

Manuscript Number:

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

Article Type: Research Paper

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

Corresponding Author: Dr. Kevin Carlberg, PhD

Corresponding Author's Institution: Sandia National Laboratories

First Author: Kevin Carlberg, PhD

Order of Authors: Kevin Carlberg, PhD; Jaideep Ray, PhD; Bart van Bloemen Waanders, PhD

Abstract: Implicit numerical integration of nonlinear ODEs requires solving a system of nonlinear 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 significantly decreasing the number of linear-system solves in the reduced-order-model simulation.

Suggested Reviewers: Ulrich Hetmaniuk PhD
Assistant Professor, Department of Applied Mathematics, University of Washington
hetmaniu@u.washington.edu
He is familiar with reduced-order modeling and has conducted research in this field.

David Ryckelynck PhD
Professor, Centre des Matériaux, Ecole des Mines de Paris
David.Ryckelynck@mines-paristech.fr
He has conducted research in a priori model reduction, which relates to the proposed work.

Bernard Haasdonk PhD
Junior Professor, Department of Mathematics, Universität Stuttgart
haasdonk@mathematik.uni-stuttgart.de
He has conducted excellent research in nonlinear model reduction, and has good experience with time integrators as well.

Opposed Reviewers:

7011 East Ave
MS 9159
Livermore, CA 94550

August 28, 2012

Editor
Computational method in applied mechanics and engineering

To Whom It May Concern:

In this work, we propose a new method to improve performance for nonlinear reduced-order models. The distinguishing feature of the proposed method is that it exploits *temporal behavior* to improve performance; nearly all other work in the field focuses on exploiting *spatial behavior* in the forms of various reduced bases for the state (e.g., POD, modal analysis) and/or nonlinear function (e.g., empirical interpolation). We believe that the method is a novel contribution that can have a broad impact on the field.

Additionally, the method is pragmatic, as it can be applied to first- and second-order ODEs, and to any projection-based nonlinear model-reduction method.

No part of this work has been previously published.

Best regards,

Kevin Carlberg
Jaideep Ray
Bart van Bloemen Waanders

Highlights

Kevin Carlberg, Jaideep Ray, Bart van Bloemen Waanders

August 28, 2012

1. We propose a fundamentally new ‘forecasting’ approach to accelerate nonlinear ROMs.
2. It uses time-domain data to forecast good initial guesses, lowering # Newton its.
3. The method applies to 1st- and 2nd-order ODEs and any projection-based ROM method.
4. Experiments show the method performs best in the presence of smooth dynamics.
5. Experiments show the method can improve ROM performance by a factor of 50.

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

Kevin Carlberg^{a,*}, Jaideep Ray^{a,*}, Bart van Bloemen Waanders^{a,**}

^a*Sandia National Laboratories*

Abstract

Implicit numerical integration of nonlinear ODEs requires solving a system of nonlinear 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 significantly 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 high-fidelity models. When an implicit time 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. In particular, 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

*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.

**MS 0370, P.O. Box 5800 Albuquerque, NM 87185

Email addresses: ktcarlb@sandia.gov (Kevin Carlberg), jairay@sandia.gov (Jaideep Ray), bartv@sandia.gov (Bart van Bloemen Waanders)

URL: sandia.gov/~ktcarlb (Kevin Carlberg), csmr.ca.sandia.gov/~jairay/ (Jaideep Ray), www.cs.sandia.gov/~bartv/ (Bart van Bloemen Waanders)

1
2
3
4
5
6
7 this, these methods exploit knowledge of the system’s dominant *spatial behavior*—as observed during
8 ‘training simulations’ conducted *a priori*—to decrease the simulation’s *spatial complexity*, which we
9 define as the computational cost of each linear-system solve.¹ To do so, these methods 1) decrease the
10 dimension of the linear systems by projection, and 2) approximate vector-valued nonlinear functions
11 by sampling methods that compute only a few of the vector’s entries (e.g., empirical interpolation
12 [1, 2], Gappy POD [3]). However, these techniques are often insufficient to adequately reduce the
13 computational cost of the simulation. For example, Ref. [4] presented results for the GNAT nonlinear
14 model-reduction technique applied to a large-scale nonlinear turbulent-flow problem. The reduced-
15 order model generated solutions with sub-1% errors and reduced the spatial complexity by a factor
16 of 637. However, the total number of linear-system solves required for the reduced-order-model simu-
17 lation, which we define as the *temporal complexity*, remained large. In fact, the temporal complexity
18 was decreased by a factor of only 1.5. As a result, the total computing resources (computing cores
19 \times wall time) required for the simulation were decreased by a factor of 438, but the wall time was
20 reduced by a factor of merely 6.9. While these results are promising (especially in their ability to
21 reduce spatial complexity), the time integration of nonlinear dynamics remains problematic and often
22 precludes real-time performance.

23
24 The goal of this work is exploit knowledge of the system’s *temporal behavior* as observed during
25 the training simulations to decrease the temporal complexity of (deterministic) reduced-order-model
26 simulations. For this purpose, we first briefly review methods that exploit observed temporal behavior
27 to improve computational performance.

28 Temporal forecasting techniques have been investigated for many years with a specific focus on re-
29 ducing wall time in a stable manner with maximal accuracy. The associated body of work is large and
30 a comprehensive review is beyond the scope of this paper. However, this work focuses on time integra-
31 tion for reduced-order models plagued by highly nonlinear dynamics; several categories of specialized
32 research efforts provide an appropriate context for this research.

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

45 As fundamental as time-series analysis, time integrators for ordinary differential equations (ODEs)
46 employ Taylor-series expansions to provide reasonably accurate forecasts of the state or the unknown
47 at each time step. Time integrators employ such a forecast for two purposes. First, algorithms with
48 adaptive time steps employ interpolation to obtain solutions (and their time derivatives) at arbitrary
49 points in time. Implicit time integrators for nonlinear ODEs, which require the iterative solution to
50 nonlinear algebraic systems at each time step, use past history (of the current trajectory) to forecast
51 an accurate guess of the unknown in the algebraic system, e.g., Ref. [13]. Again, forecasting by
52 Taylor-series expansion makes no use of the temporal behavior observed during training simulations.

53 Closely connected to time integration but specialized to leverage developments in high-performance
54
55

56
57 ¹A sequence of linear systems arises at each time step when a Newton-like method is employed to solve the system
58 of nonlinear algebraic equations.
59
60

1
2
3
4
5
6
7 computing, time parallel methods can offer computational speedup when integrating ODEs. Dating
8 back to before the general availability of parallel computers, researchers speculated about the benefits
9 of decomposing the temporal domain across multiple processors [14]. Advancements have been made
10 from parallel multigrid to parareal techniques [15, 16, 17, 18]. Although time-domain decomposition
11 algorithms have demonstrated speedup, they are limited in comparison to the spatial domain decom-
12 position methods and they require a careful balance between stability and computational efficiency
13 [19]. It is possible that these methods could further improve performance in a model-reduction setting
14 [20] (and could complement the method proposed in this work), but near real-time performance is
15 likely unachievable through time-parallel methods alone.

16 To some extent, exploiting temporal behavior has been explored in nonlinear model reduction.
17 Bos et al. [21] proposed a reduced-order model in the context of explicit time integration wherein the
18 generalized coordinates are computed based on a best-linear-unbiased (BLU) estimate approach. Here,
19 the reduced state coordinates at time step $n + 1$ are computed using empirically derived correlations
20 between the reduced state coordinates and 1) their value at the previous time step, 2) the forcing input
21 at the previous time step, and 3) a subset of the full-order state. However, the errors incurred by this
22 time-integration procedure (compared with standard time integration of the reduced-order model) are
23 not assessed or controlled. This can be problematic in realistic scenarios, where error estimators and
24 bounds are essential. Another class of techniques called *a priori* model reduction methods [22, 23] build
25 a reduced-order model ‘on the fly’, i.e., over the course of a given time integration. These techniques
26 try to use the reduced-order model at as many time steps as possible; they use the high-fidelity model
27 when the reduced-order model is deemed to be inaccurate. So, these techniques employ the reduced-
28 order model as a tool to accelerate the high-fidelity-model simulation. In contrast, this work aims to
29 accelerate the reduced-order-model simulation itself. Further, these methods differ from the present
30 context in that there are no training experiments conducted *a priori* from which to glean information
31 about the model’s temporal behavior.
32

33 In this work, we propose a method that exploits a set of complete trajectories observed during
34 training simulations to decrease the temporal complexity of a reduced-order-model simulation. The
35 method 1) forecasts the unknown variable in the reduced-order system of nonlinear equations, and 2)
36 uses this forecast as an initial guess for the Newton-like solver. To compute the forecast, the method
37 employs the Gappy POD method [3], which extrapolates the unknown variable at future time steps
38 by exploiting 1) the unknown variable for the previous α time steps (where α is the memory of the
39 process), and 2) a database of time histories of the unknown variable. If the forecast is accurate, then
40 the Newton-like solver will require very few iterations to converge, thereby decreasing the number
41 of linear-system solves in the simulation. The method is straightforward to implement: the training
42 stage simply requires collecting an additional set of snapshots during the training simulations. The
43 reduced-order-model simulation simply requires an external routine for determining the initial guess
44 for the Newton-like solver.
45
46

47 2. Problem formulation

48
49 This section provides the context for this work. Section 2.1 describes the class of full-order models
50 we consider, which includes first- and second-order ODEs numerically solved by implicit time integra-
51 tion. Section 2.2 describes the reduced-order modeling strategies for which the proposed technique is
52 applicable.
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7 *2.1. Full-order model*

8 *2.1.1. First- and second-order ODEs*

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

11
12
$$\dot{x} = f(x; t, p(t), q) \tag{1}$$

13
14
$$x(0, p, q) = x^0(q). \tag{2}$$

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

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

21
22
$$\ddot{x} = g(x, \dot{x}; t, p(t), q) \tag{3}$$

23
24
$$x(0, p, q) = x^0(q) \tag{4}$$

25
26
$$\dot{x}(0, p, q) = v^0(q). \tag{5}$$

27 Here, the function $g : \mathbb{R}^N \times \mathbb{R}^N \times [0, T] \times \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^N$ is nonlinear in at least its first two arguments,
28 and the parameterized initial velocity is denoted by $v^0 : \mathbb{R}^p \rightarrow \mathbb{R}^N$.²

29
30
31 *2.1.2. Implicit time integration*

32 Given forcing and parametric inputs, the numerical solution to the full-order model described by
33 (1)–(2) or (3)–(5) can be computed via numerical integration. For systems exhibiting stiffness, an
34 implicit integration method is often the most computationally efficient choice; it is even essential
35 in many cases [24]. When an implicit time integrator is employed, s coupled systems of nonlinear
36 equations are solved at each time step $n = 1, \dots, M$:

37
38
$$R_i^n(w^{n,1}, \dots, w^{n,s}; p, q) = 0, \quad i = 1, \dots, s. \tag{6}$$

39 Here, the function $R_i^n : \mathbb{R}^N \times \dots \times \mathbb{R}^N \times \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^N$ is nonlinear in at least its first s arguments and
40 the unknowns $w^{n,i} \in \mathbb{R}^N$, $i = 1, \dots, s$ are implicitly defined by (6). As discussed in Appendix A and
41 Appendix B, the unknowns $w^{n,i}$ represent the state, velocity, or acceleration at points $t^{n-1} + c_i h^n$,
42 where $c_i \in [0, 1]$, $i = 1, \dots, s$ is defined by the time integrator:

43
44
45
$$w^{n,i} \equiv w^{n,i}(p, q) \equiv w(t^{n-1} + c_i h^n; p, q). \tag{7}$$

46 So, 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
47 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.

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

49
50
51
52
53
$$x^n = \gamma x^{n-1} + \sum_{i=1}^s \delta_i w^{n,i}, \tag{8}$$

54
55
56
57
58 ²Note that an N -dimensional second-order ODE can be rewritten as $2N$ -dimensional first-order ODE.

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

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

where ϵ and ξ_i , $i = 1, \dots, s$ 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 (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 .³ 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 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.⁴ 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 unknowns.

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 rest of this paper assumes $s = 1$, and 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 inputs $D_{\text{train}} \equiv \{(p^k, q^k)\}_{k=1}^{n_{\text{train}}}$ 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) dimension 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 $(p, q) \notin D_{\text{train}}$ during the ‘online’ stage.

³Assuming 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 $\mathcal{O}(\omega^2 s N K M)$ if a direct linear solver is used. It is $\mathcal{O}(L \omega s N K M)$ 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.

⁴This 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 improves stability and therefore permits a larger time-step size. As a result, the reduced-order state equations can be solved fewer times than the full-order state equations.

1
2
3
4
5
6
7 *2.2.1. Dimension reduction*

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

10
$$\tilde{x}(t, p, q) = x^0(q) + \Phi \hat{x}(t, p, q) \tag{10}$$

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

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

13
14
15 Here, the trial basis (in matrix form) is denoted by $\Phi \equiv [\phi_1 \cdots \phi_{\hat{N}}] \in \mathbb{R}^{N \times \hat{N}}$ with $\Phi^T \Phi = I$. The
16 generalized state is denoted by $\hat{x} \equiv [\hat{x}_1 \cdots \hat{x}_{\hat{N}}]^T \in \mathbb{R}^{\hat{N}}$. When the unknown variable computed at
17 each time step (see Section 2.1.2) corresponds to the state, velocity, or acceleration, we can express it
18 as

19
$$w(t, p, q) = w^0(q) + \Phi \hat{w}(t, p, q), \tag{13}$$

20 where $\hat{w} \equiv [\hat{w}_1 \cdots \hat{w}_{\hat{N}}]^T \in \mathbb{R}^{\hat{N}}$ denotes the vector of generalized unknowns.

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

22
$$\Phi \dot{\hat{x}} = f(x^0(q) + \Phi \hat{x}; t, p(t), q), \tag{14}$$

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

24
$$\Phi \ddot{\hat{x}} = g(x^0(q) + \Phi \hat{x}, \Phi \dot{\hat{x}}; t, p(t), q). \tag{15}$$

25 The overdetermined ODEs described by (14) and (15) may not be solvable, because $\text{image}(f) \not\subset$
26 $\text{range}(\Phi)$ and $\text{image}(g) \not\subset \text{range}(\Phi)$ in general. Several methods exist to compute a solution.

27 *Project, then discretize in time.* This class of model-reduction methods first carries out a projection
28 process on the ODE followed by a time-integration of the resulting low-dimensional ODE. The (Petrov–
29 Galerkin) projection process enforces orthogonality of the residual corresponding to overdetermined
30 ODE (14) or (15) to an \hat{N} -dimensional test subspace $\text{range}(\Psi)$, with $\Psi \in \mathbb{R}^{N \times \hat{N}}$. For first-order ODEs,
31 this leads to

32
$$\dot{\hat{x}} = (\Psi^T \Phi)^{-1} \Psi^T f(x^0(q) + \Phi \hat{x}; t, p(t), q). \tag{16}$$

33 For second-order ODEs, the result is

34
$$\ddot{\hat{x}} = (\Psi^T \Phi)^{-1} \Psi^T g(x^0(q) + \Phi \hat{x}, \Phi \dot{\hat{x}}; t, p(t), q). \tag{17}$$

35 Galerkin projection corresponds to the case where $\Psi = \Phi$.

36 Because Eq. (16) (resp. (17)) is an ODE of the same form as (1) (resp. (3)), it can be solved using
37 the same numerical integrator that was used to solve (1) (resp. (3)). Further, the same time-step sizes
38 are often employed, as the time-step size is determined by accuracy (not stability) for implicit time
39 integrators. For both first- and second-order ODEs, this again leads to a system of nonlinear equations
40 to be solved at each time step $n = 1, \dots, M$:

41
$$(\Psi^T \Phi)^{-1} \Psi^T R^n(w^0(q) + \Phi \hat{w}^n; p, q) = 0. \tag{18}$$

42 The unknown \hat{w}^n can be computed by applying Newton’s method to (18). Then, the explicit updates
43 (8)–(9) can proceed as usual.

1
2
3
4
5
6
7 *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 (14) or (15). However, the
8 resulting algebraic system of N nonlinear equations in \hat{N} unknowns remains overdetermined:
9

$$10 \quad R^n (w^0(q) + \Phi \hat{w}^n; p, q) = 0. \quad (19)$$

11
12 To compute a unique solution to (19), orthogonality of the discrete residual R^n to a test subspace
13 range (Ψ) can be enforced. However, this leads to a reduced system of nonlinear equations equivalent
14 to (18). So, in this case, the two classes of model-reduction techniques are equivalent.

15 On the other hand, to compute a unique solution to (19), the discrete-residual norm can be mini-
16 mized [26, 4, 27, 28, 29], which ensures *discrete optimality* [4]:
17

$$18 \quad \hat{w}^n = \arg \min_{y \in \mathbb{R}^{\hat{N}}} \|R^n (w^0(q) + \Phi y; p, q)\|_2^2. \quad (20)$$

19
20 The unknown \hat{w}^n can be computed by applying a Newton-like nonlinear least-squares method (e.g.,
21 Gauss–Newton, Levenberg–Marquardt) to (20). Again, explicit updates (8)–(9) can proceed after the
22 unknowns are computed.
23

24 2.2.2. Spatial-complexity reduction

25 For nonlinear dynamical systems, the dimension reduction described in Section 2.2.1 is insufficient
26 to guarantee a reduction in the computational cost of each linear-system solve. The reason is that
27 the full-order residual depends on the state, so it must be recomputed and subsequently projected or
28 minimized at each Newton-like iteration.
29

30 For this reason, nonlinear model-reduction techniques employ a procedure to reduce the spatial-
31 complexity, i.e., decrease the computational cost of computing and projecting or minimizing the nonlin-
32 ear residual.⁵ In particular, the class of ‘function sampling’ techniques replace the full-order nonlinear
33 residual with an approximation $\tilde{R} \approx R$ that is inexpensive to compute. Then, $R^n \leftarrow \tilde{R}^n$ is employed
34 in (18) or (20) to compute the unknowns \hat{w}^n .
35

Methods in this class can be categorized as follows:

- 36 1. *Collocation approaches.* These methods employ a residual approximation that sets many of the
37 residual’s entries to zero:

$$38 \quad \tilde{R}^n = \mathbf{Z}^T \mathbf{Z} R^n. \quad (21)$$

39 Here, \mathbf{Z} is a restriction matrix consisting of selected rows of $I_{N \times N}$. This approach has been
40 developed for Galerkin projection [30, 22] and discrete-residual minimization [29].

- 41 2. *Function-reconstruction approaches.* These methods employ a residual approximation that com-
42 putes a few entries of the residual or nonlinear function, and subsequently ‘fills in’ the remaining
43 entries via interpolation or least-squares regression. That is, these methods apply one of the
44 following approximations:
45

$$46 \quad \tilde{R}^n = \Phi_R (\mathbf{Z} \Phi_R)^+ \mathbf{Z} R^n \quad (22)$$

$$47 \quad \tilde{f} = \Phi_f (\mathbf{Z} \Phi_f)^+ \mathbf{Z} f \quad (23)$$

$$48 \quad \tilde{g} = \Phi_g (\mathbf{Z} \Phi_g)^+ \mathbf{Z} g. \quad (24)$$

49
50 Here, Φ_R , Φ_f , and Φ_g are empirically derived bases used to approximate the nonlinear residual,
51 velocity, and acceleration, respectively.⁶ A superscript $+$ denotes the Moore–Penrose pseu-
52 doinverse. This approach has been developed for Galerkin projection [30, 21, 2, 31, 32] and
53 discrete-residual minimization [26, 4].
54
55
56

57 ⁵Such techniques are occasionally referred to as ‘hyper-reduction’ techniques [22].

58 ⁶When the bases are computed via POD, this technique is known as Gappy POD [3].

3. Temporal-complexity reduction

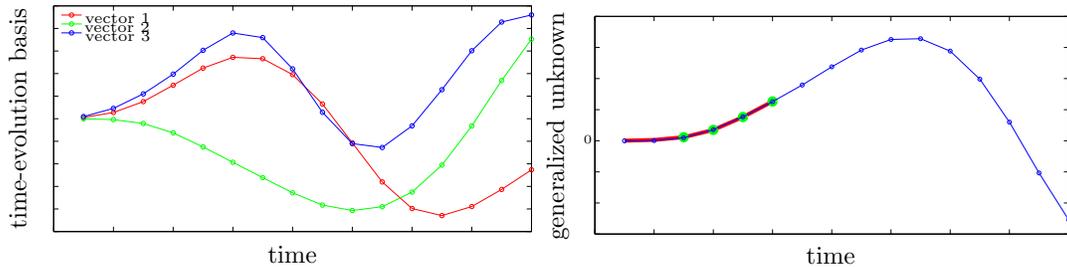
While the model-reduction approaches described in the previous section decrease the computational cost of each linear-system solve (i.e., spatial complexity), they do not necessarily decrease the number of linear-system solves (i.e., temporal complexity). The goal of this work is devise a method that 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 , $j = 1, \dots, \hat{N}$. Each basis represents the 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.



(a) Offline: the computed time-evolution POD basis (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

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

The proposed method is expected to be effective if the temporal behavior of the generalized unknowns is similar across input variation. The method is independent of the dimension-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 p and parametric inputs 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:

$$Y_k = [w^0(p_k, q_k) \cdots w^{M-1}(p_k, q_k)] \quad (25)$$

for $k = 1, \dots, n_{\text{train}}$, with $Y_k \in \mathbb{R}^{N \times M}$. Here, $p_k \in \mathbb{R}^p$ denotes the forcing inputs for training simulation k , and $q_k \in \mathbb{R}^q$ denotes the parametric inputs for training simulation k .

In some cases, the snapshot matrices Y_k , $k = 1, \dots, n_{\text{train}}$ are already available from computing the trial basis Φ . This occurs, for example, when proper orthogonal decomposition (POD) is employed to compute Φ and the time integrator’s unknown is the state vector.

2. Compute the corresponding snapshots of the generalized unknown:

$$\begin{aligned} \hat{Y}_k &\equiv \Phi^T [Y_k - w^0(q_k) \mathbf{1}^T] \\ &= [\hat{w}^0(p_k, q_k) \cdots \hat{w}^{M-1}(p_k, q_k)] \end{aligned} \quad (26)$$

for $k = 1, \dots, n_{\text{train}}$, where orthogonality of the trial basis $\Phi^T \Phi = I$ has been used. Here, $\hat{Y}_k \in \mathbb{R}^{\hat{N} \times M}$ and $\mathbf{1} \in \mathbb{R}^M$ denotes a vector of ones.

3. Compute the time-evolution bases via the singular value decomposition. Defining $\hat{Y}_k \equiv [\hat{y}_{1,k} \cdots \hat{y}_{\hat{N},k}]^T$, where $\hat{y}_{j,k} \in \mathbb{R}^M$ can be interpreted as a snapshot of the time evolution of the j th generalized unknown \hat{w}_j during training simulation k , this step amounts to

$$[\hat{y}_{j,1} \cdots \hat{y}_{j,n_{\text{train}}}] = U_j \Sigma_j V_j^T \quad (28)$$

$$\Xi_j = [u_{j,1} \cdots u_{j,a_j}], \quad (29)$$

for $j = 1, \dots, \hat{N}$. Here, $U_j \equiv [u_{j,1} \cdots u_{j,n_{\text{train}}}] \in \mathbb{R}^{M \times n_{\text{train}}}$ and $a_j \leq n_{\text{train}}$.

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

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. When the number of Newton iterations exceeds a threshold value τ , this indicates a poor forecast; in this case, the forecast is recomputed. If the forecast is accurate, then the number of iterations needed to converge from the (improved) initial guess will be

Algorithm 1 Online: temporal-complexity-reduction method

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

Output: Generalized state at all M time steps: \hat{x}^n , $n = 1, \dots, M$

```

1: for  $n = 1, \dots, M$  do
2:   if forecast  $\hat{w}(t^{n-1} + c_1 h^n)$  is available then
3:     Set initial guess for Newton solver to  $\hat{w}_j^{n(0)} = \hat{w}_j(t^{n-1} + c_1 h^n)$ ,  $j = 1, \dots, \hat{N}$ 
4:   else
5:     Use typical initial guess for Newton solver (e.g., polynomial extrapolation of unknown)
6:   end if
7:   Solve reduced-order equations (18) or (20) with a Newton-like method and specified initial guess.
   The number of Newton-like iterations required for convergence is denoted by  $K$ 
8:   if  $K > \tau$  and  $(n - 1) \geq \max_j a_j$  then {recompute forecast}
9:     Set memory  $\alpha \leftarrow \min(n - 1, \alpha_{\max})$ 
10:    Compute forecasting coefficients  $z_j$ ,  $j = 1, \dots, \hat{N}$  using the unknown at the previous  $\alpha$  time
    steps (see Eq. (30))
11:    Set forecast to be  $\hat{\mathbf{w}}_j = \Xi_j z_j$  and define  $\hat{w}_j \equiv h^{-1}(\hat{\mathbf{w}}_j)$ ,  $j = 1, \dots, \hat{N}$ 
12:  end if
13: end for

```

drastically reduced, thereby decreasing \bar{K} and hence the temporal complexity. Algorithm 1 outlines the proposed technique.

To compute the forecasting coefficients in step 10 of Algorithm 1, we propose using the Gappy POD approach of Everson and Sirovich [3]. This approach computes coefficients z_j via the following linear least-squares problem:

$$z_j = \arg \min_{z \in \mathbb{R}^{a_j}} \|Z(n, \alpha) \Xi_j z - Z(n, \alpha) h(\hat{w}_j)\| \quad (30)$$

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

$$Z(n, \alpha) \equiv [e_{n-\alpha-1} \cdots e_{n-1}]^T, \quad (31)$$

where e_i denotes the i th canonical unit vector. Note that $\alpha \geq a_j$ is required for there to be a unique solution to (30). The function h in (30) ‘unrolls’ time according to the time discretization; we define $h : x \mapsto \mathbf{x}$ with $\mathbf{x} \equiv [\mathbf{x}_1 \cdots \mathbf{x}_M]^T \in \mathbb{R}^M$ as

$$\mathbf{x}_n = x(t^{n-1} + c_1 h^n), \quad n = 1, \dots, M. \quad (32)$$

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. (17) with $\Psi = \Phi$), Galerkin projection with collocation (Eq. (21)), and Galerkin projection with least-squares reconstruction of the residual (Eq. (22)). We consider a sequence of problems that pose increasing difficulty to the method.

Section 4.2 considers the ideal scenario for the method: the online inputs are identical to the training inputs, and the reduced bases are not truncated. In this case, the temporal behavior of the

1
2
3
4
5
6
7 system is exactly predictable, because (in exact arithmetic) the online response is the same as the
8 training response. Therefore, we expect the proposed method to work extremely well in this case.

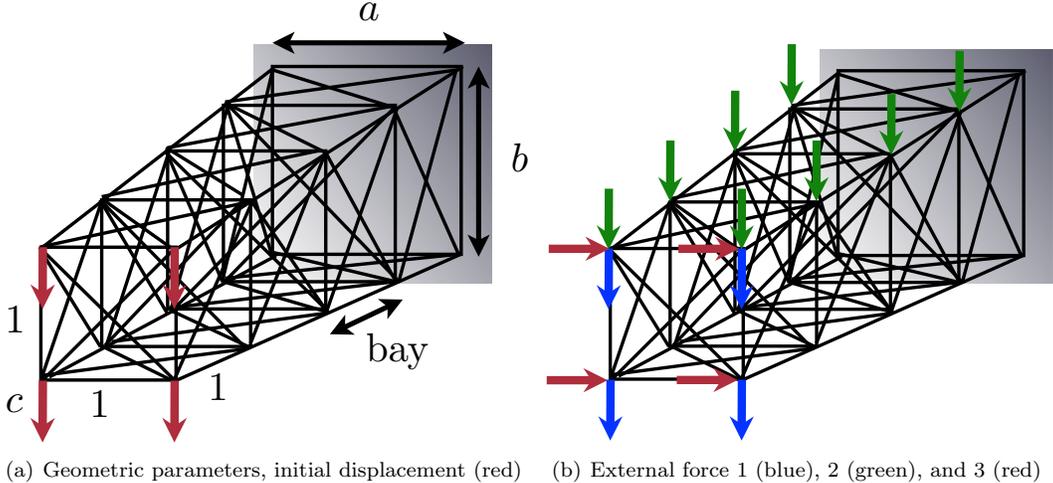
9 Section 4.3 assesses the method’s performance in a more challenging setting. Here, the online
10 inputs are different from the training inputs (i.e., predictive scenario), so the temporal behavior is
11 not identical to that observed during the training simulations. The parametric inputs correspond
12 to material properties and shape parameters, and the external force is set to zero; this leads to a
13 free-vibration problem. As a result, the dynamics encountered in this example are relatively smooth.

14 Section 4.4 considers a more challenging predictive scenario wherein rich dynamics—generated from
15 a high-frequency external force—characterize the response. Here, the parametric inputs correspond to
16 the magnitudes of the high-frequency forces. All material-property and shape variables are held fixed.

17 Finally, Section 4.5 pushes the method to its limit by considering a predictive scenario characterized
18 by high-frequency external forces as well as variations in material properties and shape variables.
19

20
21 *4.1. Problem setup*

22 Figure 2(a) depicts the parameterized, non-conservative clamped-free truss structure, where the
23 arrows indicate the initial displacement of magnitude c . The full-order model is constructed by the
24



44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Figure 2: Clamped-free parameterized truss structure

finite-element method. It consists of sixteen 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 750
bays, which leads to 9×10^3 degrees of freedom in the full-order model. The bar elements model
geometric nonlinearity, which results in a high-order nonlinearity in the strain energy. Each bay has a
(unitless) depth of 1 and a cross-sectional area determined by the parameterized geometry.

The parameters $q_i \in [-1, 1]$, $i = 1, \dots, 9$ have the following effect on the model:

$$\begin{aligned}
& \text{density} = 1 + 0.5q_1 \\
& \text{bar area} = 1 + 0.5q_2 \\
& \text{modulus of elasticity} = 1 + 0.5q_3 \\
& \text{base width } a = 1 + 0.5q_4 \\
& \text{base height } b = 1 + 0.5q_5 \\
& \text{initial end displacement } c = 1 + 0.5q_6 \\
& \text{magnitude } \gamma_1 = 0.5(1 + q_7) \\
& \text{magnitude } \gamma_2 = 0.5(1 + q_8) \\
& \text{magnitude } \gamma_3 = 0.5(1 + q_9)
\end{aligned}$$

The equations of motion for this model are

$$M(q)\ddot{x} + C(q)\dot{x} + f_{\text{int}}(x; q) = p(t; q). \quad (33)$$

Here, $M(q) \in \mathbb{R}^{N \times N}$ denotes the symmetric-positive-definite mass matrix, the internal forced is denoted by $f_{\text{int}} : \mathbb{R}^N \times \mathcal{D} \rightarrow \mathbb{R}^N$ with $(x, q) \mapsto f_{\text{int}}(x; q)$, and the symmetric-positive-semidefinite Rayleigh viscous damping matrix, denoted by $C(q) \in \mathbb{R}^{N \times N}$, is of the form

$$C(q) = \alpha M(q) + \beta \nabla_x f_{\text{int}}(x^0; q). \quad (34)$$

Note that $\nabla_x f_{\text{int}}(x^0; q)$ represents the tangent stiffness matrix at the initial condition. In this work, the Rayleigh coefficients α and β are determined by matching the first two damped frequencies with the first two undamped frequencies of the nominal structure ($q_i = 0$, $i = 1, \dots, 6$), while enforcing a damping ratio of $\zeta = \sin^{-1}(2^\circ)$ for the first two modes [38, Eq. (7)].

We consider an external force composed of three forces:

$$p(q, t) = \sum_{i=1}^3 \mathbf{r}_i r_i(q, t), \quad (35)$$

where $\mathbf{r}_i \in \mathbb{R}^N$ and $r_i : \mathbb{R}^q \times [0, T] \rightarrow \mathbb{R}$ for $i = 1, \dots, 3$. Figure 2(b) depicts the forces' spatial distributions, which lead to vectors \mathbf{r}_i , $i = 1, \dots, 3$ through the finite-element formulation. The parameterized, time-dependent magnitudes of these functions are:

$$r_1(q, t) = \gamma_1(q) \mathbf{1}_{\mathbb{R}_+}(t - T/2) \sin(\lambda_1(t - T/2)) \quad (36)$$

$$r_2(q, t) = \gamma_2(q) \mathbf{1}_{\mathbb{R}_+}(t - T/2) \sin(\lambda_2(t - T/2)) \quad (37)$$

$$r_3(q, t) = \gamma_3(q) \mathbf{1}_{\mathbb{R}_+}(t - T/2) \sin(\lambda_3(t - T/2)), \quad (38)$$

where γ_i , $i = 1, \dots, 3$ denotes the maximum force magnitudes, and

$$\mathbf{1}_A(\tau) = \begin{cases} 1, & \tau \in A \\ 0, & \text{otherwise} \end{cases} \quad (39)$$

is the indicator function. The frequencies are (arbitrarily) set to $\lambda_1 = \omega_6$, $\lambda_2 = 5\omega_6$, $\lambda_3 = 0.5\omega_6$, where ω_6 denotes the sixth largest natural frequency of the structure in its nominal configuration ($q_i = 0$, $i = 1, \dots, 6$).

The equations of motion (33) can be rewritten in the standard form of Eqs. (3)–(5) as

$$\ddot{x} = M(q)^{-1} (p(t; q) - C(q)\dot{x} - f_{\text{int}}(x; q)) \quad (40)$$

$$x(0, p, q) = x^0(q) \quad (41)$$

$$\dot{x}(0, p, q) = v^0(q). \quad (42)$$

The nonlinear function defining the second-order ODE is

$$g(x, \dot{x}; t, p, q) = M(q)^{-1} (p(t; q) - C(q)\dot{x} - f_{\text{int}}(x; q)). \quad (43)$$

We employ an implicit Nyström time integrator to compute the numerical solution to Eqs. (40)–(42). 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: $w^n = \ddot{x}(t^{n-1} + 1/2h)$, $n = 1, \dots, M$. Multiplying the corresponding residual by $M(q)$ yields

$$R^n(w^n) = M(q)w^n + C(q) \left[\dot{x}^{n-1} + \frac{1}{2}hw^n \right] + f_{\text{int}} \left(x^{n-1} + \frac{1}{2}h\dot{x}^{n-1} + \frac{1}{4}h^2w^n; q \right) - p(t^{n-1} + \frac{1}{2}h; q). \quad (44)$$

To solve $R^n(w^n) = 0$ at each time step, Newton’s method is applied. Each linearized system is solved directly using the Cholesky factorization (the Jacobian of the residual is symmetric), and convergence of Newton’s method is declared when the 2-norm of the residual is less than or equal to $10^{-4}\|R^n(0)\|_2$.

The time-interval length is set to $T = 25$. A time-step size of $h = 0.6$ is employed in the unforced case; it is set to $h = 0.5$ in the forced case. These values were determined by a convergence study on the nominal configuration defined by $q_i = 0$, $i = 1, \dots, 6$ and $q_i = -1$, $i = 7, \dots, 9$ (unforced) or $q_i = -1$, $i = 7, \dots, 9$ (forced).

To construct the reduced-order models, we collect snapshots of the required quantities for $(p, q) \in D_{\text{train}}$ and $t \in [0, T]$. The trial basis Φ is determined via POD. Snapshots of the state are collected

$$\mathcal{X}_x = \{x^{n-1} + h\dot{x}^{n-1} + \frac{h}{2}\ddot{x}^{n,1} - x(0; q) \mid n = 1, \dots, M; (q, p) \in D_{\text{train}}\} \quad (45)$$

and the trial basis is set to $\Phi = \Phi^e(\mathcal{X}_x, \nu_x)$, where $\nu_x \in [0, 1]$ is an ‘energy criterion’ and Φ^e is defined by Algorithm 2 in Appendix C. For Galerkin projection with least-squares residual reconstruction, the following snapshots are collected during the (full-order model) training simulations:

$$\mathcal{X}_R = \{R^n(w^{n(k)}) \mid n = 1, \dots, M; k = 0, \dots, K(n) - 1; (q, p) \in D_{\text{train}}\}. \quad (46)$$

Here, $K(n)$ denotes the number of Newton steps taken at time step n . The basis Φ_R is set to $\Phi_R = \Phi^e(\mathcal{X}_R, \nu_R)$ with $\nu_R \in [0, 1]$. The same sampling matrix \mathbf{Z} is used for the collocation and Gappy POD approximations; it is determined using the GNAT model-reduction method’s approach for selecting the sample matrix [26, 4]. The number of rows in \mathbf{Z} is set to be twice the number of columns in Φ_R .

The output of interest is the downward displacement of the bottom-left point at the tip of the structure (denoted by d) for all time steps. To quantify the performance of the reduced-order models,

the following metrics are used:

$$\varepsilon = \frac{\frac{1}{M} \sum_{n=0}^M |d^n - d_{\text{FOM}}^n|}{\max_n d_{\text{FOM}}^n - \min_n d_{\text{FOM}}^n} \quad (47)$$

$$\kappa = \frac{\bar{K}_{\text{FOM}}}{\bar{K}} \quad (48)$$

$$S = \frac{T_{\text{FOM}}}{T} \quad (49)$$

Here, 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 per time step. 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.

In all experiments, the forecasting method is compared with a ‘no forecasting’ case. For this case, the initial guess corresponds to a first-order approximation of the displacement within the time interval: $x^{(0)}(t) = x^{n-1} + (t - t^{n-1}) \dot{x}^{n-1}$ for $t \in [t^{n-1}, t^n]$, or equivalently $\ddot{x}^{n,1} = w^{n(0)} = 0$. Also, the forecasting method always employs untruncated time-evolution bases: $a_j = n_{\text{train}}$ for $j = 1, \dots, \hat{N}$.

4.2. Ideal case: 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 ($\nu_x = \nu_R = 1.0$). In this scenario, the full-order model’s temporal behavior encountered online is exactly the same as that observed during training simulations; for this reason, we expect the proposed method to perform very well. We consider a single configuration ($n_{\text{train}} = 1$) characterized by $q_i = 0$, $i = 1, \dots, 9$. For the forecasting technique, the Newton-step threshold is set to $\tau = 0$ and the maximum memory is set to $\alpha_{\text{max}} = 12$.

Table 1 and Figure 3 report the results. First, note that the relative errors generated by ROMs are essentially zero. This is expected, because the reduced bases are not truncated and the inputs are invariant. Next, note that the Galerkin ROM 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 significant speedups. Also, it is evident that the reduced-order models exhibit no temporal-complexity reduction (i.e., $\kappa \leq 1.0$) in the absence of the proposed forecasting technique.

When the models employ the proposed forecasting technique, the number of Newton iterations drastically decreases, leading to temporal-complexity reductions of $\kappa = 49.5$ for two ROMs and $\kappa = 6.25$ for the third. In turn, this leads to significantly improved wall-time speedups in all cases. This 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 bases have not been truncated. So, the forecast is ‘perfect’ after only one time step for the Galerkin and Galerkin with Gappy POD ROMs; no Newton steps are required after this. The next sections investigate the forecasting method’s performance in the (more realistic) case of varying inputs and truncated bases.

4.3. Unforced dynamics, varying structure

This experiment assesses the performance of the method when applied to the problem in the absence of external forcing, but subject to changes in the structure’s shape, material properties, and initial condition. This will test the method for a problem exhibiting ‘smooth’ dynamics, as it amounts to a free vibration problem. We randomly choose six training configurations ($n_{\text{train}} = 6$) for which we

ROM method	relative error ε	No forecasting			With forecasting		
		Newton its \bar{KM}	speedup S	reduction factor κ	Newton its \bar{KM}	speedup S	reduction factor κ
Galerkin	8.64×10^{-12}	99	1.01	1.0	2	1.84	49.5
Galerkin + Gappy POD	8.64×10^{-12}	99	36.4	1.0	2	69.3	49.5
Galerkin + collocation	2.12×10^{-5}	100	36.5	0.99	16	61.9	6.25

Table 1: Limiting case: forecast performance

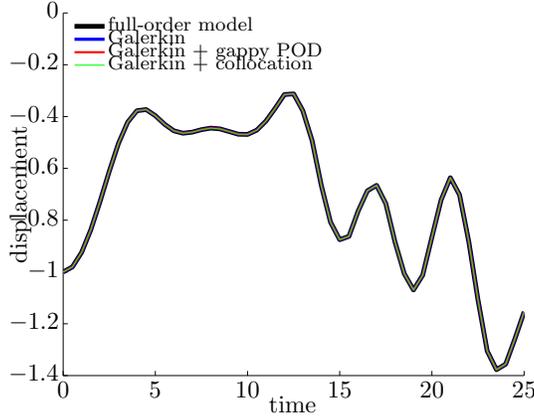


Figure 3: Limiting case: all ROMs generate near-zero error

collect snapshot in the offline stage $D_{\text{train}} = \{p(q^{\text{sample},k}), q^{\text{sample},k}\}_{k=1}^6$; then, we deploy the ROMs on two randomly chosen online configurations $q^{*,1}$ and $q^{*,2}$. Table 2 reports the values of the parametric inputs for these configurations.

configuration	q_1	q_2	q_3	q_4	q_5	q_6
$q^{\text{sample},1}$	-0.2999	-0.9735	0.2374	0.5872	0.9248	0.4786
$q^{\text{sample},2}$	0.0660	0.3187	-0.1555	-0.8393	0.6152	0.3911
$q^{\text{sample},3}$	0.2395	-0.6085	0.2981	-0.8126	0.6181	-0.3415
$q^{\text{sample},4}$	-0.0033	0.3666	-0.5749	-0.4519	0.2338	0.0790
$q^{\text{sample},5}$	-0.5650	-0.9869	0.5075	0.2002	0.8572	0.8681
$q^{\text{sample},6}$	0.9286	-0.8243	-0.4919	-0.9116	-0.7414	0.3112
$q^{*,1}$	-0.8722	-0.3269	-0.5777	-0.4507	0.3181	-0.1177
$q^{*,2}$	0.0059	-0.2022	-0.6903	0.7106	0.7589	0.8779

Table 2: Unforced dynamics, varying structure: parametric inputs. Note that $q_i = -1$ for $i = 7, 8, 9$, which sets the external-force magnitudes to zero.

This experiment employs truncation criteria of $\nu_x = \nu_R = 0.9999$ for the reduced bases. Again, the forecasting technique employs $\tau = 0$ and $\alpha_{\text{max}} = 12$. Figure 4 and Table 3 report the results for this experiment. First, note that all ROMs generate very small relative errors ($\varepsilon < 10^{-2}$) even though a mere 6 training points were (randomly) selected in a six-dimensional input parameter space. This strong performance can be attributed to the relative smooth dynamics characterizing the problem. Again, we observe that spatial-complexity reduction is necessary to generate significant speedups (the Galerkin ROM has a speedup of roughly one). We also note that—in the absence of the proposed

forecasting technique—the ROMs exhibit no temporal-complexity reduction ($\kappa = 1.0$).

When the models employ the proposed forecasting technique, the number of total Newton steps is nearly cut in half, as all methods generate temporal-complexity reduction factors of $\kappa = 1.71$. This also leads to a significant improvement in speedup for all reduced-order models. These results indicate that the proposed technique is effective in realistic scenarios where the reduced bases are truncated and the inputs vary.

To gain insight into the method’s potential, Figure 5 depicts the time evolution of the first generalized unknown \hat{w}_1 for the online and training inputs; note that this is one of the forecasted variables. The online inputs lead to different frequency content of the generalized unknown’s temporal behavior, even though the qualitative behavior of the response is similar. The forecasting method performs well in spite of this frequency shift. This indicates that the method is reasonably robust with respect to frequency shifts in the system’s temporal response.

Online inputs	ROM method	relative error ε	No forecasting			With forecasting		
			Newton its	speedup S	reduction factor κ	Newton its	speedup S	reduction factor κ
$q^{*,1}$	Galerkin	2.04×10^{-3}	82	0.998	1.00	48	1.37	1.71
	Gal. + Gappy	1.77×10^{-3}	82	58.2	1.00	48	82.4	1.71
	Gal. + coll.	1.83×10^{-3}	82	59.4	1.00	48	80.8	1.71
$q^{*,2}$	Galerkin	3.86×10^{-3}	82	1.03	1.00	48	1.55	1.71
	Gal. + Gappy	3.64×10^{-3}	82	54	1.00	48	86.1	1.71
	Gal. + coll.	3.19×10^{-3}	82	55.5	1.00	48	88.7	1.71

Table 3: Unforced dynamics: forecast performance

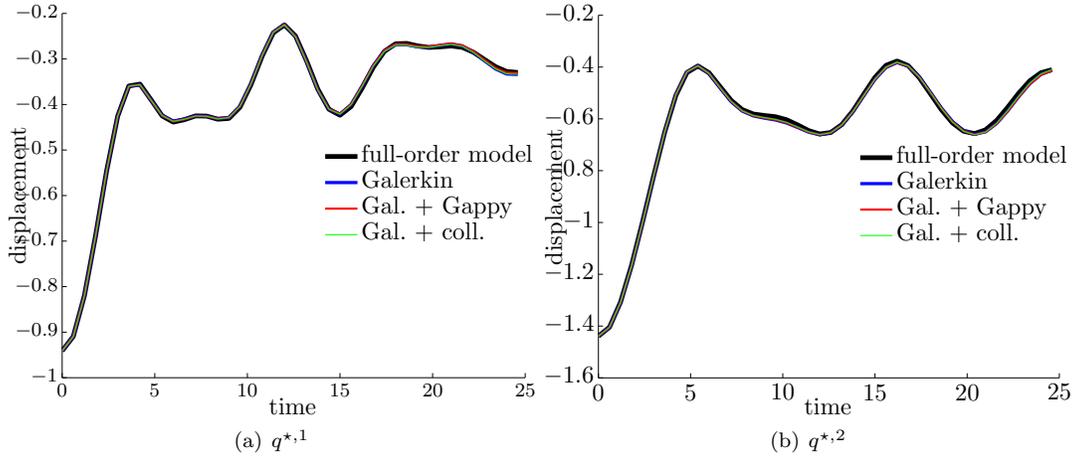


Figure 4: Unforced dynamics: responses generated by reduced-order models

4.4. Forced dynamics, fixed structure

This experiment analyzes the method for the problem in the absence of shape and material-property changes, but with external forcing and changes in the initial condition. Because the external forces are of relatively high frequency (close to the largest natural frequency of the structure), we expect this problem to be characterized by ‘rich’ dynamics: the first half of the time interval is a free-vibration

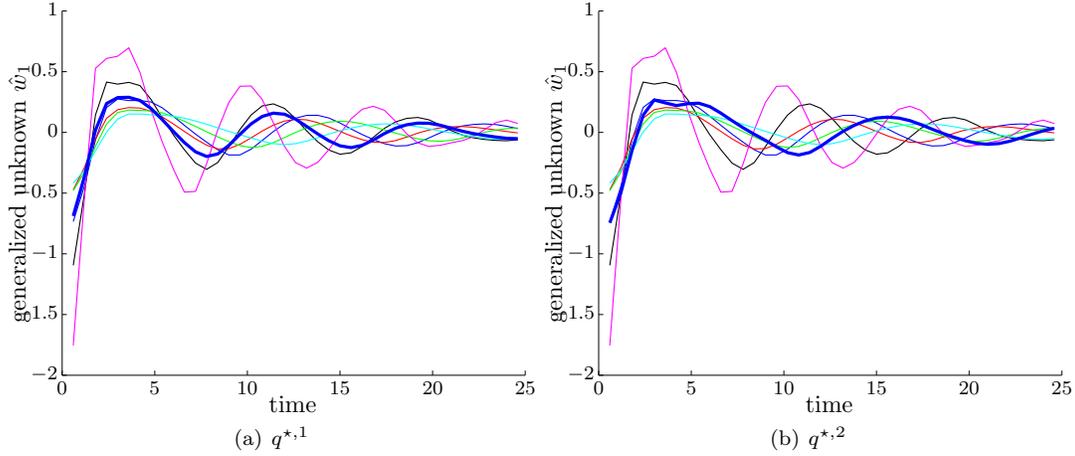


Figure 5: Unforced dynamics. First generalized unknown at online inputs (bold curve) and training inputs (thin curves)

problem, while the second half of the time interval is a high-frequency forced response. As before, we randomly choose six training configurations and two online configurations; Table 4 reports the associated parametric inputs.

configuration	q_6	q_7	q_8	q_9
$q^{\text{sample},1}$	0.4786	0.0001	0.4019	-0.9561
$q^{\text{sample},2}$	0.3911	-0.2747	0.7756	0.4022
$q^{\text{sample},3}$	-0.3415	-0.4041	0.7287	0.4164
$q^{\text{sample},4}$	0.0790	-0.2275	-0.4506	0.4536
$q^{\text{sample},5}$	0.8681	0.9115	0.6784	0.2574
$q^{\text{sample},6}$	0.3112	0.9821	0.8428	-0.1354
$q^{*,1}$	-0.1177	0.5200	-0.4993	-0.2177
$q^{*,2}$	0.8779	0.1817	-0.2161	-0.3058

Table 4: Forced dynamics, fixed structure: parametric inputs. Note that $q_i = 0$ for $i = 1, \dots, 5$, which causes shape and material-property inputs to be fixed.

Again, we employ truncation criteria of $\nu_x = \nu_R = 0.9999$ and forecasting parameters $\tau = 0$ and $\alpha_{\max} = 12$. Figure 6 and Table 5 report the results for this experiment.

Similar results to the experiment in Section 4.3 can be observed. All relative errors are reasonably small (below 2×10^{-2}), although the responses deviate from the full-order model around $t = 20$. Also, the proposed forecasting technique significantly improves the temporal-complexity reduction factor κ and leads to improvements in speedups. However, the performance of the proposed technique is not quite as strong as in the unforced case (compare Tables 3 and 5). This is likely attributable to the presence of richer dynamics in the present experiment. Figure 7 depicts this: the temporal behavior of the first generalized coordinate is much less smooth than in the unforced case. However, because the temporal behavior remains qualitatively similar when the inputs are varied, the forecasting method can effectively exploit the training data to generate an accurate forecast. This highlights one advantage of the proposed method: it is relatively insensitive to the smoothness of the temporal response. Instead, it depends much more strongly on how this response is affected by input variation.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

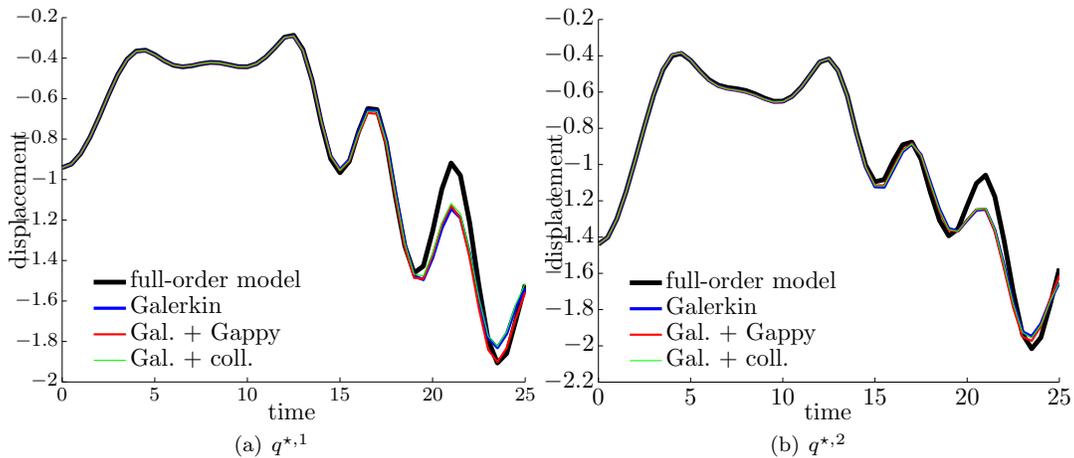


Figure 6: Forced dynamics, fixed structure: responses generated by reduced-order models

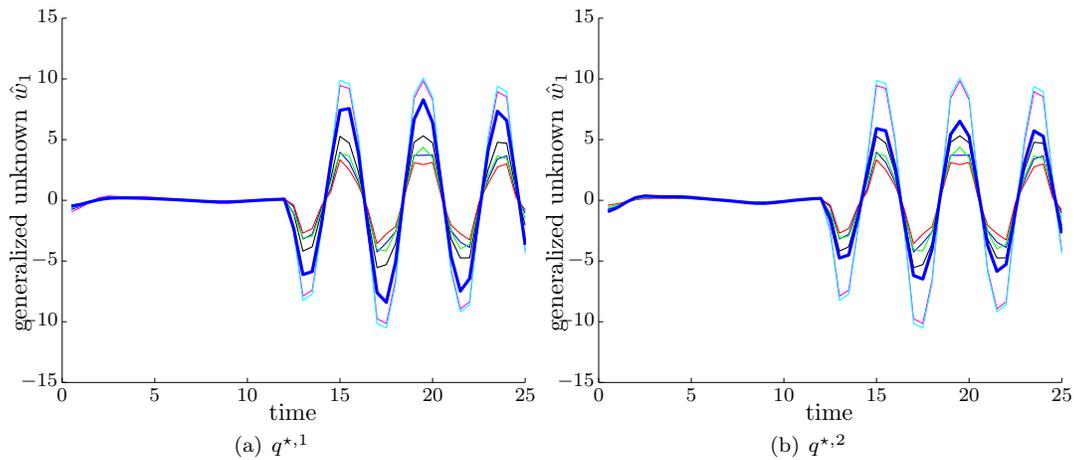


Figure 7: Forced dynamics, fixed structure. First generalized unknown at online inputs (bold curve) and training inputs (thin curves)

Online inputs	ROM method	relative error ε	No forecasting			With forecasting		
			Newton its	speedup S	reduction factor κ	Newton its	speedup S	reduction factor κ
$q^{*,1}$	Galerkin	1.51×10^{-2}	100	1.02	1.00	60	1.47	1.67
	Gal. + Gappy	1.39×10^{-2}	100	52.7	1.00	61	75.2	1.64
	Gal. + coll.	1.38×10^{-2}	100	49.5	1.00	71	70.9	1.41
$q^{*,2}$	Galerkin	1.50×10^{-2}	100	1.02	1.00	60	1.52	1.67
	Gal. + Gappy	1.40×10^{-2}	100	52.1	1.00	61	73.2	1.64
	Gal. + coll.	1.41×10^{-2}	100	54.3	1.00	68	71.8	1.47

Table 5: Forced dynamics, fixed structure: forecast performance

4.5. Forced dynamics, varying structure

This experiment pushes the boundaries of the method further. Here, we consider both external forces and variations in the structure’s shape and material properties. As a result, we expect rich dynamics and a larger variation in the system’s dynamics in the input space. Table 6 reports the randomly-chosen parametric inputs for this experiment.

configuration	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
$q^{\text{sample},1}$	-0.2999	-0.9735	0.2374	0.5872	0.9248	0.4786	0.0001	0.4019	-0.9561
$q^{\text{sample},2}$	0.0660	0.3187	-0.1555	-0.8393	0.6152	0.3911	-0.2747	0.7756	0.4022
$q^{\text{sample},3}$	0.2395	-0.6085	0.2981	-0.8126	0.6181	-0.3415	-0.4041	0.7287	0.4164
$q^{\text{sample},4}$	-0.0033	0.3666	-0.5749	-0.4519	0.2338	0.0790	-0.2275	-0.4506	0.4536
$q^{\text{sample},5}$	-0.5650	-0.9869	0.5075	0.2002	0.8572	0.8681	0.9115	0.6784	0.2574
$q^{\text{sample},6}$	0.9286	-0.8243	-0.4919	-0.9116	-0.7414	0.3112	0.9821	0.8428	-0.1354
$q^{*,1}$	-0.8722	-0.3269	-0.5777	-0.4507	0.3181	-0.1177	0.5200	-0.4993	-0.2177
$q^{*,2}$	0.0059	-0.2022	-0.6903	0.7106	0.7589	0.8779	0.1817	-0.2161	-0.3058

Table 6: Forced dynamics, varying structure: parametric inputs

As before, we employ the following parameters: $\nu_x = \nu_R = 0.9999$, $\tau = 0$, and $\alpha_{\max} = 12$. Figure 8 and Table 7 report the results for this experiment.

For these experiments, we observe that the relative errors generated by the reduced-order models are significantly larger than in the previous experiments. This illustrates that the inputs are inducing a wider variety of dynamics, which reduced-order models have difficulty capturing. For this reason, we expect the proposed forecasting method to also perform worse, as the temporal behavior is (likely) also more difficult to predict. This is certainly the case: the temporal-complexity-reduction factors κ are lower than in previous experiments. Nonetheless, the proposed technique results in significantly improved performance in all but one case. The only exception is for the Galerkin ROM applied to $q^{*,1}$; here, the forecasting technique decreases κ from 1.39 (without forecasting) to 1.33. This illustrates that the forecasting method can lead to degraded performance if the forecast is inaccurate; this can occur if the temporal behavior of the response is sufficiently rich and varies significantly with parameter variation.

Thus, as expected, the method appears to perform best when the dynamics of the problem are relatively smooth and the temporal behavior does not drastically change as inputs vary.

4.6. Parameter study: maximum memory α_{\max} and Newton-step threshold τ

In this experiment, we perform a parameter study to determine the parameters α_{\max} and τ that lead to the best performance for the method. For this purpose, we run the experiments described

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

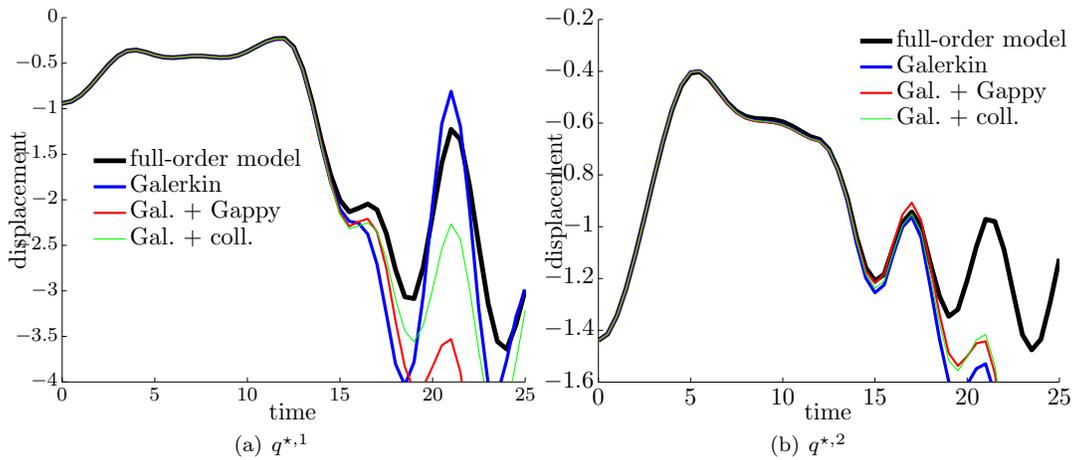


Figure 8: Forced dynamics, varying structure: responses generated by reduced-order models

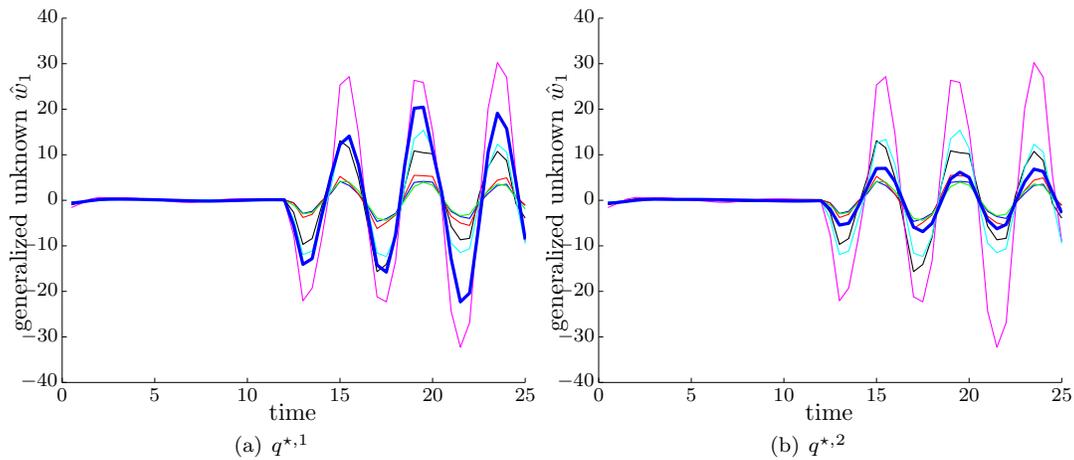


Figure 9: Forced dynamics, varying structure. First generalized unknown at online inputs (bold curve) and training inputs (thin curves)

Online inputs	ROM method	relative error ε	No forecasting			With forecasting		
			Newton its	speedup S	reduction factor κ	Newton its	speedup S	reduction factor κ
$q^{*,1}$	Galerkin	4.95×10^{-2}	104	1.21	1.39	109	1.38	1.33
	Gal. + Gappy	1.65×10^{-1}	124	8	1.17	94	9.48	1.54
	Gal. + coll.	6.93×10^{-2}	120	8.12	1.21	90	11	1.61
$q^{*,2}$	Galerkin	1.40×10^{-1}	95	1.04	1.02	62	1.47	1.56
	Gal. + Gappy	1.17×10^{-1}	95	7.47	1.02	64	10.4	1.52
	Gal. + coll.	1.21×10^{-1}	100	7.36	0.97	73	9.98	1.33

Table 7: Forced dynamics, varying structure: forecast performance

in Sections 4.3–4.5 for $\alpha_{\max} = 6, 9, 12, 15$ and $\tau = 0, 1$. Then, we compute \mathfrak{k} as the average value of $\kappa_{\text{forecast}}/\kappa_{\text{no}}$ over all experiments and all three reduced-order models. Here, κ_{forecast} designates the temporal-complexity-reduction factor achieved when the proposed forecasting method is used; κ_{no} denotes the temporal-complexity-reduction factor obtained without forecasting. Similarly, \mathfrak{s} denotes the average value of $S_{\text{forecast}}/S_{\text{no}}$ over all experiments and reduced-order models. Here, S_{forecast} denotes the wall-time speedup obtained when the proposed method is used; S_{no} is the speedup when the method is not employed.

Figure 10 reports the results for the parameter study. It is evident that the parameters that lead to best performance for this problem in both temporal-complexity-reduction improvement (\mathfrak{k}) and speedup improvement (\mathfrak{s}) are $\tau = 0$ and $\alpha_{\max} = 12$. Interestingly, $\alpha_{\max} = 6$ yields the worst performance in temporal-complexity reduction. This choice corresponds to interpolation in the Gappy POD forecast, as the forecast employs six basis vectors. This observation is consistent with those reported in Refs. [26, 4]: interpolation when applied with Gappy POD rarely leads to the best performance in reduced-order modeling.

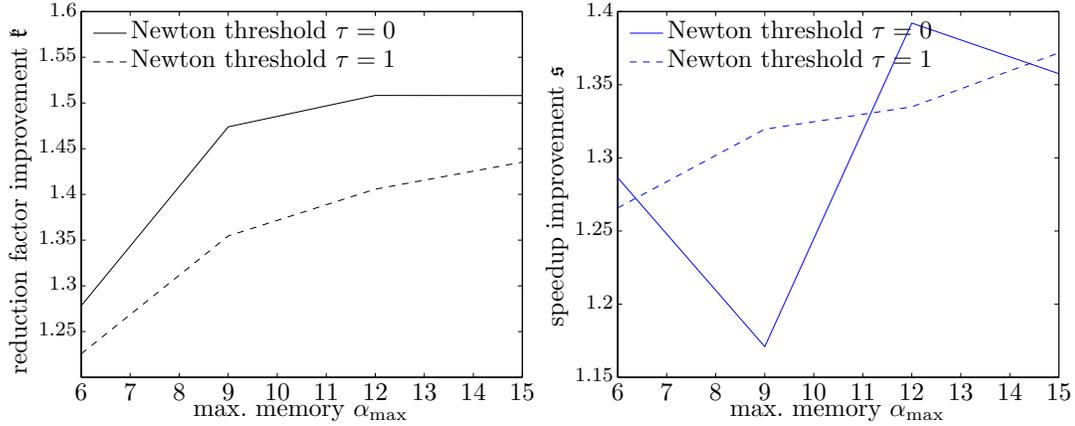


Figure 10: Parameter study: performance averaged over all experiments and reduced-order models

4.7. Parameter study: dimension of trial subspace \hat{N}

This experiment analyzes the performance of the proposed technique as a function of trial-subspace dimension \hat{N} . We consider the experimental setup in Section 4.3, and run the experiment for $\nu_x =$

$\nu_R = 0.9, 0.99, 0.999, 0.9999, 0.99999, 0.999999$. The temporal-complexity method employs $\tau = 0$ and $\alpha_{\max} = 12$.

Figure 11 reports the results. These results indicate that as the trial-subspace dimension increases, the error in the reduced-order models' responses decreases monotonically and the performance of the temporal-complexity-reduction method improves, plateauing at $\kappa \approx 1.7$. This can be explained as follows: when the trial-subspace dimension increases, the ROMs become more accurate, so they generate responses closer to that of the full-order model. Because the time-evolution bases are generated from full-order-model training simulations, a more accurate ROM implies a more accurate forecast. In turn, an accurate forecast leads to a greater reduction in temporal complexity.

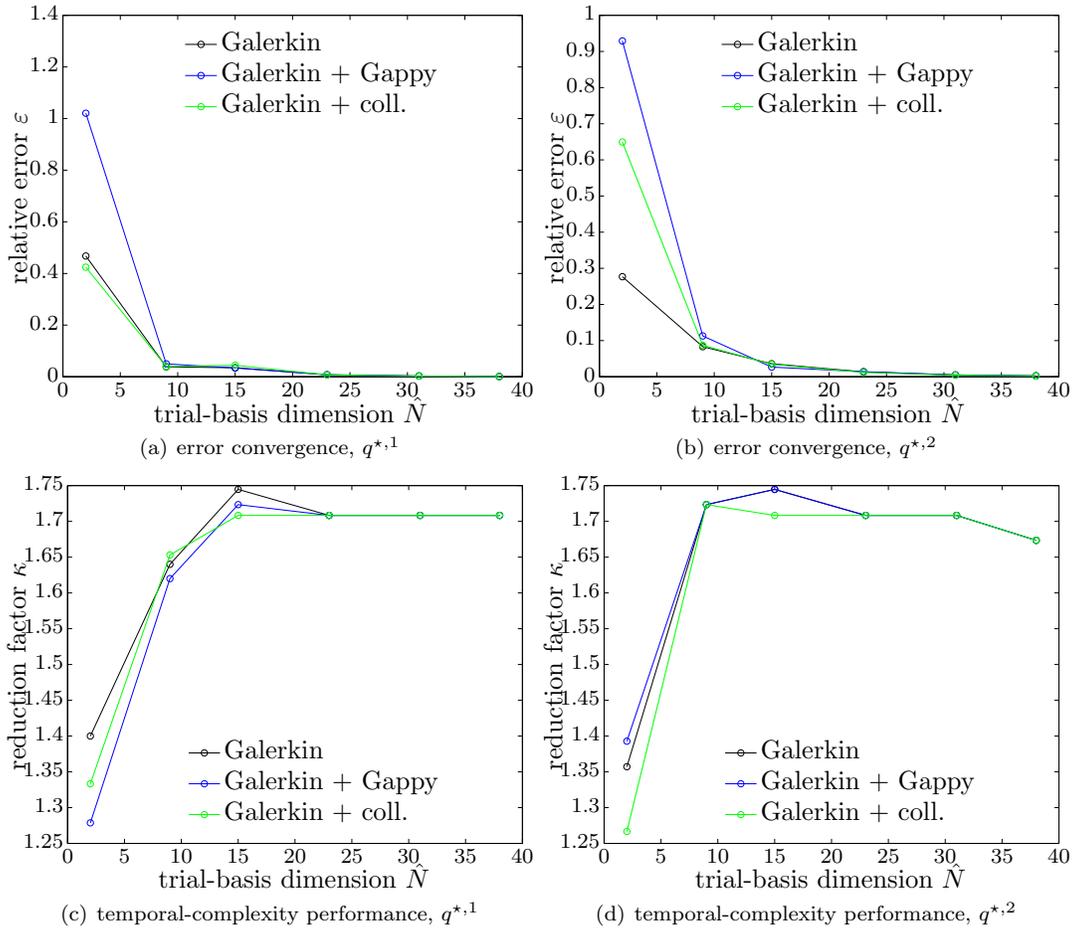


Figure 11: Unforced dynamics, varying structure. Effect of trial-basis dimension \hat{N} on error and temporal-complexity-reduction method performance.

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 and trial-subspace dimension 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.

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 q , and define $\bar{f}(x, t) \equiv f(x; t, p(t))$ such that

$$\dot{x} = \bar{f}(x, t). \quad (\text{A.1})$$

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

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

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

$$R^n(w^n) = \alpha_0 w^n - h \beta_0 \bar{f}(w^n, t^n) + \sum_{j=1}^k \alpha_j x^{n-j} - h \sum_{j=1}^k \beta_j \bar{f}(x^{n-j}, t^{n-j}) \quad (\text{A.3})$$

and the explicit state update is simply

$$x^n = w^n. \quad (\text{A.4})$$

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

1
2
3
4
5
6
7 *Appendix A.2. Implicit Runge–Kutta schemes*

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

9
10
11
$$R_i^n(w^{n,1}, \dots, w^{n,s}) = w^{n,i} - \bar{f}(x^{n-1} + h \sum_{j=1}^s a_{ij} w^{n,i}, t^{n-1} + c_i h), \quad i = 1, \dots, s \quad (\text{A.5})$$

12 with the following explicit computation of the state:

13
14
15
$$x^n = x^{n-1} + h \sum_{i=1}^s b_i w^{n,i}. \quad (\text{A.6})$$

16
17
18 The unknowns correspond to the velocity \dot{x} at times $t^{n-1} + c_i h$, $i = 1, \dots, s$.

19
20
21 **Appendix B. Implicit time-integration schemes: second-order ODEs**

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

24
25
$$\ddot{x} = \bar{g}(x, \dot{x}, t). \quad (\text{B.1})$$

26
27 *Appendix B.1. Implicit Nyström method*

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

30
31
32
$$R_i^n(w^{n,1}, \dots, w^{n,s}) = w^{n,i} - \bar{g} \left(x^{n-1} + c_i h \dot{x}^{n-1} + h^2 \sum_{j=1}^s \bar{a}_{ij} w^{n,i}, \dot{x}^{n-1} + h \sum_{j=1}^s \hat{a}_{ij} w^{n,i}, t^{n-1} + c_i h \right), \quad i = 1, \dots, s. \quad (\text{B.2})$$

33
34
35 The state and velocity are updated explicitly as

36
37
$$x^n = x^{n-1} + h \dot{x}^{n-1} + h^2 \sum_{i=1}^s \bar{b}_i w^{n,i} \quad (\text{B.3})$$

38
39
40
$$\dot{x}^n = \dot{x}^{n-1} + h \sum_{i=1}^s \hat{b}_i w^{n,i}. \quad (\text{B.4})$$

41
42
43 The unknowns correspond to the acceleration \ddot{x} at times $t^{n-1} + c_i h$, $i = 1, \dots, s$.

44
45 *Appendix B.2. Implicit Newmark method*

46 The implicit Newmark method leads to the following residuals:

47
48
$$R^n(w^n) = w^n - \bar{g} \left(x^{n-1} + h \dot{x}^{n-1} + \frac{h^2}{2} [(1 - 2\beta) \ddot{x}^{n-1} + 2\beta w^n], \dot{x}^{n-1} + h [(1 - \gamma) \ddot{x}^{n-1} + \gamma w^n], t^n \right) \quad (\text{B.5})$$

49
50
51 The state and velocity are explicitly updated as

52
53
54
$$x^n = x^{n-1} + h \dot{x}^{n-1} + \frac{h^2}{2} [(1 - 2\beta) \ddot{x}^{n-1} + 2\beta w^n] \quad (\text{B.6})$$

55
56
$$\dot{x}^n = \dot{x}^{n-1} + h [(1 - \gamma) \ddot{x}^{n-1} + \gamma w^n]. \quad (\text{B.7})$$

57
58 Here, the unknown corresponds to the acceleration \ddot{x} at time t^n .

1
2
3
4
5
6
7 **Appendix C. Proper orthogonal decomposition**

8 Algorithm 2 describes the method for computing a proper-orthogonal-decomposition (POD) basis
9 given a set of snapshots. The method essentially amounts to computing the singular value decompo-
10 sition of the snapshot matrix. The left singular vectors define the POD basis.
11

12 **Algorithm 2** Proper-orthogonal-decomposition basis computation (normalized snapshots)
13

14 **Input:** Set of snapshots $\mathcal{X} \equiv \{w_i\}_{i=1}^{n_w} \subset \mathbb{R}^N$, energy criterion $\nu \in [0, 1]$

15 **Output:** $\Phi^e(\mathcal{X}, \nu)$

- 16 1: Compute thin singular value decomposition $W = U\Sigma V^T$, where $W \equiv [w_1/\|w_1\| \cdots w_{n_w}/\|w_{n_w}\|]$.
17 2: Choose dimension of truncated basis $\hat{N} = n_e(\nu)$, where

$$18 \quad n_e(\nu) \equiv \arg \min_{i \in \mathcal{V}(\nu)} i \quad (\text{C.1})$$

$$19 \quad \mathcal{V}(\nu) \equiv \{n \in \{1, \dots, n_w\} \mid \sum_{i=1}^n \sigma_i^2 / \sum_{i=1}^{n_w} \sigma_i^2 \geq \nu\}, \quad (\text{C.2})$$

20 and $\Sigma \equiv \text{diag}(\sigma_i)$.

- 21 3: $\Phi^e(\mathcal{X}, \nu) = [u_1 \cdots u_{\hat{N}}]$, where $U \equiv [u_1 \cdots u_{n_w}]$.
-

22
23
24
25
26
27
28
29 **References**

- 30
31 [1] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An ‘empirical interpolation’ method:
32 application to efficient reduced-basis discretization of partial differential equations, *Comptes*
33 *Rendus Mathématique Académie des Sciences* 339 (2004) 667–672.
34
35 [2] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation,
36 *SIAM Journal on Scientific Computing* 32 (2010) 2737–2764.
37
38 [3] R. Everson, L. Sirovich, Karhunen–Loève procedure for gappy data, *Journal of the Optical Society*
39 *of America A* (1995) 1657–1664.
40
41 [4] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The GNAT method for nonlinear model re-
42 duction: effective implementation and application to computational fluid dynamics and turbulent
43 flows, arXiv e-print (2012).
44
45 [5] P. Diggle, *Time Series: A Biostatistical Introduction*, Clarendon Press, 1990.
46
47 [6] C. Gouieroux, *ARCH models and financial applications*, Springer-Verlag, 1997.
48
49 [7] R. F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of U. K.
50 inflation, *Econometrica* 50 (1982) 987–1008.
51
52 [8] F. A. Graybill, *An introduction to linear statistical models*, McGraw-Hill, New York, 1961.
53
54 [9] M. Hollander, D. A. Wolfe, *Nonparametric statistical methods*, Wiley, New York, 1973.
55
56 [10] D. B. Percival, A. T. Walden, *Spectral Analysis for Physical Applications*, Cambridge University
57 Press, 1993.
58
59 [11] C. C. Holt, Forecasting seasonals and trends by exponentially weighted moving averages, *Inter-*
60 *national Journal of Forecasting* 20 (2004) 5–10.
61
62
63
64
65

- 1
2
3
4
5
6
7 [12] P. R. Winters, Forecasting sales by exponentially weighted moving averages, *Management Science* 6 (1960) 324–342.
8
9
10 [13] P. Brown, G. Byrne, A. Hindmarsh, VODE: A variable-coefficient ODE solver, *SIAM Journal on*
11 *Scientific and Statistical Computing* 10 (1989) 1038–1051.
12
13 [14] J. Nievergelt, Parallel methods for integrating ordinary differential equations, *Communications*
14 *of the ACM* 7 (1964) 731–733.
15 [15] M. J. Gander, A waveform relaxation algorithm with overlapping splitting for reaction diffusion
16 equations, *Numerical Linear Algebra with Applications* 1 (1993).
17
18 [16] G. Horton, S. Vandewalle, A space-time multigrid methods for parabolic partial differential
19 equaions, *SIAM J. Sci. Comput.* 16 (1995) 848–864.
20
21 [17] J.-L. Lions, Y. Maday, G. Turinici, A parareal in time discretization of pdes., *C.R. Acad. Sci.*
22 *Paris, Serie I* (2001).
23
24 [18] J. Cortial, Time-parallel methods for accelerating the solution of structural dynamics problems,
25 *Stanford University*, 2011.
26
27 [19] C. Farhat, M. Chandesris, Time-decomposed parallel time-integrators: theory and feasibility
28 studies for fluid, structure, and fluid-structure applications, *International Journal for Numerical*
29 *Methods in Engineering* 58 (2003) 1397–1434.
30
31 [20] C. Harden, Real time computing with the parareal algorithm (2008).
32
33 [21] R. Bos, X. Bombois, P. Van den Hof, Accelerating large-scale non-linear models for monitoring
34 and control using spatial and temporal correlations, in: *Proceedings of the American Control*
35 *Conference*, volume 4, pp. 3705–3710.
36
37 [22] D. Ryckelynck, A priori hyperreduction method: an adaptive approach, *Journal of Computational*
38 *Physics* 202 (2005) 346–366.
39
40 [23] T. Kim, D. James, Skipping steps in deformable simulation with online model reduction.
41
42 [24] E. Hairer, G. Wanner, *Solving ordinary differential equations II: stiff and differential-algebraic*
43 *problems*, Springer Verlag, 2002.
44
45 [25] P. Krysl, S. Lall, J. E. Marsden, Dimensional model reduction in non-linear finite elements
46 dynamics of solids and structures, *Int. J. Numer. Meth. Engng* 51 (2001) 479–504.
47
48 [26] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares
49 Petrov–Galerkin projection and compressive tensor approximations, *International Journal for*
50 *Numerical Methods in Engineering* 86 (2011) 155–181.
51
52 [27] T. Bui-Thanh, K. Willcox, O. Ghattas, Model reduction for large-scale systems with high-
53 dimensional parametric input space, *SIAM Journal on Scientific Computing* 30 (2008) 3270–3288.
54
55 [28] T. Bui-Thanh, K. Willcox, O. Ghattas, Parametric reduced-order models for probabilistic analysis
56 of unsteady aerodynamic applications, *AIAA Journal* 46 (2008).
57
58 [29] P. A. LeGresley, Application of Proper Orthogonal Decomposition (POD) to Design Decomposi-
59 tion Methods, Ph.D. thesis, Stanford University, 2006.
60
61
62
63
64
65

- 1
2
3
4
5
6
7 [30] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by
8 proper orthogonal decomposition, *IEEE Transactions on Automatic Control* 53 (2008) 2237–2251.
9
10 [31] D. Galbally, K. Fidkowski, K. Willcox, O. Ghattas, Non-linear model reduction for uncertainty
11 quantification in large-scale inverse problems, *International Journal for Numerical Methods in*
12 *Engineering* (2009).
13
14 [32] M. Drogmann, B. Haasdonk, M. Ohlberger, Reduced basis approximation for nonlinear parame-
15 terized evolution equations based on empirical operator interpolation, *SIAM Journal on Scientific*
16 *Computing* (2012).
17
18 [33] T. Bui-Thanh, D. Murali, K. Willcox, Proper orthogonal decomposition extensions for parametric
19 applications in compressible aerodynamics, *AIAA Paper 2003-4213*, 21st Applied Aerodynamics
20 Conference, Orlando, FL (2003).
21
22 [34] T. Bui-Thanh, M. Damodaran, K. Willcox, Aerodynamic data reconstruction and inverse design
23 using proper orthogonal decomposition, *AIAA Journal* 42 (2004) 1505–1516.
24
25 [35] D. Venturi, G. E. Karniadakis, Gappy data and reconstruction procedures for flow past a cylinder,
26 *Journal of Fluid Mechanics* 519 (2004) 315–336.
27
28 [36] K. Willcox, Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition,
29 *Computers and Fluids* 35 (2006) 208–226.
30
31 [37] T. D. Robinson, M. S. Eldred, K. Willcox, R. Haimes, Strategies for multifidelity op-
32 timization with variable dimensional hierarchical models, *AIAA Paper 2006-1819*, 47th
33 AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference,
34 Newport, RI (2006).
35
36 [38] I. Chowdhury, S. Dasgupta, Computation of Rayleigh damping coefficients for large systems, *The*
37 *Electronic Journal of Geotechnical Engineering* 8 (2003).
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65