf{U}_V$ toward this stated goal. For example, one can formulate an optimization problem to match the potential gradient at training points [8]; this effectively leads to a parameter-independent matrix

$\mathbf{U}_V$. However, we found that this approach led to significant errors for many problems, especially as the problem size and response nonlinearity increased. Instead, we pursue an idea motivated by the upcoming analysis in Section 6.1, which considers the first two terms in a Taylor expansion of $\mathbf{V}^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star)+\mathbf{V}\mathbf{q}_r^\star;\mu^\star)$ about the reference configuration.

In practice, we often find that the trajectories of dynamical systems are localized in the configuration space. This is particularly true for mechanical oscillators encountered in structural dynamics, where the trajectory does not deviate drastically from equilibrium. Using this observation, we focus our approximation efforts on accurately capturing the behavior of the potential in a neighborhood of the online reference configuration $\bar{\mathbf{q}}(\mu^\star)$. Implicitly, this assumes that the online configurations do not greatly diverge from this point. To this end, consider computing $\mathbf{U}_V(\mu^\star)$ online such that the approximation $\mathbf{U}_V(\mu^\star)^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star)+\mathbf{U}_V(\mu^\star)\mathbf{q}_r^\star;\mu^\star)$ matches $\mathbf{V}^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star)+\mathbf{V}\mathbf{q}_r^\star;\mu^\star)$ to first order about the reference configuration:

$$\begin{aligned}&\mathbf{U}_V(\mu^\star)^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)+\mathbf{U}_V(\mu^\star)^T\nabla_{\mathbf{qq}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)\mathbf{U}_V(\mu^\star)\mathbf{q}_r^\star\\&=\mathbf{V}^T\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)+\mathbf{V}^T\nabla_{\mathbf{qq}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)\mathbf{V}\mathbf{q}_r^\star,\quad\forall\mathbf{q}_r^\star\in\mathbb{R}^n.\end{aligned}\tag{6.2}$$

Notice that the high-order terms amount to approximating a reduced Hessian (defined via the dense matrix $\mathbf{V}$) by a second reduced Hessian (defined via the sparse matrix $\mathbf{U}_V(\mu^\star)$). This is equivalent to online problem (P1) presented in Section 5 that was addressed by the RBS algorithm (as well as a matrix gappy POD approach). This RBS algorithm is supported by Theorem 5.1, which shows that an exact approximation of the reduced Hessian is possible under certain assumptions. While these assumptions do not always hold, the theorem gives an expectation that a good approximation can be found under more general circumstances. Unfortunately, the presence of the low-order terms in Eq. (6.2) alters the character of the reduced approximation and so Theorem 5.1 no longer applies. In this case, the matrix $\mathbf{U}_V(\mu^\star)$ must serve to capture both gradient and Hessian information simultaneously, which introduces restrictive assumptions in order to obtain an equivalent result to Theorem 5.1; this will be shown later in Lemma 1.

To avoid the limitations associated with these restrictions, we set the reference configuration to equilibrium, i.e., $\bar{\mathbf{q}}(\mu)=\mathbf{q}_0(\mu)$ with equilibrium defined as $\nabla_{\mathbf{q}}V(\mathbf{q}_0(\mu);\mu)=0$. This forces the low-order Taylor terms to zero and simplifies Eq. (6.2) to

$$\mathbf{U}_V(\mu^\star)^T\nabla_{\mathbf{qq}}V(\mathbf{q}_0(\mu^\star);\mu^\star)\mathbf{U}_V(\mu^\star)=\mathbf{V}^T\nabla_{\mathbf{qq}}V(\mathbf{q}_0(\mu^\star);\mu^\star)\mathbf{V}.\tag{6.3}$$

Now, Theorem 5.1 holds, implying that Equation (6.3) can be exactly solved when $m=n$. For this reason, we compute $\mathbf{U}_V(\mu^\star)$ online to satisfy (6.3) using n sample indices. Specifically, we compute it according to

$$\underline{\mathbf{U}_V}(\mu^\star)=\begin{bmatrix}\mathbf{X}\\\mathbf{0}_{(m-n)\times n}\end{bmatrix},\tag{6.4}$$

where $\mathbf{X}$ is given by solving $\mathbf{L}_1^T\mathbf{X}=\mathbf{L}_2^T$. Here, $\mathbf{L}_2\in\mathbb{R}^{n\times n}$ denotes the lower-triangular Cholesky factor of $\mathbf{V}^T\nabla_{\mathbf{qq}}V(\mathbf{q}_0(\mu^\star);\mu^\star)\mathbf{V}$, $\mathbf{L}_1\in\mathbb{R}^{n\times n}$ denotes the lower-triangular Cholesky factor of ${\mathbf{P}_1}^T\nabla_{\mathbf{qq}}V(\mathbf{q}_0(\mu^\star);\mu^\star)\mathbf{P}_1$, and $\mathbf{P}_1$ represents the first n columns of $\mathbf{P}$. We defer discussing the computational cost for this approach to Section 6.2, and now return to the previously alluded difficulties associated with solving (6.2) when the reference configuration does not correspond to equilibrium.

Remark 3. For problems where the potential V does not have a stationary point, there is no equilibrium configuration. As such, a different approach to compute $\underline{\mathbf{U}_V}(\mu^\star)$ is required; however such an approach will not guarantee first-order consistency (6.2). For example, if $m>p$, then $\underline{\mathbf{U}_V}(\mu^\star)$ could be computed online as the solution to the optimization problem

$$\begin{aligned}&\underset{\mathbf{X}\in\mathbb{R}_*^{m\times n}}{\text{minimize}}\quad\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij}|\\&\text{subject to}\quad\begin{bmatrix}\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)^T\\\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^1);\mu^1)^T\\\vdots\\\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^p);\mu^p)^T\end{bmatrix}\mathbf{P}\mathbf{X}=\begin{bmatrix}\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)^T\\\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^1);\mu^1)^T\\\vdots\\\nabla_{\mathbf{q}}V(\bar{\mathbf{q}}(\mu^p);\mu^p)^T\end{bmatrix}\mathbf{V},\end{aligned}\tag{6.5}$$

where x_{ij} denotes the (i,j) entry of matrix $\mathbf{X}$.

6.1. Solvability of the two-term Taylor equation. The method presented in the previous section was motivated by difficulties in inexpensively approximating the reduced gradient of a nonlinear function. In this section, we provide insight into these difficulties by investigating a much simpler situation: the solvability of the two-term Taylor Eq. (6.2), which we write in matrix/vector form as

$$\underline{\mathbf{U}_V}^T\mathbf{P}^T\mathbf{c} + \underline{\mathbf{U}_V}^T\mathbf{P}^T\mathbf{A}\mathbf{P}\underline{\mathbf{U}_V}\mathbf{q}_r^\star = \mathbf{V}^T\mathbf{c} + \mathbf{V}^T\mathbf{A}\mathbf{V}\mathbf{q}_r^\star, \quad \forall \mathbf{q}_r^\star \in \mathbb{R}^n. \tag{6.6}$$

Here, we have set $\mathbf{c} = \nabla_{\mathbf{q}} V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)$ and $\mathbf{A} = \nabla_{\mathbf{qq}} V(\bar{\mathbf{q}}(\mu^\star);\mu^\star)$. We have also dropped dependence on $\mu^\star$ such that $\mathbf{c}$ and $\mathbf{A}$ are parameter independent; this is equivalent to restricting Eq. (6.2) to a single instance of $\mu^\star$. This is not ideal, as we would typically wish to minimize online costs by computing a single $\underline{\mathbf{U}_V}$ offline that is valid for *all* subsequent online calculations. However, we now show that it is not always possible to satisfy Eq. (6.2), even when one is restricted to finding a $\underline{\mathbf{U}_V}$ for a single instance of $\mu^\star$.

As Eq. (6.6) must hold for all $\mathbf{q}_r^\star$, we have the following necessary and sufficient conditions:

$$\underline{\mathbf{U}_V}^T\mathbf{P}^T\mathbf{A}\mathbf{P}\underline{\mathbf{U}_V} = \mathbf{V}^T\mathbf{A}\mathbf{V} \quad \text{and} \quad \underline{\mathbf{U}_V}^T\mathbf{P}^T\mathbf{c} = \mathbf{V}^T\mathbf{c}. \tag{6.7}$$

It is possible to show that satisfying these conditions is equivalent to finding a $\widetilde{\mathbf{U}_V} \in \mathbb{R}^{m\times n}$ such that

$$\widetilde{\mathbf{U}_V}^T\widetilde{\mathbf{U}_V} = I \quad \text{and} \quad \widetilde{\mathbf{U}_V}^T\mathbf{P}^T\tilde{\mathbf{c}} = \widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}, \tag{6.8}$$

where $\widetilde{\mathbf{V}}^T\widetilde{\mathbf{V}} = I$. The definitions of $\widetilde{\mathbf{U}_V}, \widetilde{\mathbf{V}}$, and $\tilde{\mathbf{c}}$ are given below. The key point is that the necessary and sufficient conditions for Eq. (6.8) amount to finding an orthogonal matrix $\widetilde{\mathbf{U}_V}$ such that $\widetilde{\mathbf{U}_V}^T\mathbf{P}^T\tilde{\mathbf{c}} = \widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}$ for a given orthogonal matrix $\widetilde{\mathbf{V}}$ and a given vector, $\tilde{\mathbf{c}}$. In the simple case when $\mathbf{A} = \mathbf{I}$ and $\mathbf{V}$ is orthogonal, we have $\widetilde{\mathbf{U}_V} = \underline{\mathbf{U}_V}$ and $\widetilde{\mathbf{V}} = \mathbf{V}$. More generally, we have the following definitions:

$$\widetilde{\mathbf{U}_V} = \mathbf{P}^T\mathbf{L}^T\mathbf{P}\underline{\mathbf{U}_V}\mathbf{L}_\phi^{-T}, \quad \widetilde{\mathbf{V}} = \mathbf{L}^T\mathbf{V}\mathbf{L}_\phi^{-T}, \quad \tilde{\mathbf{c}} = \mathbf{L}^{-1}\mathbf{c},$$

where $\mathbf{L}$ is the lower-triangular Cholesky factor of $\mathbf{A}$, and $\mathbf{L}_\phi$ is the lower-triangular Cholesky factor of $\mathbf{V}^T\mathbf{A}\mathbf{V}$. The above equivalence relies on the identities $\mathbf{P}^T\mathbf{L}\mathbf{P}\mathbf{P}^T\mathbf{L}^T\mathbf{P} = \mathbf{P}^T\mathbf{A}\mathbf{P}$ and $\mathbf{P}^T\mathbf{L}\mathbf{P}\mathbf{P}^T\mathbf{L}^{-1} = \mathbf{P}^T$. These hold due to the lower-triangular form of the matrix $\mathbf{L}$.

The following lemma addresses the conditions under which Eq. (6.8) (equivalently Eq. (6.6)) holds.

LEMMA 1. *Consider the equations*

$$\widetilde{\mathbf{U}_V}^T\widetilde{\mathbf{U}_V} = I \quad \textit{and} \quad \widetilde{\mathbf{U}_V}^T\mathbf{P}^T\tilde{\mathbf{c}} = \widetilde{\mathbf{V}}^T\tilde{\mathbf{c}} \tag{6.9}$$

with the following matrices given: $\mathbf{P} \in \{0,1\}^{N\times m}$ *consists of selected columns of the identity matrix (see prior definition),* $\widetilde{\mathbf{V}} \in \mathbb{R}^{N\times n}$ *with* $\widetilde{\mathbf{V}}^T\widetilde{\mathbf{V}} = I$*, and* $\tilde{\mathbf{c}} \in \mathbb{R}^{N\times 1}$*. Then, assuming that* $m \geq n$*, some* $\widetilde{\mathbf{U}_V} \in \mathbb{R}^{m\times n}$ *exists such that Eq.* (6.9) *is satisfied if and only if*

$$||\widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 = ||\mathbf{P}^T\tilde{\mathbf{c}}||_2 \;\; \textit{and} \;\; m = n \quad \textit{or} \quad ||\widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 \leq ||\mathbf{P}^T\tilde{\mathbf{c}}||_2 \;\; \textit{and} \;\; m > n. \tag{6.10}$$

See Section S1.3 for the proof.

Eq. (6.10) is satisfied if either $\tilde{\mathbf{c}} = 0$ (i.e., the equilibrium configuration is taken as the reference configuration) or if $m = N$. Unfortunately, however, Eq. (6.10) is not guaranteed to be satisfiable in more general situations. Specifically, when $m > n$, the condition $||\widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 \leq ||\mathbf{P}^T\tilde{\mathbf{c}}||_2$ corresponds to comparing the magnitude of a vector of length n obtained by rotating and dropping components with a second vector of length m obtained by simply dropping components. If $\widetilde{\mathbf{V}}$ and $\tilde{\mathbf{c}}$ are not correlated, then one could perhaps hope that on average the vector with more components would generally have a larger magnitude. However, when $\tilde{\mathbf{c}}$ lies completely within the subspace spanned by the columns of $\widetilde{\mathbf{V}}$ and all components of $\tilde{\mathbf{c}}$ are non-zero, then $||\widetilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 = ||\tilde{\mathbf{c}}||_2$ and so satisfying the necessary and sufficient conditions requires 'full sampling' $m = N$. While this scenario may be considered pessimistic, one can expect that a very large value of m will be required when $\tilde{\mathbf{c}}$ lies primarily in the range space of $\widetilde{\mathbf{V}}$. In general, there is no guarantee that even the simplified (i.e., parameter-independent) form of the two-term Taylor equation is solvable. When one also considers that Eq. (6.6) corresponds to the restriction of Eq. (6.2) to a single instance of $\mu^\star$, the above result should be seen as quite discouraging.

Therefore, this analysis supports the approach proposed earlier in the section, i.e., setting the reference configuration to equilibrium—which results in $\tilde{\mathbf{c}} = 0$—and computing a parameter-dependent matrix $\underline{\mathbf{U}_V}(\mu^\star)$ via Eq. (6.4), which is valid only for a single online point $\mu^\star$ but can be used for all reduced configuration variables $\mathbf{q}_r^\star \in \mathbb{R}^n$ that may arise during the online evaluation. This approach avoids the difficulties introduced by a nonzero $\tilde{\mathbf{c}}$ and guarantees solvability of the two-term Taylor expression with $m = n$.

6.2. Implementation and cost. Procedure 4 summarizes the offline/online strategy for implementing the RBS strategy for approximating the potential energy. This method satisfies the online computational

Procedure 4: Reduced-basis sparsification for potential energy

Offline stage

1 Determine the sampling matrix $\mathbf{P}$.

Online stage (given $\mu^\star$)

1 Compute $\mathbf{V}^T \nabla_{\mathbf{qq}} V\left(\mathbf{q}_0\left(\mu^\star\right); \mu^\star\right) \mathbf{V}$.
2 Compute ${\mathbf{P}_1}^T \nabla_{\mathbf{qq}} V\left(\mathbf{q}_0\left(\mu^\star\right); \mu^\star\right) \mathbf{P}_1$.
3 Solve Equation (6.4) for $\underline{\mathbf{U}_V}(\mu^\star)$.
4 For any $\mathbf{q}_r^\star \in \mathbb{R}^n$, set $\tilde{V}_r(\mathbf{q}_r^\star; \mu^\star) = V(\mathbf{q}_0(\mu^\star) + \mathbf{P}\underline{\mathbf{U}_V}(\mu^\star)\mathbf{q}_r^\star; \mu^\star)$, and compute the gradient as $\nabla_{\mathbf{q}_r} \tilde{V}_r(\mathbf{q}_r^\star; \mu^\star) = \underline{\mathbf{U}_V}(\mu^\star)^T \mathbf{P}^T \nabla_q V(\mathbf{q}_0(\mu^\star) + \mathbf{P}\underline{\mathbf{U}_V}(\mu^\star)\mathbf{q}_r^\star; \mu^\star)$

cost requirements of problem (P2) with one exception: online step 1 incurs an N-dependent operation count. However, online steps 1–3 depend only on the online point $\mu^\star$ and not on the reduced configuration variables $\mathbf{q}_r^\star$. Thus, these steps are performed only once per parameter instance, and their cost can be amortized over all online-queried values of $\mathbf{q}_r^\star$. As a result, this does not preclude computational savings, as will be shown in the numerical results reported in Section 7. Note that online step 2 is equivalent to computing just $\mathcal{O}(n^2)$ entries of $\nabla_{\mathbf{qq}} V$, which can be completed at a cost independent of N. Step 3 requires $\mathcal{O}(n^3)$ operations.

REMARK 4. Most nonlinear reduced-order modeling methods [3, 25, 9, 15, 11, 6, 7] assume 'H-independence' [11], which states that the Jacobian of the vector-valued nonlinear function is sparse; in the present context, this corresponds to sparsity of the matrix $\nabla_{\mathbf{qq}} V$. When this assumption holds, the proposed methodology incurs low online computational cost. This efficiency results from the fact that computing $\mathbf{P}^T \nabla_{\mathbf{q}} V(\mathbf{q}_0(\mu^\star) + \mathbf{P}\underline{\mathbf{U}_V}(\mu^\star)\mathbf{q}_r^\star; \mu^\star)$ in Step 4 of Procedure 4 requires that only m components of the gradient $\nabla_{\mathbf{q}} V$ be evaluated; if H-independence holds, then these m components depend on only $\mathcal{O}(m)$ components of the argument $\mathbf{q}_0(\mu^\star) + \mathbf{P}\mathbf{X}\mathbf{q}_r$, leading to an N-independent operation count.

Unfortunately, H-independence does not hold for some problems in Lagrangian dynamics. For example molecular-dynamics models can be characterized by a potential that includes interaction terms between all particles, resulting in a dense matrix $\nabla_{\mathbf{qq}} V$. Here, the proposed method can still achieve efficiency by 'centering' the configuration space at equilibrium such that $\mathbf{q}_0(\mu) = 0$, $\forall \mu \in \mathcal{D}$. In this case, the method requires computing only m components of the argument $\mathbf{q}_0(\mu^\star) + \mathbf{P}\mathbf{X}\mathbf{q}_r$ in Step 4 of Procedure 4 regardless of the sparsity of the matrix $\nabla_{\mathbf{qq}} V$. This efficiency is achievable due to the fact that the method injects 'sparsification' in the argument of the nonlinear function. This ability to achieve an N-independent operation count when H-independence is violated distinguishes this method from others in the literature.

7. Numerical experiments. Although the Galerkin and proposed reduced-order models have a *theoretical* advantage over the gappy POD and collocation reduced-order models in terms of preserving Lagrangian structure, it is unclear *a priori* if this translates to improved numerical results in practice. This section investigates this question by applying the model-reduction techniques to a practical problem: the clamped–free truss structure shown in Figure 7.1.

We set the material properties to those of aluminum, i.e., density $\rho = 2700$ kg/m^3 and elastic modulus $E = 62 \times 10^9$ Pa. The external force is composed of four components:

$$\mathbf{f}(\mu, t) = \sum_{i=1}^{4} r_i(\mu, t)\mathbf{r}_i,$$

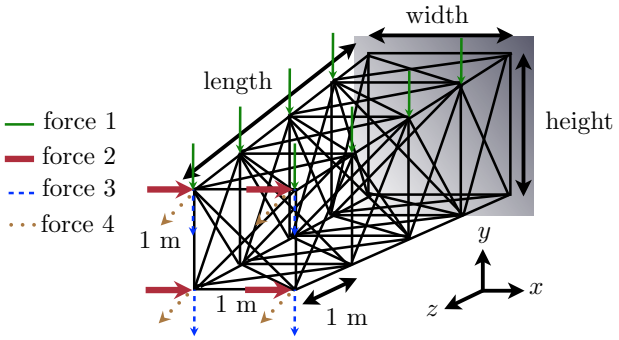


FIG. 7.1. *Clamped–free parameterized truss structure*

where $\mathbf{r}_i \in \mathbb{R}^N$, $i = 1, \ldots, 4$ correspond to unit loads uniformly distributed across designated nodes and $r_i : \mathcal{D} \times [0, \mathsf{T}] \to \mathbb{R}$, $i = 1, \ldots, 4$ denote the component-force magnitudes. Figure 7.1 depicts the spatial distribution of the forces, which lead to vectors $\mathbf{r}_i$, $i = 1, \ldots, 4$ through the finite-element formulation described below. The parameterized, time-dependent magnitudes of these forces are

$$r_i(\mu, t) = \begin{cases} \gamma_i(\mu) \sin\left(\lambda_i(\mu)\left(t - \mathsf{T}/4\right)\right), & t \geq \mathsf{T}/4 \\ 0, \quad \text{otherwise} \end{cases},$$

where $\gamma_i : \mathcal{D} \to \mathbb{R}$ and $\lambda_i : \mathcal{D} \to \mathbb{R}$, $i = 1, \ldots, 4$ denote the maximum force magnitudes and force frequencies, respectively. Similarly, the initial condition is composed of four components $\mathbf{q}(0; \mu) = \sum_{i=1}^{4} s_i(\mu)\mathbf{s}_i$, where $\mathbf{s}_i$ is the steady-state displacement of the truss subjected to load $\mathbf{r}_i\gamma_i(\bar{\mu})$ with $\bar{\mu} = (0, \ldots, 0)$ denoting the nominal point in parameter space. The equilibrium configuration is simply the undeformed truss represented by $\mathbf{q}_0(\mu) = 0$; thus, the configuration space is centered at equilibrium.

The truss is parameterized by 16 parameters $\mu \equiv (\mu_1, \ldots, \mu_{16}) \in [-1, 1]^{16}$ that affect the geometry, initial condition, and applied force as described in Table 7.1.

length (m)	bar cross-sectional area (m^2)	width (m)	height (m)	initial condition max magnitude (N) s_i	external-force magnitude γ_i	external-force frequency λ_i
$200 + 50\mu_1$	$0.0025(1 + 0.5\mu_2)$	$10(1 + \mu_3)$	$10(1 + \mu_4)$	$\underline{f}_i(1 + 0.5\mu_{i+4})$	$\underline{f}_i(1 + 0.5\mu_{i+8})$	$3\omega_0(1 + 0.5\mu_{i+12})$

TABLE 7.1

Effect of parameters on truss geometry, initial conditions, and applied forces. Here, $\underline{f}_i$, $i = 1, \ldots, 4$ denote the nominal force magnitudes (specified within each experiment) and ω_0 denotes the lowest-magnitude eigenvalue at the nominal point $\bar{\mu}$.

The problem is discretized by the finite-element method. The model consists of 16 three-dimensional bar elements per bay with three degrees of freedom per node; this results in 12 degrees of freedom per bay. We consider a problem with 250 bays, and therefore $N = 3 \times 10^3$ degrees of freedom in the full-order model. The bar elements model geometric nonlinearity, which causes a high-order nonlinearity in the potential energy V. This discretization leads to a Lagrangian-dynamical-system model with configuration manifold $Q = \mathbb{R}^N$, Riemannian metric $g(\mathbf{v}, \mathbf{w}; \mu) = \mathbf{v}^T\mathbf{M}(\mu)\mathbf{w}$, nonlinear potential-energy function V, and dissipation function $\mathcal{F}(\dot{\mathbf{q}}; \mu) \equiv \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{C}(\mu)\dot{\mathbf{q}}$. Here, $\mathbf{C}(\mu) = \alpha\mathbf{M}(\mu) + \beta\nabla_{\mathbf{qq}}V(0; \mu)$ corresponds to Rayleigh damping. Here, α and β are chosen such that the damping ratio is a specified value ζ for the uncoupled ODEs associated with the smallest two eigenvalues of the matrix pencil $(\mathbf{M}(\bar{\mu}), \nabla_{\mathbf{qq}}V(0; \bar{\mu}))$ [10].

To numerically solve the Lagrangian equations of motion in the time interval $[0, \mathsf{T}]$ with $\mathsf{T} = 25$ seconds, we employ the implicit midpoint rule (a symplectic integrator). This ensures that the numerical solution will yield symplectic time-evolution maps in the conservative case. We employ a globalized Newton solver with a More–Thuente linesearch [12] to solve the system of nonlinear algebraic equations arising at each time step. Convergence of the Newton iterations is declared when the residual norm reaches 10^{-6} of its value computed using a zero acceleration and the values of the displacement and velocity at the beginning of the timestep. The linear system arising at each Newton iteration is solved directly.

The experiments compare the performance of four reduced-order models: Galerkin projection (Section 3.1), collocation (Section 3.2.1), and gappy POD (Section 3.2.2), and the proposed structure-preserving methods. All reduced-order models (ROMs) employ the same POD reduced basis $\mathbf{V}$, which is computed by applying Algorithm S1 with snapshots of the configuration variables and an energy criterion $\eta \leftarrow \eta_{\mathbf{q}} \in [0,1]$ specified within each experiment. The POD bases $\mathbf{W}_{\boldsymbol{\theta}}$ employed by the gappy POD approach (see Section 3.2.2) are generated in the same way. In all cases, snapshots are only collected for the first half of the time interval at the training points; as a result, the second half of the time interval can be considered predictive—even for the training set.

Reduced-order models with complexity reduction employ the same sampling matrix $\mathbf{P}$, which is generated using GNAT's greedy sample-mesh algorithm [7, Algorithm 3].[6] These models are also implemented using the sample-mesh concept [7, Section 5]. To solve optimization problems (5.2), we use the Poblano toolbox for unconstrained optimization [12]. The initial guess for each of these problems is chosen as $\mathbf{P}^T\mathbf{P}\mathbf{V}$. In practice, we always found the constraints to be inactive at the unconstrained solution to (5.11); therefore, this reduces to a linear least-squares problem that we solve directly.

To compare the performance of the reduced-order models, we will consider the response quantity of interest to be the y-displacement of the bottom-left node of the end face of the truss in Figure 7.1; we denote this (parameterized, time-dependent) quantity by $y \in \mathbb{R}$. The reported errors will be a normalized 1-norm (in time) of the error in this quantity:

$$\text{error} = \frac{\sum_{t \in \mathsf{T}_{\text{sample}}(\mu^\star)} |y_{\text{ROM}}(t;\mu^\star) - y_{\text{HFM}}(t;\mu^\star)|}{|\mathsf{T}_{\text{sample}}(\mu^\star)|(\max_{t \in \mathsf{T}_{\text{sample}}(\mu^\star)} y_{\text{HFM}}(t;\mu^\star) - \min_{t \in \mathsf{T}_{\text{sample}}(\mu^\star)} y_{\text{HFM}}(t;\mu^\star))}. \tag{7.1}$$

Here, y_{ROM} denotes the response computed by a reduced-order model, y_{HFM} is the high-fidelity 'truth' response, and $\mathsf{T}_{\text{sample}}(\mu^\star) \subset [0,\mathsf{T}]$ denotes the time instances selected by the time integrator for online point $\mu^\star$.[7] In addition to the error in Eq. 7.1, we will compare the speedup achieved by the reduced-order models, measured as the ratio of the reduced-order-model simulation time to the full-order-model simulation time. All computations are carried out in Matlab on a Mac Pro with 2 × 2.93 GHz 6-Core Intel Xeon processors and 64 GB of memory. Section S4 contains supplementary plots for these experiments.

7.1. Conservative case. We first consider the conservative case characterized by zero damping $\zeta = 0$ and no external forces $\mu_i = -2$ for $i = 9,\dots,16$. As a result, we are free to vary parameters μ_i, $i = 1,\dots,8$ that affect only the geometry and initial condition. We set the nominal forces that affect the initial condition to $\underline{f}_1 = \underline{f}_2 = 2\text{kg}\times 9.81\text{m/s}^2$ and $\underline{f}_3 = \underline{f}_4 = 0.4\text{kg}\times 9.81\text{m/s}^2$. This scenario is particularly interesting, as the full-order model corresponds to a conservative Lagrangian-dynamical system with energy conservation and symplectic time-evolution maps. Because we numerically solve the equations of motion using the (symplectic) implicit midpoint rule, the numerical solution is also characterized by a symplectic time-evolution map. This will also hold for reduced-order models that preserve Lagrangian structure, i.e., the Galerkin reduced-order model and the two proposed techniques. Note also that the dynamics of undamped, unforced structures are typically quite stiff, which often leaves reduced-order models prone to instabilities.

We first perform a timestep-verification study for the nominal point $\bar{\mu}$ characterized by $\bar{\mu}_i = 0$, $i = 1,\dots 8$ to ensure we employ an appropriate timestep. A timestep size of $\Delta t = 0.008$ seconds yields an observed convergence rate in the time-averaged tip displacement of 1.98, which is close to the asymptotic rate of convergence of the implicit midpoint rule, and an approximated error in the time-averaged tip displacement using Richardson extrapolation of 5.16×10^{-7}. We can therefore declare this to be an appropriate timestep size for the numerical experiments. Further, we note that the average number of Newton iterations per timestep is 3.15, so the geometric nonlinearity is significant.

[6] Greedy-algorithm parameters are $\Phi_R = \Phi_J = \mathbf{W}_{\mathbf{r}}$ a POD basis computed using Algorithm S1 with snapshots of the numerical residual over all timesteps and Newton iterations during the full-order-model training simulations and an energy criterion of $\eta \leftarrow \eta_{\mathbf{r}} = 1 - 10^{-2}$, a target number of sample nodes $n_s = m/\nu$ with $\nu = 3$ unknowns per node (the x-, y-, and z-displacements), an empty seeded sample-node set $\mathcal{N} = \emptyset$, and n_c equal to the number of columns in $\mathbf{W}_{\mathbf{r}}$.

[7] We employ this error measure because it is insensitive to shifts in the average value of the displacement, unlike other measures such as the average 1-norm.

7.1.1. Fixed parameters. We now test the model-reduction techniques in a fixed-parameters scenario. That is, we employ the nominal point in the parameter space for both the training and online points: $\{\mu^i\} = \bar{\mu}$ and $\mu^\star = \bar{\mu}$. Recall that we only collect snapshots for the first half of the time interval, so the second half can be considered a predictive regime. Note that the two proposed structure-preserving methods are the same for this case: they both exactly approximate the reduced mass matrix when the parameters are fixed.

The POD reduced basis $\mathbf{V}$ is generated using an energy criterion of $\eta_{\mathbf{q}} = 1 - 10^{-5}$ in Algorithm S1; this leads to a basis dimension of only $n = 11 \ll N$. The gappy POD-based reduced-order model employs an energy criterion of 1 (i.e., no truncation) for its reduced bases $\mathbf{W}_{\boldsymbol{\theta}}$ (see Section 3.2.2). Figure 7.2 reports results for the reduced-order models as the number of sample indices varies.[8]

First, note that the Galerkin reduced-order model is accurate (relative error of 5.42%); however, it yields a speedup of only 1.69. This is to be expected, as it preserves Lagrangian structure, but has no complexity-reduction mechanism (see Section 3.1.1). In addition, the proposed reduced-order model—which also preserves structure, yet has a complexity-reduction mechanism—yields a stable and accurate response regardless of the number of sample points chosen. For example, 0.4% sampling yields a relative error of 7.3% and a speedup of 207.0. Sampling 2% of the indices yields an error of 0.71% and a speedup of 34.5, and sampling 5% of the indices leads to 0.48% error and a speedup of 15.7. Note that sampling beyond 5% does not improve the method's accuracy; however, it degrades the speedup, as it requires computing more entries of the vector-valued functions.

The other complexity-reducing reduced-order models (gappy POD and collocation) are *always* unstable except for collocation in the full-sampling case, when it is equivalent to Galerkin. This clearly highlights the practical benefits of preserving structure in model reduction, as existing structure-destroying complexity-reduction methods failed, even in the relatively simple scenario of fixed parameter values.[9]

7.1.2. Varying parameters. We now consider a fully predictive scenario with $\mu^\star \notin \{\mu^i\}$. We use $p = 6$ training points and determine $\{\mu^i\}$ using Latin hypercube sampling [22]. Three online points are chosen randomly. Figure 7.3 depicts the tip displacement for the training points. Note that the responses are significantly different from one another. The two proposed structure-preserving reduced-order models will now be different from one another, as the parameters are varying, which means that the parameterized mass matrix will be approximated differently by the two techniques (see Table 4.1).

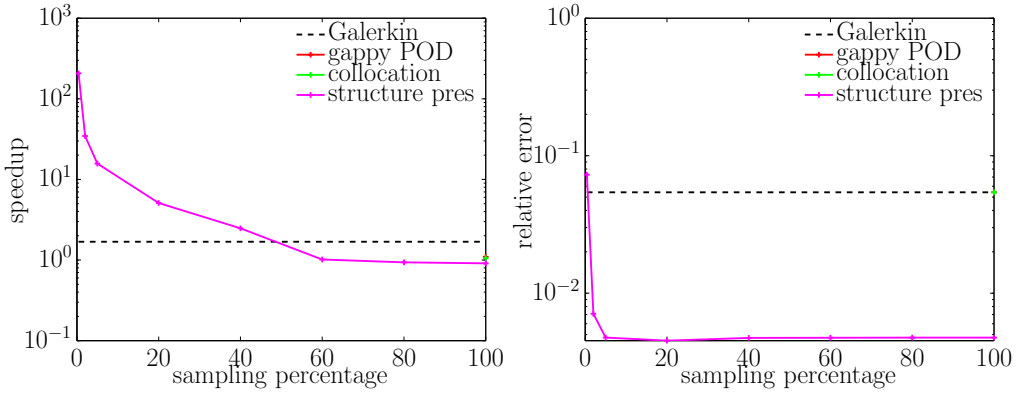
The reduced-order models employ a POD reduced basis with a truncation energy criterion of $\eta_{\mathbf{q}} = 1 - 10^{-6}$, which yields a basis dimension of $n = 147 \ll N$. Again, the gappy POD-based reduced-order model employs a truncation criterion of $\eta_{\boldsymbol{\theta}} = 1$ for its reduced bases. Figure 7.4 reports the reduced-order-model performances for first randomly chosen online point; Figures S4.1 and S4.2 report results for the other two points.

Again, note that the Galerkin reduced-order model is stable and accurate, as it generates relative errors of 1.87%, 14.5%, and 9.16% at the three online points, respectively. However, it yields discouraging speedups of 0.81 (i.e., the simulation was *slower* than for the full-order model), 1.61, and 1.32 at these points. The proposed structure-preserving methods are always stable and quite accurate. They yield nearly the same performance, although method two (which employs the matrix gappy POD approximation) generates lower errors for online points with 4.9% sampling. From Figure 7.4, note that the high-frequency oscillations that characterize the proposed methods' responses are smoothed out when the sampling percentage reaches 20%. In particular, proposed method 2 generates speedups of 15.9, 28.5, and 26.2 and relative errors of 11.6%, 13.0%, and 11.6% for 4.9% sampling. For 20% sampling, the method generates speedups of 4.84, 9.82, and 7.72, and relative errors of 1.51%, 5.83%, and 1.09%.

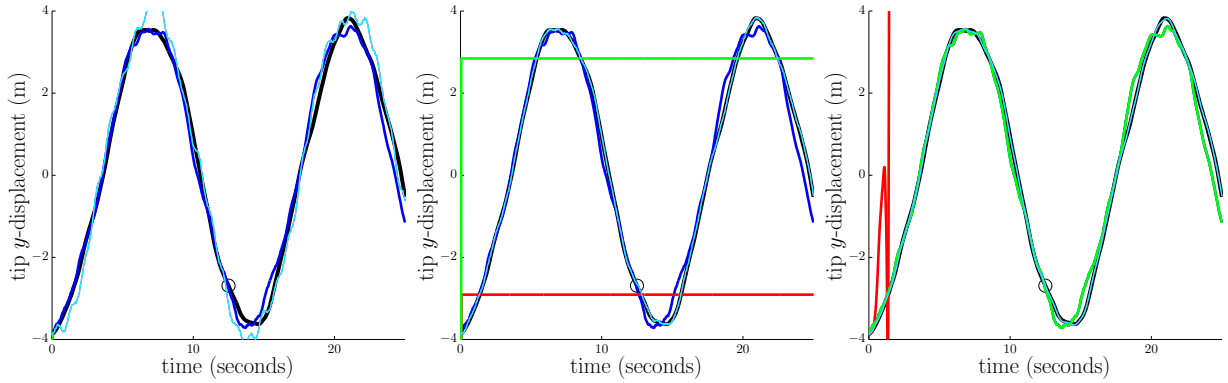
In this example, the gappy POD reduced-order model is unstable for all sampling percentages, and the collocation reduced-order model is only stable for 100% sampling (at which point it is mathematically equivalent to the Galerkin reduced-order model). This is not surprising, as these methods do not preserve problem structure, nor do they guarantee energy conservation. This poor performance can be attributed to the stiff dynamics that characterize the considered conservative Lagrangian dynamical system, which lead to instabilities for both reduced-order models.

[8] In all response plots, a 'flat line' indicates that the nonlinear solver failed to converge after 500 Newton iterations at three different time steps.

[9] We will show in Section 7.2.1 that introducing dissipation improves the performance of both collocation and gappy POD. Note that gappy POD was also unstable for other attempted energy criteria of $\eta_{\boldsymbol{\theta}} = 1 - 10^{-9}$ and $\eta_{\boldsymbol{\theta}} = 1 - 10^{-8}$.



(a) Performance as a function of sampling percentage $m/N \times 100\%$. Missing data points indicate unstable responses.



(b) 0.4% sampling (c) 2% sampling (d) 100% sampling

FIG. 7.2. *Conservative, fixed-parameters case. Reduced-order model performance as a function of sampling percentage $m/N \times 100\%$. Legend for bottom plots: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (circle).*

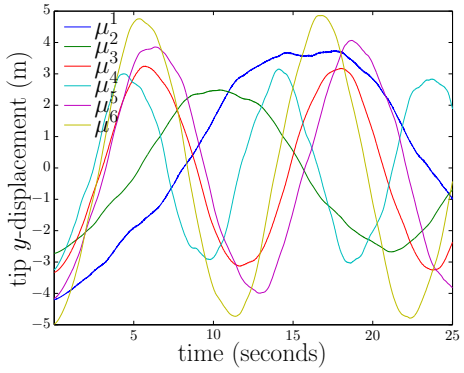
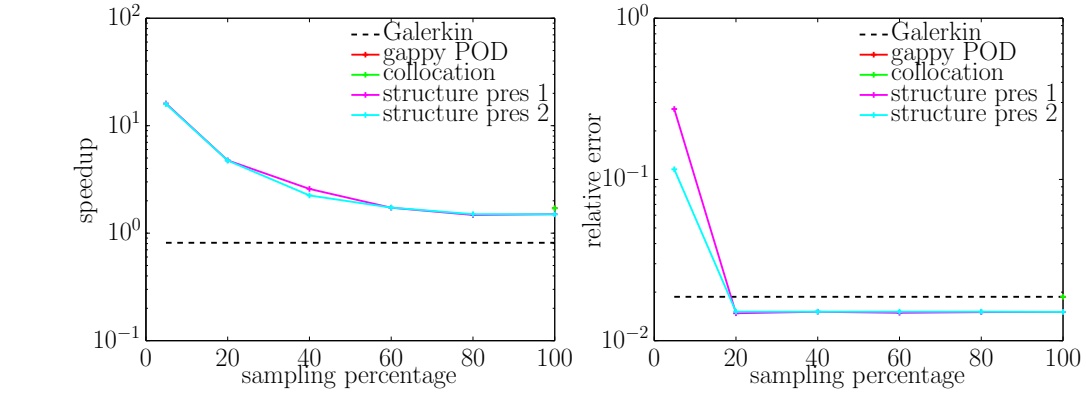


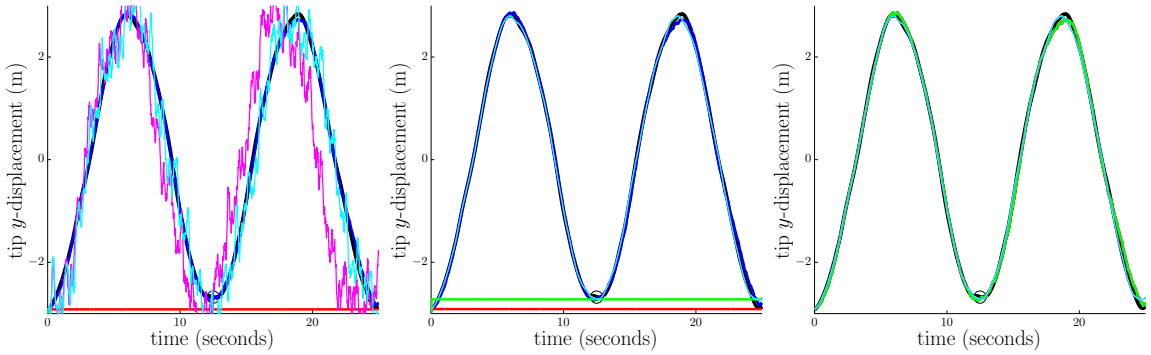
FIG. 7.3. *Conservative, parameter-varying case: tip displacement for the training set $\{\mu^i\}$.*

This example showcases the practical importance of preserving Lagrangian structure: the proposed structure-preserving reduced-order models are the only models that yield both *fast* and *accurate* results.

7.2. Non-conservative case. We now consider the non-conservative case in which the non-conservative dissipative and external forces are nonzero. That is, we set $\zeta = \sin(5°)$ and all parameters μ_i, $i = 1, \ldots, 16$ are free to vary. We again set the nominal forces to $\bar{f}_1 = \bar{f}_2 = 2\text{kg} \times 9.81\text{m/s}^2$ and



(a) Performance as a function of sampling percentage $m/N \times 100\%$. Missing data points indicate unstable responses.



(b) 4.9% sampling (c) 20% sampling (d) 100% sampling

FIG. 7.4. *Conservative, parameter-varying case: reduced-order model performance as a function of sampling percentage* $m/N \times 100\%$ *for online point 1.*

$\bar{f}_3 = \bar{f}_4 = 0.4\text{kg} \times 9.81\text{m/s}^2$. As before, we perform a timestep-verification study for the nominal point $\bar{\mu}$ characterized by $\bar{\mu}_i = 0$, $i = 1, \dots 16$ to discover an appropriate timestep. A timestep size of $\Delta t = 0.1$ seconds leads to an approximated error using Richardson extrapolation of 1.07×10^{-4}. We can therefore declare this to be an appropriate timestep size for the numerical experiments. Further, we note that the average number of Newton iterations per timestep is 2.56, so the nonlinearity remains significant.

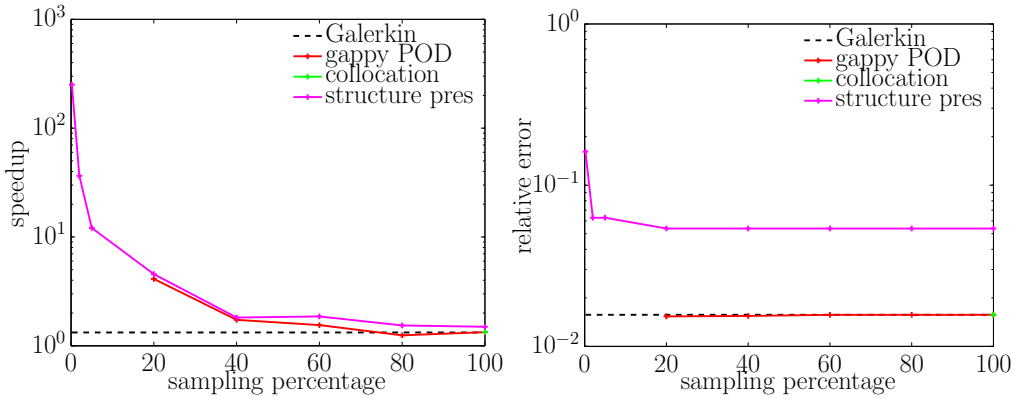
7.2.1. Fixed parameters. We again test the different methods for the fixed-parameters case where $\{\mu^i\} = \bar{\mu}$ and $\mu^\star = \bar{\mu}$. As above, we only collect snapshots for the first half of the time interval, and the proposed structure-preserving methods yield the same results. The POD reduced basis $\mathbf{V}$ is generated using an energy criterion of $\eta_{\mathbf{q}} = 1 - 10^{-5}$, which leads to a basis dimension of $n = 6 \ll N$. The gappy POD-based reduced-order model employs an energy criterion of 1 for its reduced bases $\mathbf{W}_{\boldsymbol{\theta}}$. Figure 7.5 reports results for the reduced-order models as the number of sample indices varies.

Again, the Galerkin reduced-order model is accurate, with a relative error of 1.57%, but produces a speedup of only 1.33. The proposed structure-preserving method is always stable as expected. Its performance is dependent upon the sampling percentage, with (arguably) the best performance achieved for 2% sampling (6.28% error and 36.5 speedup). For 0.2% sampling, the method produces 16.1% error and a speedup of 251; 20% sampling leads to 5.39% error and a speedup of 4.6.

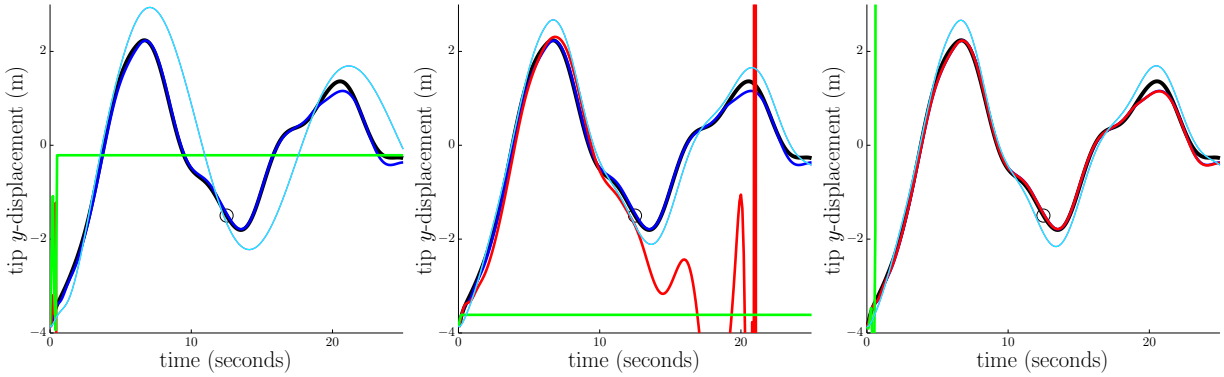
The gappy POD reduced-order model is unstable for 0.2%, 2%, and 5% sampling, but stabilizes at 20%; compared to the conservative case, this stability likely results from less stiff dynamics due to the presence of damping. This yields its best performance of 1.53% error, but only a 4.1 speedup.[10] The collocation

[10] A truncation criterion of 1 yielded the best performance for Gappy POD. For $\eta_{\boldsymbol{\theta}} = 1 - 10^{-9}$, Gappy POD was unstable for all sampling percentages. It was also unstable for all sampling percentages when it employed an energy criterion of $\eta_{\boldsymbol{\theta}} = 1 - 10^{-8}$.

reduced-order model is stable only for full sampling, when it is equivalent to Galerkin.



(a) Performance as a function of sampling percentage $m/N \times 100\%$. Missing data points indicate unstable responses.



(b) 0.2% sampling (c) 5% sampling (d) 20% sampling

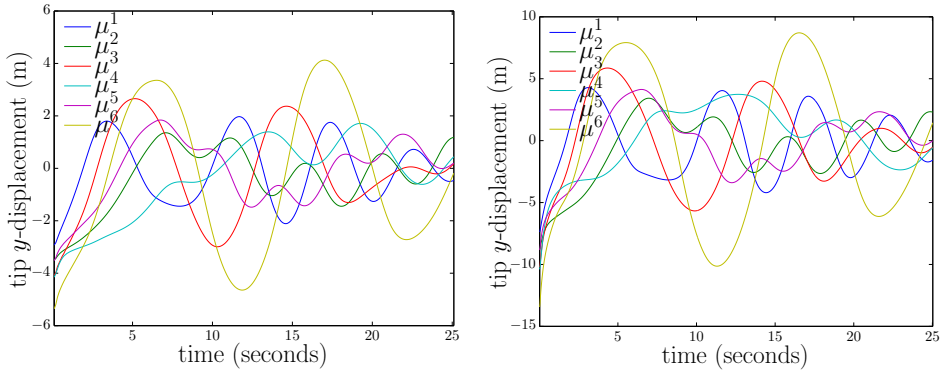
FIG. 7.5. *Non-conservative, fixed-parameters case. Reduced-order model performances. Legend for bottom plots: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (circle).*

7.2.2. Varying parameters. We now consider the parameter-varying case where $\mu^\star \notin \{\mu^i\}$. We again employ $p = 6$ training points and determine $\{\mu^i\}$ using Latin hypercube sampling. We choose three online points randomly in the parameter space. Figure 7.6(a) shows the tip displacement for the training points; clearly, the responses are significantly different from one another. Because we are in a fully predictive scenario, the two proposed structure-preserving reduced-order models again yield different results. All reduced-order models employ an energy criterion of $\eta_{\mathbf{q}} = 1 - 10^{-5}$, which leads to a basis dimension of $n = 12$. We employ $\eta_{\boldsymbol{\theta}} = 1$ for the gappy POD reduced-order model.

Figure 7.7 reports the results for this predictive study online point 1; Figures S4.3 and S4.4 provide results for points 2 and 3. At all three points, Galerkin is accurate (relative errors of 7.5%, 9.8%, and 13.5%), but does not yield significant speedups (speedups of 1.4, 1.2, and 1.1). As is apparent from the plots, the two proposed structure-preserving methods yield nearly the same performance. At 0.4% sampling, method 1 produces relative errors of 2.82%, 11.0%, and 10.3% and speedups of 96.3, 73.3, and 82.3. At 2% sampling, method 1 yields relative errors of 4.38%, 10.9%, and 7.97% and speedups of 21.6, 19.2, and 16.8.

In this example, gappy POD does not stabilize until 40% sampling, at which point the speedup is less than 1. Thus, gappy POD does not yield performance improvement for this problem. Collocation stabilizes at 80% sampling, and also fails to generate any performance improvement.

7.3. Effect of nonlinearity. We now aim to characterize the dependence of problem nonlinearity on the proposed methods' performances. Recall from Section 6 that the potential-energy approximation is



(a) Section 7.2.1 experiments

(b) Section 7.3 experiments (higher nonlinearity)

FIG. 7.6. *Non-conservative, parameter-varying case: tip displacement for the training set* $\{\mu^i\}$ *for two sets of experiments.*

computed by matching the gradient of the potential energy to first order about the equilibrium configuration $\mathbf{q}_0(\mu^\star)$. In the presence of stronger nonlinearity, we expect the configuration to deviate further from equilibrium, which should degrade the accuracy of the approximation.

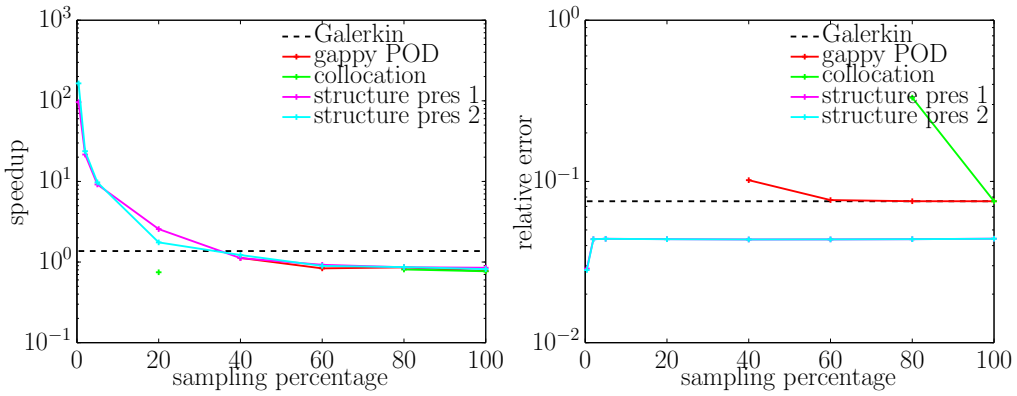
To numerically assess the effect of nonlinearity, we repeat the experiments from Section 7.2.2 using the same training and online points, but we increase the nominal forces by a factor of 2.5 to $\underline{f}_1 = \underline{f}_2 = 5\text{kg} \times 9.81\text{m/s}^2$ and $\underline{f}_3 = \underline{f}_4 = 1\text{kg} \times 9.81\text{m/s}^2$. We first perform a timestep-verification study for the nominal point $\bar{\mu}$. As expected, a smaller timestep size of $\Delta t = 0.025$ seconds is required, as it corresponds to an approximated error using Richardson extrapolation of 3.62×10^{-4}.

Figure 7.6(b) displays the tip displacement for the training points. Note that the responses are similar to those for the previous study (see Figure 7.6(a)), but have larger magnitudes that imply a stronger geometric nonlinearity. The reduced-order models employ a POD reduced basis of dimension $n = 14$, which was obtained by an energy criterion of $\eta_{\mathbf{q}} = 1 - 10^{-5}$; gappy POD uses $\eta_{\boldsymbol{\theta}} = 1$ for its nonlinear-function bases.

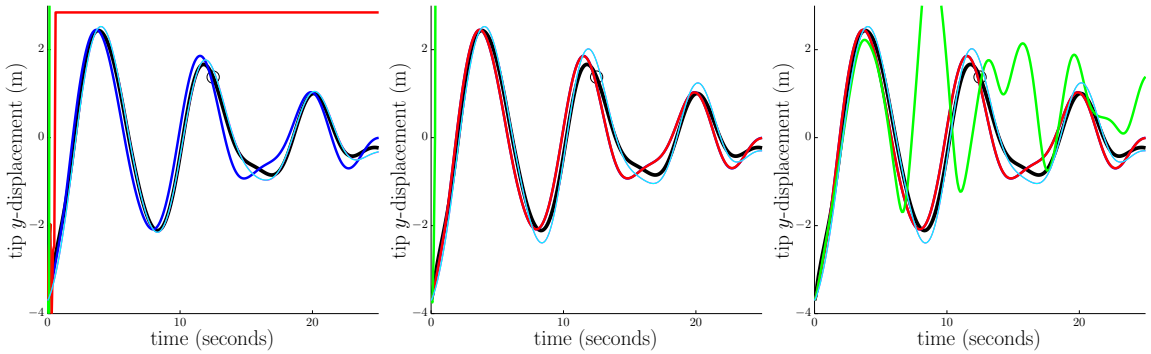
Figure 7.8 reports the reduced-order models' performances for online point 1; Figures S4.5 and S4.6 provide results for online points 2 and 3. As in the previous case, Galerkin is accurate (relative errors of 3.0%, 8.3%, and 10.0% at the online points), but does not generate significant speedups (1.71, 1.67, and 1.0). The proposed structure-preserving techniques again yield very similar results to each other; however, the errors are significantly larger than in the experiments from Section 7.2.2 characterized by a less severe nonlinearity. For 0.5% sampling, proposed method 1 yields relative errors of 11.1%, 21.3%, and 15.9% at the online points and speedups of 160, 116.4, and 98.9. Thus, increasing the nonlinearity in the problem does have a deleterious effect on the methods' performances.

However, it is important to note that other complexity-reducing reduced-order models fail to generate significant performance improvement on this more highly nonlinear problem. In particular collocation is always unstable for a sampling percentage less than 60%, and gappy POD is always unstable when the percentage is less than 80%. As a result, the best speedup obtained by either of the methods is only 2.77 (collocation for 60% sampling for online point 2).

7.4. Sampling percentage performance. Table 7.2 reports the sampling percentage yielding the best performance of the proposed method for the numerical experiments in the previous sections. The sampling percentage leading to the best performance is problem dependent. However, a sampling percentage less than 2% yielded the best results in most cases. This can be attributed to the observed independence of accuracy on sampling percentage once sampling exceeds roughly 2% in most cases, contrasted with the strong dependence of speedup on sampling percentage. The single exception—the conservative, varying-parameters case—is a highly stiff problem wherein parameter dependence excites a richly varying set of dynamics. This is evidenced by the fact that the problem was characterized by the largest basis dimension across all experiments ($n = 147$).



(a) Performance as a function of sampling percentage $m/N \times 100\%$. Missing data points indicate unstable responses.



(b) 0.4% sampling (c) 40% sampling (d) 80% sampling

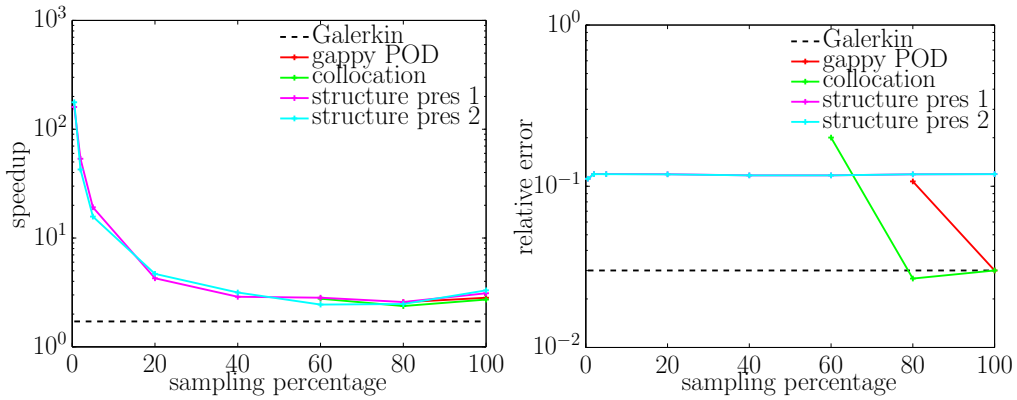
FIG. 7.7. *Non-conservative, parameter-varying case. Reduced-order model performances for online point 1. Legend for bottom plots: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (circle).*

Section	description	best sampling percentage(s)	error	speedup
7.1.1	conservative, fixed params	0.4%⋆ 2%	7.3% 0.71%	207 34.5
7.1.2	conservative, varying params	20%	5.83%	9.82
7.2.1	non-conservative, fixed params	2%	6.28%	36.5
7.2.2	non-conservative, varying params	0.4%⋆	2.82%	96.3
7.3	non-conservative, highly nonlinear, varying params	0.5%⋆	11.1%	160

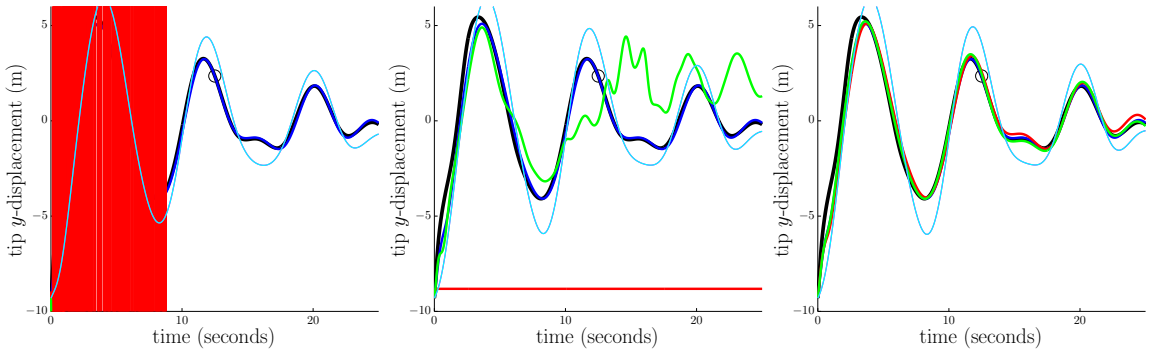
TABLE 7.2

Best sampling percentages for experiments in Sections 7.1–7.3. An asterisk indicates the minimal sampling percentage, i.e., $m = n$. Results correspond to online point 1.

8. Conclusions. This paper has presented an efficient structure-preserving model-reduction strategy applicable to simple mechanical systems. The methodology directly approximates the quantities that define the problem's Lagrangian structure and subsequently derives the equations of motion, while ensuring low online computational cost. The method is distinct from typical model-reduction methods for nonlinear ODEs; these methods are typically based on collocation and DEIM/gappy POD techniques that approximate the equations of motion and destroy Lagrangian structure. At the core of the methodology are the reduced-basis sparsification (RBS) and matrix gappy POD techniques for approximating parameterized reduced matrices while preserving symmetry and positive definiteness; we also employed the former method to preserve potential-energy structure.



(a) Performance as a function of sampling percentage $m/N \times 100\%$. Missing data points indicate unstable responses.



(b) 0.5% sampling responses (c) 60% sampling responses (d) 80% sampling responses

FIG. 7.8. *Non-conservative, highly nonlinear parameter-varying case. Reduced-order model performances for online point 1. Legend for bottom plots: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (circle).*

Numerical experiments on a geometrically nonlinear parameterized truss structure highlight the method's benefits: preserving Lagrangian structure ensured the method generated stable responses that were often very accurate. Other model-reduction techniques were often unstable; achieving stability required too many sample indices to lead to significant performance gains for those methods. The experiments also showed that both RBS and matrix gappy POD led to nearly the same performance across a range of experiments.

Future work includes devising a method to improve the method's robustness in the presence of strong nonlinearity (e.g., by non-local approximation of the potential-energy function), applying the method to a truly large-scale problem, devising a technique-specific method for choosing the sample indices, and deriving error bounds and error estimates that rigorously assess the accuracy of the method's predictions. Finally, the RBS and matrix gappy POD methods are relevant to a wider class of problems than model reduction for Lagrangian systems; future work will investigate to their applicability to other scenarios.

Acknowledgments. The authors acknowledge Julien Cortial for insightful discussions and for providing the original truss code that was modified to generate the numerical results. The authors also acknowledge Clancy Rowley for useful comments received at the 2013 SIAM Conference on Computational Science and Engineering. This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000. The authors also acknowledge support by the Department of Energy Office of Advanced Scientific Computing Research under contract 10-014804.

REFERENCES

[1] S. Adhikari, *Damping models for structural vibration*, PhD thesis, Cambridge University, September 2000.

[2] S.S. An, T. Kim, and D.L. James, *Optimizing cubature for efficient integration of subspace deformations*, ACM Transactions on Graphics (TOG), 27 (2008), p. 165.

[3] P. Astrid, S. Weiland, K. Willcox, and T. Backx, *Missing point estimation in models described by proper orthogonal decomposition*, IEEE Transactions on Automatic Control, 53 (2008), pp. 2237–2251.

[4] 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*, Comptes Rendus Mathématique Académie des Sciences, 339 (2004), pp. 667–672.

[5] Christopher Beattie and Serkan 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.

[6] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, International Journal for Numerical Methods in Engineering, 86 (2011), pp. 155–181.

[7] 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.

[8] K. Carlberg, R. Tuminaro, and P. Boggs, *Efficient structure-preserving model reduction for nonlinear mechanical systems with application to structural dynamics*, in AIAA Paper 2012-1969, 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Honolulu, Hawaii, April 23–26 2012.

[9] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2737–2764.

[10] I. Chowdhury and S.P. Dasgupta, *Computation of Rayleigh damping coefficients for large systems*, The Electronic Journal of Geotechnical Engineering, 8 (2003).

[11] M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A937–A969.

[12] D.M. Dunlavy, T.G. Kolda, and E. Acar, *Poblano v1. 0: A Matlab toolbox for gradient-based optimization*, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Tech. Rep. SAND, 1422 (2010).

[13] R. Everson and L. Sirovich, *Karhunen–Loève procedure for gappy data*, Journal of the Optical Society of America A, 12 (1995), pp. 1657–1664.

[14] Charbel Farhat, Philip Avery, Todd Chapman, and Julien 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.

[15] D. Galbally, K. Fidkowski, K. Willcox, and O. Ghattas, *Non-linear model reduction for uncertainty quantification in large-scale inverse problems*, International Journal for Numerical Methods in Engineering, 81 (2009), pp. 1581–1608.

[16] M. A. Grepl and A. T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, ESAIM-Mathematical Modelling and Numerical Analysis (M2AN), 39 (2005), pp. 157–181.

[17] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31, Springer Verlag, 2006.

[18] R.A. Horn and C.R. Johnson, *Matrix analysis*, Cambridge university press, 1990.

[19] K. Ito and SS Ravindran, *A reduced basis method for control problems governed by pdes*, Control and estimation of distributed parameter systems, (1998), pp. 153–168.

[20] S. Lall, P. Krysl, and J.E. Marsden, *Structure-preserving model reduction for mechanical systems*, Physica D: Nonlinear Phenomena, 184 (2003), pp. 304–318.

[21] J.E. Marsden and M. West, *Discrete mechanics and variational integrators*, Acta Numerica, 10 (2001), p. 514.

[22] Michael D McKay, Richard J Beckman, and William J Conover, *Comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), pp. 239–245.

[23] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici, *Reliable real-time solution of parameterized partial differential equations: Reduced-basis output bound methods*, Journal of Fluids Engineering, 124 (2002), pp. 70–80.

[24] Christophe Prud'Homme, Dimitrios V Rovas, Karen Veroy, Luc Machiels, Yvon Maday, Anthony T Patera, Gabriel Turinici, et al., *Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods*, Journal of Fluids Engineering, 124 (2001), pp. 70–80.

[25] D. Ryckelynck, *A priori hyperreduction method: an adaptive approach*, Journal of Computational Physics, 202 (2005), pp. 346–366.

[26] T. Tonn, *Reduced-Basis Method (RBM) for Non-Affine Elliptic Parameterized PDEs*, PhD thesis, Ulm University, December 2011.

[27] B. Vandereycken, *Low-rank matrix completion by Riemannian optimization*, SIAM Journal on Optimization, 23 (2013), pp. 1214—1236.

[28] D. Wirtz, D. C. Sorensen, and B. Haasdonk, *A-posteriori error estimation for DEIM reduced nonlinear dynamical systems*, Preprint Series, Stuttgart Research Centre for Simulation Technology, (2012).

SUPPLEMENTARY MATERIAL: PRESERVING LAGRANGIAN STRUCTURE IN NONLINEAR MODEL REDUCTION WITH APPLICATION TO STRUCTURAL DYNAMICS

KEVIN CARLBERG*, RAY TUMINARO†, AND PAUL BOGGS‡

S1. Proofs.

S1.1. Proof of Theorem 5.1. The proof relies on a generalization of the Cauchy interlacing theorem. We begin by restating Theorem 4.3.10 from Ref. [2].

THEOREM S1.1. *Let two sequences of interlacing real numbers be given by* $(\lambda_i^{(r)})_{i=1}^n$ *and* $(\lambda_i^{(s)})_{i=1}^m$ *as described by inequality* (5.5) *when* $m = n+1$. *Define* $\mathbf{\Lambda}^{(r)} = \text{diag}(\lambda_i^{(r)})$ *and* $\mathbf{\Lambda}^{(s)} = \text{diag}(\lambda_i^{(s)})$. *Then, there exists a real number* $\alpha \in \mathbb{R}$ *and a vector* $\mathbf{y} \in \mathbb{R}^n$ *such that* $\mathbf{\Lambda}^{(s)}$ *are the eigenvalues of the real symmetric matrix*

$$\hat{\mathbf{B}}^{(\text{bordered})} \equiv \begin{pmatrix} \mathbf{\Lambda}^{(r)} & \mathbf{y} \\ \mathbf{y}^T & \alpha \end{pmatrix}.$$

The following corollary is a direct consequence of the above theorem.

COROLLARY S1.2. *Given* $\hat{\mathbf{B}}^{(s)} \in \text{SPD}\,(m)$ *and* $\hat{\mathbf{B}}^{(r)} \in \text{SPD}\,(m-1)$, *where* $\text{SPD}\,(k)$ *denotes the set of* $k \times k$ *symmetric positive-definite matrices, whose eigenvalues interlace, then*

$$\exists\ \mathbf{U}_m \text{ such that } \mathbf{U}_m^T \hat{\mathbf{B}}^{(s)} \mathbf{U}_m = \hat{\mathbf{B}}^{(r)} \quad \text{with} \quad \mathbf{U}_m^T \mathbf{U}_m = I. \tag{S1.1}$$

Proof. Using the above theorem, a matrix $\hat{\mathbf{B}}^{(\text{bordered})} \in \text{SPD}\,(m)$ exists that shares the same eigenvalues with $\hat{\mathbf{B}}^{(s)}$. Let $\mathbf{Q}^{(\text{bordered})}$, $\mathbf{Q}^{(s)}$, and $\mathbf{Q}^{(r)}$ be the (square, orthogonal) matrices of eigenvectors for $\hat{\mathbf{B}}^{(\text{bordered})}$, $\hat{\mathbf{B}}^{(s)}$, and $\hat{\mathbf{B}}^{(r)}$, respectively. Then,

$$\left(\mathbf{Q}^{(\text{bordered})}\right)^T \hat{\mathbf{B}}^{(\text{bordered})} \mathbf{Q}^{(\text{bordered})} = \mathbf{\Lambda}^{(s)} = \mathbf{Q}^{(s)} \hat{\mathbf{B}}^{(s)} (\mathbf{Q}^{(s)})^T,$$

which implies that

$$\hat{\mathbf{B}}^{(\text{bordered})} = \mathbf{Q}^{(\text{bordered})} (\mathbf{Q}^{(s)})^T \hat{\mathbf{B}}^{(s)} \mathbf{Q}^{(s)} (\mathbf{Q}^{(\text{bordered})})^T.$$

From the definition of $\hat{\mathbf{B}}^{(\text{bordered})}$ it also follows that

$$[\mathbf{I} \quad \mathbf{0}] \hat{\mathbf{B}}^{(\text{bordered})} [\mathbf{I} \quad \mathbf{0}]^T = \mathbf{\Lambda}^{(r)},$$

where I is the $(m-1) \times (m-1)$ identity matrix and $\mathbf{0}$ is the zero column vector of length $m-1$, and thus

$$\hat{\mathbf{B}}^{(r)} = \mathbf{Q}^{(r)} \mathbf{\Lambda}^{(r)} (\mathbf{Q}^{(r)})^T = \mathbf{Q}^{(r)} \left[[\mathbf{I} \quad \mathbf{0}] \hat{\mathbf{B}}^{(\text{bordered})} [\mathbf{I} \quad \mathbf{0}]^T \right] (\mathbf{Q}^{(r)})^T. \tag{S1.2}$$

Combining the above, we can write

$$\hat{\mathbf{B}}^{(r)} = \mathbf{Q}^{(r)} [\mathbf{I} \quad \mathbf{0}] \mathbf{Q}^{(\text{bordered})} (\mathbf{Q}^{(s)})^T \hat{\mathbf{B}}^{(s)} \mathbf{Q}^{(s)} (\mathbf{Q}^{(\text{bordered})})^T [\mathbf{I} \quad \mathbf{0}]^T (\mathbf{Q}^{(r)})^T$$

and so (S1.1) is satisfied taking $\mathbf{U}_m = \left[\mathbf{Q}^{(r)} [\mathbf{I} \quad \mathbf{0}] \mathbf{Q}^{(\text{bordered})} (\mathbf{Q}^{(s)})^T\right]^T$. □

The generalization of the Cauchy interlacing thereom now follows.

*Harry S. Truman Fellow, Quantitative Modeling & Analysis Department ktcarlb@sandia.gov
†Numerical Analysis and Applications Department, rstumin@sandia.gov
‡Quantitative Modeling & Analysis Department (retired), ptboggs@sandia.gov

THEOREM S1.3. *Given two matrices* $\hat{\mathbf{B}}^{(s)} \in \mathrm{SPD}(m)$ *and* $\hat{\mathbf{B}}^{(r)} \in \mathrm{SPD}(n)$ *(with* $m \geq n$*), then*

$$\exists\ \mathbf{U} \quad \textit{such that} \quad \mathbf{U}^T \hat{\mathbf{B}}^{(s)} \mathbf{U} = \hat{\mathbf{B}}^{(r)} \quad \textit{with} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I} \tag{S1.3}$$

if and only if the eigenvalues $\lambda_i^{(r)}$, $i = 1, \ldots, n$ *interlace the eigenvalues* $\lambda_i^{(s)}$, $i = 1, \ldots, m$ *defined as*

$$\hat{\mathbf{B}}^{(r)} \hat{\mathbf{x}}_i^{(r)} = \lambda_i^{(r)} \hat{\mathbf{x}}_i^{(r)}, \quad i = 1, \ldots, n \tag{S1.4}$$

$$\hat{\mathbf{B}}^{(s)} \hat{\mathbf{x}}_i^{(s)} = \lambda_i^{(s)} \hat{\mathbf{x}}_i^{(s)}, \quad i = 1, \ldots, m. \tag{S1.5}$$

The definition of interlacing is given by inequality (5.5).

Proof. It is well known that if $\hat{\mathbf{B}}^{(s)} \in \mathrm{SPD}(m)$ is given along with an orthogonal $m \times n$ matrix $\mathbf{U}$ (with $m \geq n$), then the eigenvalues of $\mathbf{U}^T \hat{\mathbf{B}}^{(s)} \mathbf{U}$ must interlace those of $\hat{\mathbf{B}}^{(s)}$. This is referred to as the Cauchy interlacing theorem (e.g., see [3]).

The converse of the Cauchy interlacing theorem is less widely known. The case $m = n$ follows trivially using an eigenvalue decomposition. The case $m = n + 1$ corresponds to the above corollary. The proof is completed by generalizing the corollary to the $m > n + 1$ case. This follows from an inductive argument where one considers a projection that reduces the matrix dimension of $\hat{\mathbf{B}}^{(s)}$ by one. According to the above corollary, we have a great deal of flexibility in choosing this lower dimensional matrix if its eigenvalues interlace those of the higher dimension matrix. We then choose a lower dimensional matrix whose eigenvalues not only interlace those of $\hat{\mathbf{B}}^{(s)}$ but whose eigenvalues are also interlaced by those of $\hat{\mathbf{B}}^{(r)}$. That is,

$$\lambda_i^{(s)} \leq \mu_i \leq \lambda_{i+1}^{(s)} \quad \text{and} \quad \mu_i \leq \lambda_i^{(r)} \leq \mu_{i+m-n-1},$$

where μ_i denotes the ith smallest eigenvalue of the intermediate matrix. Rewriting this we obtain the following intervals for the eigenvalues μ_i:

$$\mu_i \geq \begin{cases} \max(\lambda_i^{(s)}, \lambda_{i-m+n+1}^{(r)}) & i \geq m - n \\ \lambda_i^{(s)} & i < m - n \end{cases}$$

and

$$\mu_i \leq \begin{cases} \min(\lambda_{i+1}^{(s)}, \lambda_i^{(r)}) & i \leq n \\ \lambda_{i+1}^{(s)} & i > n \end{cases}.$$

Using the interlacing property for $\hat{\mathbf{B}}^{(s)}$ and $\hat{\mathbf{B}}^{(r)}$, one can verify that the intervals for the μ_i are nonempty. That is, $\lambda_{i+1}^{(s)} \geq \lambda_i^{(s)}$ and for those i such that $\lambda_{i-m+n+1}^{(r)}$ is defined, we have $\lambda_{i+1}^{(s)} \geq \lambda_{i-m+n+1}^{(r)}$, $\lambda_i^{(r)} \geq \lambda_i^{(s)}$, and $\lambda_i^{(r)} \geq \lambda_{i-m+n+1}^{(r)}$ Thus, there exists an orthogonal matrix $\mathbf{U}_m$ such that the $(m-1) \times (m-1)$ matrix $\mathbf{U}_m^T \hat{\mathbf{B}}^{(s)} \mathbf{U}_m$ has eigenvalues that are interlaced by those of $\hat{\mathbf{B}}^{(r)}$. We repeat this procedure each time reducing the matrix dimension by one until the final reduction where we take the lower dimension matrix to be equal to $\hat{\mathbf{B}}^{(r)}$. This implies that there exists a set of projection matrices such that $\mathbf{U}^T \hat{\mathbf{B}}^{(s)} \mathbf{U}$ is equal to $\hat{\mathbf{B}}^{(r)}$ where $\mathbf{U} = \mathbf{U}_m \mathbf{U}_{m-1} ... \mathbf{U}_{n+1}$.
□

Equipped with the generalized Cauchy interlacing theorem, we now prove the exactness condition for the $\mathbf{A}(\mu)$ term which is restated in slightly simplified notation.

THEOREM S1.4. *Let* $\mathbf{A}(\mu)$ *have the form*

$$\mathbf{A}(\mu) = h_1(\mu)\ \mathbf{A}_1 + h_2(\mu)\ \mathbf{A}_2 \tag{S1.6}$$

where $\mathbf{A}_1 \in \mathrm{SPD}(N)$, $\mathbf{A}_2 \in \mathrm{SPD}(N)$, *and* $h_1, h_2 : \mathcal{D} \to \mathbb{R}$.

Then,

$$\exists\ \underline{\mathbf{U_A}} \quad \textit{such that} \quad \underline{\mathbf{U_A}}^T \mathbf{P}^T \mathbf{A}(\mu) \mathbf{P} \underline{\mathbf{U_A}} = \mathbf{V}^T \mathbf{A}(\mu) \mathbf{V}, \quad \forall \mu \in \mathcal{D} \tag{S1.7}$$

if and only if the eigenvalues of the general matrix pencil

$$\mathbf{B}^{(r)}\mathbf{x}_i^{(r)} = \lambda_i^{(r)}\mathbf{D}^{(r)}\mathbf{x}_i^{(r)}, \quad i = 1, \ldots, n \tag{S1.8}$$

interlace the eigenvalues of

$$\mathbf{B}^{(s)}\mathbf{x}_i^{(s)} = \lambda_i^{(s)}\mathbf{D}^{(s)}\mathbf{x}_i^{(s)}, \quad i = 1, \ldots, m \tag{S1.9}$$

where

$$\begin{aligned} \mathbf{D}^{(r)} &= \left[\mathbf{V}^T\mathbf{A}_1\mathbf{V}\right], & \mathbf{D}^{(s)} &= \left[\mathbf{P}^T\mathbf{A}_1\mathbf{P}\right], \\ \mathbf{B}^{(r)} &= \left[\mathbf{V}^T\mathbf{A}_2\mathbf{V}\right], \quad \textit{and} & \mathbf{B}^{(s)} &= \left[\mathbf{P}^T\mathbf{A}_2\mathbf{P}\right]. \end{aligned}$$

The definition of interlacing is given by

$$\lambda_i^{(s)} \leq \lambda_i^{(r)} \leq \lambda_{i+m-n}^{(s)} \quad for\ i = 1, ..., n \tag{S1.10}$$

where the eigenvalues are indexed in order of increasing magnitude.

Proof. Clearly (S1.7) can only hold for any $\mu \in \mathcal{D}$ and any functions $h_1(\mu)$ and $h_2(\mu)$ if and only if

$$\underline{\mathbf{U_A}}^T\mathbf{D}^{(s)}\underline{\mathbf{U_A}} = \mathbf{D}^{(r)} \ \text{ and } \ \underline{\mathbf{U_A}}^T\mathbf{B}^{(s)}\underline{\mathbf{U_A}} = \mathbf{B}^{(r)}. \tag{S1.11}$$

Using a carefully chosen linear transformation, it follows that proving the theorem is equivalent to proving the following:

$$\exists\ \mathbf{U} \ \text{ such that } \quad \mathbf{U}^T\hat{\mathbf{B}}^{(s)}\mathbf{U} = \hat{\mathbf{B}}^{(r)} \quad \text{ with } \quad \mathbf{U}^T\mathbf{U} = \mathbf{I} \tag{S1.12}$$

if and only if the eigenvalues $\lambda_i^{(r)}$ interlace the eigenvalues of $\lambda_i^{(s)}$, where the eigenvalues previously defined in Eqs. (S1.8)–(S1.9) also satisfy

$$\begin{aligned} \hat{\mathbf{B}}^{(r)}\hat{\mathbf{x}}^{(r)} &= \lambda^{(r)}\hat{\mathbf{x}}^{(r)} \\ \hat{\mathbf{B}}^{(s)}\hat{\mathbf{x}}^{(s)} &= \lambda^{(s)}\hat{\mathbf{x}}^{(s)}. \end{aligned} \tag{S1.13}$$

The linear transformation relies on Cholesky factorizations given by $\mathbf{D}^{(s)} = \mathbf{L}^{(s)}(\mathbf{L}^{(s)})^T$ and $\mathbf{D}^{(r)} = \mathbf{L}^{(r)}(\mathbf{L}^{(r)})^T$. These factors lead to the following definitions

$$\begin{aligned} \hat{\mathbf{B}}^{(s)} &= (\mathbf{L}^{(s)})^{-1}\mathbf{B}^{(s)}(\mathbf{L}^{(s)})^{-T}, & \hat{\mathbf{x}}^{(s)} &= (\mathbf{L}^{(s)})^T\mathbf{x}^{(s)}, \\ \hat{\mathbf{B}}^{(r)} &= (\mathbf{L}^{(r)})^{-1}\mathbf{B}^{(r)}(\mathbf{L}^{(r)})^{-T}, & \hat{\mathbf{x}}^{(r)} &= (\mathbf{L}^{(r)})^T\mathbf{x}^{(r)}, \quad \text{and} \\ \mathbf{U} &= (\mathbf{L}^{(s)})^T\underline{\mathbf{U_A}}(\mathbf{L}^{(r)})^{-T}, \end{aligned}$$

which can be used in Eqs. (S1.8), (S1.9) and (S1.11) to obtain Eqs. (S1.12) and (S1.13). The proof is completed by recognizing that Eqs. (S1.12) and (S1.13) correspond to the generalized Cauchy interlacing thereorm.
□

S1.2. Proof of Theorem 5.2. *Proof.* If condition 2 holds, then the unconstrained solution to problem (5.12) is

$$\xi_{\mathbf{A}}(\mu^\star) = \left(\left(\mathbf{P}^T \otimes \mathbf{P}^T\right)\mathbf{W_a}\right)^+ \left(\mathbf{P}^T \otimes \mathbf{P}^T\right)\mathrm{vec}\left(\mathbf{A}(\mu^\star)\right). \tag{S1.14}$$

If condition 1 holds, then the vectorized matrix can be expressed as

$$\mathrm{vec}\left(\mathbf{A}(\mu^\star)\right) = \mathbf{W_a}\mathbf{z}(\mu^\star), \tag{S1.15}$$

or equivalently

$$\mathbf{A}(\mu^\star) = \sum_{i=1}^{n_\mathbf{A}} z^i(\mu^\star)\,\underline{\mathbf{A}}_i, \tag{S1.16}$$

where $\mathbf{z} \equiv \left(z^1, \ldots, z^{n_\mathbf{A}}\right)$. Substituting Eq. (S1.15) into Eq. (S1.14) gives $\xi_\mathbf{A}(\mu^\star) = \mathbf{z}(\mu^\star)$ and so Eq. (5.8) yields

$$\tilde{\mathbf{A}}(\mu^\star) = \sum_{i=1}^{n_\mathbf{A}} z^i(\mu^\star)\, \mathbf{V}^T \underline{\mathbf{A}}_i \mathbf{V}. \tag{S1.17}$$

Comparing Eqs. (S1.17) and (S1.16) gives the exactness result: $\tilde{\mathbf{A}}(\mu^\star) = \mathbf{V}^T \mathbf{A}(\mu^\star)\mathbf{V}^T$. □

S1.3. Proof of Lemma 1. *Proof.* The equation $\widetilde{\mathbf{U}_V}^T \widetilde{\mathbf{U}_V} = I$ simply states that the m columns of $\widetilde{\mathbf{U}_V}$ are orthogonal and so any orthogonal matrix $\widetilde{\mathbf{U}_V}$ satisfies the first part of (6.9). Thus, solvability amounts to finding an orthogonal matrix $\widetilde{\mathbf{U}_V}$ such that $\tilde{\mathbf{V}}^T\tilde{\mathbf{c}} = \widetilde{\mathbf{U}_V}^T \mathbf{P}^T\tilde{\mathbf{c}}$. For a solution to exist, however, it is obviously necessary that $||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 = ||\widetilde{\mathbf{U}_V}^T \mathbf{P}^T\tilde{\mathbf{c}}||_2$. If the vector $\mathbf{P}^T\tilde{\mathbf{c}}$ lies within the span of the basis defined by the columns of $\widetilde{\mathbf{U}_V}^T$, then $\widetilde{\mathbf{U}_V}^T \mathbf{P}^T\tilde{\mathbf{c}}$ preserves its 2-norm and so $||\widetilde{\mathbf{U}_V}^T \mathbf{P}^T\tilde{\mathbf{c}}||_2 = ||\mathbf{P}^T\tilde{\mathbf{c}}||_2$. That is, application of $\widetilde{\mathbf{U}_V}^T$ corresponds to a rotation of $\mathbf{P}^T\tilde{\mathbf{c}}$ about the origin and so length is preserved. If instead the vector $\mathbf{P}^T\tilde{\mathbf{c}}$ lies only partially within the span of the orthogonal basis, then $||\widetilde{\mathbf{U}_V}^T \mathbf{P}^T\tilde{\mathbf{c}}||_2 < ||\mathbf{P}^T\tilde{\mathbf{c}}||_2$. That is, application of $\widetilde{\mathbf{U}_V}^T$ corresponds to a rotation of the component of $\mathbf{P}^T\tilde{\mathbf{c}}$ lying within the span of the orthogonal basis. This implies that a necessary condition for a solution to (6.9) is that

$$||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2 \leq ||\mathbf{P}^T\tilde{\mathbf{c}}||_2. \tag{S1.18}$$

Case 1: $m = n$
$\mathbf{P}^T\tilde{\mathbf{c}}$ must lie within the range of $\widetilde{\mathbf{U}_V}$ (as it is a full rank square matrix) and so it is necessary to have equality in (S1.18) when $m = n$. One possible $\widetilde{\mathbf{U}_V}$ in this case is obtained by first defining a $\mathbf{Q}_1 \in \mathbb{R}^{n\times n}$ and a $\mathbf{Q}_2 \in \mathbb{R}^{n\times n}$ such that the first row of $\mathbf{Q}_1$ is $\alpha_1\tilde{\mathbf{c}}^T\tilde{\mathbf{V}}$ with $\alpha_1 = 1/\|\tilde{\mathbf{c}}^T\tilde{\mathbf{V}}\|_2$. Likewise, the first row of $\mathbf{Q}_2$ is taken as $\alpha_1\tilde{\mathbf{c}}^T\mathbf{P}$; note that α_1 also normalizes this row because we assume $\|\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}\|_2 = \|\mathbf{P}^T\tilde{\mathbf{c}}\|_2$. All remaining rows are chosen so that both $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are orthogonal matrices. This gives

$$\mathbf{Q}_1\tilde{\mathbf{V}}^T\tilde{\mathbf{c}} = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2\mathbf{e}_1 = ||\mathbf{P}^T\tilde{\mathbf{c}}||_2\mathbf{e}_1 = \mathbf{Q}_2\mathbf{P}^T\tilde{\mathbf{c}},$$

where $\mathbf{e}_1$ is the first canonical unit vector of length n (first element is one and all other $n-1$ components are zero). A suitable $\widetilde{\mathbf{U}_V}$ that solves (6.9) is then given by $\widetilde{\mathbf{U}_V}^T = \mathbf{Q}_1^T\mathbf{Q}_2$. Thus, equality in (S1.18) is also sufficient when $m = n$.

Case 2: $m > n$
The matrix $\widetilde{\mathbf{U}_V}$ is now rectangular. One possible $\widetilde{\mathbf{U}_V}$ is obtained by defining $\mathbf{Q}_1$ as before while instead defining an $m \times m$ orthogonal matrix $\mathbf{Q}_{\text{full}}$ with the first row again set to $\alpha_2\tilde{\mathbf{c}}^T\mathbf{P}$ with $\alpha_2 = 1/\|\mathbf{P}^T\tilde{\mathbf{c}}\|_2$. This gives

$$\mathbf{Q}_1\tilde{\mathbf{V}}^T\tilde{\mathbf{c}} = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2\mathbf{e}_1 \quad \text{and} \quad \mathbf{Q}_{\text{full}}\mathbf{P}^T\tilde{\mathbf{c}} = ||\mathbf{P}^T\tilde{\mathbf{c}}||_2\tilde{e}_1$$

where $\tilde{e}_1 \in \mathbb{R}^{m\times 1}$ is the first canonical unit vector of length m. If $||\mathbf{P}^T\tilde{\mathbf{c}}||_2 = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2$, then a suitable $\tilde{\mathbf{V}}$ solving (6.9) is given by taking $\mathbf{Q}_2$ to be the first n rows of $\mathbf{Q}_{\text{full}}$ (as $\mathbf{Q}_1\tilde{\mathbf{V}}^T\tilde{\mathbf{c}} = \mathbf{Q}_2\mathbf{P}^T\tilde{\mathbf{c}}$) and taking $\widetilde{\mathbf{U}_V}^T = \mathbf{Q}_1^T\mathbf{Q}_2$. If $||\mathbf{P}^T\tilde{\mathbf{c}}||_2 > ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2$, then we define a vector y as an arbitrary linear combination of the last $m-n$ rows of $\mathbf{Q}_{\text{full}}$ such that y has unit norm. The first row of $\mathbf{Q}_2$ is then taken as

$$(\mathbf{Q}_2)_1 = \alpha_3 \frac{\mathbf{P}^T\tilde{\mathbf{c}}}{||\mathbf{P}^T\tilde{\mathbf{c}}||_2} + \sqrt{1-\alpha_3^2}\; y,$$

where $(\mathbf{Q})_k$ denotes the kth row of a matrix $\mathbf{Q}$ and $\alpha_3 = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2/||\mathbf{P}^T\tilde{\mathbf{c}}||_2$. The remaining rows of $\mathbf{Q}_2$ are simply $(\mathbf{Q}_{\text{full}})_k$ for $k = 2, \ldots, n$. It is easy to verify that $\mathbf{Q}_2$ is again orthogonal and that $\mathbf{Q}_2\mathbf{P}^T\tilde{\mathbf{c}} = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2\mathbf{e}_1$. Thus, $\mathbf{Q}_2\mathbf{P}^T\tilde{\mathbf{c}} = ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2\mathbf{e}_1$ and $\widetilde{\mathbf{U}_V}^T = \mathbf{Q}_1^T\mathbf{Q}_2$ is a possible solution implying that $||\mathbf{P}^T\tilde{\mathbf{c}}||_2 \geq ||\tilde{\mathbf{V}}^T\tilde{\mathbf{c}}||_2$ is a necessary and sufficient condition when $m > n$. □

S2. Solving the matrix gappy POD optimization problem. This approach reformulates the constraints of problem (5.12) in terms of eigenvalues of the reduced matrix. That is, problem (5.12) is reformulated as

$$\begin{aligned} \underset{x \equiv (x_1,\ldots,x_{n_{\mathbf{A}}})}{\text{minimize}} \quad & \|\mathbf{P}^T \mathbf{A}(\mu)\mathbf{P} - \sum_{k=1}^{n_{\mathbf{A}}} \mathbf{P}^T \underline{\mathbf{A}}_k \mathbf{P} x_k\|_F^2 \\ \text{subject to} \quad & \tilde{\lambda}_j(x) \geq \epsilon > 0, \quad j = 1, \ldots, n. \end{aligned} \tag{S2.1}$$

Here, $\tilde{\lambda}_j(x)$, $j = 1, \ldots, n$ are the eigenvalues of the low-dimensional matrix $\sum_{i=1}^{n_{\mathbf{A}}} \mathbf{V}^T \underline{\mathbf{A}}_i \mathbf{V} x_i$ and ϵ denotes a numerical threshold for defining a full-rank matrix. This problem can be numerically solved, e.g., using a gradient-based algorithm.

The gradient of the quadratic objective function is obvious. The gradient of the constraint can be derived by assuming distinct eigenvalues:

$$\frac{\partial \tilde{\lambda}_j}{\partial x_i} = \tilde{\mathbf{y}}_j^T \frac{\partial \left(\sum_{k=1}^{n_{\mathbf{A}}} \mathbf{V}^T \underline{\mathbf{A}}_k \mathbf{V} x_k \right)}{\partial x_i} \tilde{\mathbf{y}}_j \tag{S2.2}$$

$$= \tilde{\mathbf{y}}_j^T \left(\mathbf{V}^T \underline{\mathbf{A}}_i \mathbf{V} \right) \tilde{\mathbf{y}}_j. \tag{S2.3}$$

Here, $\tilde{\mathbf{y}}_j$ is the eigenvector associated with eigenvalue $\tilde{\lambda}_j$. This indicates that computing the gradient $\frac{\partial \tilde{\lambda}_j}{\partial x_i}$ is inexpensive and requires the following steps:

1. Compute the eigenvector $\tilde{\mathbf{y}}_j \in \mathbb{R}^n$ of the matrix $\sum_{k=1}^{n_{\mathbf{A}}} \mathbf{V}^T \underline{\mathbf{A}}_k \mathbf{V} x_k$.
2. Compute the low-dimensional matrix–vector product $\mathbf{w} = \left(\mathbf{V}^T \underline{\mathbf{A}}_i \mathbf{V} \right) \tilde{\mathbf{y}}_j$.
3. Compute the low-dimensional vector–vector product $\tilde{\mathbf{y}}_j^T \mathbf{w}$.

We propose using the unconstrained solution to problem (S2.1) as the initial guess. In practice, this solution is often feasible, so it is typically unnecessary to handle the constraints directly. In the rare cases where it is necessary to deal with multiple equal eigenvalues—or a number of nearby eigenvalues—the methods presented by Andrew and Tan [1] can be used to produce a numerically stable gradient of the constraint; this was not required in the numerical experiments reported in Section 7.

S3. Proper orthogonal decomposition. Algorithm S1 describes the method for computing a proper orthogonal decomposition (POD) basis given a set of snapshots. The method amounts to computing the singular value decomposition of the snapshot matrix; the left singular vectors define the POD basis.

Algorithm S1 Proper-orthogonal-decomposition basis computation (normalized snapshots)

Input: Set of snapshots $\mathcal{X} \equiv \{\mathbf{x}_i\}_{i=1}^{n_{\mathbf{x}}} \subset \mathbb{R}^N$, energy criterion $\eta \in [0, 1]$

Output: $\mathbf{W}(\mathcal{X}, \eta)$

1: Compute the thin singular value decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{X} \equiv [\mathbf{x}_1/\|\mathbf{x}_1\| \ \cdots \ \mathbf{x}_{n_{\mathbf{x}}}/\|\mathbf{x}_{n_{\mathbf{x}}}\|]$.

2: Choose dimension of truncated basis $n = n_e(\eta)$, where

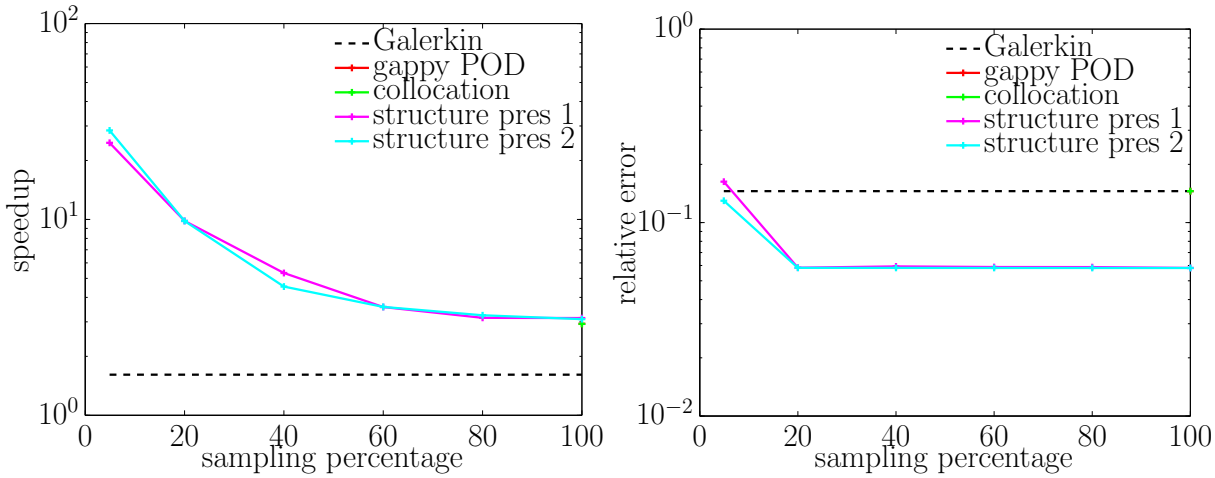
$$n_e(\eta) \equiv \min_{i \in \mathcal{V}(\eta)} i$$

$$\mathcal{V}(\eta) \equiv \{n \in \{1, \ldots, n_{\mathbf{x}}\} \mid \sum_{i=1}^{n} \sigma_i^2 / \sum_{j=1}^{n_{\mathbf{x}}} \sigma_j^2 \geq \eta\},$$

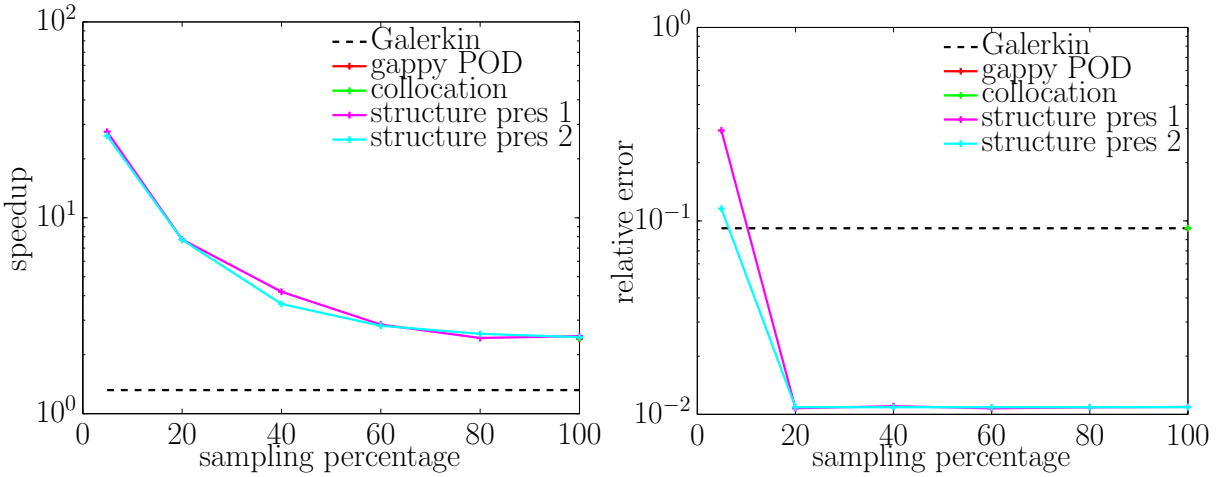
and $\Sigma \equiv \operatorname{diag}(\sigma_i)$ with $\sigma_1 \geq \cdots \geq \sigma_{n_{\mathbf{x}}} \geq 0$.

3: $\mathbf{W}(\mathcal{X}, \eta) = \left[\mathbf{u}^1 \ \cdots \ \mathbf{u}^n\right]$, where $\mathbf{U} \equiv \left[\mathbf{u}^1 \ \cdots \ \mathbf{u}^{n_{\mathbf{x}}}\right]$.

S4. Numerical experiments: extra plots. This section provides supplementary plots associated with numerical experiments carried out in Section 7.



(a) online point 2



(b) online point 3

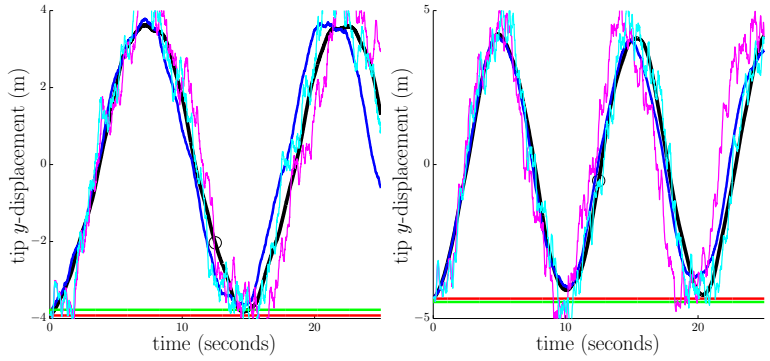
FIG. S4.1. *Conservative, varying-parameters case: reduced-order model performance as a function of sampling percentage* $m/N \times 100\%$.

REFERENCES

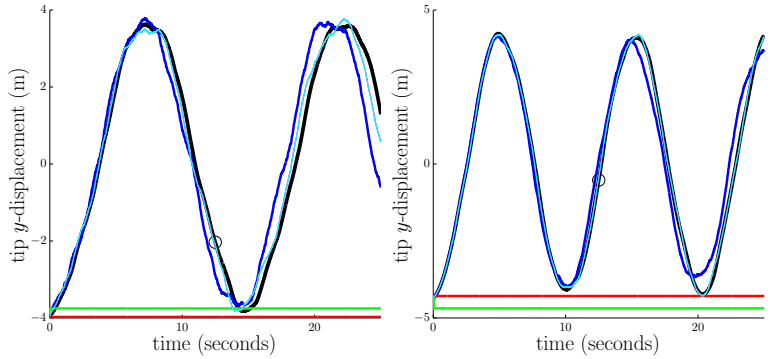
[1] ALAN L. ANDREW AND ROGER C. E. TAN, *Computation of derivatives of repeated eigenvalues and the corresponding eigenvectors of symmetric matrix pencils*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 78–100.

[2] R.A. HORN AND C.R. JOHNSON, *Matrix Analysis*, Cambridge University Press, 1990.

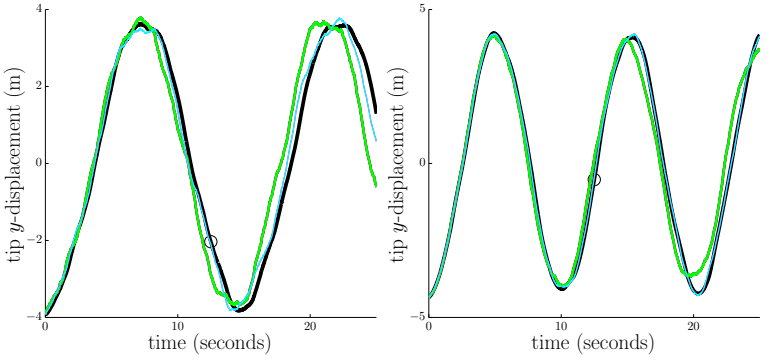
[3] BERESFORD N PARLETT, *The symmetric eigenvalue problem*, vol. 7, SIAM, 1980.



(a) 4.9% sampling

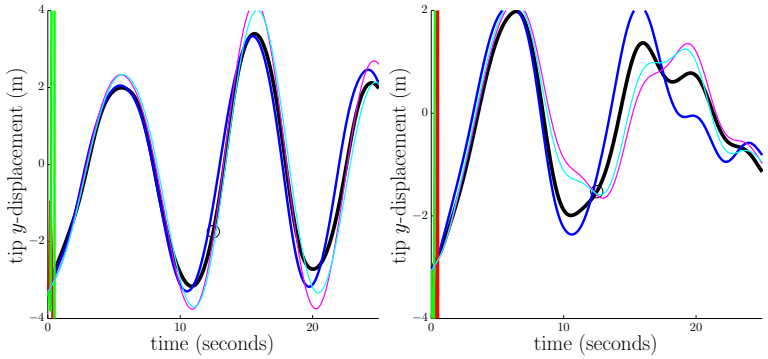


(b) 20% sampling

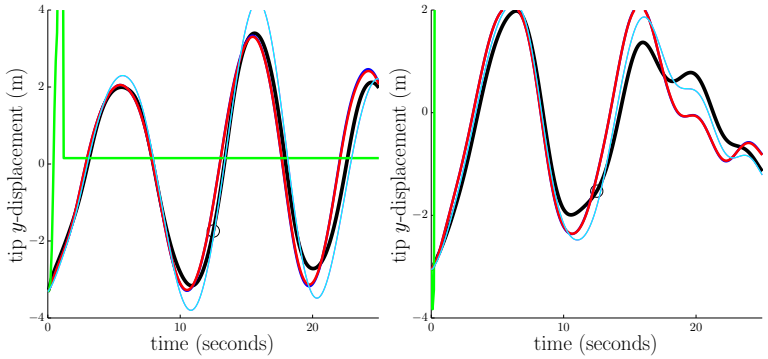


(c) 100% sampling

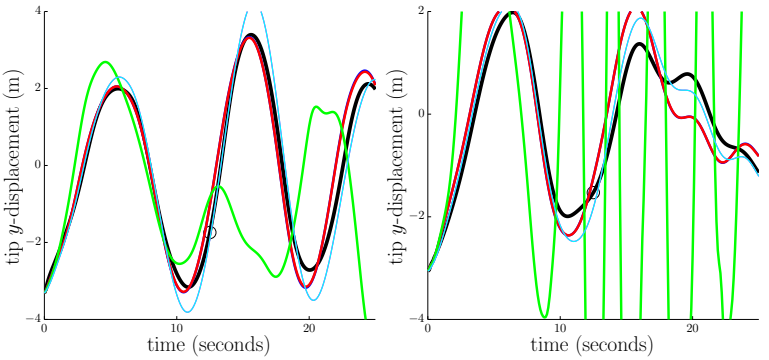
FIG. S4.2. *Conservative, varying-parameters case: reduced-order model responses as a function of sampling percentage* $m/N \times 100\%$ *for online points 2 and 3. Legend: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (black circle).*



(a) predictions at online points 2 and 3, 0.4% sampling

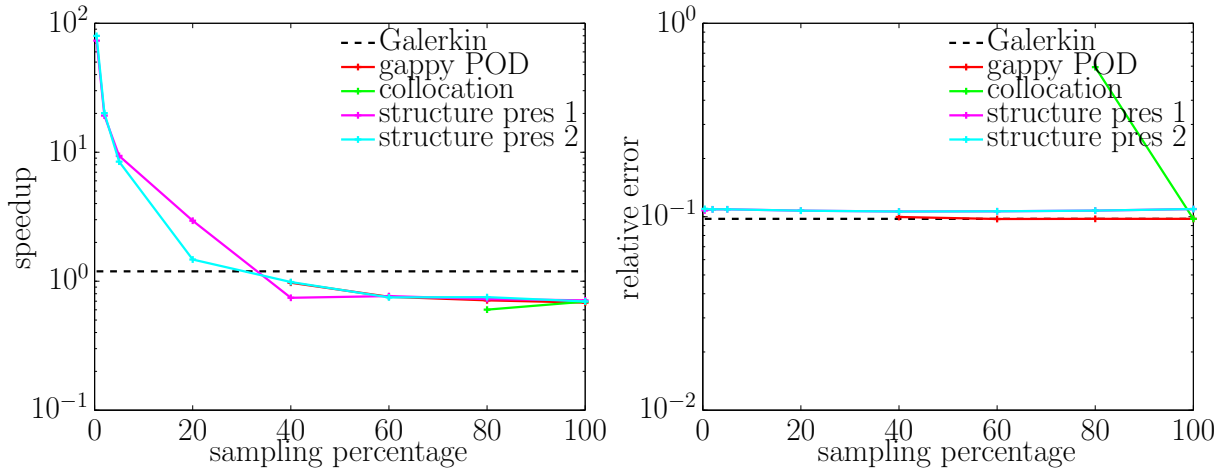


(b) predictions at three randomly chosen online points, 40% sampling

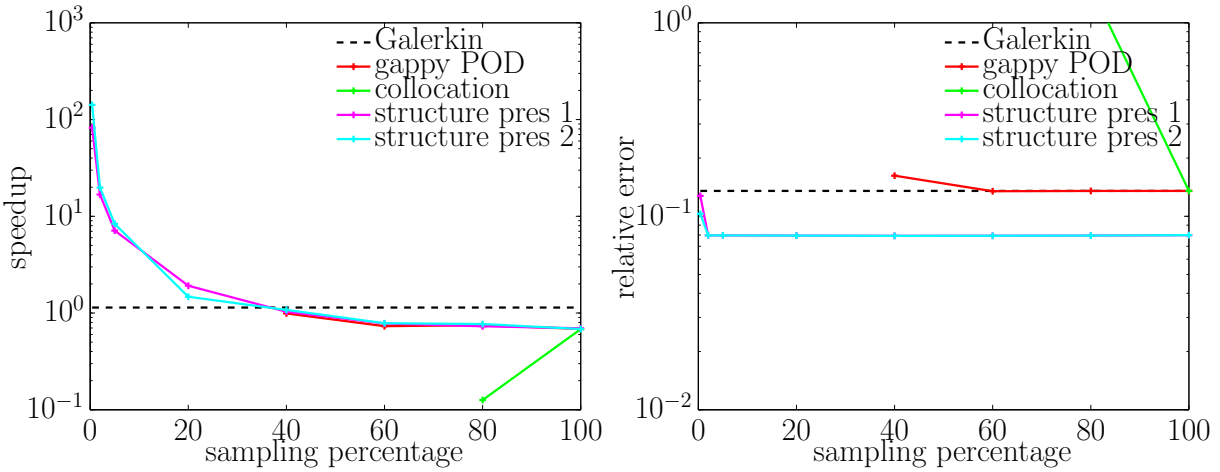


(c) predictions at three randomly chosen online points, 80% sampling

FIG. S4.3. *Non-conservative, parameter-varying case: reduced-order model responses as a function of sampling percentage* $m/N \times 100\%$. *Legend: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (black circle).*

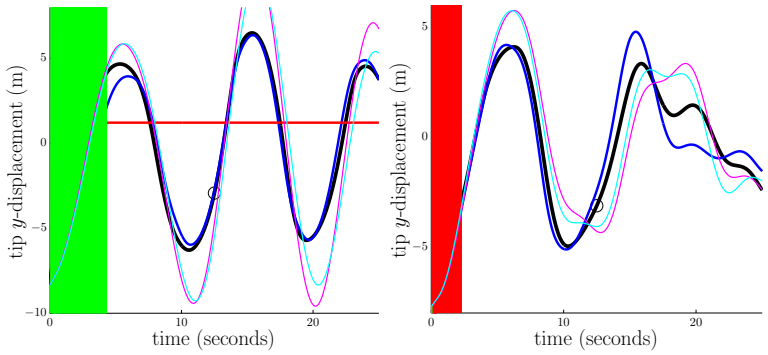


(a) online point 2

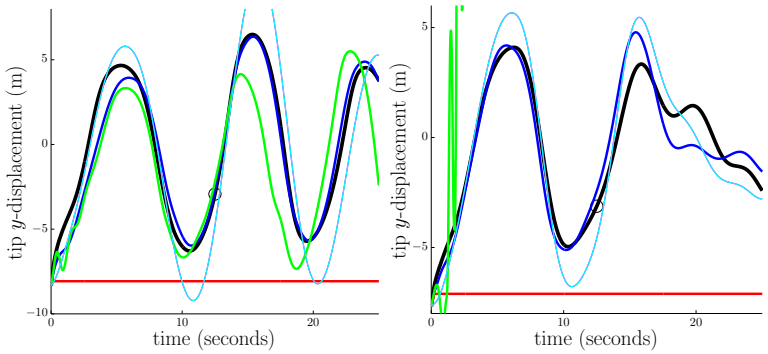


(b) online point 3

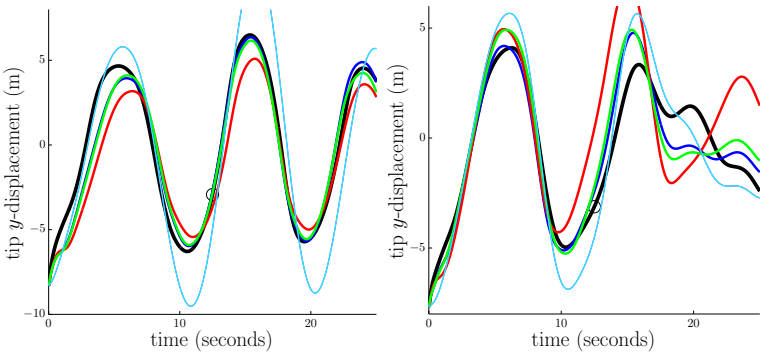
FIG. S4.4. *Non-conservative, parameter-varying case: reduced-order model performance as a function of sampling percentage* $m/N \times 100\%$.



(a) predictions at online points 2 and 3, 0.5% sampling

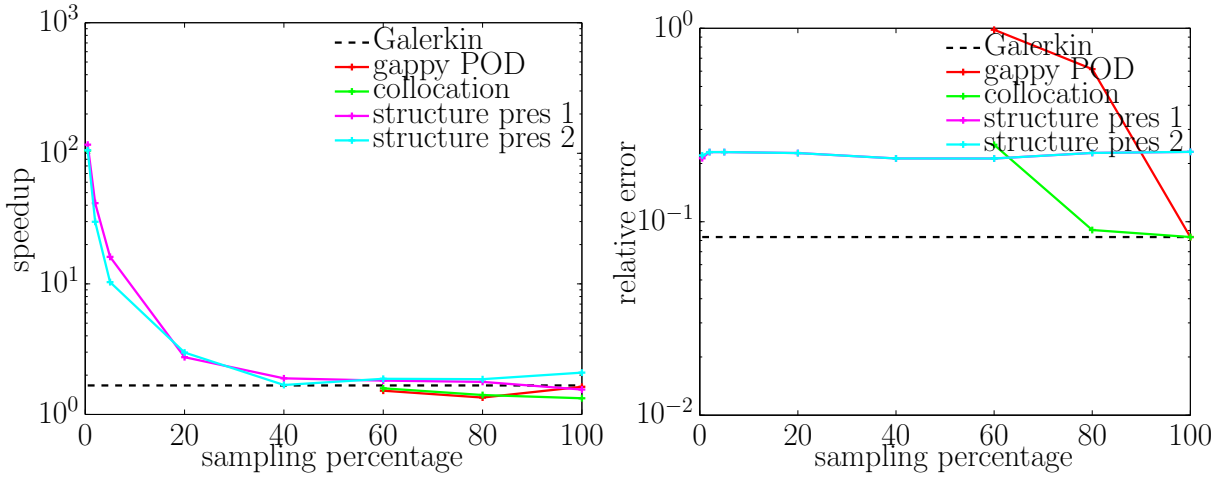


(b) predictions at online points 2 and 3, 60% sampling

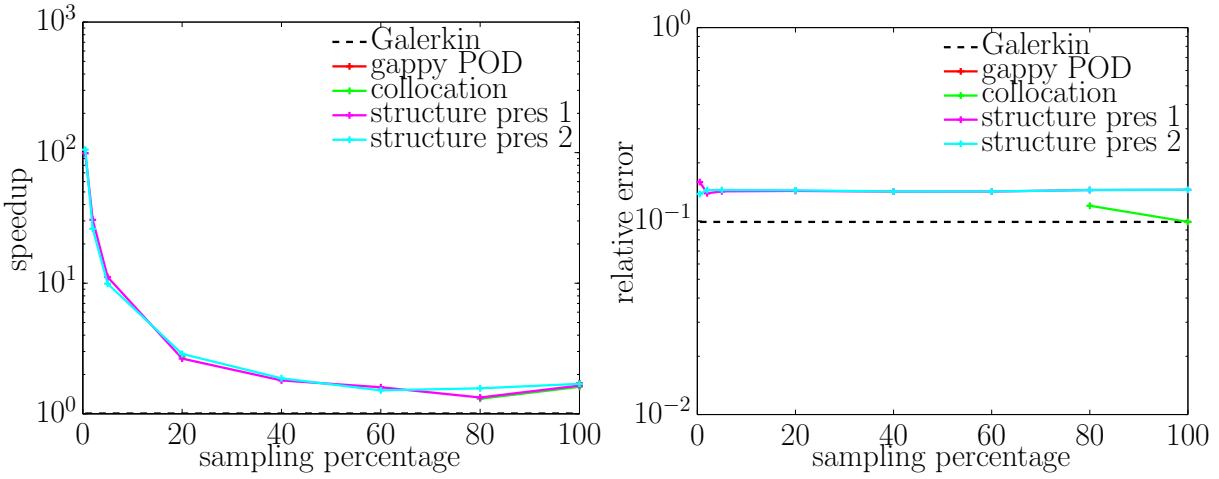


(c) predictions at online points 2 and 3, 80% sampling

FIG. S4.5. *Non-conservative, highly nonlinear parameter-varying case: reduced-order model responses as a function of sampling percentage $m/N \times 100\%$. Legend: full-order model (black), Galerkin ROM (dark blue), structure-preserving ROM method 1 (magenta), structure-preserving ROM method 2 (light blue), gappy POD ROM (red), collocation ROM (green), end of training time interval (black circle).*



(a) online point 2



(b) online point 3

FIG. S4.6. *Non-conservative, highly nonlinear parameter-varying case: reduced-order model performance as a function of sampling percentage* $m/N \times 100\%$.

Chapter 2

Decreasing the temporal complexity for nonlinear, implicit reduced-order models by forecasting

This chapter presents a method for decreasing the *temporal complexity* of nonlinear reduced-order models. Here, we define temporal complexity as the number of Newton-like iterations executed over the course of a simulation. The methodology exploits time-domain data—not just spatial-domain data—along with the gappy POD approximation method to reduce this complexity. The method can be treated as a 'free accelerator' by nonlinear model-reduction methods, as it almost always leads to speedups (as we show in experiments), and does not incur any additional solution error. This work has been submitted to Computer Methods in Applied Mechanics and Engineering and is past the first round of revisions at the time of this writing.

Decreasing the temporal complexity for nonlinear, implicit reduced-order models by forecasting

Kevin Carlberg, Jaideep Ray, and Bart van Bloemen Waanders
Sandia National Laboratories[1]
7011 East Ave, MS 9159, Livermore, CA 94550

Abstract

Implicit numerical integration of nonlinear ODEs requires solving a system of nonlinear algebraic equations at each time step. Each of these systems is often solved by a Newton-like method, which incurs a sequence of linear-system solves. Most model-reduction techniques for nonlinear ODEs exploit knowledge of system's spatial behavior to reduce the computational complexity of each linear-system solve. However, the number of linear-system solves for the reduced-order simulation often remains roughly the same as that for the full-order simulation.

We propose exploiting knowledge of the model's temporal behavior to 1) forecast the unknown variable of the reduced-order system of nonlinear equations at future time steps, and 2) use this forecast as an initial guess for the Newton-like solver during the reduced-order-model simulation. To compute the forecast, we propose using the Gappy POD technique. The goal is to generate an accurate initial guess so that the Newton solver requires many fewer iterations to converge, thereby decreasing the number of linear-system solves in the reduced-order-model simulation.

Keywords: nonlinear model reduction, Gappy POD, temporal correlation, forecasting, initial guess

1. Introduction

High-fidelity physics-based numerical simulation has become an indispensable engineering tool across a wide range of disciplines. Unfortunately, such simulations often bear an extremely large computational cost due to the large-scale, nonlinear nature of many high-fidelity models. When an implicit integrator is employed to advance the solution in time (as is often essential, e.g., for stiff problems) this large cost arises from the need to solve a sequence of high-dimensional systems of nonlinear algebraic equations—one at each time step. As a result, individual simulations can take weeks or months to complete, even when high-performance computing resources are available. This renders such simulations impractical for time-critical and many-query applications. For example, uncertainty-quantification applications (e.g., Bayesian inference problems) call for hundreds or thousands of simulations (i.e., forward solves) to be completed in days or weeks; in-the-field analysis (e.g., guidance in-field data acquisition) requires near-real-time simulation.

Projection-based nonlinear model-reduction techniques have been successfully applied to decrease the computational cost of high-fidelity simulation while retaining high levels of accuracy. To accomplish this, these methods exploit knowledge of the system's dominant *spatial behavior*—as observed during 'training simulations' conducted *a priori*—to decrease the simulation's *spatial complexity*, which we

[1]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

define as the computational cost of each linear-system solve.[2] To do so, these methods 1) decrease the dimensionality of the linear systems by projection, and 2) approximate vector-valued nonlinear functions by sampling methods that compute only a few of the vector's entries (e.g., empirical interpolation [1, 2], Gappy POD [3]). However, these techniques are often insufficient to adequately reduce the computational cost of the simulation. For example, Ref. [4] presented results for the GNAT nonlinear model-reduction technique applied to a large-scale nonlinear turbulent-flow problem. The reduced-order model generated solutions with sub-1% errors, reduced the spatial complexity by a factor of 637, and employed only 4 computing cores—a significant reduction from the 512 cores required for the high-fidelity simulation. However, the total number of linear-system solves required for the reduced-order-model simulation, which we define as the *temporal complexity*, remained large. In fact, the temporal complexity was decreased by a factor of only 1.5. As a result, the total computing resources (computing cores $\times$ wall time) required for the simulation were decreased by a factor of 438, but the wall time was reduced by a factor of merely 6.9. While these results are promising (especially in their ability to reduce spatial complexity), the time integration of nonlinear dynamics remains problematic and often precludes real-time performance.

The goal of this work is exploit knowledge of the system's *temporal behavior* as observed during the training simulations to decrease the temporal complexity of reduced-order-model simulations. For this purpose, we first briefly review methods that exploit observed temporal behavior to improve computational performance.

Temporal forecasting techniques have been investigated for many years with a specific focus on reducing wall time in a stable manner with maximal accuracy. The associated body of work is large and a comprehensive review is beyond the scope of this paper. However, this work focuses on time integration for reduced-order models of highly nonlinear dynamical systems; several categories of specialized research efforts provide an appropriate context for this research.

At the most fundamental level of temporal forecasting, a variety of statistical time-series-analysis methods exist that exploit 1) knowledge of the temporal structure, e.g., smoothness, of a model's variables, and 2) previous values of these variables for the current time series or trajectory. The connection between these methods and our work is that such forecasts can serve as an initial guess for an iterative solver (e.g., Newton's method) at an advanced point in time. However, the disconnect between such methods and the present context is that randomness and uncertainty drive time-series analysis; as such, these forecasting methods are stochastic in nature (see Refs. [5, 6, 7, 8, 9, 10, 11, 12]). In addition, the majority of time-series analyses have been applied to application domains (e.g., economics) with dynamics that are not generally modeled using partial differential equations. Finally, such forecasting techniques do not exploit a collection of observed, complete time histories from training experiments conducted *a priori*. Because such training simulations lend important insight into the spatial and temporal behavior of the model, we are interested in developing a technique that can exploit such data.

Alternatively, time integrators for ordinary differential equations (ODEs) employ polynomial extrapolations to provide reasonably accurate forecasts of the state or the unknown at each time step. Time integrators employ such a forecast for two purposes. First, algorithms with adaptive time steps employ interpolation to obtain solutions (and their time derivatives) at arbitrary points in time. Implicit time integrators for nonlinear ODEs, which require the iterative solution of nonlinear algebraic systems at each time step, use recent history (of the current trajectory) to forecast an accurate guess of the unknown in the algebraic system (see, e.g., Ref. [13]). Again, forecasting by polynomial extrapolation makes no use of the temporal behavior observed during training simulations.

Closely connected to time integration but specialized to leverage developments in high-performance

[2] A sequence of linear systems arises at each time step when a Newton-like method is employed to solve the system of nonlinear algebraic equations.

computing, time parallel methods can offer computational speedup when integrating ODEs. Dating back to before the general availability of parallel computers, researchers speculated about the benefits of decomposing the temporal domain across multiple processors [14]. Advancements have been made from parallel multigrid to parareal techniques [15, 16, 17, 18]. Although time-domain decomposition algorithms have demonstrated speedup, they are limited in comparison to the spatial domain decomposition methods and they require a careful balance between stability and computational efficiency [19]. It is possible that these methods could further improve performance in a model-reduction setting [20] (and could complement the method proposed in this work), but near real-time performance is likely unachievable through time-parallel methods alone.

To some extent, exploiting temporal behavior has been explored in nonlinear model reduction. Bos et al. [21] proposed a reduced-order model in the context of explicit time integration wherein the generalized coordinates are computed based on a best-linear-unbiased (BLU) estimate approach. Here, the reduced state coordinates at time step $n+1$ are computed using empirically derived correlations between the reduced state coordinates and 1) their value at the previous time step, 2) the forcing input at the previous time step, and 3) a subset of the full-order state. However, the errors incurred by this time-integration procedure (compared with standard time integration of the reduced-order model) are not assessed or controlled. This can be problematic in realistic scenarios, where error estimators and bounds are essential. Another class of techniques known as *a priori* model reduction methods [22, 23] build a reduced-order model 'on the fly', i.e., over the course of a given time integration. These techniques aim to use the reduced-order model at as many time steps as possible; they revert to the high-fidelity model when the reduced-order model is deemed to be inaccurate. In effect, these techniques employ the reduced-order model as a tool to accelerate the high-fidelity-model simulation. In contrast, this work aims to accelerate the reduced-order-model simulation itself. Further, these methods differ from the present context in that there are no training experiments conducted *a priori* from which to glean insight into the model's temporal behavior.

In this work, we propose a method that exploits a set of complete trajectories observed during training simulations to decrease the temporal complexity of a reduced-order-model simulation. The method 1) forecasts the unknown variable in the reduced-order system of nonlinear algebraic equations, and 2) uses this forecast as an initial guess for the Newton-like solver. To compute the forecast, the method employs the Gappy POD method [3], which extrapolates the unknown variable at future time steps by exploiting the unknown variable for the previous α time steps (where α can be interpreted as the memory of the process), and a database of time histories of the unknown variable. If the forecast is accurate, then the Newton-like solver will require very few iterations to converge, thereby decreasing the number of linear-system solves needed for the simulation. The method is straightforward to implement: the (offline) training stage simply requires collecting an additional set of snapshots during the training simulations. In some scenarios, no additional offline work is required. The (online) reduced-order-model simulation simply requires an external routine for determining the initial guess for the Newton-like solver.

2. Problem formulation

This section provides the context for this work. Section 2.1 describes the class of full-order models we consider, which includes first- and second-order ODEs numerically solved by implicit time integration. Section 2.2 describes the reduced-order modeling strategies for which the proposed technique is applicable.

2.1. Full-order model

2.1.1. First- and second-order ODEs

First, consider the parameterized nonlinear first-order ODE corresponding to the full-order model of a dynamical system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}; t, \boldsymbol{p}\left(t\right), \boldsymbol{q}\right) \tag{1}$$

$$\boldsymbol{x}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{x}^0\left(\boldsymbol{q}\right). \tag{2}$$

Here, time is denoted by $t \in [0, T]$, the time-dependent forcing inputs are denoted by $\boldsymbol{p} : [0, T] \to \mathbb{R}^{\mathrm{p}}$, the time-independent parametric inputs are denoted by $\boldsymbol{q} \in \mathcal{D} \subseteq \mathbb{R}^{\mathrm{q}}$ with $\mathcal{D}$ denoting the parameter domain, and $\boldsymbol{f} : \mathbb{R}^N \times [0, T] \times \mathbb{R}^{\mathrm{p}} \times \mathbb{R}^{\mathrm{q}} \to \mathbb{R}^N$ is nonlinear in at least its first argument. The state is denoted by $\boldsymbol{x} \equiv \boldsymbol{x}(t, \boldsymbol{p}, \boldsymbol{q}) \in \mathbb{R}^N$ with N denoting the number of degrees of freedom in the model. The parameterized initial condition is $\boldsymbol{x}^0 : \mathbb{R}^{\mathrm{p}} \to \mathbb{R}^N$.

Because this work addresses both first- and second-order ODEs, consider also the parameterized nonlinear second-order ODE corresponding to the full-order model of a dynamical system:

$$\ddot{\boldsymbol{x}} = \boldsymbol{g}\left(\boldsymbol{x}, \dot{\boldsymbol{x}}; t, \boldsymbol{p}\left(t\right), \boldsymbol{q}\right) \tag{3}$$

$$\boldsymbol{x}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{x}^0\left(\boldsymbol{q}\right) \tag{4}$$

$$\dot{\boldsymbol{x}}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{v}^0(\boldsymbol{q}). \tag{5}$$

Here, the function $\boldsymbol{g} : \mathbb{R}^N \times \mathbb{R}^N \times [0, T] \times \mathbb{R}^{\mathrm{p}} \times \mathbb{R}^{\mathrm{q}} \to \mathbb{R}^N$ is nonlinear in at least its first or second argument, and the parameterized initial velocity is denoted by $\boldsymbol{v}^0 : \mathbb{R}^{\mathrm{p}} \to \mathbb{R}^N$.[3]

2.1.2. Implicit time integration

Given forcing and parametric inputs, the numerical solution to the full-order model described by Eqs. (1)–(2) or (3)–(5) can be computed via numerical integration. For stiff systems, an implicit integration method is often the most computationally efficient choice; it is even essential in many cases [24]. When an implicit time integrator is employed, s coupled N-dimensional systems of nonlinear algebraic equations are solved at each time step $n = 1, \ldots, M$, where M denotes the total number of time steps:

$$\boldsymbol{r}_i^n\left(\boldsymbol{w}^{n,1}, \ldots, \boldsymbol{w}^{n,s}; \boldsymbol{p}, \boldsymbol{q}\right) = 0, \quad i = 1, \ldots, s. \tag{6}$$

Here, the function $\boldsymbol{r}_i^n : \mathbb{R}^N \times \cdots \times \mathbb{R}^N \times \mathbb{R}^{\mathrm{p}} \times \mathbb{R}^{\mathrm{q}} \to \mathbb{R}^N$ is nonlinear in at least one of its first s arguments and the unknowns $\boldsymbol{w}^{n,i} \in \mathbb{R}^N$, $i = 1, \ldots, s$ are implicitly defined by (6). As discussed in Appendix A and Appendix B, the unknowns $\boldsymbol{w}^{n,i}$ represent the state, velocity, or acceleration at points $t^{n-1} + c_i h^n$, where $c_i \in [0, 1]$ is defined by the time integrator:

$$\boldsymbol{w}^{n,i} \equiv \boldsymbol{w}^{n,i}(\boldsymbol{p}, \boldsymbol{q}) \equiv \boldsymbol{w}(t^{n-1} + c_i h^n; \boldsymbol{p}, \boldsymbol{q}). \tag{7}$$

Thus, a superscript n denotes the value of a quantity at time $t^n \equiv \sum_{k=1}^{n} h^k$, a superscript n, i denotes the value of a quantity at time $t^{n,i} \equiv \sum_{k=1}^{n-1} h^k + c_i h^n$, and h denotes the time-step size.

After the unknowns are computed by solving Eq. (6), the state is explicitly updated as

$$\boldsymbol{x}^n = \gamma \boldsymbol{x}^{n-1} + \sum_{i=1}^{s} \delta_i \boldsymbol{w}^{n,i}, \tag{8}$$

[3] Note that an N-dimensional second-order ODE can be rewritten as $2N$-dimensional first-order ODE.

where γ and δ_i, $i = 1, \ldots, s$ are scalars defined by the integrator. For second-order ODEs, the velocity is also updated explicitly as

$$\dot{\boldsymbol{x}}^n = \epsilon \dot{\boldsymbol{x}}^{n-1} + \sum_{i=1}^{s} \xi_i \boldsymbol{w}^{n,i}, \tag{9}$$

where ϵ and ξ_i, $i = 1, \ldots, s$ are also scalars defined by the integrator. Appendix A and Appendix B specify the form of Eqs. (6)–(9) for important classes of implicit numerical integrators for first- and second-order ODEs, respectively.

The chief computational burden of solving Eq. (1) with an implicit integrator lies in solving nonlinear equations (6) at each time step; this is typically done with a Newton-like method. In particular, if $\bar{K}$ denotes the average number of Newton-like iterations required to solve (6), then the full-order-model simulation requires solving $\bar{K}M$ linear systems of dimension sN.[4] We denote the simulation's *spatial complexity* to be the computational cost of solving each linear system; we consider the simulation's *temporal complexity* to be the total number of linear-system solves.

The spatial complexity contributes significantly to the computational burden for large-scale systems because N is large. However, the temporal complexity is also significant for such problems. First, the number of total time steps M is often proportional to a fractional power of N. This occurs because refining the mesh in space often necessitates a decrease in the time-step size to balance the spatial and temporal errors.[5] Second, the average number of Newton-like iterations $\bar{K}$ can be large when the problem is highly nonlinear and large time steps are taken, which is common for implicit integrators. Under these conditions, the initial guess for the Newton solver, which is often taken to be a polynomial extrapolation of the unknown, can be far from the true value of the unknown.

In many cases (e.g., linear multi-step methods, single-stage Runge–Kutta schemes), $s = 1$. For this reason, and for the sake of notational clarity, the remainder of this paper assumes $s = 1$, and $\boldsymbol{w}^n$ designates the value of the unknown variable at time $t^{n,1}$. However, we note that the proposed technique can be straightforwardly extended to $s > 1$.

2.2. Reduced-order model

Nonlinear model-reduction techniques aim to generate a low-dimensional model that is inexpensive to evaluate, yet captures key features of the full-order model. To do so, these methods first perform analyses of the full-order model for a set of n_{train} training parametric and forcing points $\{(\boldsymbol{p}^k, \boldsymbol{q}^k)\}_{k=1}^{n_{\text{train}}}$ during a computationally intensive 'offline' training stage. These analyses may include integrating the equations of motion, modal decomposition, etc.

Then, the data generated during these analyses are employed to decrease the the cost of each linear-system solve via two approximations: 1) dimensionality reduction, 2) nonlinear-function approximation (spatial-complexity reduction). Once these approximations are defined, the resulting reduced-order model is employed to perform computationally inexpensive analyses for any inputs during the 'online' stage.

[4] Assuming the Jacobian of the residual is sparse with an average number of nonzeros per row $\omega \ll N$, the dominant computational cost of solving Eqs. (6) for the entire simulation is $\mathcal{O}\left(\omega^2 sNKM\right)$ if a direct linear solver is used. It is $\mathcal{O}\left(L\omega sNKM\right)$ if an iterative linear solver is used. Here, L denotes the average number of matrix-vector products required to solve each linear system in the case of an iterative linear solver.

[5] This is not necessarily true for explicit time-integration schemes, when the time-step size is limited by stability rather than accuracy. In this case, Krysl et al. [25] showed that employing a low-dimensional subspace for the state may improve stability and therefore permit a larger time-step size. As a result, the reduced-order state equations can be solved fewer times than the full-order state equations.

2.2.1. Dimensionality reduction

Model-reduction techniques decrease the number of degrees of freedom by computing an approximate state $\tilde{\boldsymbol{x}} \approx \boldsymbol{x}$ that lies in an affine trial subspace of dimension $\hat{N} \ll N$:

$$\tilde{\boldsymbol{x}}(t, \boldsymbol{p}, \boldsymbol{q}) = \bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}(t, \boldsymbol{p}, \boldsymbol{q}) \tag{10}$$

$$\dot{\tilde{\boldsymbol{x}}}(t, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{\Phi}\dot{\hat{\boldsymbol{x}}}(t, \boldsymbol{p}, \boldsymbol{q}) \tag{11}$$

$$\ddot{\tilde{\boldsymbol{x}}}(t, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{\Phi}\ddot{\hat{\boldsymbol{x}}}(t, \boldsymbol{p}, \boldsymbol{q}). \tag{12}$$

Here, the trial basis (in matrix form) is denoted by $\boldsymbol{\Phi} \equiv \begin{bmatrix}\boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_{\hat{N}}\end{bmatrix} \in \mathbb{R}^{N \times \hat{N}}$ with $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{I}$. The generalized state is denoted by $\hat{\boldsymbol{x}} \equiv \begin{bmatrix}\hat{x}_1 & \cdots & \hat{x}_{\hat{N}}\end{bmatrix}^T \in \mathbb{R}^{\hat{N}}$. The reference state is $\bar{\boldsymbol{x}} \in \mathbb{R}^N$, which is often set to zero. The initial condition for the reduced-order model is obtained by projecting the full-order-model initial condition onto this affine subspace such that

$$\tilde{\boldsymbol{x}}(0, \boldsymbol{p}, \boldsymbol{q}) = \bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\left(\boldsymbol{x}^0(\boldsymbol{q}) - \bar{\boldsymbol{x}}(\boldsymbol{q})\right) \tag{13}$$

$$\dot{\tilde{\boldsymbol{x}}}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{v}^0(\boldsymbol{q}). \tag{14}$$

When the unknown variable computed at each time step (see Section 2.1.2) corresponds to the state, velocity, or acceleration, this dimensionality reduction for the state results in the following dimensionality reduction for the unknown:

$$\tilde{\boldsymbol{w}}(t, \boldsymbol{p}, \boldsymbol{q}) = \bar{\boldsymbol{w}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{w}}(t, \boldsymbol{p}, \boldsymbol{q}), \tag{15}$$

where $\bar{\boldsymbol{w}}(\boldsymbol{q}) = \bar{\boldsymbol{x}}(\boldsymbol{q})$ if the unknown is the state and $\bar{\boldsymbol{w}}(\boldsymbol{q}) = 0$ otherwise, and $\hat{\boldsymbol{w}} \equiv \begin{bmatrix}\hat{w}_1 & \cdots & \hat{w}_{\hat{N}}\end{bmatrix}^T \in \mathbb{R}^{\hat{N}}$ denotes the vector of generalized unknowns.

Substituting Eqs. (10)–(11) into (1) yields

$$\boldsymbol{\Phi}\dot{\hat{\boldsymbol{x}}} = \boldsymbol{f}\left(\bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}; t, \boldsymbol{p}(t), \boldsymbol{q}\right), \tag{16}$$

Alternatively, substituting Eq. (10)–(12) into (3) yields

$$\boldsymbol{\Phi}\ddot{\hat{\boldsymbol{x}}} = \boldsymbol{g}\left(\bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}, \boldsymbol{\Phi}\dot{\hat{\boldsymbol{x}}}; t, \boldsymbol{p}(t), \boldsymbol{q}\right). \tag{17}$$

The overdetermined ODEs described by (16) and (17) may not be solvable, because image($\boldsymbol{f}$) $\not\subset$ range($\boldsymbol{\Phi}$) and image($\boldsymbol{g}$) $\not\subset$ range($\boldsymbol{\Phi}$) in general. Several methods exist to compute an approximate solution.

Project, then discretize in time. This class of model-reduction methods first carries out a projection process on the ODE followed by a time-integration of the resulting low-dimensional ODE. The (Petrov–Galerkin) projection process enforces orthogonality of the residual corresponding to the overdetermined ODE (16) or (17) to an $\hat{N}$-dimensional test subspace range($\boldsymbol{\Psi}$), with $\boldsymbol{\Psi} \in \mathbb{R}^{N \times \hat{N}}$. Assuming $\boldsymbol{\Psi}^T\boldsymbol{\Phi}$ is invertible, this leads to the following for first-order ODEs:

$$\dot{\hat{\boldsymbol{x}}} = \left(\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{f}\left(\bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}; t, \boldsymbol{p}(t), \boldsymbol{q}\right). \tag{18}$$

For second-order ODEs, the result is

$$\ddot{\hat{\boldsymbol{x}}} = \left(\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{g}\left(\bar{\boldsymbol{x}}(\boldsymbol{q}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}, \boldsymbol{\Phi}\dot{\hat{\boldsymbol{x}}}; t, \boldsymbol{p}(t), \boldsymbol{q}\right), \tag{19}$$

Galerkin projection corresponds to the case where $\boldsymbol{\Psi} = \boldsymbol{\Phi}$.

Because Eq. (18) (resp. (19)) is an ODE of the same form as (1) (resp. (3)), it can be solved using the same numerical integrator that was used to solve Eq. (1) (resp. (3)). Further, the same time-step sizes are often employed, as the time-step size is determined by accuracy (not stability) for implicit time integrators. For both first- and second-order ODEs, this again leads to a system of nonlinear equations to be solved at each time step $n = 1, \ldots, M$:

$$\left(\boldsymbol{\Psi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{r}^n\left(\bar{\boldsymbol{w}}\left(\boldsymbol{q}\right)+\boldsymbol{\Phi}\hat{\boldsymbol{w}}^n;\boldsymbol{p},\boldsymbol{q}\right)=0. \tag{20}$$

The unknown $\hat{\boldsymbol{w}}^n$ can be computed by applying Newton's method to (20). Then, the explicit updates (8)–(9) can proceed as usual to compute the resulting state.

Discretize in time, then project. This class of model-reduction techniques first applies the same numerical integrator that was used to solve (1) to the overdetermined ODE (16) or (17). However, the resulting algebraic system of N nonlinear equations in $\hat{N}$ unknowns remains overdetermined:

$$\boldsymbol{r}^n\left(\bar{\boldsymbol{w}}\left(\boldsymbol{q}\right)+\boldsymbol{\Phi}\hat{\boldsymbol{w}}^n;\boldsymbol{p},\boldsymbol{q}\right)=0. \tag{21}$$

To compute a unique solution to Eq. (21), orthogonality of the discrete residual $\boldsymbol{r}^n$ to a test subspace range$\left(\boldsymbol{\Psi}\right)$ can be enforced. However, this leads to a reduced system of nonlinear equations equivalent to (20). So, in this case, the two classes of model-reduction techniques are equivalent.

On the other hand, to compute a unique solution to (21), the discrete-residual norm can be minimized [26, 4, 27, 28, 29], which ensures *discrete optimality* [4]:

$$\hat{\boldsymbol{w}}^n=\arg\min_{\boldsymbol{y}\in\mathbb{R}^{\hat{N}}}\left\|\boldsymbol{r}^n\left(\bar{\boldsymbol{w}}\left(\boldsymbol{q}\right)+\boldsymbol{\Phi}\boldsymbol{y};\boldsymbol{p},\boldsymbol{q}\right)\right\|_2^2. \tag{22}$$

The unknown $\hat{\boldsymbol{w}}^n$ can be computed by applying a Newton-like nonlinear least-squares method (e.g., Gauss–Newton, Levenberg–Marquardt) to problem (22). Again, explicit updates for the state (8)–(9) can proceed after the unknowns are computed.

2.2.2. Spatial-complexity reduction

For nonlinear dynamical systems, the dimensionality reduction described in Section 2.2.1 is insufficient to guarantee a reduction in the computational cost of each linear-system solve. The reason is that the full-order residual depends on the state, so it must be recomputed and subsequently projected or minimized at each Newton-like iteration.

For this reason, nonlinear model-reduction techniques employ a procedure to reduce the spatial-complexity, i.e., decrease the computational cost of computing and projecting or minimizing the nonlinear residual. Such techniques are occasionally referred to as 'hyper-reduction' techniques [22]. In particular, the class of 'function sampling' techniques replace the full-order nonlinear residual with an approximation $\tilde{\boldsymbol{r}}\approx\boldsymbol{r}$ that is inexpensive to compute. Then, $\boldsymbol{r}^n\leftarrow\tilde{\boldsymbol{r}}^n$ is employed in (20) or (22) to compute the unknowns $\hat{\boldsymbol{w}}^n$.

Methods in this class can be categorized as follows:

1. *Collocation approaches.* These methods employ a residual approximation that sets many of the residual's entries to zero:
$$\tilde{\boldsymbol{r}}^n=\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{r}^n. \tag{23}$$
Here, $\boldsymbol{Z}\in\{0,1\}^{n_Z\times N}$ is a sampling matrix consisting of $n_Z\ll N$ selected rows of $\boldsymbol{I}_{N\times N}$. This approach has been developed for Galerkin projection [30, 22] and discrete-residual minimization [29].

2. *Function-reconstruction approaches.* These methods employ a residual approximation that computes a few entries of the residual or nonlinear function, and subsequently 'fills in' the remaining entries via interpolation or least-squares regression. That is, these methods apply one of the following approximations:

$$\tilde{\boldsymbol{r}}^n = \boldsymbol{\Phi}_R \left(\boldsymbol{Z}\boldsymbol{\Phi}_R\right)^+ \boldsymbol{Z}\boldsymbol{r}^n \tag{24}$$

$$\tilde{\boldsymbol{f}} = \boldsymbol{\Phi}_f \left(\boldsymbol{Z}\boldsymbol{\Phi}_f\right)^+ \boldsymbol{Z}\boldsymbol{f} \tag{25}$$

$$\tilde{\boldsymbol{g}} = \boldsymbol{\Phi}_g \left(\boldsymbol{Z}\boldsymbol{\Phi}_g\right)^+ \boldsymbol{Z}\boldsymbol{g}. \tag{26}$$

Here, $\boldsymbol{\Phi}_R$, $\boldsymbol{\Phi}_f$, and $\boldsymbol{\Phi}_g$ are empirically derived bases used to approximate the nonlinear residual, velocity, and acceleration, respectively. A superscript + denotes the Moore–Penrose pseudoinverse. When the bases are computed via POD, this technique is known as Gappy POD [3]. This approach has been developed for Galerkin projection [30, 21, 2, 31, 32] and discrete-residual minimization [26, 4]. In particular, the discrete empirical interpolation (DEIM) method [2] is a specific case of Gappy POD for first-order ODEs, Galerkin projection, and the interpolatory case, i.e., DEIM uses approximation (25) in Eq. (18) with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$ and sets the number of sample indices n_Z equal to the number of columns in the matrix $\boldsymbol{\Phi}_f$. The GNAT method [26, 4] employs Gappy POD of the residual in a discrete residual minimization setting, i.e., GNAT uses approximation (24) in Eq. (22).

3. Temporal-complexity reduction

While the model-reduction approaches described in the previous section decrease the computational cost of each linear-system solve (i.e., spatial complexity), they do not necessarily decrease the number of linear-system solves (i.e., temporal complexity). The goal of this work is devise a method that decreases this temporal complexity while introducing no additional error.

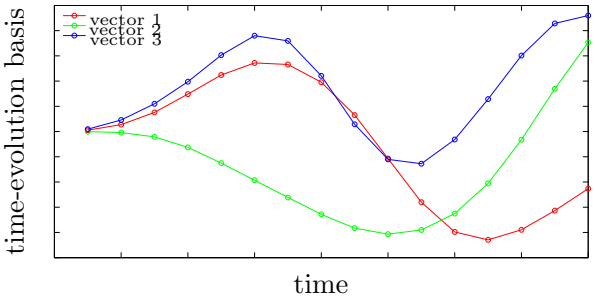
3.1. Method overview

The main idea of the proposed approach is to compute an accurate forecast of the generalized unknowns at future time steps using the Gappy POD procedure, and employ this forecast as an initial guess for the Newton-like solver at future time steps.

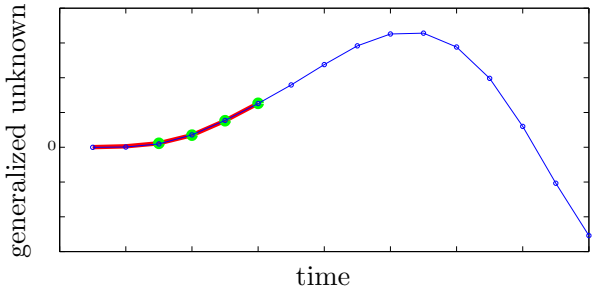
Gappy POD is a technique to reconstruct vector-valued data that has 'gaps,' i.e., entries with unknown or uncomputed values. Mathematically, the approach is equivalent to least-squares regression in one discrete-valued variable using empirically computed basis functions. It was introduced by Everson and Sirovich [3] for the purpose of image reconstruction. It has also been used for static [33, 34] and time-dependent [35, 36] flow field reconstruction, inverse design [34], design variable mapping for multi-fidelity optimization [37], and for decreasing the spatial complexity in nonlinear model reduction [30, 21, 26, 4]. This work proposes a novel application of Gappy POD: as a method for forecasting the generalized unknown at future time steps during a reduced-order-model simulation.

During the offline stage, the proposed method computes a 'time-evolution basis' for each generalized unknown $\hat{w}_j$, $j = 1, \ldots, \hat{N}$. Each basis represents the complete time-evolution of a generalized unknown as observed during training simulations. Figure 1(a) depicts this idea graphically, and Section 3.2 describes a computationally inexpensive way to compute these bases.

During the online stage, the method computes a forecast of the generalized unknowns at future time steps via Gappy POD. This forecast employs 1) the time-evolution bases and 2) the generalized unknowns computed at several previous time steps. Figure 1(b) depicts this, and Section 3.3 describes the forecasting method in detail. At future time steps, this forecast is employed as an initial guess for the Newton-like solver. If the forecast is accurate, the Newton-like solver will converge in very few iterations; if it is inaccurate, the Newton-like solver will require more iterations for convergence. Note



(a) Offline: the computed time-evolution POD basis for a generalized unknown.



(b) Online: time steps taken so far (red), recent time steps used to compute forecast (green), forecast (blue)

Figure 1: Graphical depiction of the proposed method

that the accuracy of the solution is not hampered in either case (assuming a globalization strategy is employed). If the number of Newton iterations required for convergence is large, this indicates an inaccurate initial guess. When this occurs, the method computes a new forecast using the most recently computed generalized unknowns.

The proposed method is expected to be effective if 1) the temporal behavior of the generalized unknowns is similar across input variation and 2) the original model is not too weakly nonlinear at each time step. The latter issue can hamper the proposed method's performance because it is difficult to reduce the number of Newton iterations if the original number is already very small. This situation can occur, for example, if the simulation employs a very small time step. However, this is uncommon for (unconditionally stable) implicit time integrators, where taking the largest time step while maintaining accuracy is typically the most computationally efficient approach.

The proposed method is independent of the dimensionality-reduction or spatial-complexity-reduction scheme employed by the reduced-order model; further, the method is applicable (without modification) to both first- and second-order ODEs. The next sections describe the offline and online steps of the methodology in detail.

3.2. Offline stage: compute the time-evolution bases

The objective of the offline stage is to compute the time-evolution bases that will be used for the online forecast. Ideally, the bases should be able to describe the time evolution of the generalized state for any forcing inputs $\boldsymbol{p}$ and parametric inputs $\boldsymbol{q}$. If the bases are 'bad', then the forecasting step of the algorithm will be inaccurate, and there may be no reduction in the average number of Newton-like iterations.

We propose employing a POD basis for the time evolution of the generalized unknown. This basis is computed *a priori* during 'offline' simulations of the reduced-order model in three steps:

1. Collect snapshots of the unknown during each of the n_{train} training simulations:
$$\boldsymbol{Y}_k = \left[\boldsymbol{w}^0\left(\boldsymbol{p}^k, \boldsymbol{q}^k\right) \ \cdots \ \boldsymbol{w}^{M-1}\left(\boldsymbol{p}^k, \boldsymbol{q}^k\right)\right] \tag{27}$$
for $k = 1, \ldots, n_{\text{train}}$, with $\boldsymbol{Y}_k \in \mathbb{R}^{N\times M}$. Here, $\boldsymbol{p}^k \in \mathbb{R}^{\mathrm{p}}$ denotes the forcing inputs for training simulation k, and $\boldsymbol{q}^k \in \mathbb{R}^{\mathrm{q}}$ denotes the parametric inputs for training simulation k.
2. Compute the corresponding snapshots of the generalized unknown:
$$\hat{\boldsymbol{Y}}_k \equiv \boldsymbol{\Phi}^T\left[\boldsymbol{Y}_k - \bar{\boldsymbol{w}}\left(\boldsymbol{q}_k\right)\boldsymbol{1}^T\right] \tag{28}$$
$$= \left[\hat{\boldsymbol{w}}^0\left(\boldsymbol{p}^k, \boldsymbol{q}^k\right) \ \cdots \ \hat{\boldsymbol{w}}^{M-1}\left(\boldsymbol{p}^k, \boldsymbol{q}^k\right)\right] \tag{29}$$
for $k = 1, \ldots, n_{\text{train}}$, where orthogonality of the trial basis $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{I}$ has been used. Here, $\hat{\boldsymbol{Y}}_k \in \mathbb{R}^{\hat{N}\times M}$ and $\boldsymbol{1} \in \mathbb{R}^M$ denotes a vector of ones.
3. Compute the time-evolution bases via the (thin) singular value decomposition (SVD). Defining the jth column of $\hat{\boldsymbol{Y}}_k^T$ as $\hat{\boldsymbol{y}}_{j,k} \in \mathbb{R}^M$, $j = 1, \ldots, \hat{N}$, we note that $\hat{\boldsymbol{y}}_{j,k}$ can be interpreted as a snapshot of the time evolution of the jth generalized unknown $\hat{\boldsymbol{w}}_j$ during training simulation k. Then, this step amounts to
$$\left[\hat{\boldsymbol{y}}_{j,1} \ \cdots \ \hat{\boldsymbol{y}}_{j,n_{\text{train}}}\right] = \boldsymbol{U}_j\boldsymbol{\Sigma}_j\boldsymbol{V}_j^T \tag{30}$$
$$\boldsymbol{\Xi}_j = \left[\boldsymbol{u}_{j,1} \ \cdots \ \boldsymbol{u}_{j,a_j}\right], \tag{31}$$
for $j = 1, \ldots, \hat{N}$. Here, $\boldsymbol{U}_j \equiv [\boldsymbol{u}_{j,1} \ \cdots \ \boldsymbol{u}_{j,n_{\text{train}}}] \in \mathbb{R}^{M\times n_{\text{train}}}$ and $a_j \leq n_{\text{train}}$.

After the time-evolution bases $\boldsymbol{\Xi}_j \in \mathbb{R}^{M\times a_j}$, $j = 1, \ldots, \hat{N}$ have been computed during the offline stage, they can be used to accelerate online computations via forecasting. The next section describes this.

Remark. In some cases, many of the above offline steps are already completed as part of the existing model-reduction process. For example, the snapshot matrices $\boldsymbol{Y}_k$, $k = 1, \ldots, n_{\text{train}}$ in Step 1 are already available if proper orthogonal decomposition (POD) is employed to compute $\boldsymbol{\Phi}$ and the time integrator's unknown is the state (e.g., linear multistep schemes). If additionally $n_{\text{train}} = 1$ and the POD basis is computed via the SVD of the reference-centered state snapshots, i.e., $\left[\boldsymbol{x}^0\left(\boldsymbol{p}^1, \boldsymbol{q}^1\right) - \bar{\boldsymbol{x}}\left(\boldsymbol{q}^1\right) \ \cdots \ \boldsymbol{x}^{M-1}\left(\boldsymbol{p}^1, \boldsymbol{q}^1\right) - \bar{\boldsymbol{x}}\left(\boldsymbol{q}^1\right)\right] = \bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^T$ with $\boldsymbol{\phi}_i = \bar{\boldsymbol{u}}_i$, $i = 1, \ldots, \hat{N}$, then $\hat{\boldsymbol{Y}}_1$ of Step 2 is already available as $\hat{\boldsymbol{Y}}_1 = \bar{\boldsymbol{\Sigma}}[1:\hat{N}, 1:\hat{N}]\bar{\boldsymbol{V}}[1:M, 1:\hat{N}]^T$. Here, the square bracket indicates a submatrix over the specified range of row and column indices and $\bar{\boldsymbol{U}} \equiv [\bar{\boldsymbol{u}}_1 \ \cdots \ \bar{\boldsymbol{u}}_M]$. Further, in this case the matrices $\boldsymbol{U}_j$ in Step 3 are also available as $\boldsymbol{U}_j = \boldsymbol{u}_{j,1} = \bar{\boldsymbol{v}}_j$, $j = 1, \ldots, \hat{N}$, where $\bar{\boldsymbol{V}} \equiv [\bar{\boldsymbol{v}}_1 \ \cdots \ \bar{\boldsymbol{v}}_M]$.

3.3. Online stage: forecast

During the online stage, the method employs a forecasting procedure to define the initial guess for the Newton-like solver. To compute this forecast, it uses the time evolution bases (computed offline), and the values of the generalized unknown at the previous α time steps (computed online). Here, α is considered the 'memory' of the process. Because the forecast is defined at all time steps (see the blue curve in Figure 1(b)), it is used as the initial guess at future time steps until the number of Newton iterations exceeds a threshold value τ. This indicates a poor forecast. In this case, the forecast is recomputed using the most recent values of the generalized unknown.

Algorithm 1 Online: implicit time integration with the temporal-complexity-reduction method

Input: Time-evolution bases $\boldsymbol{\Xi}_j \in \mathbb{R}^{M \times a_j}$, $j = 1, \ldots, \hat{N}$; maximum memory $\alpha_{\max}$ with $\alpha_{\max} \geq \max_j a_j$; Newton-step threshold τ

Output: Generalized state at all M time steps: $\hat{\boldsymbol{x}}^n$, $n = 1, \ldots, M$.
Generalized velocity at all M time steps if solving a second-order ODE: $\dot{\hat{\boldsymbol{x}}}^n$, $n = 1, \ldots, M$.

1: **for** $n = 1, \ldots, M$ **do** {time-step loop}
2: **if** forecast $\underline{\hat{\boldsymbol{w}}}(t^{n-1} + c_1 h^n)$ is available **then**
3: Set initial guess for Newton solver to $\hat{w}_j^{n(0)} = \underline{\hat{w}}_j(t^{n-1} + c_1 h^n)$, $j = 1, \ldots, \hat{N}$.
4: **else**
5: Use typical initial guess for Newton solver (e.g., polynomial extrapolation of unknown).
6: **end if**
7: Compute generalized unknowns $\hat{\boldsymbol{w}}^n$ by solving reduced-order equations (20) or (22) with a Newton-like method and specified initial guess $\hat{\boldsymbol{w}}^{n(0)}$.
Let K^n denote the number of Newton-like iterations required for convergence at time step n.
8: Compute the generalized state $\hat{\boldsymbol{x}}^n$ using explicit update (8). If solving a second-order ODE, also update the generalized velocity $\dot{\hat{\boldsymbol{x}}}^n$ using explicit update (9).
9: **if** $K^n > \tau$ and $(n-1) \geq \max_j a_j$ **then** {recompute forecast using most recent data}
10: Set memory $\alpha \leftarrow \min(n-1, \alpha_{\max})$.
11: Compute forecasting coefficients $\boldsymbol{z}_j$, $j = 1, \ldots, \hat{N}$ using the unknown at the previous α time steps by solving Eq. (32).
12: Set forecast to be $\underline{\hat{\mathbf{w}}}_j = \boldsymbol{\Xi}_j \boldsymbol{z}_j$ and define $\underline{\hat{w}}_j \equiv \underline{h}^{-1}\left(\underline{\hat{\mathbf{w}}}_j\right)$, $j = 1, \ldots, \hat{N}$.
13: **end if**
14: **end for**

If the forecast is accurate, then the number of iterations needed to converge from the (improved) initial guess will be drastically reduced, thereby decreasing $\bar{K}$ and hence the temporal complexity. Algorithm 1 outlines the proposed technique.

To compute the forecasting coefficients in step 11 of Algorithm 1, we propose using the Gappy POD approach introduced by Everson and Sirovich [3]. This approach computes coefficients $\boldsymbol{z}_j$ via the following linear least-squares problem:

$$\boldsymbol{z}_j = \arg\min_{\boldsymbol{z} \in \mathbb{R}^{a_j}} \| \boldsymbol{Z}(n,\alpha)\boldsymbol{\Xi}_j \boldsymbol{z} - \boldsymbol{Z}(n,\alpha)\underline{h}\left(\hat{\boldsymbol{w}}_j\right) \| \tag{32}$$

Here, the matrix $\boldsymbol{Z}(n,\alpha) \in \{0,1\}^{\alpha \times M}$ is the sampling matrix that selects entries corresponding to the previous α time steps:

$$\boldsymbol{Z}(n,\alpha) \equiv \left[\boldsymbol{e}_{n-\alpha-1} \ \cdots \ \boldsymbol{e}_{n-1}\right]^T, \tag{33}$$

where $\boldsymbol{e}_i$ denotes the ith canonical unit vector. Note that $\alpha \geq a_j$ is required for Eq. (32) to have a unique solution. The function $\underline{h}$ in (32) 'unrolls' time according to the time discretization; we define $\underline{h} : x \mapsto \mathbf{x}$ with $\mathbf{x} \equiv \left[\mathrm{x}_1 \ \cdots \mathrm{x}_M\right]^T \in \mathbb{R}^M$ as

$$\mathrm{x}_n = x(t^{n-1} + c_1 h^n), \quad n = 1, \ldots, M. \tag{34}$$

The online cost to compute this forecast is very small, as it entails solving $\hat{N}$ small-scale linear least-squares problem (32) characterized by a $\alpha \times a_j$ matrix. For this reason, it is generally advantageous to employ a small value of τ (i.e., 0 or 1), which results in a frequent (inexpensive) recomputation of the forecast.

4. Numerical experiments

These numerical experiments assess the performance of the proposed temporal-complexity-reduction method on a structural-dynamics example using three reduced-order models: Galerkin projection (Eq. (19) with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$), Galerkin projection with least-squares reconstruction of the residual (Eq. (24)), and a structure-preserving reduced-order model [38]. We do not present results for a collocation ROM (see Eq. (23)), as this approach was unstable in most cases, even when 60% of the degrees of freedom were selected as sample indices (i.e., $n_Z/N = 0.6$). Section 4.1 provides a description of the problem—a parameterized, damped clamped–free truss structure subjected to external forces—and details the experimental setup. We then consider a sequence of problems that poses increasing difficulty to the method.

Section 4.2 considers the ideal scenario for the method: the online points are identical to the training points, and the reduced bases are not truncated. In this case, the temporal behavior of the system is perfectly predictable, because (in exact arithmetic) the online response is the same as the training response. Therefore, we expect the proposed method to work extremely well.

Section 4.3 assesses the method's performance in a more challenging setting. Here, the online points differ from the training points (i.e., a predictive scenario), so the temporal behavior is not identical to that observed during the training simulations. The parametric inputs correspond to shape parameters and the initial displacement. The external force is set to zero, which leads to a damped free-vibration problem. As a result, the dynamics encountered in this example are relatively smooth.

Section 4.4 considers a more challenging predictive scenario wherein rich dynamics—generated from a high-frequency external force—characterize the response. Here, additional parametric inputs are considered, which correspond to the magnitudes and frequencies of the high-frequency forces.

Section 4.5 increases the predictive difficulty, as the allowable range of the parametric inputs is doubled, leading to a more significant variation in the responses.

Finally, Section 4.6 summarizes the proposed forecasting method's performance over all experiments and tested reduced-order models.

4.1. Problem description

Figure 2 depicts the parameterized, non-conservative clamped–free truss structure we consider. The truss is parameterized by q = 16 parametric inputs $\boldsymbol{q} \equiv (q_1, \ldots, q_{16}) \in \mathcal{D} = [-0.5, 0.5]^{16}$ that affect the geometry, initial condition, and applied force as described in Table 1. We set the material

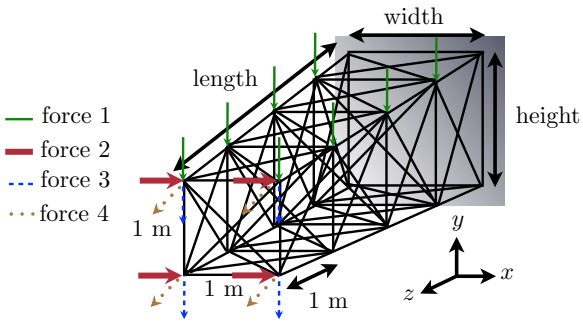


Figure 2: Clamped–free parameterized truss structure

properties to those of aluminum, i.e., density $\rho = 2700$ kg/m^3 and elastic modulus $E = 62 \times 10^9$ Pa. The external force is composed of four components:

$$\boldsymbol{f}_{\text{ext}}(\boldsymbol{q}, t) = \sum_{i=1}^{4} p_i(\boldsymbol{q}, t)\mathbf{r}_i, \tag{35}$$

length (m)	bar cross-sectional area (m^2)	width (m)	height (m)	initial condition max magnitude (N) s_i, $i = 1, \ldots, 4$	external-force magnitude γ_i, $i = 1, \ldots, 4$	external-force frequency λ_i, $i = 1, \ldots, 4$
$200 + 50q_1$	$0.0025(1 + 0.5q_2)$	$10(1 + q_3)$	$10(1 + q_4)$	$\underline{f}_i(1 + 0.5q_{i+4})$	$\underline{f}_i(1 + 0.5q_{i+8})$	$3\omega_0(1 + 0.5q_{i+12})$

Table 1: Effect of parameters on truss geometry, initial conditions, and applied forces. Here, $\underline{f}_i$, $i = 1, \ldots, 4$ denote the nominal force magnitudes (specified within each experiment) and ω_0 denotes the lowest-magnitude eigenvalue at the nominal point $\bar{\boldsymbol{q}}$.

where $\mathbf{r}_i \in \mathbb{R}^N$, $i = 1, \ldots, 4$ correspond to unit loads uniformly distributed across designated nodes and $p_i : \mathcal{D} \times [0, T] \to \mathbb{R}$, $i = 1, \ldots, 4$ denote the p = 4 forcing inputs. Figure 2 depicts the spatial distribution of the forces, which lead to vectors $\mathbf{r}_i$, $i = 1, \ldots, 4$ through the finite-element formulation described below. The parameterized, time-dependent magnitudes of these forces are

$$p_i(\boldsymbol{q}, t) = \begin{cases} \gamma_i(\boldsymbol{q}) \sin\left(\lambda_i(\boldsymbol{q})\,(t - T/4)\right), & t \geq T/4 \\ 0, \quad \text{otherwise} \end{cases}, \tag{36}$$

where $\gamma_i : \mathcal{D} \to \mathbb{R}$ and $\lambda_i : \mathcal{D} \to \mathbb{R}$, $i = 1, \ldots, 4$ denote the maximum force magnitudes and force frequencies, respectively. Similarly, the initial displacement is composed of four components $\boldsymbol{x}^0(\boldsymbol{q}) = \sum_{i=1}^{4} s_i(\boldsymbol{q})\boldsymbol{s}_i$, where $\boldsymbol{s}_i$ is the steady-state displacement of the truss subjected to load $\mathbf{r}_i\gamma_i(\bar{\boldsymbol{q}})$ with $\bar{\boldsymbol{q}} = (0, \ldots, 0)$ denoting the nominal point in parameter space. The initial velocity is set to zero $\boldsymbol{v}^0 = 0$, and the reference configuration is simply the undeformed truss (in equilibrium) represented by $\bar{\boldsymbol{x}} = 0$.

The problem is discretized by the finite-element method. The model consists of 16 three-dimensional bar elements per bay with three degrees of freedom per node; this results in 12 degrees of freedom per bay. We consider a problem with 250 bays, and therefore $N = 3 \times 10^3$ degrees of freedom in the full-order model. The bar elements model geometric nonlinearity, which results in a high-order nonlinearity in the internal force. This discretization results in the following equations of motion for the full-order model:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{x}} + \boldsymbol{f}_{\text{int}}(\boldsymbol{x}; \boldsymbol{q}) = \boldsymbol{f}_{\text{ext}}(t; \boldsymbol{q}). \tag{37}$$

Here, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{N \times N}$ denotes the symmetric-positive-definite mass matrix, the internal force is denoted by $\boldsymbol{f}_{\text{int}} : \mathbb{R}^N \times \mathcal{D} \to \mathbb{R}^N$, and the symmetric-positive-semidefinite Rayleigh viscous damping matrix, denoted by $\boldsymbol{C}(\boldsymbol{q}) \in \mathbb{R}^{N \times N}$, is of the form

$$\boldsymbol{C}(\boldsymbol{q}) = \alpha \boldsymbol{M}(\boldsymbol{q}) + \beta \nabla_{\boldsymbol{x}} \boldsymbol{f}_{\text{int}}\left(\boldsymbol{x}^0; \boldsymbol{q}\right). \tag{38}$$

Note that $\nabla_{\boldsymbol{x}} \boldsymbol{f}_{\text{int}}\left(\boldsymbol{x}^0; \boldsymbol{q}\right)$ represents the tangent stiffness matrix at the initial condition. Here, α and β are chosen such that the damping ratio is $\zeta = 15\,\text{deg}$ for the uncoupled ODEs associated with the smallest two eigenvalues of the matrix pencil $(\boldsymbol{M}(\bar{\boldsymbol{q}}), \nabla_{\boldsymbol{x}} \boldsymbol{f}_{\text{int}}(0; \bar{\boldsymbol{q}}))$ [39].

The equations of motion (37) can be rewritten in the standard form of Eqs. (3)–(5) as

$$\ddot{\boldsymbol{x}} = \boldsymbol{M}(\boldsymbol{q})^{-1}\left(\boldsymbol{f}_{\text{ext}}(t; \boldsymbol{q}) - \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{x}} - \boldsymbol{f}_{\text{int}}(\boldsymbol{x}; \boldsymbol{q})\right) \tag{39}$$

$$\boldsymbol{x}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{x}^0(\boldsymbol{q}) \tag{40}$$

$$\dot{\boldsymbol{x}}(0, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{v}^0(\boldsymbol{q}). \tag{41}$$

The nonlinear function defining the acceleration for the second-order ODE is then

$$\boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}}; t, \boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{M}(\boldsymbol{q})^{-1}\left(\boldsymbol{f}_{\text{ext}}(t; \boldsymbol{q}) - \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{x}} - \boldsymbol{f}_{\text{int}}(\boldsymbol{x}; \boldsymbol{q})\right). \tag{42}$$

We employ an implicit Nyström time integrator with constant timestep size $h = h^n$, $n = 1, \ldots, M$ to compute the numerical solution to Eqs. (39)–(41) in the time interval $[0, T]$ with $T = 25$ seconds. In particular, we employ the implicit midpoint rule for both partitions. This leads to discrete equations

(B.2) to be solved at each time step with explicit updates (B.3)–(B.4) and parameters $s = 1$, $\hat{a}_{11} = 1/2$, $\bar{a}_{11} = 1/4$, $\hat{b}_1 = 1$, $\bar{b}_1 = 1/2$, $c_1 = 1/2$. The unknowns are equivalent to the acceleration at the half time steps: $\boldsymbol{w}^n = \ddot{\boldsymbol{x}}\left(t^{n-1} + 1/2h\right)$, $n = 1, \ldots, M$. Multiplying the corresponding residual by $\boldsymbol{M}(\boldsymbol{q})$ yields

$$\boldsymbol{r}^n(\boldsymbol{w}^n) = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{w}^n + \boldsymbol{C}(\boldsymbol{q})\left[\dot{\boldsymbol{x}}^{n-1} + \frac{1}{2}h\boldsymbol{w}^n\right] + \boldsymbol{f}_{\text{int}}\left(\boldsymbol{x}^{n-1} + \frac{1}{2}h\dot{\boldsymbol{x}}^{n-1} + \frac{1}{4}h^2\boldsymbol{w}^n; \boldsymbol{q}\right) - \boldsymbol{f}_{\text{ext}}(t^{n-1} + \frac{1}{2}h; \boldsymbol{q}). \tag{43}$$

To solve $\boldsymbol{r}^n(\boldsymbol{w}^n) = 0$ at each time step, We employ a globalized Newton solver with a More–Thuente linesearch [40]. Except when noted, convergence of the Newton iterations is declared when the residual norm reaches 10^{-6} of its value computed using a zero acceleration and the values of the displacement and velocity at the beginning of the timestep. The linear system arising at each Newton iteration is solved directly.

The experiments compare the performance of three reduced-order models: Galerkin projection (Eq. (20) with $\boldsymbol{\Psi} = \boldsymbol{\Phi}$), Galerkin projection with Gappy POD residual approximation (Eq. (24)), and a model-reduction method based that preserves the classical Lagrangian structure intrinsic to the problem (Ref. [38], proposal 1). To construct the reduced-order models, we collect snapshots of the required quantities for $\boldsymbol{q} \in \mathcal{D}_{\text{train}} \subset \mathcal{D}$ and $t \in [0, T]$. The trial basis $\boldsymbol{\Phi}$ is determined via POD. We collect snapshots of the state

$$\mathcal{X}_{\boldsymbol{x}} = \{\boldsymbol{x}^{n-1} + h\dot{\boldsymbol{x}}^{n-1} + \frac{h}{2}\ddot{\boldsymbol{x}}^{n,1} \mid n = 1, \ldots, M;\ \boldsymbol{q} \in \mathcal{D}_{\text{train}}\} \tag{44}$$

and set the trial basis to $\boldsymbol{\Phi} = \boldsymbol{\Phi}^e(\mathcal{X}_{\boldsymbol{x}}, \nu_{\boldsymbol{x}})$, where $\nu_{\boldsymbol{x}} \in [0, 1]$ is an 'energy criterion' and $\boldsymbol{\Phi}^e$ is defined by Algorithm 2 in Appendix C. The reference state is set to $\bar{\boldsymbol{x}} = 0$, as this is the equilibrium state for this problem [38]. For Galerkin projection with least-squares (Gappy POD) residual reconstruction, the following snapshots are collected during the (full-order model) training simulations:

$$\mathcal{X}_{\boldsymbol{r}} = \{\boldsymbol{r}^n\left(\boldsymbol{w}^{n(k)}\right) \mid n = 1, \ldots, M;\ k = 0, \ldots, K^n - 1;\ \boldsymbol{q} \in \mathcal{D}_{\text{train}}\}. \tag{45}$$

Here, K^n denotes the number of Newton steps taken at time step n. The residual basis is set to $\boldsymbol{\Phi}_R = \boldsymbol{\Phi}^e(\mathcal{X}_{\boldsymbol{r}}, \nu_{\boldsymbol{r}})$ with $\nu_{\boldsymbol{r}} \in [0, 1]$. For the structure-preserving method, we also collect snapshots of both the mass matrix and the external forcing vector:

$$\mathcal{X}_{\boldsymbol{M}} = \{\boldsymbol{M}(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathcal{D}_{\text{train}}\} \tag{46}$$

$$\mathcal{X}_{\boldsymbol{f}_{\text{ext}}} = \{\boldsymbol{f}_{\text{ext}}(t^n; \boldsymbol{q}) \mid n = 1, \ldots, M;\ \boldsymbol{q} \in \mathcal{D}_{\text{train}}\}. \tag{47}$$

The POD basis for the external force employed by the structure-preserving method is set to $\boldsymbol{\Phi}_{\boldsymbol{f}_{\text{ext}}} = \boldsymbol{\Phi}^e\left(\mathcal{X}_{\boldsymbol{f}_{\text{ext}}}, \nu_{\boldsymbol{f}_{\text{ext}}}\right)$ with $\nu_{\boldsymbol{f}_{\text{ext}}} \in [0, 1]$

Reduced-order models with spatial-complexity reduction employ the same sampling matrix $\boldsymbol{Z}$, which is generated using GNAT's greedy sample-mesh algorithm [4, Algorithm 3].[6] These models are also implemented using the sample-mesh concept [4, Section 5]. For the structure-preserving method [38], we solve the reduced-basis-sparsification unconstrained optimization problem using the Poblano toolbox [40].[7]

[6] Greedy-algorithm parameters are $\boldsymbol{\Phi}_R = \boldsymbol{\Phi}_J = \boldsymbol{\Phi}^e_{\mathbf{r}}$ a POD basis computed using Algorithm 2 with snapshots of the numerical residual over all timesteps and Newton iterations during the full-order-model training simulations and an energy criterion of $\nu \leftarrow \nu_{\mathbf{r}} = 1 - 10^{-2}$, a target number of sample nodes $n_s = n_Z/\nu$ with $\nu = 3$ unknowns per node (the x-, y-, and z-displacements), an empty seeded sample-node set $\mathcal{N} = \emptyset$, and n_c equal to the number of columns in $\boldsymbol{\Phi}^e_{\mathbf{r}}$.

[7] The initial guess for each of these problems is chosen as $\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{\Phi}$.

In all experiments, the proposed forecasting method employs untruncated time-evolution bases: $a_j = n_{\text{train}}$, $j = 1, \ldots, \hat{N}$. We compare its performance with that of the most common approach for generating an initial guess: a polynomial extrapolation of varying degree. Note that polynomial extrapolations of different degrees employ a different number of previous solutions to generate an initial guess; in our experiments, we associate the number of previous solutions employed with a 'memory' α. For example, a zeroth-order extrapolation requires the previous solution ($\boldsymbol{w}^{n(0)} = \boldsymbol{w}^{n-1}$), so $\alpha = 1$ in this case. When no previous solution is used (i.e., $\alpha = 0$), the polynomial-extrapolation approach uses $\ddot{\boldsymbol{x}}^{n,1} = \boldsymbol{w}^{n(0)} = 0$. In all experiments, the full-order model employs a zeroth-order extrapolation for the initial guess.

The output of interest is the y-displacement of the bottom-left node of the end face of the truss in Figure 2. We denote this (parameterized, time-dependent) quantity by $d \in \mathbb{R}$. To quantify the performance of the reduced-order models, the following metrics are used:

$$\varepsilon = \frac{\frac{1}{M}\sum_{n=0}^{M} |d^n - d^n_{\text{FOM}}|}{\max_n d^n_{\text{FOM}} - \min_n d^n_{\text{FOM}}} \tag{48}$$

$$\kappa = \frac{\bar{K}_{\text{FOM}}}{\bar{K}} \tag{49}$$

$$S = \frac{\text{T}_{\text{FOM}}}{\text{T}} \tag{50}$$

Here, error measure ε designates the scaled ℓ_1 norm of the discrepancy in the output predicted by a reduced-order model. The temporal-complexity-reduction factor is denoted by κ, where $\bar{K}$ denotes the average number of Newton-like steps taken per time step over the course of a simulation. The speedup is denoted by S with T denoting the wall time required for a simulation. A subscript 'FOM' denotes a quantity computed using the full-order model.

All computations are carried out in Matlab on a Mac Pro with 2 × 2.93 GHz 6-Core Intel Xeon processors and 64 GB of memory.

4.2. Ideal case: unforced, invariant inputs, no truncation of bases

This experiment explores the ideal case for the method: the online inputs equal the training inputs, and the bases are not truncated ($\nu_{\boldsymbol{x}} = \nu_{\boldsymbol{r}} = 1.0$). The resulting basis dimensions are $\hat{N} = 100$ for the reduced basis and 329 for the residual basis (i.e., $\boldsymbol{\Phi}_f \in \mathbb{R}^{N \times 329}$). In this scenario, the full-order model's temporal behavior encountered online is exactly the same as that observed during training simulation; for this reason, we expect the proposed method to perform very well. We consider a single configuration ($n_{\text{train}} = 1$) characterized by $q_i = 0$, $i = 1, \ldots, 9$ with no applied forcing $q_i = -2$, $i = 9, \ldots, 16$. The nominal forces that affect the initial condition (see Table 1) are set to $\underline{f}_1 = \underline{f}_2 = 2\text{kg} \times 9.81\text{m/s}^2$ and $\underline{f}_3 = \underline{f}_4 = 0.4\text{kg} \times 9.81\text{m/s}^2$. The time-step size is set to $h = 0.25$ seconds, leading to $M = 100$ total time steps. This value was determined by a timestep-verification study using a timestep-refinement factor of two; a timestep of 0.25 seconds led to an approximated rate of convergence in the output quantity d at the end of the time interval of 1.40 (which is reasonably close to the scheme's asymptotic rate of convergence of 2.0) and an approximated error in this quantity (computed via Richardson extrapolation) of 0.99%.

We assess the performance of the ROMs with spatial-complexity reduction (i.e., Gappy POD and the structure-preserving ROM) using two different sets of sample indices. First, we set the number of sample nodes equal to 20% of the total nodes in the mesh (i.e., $n_s = 200$), which leads to $n_Z = 600$. We also employ a sampling fraction of 5%, which leads to $n_Z = 150$. For the forecasting technique, the Newton-step threshold is set to $\tau = 0$ and the maximum memory is set to $\alpha_{\max} = 9$. For the 'no forecasting' case, we employ a zeroth-order polynomial extrapolation. For experiments in this section,

we declare the Newton iterations to have converged when the residual norm reaches 10^{-4} of its value computed using a zero acceleration and the values of the displacement and velocity at the beginning of the timestep.

The full-order-model simulation consumed 16.8 minutes and incurred 229 Newton iterations ($\bar{K}_{\text{FOM}} = 2.29$). Table 2 and Figure 3 report the results for the reduced-order models. First, note that the relative errors generated by Galerkin and Gappy POD ROM with 20% sampling are essentially zero. This is expected, because the reduced bases are not truncated and the inputs are fixed. Further, note that the Galerkin ROM without forecasting generates no speedup; this is expected because it is not equipped with a spatial-complexity-reduction technique (see Section 2.2.2). The other two techniques—which employ spatial-complexity-reduction approximations—lead to speedups. The exception is Gappy POD with 5% sampling, which yields an unstable response; this is depicted in Figure 3(a). For this reason, subsequent experiments employ a larger sampling fraction for the Gappy POD ROM compared with the structure-preserving ROM.

Importantly, note that the reduced-order models exhibit very little temporal-complexity reduction (i.e., $\kappa \approx 1.0$) in the absence of the proposed forecasting technique. When the models employ the proposed forecasting technique, the number of Newton iterations decreases, leading to temporal-complexity reductions of $\kappa = 114.5$ for the Galerkin ROM and $\kappa = 2.26$ and $\kappa = 2.25$ for the best-performing Gappy POD and structure-preserving ROMs, respectively. In turn, this leads to improved wall-time speedups in all cases.

The Galerkin ROM case presented here can be viewed as the best possible performance for the method (applied to this problem): the temporal behavior of the system is exactly predictable, as the inputs have not changed, and the reduced basis has not been truncated. So, the forecast is 'perfect' after only one time step for the Galerkin ROM. This means that for each time step after the first one, the initial guess generated by the forecasting method is equal to the solution at that time step, so no Newton steps are needed to compute the solution. As a result, no Newton iterations are carried out beyond the first time step. The next sections investigate the forecasting method's performance in the (more realistic) case of varying inputs and truncated bases.

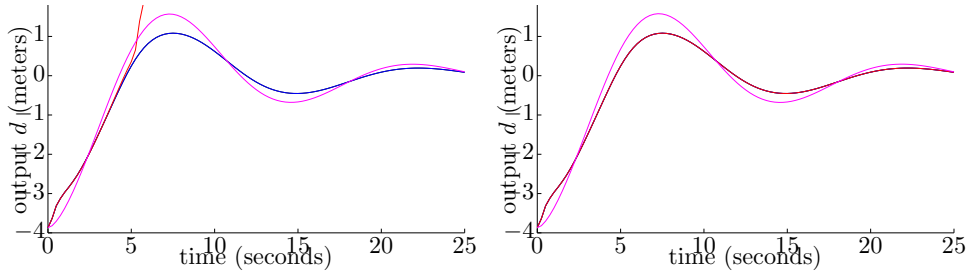
Remark. Note that the speedup (2.00) of the Galerkin ROM with forecating is not nearly as significant as the reduction factor (114.5), as Newton iterations are not the only aspect of the simulation that contribute to computational time. For example, the solution, velocity, and acceleration are updated at each time step, the residual is computed at each time step to check for convergence, outputs are computed, etc. We expect these two values to align more closely for problems where the computational cost of the Newton iterations dominates the overall simulation time.

ROM method	sampling fraction n_Z/N	relative error ε	No forecasting			With forecasting		
			Newton its $\bar{K}M$	speedup S	reduction factor κ	Newton its $\bar{K}M$	speedup S	reduction factor κ
Galerkin	-	8.93×10^{-6}	209	0.955	1.10	2	2.00	114.5
Gappy POD	0.2	1.60×10^{-5}	209	2.97	1.10	101	4.34	2.26
	0.05	unstable	-	-	-	-	-	-
structure preserving	0.2	5.06×10^{-2}	199	3.40	1.15	107	4.27	2.14
	0.05	4.98×10^{-2}	199	12.7	1.15	102	16.3	2.25

Table 2: Ideal case: forecast performance.

4.3. Unforced, varying inputs

We now consider a fully predictive scenario with $\boldsymbol{q}^\star \notin \mathcal{D}_{\text{train}}$. Again, we set the forces to zero, which implies $q_i = -2$, $i = 9, \ldots, 16$. We use $n_{\text{train}} = 6$ training points and determine $\mathcal{D}_{\text{train}}$ using Latin hypercube sampling [41]. We randomly select two online points. Figure 4 depicts the tip displacement



(a) sampling fraction $n_Z/N = 0.05$ (b) sampling fraction $n_Z/N = 0.20$

Figure 3: Ideal case: Online responses for the full-order model (black, hidden), Galerkin ROM (blue) and Gappy POD ROM (red), and structure-preserving ROM (magenta) for different sampling fractions. Note that the Gappy POD ROM is unstable for a sampling fraction of 0.05.

for the training points. As the problem setup is the same as the previous section (except for the parameter variation), we employ the same timestep size of $h = 0.25$ seconds, leading to $M = 100$ time steps.

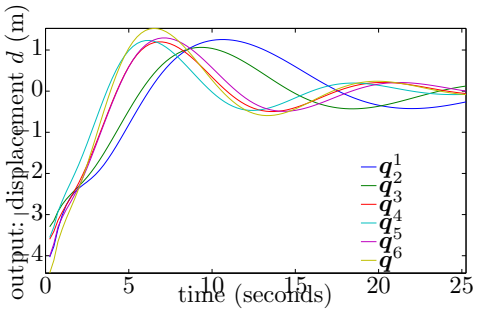
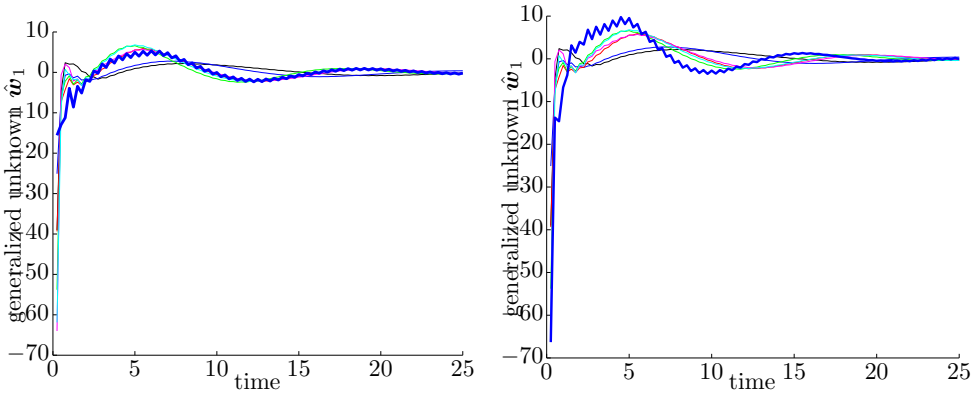


Figure 4: Unforced, varying inputs: Full-order model responses at training points in parameter space.

To gain insight into the proposed method's potential, Figure 5 depicts the time evolution of the first generalized unknown $\hat{\boldsymbol{w}}_1$—which is one of the forecasted variables—for the online and training points. Importantly, note that the qualitative response of this unknown is quite similar across parameter variation, which suggests that the forecasting method has the potential to generate accurate forecasts.

To construct the reduced-order models, we employ truncation critera of $\nu_{\boldsymbol{x}} = 1 - 10^{-5}$, which leads to a basis dimension of $\hat{N} = 8$, and $\nu_{\boldsymbol{r}} = 1 - 10^{-9}$ for Gappy POD, which results in a dimension of 316 for the residual basis. For the structure-preserving ROM, we sample 5% of the indices such that $n_Z = 150$; as this led to instabilities for Gappy POD, we sample 60% of the indices (i.e., $n_Z = 1800$) for that method.

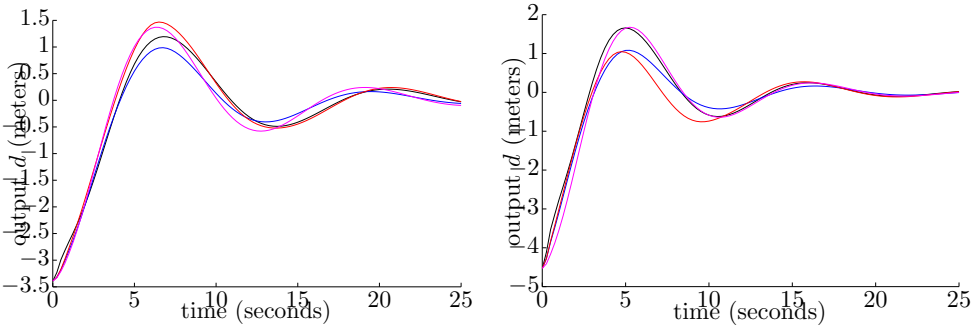
Figure 6 reports the responses of the full-order model and all three reduced-order models. The full-order-model simulation required 18.5 minutes and 307 total Newton iterations ($\bar{K}_{\text{FOM}} = 3.07$) for online point $\boldsymbol{q}^{\star,1}$ and 20.4 minutes and 347 Newton iterations ($\bar{K}_{\text{FOM}} = 3.47$) for online point $\boldsymbol{q}^{\star,2}$. Note that the reduced-order models are very accurate at the prediction points. At online point $\boldsymbol{q}^{\star,1}$, they generate relative errors ε of 3.33×10^{-2} (Galerkin), 2.56×10^{-2} (Gappy POD), and 4.66×10^{-2} (structure preserving). At online point $\boldsymbol{q}^{\star,2}$, the relative errors are 3.48×10^{-2} (Galerkin), 4.07×10^{-2}



(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 5: Unforced, varying inputs: First generalized unknown at online point (bold curve) and training points (thin curves).

(Gappy POD), and 2.45×10^{-2} (structure preserving).[8]



(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 6: Unforced, varying inputs: Online responses for the full-order model (black), Galerkin ROM (blue), Gappy POD ROM (red), and structure-preserving ROM (magenta).

Figure 7 reports the Newton-iteration and wall-time performance of the reduced-order models for different forecasting strategies at the two online points. First, note that the proposed forecasting method always yields better performance than polynomial extrapolation, regardless of the values for the forecasting parameters or polynomial degree. Second, observe that the performance of the proposed forecasting method is relatively insensitive to its parameters τ and $\alpha_{\max}$. Also, note that adding 'memory' to the polynomial extrapolation forecast—which yields a higher-degree extrapolant—is almost always deleterious to its performance. In addition, improvement in wall-time speedup provided by the forecasting technique is not as strong as the improvement in number of Newton iterations. This can be attributed to the presence of other operations (e.g., solution updating, residual computa-

[8]Different initial guesses for the Newton solver lead to (slightly) different computed responses. Thus, the ROM responses in principle depend on the forecasting method. However, the resulting differences in errors were negligible in these experiments; therefore, we only report the ROM error generated by an initial guess of zero (i.e., polynomial forecast, $\alpha = 0$ in Figure 6).

tion to check for convergence) that contribute to the simulation time (see the remark in Section 4.2). Finally, observe that the speedups generated by the structure-preserving method are far superior to those generated by Galerkin and Gappy POD. This is due to the fact that the structure-preserving method employed only $n_Z = 150$, whereas Galerkin is not equipped with a spatial-complexity-reduction mechanism and Gappy POD required $n_Z = 1800$ to generate a stable response.

4.4. Forced, varying inputs

In this section, we activate the external forcing, thereby allowing $q_i \in [-0.5, 0.5]$, $i = 1, \ldots, 16$. The timestep was again set to $h = 0.25$ seconds, leading to $M = 100$ time steps. This value was again determined by a timestep-verification study at the nominal configuration $\bar{\boldsymbol{q}}$ using a refinement factor of two. The approximated rate of convergence in the output quantity at the end of the time interval for this timestep size was determined to be 1.67 (close to the asymptotic value of 2.0), and the error in this quantity as approximated by Richardson extrapolation was 1.33%. As before, we used Latin hypercube sampling to determine the $n_{\text{train}} = 6$ training points; Figure 8(a) reports the full-order-model responses at these points. Note that parameter variation leads to significant changes in the response. We randomly select two online points at which we will perform prediction with the ROMs.

Figure 9 depicts the time evolution of the first generalized unknown $\hat{\boldsymbol{w}}_1$ for the online and training points. As before, there is qualitative similarity of this forecasted variable for the different points; this suggests the forecasting method can again realize computational savings. Also, note that the character of the response changes appreciably when the external force is activated at $t = 6.25$ seconds.

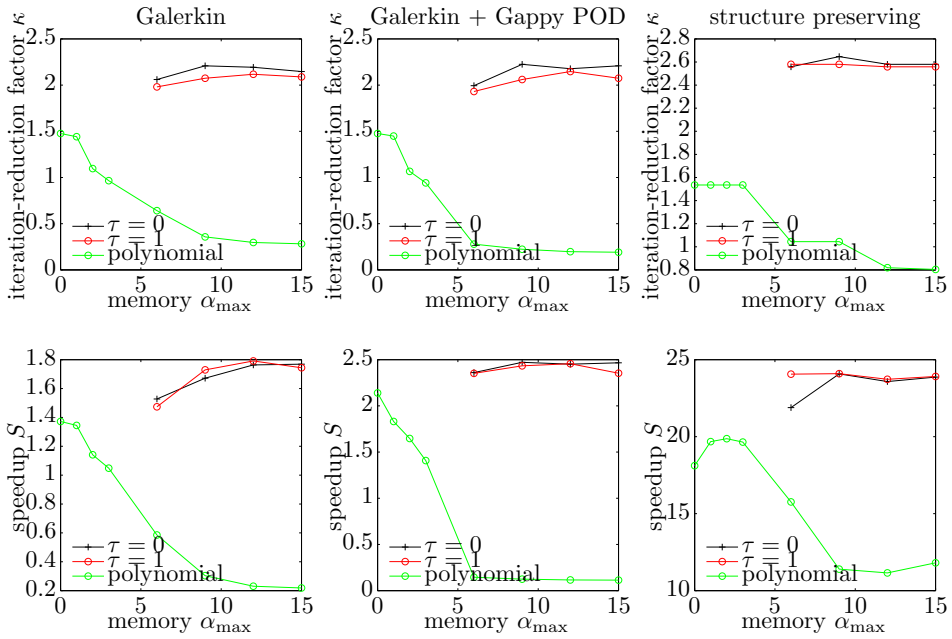
The reduced-order models employ trunction critera of $\nu_{\boldsymbol{x}} = 1 - 10^{-6}$ (basis dimension of $\hat{N} = 16$) and a residual-basis dimension of 1800. The structure-preserving method approximates the external force via Gappy POD (see Ref. [38]); for this purpose, it employs a truncation criterion of $\nu_{\boldsymbol{f}_{\text{ext}}} = 1$, leading to a basis dimension of 4.[9] Again, the Gappy POD ROM employs a sampling rate of 60% ($n_Z = 1800$) and the structure-preserving ROM employs a samping percentage of 5% ($n_Z = 150$).[10]

Figure 10 reports the responses of the full-order model and the reduced-order models at the online prediction points. The full-order model consumed 20.3 minutes and 330 Newton iterations ($\bar{K}_{\text{FOM}} = 3.3$) at online point $\boldsymbol{q}^{\star,1}$ and 22.6 minutes and 360 Newton iterations ($\bar{K}_{\text{FOM}} = 3.6$) at point $\boldsymbol{q}^{\star,2}$. The relative errors ε of the ROMs at online point $\boldsymbol{q}^{\star,1}$ are 1.56×10^{-2} (Galerkin), 1.56×10^{-1} (Gappy POD), and 5.78×10^{-2} (structure-preserving). For online point $\boldsymbol{q}^{\star,2}$, the errors are 2.41×10^{-2} (Galerkin), 1.68×10^{-1} (Gappy POD), and 3.05×10^{-2} (structure-preserving). Note that the Galerkin and structure-preserving ROMs are quite accurate, but the Gappy POD ROM incurs significant errors.

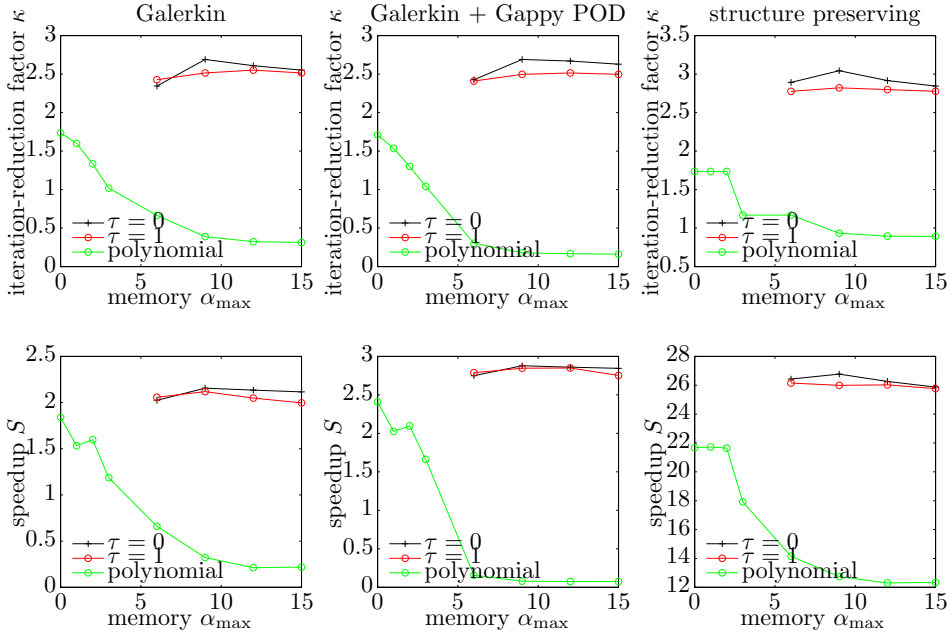
Figure 11 reports the Newton-iteration and wall-time performance of the ROMs for different forecasting strategies. The results are very similar to those for the unforced case: the proposed forecasting method nearly always exhibits performance superior to that of polynomial extrapolation, the proposed method is relatively insensitive to the parameters τ and $\alpha_{\max}$, and high-order polynomial extrapolation performs very poorly. In addition, improvement in iteration-reduction factor κ exceeds the improvement in speedup S, and the structure-preserving method generates the largest speedups due to the fact it employs the smallest number of sample indices. Additionally, notice the 'missing' data points for polynomial extrapolation with $\alpha_{\max} = 12$ and $\alpha_{\max} = 15$ for the Gappy POD ROM; these missing data indicate that the Gappy POD ROM did not converge for these forecasts. This implies that the initial guesses were so poor that the globalized Newton method failed to generate an acceptable solution within the alloted 500 Newton iterations at least one time step.

Also, note that employing $\tau = 0$ appears to systematically outperform $\tau = 1$ in terms of the iteration-reduction factor κ metric. However, this does not always lead to an improvement in speedup

[9] Note that the external force is composed of only four linearly independent components $\mathbf{r}_i$, $i = 1, \ldots, 4$ (see Eq. (35)).
[10] The Gappy POD ROM was unstable for $n_Z = 150$.

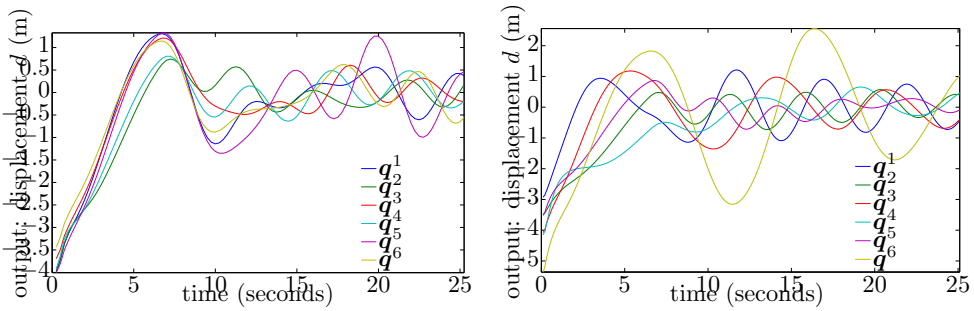


(a) online point $\boldsymbol{q}^{\star,1}$



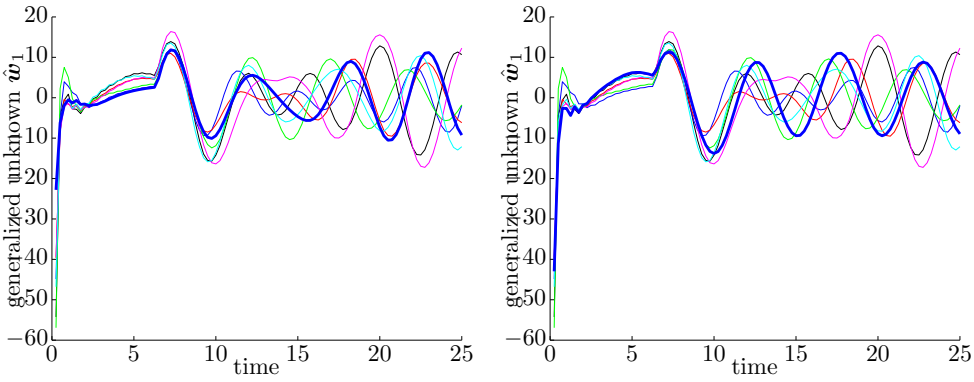
(b) online point $\boldsymbol{q}^{\star,2}$

Figure 7: Unforced, varying inputs: Performance of the forecasting method. The proposed forecasting method decreases both the number of requried Newton iterations and simulation time compared with polynomial extrapolation in nearly all cases.



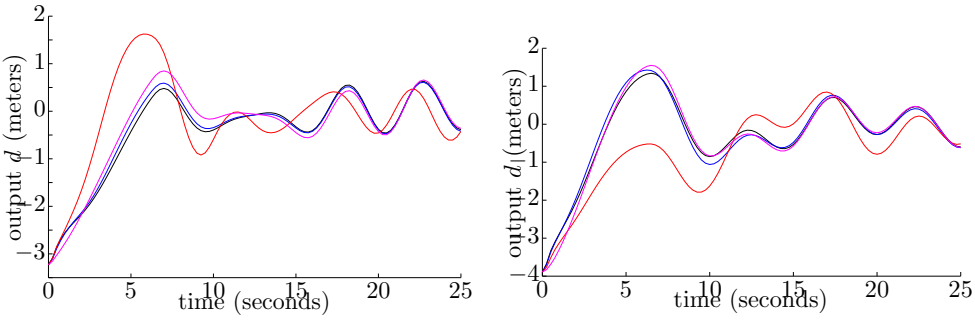
(a) Section 4.4 (b) Section 4.5: larger parameter variation

Figure 8: Forced, varying inputs: Full-order model responses at training points in parameter space. Note that larger parameter variation leads to larger parameter-induced changes in the output.



(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 9: Forced, varying inputs: First generalized unknown at online point (bold curve) and training points (thin curves).



(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 10: Forced, varying inputs: Online responses for the full-order model (black), Galerkin ROM (blue), Gappy POD ROM (red), and structure-preserving ROM (magenta).

(see the structure-preserving ROM for $\boldsymbol{q}^{\star,1}$). This can be attributed to the fact that employing $\tau = 0$ results in more frequent forecast recomputation (i.e., whenever the number of Newton iterations exceeds zero) than the $\tau = 1$ case.

These results highlight that the proposed forecasting method is applicable even for the more challenging problem of parameterized forced responses.

4.5. Forced, varying inputs, larger parameter variation.

In this section, we assess the performance of the method for the same problem as Section 4.4, but with an increased parameter variation, i.e., $\mathcal{D} = [-1, 1]^{16}$. This poses a greater challenge for both the reduced-order models and the forecasting method, as they now rely on training data from the same number of points (we keep $n_{\text{train}} = 6$) in a larger parameter domain. As the model now undergoes larger parameter variation, we decrease the timestep size to $h = 0.1$ seconds, leading to $M = 250$ total time steps.[11] Note that this timestep remains in the asymptotic range of convergence for the nominal configuration $\bar{\boldsymbol{q}}$, as it is smaller than the previously verified value of 0.25 seconds. Again, training points are chosen by Latin hypercube sampling, and the online points are selected randomly. Figure 8(b) reports the full-order-model responses at the training points; note that the changes in the response are in fact more significant than for the previous case with smaller parameter variation.

Figure 12 again reports the time evolution of the first generalized unknown. Note that again there is similar qualitative structure across parameter variation.

The attributes for the reduced-order models are the same as in Section 4.4, with one exception: a criterion of $\nu_{\boldsymbol{x}} = 1 - 10^{-4}$ is employed for the state, which associates with a basis dimension of $\hat{N} = 10$. Note that basis dimension is larger than in the previous case.

Figure 13 depicts the full-order-model response along with those for the reduced-order models. The full-order model took 36.7 minutes and 605 Newton iterations ($\bar{K}_{\text{FOM}} = 2.42$) at online point $\boldsymbol{q}^{\star,1}$ and 37.3 minutes and 595 Newton iterations ($\bar{K}_{\text{FOM}} = 2.38$) at point $\boldsymbol{q}^{\star,2}$. As before, the Galerkin and structure-preserving ROMs are more accurate than the Gappy POD ROMs. The relative errors ε at point $\boldsymbol{q}^{\star,1}$ are 4.19×10^{-2} (Galerkin), 1.29×10^{-1} (Gappy POD), and 3.65×10^{-2} (structure preserving). At online point $\boldsymbol{q}^{\star,2}$, the associated errors are 6.91×10^{-2} (Galerkin), 1.73×10^{-1} (Gappy POD), and 5.67×10^{-2} (structure preserving).

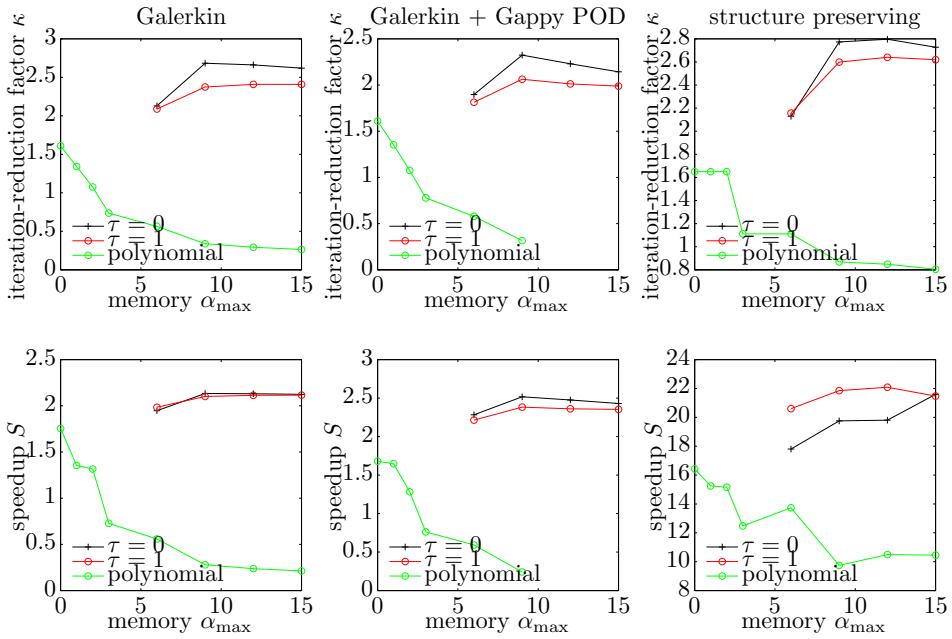
Figure 14 reports the Newton-iteration and wall-time results for the different forecasting strategies. Note that the results are extremely similar to those in Section 4.4. The primary exception can be seen by comparing Figure 14 with 11: the iteration-reduction factor κ and speedup S performance of the reduced-order models has decreased. This can be attributed to the challenge of larger parameter variation, as the ROMs are now responsible for capturing a wider range of physics.

From this set of experiments, we conclude that the proposed technique can improve ROM performance even for problems with relatively large parameter variation.

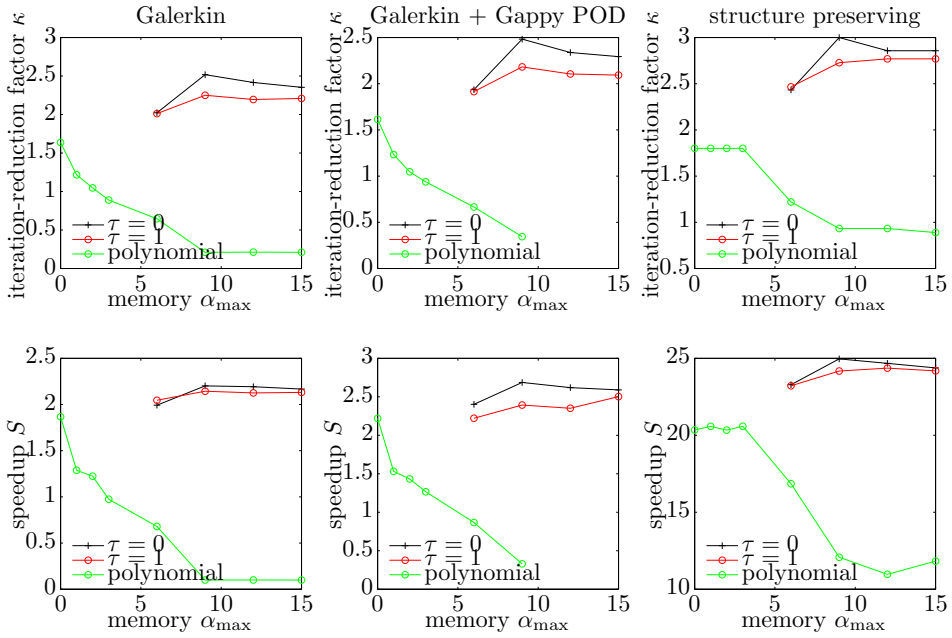
4.6. Average performance

Finally, we summarize the performance of the forecasting techniques over the complete set of experiments. Figure 15 reports average, minimum, and maximum values of the reduction-factor improvement $\mathfrak{k}$, and speedup improvement $\mathfrak{s}$ over all experiments (i.e., all three experiments in Sections 4.3–4.5, all three reduced-order models, and both online points $\boldsymbol{q}^{\star,1}$ and $\boldsymbol{q}^{\star,2}$). Here, $\mathfrak{k} = \kappa/\kappa_{\text{no}}$ and $\mathfrak{s} = S/S_{\text{no}}$ can each be computed for a given ROM simulation; a subscript 'no' indicates the value of the variable for a zero initial guess (i.e., polynomial extrapolation with $\alpha = 0$). First, note that the proposed method always outperforms polynomial forecasting in the mean, maximum, and minimum achieved performance for both reduction-factor improvement $\mathfrak{k}$ and speedup improvement $\mathfrak{s}$. Secondly,

[11] The full-order model did not converge for several of the training points when $h = 0.25$ seconds was employed.

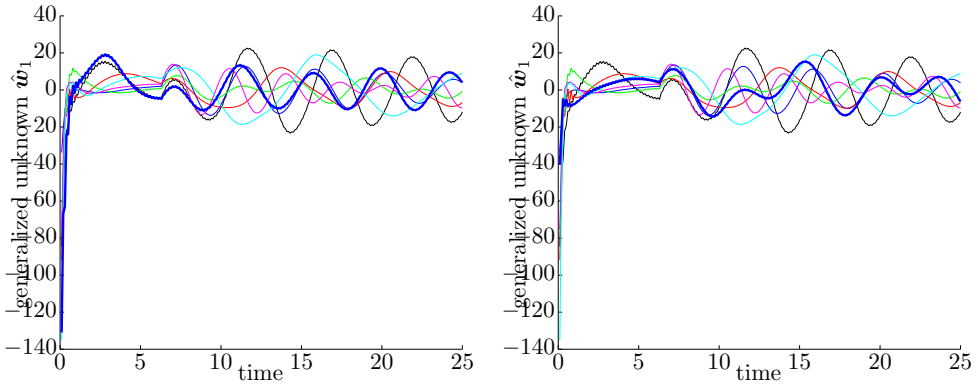


(a) online point $\boldsymbol{q}^{\star,1}$



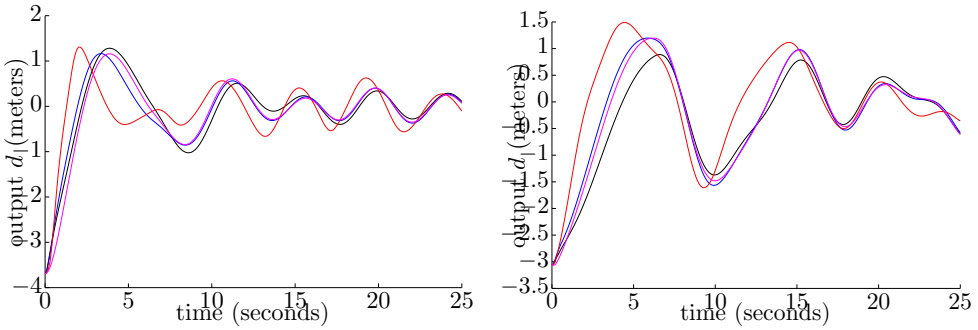
(b) online point $\boldsymbol{q}^{\star,2}$

Figure 11: Forced, varying inputs: Performance of the forecasting method. For all reduced-order models, the proposed forecasting method decreases both the number of requried Newton iterations and simulation time compared with polynomial extrapolation.



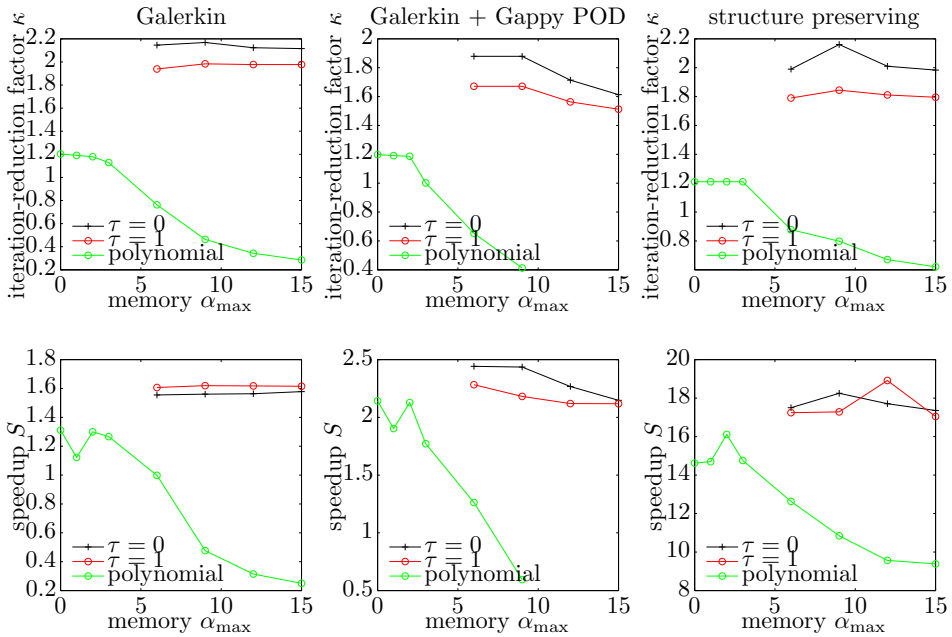
(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 12: Forced, varying inputs, larger parameter variation: Parameter dependence of the first generalized coordinate.

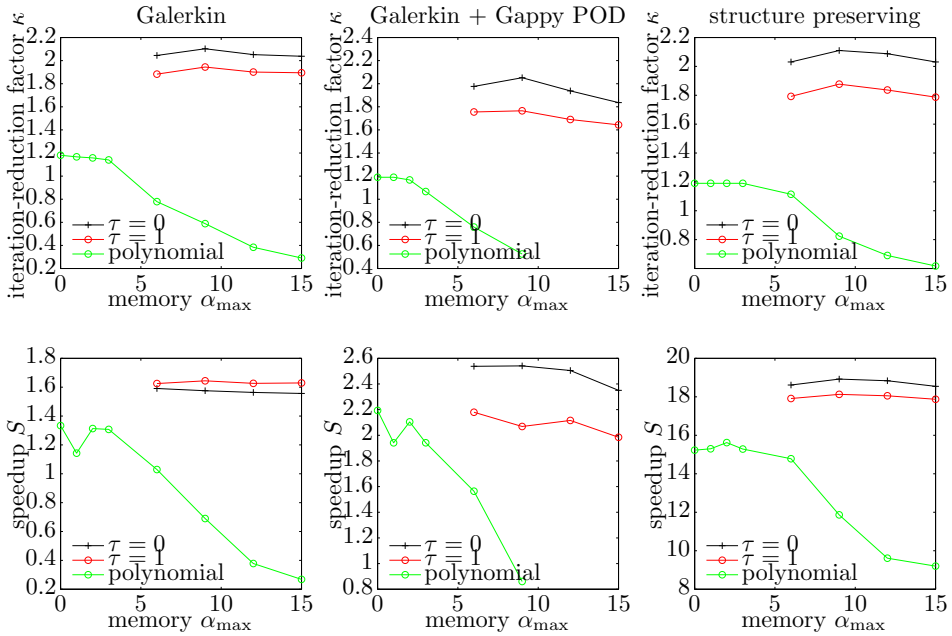


(a) online point $\boldsymbol{q}^{\star,1}$ (b) online point $\boldsymbol{q}^{\star,2}$

Figure 13: Forced, varying inputs, larger parameter variation: Online responses for the full-order model (black), Galerkin ROM (blue), Gappy POD ROM (red), and structure-preserving ROM (magenta)..



(a) online point $\boldsymbol{q}^{\star,1}$

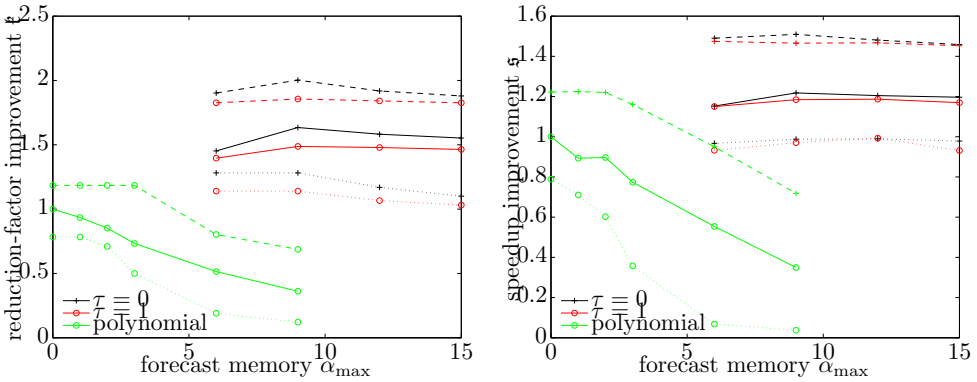


(b) online point $\boldsymbol{q}^{\star,2}$

Figure 14: Forced, varying inputs, larger parameter variation: Performance of the forecasting method. For all reduced-order models, the proposed forecasting method decreases both the number of required Newton iterations and simulation time compared with polynomial extrapolation.

the maximum, minimum, and average performance of polynomial forecasting were all made worse by increasing the polynomial degree.

Finally, the best average performance was achieved for a forecast memory of $\alpha_{\max} = 9$ and Newton-iteration criterion of $\tau = 0$. In this case, the iteration-reduction factor was increased by 63% on average; the speedup was improved by 22% on average. Critically, note that these temporal-complexity gains incur *no additional error*, and so they strictly serve to improve the performance of the ROMs with no penalty.



(a) Reduction-factor improvement

(b) Speedup improvement

Figure 15: Performance of forecasting methods quantified over all reduced-order models, problems, and online prediction points. The mean (solid line), maximum (dashed line), and minimum (dotted) are reported.

5. Conclusions

This paper has described a method for decreasing the temporal complexity of nonlinear reduced-order models in the case of implicit time integration. The method exploits knowledge of the dynamical system's temporal behavior in the form of 'time-evolution bases'; one such basis is generated for each generalized coordinate of the time integrator's unknown during the (offline) training stage. During the (online) deployed stage, these time-evolution bases are used—along with the solution at recent time steps—to forecast the unknown at future time steps via Gappy POD. If this forecast is accurate, the Newton-like solver will converge in very few iterations, leading to computational-cost savings.

Numerical experiments demonstrated the potential of the method to significantly improve the performance of nonlinear reduced-order models, even in the presence of high-frequency content in the dynamics. The experiments also demonstrated the effect of input parameters on the method's performance, and provided a parameter study to analyze the effect of the method's parameters.

Future work includes devising a way to directly handle frequency and phase shifts in the response, as well as time-shifted temporal behavior.

Acknowledgments

The authors acknowledge Julien Cortial for providing the original nonlinear-truss code that was modified to generate the numerical results, as well as the anonymous reviewers for their insightful suggestions.

This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly

owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000.

Appendix A. Implicit time-integration schemes: first-order ODEs

For notational simplicity, consider a system without parametric inputs $\boldsymbol{q}$, and define $\bar{\boldsymbol{f}}(\boldsymbol{x},t) \equiv \boldsymbol{f}\left(\boldsymbol{x};t,\boldsymbol{p}\left(t\right)\right)$ such that

$$\dot{\boldsymbol{x}} = \bar{\boldsymbol{f}}\left(\boldsymbol{x},t\right). \tag{A.1}$$

Further, denote by h the time-step size at time step n.

Appendix A.1. Implicit linear multi-step schemes

A linear k-step method applied to first-order ODEs can be expressed as

$$\sum_{j=0}^{k} \alpha_j \boldsymbol{x}^{n-j} = h \sum_{j=0}^{k} \beta_j \bar{\boldsymbol{f}}\left(\boldsymbol{x}^{n-j}, t^{n-j}\right), \tag{A.2}$$

where $\alpha_0 \neq 0$ and $\sum_{j=0}^{k} \alpha_j = 0$ is necessary for consistency. These methods are implicit if $\beta_0 \neq 0$. In this case, the form of the residual is

$$\boldsymbol{r}^n\left(\boldsymbol{w}^n\right) = \alpha_0 \boldsymbol{w}^n - h\beta_0 \bar{\boldsymbol{f}}(\boldsymbol{w}^n, t^n) + \sum_{j=1}^{k} \alpha_j \boldsymbol{x}^{n-j} - h \sum_{j=1}^{k} \beta_j \bar{\boldsymbol{f}}\left(\boldsymbol{x}^{n-j}, t^{n-j}\right) \tag{A.3}$$

and the explicit state update is simply

$$\boldsymbol{x}^n = \boldsymbol{w}^n. \tag{A.4}$$

Therefore, the unknown is the state at time t^n.

Appendix A.2. Implicit Runge–Kutta schemes

For an s-stage Runge–Kutta scheme, the form of the residual is

$$\boldsymbol{r}_i^n\left(\boldsymbol{w}^{n,1}, \ldots, \boldsymbol{w}^{n,s}\right) = \boldsymbol{w}^{n,i} - \bar{\boldsymbol{f}}(\boldsymbol{x}^{n-1} + h \sum_{j=1}^{s} a_{ij} \boldsymbol{w}^{n,i}, t^{n-1} + c_i h), \quad i = 1, \ldots, s \tag{A.5}$$

with the following explicit computation of the state:

$$\boldsymbol{x}^n = \boldsymbol{x}^{n-1} + h \sum_{i=1}^{s} b_i \boldsymbol{w}^{n,i}. \tag{A.6}$$

The unknowns correspond to the velocity $\dot{\boldsymbol{x}}$ at times $t^{n-1} + c_i h$, $i = 1, \ldots, s$.

Appendix B. Implicit time-integration schemes: second-order ODEs

For notational simplicity, consider a second-order differential equations without parametric inputs $\boldsymbol{q}$ and define $\bar{\boldsymbol{g}}\left(\boldsymbol{x}, \dot{\boldsymbol{x}}, t\right) \equiv \boldsymbol{g}\left(\boldsymbol{x}, \dot{\boldsymbol{x}}; t, p(t)\right)$ such that

$$\ddot{\boldsymbol{x}} = \bar{\boldsymbol{g}}\left(\boldsymbol{x}, \dot{\boldsymbol{x}}, t\right). \tag{B.1}$$

Appendix B.1. Implicit Nyström method

Nyström methods are partitioned Runge–Kutta schemes applied to second-order ODEs. They lead to the following representation for the residuals:

$$\boldsymbol{r}_i^n\left(\boldsymbol{w}^{n,1},\dots,\boldsymbol{w}^{n,s}\right) = \boldsymbol{w}^{n,i} - \bar{\boldsymbol{g}}\left(\boldsymbol{x}^{n-1} + c_i h\dot{\boldsymbol{x}}^{n-1} + h^2\sum_{j=1}^{s}\bar{a}_{ij}\boldsymbol{w}^{n,i}, \dot{\boldsymbol{x}}^{n-1} + h\sum_{j=1}^{s}\hat{a}_{ij}\boldsymbol{w}^{n,i}, t^{n-1} + c_i h\right), \tag{B.2}$$

$i = 1,\dots,s$. The state and velocity are updated explicitly as

$$\boldsymbol{x}^n = \boldsymbol{x}^{n-1} + h\dot{\boldsymbol{x}}^{n-1} + h^2\sum_{i=1}^{s}\bar{b}_i\boldsymbol{w}^{n,i} \tag{B.3}$$

$$\dot{\boldsymbol{x}}^n = \dot{\boldsymbol{x}}^{n-1} + h\sum_{i=1}^{s}\hat{b}_i\boldsymbol{w}^{n,i}. \tag{B.4}$$

The unknowns correspond to the acceleration $\ddot{\boldsymbol{x}}$ at times $t^{n-1} + c_i h$, $i = 1,\dots,s$.

Appendix B.2. Implicit Newmark method

The implicit Newmark method leads to the following residuals:

$$\boldsymbol{r}^n(\boldsymbol{w}^n) = \boldsymbol{w}^n - \bar{\boldsymbol{g}}\left(\boldsymbol{x}^{n-1} + h\dot{\boldsymbol{x}}^{n-1} + \frac{h^2}{2}\left[(1-2\beta)\,\ddot{\boldsymbol{x}}^{n-1} + 2\beta\boldsymbol{w}^n\right], \dot{\boldsymbol{x}}^{n-1} + h\left[(1-\gamma)\,\ddot{\boldsymbol{x}}^{n-1} + \gamma\boldsymbol{w}^n\right], t^n\right) \tag{B.5}$$

The state and velocity are explicitly updated as

$$\boldsymbol{x}^n = \boldsymbol{x}^{n-1} + h\dot{\boldsymbol{x}}^{n-1} + \frac{h^2}{2}\left[(1-2\beta)\,\ddot{\boldsymbol{x}}^{n-1} + 2\beta\boldsymbol{w}^n\right] \tag{B.6}$$

$$\dot{\boldsymbol{x}}^n = \dot{\boldsymbol{x}}^{n-1} + h\left[(1-\gamma)\,\ddot{\boldsymbol{x}}^{n-1} + \gamma\boldsymbol{w}^n\right]. \tag{B.7}$$

Here, the unknown corresponds to the acceleration $\ddot{\boldsymbol{x}}$ at time t^n.

Appendix C. Proper orthogonal decomposition

Algorithm 2 describes the method for computing a proper-orthogonal-decomposition (POD) basis given a set of snapshots. The method essentially amounts to computing the singular value decomposition of the snapshot matrix. The left singular vectors define the POD basis.

References

[1] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathématique Académie des Sciences 339 (2004) 667–672.

[2] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (2010) 2737–2764.

[3] R. Everson, L. Sirovich, Karhunen–Loève procedure for gappy data, Journal of the Optical Society of America A 12 (1995) 1657–1664.

Algorithm 2 Proper-orthogonal-decomposition basis computation (normalized snapshots)

Input: Set of snapshots $\mathcal{X} \equiv \{\boldsymbol{w}_i\}_{i=1}^{n_w} \subset \mathbb{R}^N$, energy criterion $\nu \in [0,1]$
Output: $\boldsymbol{\Phi}^e(\mathcal{X}, \nu)$

1: Compute thin singular value decomposition $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{W} \equiv [\boldsymbol{w}_1/\|\boldsymbol{w}_1\| \ \cdots \ \boldsymbol{w}_{n_w}/\|\boldsymbol{w}_{n_w}\|]$.
2: Choose dimension of truncated basis $\hat{N} = n_e(\nu)$, where

$$n_e(\nu) \equiv \arg\min_{i \in \mathcal{V}(\nu)} i \tag{C.1}$$

$$\mathcal{V}(\nu) \equiv \{n \in \{1, \ldots, n_w\} \mid \sum_{i=1}^{n} \sigma_i^2 / \sum_{i=1}^{n_w} \geq \nu\}, \tag{C.2}$$

and $\boldsymbol{\Sigma} \equiv \mathrm{diag}(\sigma_i)$.
3: $\boldsymbol{\Phi}^e(\mathcal{X}, \nu) = \left[\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_{\hat{N}}\right]$, where $\boldsymbol{U} \equiv [\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_{n_w}]$.

[4] K. Carlberg, C. Farhat, J. Cortial, 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) 623–647.

[5] P. Diggle, Time Series: A Biostatistical Introduction, Clarendon Press, 1990.

[6] C. Gourieroux, ARCH models and financial applications, Springer-Verlag, 1997.

[7] R. F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of U. K. inflation, Econometrica 50 (1982) 987–1008.

[8] F. A. Graybill, An introduction to linear statistical models, McGraw-Hill, New York, 1961.

[9] M. Hollander, D. A. Wolfe, Nonparametric statistical methods, Wiley, New York, 1973.

[10] D. B. Percival, A. T. Walden, Spectral Analysis for Physical Applications, Cambridge University Press, 1993.

[11] C. C. Holt, Forecasting seasonals and trends by exponentially weighted moving averages, International Journal of Forecasting 20 (2004) 5–10.

[12] P. R. Winters, Forecasting sales by exponentially weighted moving averages, Management Science 6 (1960) 324–342.

[13] P. Brown, G. Byrne, A. Hindmarsh, VODE: A variable-coefficient ODE solver, SIAM Journal on Scientific and Statistical Computing 10 (1989) 1038–1051.

[14] J. Nievergelt, Parallel methods for integrating ordinary differential equations, Communications of the ACM 7 (1964) 731–733.

[15] M. J. Gander, A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations, Numerical Linear Algebra with Applications 6 (1999) 125–145.

[16] G. Horton, S. Vandewalle, A space-time multigrid methods for parabolic partial differential equaions, SIAM J. Sci. Comput. 16 (1995) 848–864.

[17] J. Lions, Y. Maday, G. Turinici, A "parareal" in time discretization of PDEs, Comptes Rendus de l'Academie des Sciences Series I Mathematics 332 (2001) 661–668.

[18] J. Cortial, Time-parallel methods for accelerating the solution of structural dynamics problems, Ph.D. thesis, Stanford University, 2011.

[19] C. Farhat, M. Chandesris, Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications, International Journal for Numerical Methods in Engineering 58 (2003) 1397–1434.

[20] C. Harden, Real time computing with the parareal algorithm, Ph.D. thesis, Florida State University, 2008.

[21] R. Bos, X. Bombois, P. Van den Hof, Accelerating large-scale non-linear models for monitoring and control using spatial and temporal correlations, Proceedings of the American Control Conference 4 (2004) 3705–3710.

[22] D. Ryckelynck, A priori hyperreduction method: an adaptive approach, Journal of Computational Physics 202 (2005) 346–366.

[23] T. Kim, D. James, Skipping steps in deformable simulation with online model reduction, ACM Transactions on Graphics 28 (2009) 1–9.

[24] E. Hairer, G. Wanner, Solving ordinary differential equations II: stiff and differential-algebraic problems, Springer Verlag, 2002.

[25] P. Krysl, S. Lall, J. E. Marsden, Dimensional model reduction in non-linear finite elements dynamics of solids and structures, Int. J. Numer. Meth. Engng 51 (2001) 479–504.

[26] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations, International Journal for Numerical Methods in Engineering 86 (2011) 155–181.

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

[28] T. Bui-Thanh, K. Willcox, O. Ghattas, Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications, AIAA Journal 46 (2008) 2520–2529.

[29] P. A. LeGresley, Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods, Ph.D. thesis, Stanford University, 2006.

[30] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Transactions on Automatic Control 53 (2008) 2237–2251.

[31] D. Galbally, K. Fidkowski, K. Willcox, O. Ghattas, Non-linear model reduction for uncertainty quantification in large-scale inverse problems, International Journal for Numerical Methods in Engineering 81 (2009) 1581–1608.

[32] M. Drohmann, B. Haasdonk, M. Ohlberger, Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation, SIAM Journal on Scientific Computing 34 (2012) A937–A969.

[33] T. Bui-Thanh, D. Murali, K. Willcox, Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics, AIAA Paper 2003-4213, 21st Applied Aerodynamics Conference, Orlando, FL (2003).

[34] T. Bui-Thanh, M. Damodaran, K. Willcox, Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition, AIAA Journal 42 (2004) 1505–1516.

[35] D. Venturi, G. E. Karniadakis, Gappy data and reconstruction procedures for flow past a cylinder, Journal of Fluid Mechanics 519 (2004) 315–336.

[36] K. Willcox, Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition, Computers and Fluids 35 (2006) 208–226.

[37] T. D. Robinson, M. S. Eldred, K. Willcox, R. Haimes, Strategies for multifidelity optimization with variable dimensional hierarchical models, AIAA Paper 2006-1819, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Newport, RI (2006).

[38] K. Carlberg, R. Tuminaro, P. Boggs, Preserving Lagrangian structure in nonlinear model reduction with application to structural dynamics, arXiv preprint 1401.8044 (2014).

[39] I. Chowdhury, S. Dasgupta, Computation of Rayleigh damping coefficients for large systems, The Electronic Journal of Geotechnical Engineering 8 (2003).

[40] D. Dunlavy, T. Kolda, E. Acar, Poblano v1. 0: A Matlab toolbox for gradient-based optimization, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Tech. Rep. SAND 1422 (2010).

[41] M. D. McKay, R. J. Beckman, W. J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (1979) 239–245.

Chapter 3

Adaptive *h*-refinement for reduced-order models

This chapter presents a method for adaptively refining projection-based reduced-order models *a posteriori* using a goal-oriented framework inspired by mesh-adaptive *h*-refinement. The technique allows reduced-order models to generate arbitrarily accurate answers (at the cost of an adaptively-increased basis dimension), which allows it to capture physical phenomena not present in the training data. Thus, the proposed technique acts as a 'failsafe' mechanism. This work has been accepted for publication in the International Journal for Numerical Methods in Engineering. It is in press at the time of this writing.

Adaptive h-refinement for reduced-order models

Kevin Carlberg*

Sandia National Laboratories

Abstract

This work presents a method to adaptively refine reduced-order models *a posteriori* without requiring additional full-order-model solves. The technique is analogous to mesh-adaptive h-refinement: it enriches the reduced-basis space online by 'splitting' a given basis vector into several vectors with disjoint support. The splitting scheme is defined by a tree structure constructed offline via recursive k-means clustering of the state variables using snapshot data. The method identifies the vectors to split online using a dual-weighted-residual approach that aims to reduce error in an output quantity of interest. The resulting method generates a hierarchy of subspaces online without requiring large-scale operations or full-order-model solves. Further, it enables the reduced-order model to satisfy *any prescribed error tolerance* regardless of its original fidelity, as a completely refined reduced-order model is mathematically equivalent to the original full-order model. Experiments on a parameterized inviscid Burgers equation highlight the ability of the method to capture phenomena (e.g., moving shocks) not contained in the span of the original reduced basis.

Keywords: adaptive refinement, h-refinement, model reduction, dual-weighted residual, adjoint error estimation, clustering

1. Introduction

Modeling and simulation of parameterized systems has become an essential tool across a wide range of industries. However, the computational cost of executing high-fidelity large-scale simulations is infeasibly high for many time-critical applications. In particular, many-query scenarios (e.g., sampling for solving statistical inverse problems) can require thousands of simulations corresponding to different input-parameter instances of the system; real-time contexts (e.g., model predictive control) require simulations to execute in mere seconds.

Reduced-order models (ROMs) have been developed to mitigate this computational bottleneck. First, they execute an 'offline' stage during which computationally expensive training tasks (e.g., evaluating the high-fidelity model at several points in the input-parameter space) compute a representative low-dimensional reduced basis for the system state. Then, during the inexpensive 'online' stage, these methods quickly compute approximate solutions for arbitrary points in the input space via a projection process of the high-fidelity full-order-model (FOM) equations onto the low-dimensional subspace spanned by the reduced basis. They also introduce other approximations in the presence of general (i.e., not low-order polynomial) nonlinearities. See Ref. [1] and references within for a survey of current methods.

While reduced-order models almost always generate *fast* online predictions, there is no guarantee that they will generate sufficiently *accurate* online predictions. In fact, the accuracy of online predictions is predicated on the relevance of the training data to the online problem: if a physical phenomenon was not observed during the offline stage, then this feature will be missing from online predictions. In general, the most one can guarantee *a priori* is that the ROM solution error is bounded by a prescribed scalar over a

*7011 East Ave, MS 9159, Livermore, CA 94550. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

Email address: ktcarlb@sandia.gov (Kevin Carlberg)

URL: sandia.gov/~ktcarlb (Kevin Carlberg)

finite set of 'training points' in the input-parameter space [2]. While reduced-order models can be accurate at online points contained within a reasonable neighborhood of these training points (see, e.g., Ref. [3]), they are generally inaccurate for points far outside this set.

This lack of error control[1] precludes ROMs from being employed in many contexts. For example, PDE-constrained optimization requires the solution to satisfy a prescribed forcing sequence to guarantee convergence [4]. In uncertainty quantification, if the epistemic uncertainty due to the ROM solution error dominates other sources of uncertainty, the ROM cannot be exploited in a useful manner. When simulating parameterized highly nonlinear dynamical systems, it is unlikely that any amount of training will fully encapsulate the range of complex phenomena that can be encountered online; such problems require an efficient refinement mechanism to generate accurate ROM predictions.

A few methods exist to improve a ROM solution when it is detected to be inaccurate; however, they entail *large-scale* operation counts. The most common approach is to revert to the high-fidelity model, solve the associated high-dimensional equations for the current time step or optimization iteration, add the solution to the reduced basis, and proceed with the enriched reduced-order model [5, 6, 7]. Another approach adaptively improves the reduced-order model *a posteriori* by generating a Krylov subspace [8]; here, the reduced-order model serves to accelerate the full-order solve to any specified tolerance. As our goal is to improve the reduced-order model efficiently, i.e., without incurring large-scale operations, none of these methods is appropriate.

Instead, this work proposes a novel approach inspired by mesh-adaptive h-refinement. The main idea is to adaptively refine an inaccurate ROM *online* by 'splitting' selected reduced basis vectors into multiple vectors with disjoint discrete support. This splitting technique is defined by a tree structure generated offline by applying k-means clustering to the state variables. The method uses a dual-weighted residual approach to select vectors to split online. The resulting method generates a hierarchy of subspaces online without requiring any large-scale operations or high-fidelity solves. Most importantly, the methodology acts as a 'failsafe' mechanism for the ROM: h-adaptivity enables the ROM to satisfy *any prescribed error tolerance* online, as a fully refined ROM is mathematically equivalent to the original full-order model under modest conditions.

As a final note, some 'adaptive' methods exist to tailor the ROM to specific regions of the input space [9, 10, 11, 12, 13, 14], time domain [13, 15], and state space [16, 14]. However, these methods are primarily *a priori* adaptive: they construct separate ROMs for each region *offline* with the goal of reducing the ROM dimension. While they can be used to improve the ROM *a posteriori*, e.g., by restarting the greedy algorithm online, doing so incurs additional full-order-model solves, which is what we aim to avoid.

In the remainder of this paper, matrices are denoted by capitalized bold letters, vectors by lowercase bold letters, scalars by lowercase letters, and sets by capitalized letters. The columns of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times k}$ are denoted by $\boldsymbol{a}_i \in \mathbb{R}^m$, $i \in \mathbb{N}(k)$ with $\mathbb{N}(a) := \{1, \dots, a\}$ such that $\boldsymbol{A} := [\boldsymbol{a}_1 \ \cdots \ \boldsymbol{a}_k]$. The scalar-valued matrix elements are denoted by $a_{ij} \in \mathbb{R}$ such that $\boldsymbol{a}_j := [a_{1j} \ \cdots \ a_{mj}]^T$, $j \in \mathbb{N}(k)$.

2. Problem formulation

2.1. Full-order model

Consider solving a parameterized sequence of systems of equations

$$\tilde{\boldsymbol{r}}^k(\boldsymbol{x}^k; \boldsymbol{\mu}) = 0 \tag{1}$$

for $k \in \mathbb{N}(t)$, where $\boldsymbol{x}^k \in \mathbb{R}^n$ denotes the state at iteration k, $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^{n_\mu}$ denotes the input parameters (e.g., boundary conditions), $\tilde{\boldsymbol{r}}^k : \mathbb{R}^n \times \mathbb{R}^{n_\mu} \to \mathbb{R}^n$ denotes the residual operator at iteration k, and t denotes maximum number of iterations. This formulation is quite general, as it describes, e.g., parameterized systems of linear equations ($t = 1$, $\tilde{\boldsymbol{r}} : (\boldsymbol{x}; \boldsymbol{\mu}) \mapsto \boldsymbol{b}(\boldsymbol{\mu}) - \boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{x}$) such as those arising from the finite-element discretization of elliptic PDEs, and parameterized ODEs $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\mu})$ after time discretization by an implicit linear multistep method (e.g., $\tilde{\boldsymbol{r}}^k : (\boldsymbol{x}^k; \boldsymbol{\mu}) \mapsto \boldsymbol{x}^k - \boldsymbol{x}^{k-1} - \Delta t \boldsymbol{f}\left(\boldsymbol{x}^k; \boldsymbol{\mu}\right)$ for the backward Euler scheme) such

[1] Note that reduced-order-model error bounds—which exist for many problems—serve to quantify the error, while error control implies reducing this error *a posteriori*.

as those arising from the space- and time-discretization of parabolic and hyperbolic PDEs. Assume that we are primarily interested in computing outputs

$$z^k = g(\boldsymbol{x}^k; \boldsymbol{\mu}) \tag{2}$$

with $z^k \in \mathbb{R}$ and $g : \mathbb{R}^n \times \mathbb{R}^{n_{\boldsymbol{\mu}}} \to \mathbb{R}$.

When the dimension n is 'large', computing the outputs of interest z^k by first solving Eq. (1) and subsequently computing outputs via Eq. (2) can be prohibitively expensive. This is particularly true for many-query (e.g., statistical inversion) and real-time (e.g., model-predictive control) problems that demand a fast evaluation of the input–output map $\boldsymbol{\mu} \mapsto \{z^1, \dots, z^t\}$.

2.2. Reduced-order model

Model-reduction techniques aim to reduce the burden of solving Eq. (1) by employing a projection process. First, they execute a computationally expensive offline stage (e.g., solving Eq. (1) for a training set $\boldsymbol{\mu} \in \mathcal{D}_{\text{train}} \subset \mathcal{D}$) to construct 1) a low-dimensional trial basis (in matrix form) $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ with $p \ll n$ that (hopefully) captures the behavior of the state $\boldsymbol{x}$ throughout the parameter domain $\mathcal{D}$, and 2) an associated test basis $\boldsymbol{W} \in \mathbb{R}^{n \times p}$. Then, during the computationally inexpensive online stage, these methods approximately solve Eq. (2) for arbitrary $\boldsymbol{\mu} \in \mathcal{D}$ by searching for solutions in the trial subspace $\bar{\boldsymbol{x}} + \text{range}\,(\boldsymbol{V}) \subset \mathbb{R}^n$ (with $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ a chosen reference configuration) and enforcing the residual $\tilde{\boldsymbol{r}}^k$ to be orthogonal to the test subspace $\text{range}\,(\boldsymbol{W}) \subset \mathbb{R}^n$:

$$\boldsymbol{W}^T \tilde{\boldsymbol{r}}^k(\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k; \boldsymbol{\mu}) = 0. \tag{3}$$

Here, $\hat{\boldsymbol{x}}^k \in \mathbb{R}^p$ denotes the generalized coordinates of the reduced-order-model solution $\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k$ at iteration k. When the residual operator exhibits general nonlinear dependence on the state or is non-affine in the inputs, additional complexity-reduction approximations such as empirical interpolation [17], collocation [18, 19, 7], discrete empirical interpolation [20, 21], or gappy proper orthogonal decomposition (POD) [19, 22] are required to ensure that computing the low-dimensional residual $\boldsymbol{W}^T \tilde{\boldsymbol{r}}^k$ incurs an n-independent operation count. For simplicity, we do not consider such approximations in the present work; future work will entail extending the proposed method to such 'hyper-reduced' order models.

In many cases, the test basis can be expressed as $\boldsymbol{W} = \boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu})\,\boldsymbol{V}$. For example, $\boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu}) = I$ for Galerkin projection; balanced truncation uses $\boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu}) = \boldsymbol{Q}$, where $\boldsymbol{Q}$ is the observability Gramian of the linear time-invariant system; the least-squares Petrov–Galerkin projection [18, 22] underlying the GNAT method employs $\boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu}) = \partial \tilde{\boldsymbol{r}}^k / \partial \boldsymbol{x}\,(\boldsymbol{x}, \boldsymbol{\mu})$; for linearized compressible-flow problems, $\boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu})$ can be chosen to guarantee stability [23]. When this holds, the Petrov–Galerkin projection (3) is equivalent to a Galerkin projection performed on the modified residual $\boldsymbol{r}^k := \boldsymbol{A^n}\,(\boldsymbol{x}; \boldsymbol{\mu})^T\,\tilde{\boldsymbol{r}}^k$:

$$\boldsymbol{V}^T \boldsymbol{r}^k(\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k; \boldsymbol{\mu}) = 0, \tag{4}$$

for $k \in \mathbb{N}(p)$. In the remainder of this paper, Eq. (4) will be considered the governing equations for the reduced-order model.

2.3. Objective: adaptive refinement

The goal of this work is as follows: given a reduced basis $\boldsymbol{V}$ and online ROM solution $\hat{\boldsymbol{x}}^k$ to Eq. (4) for iteration k, 1) detect if the solution is sufficiently accurate, 2) if it is not sufficiently accurate, efficiently generate a higher-dimensional reduced basis $\boldsymbol{V}'$ with $\text{range}\,(\boldsymbol{V}) \subseteq \text{range}\,(\boldsymbol{V}')$ in a goal-oriented manner that aims to reduce errors in the output z^k, 3) compute an associated solution $\hat{\boldsymbol{x}}'^k$, 4) repeat until desired accuracy is reached.

To generate this hierarchy of subspaces efficiently, we propose an analogue to adaptive h-refinement, wherein selected basis vectors $\boldsymbol{v}_i$ are 'split' online into multiple vectors with disjoint support (i.e., the element set with nonzero entries). Like all h-refinement techniques, the proposed method consists of the following components:

1. *Refinement mechanism.* In typical h-refinement, this is defined by the mesh-refinement method applied to finite elements or volumes. The proposed method refines the solution space by splitting the support of the basis vectors using a tree structure constructed via k-means clustering of the state variables. Section 3 describes this component.

2. *Error indicators.* Goal-oriented methods for h-refinement often 1) solve a coarse dual problem, 2) prolongate the adjoint solution to a representation on the fine grid, and 3) compute error estimates of the output using first-order analysis. The proposed method employs an analogous goal-oriented dual-weighted residual approach. Section 4 presents this.
3. *An adaptive algorithm.* The proposed algorithm identifies when refinement is required online and employs error indicators decide on the particular refinement, i.e., which basis vectors should be refined, and how they should be refined. Section 5 provides this algorithm.

3. Refinement mechanism

The method assumes that an initial reduced basis $\boldsymbol{V}^{(\mathbf{0})} \in \mathbb{R}^{n\times p^{(0)}}$ is provided, which is subsequently 'split' to add fidelity to the ROM online. Section 3.1 describes the tree data structure that constitutes the splitting mechanism, Section 3.2 describes how this mechanism leads to an algebraic refinement strategy, Section 3.3 highlight critical properties of the refinement method, and Section 3.4 describes construction of the tree via k-means clustering.

3.1. Tree data structure

To begin, we define a tree data structure that characterizes the refinement mechanism. The tree is characterized by a *child* function $C : \mathbb{N}(m) \to \mathcal{P}(\mathbb{N}(m))$ that describes the topology of the tree and an *element* function $E : \mathbb{N}(m) \to \mathcal{P}(\mathbb{N}(n))$ that describes the set of nonzero vector entries associated with each tree node. Here, m denotes the number of nodes in the tree and $\mathcal{P}$ denotes the powerset.

Each basis vector $\boldsymbol{v}_i$, $i \in \mathbb{N}(p)$ is characterized by a particular node on the tree $d_i \in \mathbb{N}(m)$, a set of nonzero entries (i.e., support) $E(d_i)$, and possible splits $C(d_i)$. If a given vector $\boldsymbol{v}_i$ is split, then it is replaced in the basis by $q_i := \text{card}(C(d_i))$ child vectors whose set of nonzero entries is defined by $E(k)$, $k \in C(d_i)$; the values of these nonzero entries are the same as those of the original vector $\boldsymbol{v}_i$.

We enforce the following conditions for the tree:

1. The root node includes all elements: $E(1) = \mathbb{N}(n)$, which is consistent with the possibly global support of the original reduced basis $\boldsymbol{V}^{(\mathbf{0})}$.
2. The children have disjoint support, and the union of their support equals that of the parent: For all $i \in \mathbb{N}(m)$,

$$E(j) \cap E(k) = \emptyset, \quad \forall j,k \in C(i),\ j \neq k \tag{5}$$

$$\bigcup_{j\in C(i)} E(j) = E(i). \tag{6}$$

3. Each element is associated with a single leaf node:

$$\forall l \in \mathbb{N}(n),\ \exists i \in \mathbb{N}(m) \mid E(i) = l,\ C(i) = \emptyset. \tag{7}$$

As will be shown, these requirements guarantee several critical properties of the method.

Example. Consider an example with $n = 6$ and an initial reduced basis $\boldsymbol{V}^{(\mathbf{0})} = \boldsymbol{v}_1^{(\mathbf{0})}$ of dimension 1. Figure 1 depicts an example of a tree structure for this case.

Suppose the basis has been split into $p = 4$ according to the tree in Figure 1 with $d_1 = 2$, $d_2 = 7$, $d_3 = 9$, and $d_4 = 10$; then, the refined reduced basis is

$$\boldsymbol{V} = \begin{bmatrix} v_{11}^{(0)} & 0 & 0 & 0 \\ 0 & v_{21}^{(0)} & 0 & 0 \\ v_{31}^{(0)} & 0 & 0 & 0 \\ v_{41}^{(0)} & 0 & 0 & 0 \\ 0 & 0 & v_{51}^{(0)} & 0 \\ 0 & 0 & 0 & v_{61}^{(0)} \end{bmatrix}. \tag{8}$$

■

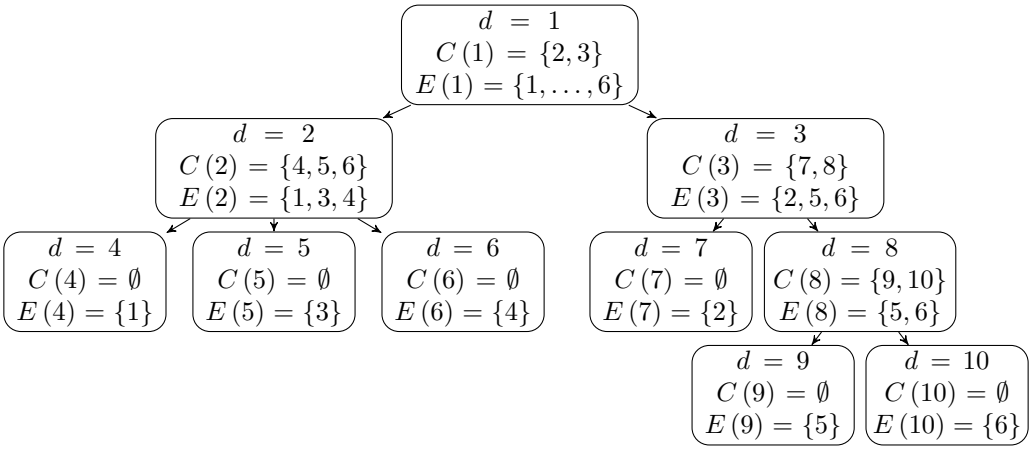


Figure 1: Tree example with $n = 6$

In the sequel, we overload the child function for the two-argument case such that $C(i, j)$ denotes the jth child node of parent node i, where ordering of the children is implied by the binary relation $\leq$ on the natural numbers. Similarly, the overloaded element function $E(i, j)$ is the jth element for node i; again, ordering of the elements is implied by the relation $\leq$ on the natural numbers.

3.2. Refinement via basis splitting

We now put the basis-splitting methodology in the framework of typical h-refinement techniques. First, define a 'coarse' basis $\boldsymbol{V}^H \in \mathbb{R}^{n \times p}$, which is initially equal to the nominal basis $\boldsymbol{V}^{(\mathbf{0})} \in \mathbb{R}^{n \times p^{(0)}}$ with $p^{(0)} \leq p$. As this initial basis may have global support, it is characterized by $d_i = 1$, $i \in \mathbb{N}(p^{(0)})$; this is permissible due to Condition 1 of Section 3.1. Also define a 'fine' basis corresponding to the coarse basis with all vectors split according to the children of the current node. We can express the relationship between the coarse and fine bases as

$$\boldsymbol{V}^H = \boldsymbol{V}^h \boldsymbol{I}_H^h, \tag{9}$$

where $\boldsymbol{V}^h \in \mathbb{R}^{n \times q}$ with $q \geq p$ denotes the fine basis and $\boldsymbol{I}_H^h \in \{0, 1\}^{q \times p}$ denotes the *prolongation* operator. Then, for any generalized coordinates $\hat{\boldsymbol{w}}^H \in \mathbb{R}^p$ associated with the coarse basis $\boldsymbol{V}^H$, we can compute the corresponding fine representation $\hat{\boldsymbol{w}}^h \in \mathbb{R}^q$ associated with the fine basis $\boldsymbol{V}^h$ as

$$\hat{\boldsymbol{w}}_H^h = \boldsymbol{I}_H^h \hat{\boldsymbol{w}}^H, \tag{10}$$

which ensures that $\boldsymbol{V}^H \hat{\boldsymbol{w}}^H = \boldsymbol{V}^h \hat{\boldsymbol{w}}_H^h$. Note this prolongation operator is exact, unlike typical mesh-refinement strategies, where this operator is often defined as a linear or quadratic interpolant of the coarse solution on the fine grid. The *restriction* operator is not uniquely defined, but can be set, e.g., to

$$\boldsymbol{I}_h^H = \left(\boldsymbol{I}_H^h\right)^+, \tag{11}$$

where the superscript + denotes the Moore–Penrose pseudoinverse.

Using the tree structure defined in Section 3.1, we can precisely define these quantities. We first introduce the mapping $f : (i, j) \mapsto k$, which provides the fine basis-vector index k corresponding to the jth child of the ith coarse basis vector. We define it as

$$f(i, j) = \sum_{k<i} q_k + j, \quad j \in \mathbb{N}(q_i), \quad i \in \mathbb{N}(p). \tag{12}$$

In particular, note that if node d_i is a leaf (i.e., $C(d_i) = \emptyset$), then $f(i, j)$ does not exist for any j. Similarly, the inverse mapping $f^{-1} : k \mapsto (i, j)$ yields the coarse basis-vector index i and child index j corresponding to fine basis vector k.

Now, the number of vectors in the fine reduced basis is simply

$$q = \sum_{i=1}^{p} q_i. \tag{13}$$

From Condition 2 of Section 3.1, we can write the fine reduced basis as

$$v_{ij}^h = \begin{cases} v_{il}^H, & \exists k \mid j = f(l,k),\ i \in E(C(d_l,k)) \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

and the prolongation operator induced by the proposed splitting scheme as

$$[\boldsymbol{I}_H^h]_{ij} = \begin{cases} 1, & \exists k \mid i = f(j,k) \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

3.3. Properties

This section highlights several key properties of this refinement method.

Lemma 1 (Hierarchical subspaces). *The method generates a hierarchy of subspaces such that* $\text{range}\left(\boldsymbol{V}^H\right) \subseteq \text{range}\left(\boldsymbol{V}^h\right)$.

Proof *This result is self-evident from Eq.* (9)*, as*

$$\text{range}\left(\boldsymbol{V}^H\right) = \{\boldsymbol{V}^h\boldsymbol{w} \mid \boldsymbol{w} \in \text{range}\left(\boldsymbol{I}_H^h\right) \subseteq \mathbb{R}^q\} \subseteq \{\boldsymbol{V}^h\boldsymbol{w} \mid \boldsymbol{w} \in \mathbb{R}^q\} = \text{range}\left(\boldsymbol{V}^h\right). \tag{16}$$

Theorem 1 (Monotonic convergence). *If the reduced-order model* (4) *is a priori convergent, i.e., its solution satisfies*

$$\boldsymbol{V}\hat{\boldsymbol{x}}^k = \arg\min_{\boldsymbol{w}\in\text{range}(\boldsymbol{V})} \|\boldsymbol{x}^k - \bar{\boldsymbol{x}} - \boldsymbol{w}\|_{\boldsymbol{\Theta}}, \tag{17}$$

for some norm $\|\cdot\|_{\boldsymbol{\Theta}}$*, then the proposed refinement method guarantees monotonic convergence of the reduced-order-model solution, i.e.,*

$$\|\boldsymbol{x}^k - \bar{\boldsymbol{x}} - \boldsymbol{V}^h(\hat{\boldsymbol{x}}^h)^k\|_{\boldsymbol{\Theta}} \leq \|\boldsymbol{x}^k - \bar{\boldsymbol{x}} - \boldsymbol{V}^H(\hat{\boldsymbol{x}}^H)^k\|_{\boldsymbol{\Theta}}. \tag{18}$$

Proof *This follows directly from Lemma 1, as the coarse-basis solution is contained in the span of the fine basis* $\boldsymbol{V}^H(\hat{\boldsymbol{x}}^H)^k \in \text{range}\left(\boldsymbol{V}^H\right) \subseteq \text{range}\left(\boldsymbol{V}^h\right)$.

One example of a reduced-order model that satisfies the conditions of Theorem 1 arises when the residual is linear in the state and its Jacobian $\partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})$ is symmetric and positive definite. In this case, a Galerkin-projection ROM satisfies Eq. (17) for $\boldsymbol{\Theta} = \partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})$ with $\|\boldsymbol{w}\|_{\partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})} := \sqrt{\boldsymbol{w}^T\partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})\,\boldsymbol{w}}$. Another example is least-squares Petrov–Galerkin applied to a parametrized system of linear equations [22], where $\boldsymbol{\Theta} = \left(\partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})\right)^T\partial\boldsymbol{r}^k/\partial\boldsymbol{x}(\boldsymbol{\mu})$.

Theorem 2 (Convergence to the full-order model). *If every element has a nonzero entry in one of the original reduced-basis vectors, i.e.,*

$$\forall l \in \mathbb{N}(n),\ \exists\ (i,j) \in \mathbb{N}(n) \times \mathbb{N}(p^{(0)}) \mid v_{ij}^{(0)} \neq 0, \tag{19}$$

and Eq. (7) *holds, then a completely split basis yields a reduced-order model equivalent to the full-order model.*

Proof *Under these conditions, a completely split basis can be written as* $\boldsymbol{V} \in \mathbb{R}^{n\times np^{(0)}}$ *with all basis vectors in the leaf-node state, i.e.,* $C(d_i) = \emptyset$*,* $i \in \mathbb{N}(np^{(0)})$*. Because Eq.* (7) *guarantees that each element is associated with a single leaf node, this implies that*

$$\forall l \in \mathbb{N}(n),\ \exists\ i \in \mathbb{N}(np^{(0)}) \mid \boldsymbol{v}_i = \boldsymbol{e}_l\beta_i, \tag{20}$$

where $\boldsymbol{e}_l \in \{0,1\}^n$ *denotes the* l*th canonical unit vector and* $\beta_i \neq 0$, $i \in \mathbb{N}(np^{(0)})$. *Eq.* (20) *implies that the completely split basis can be post-multiplied by a (weighted) permutation matrix to yield the* $n \times n$ *identity matrix* $\boldsymbol{I}_n$, *i.e.,*

$$\boldsymbol{I}_n = \boldsymbol{V}\boldsymbol{\Gamma}. \tag{21}$$

Here, the matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{np^{(0)} \times n}$ *consists of columns*

$$\boldsymbol{\gamma}_l = \frac{1}{\beta_i}\boldsymbol{e}_i, \quad i \in \{j \mid \boldsymbol{v}_j = \boldsymbol{e}_l\beta_j\}, \quad l \in \mathbb{N}(n). \tag{22}$$

Eq. (21) *implies that*

$$\text{range}\,(\boldsymbol{I}_n) = \mathbb{R}^n \subseteq \text{range}\,(\boldsymbol{V}) \subseteq \mathbb{R}^n, \tag{23}$$

which completes the proof.

Lemma 1 and Theorem 2 show that the proposed refinement method enables the reduced-order model to generate a sequence of hierarchical subspaces that converges to the full-order model under modest assumptions. Thus, the method acts as a 'failsafe' mechanism: it allows the reduced-order model to generate *arbitrarily accurate solutions*. Despite this result, the associated rate of convergence is unknown, which precludes any *a priori* guarantee that the h-adaptive ROM will remain truly low dimensional for stringent accuracy requirements. However, numerical experiments in Section 6 demonstrate that the proposed method often leads to accurate responses with low-dimensional refined bases.

Remark. Note that the refinement method does not preclude a rank-deficient basis; this can be seen from Theorem 2, wherein a completely split basis has $np^{(0)} \geq n$ columns. To detect (and remove) rank deficiency, the refinement algorithm computes a rank-revealing QR factorization after each split (Steps 14–15 of Algorithm 4 and Steps 28–29 of Algorithm 5). ■

3.4. Tree construction via k-means clustering of the state variables

Any tree that satisfies Conditions 1–3 of Section 3.1 will lead to the critical properties proved in Section 3.3. This section presents one such tree-construction approach, which executes offline and employs the following heuristic:

> State variables x_i that tend to be strongly positively or negatively correlated can be accurately represented by the same generalized coordinate, and should therefore reside in the same tree node.

Example. To justify this heuristic, consider an example with $n = 6$ degrees of freedom and $n_o = 8$ observations of the state, e.g., from a computed time history. Assume that snapshots can be decomposed as

$$\boldsymbol{X} = \sum_{i=1}^{3} \boldsymbol{y}_i \boldsymbol{z}_i^T + 0.1\boldsymbol{E} \tag{24}$$

where $\boldsymbol{E} \in [-1,1]^{n \times n_o}$ is a matrix of random uniformly distributed noise and the data matrices are

$$\boldsymbol{Z} = \begin{bmatrix} -2.2083 & -5.1072 & 2.6816 & 9.3277 & -6.4506 & -3.2548 & 4.2237 & -3.2557 \\ -2.9810 & 0.6557 & 3.0474 & 5.5252 & 2.7674 & 2.3311 & 9.6190 & -6.6484 \\ -2.4547 & 5.2676 & -3.6434 & 5.5661 & -7.5449 & 9.3079 & -2.0459 & -0.0728 \end{bmatrix}^T$$

$$\boldsymbol{Y} = \begin{bmatrix} -3.9885 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8.6843 & 0 & 0 & -1.6393 \\ 0 & -1.7288 & 0 & 6.0559 & 2.2407 & 0 \end{bmatrix}^T.$$

The sparsity structure of $\boldsymbol{Y}$ implies that the following sets of state variables are strongly correlated or anti-correlated across observations: $\{1\}$, $\{3,6\}$, and $\{2,4,5\}$. This is apparent from computing the matrix of sample correlation coefficients:

$$\boldsymbol{R} = \begin{bmatrix} 1.0000 & 0.1526 & -0.5698 & -0.1534 & -0.1554 & 0.5705 \\ 0.1526 & 1.0000 & -0.0180 & -1.0000 & -1.0000 & 0.0198 \\ -0.5698 & -0.0180 & 1.0000 & 0.0209 & 0.0212 & -1.0000 \\ -0.1534 & -1.0000 & 0.0209 & 1.0000 & 1.0000 & -0.0227 \\ -0.1554 & -1.0000 & 0.0212 & 1.0000 & 1.0000 & -0.0229 \\ 0.5705 & 0.0198 & -1.0000 & -0.0227 & -0.0229 & 1.0000 \end{bmatrix}. \tag{25}$$

Suppose we start with a one-dimensional reduced basis corresponding to the first left singular vector of $\boldsymbol{X}$

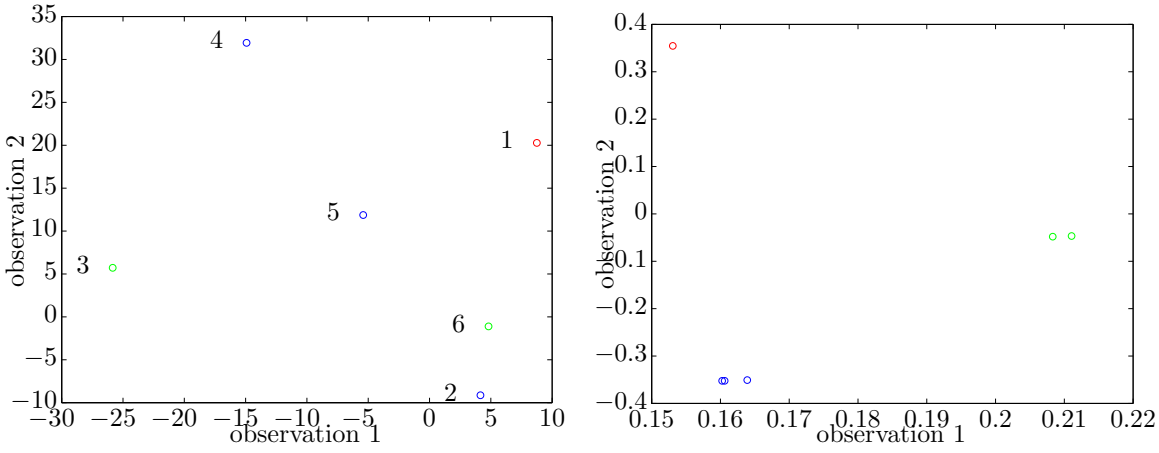
$$\boldsymbol{V}^{(\mathbf{0})} = \boldsymbol{V}^H = \boldsymbol{v}_1^H = \begin{bmatrix} -0.2609 & -0.0348 & 0.9390 & 0.1240 & 0.0463 & -0.1773 \end{bmatrix}^T.$$

Because the data nearly lie in a three-dimensional subspace of $\mathbb{R}^6$, the optimal performance of a refinement scheme would yield small error after splitting this one-dimensional basis into a basis of dimension three. Thus, consider splitting $\boldsymbol{V}^H$ into three children using a tree that follows the stated heuristic, i.e., is characterized by $C(1) = \{2,3,4\}$, $E(2) = \{1\}$, $E(3) = \{3,6\}$, and $E(4) = \{2,4,5\}$. The resulting basis becomes

$$\boldsymbol{V}^h = \begin{bmatrix} -0.2609 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9390 & 0 & 0 & -0.1773 \\ 0 & -0.0348 & 0 & 0.1240 & 0.0463 & 0 \end{bmatrix}^T.$$

The resulting projection error of the data is merely $\|\boldsymbol{X} - \boldsymbol{V}^h \left(\boldsymbol{V}^h\right)^+ \boldsymbol{X}\|_F / \|\boldsymbol{X}\|_F = 0.0033$. By contrast, generating an alternative three-dimensional fine basis $\bar{\boldsymbol{V}}^h$ by splitting the basis using a (similar) tree characterized by $E(2) = \{1\}$, $E(3) = \{3,5\}$, $E(3) = \{2,4,6\}$, yields a much larger error of $\|\boldsymbol{X} - \bar{\boldsymbol{V}}^h \left(\bar{\boldsymbol{V}}^h\right)^+ \boldsymbol{X}\|_F / \|\boldsymbol{X}\|_F = 0.4948$.

One way to identify these correlated variables is to employ k-means clustering [24] after pre-processing the data by 1) normalizing observations of each variable (to enable clustering to detect correlation), and 2) negating the observation vector if the first observation is negative (to enable clustering to detect anti-correlation). This is visualized in Figure 2 for the current example. Note that correlated and anti-correlated variables have a small Euclidean distance between them after this processing; this allows k-means clustering to identify them as a group. ■



(a) before processing (state variables labeled) (b) after processing

Figure 2: First two observations of the state variables (i.e., first two columns of $\boldsymbol{X}$) for the example in Section 3.4. After processing these observations by normalization and origin flipping, correlated and anti-correlated state variables are separated by small geometric distances and can thus be grouped via clustering.

Algorithm 1 Tree construction via recursive k-means clustering (offline)

Input: n_o snapshots of the reference-centered[2] state in matrix form $\boldsymbol{X} \in \mathbb{R}^{n \times n_o}$, number of means $\bar{k}$
Output: child function C, element function E, and number of nodes m

for $i = 1, \ldots, n$ **do**
  Normalize rows of $\boldsymbol{X}$ to capture correlation by clustering $\boldsymbol{x}_i^T \leftarrow \boldsymbol{x}_i^T / \|\boldsymbol{x}_i^T\|$
  if $x_{i1} < 0$ **then** {Flip over origin to capture negative correlation by clustering}
    $\boldsymbol{x}_i^T \leftarrow -\boldsymbol{x}_i^T$
  end if
end for
Set root node to contain all elements $E(1) = \mathbb{N}(n)$.
Initialize recent-node set $D \leftarrow \{1\}$ and node count $m \leftarrow 1$.
while $\text{card}(D) > 0$ **do**
  $\bar{D} \leftarrow D$, $D \leftarrow \emptyset$
  for $i = 1, \ldots, \text{card}\left(\bar{D}\right)$ **do**
    Set splitting node to the ith element of the recent-node set $d \leftarrow \bar{D}(i)$, where ordering is implied by $\geq$ on the natural numbers.
    if $E(d) = \emptyset$ **then** {No elements to split}
      Continue
    end if
    Select snapshots of current elements $\bar{x}_{jk} \leftarrow x_{E(d,j)k}$, $j \in \mathbb{N}(\text{card}(E(d)))$, $k \in \mathbb{N}(n_o)$
    $(\bar{E}_1, \ldots, \bar{E}_{n_c}) = \text{kmeans}\left(\bar{\boldsymbol{X}}, \bar{k}\right)$, where $\bar{E}_j \subset \mathbb{N}(\text{card}(E(d)))$ denotes the set of elements in cluster j, and n_c denotes the number of non-empty clusters.
    if $n_c = 1$ **then** {Cannot have only one child}
      for $j = 1, \ldots, \text{card}(E(d))$ **do** {Make all children into leaf nodes}
        $\bar{E}_j = j$
      end for
    end if
    for $j = 1, \ldots, n_c$ **do**
      $m \leftarrow m + 1$
      $D \leftarrow D \cup m$
      $E(m) = \{E(d, j) \mid j \in \bar{E}_j\}$
      $C(d, j) = m$
    end for
  end for
end while

To this end, we construct the tree offline by recursively applying k-means clustering to observations of the state variables (after reference subtraction, normalization, and origin flipping). Algorithm 1 describes the method. The n_o observations of these variables are obtained from snapshot data, which are often available, e.g., when the reduced basis is constructed via proper orthogonal decomposition.

4. Dual-weighted residual error indicators

To compute error indicators for refinement, we propose a goal-oriented dual-weighted residual methodology based on adjoint solves. It can be considered a model-reduction adaptation of duality-based error-control methods developed for differential equations [25, 26], finite-element discretizations [27, 28, 29, 30], finite-volume discretizations [31, 32, 33], and discontinuous Galerkin discretizations [34, 35]. Because the proposed method performs refinement online at the iteration level, it requires error indicators associated with the error in ROM output at iteration k, i.e., $g(\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k; \boldsymbol{\mu})$. To simplify notation in this section, we set

[2] This implies that the reference state $\bar{\boldsymbol{x}}$ should be subtracted from the state snapshots.

$\bar{\boldsymbol{x}} = 0$ and write the associated single solve (Eq. (4) for a single iteration and parameter instance) simply as

$$\boldsymbol{V}^T \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{x}}) = 0. \tag{26}$$

First, we approximate the output due to the (unknown) fine solution $\hat{\boldsymbol{x}}^h$ to first-order about the coarse solution $\hat{\boldsymbol{x}}^H$:

$$g\left(\boldsymbol{V}^h\hat{\boldsymbol{x}}^h\right) \approx g\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right) + \frac{\partial g}{\partial \boldsymbol{x}}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)\boldsymbol{V}^h\left(\hat{\boldsymbol{x}}^h - \boldsymbol{I}_H^h\hat{\boldsymbol{x}}^H\right), \tag{27}$$

where we have used Eq. (9) to relate the coarse and fine bases. Similarly, we can approximate the fine residual to first order about the coarse solution as

$$0 = (\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^h\hat{\boldsymbol{x}}^h\right) \approx (\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right) + (\boldsymbol{V}^h)^T\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)\boldsymbol{V}^h\left(\hat{\boldsymbol{x}}^h - \boldsymbol{I}_H^h\hat{\boldsymbol{x}}^H\right). \tag{28}$$

Solving for the state error yields

$$\left(\hat{\boldsymbol{x}}^h - \boldsymbol{I}_H^h\hat{\boldsymbol{x}}^H\right) \approx -\left[(\boldsymbol{V}^h)^T\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)\boldsymbol{V}^h\right]^{-1}(\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right) \tag{29}$$

Substituting (29) in (27) yields

$$g\left(\boldsymbol{V}^h\hat{\boldsymbol{x}}^h\right) - g\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right) \approx -\left(\hat{\boldsymbol{y}}^h\right)^T(\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right). \tag{30}$$

where the fine adjoint solution $\hat{\boldsymbol{y}}^h \in \mathbb{R}^q$ satisfies

$$(\boldsymbol{V}^h)^T\frac{\partial \boldsymbol{r}^k}{\partial \boldsymbol{x}}(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H)^T\boldsymbol{V}^h\hat{\boldsymbol{y}}^h = (\boldsymbol{V}^h)^T\frac{\partial g}{\partial \boldsymbol{x}}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)^T. \tag{31}$$

Because we would like to avoid q-dimensional solves associated with the fine basis $\boldsymbol{V}^h$, we approximate $\hat{\boldsymbol{y}}^h$ as the prolongation of the coarse adjoint solution

$$\hat{\boldsymbol{y}}_H^h = \boldsymbol{I}_H^h\hat{\boldsymbol{y}}^H, \tag{32}$$

where $\hat{\boldsymbol{y}}^H$ satisfies

$$\left(\boldsymbol{V}^H\right)^T\frac{\partial \boldsymbol{r}^k}{\partial \boldsymbol{x}}(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H)^T\boldsymbol{V}^H\hat{\boldsymbol{y}}^H = \left(\boldsymbol{V}^H\right)^T\frac{\partial g}{\partial \boldsymbol{x}}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)^T \tag{33}$$

Substituting the approximation $\hat{\boldsymbol{y}}_H^h$ for $\hat{\boldsymbol{y}}^h$ in (30) yields a cheaply computable error estimate

$$g\left(\boldsymbol{V}^h\hat{\boldsymbol{x}}^h\right) - g\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right) \approx -\left(\hat{\boldsymbol{y}}_H^h\right)^T(\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right). \tag{34}$$

The right-hand side can be bounded as

$$\left|\left(\hat{\boldsymbol{y}}_H^h\right)^T(\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)\right| \leq \sum_{i\in\mathbb{N}(q)}\delta_i^h, \tag{35}$$

where the error indicators $\delta_i^h \in \mathbb{R}_+$, $i \in \mathbb{N}(q)$ are

$$\delta_i^h = \left|\left[\hat{y}_H^h\right]_i\left(\boldsymbol{v}_i^h\right)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right)\right|. \tag{36}$$

Meyer and Matties [36] also proposed a dual-weighted residual method for reduced-order models. However, their approach was not applied to adaptive refinement and did not consider a hierarchy of reduced bases; further, their proposed dual solve was carried out on the full-order model, which is infeasibly expensive for the present context.

Remark. Some mesh-refinement techniques [31, 32] advocate computing refinement indicators that minimize the error in the computable correction

$$\left(\hat{\boldsymbol{y}}^h - \hat{\boldsymbol{y}}_H^h\right)^T(\boldsymbol{V}^h)^T\boldsymbol{r}\left(\boldsymbol{V}^H\hat{\boldsymbol{x}}^H\right).$$

To approximate this quantity, they employ prolongation operators of varying fidelity, e.g., linear and quadratic interpolants. Such a strategy is not straightforwardly applicable to the current context, as the prolongation operator $\boldsymbol{I}_H^h$ is exact. ■

Algorithm 2 Error estimates (online)

Input: coarse reduced basis $\boldsymbol{V}^H$, coarse solution $\hat{\boldsymbol{x}}^H$
Output: fine reduced basis $\boldsymbol{V}^h$, fine error-estimate vector $\boldsymbol{\delta}^h$
1: Solve coarse adjoint problem (33) for $\hat{\boldsymbol{y}}^H$.
2: Define prolongation operator $\boldsymbol{I}_H^h$ via Eq. (15).
3: Define fine reduced basis $\boldsymbol{V}^h$ via Eq. (9) and fine representation of adjoint solution $\hat{\boldsymbol{y}}_H^h$ via Eq. (32)
4: Compute fine error-estimate vector $\boldsymbol{\delta}^h$ via Eq. (36)

Algorithm 3 Adaptive h-refinement (online)

Input: iteration k, basis $\boldsymbol{V}$, ROM solver tolerance ϵ_{ROM}, FOM solver tolerance ϵ
Output: updated basis $\boldsymbol{V}$, generalized state $\hat{\boldsymbol{x}}^k$
1: Compute ROM solution $\hat{\boldsymbol{x}}^k$ satisfying $\|\boldsymbol{V}^T\boldsymbol{r}^k(\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k;\boldsymbol{\mu})\| \leq \epsilon_{\text{ROM}}$.
2: **if** FOM not converged $\|\boldsymbol{r}^k(\bar{\boldsymbol{x}} + \boldsymbol{V}\hat{\boldsymbol{x}}^k;\boldsymbol{\mu})\| > \epsilon$ **then**
3: Refine basis via Algorithm 4: $\boldsymbol{V} \leftarrow \text{Refine}\left(\boldsymbol{V}, \hat{\boldsymbol{x}}^k\right)$.
4: Return to Step 1.
5: **end if**
6: **if** $\mod(k, n_{\text{reset}}) = 0$ **then**
7: Reset basis $\boldsymbol{V} \leftarrow \boldsymbol{V}^{(\mathbf{0})}$.
8: **end if**

5. Adaptive h-refinement algorithm

We now return to the original objective of this paper: adaptively refine the reduced-order model online. Algorithm 3 describes our proposed methodology for achieving this within a time-integration scheme. Step 1 first computes the reduced-order-model solution satisfying a tolerance ϵ_{ROM}. Then in Step 2, refinement occurs if the norm of the *full-order residual* is above a desired threshold ϵ. Note that other (inexpensive) error indicators could be used to flag refinement, e.g., error surrogates [37]. Refinement continues until this full-order tolerance is satisfied; note that any tolerance can be reached, as a completely split basis yields a reduced-order model equivalent to the full-order model (see Section 3.1). Finally, Step 7 resets the basis every n_{reset} time iterations. This ensures 1) the basis does not grow monotonically, and 2) work performed to refine the basis can be amortized over subsequent time steps, where the solution is unlikely to significantly change. Note that if Step 1 entails an iterative solve (e.g., Newton), then the pre-refinement solution can be employed as an initial guess.

Algorithm 4 describes the proposed method for refining the basis using the refinement mechanism and error indicators presented in Sections 3 and 4, respectively. Appendix Appendix A describes a more sophisticated approach wherein the basis vectors are not split into all possible children; the children are separated into groups, each of which contributes roughly the same fraction of that vector's error.

First, Step 1 of Algorithm 4 computes error estimates for the fine basis (i.e., current basis with all vectors split into all possible children) using the dual-weighted residual approach. Step 3 marks the parent basis vectors to refine: those with above-average error contribution from its children. Steps 5–8 split the parent vector i into vectors corresponding to its q_i children according to the defined tree. Steps 9–12 update the reduced basis and tree nodes. Because this split does not guarantee a full-ranks basis, Step 14 performs an efficient QR factorization with column pivoting to identify 'redundant' basis vectors. Step 15 subsequently removes these vectors from the basis and Step 16 performs the necessary bookkeeping for the tree nodes.

6. Numerical experiments: parameterized inviscid Burgers' equation

We assess the method's performance on the parameterized inviscid Burgers' equation. While simple, this problem is particularly challenging for reduced-order models. This arises from the fact that ROMs approximate the solution as a linear combination of spatially fixed reduced-basis functions; as such, they work well when the dynamics are primarily Eulerian, i.e., are fixed with respect to the underlying grid. However, when the dynamics are Lagrangian in nature and exhibit motion with respect to the underlying

Algorithm 4 Refine (online)

Input: initial basis $\boldsymbol{V}$, reduced solution $\hat{\boldsymbol{x}}$
Output: refined basis $\boldsymbol{V}$

Compute fine error-estimate vector and fine reduced basis via Algorithm 2:
$\left(\boldsymbol{\delta}^h, \boldsymbol{V}^h\right) \leftarrow$ Error estimates $(\boldsymbol{V}, \hat{\boldsymbol{x}})$.
Put local error estimates in parent–child format $\eta_{ij} = \delta^h_{f(i,j)}$, $i \in \mathbb{N}(p)$, $j \in \mathbb{N}(q_i)$.
Mark basis vectors to refine $I = \{i \mid \sum_j \eta_{ij} \geq 1/p \sum_{kj} \eta_{kj}\}$
for $i \in I$ **do** {Split $\boldsymbol{v}_i$ into q_i vectors}
  for $k \in \mathbb{N}(q_i)$ **do**
    $\boldsymbol{x}_k = \boldsymbol{v}^h_{f(i,k)}$
    $\bar{d}_k = C\,(d_i, k)$
  end for
  $\boldsymbol{v}_i \leftarrow \boldsymbol{x}_1$, $d_i \leftarrow \bar{d}_1$
  for $k = 2, \ldots q_i$ **do**
    $\boldsymbol{v}_{p+k-1} \leftarrow \boldsymbol{x}_k$, $d_{p+k-1} \leftarrow \bar{d}_k$,
  end for
end for
Compute thin QR factorization with column pivoting $\boldsymbol{V} = \boldsymbol{Q}\boldsymbol{R}$, $\boldsymbol{R}\bar{\boldsymbol{\Pi}} = \bar{\boldsymbol{Q}}\bar{\boldsymbol{R}}$.
Ensure full-rank matrix $\boldsymbol{V} \leftarrow \boldsymbol{V}\,[\bar{\boldsymbol{\pi}}_1 \;\cdots\; \bar{\boldsymbol{\pi}}_r]$, where r denotes the numerical rank of $\boldsymbol{R}$.
Update tree $[d_1 \;\cdots\; d_r] \leftarrow [d_1 \;\cdots\; d_p]\,[\bar{\boldsymbol{\pi}}_1 \;\cdots\; \bar{\boldsymbol{\pi}}_r]$.

grid (e.g., moving shocks), reduced-order models generally fail to capture the critical phenomenon at every time step and parameter instance.

We employ the problem setup described in Ref. [38]. Consider the parameterized initial boundary value problem

$$\frac{\partial u(x,\tau)}{\partial \tau} + \frac{1}{2}\frac{\partial\left(u^2\,(x,\tau)\right)}{\partial x} = 0.02e^{\mu_2 x} \tag{37}$$

$$u(0,\tau) = \mu_1,\ \forall \tau > 0 \tag{38}$$

$$u(x,0) = 1,\ \forall x \in [0,\ 100]\,, \tag{39}$$

where μ_1 and μ_2 are two real-valued input variables. Godunov's scheme discretizes the problem, which leads to a finite-volume formulation consistent with the original formulation in Eq. (1). The one-dimensional domain is discretized using a grid with 251 nodes corresponding to coordinates coordinates $x_i = i \times (100/250)$, $i = 0, \ldots, 250$. Hence, the resulting full-order model is of dimension $n = 250$. The solution $u(x,\tau)$ is computed in the time interval $\tau \in [0, 50]$ using a uniform computational time-step size $\Delta t = 0.05$, leading to $t = 1000$ total time steps.

For simplicity, we employ a POD–Galerkin ROM. During the offline stage, snapshots of the state are collected for the first t_{train} time steps at training inputs. Then, the initial condition is subtracted from these snapshots, and they are concatenated column-wise to generate the snapshot matrix. Finally, the thin singular value decomposition of the snapshot matrix is computed, and the initial reduced basis $\boldsymbol{V^{(0)}}$ is set to the first $p^{(0)}$ left singular vectors. During the online stage, a Galerkin projection is employed using this reduced basis. For all experiments, the initial condition is set to the reference condition, i.e., $\bar{\boldsymbol{x}} = \boldsymbol{x}^0$. For h-adaptivity, we set the number of means to $\bar{k} = 10$ in Algorithm 1. For Algorithm 2, the output of interest is set to the residual norm, i.e., $g(\boldsymbol{x}^k; \boldsymbol{\mu}) = \|\tilde{\boldsymbol{r}}^k(\boldsymbol{x}^k; \boldsymbol{\mu})\|_2^2$. For Algorithm 3, the ROM tolerance is set to $\epsilon_{\text{ROM}} = 5 \times 10^{-3}$.[3] The basis-reset frequency n_{reset} will vary during the experiments. Step 1 incurs a Newton solve; when refinement has occurred, the initial guess is set to the converged solution from the previous refinement level. Finally, the experiments employ the (more complex) Refine method defined by Algorithm 5 with a child-partition factor $\alpha = 2$.

[3] For the ROMs without adaptivity, the ROM convergence tolerance is set to $\epsilon_{\text{ROM}} = 1 \times 10^{-5}$.

Note that because the residual operator is nonlinear in the state, a projection alone is insufficient to generate computational savings over the full-order model. Future work will address extending the proposed h-refinement method to ROMs equipped with a complexity reduction mechanism such as empirical interpolation or gappy POD.

6.1. Fixed inputs

For this example, the input parameters are set to $\mu_1 = 3$ and $\mu_2 = 0.02$. However, the problem can be considered to be predictive, as we only collect snapshots in the time interval $\tau_{\text{train}} \in [0, 7.5]$, i.e., for the first $t_{\text{train}} = 150$ time steps. This choice is made to introduce a significant challenge for the ROM: while the (unrefined) reduced basis captures discontinuities that arise in the first 150 time steps, it will not capture such discontinuities that arise outside of this time interval.[4]

Table 1 reports results for typical POD–Galerkin ROMs of differing dimensions, as well as results for the proposed h-refinement method with different parameters and a FOM tolerance in Algorithm 3 of $\epsilon = 0.05$. Here, the relative error is defined as

$$\text{relative error} = \frac{1}{t}\sum_{k=1}^{t} \|u_{\text{FOM}}(\cdot,\tau^k) - u_{\text{ROM}}(\cdot,\tau^k)\|_{L_2} / \|u_{\text{FOM}}(\cdot,\tau^k)\|_{L_2}.$$

Figure 3 compares the solutions predicted by POD–Galerkin with no basis truncation (i.e., $p = 150$) and that of the proposed method with an initial basis size of $p^{(0)} = 10$ with $\boldsymbol{V}^{(\mathbf{0})} \in \mathbb{R}^{n\times p^{(0)}}$ and a basis-reset frequency of $n_{\text{reset}} = 50$.

	no adaptivity			h-adaptivity				
initial basis dimension $p^{(0)}$	10	45	150	5	10	20	10	10
basis-reset frequency n_{reset}				50	50	50	100	25
average basis dimension per Newton iteration $\bar{p}$	10	45	150	41.4	44.3	58	73	37
average number of Refine calls per time step				0.20	0.19	0.14	0.13	0.28
relative error (%)	45.8	43.9	8.5	0.3	0.5	0.2	0.2	0.3
online time (seconds)	1.4	2.14	5.77	5.53	4.63	7.27	6.90	7.46

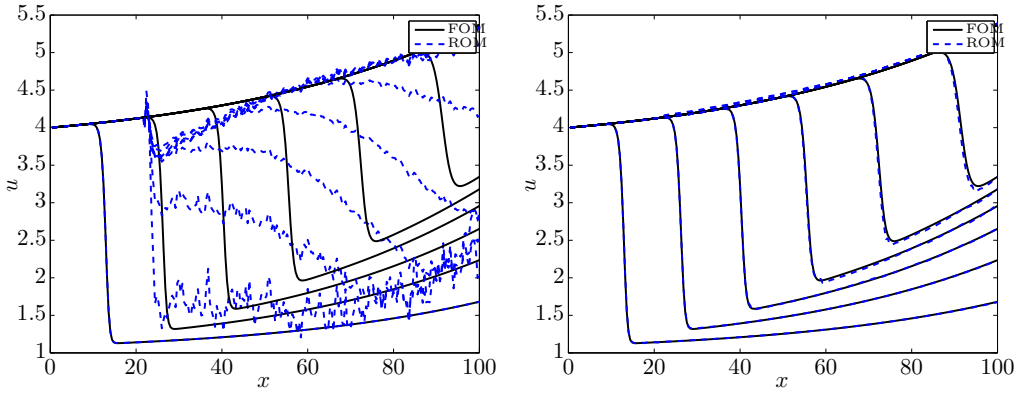
Table 1: Comparison between POD–Galerkin ROMs without refinement and with h-adaptive refinement for the fixed-inputs case.

First, note that the reduced-order model is highly inaccurate (even when the basis is not truncated) unless equipped with h-adaptivity. The reason for this is simple: the training has not captured the flow regime with shock locations past approximately $x = 60$. This illustrates a powerful capability of the proposed h-adaptation methodology: it enables ROMs to be incrementally refined to capture previously unobserved phenomena. In fact, the average basis dimension (per Newton iteration) for the best-performing h-adaptive ROM ($p^{(0)} = 10$, $n_{\text{reset}} = 50$) is only $\bar{p} = 44.3$, which is smaller than the basis dimensions for ROMs without adaptivity ($p = 45$ and $p = 150$) that yield much higher errors (43.9% and 8.5%, respectively).

Second, adaptation parameters $p^{(0)}$ and n_{reset} both lead to a performance tradeoff. When $p^{(0)}$ is small, it leads to smaller average basis sizes $\bar{p}$. However, it increases the number of Refine calls per time step, as the smaller basis must be refined more times to achieve desired accuracy. Similarly, resetting the basis more frequently (smaller n_{reset}) leads to a smaller $\bar{p}$, but more average refinement steps. As such, an intermediate value of both parameters leads to the shortest online evaluation time.

Finally, notice that the online evaluation time for the adaptive ROM with an average basis size of $\bar{p} = 44.3$ is roughly twice that of a non-adaptive ROM with roughly the same basis size $p = 45$. This discrepancy in evaluation time can be attributed to the overhead in performing the adaptation. For larger problem sizes, one would expect this overhead to be smaller relative to the total online evaluation time.

[4]Note that the refinement method can also be applied when the original reduced basis captures all relevant online phenomena; however, the need for *a posteriori* refinement is weaker in this case.



(a) no adaptivity, no truncation ($p = 150$)

(b) h-adaptivity, $p^{(0)} = 10$, $n_{\text{reset}} = 50$

Figure 3: Comparison of solutions computed by POD–Galerkin with and without h-adaptivity for the fixed-inputs case.

Next, we assess the performance of the h-refinement method as the full-order-model tolerance ϵ in Algorithm 3 varies. Table 2 and Figure 4 report the results. As expected, the proposed method allows the ROM to achieve any of the prescribed tolerances. As the tolerance becomes more rigorous, the ROM solution improves; however, it does so at increased computational cost, as both the average basis dimension $\bar{p}$ and number of Refine calls per time step increase to satisfy the requirement.

	$\epsilon = 0.35$	$\epsilon = 0.05$	$\epsilon = 0.01$
average basis dimension per Newton iteration $\bar{p}$	33.6	44.2507	53.9
average number of Refine calls per time step	0.115	0.189	0.212
relative error (%)	12.2	0.51	0.078
online time (seconds)	4.61	4.63	7.64

Table 2: Effect of full-order-model tolerance ϵ on h-adaptive refinement for $p^{(0)} = 10$ and $n_{\text{reset}} = 50$ for the fixed-inputs case.

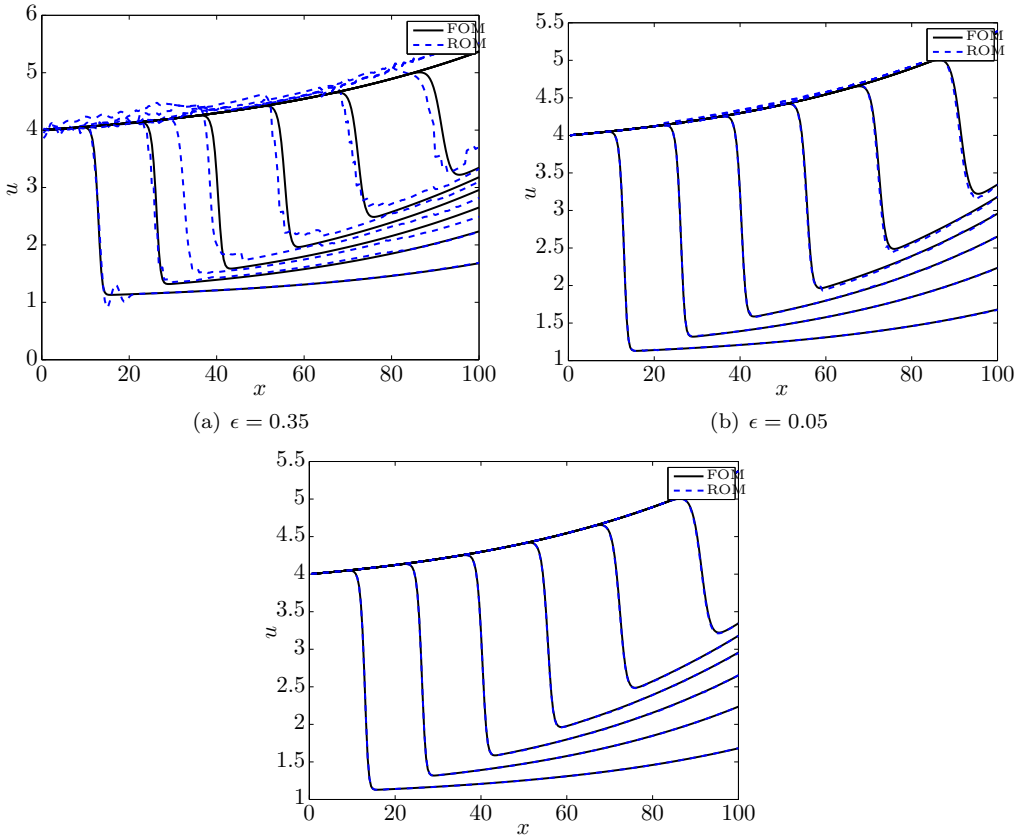
6.2. Input variation

For this experiment, we assess the proposed methodology in an input-varying scenario. In particular, the offline stage collects snapshots in the time interval $\tau_{\text{train}} \in [0, 2.5]$ for the training set $\{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^3\}$ described in Table 3, which is constructed by uniformly sampling the input space along $(\mu_1, \mu_2) = (3\alpha, 0.02\alpha)$, $\alpha \in [1, 3]$.

Figure 5 and Table 4 report the results for this experiment. The same phenomena are prevalent as were apparent in the previous experiment. The primary difference is that the POD–Galerkin model without adaptivity performs better than previously (due to more informative snapshots). However, h-adaptivity is still required to drive errors below 1%. Note that the proposed method compensated for an unsophisticated uniform-sampling of the input space. The method would still be applicable for more rigorous (e.g., POD–Greedy [39]) sampling methods, which would lead to a more robust initial basis $\boldsymbol{V}^{(\mathbf{0})}$ and reduce the burden of h-adaptivity to generate accurate results.

Table 3: Offline and online inputs for the inviscid Burgers equation

Input variables	Training point $\boldsymbol{\mu}^1$	Training point $\boldsymbol{\mu}^2$	Training point $\boldsymbol{\mu}^3$	Online point $\boldsymbol{\mu}^\star$
μ_1	3	6	9	4.5
μ_2	0.02	0.05	0.075	0.038

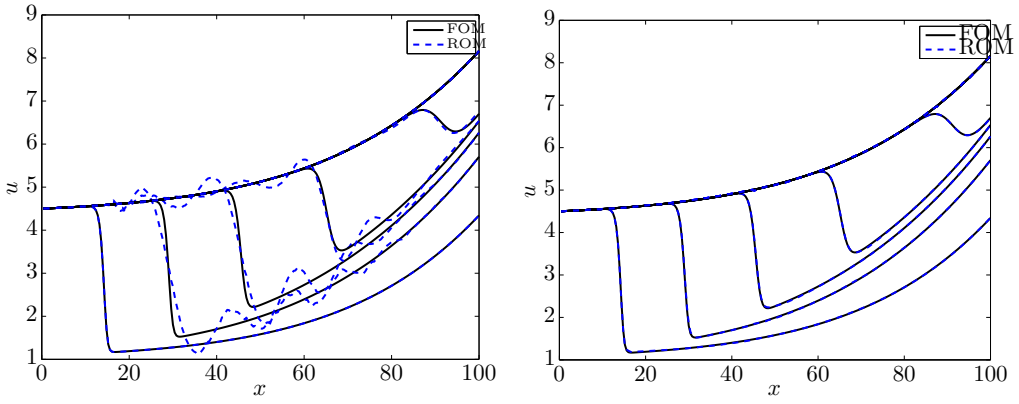


(a) $\epsilon = 0.35$

(b) $\epsilon = 0.05$

(c) $\epsilon = 0.01$

Figure 4: Comparison of solutions computed by h-adaptive POD–Galerkin for different full-order-model tolerances ϵ for the fixed-inputs case.



(a) no adaptivity, no truncation ($p = 150$)

(b) h-adaptivity, no truncation ($p^{(0)} = 20$), $n_{\text{reset}} = 100$

Figure 5: Comparison of solutions computed by POD–Galerkin with and without adaptivity for the varying-inputs case.

	no adaptivity			h-adaptivity				
initial basis dimension $p^{(0)}$	10	78	150	5	20	30	20	20
basis-reset frequency n_{reset}				100	100	100	200	50
average basis dimension per Newton iteration $\bar{p}$	10	78	150	69.8	77.2	87.6	130.6	65.6
average number of Refine calls per time step				0.20	0.072	0.07	0.044	0.11
relative error (%)	41.8	1.7	1.4	0.22	0.14	0.45	0.53	0.70
online time (seconds)	1.75	3.54	8.55	6.41	6.06	8.11	9.11	8.78

Table 4: Comparison between POD–Galerkin ROMs without refinement and with h-adaptive refinement for the input-variation case.

7. Conclusions

This work has presented an adaptive h-refinement method for reduced-order models. Key components include 1) an h-refinement mechanism based on basis splitting and tree structure constructed via k-means clustering, 2) dual-weighted residual error indicators, and 3) an adaptive algorithm to moderate when and how to perform the refinement. In contrast to existing *a priori* adaptive methods, the proposed technique provides a mechanism to improve the ROM solution *a posteriori*. As opposed to existing *a posteriori* methods, the proposal does so without incurring any large-scale operations. Numerical examples on the inviscid Burgers equation highlighted the method's ability to accurately predict phenomena not present in the training data used to construct the reduced basis.

Future research directions include incorporating complexity reduction (e.g., empirical interpolation, gappy POD) into the refinement process. In particular, as the reduced basis is refined, sample points (and dual reduced-basis vectors) must be added in a systematic way to ensure the reduced-order model remains solvable. Similar to the manner in which the tree defining the (complete) splitting mechanism is constructed offline, one could generate a hierarchy of these sample points offline from the training data, e.g., by executing [22, Algorithm 3] for n_s equal to the number of nodes in the mesh. In addition, it would be interesting to incorporate a more sophisticated adaptive *coarsening* technique (compared to the simple basis-resetting mechanism in Step 7 of Algorithm 3); for example, one could combine basis vectors whose generalized coordinates are strongly correlated (or anti-correlated) over recent time steps. Further, it would be interesting to pursue adaptive p-refinement methods, wherein other basis vectors (e.g., truncated POD vectors, discrete wavelets) with possibly global support are added from a library to enrich the reduced basis. In addition, it would be useful to pursue alternative tree-construction methods that satisfy Conditions 1–3 of Section 3.1. Assessing the effect of the proposed refinement method on ROM stability would also constitute an interesting investigation. Finally, it would be advantageous to incorporate Richardson extrapolation in the refinement method to better approximate the outputs of interest; however, this requires knowledge of the convergence rate of the reduced-order model with respect to adding basis vectors.

Appendix A. Refinement algorithm with multiple trees

This section presents a more sophisticated refinement mechanism than that that presented in Section 5. In particular, when a vector is flagged for refinement, it is not necessarily split into all its children. Rather, its children are separated into groups, each of which contributes roughly the same fraction α of the total error for that parent vector. This avoids over-refinement when the number of children is relatively large. However, this leads to an increase in required bookkeeping, as the tree structure changes when children merge: the tree must be altered and separately maintained for each vector. Thus, each basis vector $\boldsymbol{v}_i$, $i = 1, \ldots, p$ will be characterized by its own tree C_i, E_i with m_i nodes, as well as a node on that tree $d_i \in \mathbb{N}(m_i)$.

Algorithm 5 describes the modifications needed to Algorithm 4 to enable this feature. Key modifications include the following. Steps 7–22 separate the children of the parent vector's tree node d_i into groups; the resulting maintenance of the tree structures is performed in Steps 18–19.[5] In steps 23–26, not only is the

[5]Only the lower levels of the tree must be updated, as the current methodology never traverses up a tree.

Algorithm 5 Refine (child grouping) (online)

Input: initial basis $\boldsymbol{V}$, reduced solution $\hat{\boldsymbol{x}}$, child-partition factor $\alpha \leq 1$
Output: refined basis $\boldsymbol{V}$

1: Compute fine error-estimate vector and fine reduced basis via Algorithm 2:
   $\left(\boldsymbol{\delta}^h, \boldsymbol{V}^h\right) \leftarrow$ Error estimates $(\boldsymbol{V}, \hat{\boldsymbol{x}})$.
2: Put local error estimates in parent–child format $\eta_{ij} = \delta^h_{f(i,j)}$, $i \in \mathbb{N}(p)$, $j \in \mathbb{N}(q_i)$.
3: Mark basis vectors to refine $I = \{i \mid \sum_j \eta_{ij} \geq 1/p \sum_{kj} \eta_{kj}\}$
4: **for** $i \in I$ **do** {Split $\boldsymbol{v}_i$ into k vectors}
5: $p \leftarrow \dim(\text{range}(\boldsymbol{V}))$
6: Initialize additional-vector count $k \leftarrow 0$ and handled child-node set $D \leftarrow \emptyset$
7: **while** $D \neq \mathbb{N}(q_i)$ **do** {Divide child nodes into groups with roughly equal error}
8: $D_k = \arg\min_{z \subset K} \text{card}(z)$, where $K = \{z \subset \mathbb{N}(q_i) \setminus D \mid \sum_{j \in z} \eta_{ij} \geq \alpha \sum_j \eta_{ij}\}$.
9: **if** $D_k = \emptyset$ **then**
10: Take all remaining children $D_k = \mathbb{N}(q_i) \setminus D$
11: **end if**
12: $\boldsymbol{x}_k = \sum_{j \in D_k} \boldsymbol{v}^h_{f(i,j)}$
13: Update tree: $\bar{C}_k \leftarrow C_i$, $\bar{E}_k \leftarrow E_i$
14: **if** $\text{card}(D_k) = 1$ **then** {Use the same tree}
15: $\bar{d}_k = C_i(d_i, D_k)$
16: **else** {Alter the tree}
17: $\bar{d}_k = d_i$
18: $\bar{C}_k\left(\bar{d}_k\right) = \{C_i(d_i, k) \mid k \in D_k\}$
19: $\bar{E}_k\left(\bar{d}_k\right) = \bigcup_{k \in \bar{C}_k(d_i)} E_i(k)$
20: **end if**
21: $k \leftarrow k + 1$, $D \leftarrow D \cup D_k$
22: **end while**
23: $\boldsymbol{v}_i \leftarrow \boldsymbol{x}_0$, $C_i \leftarrow \bar{C}_0$, $E_i \leftarrow \bar{E}_0$ $d_i \leftarrow \bar{d}_0$
24: **for** $l = 1, \ldots, k$ **do**
25: $\boldsymbol{v}_{p+l} \leftarrow \boldsymbol{x}_l$, $C_{p+l} \leftarrow \bar{C}_l$, $E_{p+l} \leftarrow \bar{E}_l$, $d_{p+l} \leftarrow \bar{d}_l$
26: **end for**
27: **end for**
28: Compute thin QR factorization with column pivoting $\boldsymbol{V} = \boldsymbol{Q}\boldsymbol{R}$, $\boldsymbol{R}\bar{\boldsymbol{\Pi}} = \bar{\boldsymbol{Q}}\bar{\boldsymbol{R}}$.
29: Ensure full-rank matrix $\boldsymbol{V} \leftarrow \boldsymbol{V}[\bar{\boldsymbol{\pi}}_1 \ \cdots \ \bar{\boldsymbol{\pi}}_r]$, where r denotes the numerical rank of $\boldsymbol{R}$.
30: Update tree $[C_1 \ \cdots \ C_r] \leftarrow [C_1 \ \cdots \ C_p][\bar{\boldsymbol{\pi}}_1 \ \cdots \ \bar{\boldsymbol{\pi}}_r]$;
   $[E_1 \ \cdots \ E_r] \leftarrow [E_1 \ \cdots \ E_p][\bar{\boldsymbol{\pi}}_1 \ \cdots \ \bar{\boldsymbol{\pi}}_r]$;
   $[d_1 \ \cdots \ d_r] \leftarrow [d_1 \ \cdots \ d_p][\bar{\boldsymbol{\pi}}_1 \ \cdots \ \bar{\boldsymbol{\pi}}_r]$.

basis updated, but the trees are as well. Finally, Step 30 performs the necessary bookkeeping for the tree structures due to the removal of redundant basis vectors.

Acknowledgments

The author acknowledges Matthew Zahr for providing the model-reduction testbed that was modified to generate the numerical results, Seshadhri Comandur for helpful discussions related to tree construction and clustering, and the anonymous reviewers for providing insightful remarks and suggestions. This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000.

References

[1] Benner, P., Gugercin, S., and Willcox, K., "A survey of model reduction methods for parametric systems," *Max Planck Institute Magdeburg Preprints*, Vol. MPIMD/13–14, 2013.

[2] Patera, A. T. and Rozza, G., *Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations*, MIT, 2006.

[3] Gunzburger, M. D., Peterson, J. S., and Shadid, J. N., "Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data," *Computer methods in applied mechanics and engineering*, Vol. 196, No. 4, 2007, pp. 1030–1047.

[4] Heinkenschloss, M. and Vicente, L., "Analysis of inexact trust-region SQP algorithms," *SIAM Journal on Optimization*, Vol. 12, No. 2, 2002, pp. 283–302.

[5] Eldred, M. S., Weickum, G., and Maute, K., "A multi-point reduced-order modeling approach of transient structural dynamics with application to robust design optimization," *Structural and Multidisciplinary Optimization*, Vol. 38, No. 6, 2009, pp. 599–611.

[6] Arian, E., Fahl, M., and Sachs, E. W., "Trust-region proper orthogonal decomposition for flow control," Tech. Rep. 25, ICASE, 2000.

[7] Ryckelynck, D., "A priori hyperreduction method: an adaptive approach," *Journal of Computational Physics*, Vol. 202, No. 1, 2005, pp. 346–366.

[8] Carlberg, K. and Farhat, C., "An Adaptive POD-Krylov Reduced-Order Model for Structural Optimization," *8th World Congress on Structural and Multidisciplinary Optimization, Lisbon, Portugal*, June 1–5 2009.

[9] Amsallem, D. and Farhat, C., "An Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity," *AIAA Journal*, Vol. 46, No. 7, July 2008, pp. 1803–1813.

[10] Amsallem, D., Cortial, J., Carlberg, K., and Farhat, C., "A method for interpolating on manifolds structural dynamics reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 80, No. 9, 2009, pp. 1241–1258.

[11] Eftang, J. L., Patera, A. T., and Rønquist, E. M., "An 'hp' certified reduced basis method for parametrized elliptic partial differential equations," *SIAM Journal on Scientific Computing*, Vol. 32, No. 6, 2010, pp. 3170–3200.

[12] Haasdonk, B., Dihlmann, M., and Ohlberger, M., "A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space," *Mathematical and Computer Modelling of Dynamical Systems*, Vol. 17, No. 4, 2011, pp. 423–442.

[13] Drohmann, M., Haasdonk, B., and Ohlberger, M., "Adaptive Reduced Basis Methods for Nonlinear Convection–Diffusion Equations," *Finite Volumes for Complex Applications VI Problems & Perspectives*, Springer, 2011, pp. 369–377.

[14] Peherstorfer, B., Butnaru, D., Willcox, K., and Bungartz, H.-J., "Localized discrete empirical interpolation method," *SIAM Journal on Scientific Computing*, Vol. 36, No. 1, 2014, pp. A168—A192.

[15] Dihlmann, M., Drohmann, M., and Haasdonk, B., "Model reduction of parametrized evolution problems using the reduced basis method with adaptive time partitioning," .

[16] Amsallem, D., Zahr, M. J., and Farhat, C., "Nonlinear model order reduction based on local reduced-order bases," *International Journal for Numerical Methods in Engineering*, Vol. 92, No. 10, December 2012, pp. 891–916.

[17] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T., "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathématique Académie des Sciences*, Vol. 339, No. 9, 2004, pp. 667–672.

[18] LeGresley, P. A., *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*, Ph.D. thesis, Stanford University, 2006.

[19] Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing point estimation in models described by proper orthogonal decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.

[20] Chaturantabut, S. and Sorensen, D. C., "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[21] Drohmann, M., Haasdonk, B., and Ohlberger, M., "Reduced Basis Approximation for Nonlinear Parametrized Evolution Equations based on Empirical Operator Interpolation," *SIAM Journal on Scientific Computing*, Vol. 34, No. 2, 2012, pp. A937–A969.

[22] 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.

[23] Barone, M. F., Kalashnikova, I., Segalman, D. J., and Thornquist, H. K., "Stable Galerkin reduced order models for linearized compressible flow," *Journal of Computational Physics*, Vol. 228, No. 6, 2009, pp. 1932–1946.

[24] Lloyd, S., "Least squares quantization in PCM," *Information Theory, IEEE Transactions on*, Vol. 28, No. 2, 1982, pp. 129–137.

[25] Estep, D., "A posteriori error bounds and global error control for approximation of ordinary differential equations," *SIAM Journal on Numerical Analysis*, Vol. 32, No. 1, 1995, pp. 1–48.

[26] Pierce, N. A. and Giles, M. B., "Adjoint recovery of superconvergent functionals from PDE approximations," *SIAM review*, Vol. 42, No. 2, 2000, pp. 247–264.

[27] Babuška, I. and Miller, A., "The post-processing approach in the finite element method—part 1: Calculation of displacements, stresses and other higher derivatives of the displacements," *International Journal for numerical methods in engineering*, Vol. 20, No. 6, 1984, pp. 1085–1109.

[28] Becker, R. and Rannacher, R., *Weighted a posteriori error control in finite element methods*, Vol. preprint no. 96-1, Universitat Heidelberg, 1996.

[29] Rannacher, R., "The dual-weighted-residual method for error control and mesh adaptation in finite element methods," *MAFELEAP*, Vol. 99, 1999, pp. 97–115.

[30] Bangerth, W. and Rannacher, R., *Adaptive finite element methods for differential equations*, Springer, 2003.

[31] Venditti, D. and Darmofal, D., "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.

[32] 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.

[33] Park, M. A., "Adjoint-based, three-dimensional error prediction and grid adaptation," *AIAA journal*, Vol. 42, No. 9, 2004, pp. 1854–1862.

[34] 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.

[35] Fidkowski, K. J., *A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[36] Meyer, M. and Matthies, H., "Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods," *Computational Mechanics*, Vol. 31, No. 1, 2003, pp. 179–191.

[37] Drohmann, M. and Carlberg, K., "The ROMES method for reduced-order-model uncertainty quantification," *arXiv preprint 1405.5170*, 2014.

[38] Rewienski, M. J., *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[39] Haasdonk, B. and Ohlberger, M., "Reduced basis method for finite volume approximations of parametrized linear evolution equations," *ESAIM-Mathematical Modelling and Numerical Analysis*, Vol. 42, No. 02, 2008, pp. 277–302.

Chapter 4

The ROMES method for statistical modeling of reduced-order-model error

This chapter introduces the reduced-order model error surrogate (ROMES) method for statistically quantifying reduced-order model errors. Rather than quantify the error using an error bound, we instead construct a statistical model of the error, which leads to a *distribution* over the error for a given reduced-order-model simulation. This can be interpreted as the epistemic uncertainty introduced by the ROM, which can be incorporated into uncertainty-quantification tasks. This work has been submitted to the SIAM Journal on Uncertainty Quantification and is past the first round of revisions at the time of this writing.

SIAM/ASA J. Uncertainty Quantification
Vol. xx, pp. x

The ROMES method for statistical modeling of reduced-order-model error

Martin Drohmann* and Kevin Carlberg*

Abstract. This work presents a technique for statistically modeling errors introduced by reduced-order models. The method employs Gaussian-process regression to construct a mapping from a small number of computationally inexpensive 'error indicators' to a distribution over the true error. The variance of this distribution can be interpreted as the (epistemic) uncertainty introduced by the reduced-order model. To model normed errors, the method employs existing rigorous error bounds and residual norms as indicators; numerical experiments show that the method leads to a near-optimal expected effectivity in contrast to typical error bounds. To model errors in general outputs, the method uses dual-weighted residuals—which are amenable to uncertainty control—as indicators. Experiments illustrate that correcting the reduced-order-model output with this surrogate can improve prediction accuracy by an order of magnitude; this contrasts with existing 'multifidelity correction' approaches, which often fail for reduced-order models and suffer from the curse of dimensionality. The proposed error surrogates also lead to a notion of 'probabilistic rigor', i.e., the surrogate bounds the error with specified probability.

Key words. model reduction, uncertainty quantification, a posteriori error estimation, Gaussian processes, supervised machine learning

AMS subject classifications. 65G99, 65Y20, 62M86

1. Introduction.

As computing power increases, computational models of engineered systems are being employed to answer increasingly complex questions that guide decision making, often in time-critical scenarios. It is becoming essential to rigorously quantify and account for both aleatory and epistemic uncertainties in these analyses. Typically, the high-fidelity computational model can be viewed as providing a (costly-to-evaluate) mapping between system *inputs* (e.g., uncertain parameters, decision variables) and system *outputs* (e.g., outcomes, measurable quantities). For example, data assimilation employs collected sensor data (outputs) to update the distribution of uncertain parameters (inputs) of the model; doing so via Bayesian inference requires sampling from the posterior distribution, which can entail thousands of forward model simulations. The computational resources (e.g., weeks on a supercomputer) required for large-scale simulations preclude such high-fidelity models from being feasibly deployed in such scenarios.

To avoid this bottleneck, analysts have turned to surrogate models that approximate the input–output map of the high-fidelity model, yet incur a fraction of their computational cost. However, to be rigorously incorporated in uncertainty-quantification (UQ) contexts, it is critical to quantify the *additional uncertainty* introduced by such an approximation. For example, Bayesian inference aims to sample from the posterior distribution

$$\mathrm{P}[\boldsymbol{\mu}|\bar{s}] \propto \mathrm{P}[\boldsymbol{\mu}]\mathrm{P}[\bar{s}|\boldsymbol{\mu}], \tag{1.1}$$

*Sandia National Laboratories
7011 East Ave, MS 9159, Livermore, CA 94550, ({mdrohma, ktcarlb}@sandia.gov). Questions, comments, or corrections to this document may be directed to these email addresses.

where $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_{\boldsymbol{\mu}}}$ denote system inputs, $\bar{s} \in \mathbb{R}$ denotes the measured output,[1] $\mathrm{P}[\boldsymbol{\mu}]$ represents the prior, and $\mathrm{P}[\bar{s}|\boldsymbol{\mu}]$ denotes the likelihood function. Typically, the measured output is modeled as $\bar{s} = s(\boldsymbol{\mu}) + \varepsilon$, where $s : \mathcal{P} \to \mathbb{R}$ denotes the outputs predicted by the high-fidelity model for inputs $\boldsymbol{\mu}$, and ε is a random variable representing measurement noise. Sampling from this posterior distribution (e.g., via Markov-chain Monte–Carlo or importance sampling) is costly, as each sample requires at least one evaluation of the high-fidelity input–output map $\boldsymbol{\mu} \mapsto s$ that appears in the likelihood function.

When a surrogate model is employed, the measured output becomes $\bar{s} = s_{\mathrm{surr}}(\boldsymbol{\mu}) + \delta_s(\boldsymbol{\mu}) + \varepsilon$, where $s_{\mathrm{surr}} : \mathcal{P} \to \mathbb{R}$ denotes the output predicted by the surrogate model, and $\delta_s : \mathcal{P} \to \mathbb{R}$ represents the *surrogate-model output error* or bias. In this case, posterior sampling requires only evaluations of the *surrogate-model* input–output map $\boldsymbol{\mu} \mapsto s_{\mathrm{surr}}$—which is computationally inexpensive—as well as evaluation of the surrogate-model error $\delta_s(\boldsymbol{\mu})$, which is not precisely known in practice. As such, it can be considered a source of *epistemic uncertainty*, as it can be reduced in principle by employing the original high-fidelity model (or a higher fidelity surrogate model). The goal of this work is to construct a statistical model of this surrogate-model error $\widetilde{\delta}_s(\boldsymbol{\mu})$ that is 1) cheaply computable, 2) exhibits low variance (i.e., introduces minimal epistemic uncertainty), and 3) whose distribution can be numerically validated.

Various approaches have been developed for different surrogate models to quantify the surrogate error $\delta_s(\boldsymbol{\mu})$. Surrogate models can be placed into three categories [15]: 1) data fits, 2) lower-fidelity models, and 3) reduced-order models. Data fits employ supervised machine-learning methods (e.g., Gaussian processes, polynomial interpolation [17]) to directly model the high-fidelity input–output map. Within this class of surrogates, it is possible to statistically model the error for stochastic-process data fits, as a prediction for inputs $\boldsymbol{\mu}$ yields a mean $s_{\mathrm{surr}}(\boldsymbol{\mu})$ and a mean-zero distribution $\delta_s(\boldsymbol{\mu})$ that can be associated with epistemic uncertainty. While such models are (unbeatably) fast to query and non-intrusive to implement,[2] they suffer from the curse of dimensionality and lack access to the underlying model's physics, which can hinder predictive robustness.

Lower-fidelity models simply replace the high-fidelity model with a 'coarsened' model obtained by neglecting physics, coarsening the mesh, or employing lower-order finite elements, for example. While such models remain physics based, they often realize only modest computational savings. For such problems, 'multifidelity correction' methods have been developed, primarily in the optimization context. These techniques model the mapping $\boldsymbol{\mu} \mapsto \delta_s$ using a data-fit surrogate; they either enforce 'global' zeroth-order consistency between the corrected surrogate prediction and the high-fidelity prediction at training points [19, 23, 27, 33, 29], or 'local' first- or second-order consistency at trust-region centers [2, 15]. Such approaches tend to work well when the surrogate-model error exhibits a lower variance than the high-fidelity response [29] and the input-space dimension is small.

Reduced-order models (ROMs) employ a projection process to reduce the state-space dimensionality of the high-fidelity computational model. Although intrusive to implement, such physics-based surrogates often lead to more significant computational gains than lower-

[1] This work considers one output for notational simplicity. All concepts can be straightforwardly extended to multiple outputs. The numerical experiments treat the case of multiple outputs.

[2] Their construction requires only black-box evaluations of the input–output map of the high-fidelity model.

	ROM	data fits	multifidelity correction	ROM + ROMES
non–intrusive	×	✓	✓	×
output-error correction	×	N/A	✓	✓
rigorous error bounds	✓	×	×	(✓)*
tight error bounds	(✓)†	×	×	✓

* probabilistically rigorous
† good effectivity can only be obtained with very intrusive methods.

Table 1
Features of different surrogate models

fidelity models, and higher robustness than data fits. For such models, error analysis has been limited primarily to computing rigorous *a posteriori* error bounds $\Delta_s(\boldsymbol{\mu})$ satisfying $|\delta_s(\boldsymbol{\mu})| \leq \Delta_s(\boldsymbol{\mu})$ [7, 21, 36]. Especially for nonlinear problems, however, these error bounds are often highly ineffective, i.e., they overestimate the actual error by orders of magnitude [13]. To overcome this shortcoming and obtain tighter bounds, the ROM must be equipped with complex machinery that both increases the computational burden [41, 24] and is intrusive to implement (e.g., reformulate the discretization of the high-fidelity model [39, 42]). Further, *rigorous* bounds are not directly useful for uncertainty quantification (UQ) problems, where a statistical error model that is unbiased, has low variance, and is stochastic is more useful. Recent work [29, Section IV.D] has applied multifidelity correction to ROMs. However, the method did not succeed because the ROM error is often a highly oscillatory function of the inputs and therefore typically exhibits a *higher* variance than the high-fidelity response.

In this paper, we introduce the ROM Error Surrogates (ROMES) method that aims to combine the utility of multifidelity correction with the computational efficiency and robustness of reduced-order modeling. Table 1 compares the proposed approach with existing surrogate-modeling techniques. Similar to the multifidelity-correction approach, we aim to model the ROM error δ_s using a data-fit surrogate. However, as directly approximating the mapping $\boldsymbol{\mu} \mapsto \delta_s$ is ineffective for ROMs, we instead exploit the following key observation: ROMs often generate a small number of physics-based, cheaply computable *error indicators* $\boldsymbol{\rho} : \mathcal{P} \to \mathbb{R}^q$ that correlate strongly with the true error $\delta_s(\boldsymbol{\mu})$. Examples of indicators include the residual norm, dual-weighted residuals, and the rigorous error bounds discussed above. To this end, ROMES approximates the *low-dimensional, well-behaved* mapping $\boldsymbol{\rho}(\boldsymbol{\mu}) \mapsto \delta_s(\boldsymbol{\mu})$ using Gaussian-process regression, which is a stochastic-process data-fit method. Note that ROMES constitutes a generalization of the multifidelity correction approach, as the inputs (or features) of the error model can be any user-defined error indicator—they need not be the system inputs $\boldsymbol{\mu}$. Figure 1 depicts the propagation of information for the proposed method.

In addition to constructing an error surrogate for the system outputs, ROMES can also be used to construct a statistical model for the norm of the error in the system state. Further, ROMES can be used to generate error bounds with 'probabilistic rigor', i.e., an error bound that overestimates the error with a specified probability.

Next, Section 2 introduces the problem formulation and provides a general but brief in-

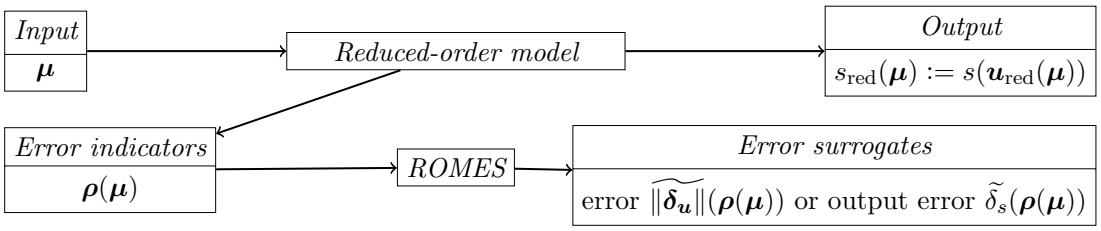


Figure 1. *ROMES method. The output quantities of interest can be 'corrected' by adding the ROM error surrogate to the ROM output prediction, i.e.,* $s(\boldsymbol{\mu}) \approx s_{corr}(\boldsymbol{\mu}) := s_{red}(\boldsymbol{\mu}) + \widetilde{\delta}_s(\boldsymbol{\rho}(\boldsymbol{\mu}))$.

troduction to model reduction. In Section 3, we introduce the ROMES method, including its objectives, ingredients, and some choices of these ingredients for particular errors. Section 4 briefly summarizes the Gaussian-process kernel method [35] and the relevance vector machine [38], which are the two machine-learning algorithms we employ to construct the ROMES surrogates. However, the ROMES methodology does not rely on these two techniques, as any supervised machine learning algorithm that generates a stochastic process can be used, as long as it generates a statistical model that meets the important conditions described in Section 3.1. Section 5 analyses the performance of the method when applied with the reduced-basis method to solve Poisson's equation in two dimensions using nine system inputs. The method is also compared with existing rigorous error bounds for normed errors, and with the multifidelity correction approach for errors in general system outputs.

For additional information on the reduced-basis method, including the algorithms to generate the reduced-basis spaces and the computation of error bounds, we refer to the supplementary Section S1.

2. Problem formulation.

This section details aspects of the high-fidelity and reduced-order models that are important for the ROMES surrogates. We begin with a formulation of the high-fidelity model in Section 2.1 and the reduced-order model in Section 2.2. Finally, we elaborate on the errors introduced by the model-reduction process and possible problems with their quantification in Section 2.3.

2.1. High-fidelity model.

Consider solving a parameterized systems of equations

$$\boldsymbol{r}(\boldsymbol{u}; \boldsymbol{\mu}) = 0, \tag{2.1}$$

where $\boldsymbol{u} : \mathcal{P} \to \mathbb{R}^n$ denotes the state implicitly defined by Eq. (2.1), $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{n_{\boldsymbol{\mu}}}$ denotes the system inputs, and $\boldsymbol{r} : \mathbb{R}^n \times \mathbb{R}^{n_{\boldsymbol{\mu}}} \to \mathbb{R}^n$ denotes the residual operator. This model is appropriate for stationary problems, e.g., those arising from the finite-element discretization of elliptic PDEs. For simplicity, assume we are interested in computing a single output

$$s(\boldsymbol{\mu}) := g(\boldsymbol{u}(\boldsymbol{\mu})) \tag{2.2}$$

with $s : \mathbb{R}^{n_{\boldsymbol{\mu}}} \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$.

When the dimension n of the high-fidelity model is 'large', computing the system outputs s by first solving Eq. (2.1) and subsequently applying Eq. (2.2) can be prohibitively expensive. This is particularly true for many-query problems arising in UQ such as Bayesian inference, which may require thousands of input–output map evaluations $\boldsymbol{\mu} \mapsto s$.

2.2. Reduced-order model. Model-reduction techniques aim to reduce the burden of solving Eq. (2.1) by employing a projection process. First, they execute a computationally expensive *offline stage* (e.g., solving Eq. (2.1) for a training set $\boldsymbol{\mu} \in \mathcal{P}_{\text{train}} \subset \mathcal{P}$) to construct 1) a low-dimensional trial basis (in matrix form) $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ with $p \ll n$ that (hopefully) captures the behavior of the state $\boldsymbol{u}$ throughout the parameter domain $\mathcal{P}$, and 2) an associated test basis $\boldsymbol{W} \in \mathbb{R}^{n \times p}$. Then, during the computationally inexpensive *online stage*, these methods approximately solve Eq. (2.1) for arbitrary $\boldsymbol{\mu} \in \mathcal{P}$ by searching for solutions in the trial subspace range $(\boldsymbol{V}) \subset \mathbb{R}^n$ and enforcing orthogonality of the residual $\boldsymbol{r}$ to the test subspace range $(\boldsymbol{W}) \subset \mathbb{R}^n$:

$$\boldsymbol{W}^t \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}; \boldsymbol{\mu}) = 0. \tag{2.3}$$

Here, the state is approximated as $\boldsymbol{u}_{\text{red}}(\boldsymbol{\mu}) := \boldsymbol{V}\hat{\boldsymbol{u}}(\boldsymbol{\mu})$ and the reduced state $\hat{\boldsymbol{u}}(\boldsymbol{\mu}) \in \mathbb{R}^p$ is implicitly defined by Eq. (2.3). The ROM-predicted output is then $s_{\text{red}}(\boldsymbol{\mu}) := g\left(\boldsymbol{u}_{\text{red}}(\boldsymbol{\mu}); \boldsymbol{\mu}\right)$.

When the residual operator is nonlinear in the state or non-affine in the inputs, additional complexity-reduction approximations such as empirical interpolation [5, 22], collocation [25, 3, 37], discrete empirical interpolation [12, 18, 14], or gappy proper orthogonal decomposition (POD) [10, 11] are required to ensure that computing the low-dimensional residual $\boldsymbol{W}^t\boldsymbol{r}$ incurs an n-independent operation count. In this case, the residual is approximated as $\tilde{\boldsymbol{r}} \approx \boldsymbol{r}$ and the reduced-order equations become

$$\boldsymbol{W}^t \tilde{\boldsymbol{r}}(\boldsymbol{V}\hat{\boldsymbol{u}}; \boldsymbol{\mu}) = 0. \tag{2.4}$$

When the output operator is nonlinear and the vector $\partial g / \partial \boldsymbol{u}$ is dense, approximations in the output calculation are also required to ensure an n-independent operation count.

Section S1 describes in detail the construction of a reduced-order model using the reduced-basis method applied to a parametrically coercive, affine, linear, elliptic PDE.

2.3. Reduced-order-model error bounds. One is typically interested in quantifying two types of error incurred by model reduction: the state-space error $\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu}) := \boldsymbol{u}(\boldsymbol{\mu}) - \boldsymbol{u}_{\text{red}}(\boldsymbol{\mu}) \in \mathbb{R}^n$ and the output error $\delta_s(\boldsymbol{\mu}) := s(\boldsymbol{\mu}) - s_{\text{red}}(\boldsymbol{\mu}) \in \mathbb{R}$. In particular, many ROMs are equipped with computable, rigorous error bounds for these quantities [31, 20, 30, 8, 13]:

$$\Delta_{\boldsymbol{u}}(\boldsymbol{\mu}) \geq \left\|\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})\right\|, \qquad \Delta_s(\boldsymbol{\mu}) \geq \left|\delta_s(\boldsymbol{\mu})\right| \tag{2.5}$$

In cases, where the norm of the residual operator can be estimated tightly, lower bounds also exist:

$$\Delta_{\boldsymbol{u}}^{\text{LB}}(\boldsymbol{\mu}) \leq \left\|\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})\right\|, \qquad \Delta_s^{\text{LB}}(\boldsymbol{\mu}) \leq \left|\delta_s(\boldsymbol{\mu})\right|. \tag{2.6}$$

The performance of an upper bound is usually quantified by its *effectivity*, i.e., the factor by which it overestimates the true error

$$\eta^{\boldsymbol{u}}(\boldsymbol{\mu}) := \frac{\Delta_{\boldsymbol{u}}(\boldsymbol{\mu})}{\left\|\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})\right\|} \geq 1, \qquad \eta^{s}(\boldsymbol{\mu}) := \frac{\Delta_s(\boldsymbol{\mu})}{\left|\delta_s(\boldsymbol{\mu})\right|} \geq 1. \tag{2.7}$$

The closer these values are to 1, the 'tighter' the bound. For coercive PDEs, these effectivities can be controlled by choosing a tight lower bound of the coercivity constant.

While this can be easily accomplished for stationary, linear problems, it is difficult to find tight lower bounds in almost all other cases. In fact, the resulting bounds often overestimate the error by orders of magnitude [36, 13]. Because effectivity is critically important in practice, various efforts have been undertaken to improve the tightness of the bounds. Huynh et al. [24] developed the successive constraint method for this purpose; the method approximates the coercivity-constant lower bounds by solving small linear programs online, which depend on additional expensive offline computations. Alternatively, Refs. [39, 42] reformulate the entire discretization of time-dependent problems using a space–time method that improves the error bounds by incorporating solutions to dual problems. Another approach [41] aims to approximate the coercivity constant by eigenvalue analysis of the reduced system matrices. These methods all bloat the offline and the online computation time and often incur intrusive changes to the high-fidelity-model implementation.

Regardless of effectivity, rigorous bounds satisfying inequalities (2.5) are not directly useful for quantifying the epistemic uncertainty incurred by employing the ROM. Rather, a *statistical model* that reflects our knowledge of these errors would be more appropriate. For such a model, the mean of the distribution would provide an expected error; the variance would provide a notion of epistemic uncertainty. The most straightforward way to achieve this would be to model the error as a uniform probability distribution on an interval whose boundaries correspond to the lower and upper bounds. Unfortunately, such an approach leads to wide, uninformative intervals when the bounds suffer from poor effectivity; this will be demonstrated in the numerical experiments.

Instead, we exploit the following observation: error bounds tend to be strongly correlated with the true error. Figure 2 depicts this observed structure for a reduced-basis ROM applied to an elliptic PDE (see Section 5.1 for details). On a logarithmic scale, the true error exhibits a roughly linear dependence on both the bound and the residual norm, and the variance of the data is fairly constant. As will be shown in Section 5.3, employing a multifidelity correction approach wherein the error is modeled as a function of the inputs $\boldsymbol{\mu}$ does not work well for this example, both because the input-space dimension is large (nine) and the error is a highly oscillatory function of these inputs.

Therefore, we propose constructing a stochastic process that maps such *error indicators* to a *random variable for the error*. For this purpose, we employ Gaussian-process regression. The approach leverages one strength of ROMs compared to other surrogate models: ROMs generate strong 'physics-based' error indicators (e.g., error bounds) in addition to output predictions. The next section describes the proposed method.

3. The ROMES method.

The objective of the ROMES method is to construct a statistical model of the deterministic, but generally unknown ROM error $\delta(\boldsymbol{\mu})$ with $\delta : \mathcal{P} \to \mathbb{R}$ denoting an $\mathbb{R}$-valued error that may represent the norm of the state-space error $\|\boldsymbol{\delta}_{\boldsymbol{u}}\|$, the output error δ_s, or its absolute value $|\delta_s|$, for example. The distribution of the random variable representing the error should reflect our (epistemic) uncertainty about its value. We assume that we can employ a set of training points $\delta(\boldsymbol{\mu}_n)$, $n = 1, \ldots, N$ to construct this model.

3.1. Statistical model.

Define a probability space $(\Omega, \mathcal{F}, P)$. We seek to approximate the deterministic mapping $m : \boldsymbol{\mu} \mapsto d(\delta(\boldsymbol{\mu}))$ by a stochastic mapping $\tilde{m} : \boldsymbol{\rho}(\boldsymbol{\mu}) \mapsto \tilde{d}$ with $\tilde{d} : \Omega \to \mathbb{R}$ a real-valued random variable. Here, $d : \mathbb{R} \to \mathbb{R}$ is an invertible *transformation function* (e.g.,

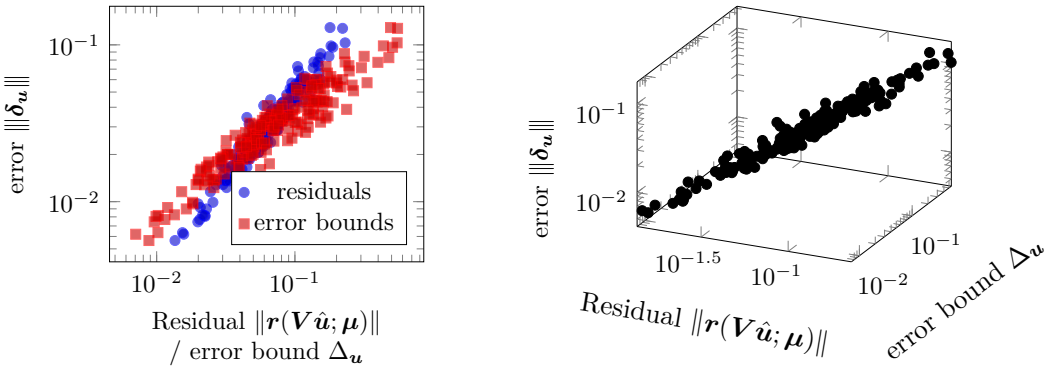


Figure 2. *Relationship between RB error bounds $\Delta_{\boldsymbol{u}}$, residual norms $\|\boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}};\boldsymbol{\mu})\|$, and the true state-space errors $|||\boldsymbol{\delta}_{\boldsymbol{u}}|||$, visualized by evaluation of 200 random sample points in the input space. Here, $|||\cdot|||$ denotes the energy norm defined in Section 5.1.*

logarithm) that can be specified to facilitate construction of the statistical model. We can then interpret the statistical model of the error as a random variable $\tilde{\delta} : \boldsymbol{\rho} \mapsto d^{-1}(\tilde{m}(\boldsymbol{\rho}))$.

Three ingredients must be selected to construct this mapping $\tilde{m}$: 1) the error indicators $\boldsymbol{\rho}$, 2) the transformation function d, and 3) the methodology for constructing the statistical model from the training data. We will make these choices such that the stochastic mapping satisfies the following conditions:

1. The indicators $\boldsymbol{\rho}(\boldsymbol{\mu})$ are *cheaply computable* and *low dimensional* given any $\boldsymbol{\mu} \in \mathcal{P}$. In practice, they should also incur a reasonably small implementation effort, e.g., not require modifying the underlying high-fidelity model.
2. The mapping $\tilde{m}$ exhibits *low variance*, i.e., $\mathrm{E}\left[(\tilde{m}(\boldsymbol{\rho}(\boldsymbol{\mu})) - \mathrm{E}\left[\tilde{m}(\boldsymbol{\rho}(\boldsymbol{\mu}))\right])^2\right]$ is 'small' for all $\boldsymbol{\mu} \in \mathcal{P}$. This ensures that little additional epistemic uncertainty is introduced.
3. The mapping $\tilde{m}$ is *validated*:

$$\omega_{\text{validation}}(\omega) \approx \omega, \quad \forall \omega \in [0,1), \tag{3.1}$$

where $\omega_{\text{validation}}(\omega)$ is the frequency with which validation data lie in the ω-confidence interval predicted by the statistical model

$$\omega_{\text{validation}}(\omega) := \frac{\mathrm{card}\left(\{\boldsymbol{\mu} \in \mathcal{P}_{\text{validation}} \mid d(\delta(\boldsymbol{\mu})) \in C_\omega(\boldsymbol{\mu})\}\right)}{\mathrm{card}\left(\mathcal{P}_{\text{validation}}\right)}. \tag{3.2}$$

Here, the validation set $\mathcal{P}_{\text{validation}} \subset \mathcal{P}$ should not include any of the points $\boldsymbol{\mu}_n$, $n = 1, \ldots, N$ employed to train the error surrogate, and the confidence interval $C_\omega(\boldsymbol{\mu}) \subset \mathbb{R}$, which is centered at the mean of $\tilde{m}(\boldsymbol{\rho}(\boldsymbol{\mu}))$, is defined for all $\boldsymbol{\mu} \in \mathcal{P}$ such that

$$\mathrm{P}[\tilde{m}(\boldsymbol{\rho}(\boldsymbol{\mu})) \in C_\omega(\boldsymbol{\mu})] = \omega. \tag{3.3}$$

In essence, validation assesses whether or not the data do indeed behave as random variables with probability distributions predicted by the statistical model.

The next section describes the proposed methodology for selecting indicators $\boldsymbol{\rho}$ and transformation function d. For constructing the mapping $\tilde{m}$ from training points, we will employ the two supervised machine learning algorithms described in Section 4: the Gaussian process (GP) kernel method and the relevance vector machine (RVM). Note that these are merely guidelines for model construction, as there are usually no strong analytical tools to prove that the mapping behaves according to a certain probability distribution. Therefore, any choice must be computationally validated according to condition 3 above.

3.2. Choosing indicators and transformation function. The class of multifidelity-correction algorithms can be cast within the framework proposed in Section 3.1. In particular, when a stochastic process is used to model additive error, these methods are equivalent to the proposed construction with ingredients $\delta = \delta_s$, $\boldsymbol{\rho} = \boldsymbol{\mu}$, and $d = \mathrm{id}_{\mathbb{R}}$ with $\mathrm{id}_{\mathbb{R}}(x) = x$, $\forall x \in \mathbb{R}$ the identity function over $\mathbb{R}$. However, as previously discussed, the mapping $\boldsymbol{\mu} \mapsto \delta_s$ can be highly oscillatory and non-smooth for reduced-order models; further, this approach is infeasible for high-dimensional input spaces, i.e., $n_{\boldsymbol{\mu}}$ large. This was shown by Ng and Eldred [29, Section IV.D]; we also demonstrate this in the numerical experiments of Section 5.3.

Note that all indicators and errors proposed in this section should be scaled (e.g., via linear transformations) in practice such that they exhibit roughly the same range. This 'feature scaling' task is common in machine-learning and is particularly important when the ROMES surrogate employs multiple indicators.

3.2.1. Normed and compliant outputs. As discussed in Section 2.3, many ROMs are equipped with bounds for normed errors. Further, there is often a strong, well-behaved relationship between such error bounds and the normed error (see Figure 2). In the case of compliant outputs, the error is always non-negative, i.e., $\delta_s = |\delta_s|$ (see Section S1.3), so we can treat this error as a normed error.

To this end, we propose employing error bounds as indicators for the errors in the compliant output $|\delta_s|$ and in the state $\|\boldsymbol{\delta_u}\|$. However, because the bound effectivity often lies in a small range (even for a large range of errors) [36], employing a logarithmic transformation is appropriate. To see this, consider a case where the effectivity η of the error bound, defined as

$$\eta(\boldsymbol{\mu}) := \frac{\Delta(\boldsymbol{\mu})}{\delta(\boldsymbol{\mu})} \geq 1, \quad \forall \boldsymbol{\mu} \in \mathcal{P}, \tag{3.4}$$

lies within a range $\eta_1 \leq \eta(\boldsymbol{\mu}) \leq \eta_2$, $\forall \boldsymbol{\mu} \in \mathcal{P}$. Then the relationship between the error bound and the true error is

$$\frac{\Delta(\boldsymbol{\mu})}{\eta_1} \geq \delta(\boldsymbol{\mu}) \geq \frac{\Delta(\boldsymbol{\mu})}{\eta_2} \tag{3.5}$$

$$\log \Delta(\boldsymbol{\mu}) - \log \eta_1 \geq \log \delta(\boldsymbol{\mu}) \geq \log \Delta(\boldsymbol{\mu}) - \log \eta_2 \tag{3.6}$$

for all $\boldsymbol{\mu} \in \mathcal{P}$. In this case, one would expect an affine model mapping $\log \Delta(\boldsymbol{\mu})$ to $\log \delta(\boldsymbol{\mu})$ with constant Gaussian noise to accurately capture the relationship. So, employing a logarithmic transformation permits the use of simpler surrogates that assume a constant variance in the error variable. Therefore, we propose employing $\boldsymbol{\rho} = \log \Delta$ and $d = \log$ for statistical modeling of normed errors and compliant outputs.

A less computationally expensive candidate for an indicator is simply the logarithm of the residual norm $\boldsymbol{\rho} = \log r$, where r is the Euclidean norm of the residual vector

$$r(\boldsymbol{\mu}) := \|\boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}(\boldsymbol{\mu}); \boldsymbol{\mu})\|_2 \,. \tag{3.7}$$

For more information on the efficient computation of (3.7), we refer to Section S1.3.1. One would expect this choice of indicator to produce a similar model to that produced by the logarithm of the error bound: the error bound is often equal to the residual norm divided by a (costly-to-compute) coercivity-constant lower bound (see Section S1.3). Further, employing the residual norm leads to a model that is less sensitive to strong variations in this approximated lower bound.

Returning to the example depicted in Figure 2, the relationship between the error bound and the energy norm of the state error in log-log space is roughly linear, and the variance is relatively small. The same is true of the relationship between the (computationally cheaper) residual norm and the true error. As expected, these relationships can be accurately modeled as a stochastic process with a linear mean function and constant variance (more details in Section 5.1). Here, strong candidates for ROMES error indicators include $\boldsymbol{\rho}_1(\boldsymbol{\mu}) := r(\boldsymbol{\mu}), \boldsymbol{\rho}_2(\boldsymbol{\mu}) := \Delta(\boldsymbol{\mu})$, and $\boldsymbol{\rho}_3(\boldsymbol{\mu}) := (r(\boldsymbol{\mu}), \Delta(\boldsymbol{\mu}))$. In the experiments in Section 5, we will consider the first choice, which is the least expensive and intrusive option, yet leads to excellent results. For cases where the data are less well behaved, more error indicators can be included, e.g., linear combinations of inputs or the output prediction.

Unfortunately, this set of ROMES ingredients is not applicable to errors in *general* outputs of interest because the logarithmic transformation function assumes strictly positive errors. The next section presents a strategy for handling this general case.

Remark 3.1 (Log-Normal distribution). *In the case where $d = \log$, the error models $\widetilde{\delta}(\boldsymbol{\mu})$, $\boldsymbol{\mu} \in \mathcal{P}$ are random variables with log-normal distribution. If one is interested in the most probable error, one might think to use the expected value of $\widetilde{\delta}$. However, the maximum of the probability distribution function of a log-normally distributed random variable is defined by its* mode*, which is less than the expected value. We therefore use* mode$(\widetilde{\delta})$ *if scalar values for the estimation of the output error or the reduced state error are required.*

3.2.2. General outputs. This section describes the ROMES ingredients we propose for modeling the error δ_s in a general output $s(\boldsymbol{\mu}) := g(\boldsymbol{u}(\boldsymbol{\mu}))$. Dual-weighted-residual approaches are commonly adopted for approximating general output errors in the context of *a posteriori* adaptivity [16, 4, 6, 34, 40, 26], model-reduction adaptivity [9], and model-reduction error estimation [32, 8, 39, 28]. The latter references compute adjoint solutions in order to improve the accuracy of ROM output-error bounds. The computation of these adjoint solutions entails a low-dimensional linear solve; thus, they are efficiently computable and can potentially serve as error indicators for the ROMES method.

The main idea of dual-weighted-residual error estimation is to approximate the output error to first-order using the solution to a dual problem. For notational simplicity in this section, we drop dependence on the inputs $\boldsymbol{\mu}$.

To begin, we approximate the output arising from the (unknown) high-fidelity state $\boldsymbol{u}$ to first order about the ROM-approximated state $\boldsymbol{V}\hat{\boldsymbol{u}}$:

$$g\left(\boldsymbol{u}\right) \approx g\left(\boldsymbol{V}\hat{\boldsymbol{u}}\right) + \frac{\partial g}{\partial \boldsymbol{u}}\left(\boldsymbol{V}\hat{\boldsymbol{u}}\right)\left(\boldsymbol{u} - \boldsymbol{V}\hat{\boldsymbol{u}}\right) \tag{3.8}$$

with $g : \mathbb{R}^n \to \mathbb{R}$ and $\frac{\partial g}{\partial \boldsymbol{u}} : \mathbb{R}^n \to \mathbb{R}^{1\times n}$. Similarly, we can approximate the residual to first order about the approximated state as

$$0 = \boldsymbol{r}(\boldsymbol{u}) \approx \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}) + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}(\boldsymbol{V}\hat{\boldsymbol{u}})(\boldsymbol{u} - \boldsymbol{V}\hat{\boldsymbol{u}}), \tag{3.9}$$

where $\boldsymbol{r} : \mathbb{R}^n \to \mathbb{R}^n$ with $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$. Solving for the error yields

$$(\boldsymbol{u} - \boldsymbol{V}\hat{\boldsymbol{u}}) \approx -\left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}(\boldsymbol{V}\hat{\boldsymbol{u}})\right]^{-1} \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}). \tag{3.10}$$

Substituting (3.10) in (3.8) leads to

$$g(\boldsymbol{u}) - g(\boldsymbol{V}\hat{\boldsymbol{u}}) \approx \boldsymbol{y}^T \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}), \tag{3.11}$$

where the dual solution $\boldsymbol{y} \in \mathbb{R}^n$ satisfies

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^t(\boldsymbol{V}\hat{\boldsymbol{u}})\boldsymbol{y} = -\frac{\partial g}{\partial \boldsymbol{u}}^t(\boldsymbol{V}\hat{\boldsymbol{u}}). \tag{3.12}$$

Approximation (3.11) is first-order accurate; therefore, it is exact in the case of linear outputs and a linear residual operator. In the general nonlinear case, this approximation is accurate in a neighborhood of the ROM-approximated state $\boldsymbol{V}\hat{\boldsymbol{u}}$.

Because we would like to avoid high-dimensional solves, we approximate $\boldsymbol{y}$ as the reduced-order dual solution $\boldsymbol{y}_{\text{red}} := \boldsymbol{Y}\hat{\boldsymbol{y}} \in \mathbb{R}^n$, where $\hat{\boldsymbol{y}}$ satisfies

$$\boldsymbol{Y}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^t(\boldsymbol{V}\hat{\boldsymbol{u}})\boldsymbol{Y}\hat{\boldsymbol{y}} = \boldsymbol{Y}^T \frac{\partial g}{\partial \boldsymbol{u}}^t(\boldsymbol{V}\hat{\boldsymbol{u}}), \tag{3.13}$$

and $\boldsymbol{Y} \in \mathbb{R}^{n\times p_y}$ with $p_y \ll n$ is a reduced basis (in matrix form) for the dual system. Section S1.3.2 provides details on the construction of $\boldsymbol{Y}$ for elliptic PDEs. Substituting the approximation $\boldsymbol{y}_{\text{red}}$ for $\boldsymbol{y}$ in (3.11) yields a cheaply computable error estimate

$$g(\boldsymbol{u}) - g(\boldsymbol{V}\hat{\boldsymbol{u}}) \approx \boldsymbol{y}_{\text{red}}^t \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}}). \tag{3.14}$$

This relationship implies that one can construct an accurate, cheaply computable ROMES model for general-output error $\delta = \delta_s = g(\boldsymbol{u}) - g(\boldsymbol{V}\hat{\boldsymbol{u}})$ by employing indicators $\boldsymbol{\rho} = \boldsymbol{y}_{\text{red}}^t \boldsymbol{r}(\boldsymbol{V}\hat{\boldsymbol{u}})$ and transformation function $d = \text{id}_{\mathbb{R}}$ the identity function over $\mathbb{R}$.

Remark 3.2 (Uncertainty control for dual-weighted-residual error indicators). *The accuracy of the reduced-order dual solution can be controlled by changing p_y—the dimension of the dual basis $\boldsymbol{Y}$. In general, one would expect an increase in p_y to lead to a lower-variance ROMES surrogate at the expense of a higher dimensional dual problem (3.13). The experiments in Section 5.6 highlight this uncertainty-control attribute.*

3.3. Probabilistically rigorous error bounds. Clearly, the ROMES surrogate does not strictly bound the error, even when error bounds are used as indicators. That is, the mean probability of overestimation is generally less than one

$$c := \frac{1}{|\mathcal{P}|}\int_{\boldsymbol{\mu}\in\mathcal{P}} \mathrm{P}[\tilde{m}(\boldsymbol{\rho}(\boldsymbol{\mu})) > d(\delta(\boldsymbol{\mu}))]d\boldsymbol{\mu} < 1. \tag{3.15}$$

This frequency of overestimation depends on the probability distribution of the random variable $\tilde{m}(\boldsymbol{\rho})$. Using the machine learning methods proposed in the next section, we infer normally distributed random variables

$$\tilde{m}(\boldsymbol{\rho}) \sim \mathcal{N}(\nu(\boldsymbol{\rho}), \overline{\sigma}^2(\boldsymbol{\rho})) \tag{3.16}$$

with mean $\nu(\boldsymbol{\rho})$ and variance $\overline{\sigma}^2(\boldsymbol{\rho})$. If the model is perfectly validated, then the mean probability of overestimation is $c = 0.5$. However, knowledge about the distribution of the random variable can be used to control the overestimation frequency. In particular, the modified surrogate

$$\tilde{m}^c(\boldsymbol{\rho}) := \tilde{m}(\boldsymbol{\rho}) + m_{\mathrm{LB}}(\boldsymbol{\rho}, c) \tag{3.17}$$

enables *probabilistic rigor*: it bounds the error with mean specified probability c assuming the model is perfectly validated. Here, m_{LB} fulfills

$$\mathrm{P}[X > m_{\mathrm{LB}}(\boldsymbol{\rho}, c)] = c, \qquad \text{for } X \sim \mathcal{N}(0, \overline{\sigma}^2(\boldsymbol{\rho})). \tag{3.18}$$

This value can be computed as

$$m_{\mathrm{LB}}(\boldsymbol{\rho}) = \sqrt{2}\overline{\sigma}(\boldsymbol{\rho})\mathrm{erf}^{-1}(2c-1) \tag{3.19}$$

where erf^{-1} is the inverse of the *error function*.

4. Gaussian processes. This section describes the two methods we employ to construct the stochastic mapping $\tilde{m} : \boldsymbol{\rho}(\boldsymbol{\mu}) \mapsto \tilde{d}$:

(i) *Gaussian process kernel regression* (i.e., kriging) [35] and
(ii) the *relevance vector machine* (RVM) [38].

Both methods are examples of supervised learning methods that generate a stochastic process from a set of N training points for independent variables $\mathbf{x} := (\boldsymbol{x}_n)_{n=1}^N$ and a dependent variable $\mathbf{y} := (y_n)_{n=1}^N$. Using these training data, the methods generate predictions $\tilde{y}(\boldsymbol{x}_m^*; \theta^{\mathrm{ML}})$, $m = 1, \ldots, M$ associated with a set of M prediction points $\mathbf{x}^* := (\boldsymbol{x}_m^*)_{m=1}^M$. Here, θ^{ML} denotes hyperparameters that are inferred using a Bayesian approach; the predictions are random variables with a multivariate normal distribution.

In the context of ROMES, the independent variables correspond to error indicators $\boldsymbol{x}_n = \boldsymbol{\rho}(\boldsymbol{\mu}_n)$ with $\boldsymbol{\mu}_n \in \mathcal{P}$, $n = 1, \ldots, N$ and the dependent variable corresponds to the (transformed) reduced-order-model error such that $y_n = d(\delta(\boldsymbol{\mu}_n))$, $n = 1, \ldots, N$. To make this paper as self-contained as possible, the following sections briefly present and compare the two approaches.

4.1. GP kernel method. A Gaussian process is defined as a collection of random variables such that any finite number of them has a joint Gaussian distribution. The GP kernel method constructs this Gaussian process via Bayesian inference using the training data and a specified kernel function. To begin, the prior distribution is set to

$$\tilde{y}_{\mathrm{prior}}(\underline{\mathbf{x}}) \sim \mathcal{N}\left(0, \mathbf{K}(\underline{\mathbf{x}}, \underline{\mathbf{x}}) + \sigma^2 \mathbf{I}_{N+M}\right) \tag{4.1}$$

with $\underline{\mathbf{x}} := (\underline{\boldsymbol{x}}_i)_{i=1}^{N+M} = (\mathbf{x}, \mathbf{x}^*)$. Here, the GP kernel assumes that the covariance between any two points can be described analytically by a kernel k with additive noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{M+N})$ such that

$$\mathbf{K}(\underline{\mathbf{x}}, \underline{\mathbf{x}}) = \left(k(\underline{\boldsymbol{x}}_i, \underline{\boldsymbol{x}}_j)\right)_{1 \le i,j \le N+M}. \tag{4.2}$$

In this work, we employ the most commonly used squared-exponential-covariance kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2l^2}\right), \tag{4.3}$$

which induces high correlation between geometrically nearby points. Here, $l \in \mathbb{R}$ is the 'width' hyperparameter.

Assuming the predictions are generated as independent samples from the stochastic process,[3] the GP kernel method then generates predictions for each point $\boldsymbol{x}^* \in \mathbf{x}^*$. These predictions correspond to random variables with posterior distributions $\tilde{y}(\boldsymbol{x}^*; \theta) \sim \mathcal{N}(\nu(\boldsymbol{x}^*), \overline{\sigma}^2(\boldsymbol{x}^*))$ with

$$\nu(\boldsymbol{x}^*) = \mathbf{K}(\boldsymbol{x}^*, \mathbf{x}) \left(\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{y} \tag{4.4}$$

$$\overline{\sigma}^2(\boldsymbol{x}^*) = \Sigma(\boldsymbol{x}^*) + \sigma^2 \tag{4.5}$$

$$\Sigma(\boldsymbol{x}^*) = \mathbf{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \mathbf{K}(\boldsymbol{x}^*, \mathbf{x}) \left(\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{K}(\mathbf{x}, \boldsymbol{x}^*). \tag{4.6}$$

More details on the derivation of these expressions can be found in Ref. [35, ch 2.2].

The hyperparameters $\theta := (l^2, \sigma^2)$ can be set to the maximum-likelihood values θ^{ML} computed as the solution to an optimization problem

$$\theta^{\mathrm{ML}} = \arg\max_{\theta} \mathcal{L}(\theta) \tag{4.7}$$

with the log-likelihood function defined as

$$\mathcal{L}(l^2, \sigma^2) = -\frac{1}{2} \mathbf{y}^t \left(\mathbf{K}(\mathbf{x}, \mathbf{x}; l^2) + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{y} - \frac{1}{2} \log \left|\mathbf{K}(\mathbf{x}, \mathbf{x}; l^2) + \sigma^2 \mathbf{I}_N\right| - \frac{N}{2} \log 2\pi. \tag{4.8}$$

For details on the derivation of the log likelihood function and problem (4.7), we refer to Ref. [35, ch 5.4].

Remark 4.1. *The noise component σ^2 of posterior covariance $\overline{\sigma}^2(\boldsymbol{x}^*)$ accounts for uncertainty in the assumed GP structure. It plays a crucial role for the ROMES method: it accounts for the non-uniqueness of the mapping $\boldsymbol{\rho} \mapsto \delta$, as it is possible for $\delta(\boldsymbol{\mu}_i) \neq \delta(\boldsymbol{\mu}_j)$ even if $\boldsymbol{\rho}(\boldsymbol{\mu}_i) = \boldsymbol{\rho}(\boldsymbol{\mu}_j)$. In particular, this noise component represents the 'information loss' incurred by employing the error indicators in lieu of the system inputs as independent variables in the GP. Therefore, this component can be interpreted as the* inherent uncertainty in the error due to the non-uniqueness of the mapping $\boldsymbol{\rho} \mapsto \delta$.

[3] Typically in the GP literature, predictions at all points $\mathbf{x}^*$ are generated simultaneously as a single sample from the Gaussian process. In this work, we treat all predictions as arising from independent samples of the GP.

On the other hand, the remaining term $\Sigma(\boldsymbol{x}^*)$ of the posterior variance quantifies the uncertainty in the mean prediction. This decreases as the number of training points increases. Therefore, $\Sigma(\boldsymbol{x}^*)$ can be interpreted as the uncertainty due to a lack of training data.

For example, the multifidelity-correction approach employs $\boldsymbol{\rho} = \boldsymbol{\mu}$ and therefore should be characterized by $\sigma^2 = 0$, as the mapping $\boldsymbol{\mu} \mapsto \delta$ is unique. However, due to the high-dimensional nature of the system-input space $\mathcal{P}$ in many problems, the uncertainty due to lack of training $\Sigma(\boldsymbol{x}^*)$ can be very large unless many training points are employed. On the other hand, the ROMES method aims to significantly reduce $\Sigma(\boldsymbol{x}^*)$ by employing a small number of indicators, albeit at the cost of a nonzero σ^2.

In light of this remark, we will employ two different types of ROMES models: one that includes the uncertainty due to a lack of training data (i.e., variance $\overline{\sigma}^2(\boldsymbol{x}^*)$), and one that neglects this uncertainty (i.e., variance σ^2).

4.2. Relevance vector machine (RVM) method. The RVM [38] is based on a parameterized discretization of the predictive random variable

$$\tilde{y}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \phi_k(\boldsymbol{x}) + \epsilon = \boldsymbol{\phi}(\boldsymbol{x})^t \mathbf{w} + \epsilon, \tag{4.9}$$

with specified basis functions $\boldsymbol{\phi}(\boldsymbol{x}) := [\phi_1(\boldsymbol{x}) \ \cdots \ \phi_K(\boldsymbol{x})]^t \in \mathbb{R}^K$, a corresponding set of random variables $\mathbf{w} := [w_1 \ \cdots \ w_K]^t \in \mathbb{R}^K$, with $w_k \sim \mathcal{N}(0, \beta_k^2)$ for $k = 1, \ldots, K$ and noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The hyperparameters $\boldsymbol{\beta} = [\beta_1 \ \cdots \ \beta_K]^t \in \mathbb{R}^K$ define the prior probability distribution, and are usually chosen by a likelihood maximization over the training samples. Radial basis functions

$$\phi_k^{RBF}(\boldsymbol{x}) = \exp\left(-\frac{1}{r^2}\left\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}\right\|_2^2\right), \quad k = 1, \ldots, K \tag{4.10}$$

constitute the most common choice for basis functions. For the ROMES method, we often expect a linear relationship between the indicators and true errors, likely with a small-magnitude high-order-polynomial deviation. Therefore, we also consider Legendre polynomials [1, Ch.8]

$$\phi_k^{Leb}(\boldsymbol{x}) = P_k(\boldsymbol{x}), \quad k = 1, \ldots, K. \tag{4.11}$$

Note that both sets of basis functions are dependent on the training data: while the centering points $\bar{\boldsymbol{x}}_k$, $k = 1, \ldots, K$ in the radial basis functions can be chosen arbitrarily, they are typically chosen to be equal to the training points. The domain of the Legendre polynomials, on the other hand, is nominally $[-1, 1]$; therefore the independent variables must be appropriately scaled to ensure the range of training and prediction points $(\mathbf{x}, \mathbf{x}^*)$ is included in this interval.

The RVM method also employs a Bayesian approach to construct the model from training data. In particular, the vector of hyperparameters $\boldsymbol{\beta}$ affects the variance of the Gaussian random variables $\mathbf{w}$. If these hyperparameters are computed by a maximum-likelihood or a similar optimization algorithm, large values for these hyperparameters identify insignificant components that can be removed. Therefore, in the ROMES context, the RVM can be used

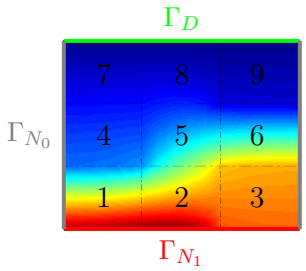


Figure 3. *Domain and sample solution $\boldsymbol{u}(\boldsymbol{\mu})$ for the thermal-block problem.*

to filter out the least significant error indicators. Apart from this detail, the RVM can be considered a special case of the GP kernel method with kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{K} \frac{1}{\beta_k} \phi_k(\boldsymbol{x}_i)\phi_k(\boldsymbol{x}_j). \tag{4.12}$$

5. Numerical experiments. This section analyzes the performance of the ROMES method on Poisson's equation with nine system inputs, using the reduced-basis method to generate the reduced-order model. First, Section 5.1 introduces the test problem. Section 5.2 discusses implementation and validation of the ROMES models. Section 5.3 compares the ROMES method to the multifidelity-correction approach characterized by employing the model inputs as error indicators. Section 5.4 compares the ROMES stochastic error estimate to the error bound given by the reduced-basis method. Section 5.5 compares the performance of the two machine-learning algorithms: the Gaussian process kernel method and the relevance vector machine. Finally, Section 5.6 considers non-compliant and multiple output functionals, which ROMES handles via dual-weighted-residual error indicators.

5.1. Problem setup. Consider a finite-element model of heat transport on a square domain $\Omega := \cup_{i=1}^{9} \Omega_i$ composed of nine parameterized materials. The block is cooled along the top boundary to a reference temperature of zero, a nonzero heat flux is specified on the bottom boundary, and the leftmost boundary is adiabatic. The compliant output functional for this problem is defined as the integral over the Neumann domain Γ_{N_1}

$$\bar{g}(u(\boldsymbol{\mu})) = \int_{\Gamma_{N_1}} u(\boldsymbol{\mu}) \mathrm{d}x, \quad \boldsymbol{\mu} \in \mathcal{P}, \tag{5.1}$$

where the parameter domain is set to $\mathcal{P} = [0.1, 10]^9$ and u is the continuous representation of the finite-element solution. The state variable $u(\boldsymbol{\mu}) \in X = H_0^1 := \left\{ w \in H^1(\Omega) \mid w|_{\Gamma_D} = 0 \right\}$ fulfills the weak form of the parameterized Poisson's equation: find $u(\boldsymbol{\mu}) \in X$, such that

$$a(u(\boldsymbol{\mu}), v) = f(v) \quad \text{for all } v \in X. \tag{5.2}$$

Here, the bilinear form $a : X \times X \to X$ and the functional $f : X \to X$ are defined as

$$a(u, v) := \int_{\Omega} b(x; \boldsymbol{\mu}) \nabla u(\boldsymbol{\mu}) \cdot \nabla v \, \mathrm{d}x, \qquad f(v) := \int_{\Gamma_{N_1}} v \, \mathrm{d}x \tag{5.3}$$

with boundary conditions

$$\nabla b(x;\boldsymbol{\mu})u(\boldsymbol{\mu})\cdot n = 0 \quad \text{on } \Gamma_{N_0}, \qquad \nabla b(x;\boldsymbol{\mu})u(\boldsymbol{\mu})\cdot n = 1 \quad \text{on } \Gamma_{N_1}. \tag{5.4}$$

We define the coefficient function $b : \Omega \times \mathcal{P} \to \mathbb{R}$ as

$$b(x;\boldsymbol{\mu}) = \sum_{i=1}^{9} \mu_i \mathbb{1}_{\Omega_i}(x)\,, \tag{5.5}$$

where μ_i denotes the ith component of the parameter vector $\boldsymbol{\mu}$, and the indicator function $\mathbb{1}_A(x) = 1$ if $x \in A$ and is zero otherwise. Figure 3 depicts the composition of the domain and the location of the boundary conditions.

By replacing the infinite-dimensional function space X with the (finite) n-dimensional finite-element space $X_\mathrm{h} \subset X$ in problem (5.2), one can compute the parameter-dependent state function $u_\mathrm{h}(\boldsymbol{\mu}) \in X_\mathrm{h}$ represented by vectors containing the function's degrees of freedom $\boldsymbol{u}(\boldsymbol{\mu}) \in \mathbb{R}^n$ (see Section S1.1). In the experiments, the domain is discretized by triangular finite elements, which results in a finite-element space X_h of dimension $n = 10^4$. The high-fidelity output (in the notation of Section 2.1) is then $g(\boldsymbol{u}(\boldsymbol{\mu})) := \bar{g}(u_\mathrm{h}(\boldsymbol{\mu}))$, $\boldsymbol{\mu} \in \mathcal{P}$.

As described in Section S1.4, we employ a greedy algorithm[4] to generate a reduced-basis space $X_\mathrm{red} \subset X_\mathrm{h}$ of dimension $p \ll n$. The algorithm employs a training set of 100 randomly selected points (i.e., $\mathrm{card}\,(\mathcal{P}_\mathrm{greedy}) = 100$ in Section S1.4), until the maximum computed error bound in the training set is less than 1; it stops after $p = 11$ iterations.

Replacing X_h with X_red in Eq. (5.2) leads to reduced state functions $u_\mathrm{red}(\boldsymbol{\mu}) \in X_\mathrm{red}$ for all $\boldsymbol{\mu} \in \mathcal{P}$. As before, these solutions can be represented by vectors $\boldsymbol{u}_\mathrm{red}(\boldsymbol{\mu}) := \boldsymbol{V}\hat{\boldsymbol{u}}(\boldsymbol{\mu}) \in \mathbb{R}^n$, where $\boldsymbol{V} \in \mathbb{R}^{n\times p}$ is the discrete representation of a basis for the function space X_red.

In the following, we analyze two types of error: (i) the energy norm of the state-space error $|||\boldsymbol{\delta}_{\boldsymbol{u}}||| = |||\boldsymbol{u} - \boldsymbol{u}_\mathrm{red}||| := a(u_\mathrm{h} - u_\mathrm{red}, u_\mathrm{h} - u_\mathrm{red})$ and (ii) the output error $\delta_s = g(u_\mathrm{h}) - g(u_\mathrm{red})$. Because the output functional in this case is compliant (i.e., $g = f$ and a is symmetric), the output error is always non-negative; see Eq. (S1.20) of Section S1.3. For more details regarding the finite-element discretization, the reduced-order-model generation, and error bounds, consult Section S1 of the Supplementary Materials.

5.2. ROMES implementation and validation. We first compute ROMES surrogates for the two errors $|||\boldsymbol{\delta}_{\boldsymbol{u}}|||$ and (compliant) δ_s. As proposed in Section 3.2.1, the three ROMES ingredients we employ are: 1) log-residual-norm error indicators $\boldsymbol{\rho}(\boldsymbol{\mu}) = \log(r(\boldsymbol{\mu}))$, 2) a logarithmic transformation function $d = \log$, and 3) both the GP kernel and the RVM supervised machine-learning methods. To train the surrogates, we compute $|||\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})|||$, $\delta_s(\boldsymbol{\mu})$, and $\boldsymbol{\rho}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \bar{\mathcal{P}} \subset \mathcal{P}$ with $\mathrm{card}\left(\bar{\mathcal{P}}\right) = 2000$. The first $N = 100$ points comprise the ROMES training set $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N\} =: \mathcal{P}_\mathrm{learn} \subset \bar{\mathcal{P}}$ and the following 1900 points define a validation set $\mathcal{P}_\mathrm{validation} \subset \bar{\mathcal{P}}$; note that the validation set was not used to construct the error surrogates. Reported results relate to statistics computed over this validation set.

For the kernel method, we employ the squared exponential covariance kernel (4.3). For the RVM method, we choose Legendre polynomials P_k as basis functions, as we expect a

[4] All reduced-basis computations are conducted with the reduced-basis library RBMatlab (http://www.morepas.org/software/).

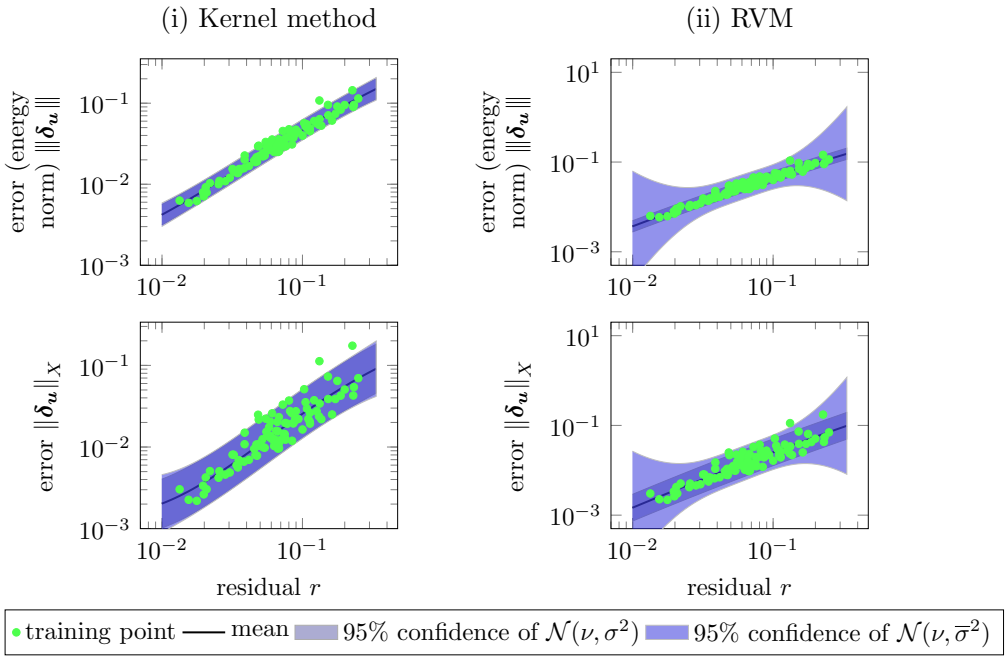


Figure 4. *Visualization of ROMES surrogates ($\delta = |||\boldsymbol{\delta_u}|||$ and $\|\boldsymbol{\delta_u}\|_X$, $\boldsymbol{\rho} = \log r$, $d = \log$), computed using $N = 100$ training points and the (i) GP kernel method and (ii) RVM.*

linear relationship between the indicators and true errors (see Section 4.2). Because Legendre polynomials are defined on the interval $[-1, 1]$, we must transform and scale this domain to span the possible range of indicator values. For this purpose, we apply the heuristic of setting the domain of the polynomials to be 20% larger than the interval bounded by the smallest and largest indicator values:

$$[\boldsymbol{\rho}_{\min} - 0.1(\boldsymbol{\rho}_{\max} - \boldsymbol{\rho}_{\min}), \boldsymbol{\rho}_{\max} + 0.1(\boldsymbol{\rho}_{\max} - \boldsymbol{\rho}_{\min})]\,, \tag{5.6}$$

where $\boldsymbol{\rho}_{\min} = \min_{\boldsymbol{\mu} \in \mathcal{P}_{\text{learn}}} \boldsymbol{\rho}(\boldsymbol{\mu})$ and $\boldsymbol{\rho}_{\max} = \max_{\boldsymbol{\mu} \in \mathcal{P}_{\text{learn}}} \boldsymbol{\rho}(\boldsymbol{\mu})$. We include Legendre polynomials of orders 0 to 4; however, the RVM method typically discards the higher order polynomials due to the near-linear relation between indicators and errors.

Figure 4 depicts the ROMES surrogate $\widetilde{|||\boldsymbol{\delta_u}|||}$ generated by both machine-learning methods using all 100 training points. For comparison, we also create ROMES surrogates $\widetilde{\|\boldsymbol{\delta_u}\|_X}$ for errors in the parameter-independent norm $\|\cdot\|_X$ of the state space $X = H_0^1$. In addition to the expected mean of the inferred surrogate, the figure displays two 95%-confidence intervals for the prediction (see Remark 4.1):

(i) The darker shaded interval corresponds to the confidence interval arising from the inherent uncertainty in the error due to the non-uniqueness of the mapping $\boldsymbol{\rho} \mapsto |||\boldsymbol{\delta_u}|||$, i.e., the inferred variance σ^2 of Eq. (4.5).

(ii) The lighter shaded interval also includes the 'uncertainty in the mean' due to a lack of training data, i.e., Σ of Eq. (4.5). With an increasing number of training points, this area

should be indistinguishable from the darker one.

All ROMES models find a linear trend between the indicators and the errors, where the variance is slightly larger for the parameter-independent norm. This larger variance can be attributed to the larger range of the coercivity constants the parameter-independent norm (see Section S1.3). For this example, however, both ROMES are functional. In the following examples, we focus on the energy norm only.

Note that the 'uncertainty in the mean' is dominant for the RVM surrogate. This can be explained as follows: the high-order polynomials have values close to zero near the mean of the data. As such, the training data are not very informative for the coefficients of these polynomials. This results in a large inferred variance for those coefficients. Section 5.5 further compares the two machine-learning methods; due to its superior performance, we now proceed with the kernel method.

We now assess the validity of the Gaussian-process assumptions underlying the ROMES surrogates $\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}$ and $\widetilde{\delta}_s$, i.e., Condition 3 of Section 3.1. From the discussion in Remark 4.1, we know if the underlying GP model form is correct, then as the number of training points increases, the uncertainty about the mean decreases and the set $\{D(\boldsymbol{\mu}) \mid \boldsymbol{\mu} \in \mathcal{P}_{\text{validation}}\}$ with

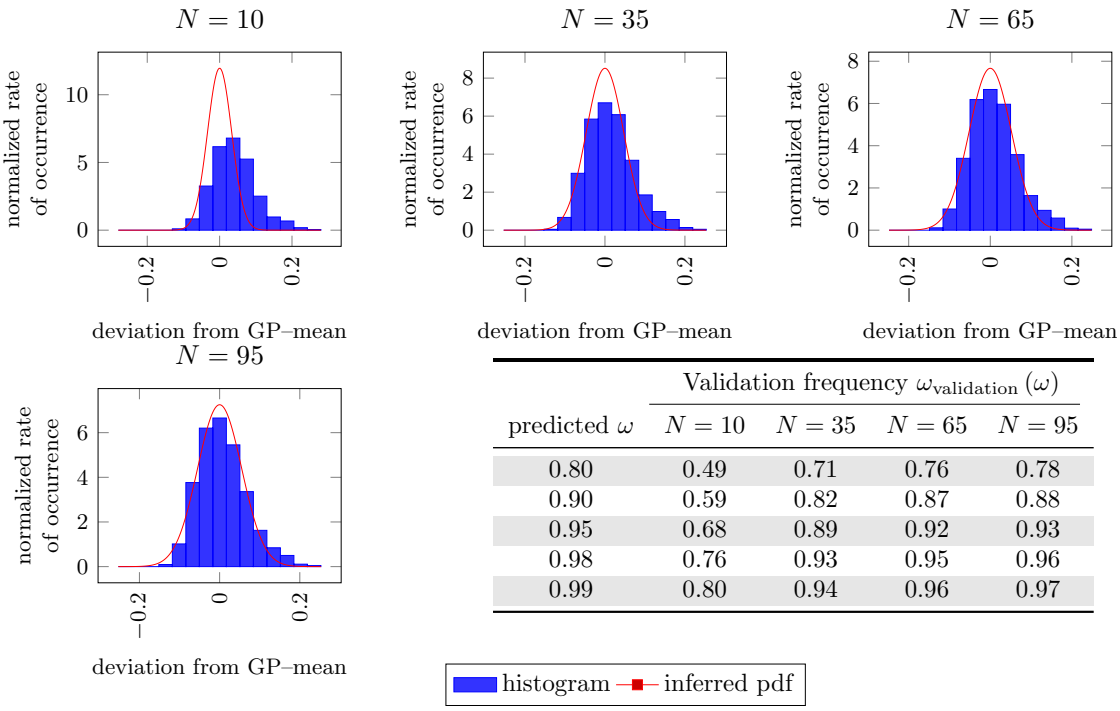
$$D(\boldsymbol{\mu}) := d\left(|||\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})|||\right) - \mathrm{E}\left[d\left(\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}(\boldsymbol{\rho}(\boldsymbol{\mu}))\right)\right] = d\left(|||\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})|||\right) - \nu\left(\boldsymbol{\rho}(\boldsymbol{\mu})\right) \tag{5.7}$$

should behave like samples from the distribution $\mathcal{N}(0, \sigma^2)$. Figure 5 reports this validation test and verifies that this condition does indeed hold for a sufficiently large number of training points.

Further, we can validate the inferred confidence intervals as proposed in Eq. (3.1). The table within Figure 5 reports $\omega_{\text{validation}}(\omega)$ (see Eq. (3.2)), which represents the frequency of observed predictions in the validation set that lie within the inferred confidence interval C_ω. We declare the ROMES model to be validated, as $\omega_{\text{validation}}(\omega) \approx \omega$ for several values of ω as the number of training points increases.

The results for the ROMES surrogate $\widetilde{\delta}_s$ are very similar to those presented in Figure 5 and will be further discussed in Section 5.3. Note that the inferred Gaussian process is well-converged with a moderately sized training set consisting of only $N = 35$ points.

5.3. Output error: comparison with multifidelity correction. As discussed in Section 3.2, multifidelity-correction methods construct a surrogate $\widetilde{\delta_{s,\text{MF}}}$ of the output error using the system inputs as error-surrogate inputs, i.e., $\delta = \delta_s$, $\boldsymbol{\rho} = \boldsymbol{\mu}$, and $d = \mathrm{id}_{\mathbb{R}}$. In this section, we construct this multifidelity correction surrogate using the same GP kernel method as ROMES. Ref. [29] demonstrated that this error surrogate fails to improve the 'corrected output' when the low-fidelity model corresponds to a reduced-order model. We now verify this result and show that—in contrast to the multifidelity correction approach—the ROMES surrogate $\widetilde{\delta}_s$ constructed via the GP kernel method with $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, and $d = \log$ yields impressive results: on average, the output 'corrected' by the ROMES surrogate reduces the error by an order of magnitude, and the Gaussian-process assumptions can be validated. The validation quality improves as the number of training points increases, but a moderately sized set of only $N = 20$ training points leads to a converged surrogate.



predicted ω	Validation frequency $\omega_{\text{validation}}$ (ω)			
	$N = 10$	$N = 35$	$N = 65$	$N = 95$
0.80	0.49	0.71	0.76	0.78
0.90	0.59	0.82	0.87	0.88
0.95	0.68	0.89	0.92	0.93
0.98	0.76	0.93	0.95	0.96
0.99	0.80	0.94	0.96	0.97

Figure 5. *Gaussian-process validation for the ROMES surrogate (GP kernel, $\delta = \|\|\boldsymbol{\delta}_{\boldsymbol{u}}\|\|$, $\boldsymbol{\rho} = \log r$, $d = \log$) with a varying number of training points N. The histogram corresponds to samples of $D(\boldsymbol{\mu})$ and the red curve depicts the probability distribution function $\mathcal{N}(0, \sigma^2)$. The table reports how often the actual error lies in the inferred confidence intervals.*

The reason multifidelity correction fails for most reduced-order models is twofold. First, the mapping $\boldsymbol{\mu} \mapsto \delta_s$ can be highly oscillatory in the input space. This behavior arises from the fact the the reduced-order model error is zero at the (greedily-chosen) ROM training points but grows (and can grow quickly) away from these points. Such complex behavior requires a large number of error-surrogate training points to accurately capture. In addition, the number of system inputs is often large (in this case nine); this introduces curse-of-dimensionality difficulties in modeling the error. Figure 6(ii) visualizes this problem. The depicted mapping between the first two parameter components μ_1, μ_2 and the output error $\delta_s(\boldsymbol{\mu})$ displays no structured behavior. As a result, there is no measurable improvement of the corrected output $s_{\text{red}} + \widetilde{\delta_{s,\text{MF}}}$ over the evaluation of the ROM output s_{red} alone.

In order to quantify the performance of the error surrogates, we introduce a normalized *expected improvement*

$$I(\widetilde{\delta}, \boldsymbol{\mu}) := \left| \frac{\delta_s(\boldsymbol{\mu}) - \text{mode}\left(\widetilde{\delta}(\boldsymbol{\rho}(\boldsymbol{\mu}))\right)}{\delta_s(\boldsymbol{\mu})} \right|. \tag{5.8}$$

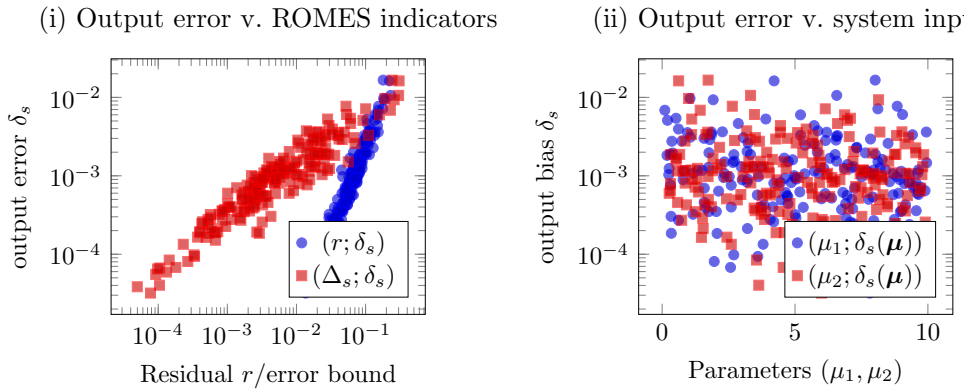


Figure 6. *Relationship between (i) ROMES error indicators and the compliant-output error and (ii) the first two parameter components and the (compliant) output error, visualized by evaluation of 200 random sample points in the input space. Clearly, the observed structure in the former relationship is more amenable to constructing a Gaussian process.*

If this value is less than one, then the exected corrected output $s_{\text{red}} + \widetilde{\delta}$ is more accurate than the ROM output s_{red} itself for point $\boldsymbol{\mu} \in \mathcal{P}$, i.e., the additive error surrogate *improves* the prediction of the ROM. On the other hand, values above one indicate that the error surrogate *worsens* the ROM prediction.

Figure 7 reports the mean, median, standard deviations, and extrema for the expected improvement (5.8) evaluated for all validation points $\mathcal{P}_{\text{validation}}$ and a varying number of training points. Here, we also compare with the performance of the error surrogate $\widetilde{\delta_{\text{uni}}}$, which is defined as a uniform distribution on the interval $\left[\Delta_s^{\text{LB}}(\boldsymbol{\mu}), \Delta_s(\boldsymbol{\mu})\right]$, where $\Delta_s^{\text{LB}}(\boldsymbol{\mu})$ and $\Delta_s(\boldsymbol{\mu})$ are the the lower and upper bounds for the output error, respectively. Note that $\widetilde{\delta_{\text{uni}}}$ does not require training data, as it is based purely on error bounds.

The expected improvement for the ROMES output-error surrogate $I(\widetilde{\delta}_s, \boldsymbol{\mu})$ as depicted in Figure 7(i) is approximately 0.2 on average, which constitutes an improvement of nearly an order of magnitude. Further, the maximum expected improvement almost always remains below 1; this implies that the corrected ROM output is almost always more accurate than the ROM output alone.

On the other hand, the expected improvement generated by the error surrogate $\widetilde{\delta_{\text{uni}}}$ is always greater than one, which means that its correction always increases the error. This arises from the fact that the center of the interval $\left[\Delta_s^{\text{LB}}(\boldsymbol{\mu}), \Delta_s(\boldsymbol{\mu})\right]$ is a poor approximation for the true error.

In addition, Figure 7(ii) shows that the expected improvement produced by the multifidelity-correction surrogate $I\left(\widetilde{\delta_{s,\text{MF}}}, \boldsymbol{\mu}\right)$ is often far greater than one. This shows that the multifidelity-correction approach is not well suited for this problem. Presumably, with (far) more training points, these results would improve.

Again, we can validate the Gaussian-process assumptions underlying the error surrogates. For $N = 100$ training points, Figure 8 compares a histogram of deviation of the true error from the surrogate mean to the inferred probability density function. The associated table

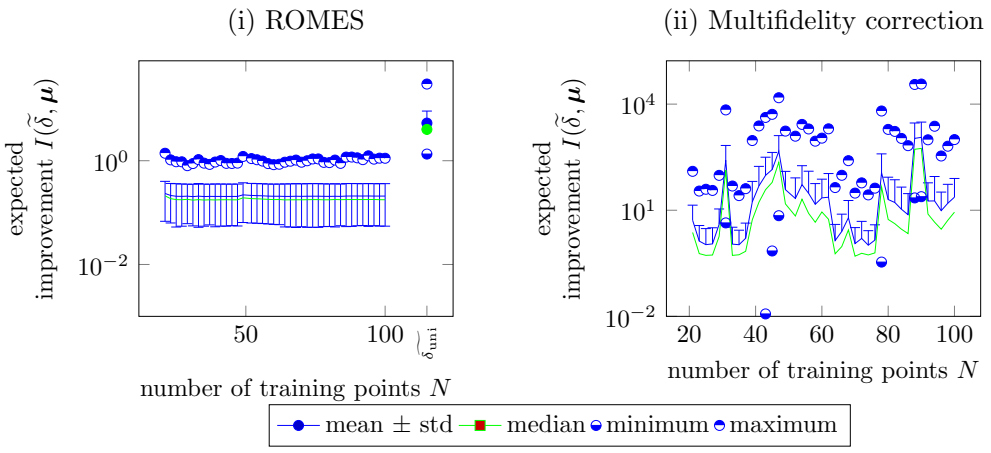
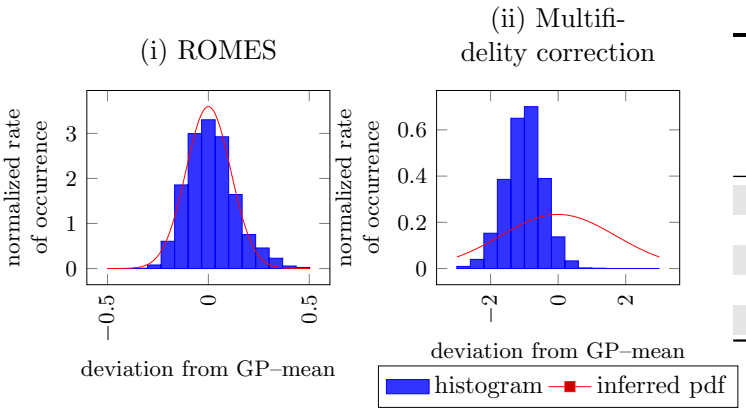


Figure 7. *Expected improvement $I(\widetilde{\delta}, \boldsymbol{\mu})$ for a varying number of training points N: (i) ROMES (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, $d = \log$) with uniform distribution based on reduced-basis error bounds $\widetilde{\delta_{uni}}$ and (ii) multifidelity correction (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \boldsymbol{\mu}$, and $d = \mathrm{id}_{\mathbb{R}}$). (1: no improvement, > 1: error worsened, < 1: error improved).*



predicted ω	Validation frequency $\omega_{\text{validation}}(\omega)$ (i)	(ii)
0.80	0.78	0.98
0.90	0.89	1.00
0.95	0.93	1.00
0.98	0.96	1.00
0.99	0.97	1.00

Figure 8. *Gaussian-process validation for the ROMES surrogate (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, $d = \log$) and multifidelity-correction surrogate (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \boldsymbol{\mu}$, and $d = \mathrm{id}_{\mathbb{R}}$) using $N = 100$ training points The histogram corresponds to samples of $D(\boldsymbol{\mu})$ and the red curve depicts the probability distribution function $\mathcal{N}(0, \sigma^2)$. The table reports how often the actual error lies in the inferred confidence intervals. Clearly, this validation test fails for the multilfidelity-correction surrogate.*

reports how often the validation data lie in inferred confidence intervals. We observe that the confidence intervals are valid for the ROMES surrogate, but are not for the multifidelity-correction surrogate, as the bins do not align with the inferred distribution. Figure 9 depicts the convergence of these confidence-interval validation metrics as the number of training points increases. As expected (see Remark 4.1) the ROMES observed confidence intervals more closely align with the confidence intervals arising from the *inherent uncertainty* (i.e., σ^2)

as the number of training points increases, as this effectively decreases the *uncertainty due to a lack of training*. In addition, only a moderate number of training points (around 20) is required to generate a reasonably converged ROMES surrogate. On the other hand the multifidelity-correction surrogate exhibits no such convergence when fewer than 100 training points are used.

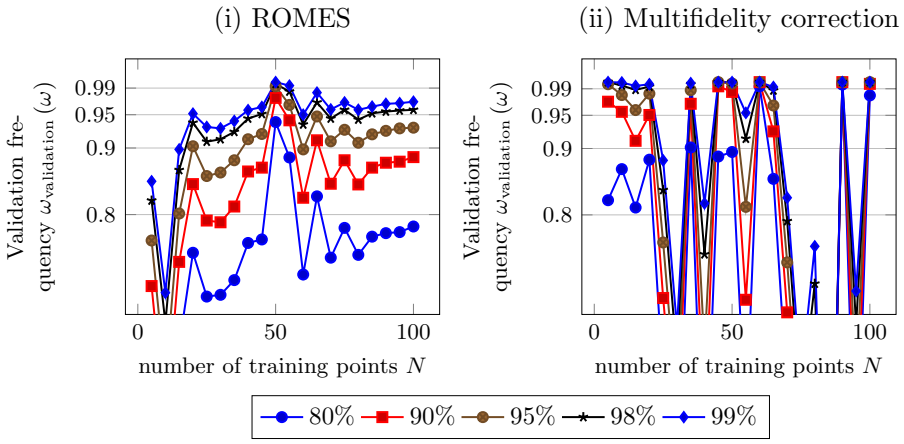


Figure 9. *Gaussian-process validation for the ROMES surrogate (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, $d = \log$) and multifidelity-correction surrogate (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \boldsymbol{\mu}$, and $d = \mathrm{id}_{\mathbb{R}}$) and a varying number of training points N. The plots depict how often the actual error lies in the inferred confidence intervals.*

5.4. Reduced-basis error bounds. In this section, we compare the reduced-basis error bound $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$ (S1.14) with the *probabilistically rigorous* ROMES surrogates $\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}^{c}$ (2.7) with rigor values of $c = 0.5$ and $c = 0.9$ as introduced Section 3.3.[5] The ROMES surrogate is constructed with the GP kernel method and ingredients $\delta = |||\boldsymbol{\delta}_{\boldsymbol{u}}|||$, $\boldsymbol{\rho} = \log r$, and $d = \log$. As discussed in Section 2.3 the error-bound effectivity (2.7) is important to quantify the performance of these bounds; a value of 1 is optimal, as it implies no over-estimation.

As the probabilistically rigorous ROMES surrogates $\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}^{c}$ are stochastic processes, we can measure their (most common) effectivity as

$$\eta(c, \boldsymbol{\mu}) := \frac{\mathrm{mode}\left(\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}^{c}(\boldsymbol{\rho}(\boldsymbol{\mu}))\right)}{|||\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})|||}. \tag{5.9}$$

The top plots of Figure 10 report the mean, median, standard deviation, and extrema of the effectivities $\eta(0.5, \boldsymbol{\mu})$ and $\eta(0.9, \boldsymbol{\mu})$ for all validation points $\boldsymbol{\mu} \in \mathcal{P}_{\text{validation}}$. Again, we compare with $\widetilde{\delta_{\text{uni}}}$, which is a uniform distribution on an interval whose endpoints correspond to the lower and upper bounds for the error $|||\boldsymbol{\delta}_{\boldsymbol{u}}(\boldsymbol{\mu})|||$. We also compare with the corresponding

[5] Note that $c = 0.5$ implies no modification to the original ROMES surrogate, as $\widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}^{0.5} = \widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}$ (see Eqs. (3.17)–(3.19)).

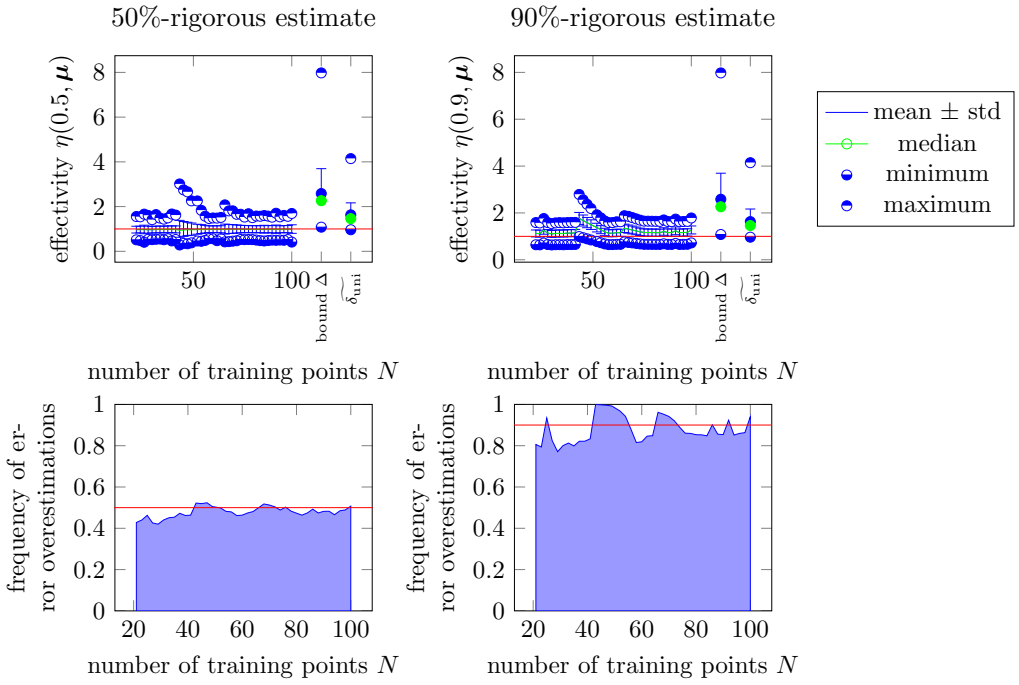


Figure 10. *Validation of the probabilistically rigorous ROMES surrogates* $\widetilde{\|\|\boldsymbol{\delta_u}\|\|}^c$ *(GP kernel,* $\delta = \|\|\boldsymbol{\delta_u}\|\|$*,* $\boldsymbol{\rho} = \log r$*,* $d = \log$*) and comparison with RB error upper bound* $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$ *and uniform distribution based on reduced-basis error bounds* $\widetilde{\delta_{uni}}$*. The top plots compare statistics of the effectivities* $\eta(c, \boldsymbol{\mu})$ *with* $c = 0.5$ *and* $c = 0.9$ *of the probabilistically rigorous ROMES surrogates with the RB error-bound surrogates. The bottom plots compare the frequency of error overestimation* $c_{validation}$ *with the desired value* c *(red line).*

statistics for the effectivity of the RB error bound $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$. The lower bound for the coercivity constant that is needed in the RB error bound $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$ is chosen as the minimum over all parameter components $\alpha_{\text{LB}}(\boldsymbol{\mu}) = \min_{i \in \{1,\dots,9\}} \mu_i$. This simple choice is effective because the example is affinely parameter dependent and linear [31, Ch. 4.2].

We observe that the ROMES surrogate yields better results than both the error bound $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$ (which produces effectivities roughly between one and eight) and the uniform distribution $\widetilde{\delta_{\text{uni}}}$ (which produces mode effectivities roughly between one and four). The 50%-rigorous ROMES surrogate has an almost perfect mean effectivity of 1 as desired. The 90%-rigorous surrogate has a higher mean effectivity as expected; however, it is only slightly higher. Furthermore, the effectivities of the ROMES surrogate exhibit a much smaller variance[6] than both $\Delta_{\boldsymbol{u}}^{\boldsymbol{\mu}}$ and $\widetilde{\delta_{\text{uni}}}$. Finally, a moderate number (around 20) of training samples is sufficient to obtain well-converged surrogates.

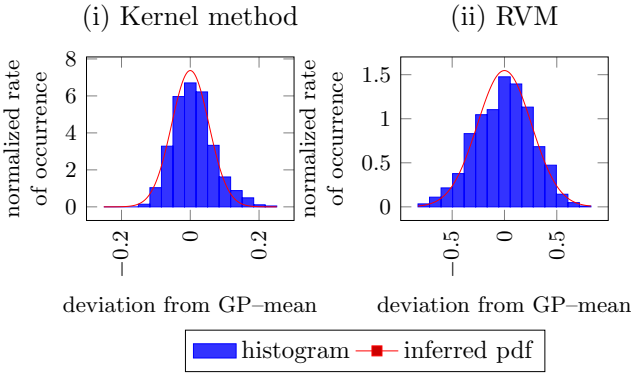
[6]The higher variance apparent between 45 and 53 training points can be explained by the fact that the minimization algorithm for the log–likelihood function stops after it exceeds the maximum number of iterations.

The bottom plots of Figure 10 report the frequency of error overestimation

$$c_{\text{validation}} := \frac{\text{card}\left(\{\boldsymbol{\mu} \in \mathcal{P}_{\text{validation}} \mid \text{median}\left(\tilde{m}\left(\boldsymbol{\rho}(\boldsymbol{\mu})\right)\right) > d(\delta(\boldsymbol{\mu}))\}\right)}{\text{card}\left(\mathcal{P}_{\text{validation}}\right)} \tag{5.10}$$

for the probabilistically rigorous ROMES surrogates (i.e., $\tilde{m} = \widetilde{|||\boldsymbol{\delta}_{\boldsymbol{u}}|||}^{c}$) as the number of training points increases to show that $c_{\text{validation}} \approx c$, which validates the rate of error overestimation (see Eq. (3.15)). Note that the overestimation frequency $c_{\text{validation}}$ converges to its predicted value c, which demonstrates that the rigor of the ROMES estimates can in fact be controlled.

5.5. Comparison of machine-learning algorithms. This section compares in detail the ROMES surrogates generated using the two machine-learning methods introduced in Section 4. Recall that Figure 4 visualizes both surrogates. We observe that both methods work well overall and generate well-converged surrogates with a modest number of training samples. As previously mentioned, the GP kernel leads to a smaller inferred variance due to more accurate and localized estimates of the mean. The RVM is characterized by global basis functions that preclude it from accurately resolving localized features of the mean and lead to large uncertainty about the mean (see Figure 4). On the other hand, the confidence intervals computed with the RVM are (slightly) better validated.



	Validation frequency $\omega_{\text{validation}}(\omega)$			
	σ^2		$\overline{\sigma}^2$	
predicted ω	(i)	(ii)	(i)	(ii)
0.80	0.78	0.78	0.84	0.93
0.90	0.88	0.89	0.93	0.98
0.95	0.93	0.95	0.95	1.00
0.98	0.96	0.97	0.97	1.00
0.99	0.97	0.99	0.99	1.00

Figure 11. *Gaussian-process validation for the ROMES surrogate using both the GP kernel method and the RVM ($\delta = |||\boldsymbol{\delta}_{\boldsymbol{u}}|||$, $\boldsymbol{\rho} = \log r$, $d = \log$) using $N = 80$ training points. The histogram corresponds to samples of $D(\boldsymbol{\mu})$ and the red curve depicts the probability distribution function $\mathcal{N}(0, \sigma^2)$. The table reports how often the actual error lies in the inferred confidence intervals.*

Figure 11(i) and (ii) report the validation test, i.e., the frequency of deviations from the inferred mean $D(\boldsymbol{\mu})$ (5.7) with a training sample containing $N = 80$ training points. We observe a smaller inferred variance σ^2 for the GP kernel method, which implies that the mean is identified more accurately. In both cases, the validation samples align well with the probability density function of the inferred distribution $\mathcal{N}(0, \sigma^2)$.

The confidence intervals of this inferred distribution can be validated, and they turn out to be (slightly) more realistic for the RVM method. The table within Figure 11 shows that the

kernel method results are usually optimistic, i.e., the actual confidence intervals are smaller than predicted. This effect can be corrected, however, by adding $\Sigma(\boldsymbol{x}^*)$ as an indicator of the uncertainty of the mean as discussed in Remark 4.1. However, doing so for the RVM prediction yields extremely wide confidence intervals due to the significant uncertainty about the RVM mean (see Figure 4).

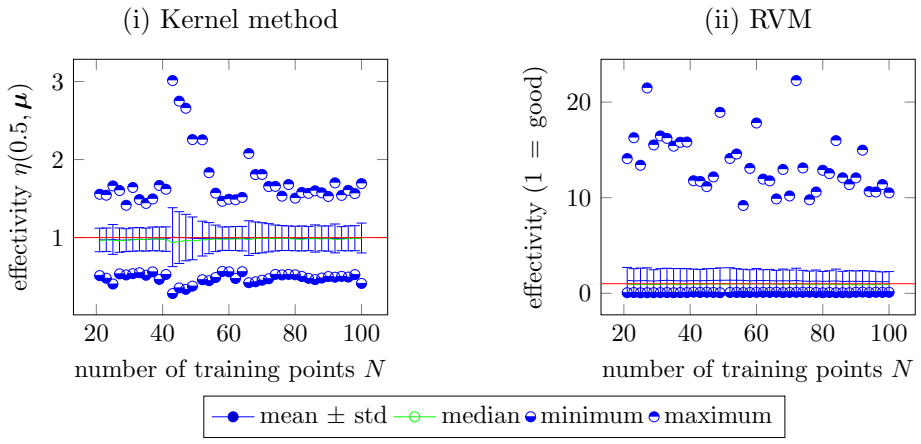


Figure 12. *Comparison of the effectivity $\eta(0.5, \boldsymbol{\mu})$ of ROMES surrogates ($\delta = |||\boldsymbol{\delta_u}|||$, $\boldsymbol{\rho} = \log r$, $d = \log$) with the GP computed via (i) the GP kernel and (ii) the relevance vector machine method.*

As the inferred variance is larger for the relevance vector machine, this also affects the performance of effectivity and improvement measures for the error surrogates. Figure 12 depicts statistics of $\eta(0.5, \boldsymbol{\mu})$ computed with both the methods, and we observe that all statistical measures are significantly better for the kernel method estimate.

We conclude that while both methods produce feasible ROMES surrogates, the GP kernel method produces consistently better results. In particular, the lower inferred variance implies that a lower amount of *epistemic uncertainty* is introduced by the error surrogate (See condition 2 from Section 3.1). This is critically important for many UQ tasks. Therefore, we recommend the GP kernel method to construct ROMES surrogates.

5.5.1. Dependence on reduced-basis size. To assess the generalizability of the ROMES method, we apply it to a ROM of higher fidelity, i.e., larger p. We construct two ROMES surrogates: one for the state-space error $\widetilde{|||\boldsymbol{\delta_u}|||}$ (GP kernel, $\delta = |||\boldsymbol{\delta_u}|||$, $\boldsymbol{\rho} = \log r$, $d = \log$) and one for the compliant-output error $\widetilde{\delta_s}$ (GP kernel, $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, $d = \log$).

We increase the reduced-basis size by decreasing the maximum error over the training set from 1.0 to 1.0×10^{-3} and applying the greedy method. The resulting reduced-basis dimension is $p = 62$. Figure 13 reports the error data and ROMES surrogates. Comparing the leftmost plot of Figure 13 with Figure 4(i) and the rightmost plot of Figure 13 with Figure 6(i) reveals that while the errors are several orders of magnitude smaller for the current experiments, the data (and the resulting ROMES surrogates) exhibit roughly the same structure as before.

Figure 14 reports the performance of these surrogates. Comparing the leftmost plot of Figure 14 with Figure 12(i) shows that the state-space error surrogate exhibits nearly identical

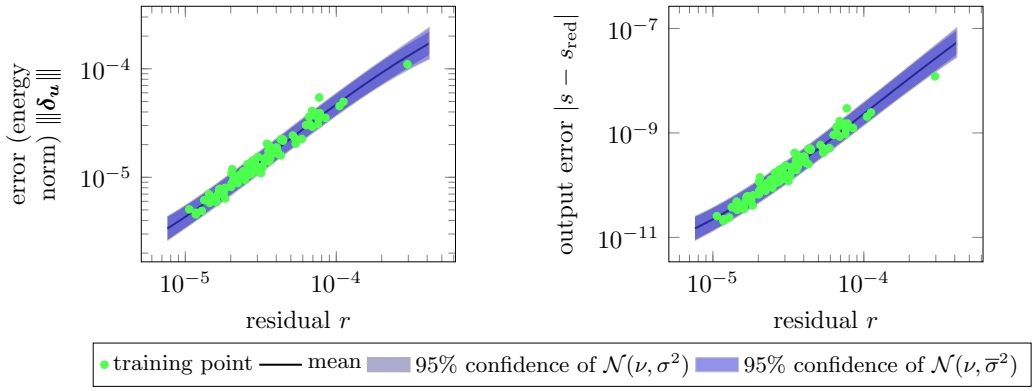


Figure 13. *Visualization of ROMES surrogates ($\delta = |||\boldsymbol{\delta}_{\boldsymbol{u}}|||$ and δ_s, $\boldsymbol{\rho} = \log r$, $d = \log$), computed using $N = 100$ training points and the GP kernel method for a higher dimensional ROM with $p = 62$.*

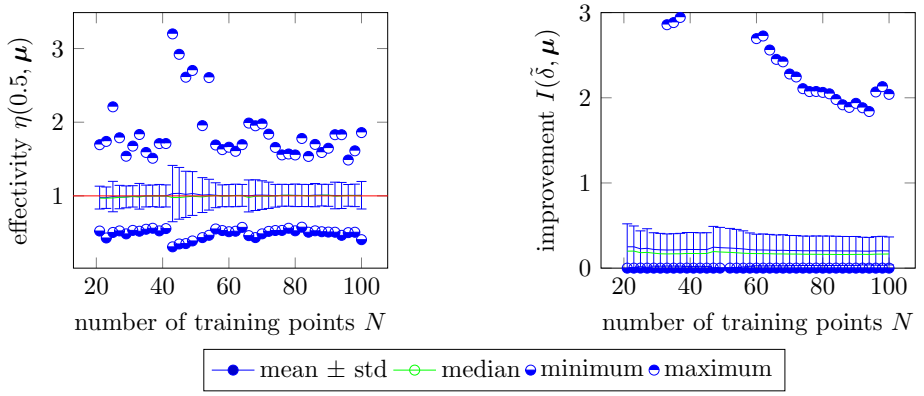


Figure 14. *Effectivity $\eta(0.5, \boldsymbol{\mu})$ of ROMES surrogate ($\delta = |||\boldsymbol{\delta}_{\boldsymbol{u}}|||$, $\boldsymbol{\rho} = \log r$, $d = \log$) and expected improvement $I(\tilde{\delta}, \boldsymbol{\mu})$ of ROMES surrogate (GP kernel, compliant $\delta = \delta_s$, $\boldsymbol{\rho} = \log r$, $d = \log$) for a higher dimensional ROM with $p = 62$ and a varying number of training points N.*

convergence for the larger- and smaller-dimension reduced-order models. As in the experiments of Section 5.4, the value of mode $(\eta(0.5, \boldsymbol{\mu}))$ is close to 1 in the mean. Comparing the rightmost plot of Figure 14 with Figure 7(i) shows that the expected improvement for the output-error correction with the surrogate $\widetilde{\delta}_s$ is around 0.2 in the mean for both the larger- and smaller-dimension reduced-order models. However, for the larger-dimension reduced-order model, more training points are required to reduce the occurrence of improvement factors larger than 1. This is an artifact of the low errors already produced by the larger-dimension ROM itself (i.e., small denominator in Eq. (5.8)).

We therefore conclude that the ROMES method is applicable to ROMs of different fidelity.

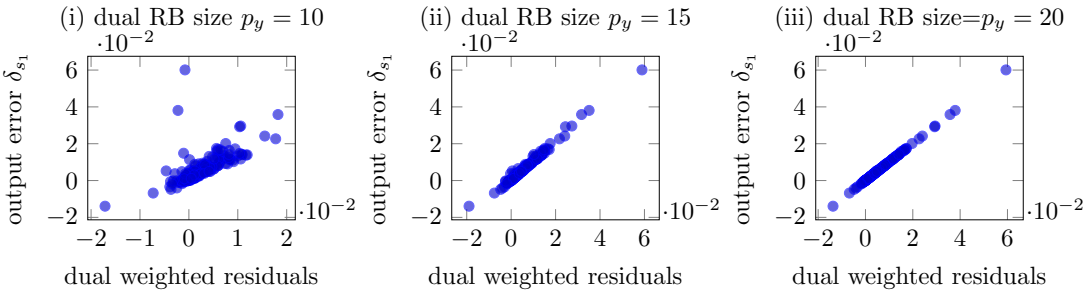


Figure 15. *Relationship between between dual-weighted-residual indicators* $\boldsymbol{\rho}_1 = \boldsymbol{y}_{red,1}(\boldsymbol{\mu})^t \boldsymbol{r}(\boldsymbol{u}_{red}; \boldsymbol{\mu})$ *and errors in the (non-compliant) first output* δ_{s_1}.

5.6. Multiple and non-compliant outputs. Finally, we assess the performance of ROMES on a model with multiple and non-compliant output functionals as discussed in Section 3.2.2. For this experiment, we set two outputs to be temperate measurements at points x_1 and x_2:

$$s_i(\boldsymbol{\mu}) := g_i(\boldsymbol{u}(\boldsymbol{\mu})) := \bar{g}_i(u(\boldsymbol{\mu})) = \int_\Omega \delta_{\text{Dirac}}(x - x_i) u(x_i; \boldsymbol{\mu}) \, \mathrm{d}x = u(x_i; \boldsymbol{\mu}), \quad i = 1, 2. \tag{5.11}$$

where δ_{Dirac} denotes the Dirac delta function. In this case, we construct a separate ROMES surrogates for each output error $\widetilde{\delta_{s_1}}$ and $\widetilde{\delta_{s_2}}$. As previously discussed, we use dual-weighted residuals as indicators $\boldsymbol{\rho}_i(\boldsymbol{\mu}) = \boldsymbol{y}_{\text{red},i}(\boldsymbol{\mu})^t \boldsymbol{r}(\boldsymbol{u}_{\text{red}}; \boldsymbol{\mu})$, $i = 1, 2$ and no transformation $d \equiv \text{id}_{\mathbb{R}}$. This necessitates the computation of approximate dual solutions, for which dual reduced-basis spaces must be generated in the offline stage. The corresponding finite element problem can be found in Eq. (S1.28), where Eq. (5.11) above provides the right-hand sides. The algebraic problems can be inferred from Eq. (S1.29), where the discrete right-hand sides are canonical unit vectors because the points x_1 and x_2 coincide with nodes of the finite-element mesh.

Like the primal reduced basis, the dual counterpart can be generated with a greedy algorithm that minimizes the approximation error for the reduced dual solutions.

To assess the ability for *uncertainty control* with the dual-weighted-residual indicators (see Remark 3.2) we generate three dual reduced bases of increasing fidelity: 1) error tolerance of 1 (basis sizes p_y of 10 and 11), 2) error tolerance of 0.5 (basis sizes p_y of 15 and 17), 3) error tolerance of 0.1 (basis sizes p_y of 20 and 23).

To train the surrogates, we compute $\delta_{s_1}(\boldsymbol{\mu})$, $\delta_{s_2}(\boldsymbol{\mu})$, $\boldsymbol{\rho}_1(\boldsymbol{\mu})$ (of varying fidelity), $\boldsymbol{\rho}_2(\boldsymbol{\mu})$ (of varying fidelity), for $\boldsymbol{\mu} \in \bar{\mathcal{P}} \subset \mathcal{P}$ with $\text{card}\left(\bar{\mathcal{P}}\right) = 500$. The first $T = 100$ points define the training set $\mathcal{P}_{\text{learn}} \subset \bar{\mathcal{P}}$ and the following 400 points constitute the validation set $\mathcal{P}_{\text{validation}} \subset \bar{\mathcal{P}}$.

Figure 15 depicts the observed relationship between indicators $\boldsymbol{\rho}_1(\boldsymbol{\mu})$ (of different fidelity) and the error in the first output $\delta_{s_1}(\boldsymbol{\mu})$. Note that as the dual-basis size p_y increases, the output error exhibits a nearly exact linear dependence on the dual-weighted residuals. This is expected, as the residual operator is linear in the state. Therefore, the RVM with a linear polynomial basis produces the best (i.e., minimum variance) results for the ROMES surrogates in this case.

Figure 16 reflects the necessity of employing a large enough dual reduced basis to compute the dual-weighted-residual error indicators. For a small dual reduced basis, there is almost no improvement in the mean, and only a slight improvement in the median; in some cases, the 'corrected' outputs are actually less accurate. However, the most accurate dual solutions yield a mean and median error improvement of two orders of magnitude. This illustrates the ability and utility of *uncertainty control* when dual-weighted residuals are used as error indicators.

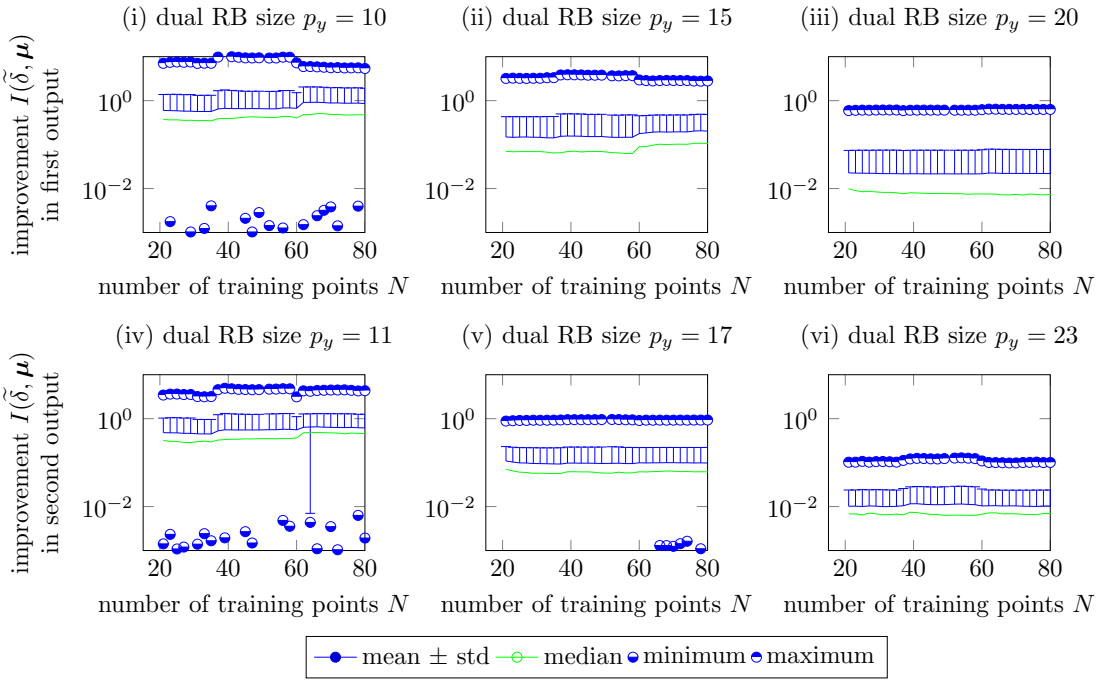


Figure 16. *Expected improvement $I(\widetilde{\delta}, \boldsymbol{\mu})$ for ROMES surrogate (RVM, $\delta = \delta_s$, $\boldsymbol{\rho}_i = \boldsymbol{y}_{red,i}(\boldsymbol{\mu})^t \boldsymbol{r}(\boldsymbol{u}_{red}; \boldsymbol{\mu})$, $i = 1, 2$, $d = id_{\mathbb{R}}$) for a varying number of training points T and different dual reduced-basis-space dimensions. Compare with Figure 7 (1: no improvement, > 1: error worsened, < 1: error improved).*

Table 17 reports validation results for the inferred confidence intervals. While the validation results are quite good (and appear to be converging to the correct values), they are not as accurate as those obtained for the compliant output.

6. Conclusions and outlook. This work presented the ROMES method for statistically modeling reduced-order-model errors. In contrast to rigorous error bounds, such statistical models are useful for tasks in uncertainty quantification. The method employs supervised machine learning methods to construct a mapping from existing, cheaply computable ROM error indicators to a *distribution* over the true error. This distribution reflects the epistemic uncertainty introduced by the ROM. We proposed ROMES ingredients (supervised-learning method, error indicators, and transformation function) that yield low-variance, numerically validated models for different types of ROM errors.

	Validation frequency $\omega_{\text{validation}}$ (ω)							
	first output				second output			
predicted ω	$N = 29$	$N = 53$	$N = 76$	$N = 100$	$N = 29$	$N = 53$	$N = 76$	$N = 100$
0.8	1.00	0.83	0.82	0.82	1.00	0.82	0.87	0.88
0.9	1.00	0.87	0.86	0.86	1.00	0.87	0.91	0.92
0.95	1.00	0.89	0.89	0.89	1.00	0.89	0.93	0.93
0.98	1.00	0.90	0.91	0.90	1.00	0.92	0.94	0.95
0.99	1.00	0.92	0.92	0.91	1.00	0.94	0.95	0.96

Figure 17. *Gaussian-process validation for the ROMES surrogates (RVM, $\delta = \delta_s$, $\boldsymbol{\rho}_i = \boldsymbol{y}_{red,i}(\boldsymbol{\mu})^t \boldsymbol{r}(\boldsymbol{u}_{red}; \boldsymbol{\mu})$, $i = 1, 2$, $d = id_{\mathbb{R}}$). The table reports how often the actual error lies in the inferred confidence intervals. The largest dual reduced-basis space dimensions ($p_y = 20$ and $p_y = 23$) are used to compute the error indicators. .*

For normed outputs, the ROMES surrogates led to effectivities with low variance and means close to the optimal value of one, as well as a notion of probabilistic rigor. This is in contrast to existing ROM error bounds, which exhibited mean effectivities close to ten; this improvement will likely be more pronounced for more complex (e.g., nonlinear, time dependent) problems. Further, the ROMES surrogates were computationally less expensive than the error bounds, as the coercivity-constant lower bound was not required.

For general outputs, the ROMES surrogate allowed the ROM output to be corrected, which yielded a near 10x accuracy improvement. Further, the uncertainty in this error could be *controlled* by modifying the dimension of the dual reduced basis. On the other hand, existing approaches (i.e., multifidelity correction) that employ system inputs (not error indicators) as inputs to the error model did not lead to improved output predictions. This demonstrated the ability of ROMES to mitigate the curse of dimensionality: although the problem was characterized by nine system inputs, only one error indicator was necessary to construct a low-variance, validated ROMES surrogate.

We foresee the combination of ROMs with ROMES error surrogates to be powerful in UQ applications, especially when the number of system inputs is large. Future work entails integrating and analyzing the ROM/ROMES combination for specific UQ problems, e.g., Bayesian inference, as well as automating the procedure for selecting ROMES ingredients for different problems. Future work will also involve integrating the ROMES surrogates into the greedy algorithm for the reduced-basis-space selection; this has the potential to improve ROM quality due to the near-optimal expected effectivities of the error surrogates.

Acknowledgments. We thank Khachik Sargsyan for his support in the understanding, selection, and development of the machine-learning algorithms. This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000. We also acknowledge support by the Department of Energy Office of Advanced Scientific Computing Research under contract 10-014804.

REFERENCES

[1] M. Abramowitz and I.A. Stegun, eds., *Handbook of mathematical functions*, vol. 55, National Bureau of Standards Applied Mathatics Series, 1972.

[2] N.M. Alexandrov, R.M. Lewis, C.R. Gumbert, L.L. Green, and P.A. Newman, *Approximation and model management in aerodynamic optimization with variable-fidelity models*, AIAA Journal of Aircraft, 38 (2001), pp. 1093–1101.

[3] P. Astrid, S. Weiland, K. Willcox, and T. Backx, *Missing point estimation in models described by proper orthogonal decomposition*, IEEE Transactions on Automatic Control, 53 (2008), pp. 2237–2251.

[4] I Babuška and A Miller, *The post-processing approach in the finite element method—part 1: Calculation of displacements, stresses and other higher derivatives of the displacements*, International Journal for numerical methods in engineering, 20 (1984), pp. 1085–1109.

[5] 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*, Comptes Rendus Mathématique Académie des Sciences, 339 (2004), pp. 667–672.

[6] Roland Becker and Rolf Rannacher, *Weighted a posteriori error control in finite element methods*, vol. preprint no. 96-1, Universitat Heidelberg, 1996.

[7] A. Buffa, Y. Maday, Anthony T. Patera, Christophe Prud'homme, and Gabriel Turinici, *A priori convergence of the greedy algorithm for the parametrized reduced basis*, ESAIM-Math. Model. Numer. Anal., 46 (2012), pp. 595–603. Special Issue in honor of David Gottlieb.

[8] C. Canuto, T. Tonn, and K. Urban, *A-posteriori error analysis of the reduced basis method for non-affine parameterized nonlinear pde's*, SIAM J. Numer. Anal, 47 (2009), pp. 2001–2022.

[9] K. Carlberg, *Adaptive h-refinement for reduced-order models*, arXiv preprint 1404.0442, (2014).

[10] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, International Journal for Numerical Methods in Engineering, 86 (2011), pp. 155–181.

[11] 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.

[12] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2737–2764.

[13] M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM J. Sci Comp, 34 (2012), pp. A937–A969.

[14] ———, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A937–A969.

[15] 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 In Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, Aug 2004.

[16] Donald Estep, *A posteriori error bounds and global error control for approximation of ordinary differential equations*, SIAM Journal on Numerical Analysis, 32 (1995), pp. 1–48.

[17] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane, *Engineering Design via Surrogate Modelling - A Practical Guide.*, Wiley, 2008.

[18] D. Galbally, K. Fidkowski, K. Willcox, and O. Ghattas, *Non-linear model reduction for uncertainty quantification in large-scale inverse problems*, International Journal for Numerical Methods in Engineering, 81 (2009), pp. 1581–1608.

[19] Shawn E Gano, John E Renaud, and Brian Sanders, *Hybrid variable fidelity optimization by using a kriging-based scaling function*, AIAA Journal, 43 (2005), pp. 2422–2433.

[20] M.A. Grepl, Y. Maday, N.C. Nguyen, and A.T. Patera, *Efficient reduced-basis treatment of non-affine and nonlinear partial differential equations*, M2AN, Math. Model. Numer. Anal., 41 (2007), pp. 575–605.

[21] M.A. Grepl and A.T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, M2AN, Math. Model. Numer. Anal., 39 (2005), pp. 157–181.

[22] M. A. Grepl, Y. Maday, N. C. Nguyen, and A. T. Patera, *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations*, ESAIM-Mathematical Modelling and Numerical Analysis (M2AN), 41 (2007), pp. 575–605.

[23] Deng Huang, TT Allen, WI Notz, and RA Miller, *Sequential kriging optimization using multiple-fidelity evaluations*, Structural and Multidisciplinary Optimization, 32 (2006), pp. 369–382.

[24] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera, *A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants*, C. R. Math. Acad. Sci. Paris Series I, 345 (2007), pp. 473–478.

[25] P. A. LeGresley, *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*, PhD thesis, Stanford University, 2006.

[26] James Ching-Chieh Lu, *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, PhD thesis, Massachusetts Institute of Technology, 2005.

[27] Andrew March and Karen Willcox, *Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives*, AIAA journal, 50 (2012), pp. 1079–1089.

[28] M. Meyer and H.G. Matthies, *Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods*, Computational Mechanics, 31 (2003), pp. 179–191.

[29] Leo Wai-Tsun Ng and Michael Eldred, *Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation*, in Structures, Structural Dynamics, and Materials and Co-located Conferences, American Institute of Aeronautics and Astronautics, Apr. 2012, pp. –.

[30] N. C. Nguyen, G. Rozza, and A.T. Patera, *Reduced basis approximation and a posteriori error estimation for the time-dependent viscous burgers' equation*, Calcolo, 46 (2009), pp. 157–185.

[31] A.T. Patera and G. Rozza, *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations*, MIT, 2007. Version 1.0, Copyright MIT 2006-2007, to appear in (tentative rubric) MIT Pappalardo Graduate Monographs in Mechanical Engineering.

[32] 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*, J. Fluids Engineering, 124 (2002), pp. 70–80.

[33] Dev Rajnarayan, Alex Haas, and Ilan Kroo, *A multifidelity gradient-free optimization method and application to aerodynamic design*, in 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, Victoria, British Columbia, AIAA, vol. 6020, 2008.

[34] Rolf Rannacher, *The dual-weighted-residual method for error control and mesh adaptation in finite element methods*, MAFELEAP, 99 (1999), pp. 97–115.

[35] C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for Machine Learning*, Adaptative computation and machine learning series, University Press Group Limited, 2006.

[36] 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*, Arch. Comput. Meth. Eng., 15 (2007), pp. 229–275.

[37] D. Ryckelynck, *A priori hyperreduction method: an adaptive approach*, Journal of Computational Physics, 202 (2005), pp. 346–366.

[38] Michael E. Tipping, *Sparse bayesian learning and the relevance vector machine*, J. Mach. Learn. Res., 1 (2001), pp. 211–244.

[39] K. Urban and A.T. Patera, *An improved error bound for reduced basis approximation of linear parabolic problems*, Math. Comp, (2013). (in print).

[40] 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.

[41] D. Wirtz, D. Sorensen, and B. Haasdonk, *A posteriori error estimation for deim reduced nonlinear dynamical systems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A311–A338.

[42] M. Yano and A.T. Patera, *A spacetime variational approach to hydrodynamic stability theory*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science, 469 (2013).

Chapter 5

The GNAT method for nonlinear model reduction: error bound

This chapter presents an error bound for the Gauss–Newton with approximated tensors (GNAT) nonlinear model-reduction method. The bound allows the GNAT error to be quantified and also highlights its advantages in terms of minimizing components of the bound (relative to other sequences of approximate solutions). This work has been published as part of the following journal article: 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, Vol. 242, p. 623647 (2013).

The GNAT method for nonlinear model reduction: error bound

Kevin Carlberg
Sandia National Laboratories
7011 East Ave, MS 9159, Livermore, CA 94550.*

Abstract

The Gauss–Newton with approximated tensors (GNAT) method is a nonlinear model-reduction method that operates on fully discretized computational models. It achieves dimension reduction by a Petrov–Galerkin projection associated with residual minimization; it delivers computational efficency by a hyper-reduction procedure based on the 'gappy POD' technique. Originally presented in Ref. [1], where it was applied to implicit nonlinear structural-dynamics models, this method is further developed here and applied to the solution of a benchmark turbulent viscous flow problem. This paper develops global state-space error bounds that justify the method's design and highlight its advantages in terms of minimizing components of these error bounds.

1 Error bounds

Here, error bounds are developed for any discrete nonlinear model-reduction method assuming that time discretization is performed using the backward-Euler scheme. These bounds highlight the advantages of the GNAT method, as it minimizes components of these error bounds.

The following ODE is considered to considered to be the full-order model for the problem at hand:

$$\begin{aligned} \frac{dw}{dt} &= F\left(w(t), t; \mu\right) \\ w(0) &= w^0(\mu). \end{aligned} \tag{1}$$

When Eq. (1) is time discretized using the backward-Euler scheme, the residual corresponding to time step n, input parameters μ, and the sequence of states computed by the high-dimensional CFD model w^n, $n = 0, \ldots, n_t$ can be written as

$$R^n(w^{n+1}; \mu) = w^{n+1} - w^n - \Delta t F(w^{n+1}, t^{n+1}; \mu). \tag{2}$$

Proposition 1.1 *Assume $f : (w, t; \mu) \mapsto w - \Delta t F(w, t; \mu)$ satisfies the following inverse Lipschitz continuity condition for the online point $\breve{\mu} \in \mathcal{D}$*

$$\frac{\|f(w, t^n; \breve{\mu}) - f(y, t^n; \breve{\mu})\|}{\|w - y\|} \geq \varepsilon > 0, \quad \forall n \in \{1, \ldots, n_t\}. \tag{3}$$

Furthermore, assume that the high-dimensional CFD model employs the backward-Euler scheme for time-integration and computes states w^n, $n = 1, \ldots, n_t$ that satisfy an absolute tolerance for the residual

$$\|R^n(w^{n+1}; \breve{\mu})\| \leq \epsilon_{\text{Newton}}, \quad \forall n \in \{0, \ldots, n_t - 1\}. \tag{4}$$

Then, for any sequence of states $\tilde{w}^n$, $n = 0, \ldots, n_t$ satisfying $\tilde{w}^0 = w^0$, a global error bound for the approximation of the state at the n-th time step is given by

$$\boxed{\|w^n - \tilde{w}^n\| \leq \sum_{k=1}^{n} a^k b_{n-k} \leq \sum_{k=1}^{n} a^k c_{n-k} \leq \sum_{k=1}^{n} a^k d_{n-k},} \tag{5}$$

*Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

where

$$
\begin{aligned}
a &\equiv \sup_{n\in\{1,\ldots,n_t\}} \sup_{w\neq y} \frac{\|w-y\|}{\|f(w,t^n;\breve{\mu})-f(y,t^n;\breve{\mu})\|} \\
b_n &\equiv \epsilon_{\text{Newton}} + \|\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| \\
c_n &\equiv \epsilon_{\text{Newton}} + \|P\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| + \|(I-P)\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| \\
d_n &\equiv \epsilon_{\text{Newton}} + \|P\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| + \|\mathsf{R}^{-1}\|\|\left(I-\mathbb{P}\right)\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| \\
P &\equiv \Phi_R\left[Z\Phi_R\right]^+ Z \\
\mathbb{P} &\equiv \Phi_R\Phi_R^T \\
Z\Phi_R &\equiv \mathsf{QR},
\end{aligned}
\tag{6}
$$

where $\mathsf{Q}\in\mathbb{R}^{n_i\times n_R}$, $\mathsf{R}\in\mathbb{R}^{n_R\times n_R}$, *and* $\bar{R}^n(w;\mu)=w-\tilde{w}^n-\Delta t F(w,t^n;\mu)$.

A provides a proof of the above error bounds. Their consequences include:

- Justification for the minimum-residual approach taken by the tier II Petrov–Galerkin ROM. Namely, by computing $\tilde{w}^{n+1}=\arg\min_{\bar{w}\in w^{n+1(0)}+\mathcal{Y}}\|\bar{R}^n(\bar{w};\breve{\mu})\|$, the tier II Petrov–Galerkin ROM selects the element of the trial subspace that minimizes b_n, $n=1,\ldots,n_t$. This in turn minimizes the tightest error bound in (5).
- Justification for using $\Phi_R=\Phi_J$ in GNAT (snapshot-collection procedures 0 and 1). In this case, the GNAT iterations are equivalent to applying the Gauss–Newton method for minimizing $\|P\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$. As a result, GNAT computes $\tilde{w}^{n+1}=\arg\min_{\bar{w}\in w^{n+1(0)}+\mathcal{Y}}\|P\bar{R}^n(\bar{w};\breve{\mu})\|$, which is the element of the trial subspace that minimizes the second term of both c_n and d_n, $n=1,\ldots,n_t$.
- Justification for computing Φ_R via POD. When computed by POD, the basis Φ_R is the orthogonal basis of dimension n_R that minimizes the average projection error over the set of residual snapshots; this projection error appears as the last term of d_n.
- The tightest bound in (5) is computable by the tier II Petrov–Galerkin ROM if the Lipschitz constant a can be computed or estimated. This is due to the computability of b_n: it requires only the tolerance ϵ_{Newton} and the residual norm at each time step.
- The bound $\sum_{k=1}^{n} a^k c_{n-k}$ (resp. $\sum_{k=1}^{n} a^k d_{n-k}$) is computable by GNAT if the Lipschitz constant a can be computed or estimated and the projection error $\|(I-P)\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$ (resp. $\|(I-\mathbb{P})\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$) can be computed or estimated. This is due to the computability of $\|P\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| = \|\left[Z\Phi_R\right]^+ Z\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$. The projection error $\|(I-P)\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$ can be estimated by

$$
\begin{aligned}
\|(I-P)\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| &\approx \|(\Phi'_R\left[Z\Phi'_R\right]^+ - \Phi_R\left[Z\Phi_R\right]^+)Z\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\| \\
&= \left\| \left(\left[Z\Phi'_R\right]^+ - \begin{bmatrix} \left[Z\Phi_R\right]^+ \\ 0_{(n'_R-n_R)\times n_i} \end{bmatrix} \right) Z\bar{R}^n(\tilde{w}^{n+1};\breve{\mu}) \right\|,
\end{aligned}
\tag{7}
$$

  where $\Phi'_R\equiv\begin{bmatrix}\phi_R^1 & \cdots & \phi_R^{n'_R}\end{bmatrix}$ for some $n'_R>n_R$ (following Ref. [2]). Alternatively, the projection error $\|(I-\mathbb{P})\bar{R}^n(\tilde{w}^{n+1};\breve{\mu})\|$ can be approximated by the sum of the squares of the singular values neglected by Φ_R (following Ref. [3]).

A Error bounds for the solution computed by a discrete nonlinear model reduction method

This section proves the error bound (5) presented in Section 1. For the sake of notational simplicity, the derivation presented here considers the approximation error arising from a given point in the input-parameter

space and therefore omits μ from the arguments of the nonlinear functions. Rewriting the residual (2) in this fashion leads to

$$R^n(w^{n+1}) = w^{n+1} - w^n - \Delta t F(w^{n+1}, t^{n+1}). \tag{8}$$

Similarly, the residual at the the n-th time step arising from any sequence of approximate solutions $\tilde{w}^n$, $n = 0, \ldots, n_t$, e.g., generated by a discrete nonlinear ROM, for the same input parameters can be written as

$$\bar{R}^n(\tilde{w}^{n+1}) = \tilde{w}^{n+1} - \tilde{w}^n - \Delta t F(\tilde{w}^{n+1}, t^{n+1}). \tag{9}$$

Subtracting (9) from (8) yields

$$R^n(w^{n+1}) - \bar{R}^n(\tilde{w}^{n+1}) = w^{n+1} - w^n - \Delta t F(w^{n+1}, t^{n+1}) - \tilde{w}^{n+1} + \tilde{w}^n + \Delta t F(\tilde{w}^{n+1}, t^{n+1}). \tag{10}$$

The above expression can be re-arranged as

$$w^{n+1} - \tilde{w}^{n+1} - \Delta t F(w^{n+1}, t^{n+1}) + \Delta t F(\tilde{w}^{n+1}, t^{n+1}) = R^n(w^{n+1}) - \bar{R}^n(\tilde{w}^{n+1}) + w^n - \tilde{w}^n. \tag{11}$$

Introducing $f : (x, t) \mapsto x - \Delta t F(x, t)$ and the inverse Lipschitz constant[1]

$$\mathcal{L}_f^n \equiv \sup_{x \neq y} \frac{\|x - y\|}{\|f(x, t^{n+1}) - f(y, t^{n+1})\|} \tag{12}$$

allows Eq. (11) to be transformed into the following bound on the *local* approximation error:

$$\|w^{n+1} - \tilde{w}^{n+1}\| \leq \mathcal{L}_f^n \left(\epsilon_{\text{Newton}} + \|\bar{R}^n(\tilde{w}^{n+1})\| + \|w^n - \tilde{w}^n\| \right). \tag{13}$$

Assuming that the initial approximation error is zero[2] ($\tilde{w}^0 = w^0$), the inequality (13) leads to the following result

$$\|w^n - \tilde{w}^n\| \leq \sum_{k=1}^{n} a^k b_{n-k}, \tag{14}$$

where $a = \mathcal{L}_f \equiv \sup_{n \in \{1, \ldots, n_t\}} \mathcal{L}_f^n$ and

$$b_n \equiv \epsilon_{\text{Newton}} + \|\bar{R}^n(\tilde{w}^{n+1})\|. \tag{15}$$

From the triangle inequality, it follows that $\|\bar{R}^n(\tilde{w}^{n+1})\| \leq \|P\bar{R}^n(\tilde{w}^{n+1})\| + \|\left(I - P\right)\bar{R}^n(\tilde{w}^{n+1})\|$ for any P. Hence, another bound for the approximation error is

$$\|w^n - \tilde{w}^n\| \leq \sum_{k=1}^{n} a^k c_{n-k}, \tag{16}$$

where

$$c_n \equiv \epsilon_{\text{Newton}} + \|P\bar{R}^n(\tilde{w}^{n+1})\| + \|(I - P)\bar{R}^n(\tilde{w}^{n+1})\| \tag{17}$$

and $c_n \geq b_n$. The bound (16) is particularly interesting for the case where $P = \Phi_R \left[Z\Phi_R\right]^+ Z$ represents the gappy POD operator because $\|P\bar{R}^n(\tilde{w}^{n+1})\| = \|\left[Z\Phi_R\right]^+ Z\bar{R}^n(\tilde{w}^{n+1})\|$ is readily computable by GNAT.

In B, it is shown that an upper bound for the gappy POD approximation error is

$$\|\left(I - P\right)\bar{R}^n(\tilde{w}^{n+1})\| \leq \|\mathsf{R}^{-1}\|\|(I - \mathbb{P})\bar{R}^n(\tilde{w}^{n+1})\|, \tag{18}$$

where $\mathbb{P} = \Phi_R\Phi_R^T$ defines the orthogonal projector onto range$\left(\Phi_R\right)$, and $Z\Phi_R = \mathsf{QR}$ is the thin QR factorization of $Z\Phi_R$ with $\mathsf{Q} \in \mathbb{R}^{n_i \times n_R}$ and $\mathsf{R} \in \mathbb{R}^{n_R \times n_R}$. Therefore from (18), it follows that yet another error bound for the approximation error is

$$\|w^n - \tilde{w}^n\| \leq \sum_{k=1}^{n} a^k d_{n-k}, \tag{19}$$

[1] Note that $\varepsilon = \frac{1}{\mathcal{L}_f^n}$ in Eq. (3).

[2] This is valid for both the Petrov–Galerkin and GNAT ROMs as they employ the same initial condition as the high-dimensional model (See Algorithm **??**).

where

$$d_n \equiv \epsilon_{\text{Newton}} + \|P\bar{R}^n(\tilde{w}^{n+1})\| + \|\mathsf{R}^{-1}\|\|\left(I-\mathbb{P}\right)\bar{R}^n(\tilde{w}^{n+1})\|. \tag{20}$$

Because $b_n \leq c_n \leq d_n$, it follows that a global bound for the approximation error at the n-th time step with $1 \leq n \leq n_t$ is given by

$$\boxed{\|w^n - \tilde{w}^n\| \leq \sum_{k=1}^{n} a^k b_{n-k} \leq \sum_{k=1}^{n} a^k c_{n-k} \leq \sum_{k=1}^{n} a^k d_{n-k}.} \tag{21}$$

B Error bound for the gappy POD approximation

This section establishes a bound for the error associated with the gappy POD approximation of a vector $g \in \mathbb{R}^N$ using a POD basis $\Phi_f \in \mathbb{R}^{N\times n_f}$ and a set of $n_i \geq n_f$ sample indices $\mathcal{I}$ that define the sample matrix Z.[3]

Define $g^* \equiv \mathbb{P}g$ with $\mathbb{P} \equiv \Phi_f\Phi_f^T$ as the orthogonal (i.e., optimal) projection of g onto range (Φ_f). Also, define the difference between g and its orthogonal projection onto range (Φ_f) as $e \equiv g - g^*$. Finally, define the gappy POD projection matrix $P \equiv \Phi_g\mathsf{R}^{-1}\mathsf{Q}^TZ$, where $Z\Phi_f = \mathsf{QR}$ is the thin QR factorization of $Z\Phi_f$ with $\mathsf{Q} \in \mathbb{R}^{n_i\times n_f}$ and $\mathsf{R} \in \mathbb{R}^{n_f\times n_f}$.

The gappy POD approximation of g is $Pg = P\left(e + g^*\right)$. It can also be written as

$$Pg = Pe + g^* \tag{22}$$

because $Pg^* = g^*$, as $g^* \in$ range (Φ_f). Substituting $g^* = g - e$ into Eq. (22) yields $(I-P)g = (I-P)e$. Therefore,

$$\|(I-P)g\|_2 = \|(I-P)e\|_2 \leq \|(I-P)\|_2\|e\|_2. \tag{23}$$

Because $\|I-P\|_2 = \|P\|_2$ for any projection matrix P not equal to 0 or I, it follows that

$$\|I-P\|_2 = \|P\|_2 = \|\Phi_g\mathsf{R}^{-1}\mathsf{Q}^TZ\|_2 = \|\mathsf{R}^{-1}\|_2. \tag{24}$$

The last equality follows from the fact that Φ_g, Z^T, and Q have orthonormal columns. Substituting (24) in (23) gives the result

$$\boxed{\|(I-P)g\|_2 \leq \|\mathsf{R}^{-1}\|_2\left\|\left(I-\mathbb{P}\right)g\right\|_2.} \tag{25}$$

References

[1] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, April 2011, pp. 155–181.

[2] Drohmann, M., Haasdonk, B., and Ohlberger, M., "Reduced Basis Approximation for Nonlinear Parametrized Evolution Equations based on Empirical Operator Interpolation," *SIAM Journal on Scientific Computing*, Vol. 34, No. 2, 2012, pp. A937–A969.

[3] Chaturantabut, S. and Sorensen, D., "A State Space Error Estimate for POD-DEIM Nonlinear Model Reduction," *SIAM Journal on Numerical Analysis*, Vol. 50, No. 1, 2012, pp. 46–63.

[4] Chaturantabut, S. and Sorensen, D. C., "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[3] This development follows closely the proof of Lemma 3.2 in Ref. [4].

DISTRIBUTION:

1	MS 9159	K. Carlberg, 8954
1	MS 0359	D. Chavez, LDRD Office, 1911
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories