INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2017; **109**:198–217
Published online 14 July 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.5283

Limited-memory adaptive snapshot selection for proper orthogonal decomposition

Geoffrey M. Oxberry[1,*,†], Tanya Kostova-Vassilevska[2], William Arrighi[3] and Kyle Chand[4]

[1]*Computational Engineering Division, Lawrence Livermore National Laboratory, L-792, PO Box 808, Livermore, CA 94550, USA*
[2]*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-561, PO Box 808, Livermore, CA 94550, USA*
[3]*Applications, Simulations, and Quality, Lawrence Livermore National Laboratory, L-560, PO Box 808, Livermore, CA 94550, USA*
[4]*Global Security Computing Applications Division, Lawrence Livermore National Laboratory, L-389, PO Box 808, Livermore, CA 94550, USA*

SUMMARY

Reduced order models are useful for accelerating simulations in many-query contexts, such as optimization, uncertainty quantification, and sensitivity analysis. However, offline training of reduced order models (ROMs) can have prohibitively expensive memory and floating-point operation costs in high-performance computing applications, where memory per core is limited. To overcome this limitation for proper orthogonal decomposition, we propose a novel adaptive selection method for snapshots in time that limits offline training costs by selecting snapshots according an error control mechanism similar to that found in adaptive time-stepping ordinary differential equation solvers. The error estimator used in this work is related to theory bounding the approximation error in time of proper orthogonal decomposition-based ROMs, and memory usage is minimized by computing the singular value decomposition using a single-pass incremental algorithm. Results for a viscous Burgers' test problem demonstrate convergence in the limit as the algorithm error tolerances go to zero; in this limit, the full-order model is recovered to within discretization error. A parallel version of the resulting method can be used on supercomputers to generate proper orthogonal decomposition-based ROMs, or as a subroutine within hyperreduction algorithms that require taking snapshots in time, or within greedy algorithms for sampling parameter space. Copyright © 2016 John Wiley & Sons, Ltd.

Received 26 October 2015; Revised 16 March 2016; Accepted 15 April 2016

KEY WORDS: proper orthogonal decomposition; reduced order model; snapshot; incremental singular value decomposition

1. INTRODUCTION

This paper describes a method for constructing proper orthogonal decomposition-based reduced order models (ROMs) from simulations of transient full order models (FOMs), ignoring parameter dependence. The proposed method reduces the offline costs of ROM snapshot selection and basis computation in both memory and floating point operations relative to existing methods in the literature. It is eventually intended to be used in constructing ROMs from large-scale FOM simulations run on supercomputers; in this work, we discuss a serial version of the algorithm to demonstrate its convergence properties. With that goal in mind, it minimizes memory requirements for POD algorithms. It can be used as a component of a procedure for selecting snapshots from simulations

*Correspondence to: Geoffrey M. Oxberry, Computational Engineering Division, Lawrence Livermore National Laboratory, L-792, PO Box 808, Livermore, CA 94550, USA.
†E-mail: oxberry1@llnl.gov

of transient FOMs with parameters (for example, the greedy approach in [1]) in order to build POD ROMs for applications such as optimization [2], uncertainty quantification, sensitivity analysis, parameter studies, and design of (computational) experiments. To motivate the need for such an algorithm, a brief survey of POD snapshot selection methods is presented; for detailed descriptions of POD, see [3–5].

1.1. Notation

In this document, scalar variables will be denoted in italics by either Roman or Greek letters. Vectors and vector-valued functions will be denoted by lowercase bold letters. Matrices will be denoted by uppercase bold letters. For a given matrix $\mathbf{A}$, $\mathcal{R}(\mathbf{A})$ denotes the range of $\mathbf{A}$. The two-norm of a vector is denoted by $\|\cdot\|$, and the infinity-norm of a vector is denoted by $\|\cdot\|_\infty$.

It will be convenient in several places to construct diagonal matrices from vectors; the function $\text{diag} : \mathbb{R}^m \to \mathbb{R}^{m\times m}$ takes as input a vector and outputs a square, diagonal matrix with that vector's entries on its main diagonal as follows: if $\mathbf{a} = (a_1, a_2, \ldots, a_m)^{\mathrm{T}}$, then

$$\text{diag}(\mathbf{a}) = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_m \end{bmatrix}.$$

1.2. Proper orthogonal decomposition

Consider the ordinary differential equation

$$\dot{\mathbf{u}}(t) = \mathbf{f}(t, \mathbf{u}(t)); \quad \mathbf{u}(0) = \mathbf{u}^0 \in \mathbb{R}^m \tag{1}$$

defining the FOM. The discrete version of POD [4] takes as input a matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$ defined by

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^1 - \bar{\mathbf{u}} & \mathbf{u}^2 - \bar{\mathbf{u}} & \ldots & \mathbf{u}^n - \bar{\mathbf{u}} \end{bmatrix}, \tag{2}$$

where $\{\mathbf{u}^j - \bar{\mathbf{u}}\}_{j=1}^n \subset \mathbb{R}^m$ are the snapshots, and $\bar{\mathbf{u}} \in \mathbb{R}^m$ is a time-independent offset. It is assumed in this paper that snapshots are to be collected from a numerical solution to the FOM defined by equation (1) and that the inner product used in POD is the standard Euclidean inner product on $\mathbb{R}^n$.

Let $\mathbf{U} = \mathbf{V}\text{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}}$ be an SVD of $\mathbf{U}$ such that the elements of $\mathbf{s}$ are arranged in nonincreasing order of magnitude. For given k, POD defines a basis matrix $\mathbf{B}$ consisting of the first k columns of $\mathbf{V}$ (i.e, the k left singular vectors corresponding to the k greatest singular values of $\mathbf{U}$) and uses this matrix to define a ROM via projection

$$\dot{\tilde{\mathbf{u}}}(t) = \mathbf{B}^{\mathrm{T}}\mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}); \quad \tilde{\mathbf{u}}(0) = \mathbf{B}^{\mathrm{T}}(\mathbf{u}^0 - \bar{\mathbf{u}}). \tag{3}$$

Upon solving equation (3), an approximation to the solution $\mathbf{u}$ of equation (1) is recovered via lifting:

$$\mathbf{v}(t) = \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}. \tag{4}$$

The motivation for using a ROM such as (3) is to approximate well the FOM solution $\mathbf{u}$ by the lifted ROM solution $\mathbf{v}$ through the assumption that $\mathbf{u}$ lies close (in some sense) to the affine subspace $\mathcal{R}(\mathbf{B}) + \bar{\mathbf{u}}$. The dimension k of the POD basis used to define the ROM is up to the user and is typically chosen to balance a trade-off between accuracy and computational costs. The offset $\bar{\mathbf{u}}$ can be interpreted as a base point of the affine subspace (linear manifold) $\mathcal{R}(\mathbf{B}) + \bar{\mathbf{u}}$ and is chosen to translate the ROM initial conditions. Common values are $\mathbf{0}$, the mean of the set $\{\mathbf{u}^j\}_{j=1}^n$, or the initial condition $\mathbf{u}^0$.

The process of constructing a ROM is typically described as the 'offline' ROM training phase, in contrast to solving the ROM in what is called the 'online' phase. Until recently, most analysis of

DOI: 10.1002/nme

ROM computational costs neglected the cost of the offline stage and focused almost exclusively on the online stage, reasoning that offline ROM costs could be amortized for many-query applications over a sufficiently large number of inexpensive online ROM solves performed in place of expensive FOM solves. In practice, the amount of memory and computational time (or floating-point operations) for the offline training phase are constrained. These resource constraints must be balanced against controlling the ROM approximation error in the online phase. The remainder of the introduction discusses these three issues – memory costs, floating-point operation costs, and error control – followed by the contributions of this work.

1.3. Offline memory and floating-point operation costs

The SVD algorithm used to compute the basis from the selected snapshots is usually the dominant contribution to the memory costs of POD because typical approaches to computing an SVD (*e.g*, [6, 7]) require storing the entire snapshot matrix at once. Recently, Paul-Dubois-Taine and Amsallem ([8], Section 3.2.5) propose two different methods for approximating the SVD by operating on blocks of snapshots, but these methods do not update directly the SVD of the original snapshot matrix. Single-pass incremental SVD methods [9–11] improve upon these approaches by updating an existing truncated SVD as new snapshots are collected, and these approaches have been adopted in recently developed ROM algorithms [12, 13]. Over a single pass of the FOM solution data, incremental SVD algorithms require the least memory and floating-point operations (in an asymptotic sense) of all deterministic algorithms available for computing the POD basis; a more detailed analysis of this point will be presented in Section 2.

The manner in which snapshots are selected also contributes to ROM training costs; for all but the simplest methods used, this dominates the floating-point operation cost of the computing the POD basis. When gathering snapshots for transient models (ignoring parameter dependence), such as the FOM in equation (1), the simplest strategy is to collect a snapshot at every time step of a computed solution and set $\bar{\mathbf{u}} = \mathbf{u}^0$, as suggested in [14] and by Carlberg. This approach has zero floating-point operation cost and useful numerical properties (as k approaches m, ROM error approaches zero), but could gather prohibitively large numbers of snapshots. A typical compromise collects snapshots at every j simulation time steps for some natural number j, as in [15], which is also computationally cheap, but lacks error control. More sophisticated approaches to control ROM error due to snapshot selection solve optimization problems that require a FOM solve at each iteration [16–18], and are prohibitively expensive in floating-point operations and memory. Greedy approaches to controlling ROM error due to snapshot selection are expensive either in memory (because they require storing the entire numerical solution of the FOM [19]), or in floating-point operations (because they recompute parts of the FOM solution instead of storing it [20]). *A priori* hyperreduction (APHR) approaches are similarly expensive because these approaches recompute parts of the FOM solution [21, 22].

1.4. Contributions

The novel approach presented in this paper instead selects snapshots on-the-fly during the solution of the FOM. In contrast to greedy methods [20], this approach does not require trade-offs between storing the entire FOM numerical solution and recomputing the FOM to achieve error control. Rather, the approach in this paper requires computing the FOM solution only once over the time interval of interest and computes the POD basis *in situ*, storing the minimum snapshot information needed for POD ROM construction.

Snapshots are selected adaptively in time by calculating the time step between snapshots using methods borrowed from adaptive time stepping for ODEs. In the adaptive snapshot time stepping calculation, a low-cost error estimator is used that places no restrictions on the form of the FOM, so long as it is an ODE. This flexibility contrasts with existing methods that assume the FOM is a semi-discretized PDE using a specific spatial discretization (e.g., finite element [21–24] and finite volume [19]). As each snapshot is selected, the POD basis is updated using a single-pass incremental SVD algorithm developed for streaming data analysis [9]. Consequently, only one column of the snapshot matrix $\mathbf{U}$ must be in memory at any given time, along with the current POD basis matrix $\mathbf{B}$.

DOI: 10.1002/nme

As suggested in [25] and [10], use of the incremental SVD for basis computation substantially reduces both memory requirements and the number of arithmetic operations performed by the SVD, independent of the snapshot selection criterion. As such, this basis updating approach is related to block SVD compression as in [8], but uses less memory because the 'blocks' in the proposed approach are of size 1 and the POD basis is truncated on-the-fly.

The proposed basis updating approach is also related to the basis extension method proposed by Haasdonk and Ohlberger for use in POD-Greedy approaches [19], which was developed for parameter-dependent transient problems, and does not consider selecting snapshots in time.

The proposed basis updating approach uses similar methods to the rank-1 incremental SVD updates for local ROMs [12]; the approach in this paper differs from that work in that the incremental SVD is applied to selecting FOM snapshots for global ROMs. In principle, the methods in this paper could be adapted to local ROMs; discussion of that scenario is deferred to future work.

Incremental SVD updates are also used in [13] on steady-state parameter-dependent problems; time-dependent problems are not considered, and ROM approximation error is computed using an approach that would require recomputing the FOM solution in the time-dependent case.

The error estimator used in this paper is related to the wide variety of error estimation and error bounding approaches in the literature (such as [15, 21–24, 26–29]). The basic idea of snapshot selection algorithm presented in this work is to equidistribute the estimated error in a manner similar to the snapshot selection approach developed by Hoppe and Liu, [18] but without resorting to solving an expensive optimization problem in both memory and arithmetic operations.

A comparison of adaptive snapshot selection to optimization-type methods for snapshot selection is not considered in this article. These methods have considerably higher theoretical cost and compute snapshot locations that minimize the error between the ROM and FOM solutions, so the ROM error from these methods could be less than the ROM error attained using adaptive snapshot selection. This theoretical analysis already yields an informative qualitative comparison of methods. We believe a concrete comparison would not provide significantly more information in the context of this article relative to the substantial amount of work involved in implementing these methods.

To the authors' knowledge, this paper is the first work to propose an inexpensive adaptive method with error control for selecting POD snapshots on-the-fly in time, using a single pass of an FOM solution. We emphasize that at no point does our algorithm require revisiting the FOM solution at earlier times, which we assume is an expensive operation either in terms of recomputing part of the FOM solution or storing tahe FOM solution in memory or on disk. Thus, our method contrasts with approaches by Ryckelynck [21]; Ryckelynck and Benziane [22]; and Dihlmann, Drohmann, and Haasdonk [20], all of which propose partially recomputing the FOM solution. In addition, it contrasts with [21] and [22] in that it uses a different model reduction method (POD instead of *a priori* hyperreduction), applies an SVD to state snapshots instead of residual snapshots, and uses estimates of the ROM error in the state variables instead of the error in a FOM residual.

Furthermore, to the authors' knowledge, it is the first work to use single-pass incremental SVD methods to append snapshots in time, dramatically reducing the memory footprint of the POD algorithm to make it more suitable for high-performance computing applications. An upcoming publication [30] describes a parallel version of the algorithm presented in this paper. Without an incremental SVD algorithm, adaptive snapshot selection on-the-fly in time becomes prohibitively expensive in the limit of a large number of snapshots collected because the only viable alternative is to compute a dense SVD after each snapshot.

The remainder of the paper proceeds as follows: Section 2 describes the incremental SVD algorithm by Brand used in the snapshot selection algorithm. Section 3 presents the error estimation arguments used in this work. Having presented the background necessary for the proposed snapshot selection algorithm, this algorithm is explained in Section 4. Section 5 illustrates the performance of the snapshot selection criterion using a viscous Burgers' equation as a case study and compares results with those obtained from an algorithm that samples at equispaced points in time. Finally, Section 6 concludes the paper with a summary and potential extensions of this work.

DOI: 10.1002/nme

2. INCREMENTAL SVD ALGORITHM

An incremental SVD algorithm from Brand [9] is used to update the SVD on-the-fly as snapshots are added. We require that an SVD algorithm used for adaptive snapshot selection compute the SVD using only a single pass over the data, because we assume that the FOM is solved only once for a given initial condition and right-hand side. Other single-pass incremental SVD algorithms are available; see Baker, Gallivan, and Van Dooren [10] for a recent review. Brand's algorithm was selected because of ease of implementation and the simplicity of describing the algorithm. It is also used in other ROM algorithms (for instance, [12, 13]).

Brand's algorithm[‡] takes as input an existing rank-k SVD of a matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$ defined by

$$\mathbf{M} = \mathbf{V}\,\mathrm{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}} + \mathbf{E}, \tag{5}$$

where $\mathbf{V}\mathrm{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}}$ is a rank-k truncated SVD of $\mathbf{M}$, $\mathbf{E} \in \mathbb{R}^{m\times n}$ is the error because of rank truncation, $\mathbf{s} \in \mathbb{R}^{k}$ is a vector containing the k largest singular values of $\mathbf{M}$ in nonincreasing order of magnitude, $\mathbf{V} \in \mathbb{R}^{m\times k}$ is a matrix containing the corresponding k left singular vectors, $\mathbf{W} \in \mathbb{R}^{n\times k}$ is a matrix containing the corresponding k right singular vectors, and $\mathbf{E} = \mathbf{0}$ if the rank of $\mathbf{M}$ is k. Let $\mathbf{c} \in \mathbb{R}^{m}$ be a column to be added to $\mathbf{M}$ and consider updating the rank-k truncated SVD of $\mathbf{M}$ in (5) to a truncated SVD of $[\mathbf{M}\ \mathbf{c}]$.

Set

$$p = \|\mathbf{c} - \mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{c}\|. \tag{6}$$

This incremental SVD algorithm arises from the identity

$$\begin{aligned}\left[\,\mathbf{V}\mathrm{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}}\ \ \mathbf{c}\,\right] &= \left[\,\mathbf{V}\ \ (\mathbf{I}-\mathbf{V}\mathbf{V}^{\mathrm{T}})\mathbf{c}/p\,\right]\begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{V}^{\mathrm{T}}\mathbf{c}\\ \mathbf{0} & p\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}^{\mathrm{T}}\\ &= \left[\,\mathbf{V}\ \mathbf{j}\,\right]\begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \boldsymbol{\ell}\\ \mathbf{0} & p\end{bmatrix}\begin{bmatrix}\mathbf{W} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}^{\mathrm{T}}.\end{aligned} \tag{7}$$

Note that the left and right matrices in the matrix triple products are semi-unitary; $\boldsymbol{\ell} \in \mathbb{R}^{k}$ is the projection of $\mathbf{c}$ onto the span of $\mathbf{V}$ in the $\mathbf{V}$ basis; p is the length of the orthogonal projection of $\mathbf{c}$ onto the orthocomplement of the range of $\mathbf{V}$.

In addition to the truncated SVD of the matrix $\mathbf{M}$ and the vector $\mathbf{c}$, Brand's algorithm also takes as input an SVD truncation tolerance $\varepsilon_{\mathrm{SVD}}$ used to determine if $\mathbf{c}$ is numerically linearly independent of the range of $\mathbf{M}$, $\mathcal{R}(\mathbf{M})$. Algorithm 1 initializes the incremental SVD when it is empty (i.e., $k = 0$), and algorithm 2 updates the incremental SVD when appending a new column when the SVD is not empty ($k > 0$). In algorithm 2, MATLAB indexing conventions are used as subscripts to denote submatrices extracted via indexing operations.

In algorithm 2, $\mathbf{Q}$ is a one-column bordered diagonal matrix, so it can be bidiagonalized in $O(k^2)$ time. The SVD of the resulting bidiagonal matrix can be computed in $O(k^2)$, so the SVD within Brand's incremental algorithm is cheap relative to the cost of a dense SVD on $[\mathbf{M}\ \mathbf{c}]$, and the entire truncated SVD takes $O(mnk^2)$ time, requiring $O(k(m+n+k))$ memory. For low-rank matrices, as is the case with the POD snapshot matrix $\mathbf{U}$ when $k \ll m$, the thin SVD is computed in $O(mnk)$ time and $O(k(m+n))$ memory [9, 10].

Steps (16) through (19) are not in the original algorithm by Brand. Step (16) tests to see if the dimension k of the singular basis $\mathbf{V}$ equals m, the number of state variables in the FOM. If it does, this and all subsequent column appends are automatically rank nonincreasing. This step preserves the orthogonality of $\mathbf{V}$ in the limit of large numbers of column appends when k approaches (or equals) m. Also of special note are steps (26) through (30), which are required in order to preserve the orthogonality of $\mathbf{V}$; these steps are in Brand's description of the algorithm[§] and are required

[‡]Brand presents multiple algorithms in his original conference paper on the incremental SVD. Here, we present the "naïve" version of Brand's incremental SVD algorithm using single-column updates, for simplicity.

[§]Specifically, Brand recommends modified Gram-Schmidt orthogonalization [9]; thin QR works well in practice.

DOI: 10.1002/nme

Algorithm 1 Initializing incremental SVD when $k = 0$.

function EMPTYINCSVD($\mathbf{c}, \varepsilon_{\text{SVD}}$)
  if $\|\mathbf{c}\| > \varepsilon_{\text{SVD}}$ **then** ▷ Truncated SVD is numerical rank $k = 1$.
    $\mathbf{s} \leftarrow [\|\mathbf{c}\|]$
    $\mathbf{V} \leftarrow \mathbf{c}/\|\mathbf{s}\|$
    $\mathbf{W} \leftarrow [1]$
  else ▷ Truncated SVD is numerical rank $k = 0$.
    $\mathbf{s} \leftarrow [\,]$
    $\mathbf{V} \leftarrow [\,]$
    $\mathbf{W} \leftarrow [\,]$
  end if
return $\mathbf{V}, \mathbf{s}, \mathbf{W}$
end function

Algorithm 2 Modified version of Brand's incremental SVD

function INCSVD($\mathbf{V}, \mathbf{s}, \mathbf{W}, \mathbf{c}, \varepsilon_{\text{SVD}}$) ▷ For POD, $\mathbf{W}$ can be omitted.
  if $k = 0$ **then** ▷ Initialize empty SVD; Algorithm 1.
    $[\mathbf{V}, \mathbf{s}, \mathbf{W}] \leftarrow$ EMPTYINCSVD($\mathbf{c}, \varepsilon_{\text{SVD}}$)
return $\mathbf{V}, \mathbf{s}, \mathbf{W}$
  end if
  $\boldsymbol{\ell} \leftarrow \mathbf{V}^{\mathrm{T}}\mathbf{c}$
  $p \leftarrow \sqrt{\mathbf{c}^{\mathrm{T}}\mathbf{c} - \boldsymbol{\ell}^{\mathrm{T}}\boldsymbol{\ell}}$ ▷ $p = \|\mathbf{c} - \mathbf{V}\boldsymbol{\ell}\| = \|\mathbf{c} - \mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{c}\|$.
  $\mathbf{j} \leftarrow (\mathbf{c} - \mathbf{V}\boldsymbol{\ell})/p$
  $\mathbf{Q} \leftarrow \begin{bmatrix} \text{diag}(\mathbf{s}) & \boldsymbol{\ell} \\ \mathbf{0} & p \end{bmatrix}$
  if $p < \varepsilon_{\text{SVD}}$ **then** ▷ If p is small, set its entry in $\mathbf{Q}$ to zero.
    $\mathbf{Q}_{\text{end,end}} \leftarrow 0$
  end if
  $[\mathbf{V}', \text{diag}(\mathbf{s}'), \mathbf{W}'^{\mathrm{T}}] \leftarrow \text{SVD}(\mathbf{Q})$ ▷ Omit $\mathbf{W}'$ if $\mathbf{W}$ omitted.

  ▷ In next line, if p small or $\mathbf{M}$ full rank, SVD rank held constant.
  if $p < \varepsilon_{\text{SVD}}$ or $k \geq m$ **then**
    $\mathbf{V} \leftarrow \mathbf{V}\mathbf{V}'_{1:k,\,1:k}$ ▷ Rotate left singular vectors.
    $\mathbf{s} \leftarrow \mathbf{s}'_{1:k}$
    $\mathbf{W} \leftarrow \mathbf{W}'_{1:k,:}$ ▷ Rotate right singular vectors; omit if $\mathbf{W}$ omitted.
  else ▷ $\mathbf{c}$ is not in the span of $\mathbf{M}$; SVD rank increases.
    $\mathbf{V} \leftarrow [\mathbf{V}\ \mathbf{j}]\mathbf{V}'$ ▷ Append, then rotate left singular vectors.
    $\mathbf{s} \leftarrow \mathbf{s}'$
    $\mathbf{W} \leftarrow \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}\mathbf{W}'$ ▷ Append and rotate step. Omit if $\mathbf{W}$ omitted.
    $k \leftarrow k + 1$
  end if ▷ In next line, ε is unit roundoff (`eps` in MATLAB).
  if $|\mathbf{V}\mathbf{T}_{1:,1}\mathbf{V}_{1:,\text{end}}| > \min(\varepsilon_{\text{SVD}}, \varepsilon \cdot m)$ **then**
    ▷ Basis has lost (numerical) orthogonality
    $[\mathbf{Q}', \mathbf{R}] \leftarrow \text{QR}(\mathbf{V})$ ▷ Reorthogonalize basis via thin QR.
    $\mathbf{V} \leftarrow \mathbf{Q}'$
  end if
return $\mathbf{V}, \mathbf{s}, \mathbf{W}$ ▷ Return SVD. Omit $\mathbf{W}$ if not given as argument previously.
end function

DOI: 10.1002/nme

because the orthogonality of $\mathbf{V}$ degrades as columns are added. The tolerance used in the orthogonality test in step (26) is the minimum of $\varepsilon \cdot m$ and ε_{SVD}, where ε is unit roundoff (`eps` in MATLAB). These orthogonalization tolerances were chosen on the grounds that $\varepsilon \cdot m$ is the same tolerance used in MATLAB for determining the rank of a matrix (in this case, $\mathbf{V}$), and the POD basis vectors should be at least as numerically linearly independent as the SVD truncation tolerance (recall that ε_{SVD} is a tolerance for testing linear independence of the snapshot from the POD basis). Removing either the $k \geq m$ test in step (16) or the reorthogonalization procedure in steps (26) through (30) degraded the accuracy of ROMs to the point of numerical instability and overflow in cases with large numbers of snapshots. Thus, in the implementation of this algorithm, it is absolutely critical that both of these features be implemented for robustness.

For POD, the most important thing to note is that only the left singular vectors and the singular values need be calculated; all computations involving right singular vectors can be omitted from the algorithm, including the use of the right singular vectors $\mathbf{W}$ as an input [25]. Consequently, for POD, the incremental SVD only requires $O(mk)$ memory; basis truncation occurs on the fly. In comparison, a dense SVD would require $O(mn^2 + m^2n + n^3)$ time and at least $O(m^2 + n^2 + mn)$ memory for the input and for the left and right singular vectors upon output [6]. An iterative SVD would require $O(mnk^2)$ time, but k must be known in advance, which is not necessarily true for POD, and the actions of the snapshot matrix and its transpose would be required [6, 9]. The storage requirements for Krylov-type iterative methods are still Ω(mn) (i.e., the full snapshot matrix) because there is no way of obtaining the action of the snapshot matrix (or its transpose) in matrix-free fashion in general, and thus are larger than necessary because $n \geq k$, and, generally, $n >> k$. The snapshot matrix is not sparse in general, so the sparse SVD methods in [7] offer no memory savings. Thus, in an asymptotic sense, the incremental SVD is the most efficient algorithm from a memory perspective, and also requires the fewest floating point operations in the low-rank limit.

In cases where $\bar{\mathbf{u}}$ is not known in advance of snapshot collection, if $\bar{\mathbf{u}}$ can be calculated on the fly as a function of the snapshots selected, then the SVD can be updated via a rank-1 update after all snapshots are selected [31]; error estimates will not reflect this rank-1 update.

Up until this point, the discussion has focused on the serial costs of incremental SVD. Although a full discussion of a parallel implementation of incremental SVD is out of the scope of this paper, we comment on it briefly, because our algorithm for POD training is developed with supercomputing in mind. Our parallel implementation of the incremental SVD in algorithms 1 and 2 is appropriate for POD training with small basis sizes (small k; here, "small" means that a dense k by k matrix fits in memory on a single process) assumes that all quantities in these algorithms are stored in distributed fashion, where blocks of rows are contiguous on each process, except for the $\mathbf{Q}$ matrix, which is stored redundantly on each process. The matrix $\mathbf{Q}$ does not require much memory because we assume k is small, and the SVD in algorithm 2, step (13) can be computed using standard linear algebra libraries such as LAPACK. The remaining steps use standard linear algebra operations such as matrix-vector and matrix-matrix multiplies that have been shown to be scalable (e.g., in distributed-memory linear algebra libraries such as ScaLAPACK and Elemental), and thus, our preliminary results indicate that our algorithm weak scales to 8192 cores on the LLNL cluster Sierra with 75% efficiency. A full discussion of parallel implementation details and scaling results is deferred to a future publication [30], because the focus on this work is on developing the initial serial verion of our POD training algorithms.

3. ERROR ESTIMATION

Controlling the error in the computed ROM solution using error estimates to determine the time interval between snapshots has obvious parallels to adaptive time stepping methods for ODEs that select time steps based on controlling estimated local truncation error [32]. In this section, an estimate of the approximation error in the ROM solution is derived so that it can be used in an error control mechanism in the adaptive snapshot selection algorithm presented in Section 4.

Recall from equation (4) that $\mathbf{v}$ approximates the solution $\mathbf{u}$ to equation (1). Define the error in that approximation by

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{8}$$

DOI: 10.1002/nme

so that $\mathbf{v} = \mathbf{u} - \mathbf{e}$. An ODE governing the error follows by substituting the definition in equation (4) into equation (8), differentiating both sides with respect to time, then substituting in the right-hand sides from the ODEs in equation (1) and (3), yielding

$$\dot{\mathbf{e}}(t) = \mathbf{f}(t, \mathbf{u}(t)) - \mathbf{B}\mathbf{B}^{\mathrm{T}}\mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}); \quad \mathbf{e}(0) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}). \tag{9}$$

Rewriting equation (9) as

$$\begin{aligned} \dot{\mathbf{e}}(t) &= (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{f}(t, \mathbf{u}(t)) + \mathbf{B}\mathbf{B}^{\mathrm{T}}[\mathbf{f}(t, \mathbf{u}(t)) - \mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}})] \\ \mathbf{e}(0) &= (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}) \end{aligned} \tag{10}$$

shows that there are two components to the error: (1) an out-of-subspace component (the first term on the right-hand side of equation (10) and (2) an in-subspace component (the second term on the right-hand side of equation (10)) [27].

Define the error estimator $\boldsymbol{\eta}$ by neglecting the in-subspace component of the error, yielding

$$\dot{\boldsymbol{\eta}}(t) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{f}(t, \mathbf{u}(t)); \quad \boldsymbol{\eta}(0) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}). \tag{11}$$

One motivation for using this error comes from trading off lower accuracy in the error estimate for lower computational cost. Note from equation (11) that integrating with respect to time yields

$$\boldsymbol{\eta}(t) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}(t) - \bar{\mathbf{u}}). \tag{12}$$

This quantity is computed as part of step (8) of algorithm 2, where t in that step is the time at which the snapshot is collected. Computing the right-hand side of equation (11) only requires evaluating the right-hand side of equation (1) – also calculated as part of solving the FOM ODE – as well as two matrix-vector multiplies, and a subtraction. No Jacobian matrix information is required, and the error estimate can be computed in real time without storing the FOM solution $\mathbf{u}$ as it his computed.

Another motivation for using only the out-of-subspace contributions to the error estimates in (11) and (12) comes from theory. Results on bounding the error in POD ROM solution show that as the out-of-subspace error approaches zero, the total error also approaches zero [27] over a compact interval in time. Therefore, error control based on estimating only the out-of-subspace error should be effective. The efficacy of this approach is confirmed in convergence studies in Section 5.

A potential disadvantage of using this estimator is that it neglects the in-subspace component of the error orthogonal to the error estimator defined by equation (11) and (12). Additional terms could be added to the right-hand side of equation (11) to improve its accuracy at the cost of requiring additional information, such as the Jacobian matrix of the right-hand side of the FOM ODE in equation (1) [15, 27, 28]. For problems requiring implicit time integration methods, this information must be provided or estimated as a matter of course. For problems using explicit time integration methods, this information is not typically provided because it is not needed; however, because only the action of the Jacobian would be necessary for improved error estimation, Jacobian-free matrix-vector products could be used as in [33] to estimate this required information, if necessary.

A more accurate error estimator will affect the growth of the estimated error, which will in turn influence when snapshots are selected. All other things equal, slightly underestimating the error might be advantageous, because fewer snapshots are preferred, especially when it is known that bounds on the error involving Lipschitz constants or logarithmic norms tend to overestimate the error. Recent work by Drohmann [34] suggests that error bounds and error estimators are strongly correlated with the actual error in the ROM and describes a procedure for modeling the true error, given error bounds or error estimates. This work could be used to relate the error estimator previously to the true error for more quantitative error control in snapshot selection.

4. SNAPSHOT SELECTION ALGORITHM

The premise of the adaptive snapshot selection algorithm is to gather snapshots on-the-fly while calculating the solution to the FOM ODE defined by equation (1). First, the algorithm is described

DOI: 10.1002/nme

informally, then the calculation of snapshot query times is presented, and finally, the algorithm is summarized in pseudocode.

Adaptive snapshot selection takes as input the FOM initial condition $\mathbf{u}^0$, an SVD truncation error tolerance ε_{SVD}, and a snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$. Concurrent to initializing the FOM ODE solution loop, adaptive snapshot selection is initialized by setting the snapshot query time t_Q to 0 (consistent with the initial condition, without loss of generality), the snapshot query time interval Δt_Q to the size of the first time step taken by the FOM ODE solver, the POD basis to an empty matrix, and the POD singular values to an empty vector. Then, within the FOM ODE solution loop, a snapshot is selected at the first time step that exceeds the current snapshot query time, and the POD basis is updated. The snapshot query time is then updated based on an estimated error criterion. This process of selecting a snapshot from the first time step after the current query time, updating the POD basis, and then updating the query time repeats until the query time exceeds the simulation end time, T, and the FOM ODE solution loop terminates.

The snapshot query time update is based on predicting the growth of the ROM approximation error in time between t_Q, the time of the most recently gathered snapshot, and $t_Q + \Delta t_Q$, the next snapshot query time. The ROM error at $t_Q + \Delta t_Q$ is estimated using a forward Euler method, so that

$$\boldsymbol{\eta}(t_Q + \Delta t_Q) = \boldsymbol{\eta}(t_Q) + \Delta t_Q \dot{\boldsymbol{\eta}}(t_Q) + \text{h.o.t}, \tag{13}$$

where $\boldsymbol{\eta}(t_Q)$ is calculated via (12), $\dot{\boldsymbol{\eta}}(t_Q)$ is calculated via (11), and higher order terms in the Taylor series are neglected. The value of Δt_Q is then updated from its current value using the formula

$$\Delta t_Q \leftarrow \text{mid}\left(\Delta t_{Q,\min}, \Delta t_{Q,\max}, \Delta t_Q \cdot \text{mid}\left(\varphi_{\min}, \varphi_{\max}, \varphi \cdot \left(\frac{1}{\|\boldsymbol{\eta}(t + \Delta t_Q)\|_e}\right)\right)\right), \tag{14}$$

where $\text{mid}(\cdot,\cdot,\cdot)$ takes the middle value of its three arguments, $\Delta t_{Q,\min}$ is a minimum query time step between snapshots, $\Delta t_{Q,\max}$ is a maximum query time step between snapshots, $0 < \varphi_{\min} < 1$ is a minimum time step scaling factor, $\varphi_{\max} > 1$ is a maximum time step scaling factor, $\varphi_{\min} < \varphi < 1$ is a time step scaling factor, and the error norm $\|\cdot\|_e$ is defined by

$$\|\mathbf{x}\|_e = \|\mathbf{x}/\varepsilon_{\text{snapshot}}\|_\infty. \tag{15}$$

The update formula (14) is inspired by methods for updating the time step in variable-step-length algorithms for solving ODEs, where the step size is estimated from a estimate of the local truncation error ([32], Section II.4). Values of $\varphi_{\min} = 0.1$, $\varphi_{\max} = 5$, and $\varphi = 0.8$ are suggested for similar time step selection formulas for ODEs; based on trial-and-error, $\varphi_{\min} = 0.05$ and $\varphi_{\max} = 10$ were used instead to adjust more aggressively the snapshot query time step in response to the error estimate. The weighted error norm can be modified to incorporate componentwise absolute and relative error tolerances, as shown in [32]; here, simple scaling by an absolute tolerance was chosen for simplicity. The error norm quotient term in step size formulas such as (14) typically has an exponent of $1/(q+1)$, where q is chosen to be either the order of the numerical method or the order of its embedded estimator, whichever is smaller. In (14), the error norm quotient term has an exponent of 1, corresponding to $q = 0$, because in this case, the error estimator is of order 1 (a first-order Taylor series approximation in time), and the embedded error estimator is zeroth-order. This embedded error estimator is zeroth-order because using a ROM in place of the FOM in a numerical method can be viewed as a zeroth-order approximation in time; the ROM right-hand side will not generally equal the first derivative of the FOM solution with respect to time.

From this informal description, the adaptive snapshot selection algorithm is defined in Algorithm 3.

Once $\mathbf{u}(t)$ is added as a snapshot, $\mathbf{u}(t)$ is in the range of $\mathbf{B}$, so $(\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{u}(t)$ is approximately $\mathbf{0}$, and this algorithm can be interpreted as controlling the estimated error between snapshots. The adaptive snapshot selection algorithm requires asymptotically negligible additional amounts of memory beyond the memory requirements of incremental SVD, so this algorithm requires $O(mk)$ memory, where k is the number of POD basis vectors.

DOI: 10.1002/nme

Algorithm 3 Sketch of offline ROM training stage: time-adaptive snapshot selection

function ADAPTIVESNAPSHOTSELECTION($\mathbf{u}^0$, ε_{SVD}, $\varepsilon_{\text{snapshot}}$)
  $t \leftarrow 0$, $t_Q \leftarrow 0$, $\mathbf{B} \leftarrow [\,]$, and $\mathbf{s} \leftarrow [\,]$ ▷ Initialize with empty basis.
  Initialize Δt_Q to size of first time step of ODE solver.
  while $t < T$ **do**
    if $t \geq t_Q$ **then**
      $[\mathbf{B}, \mathbf{s}] \leftarrow$INCSVD($\mathbf{B}$, $\mathbf{s}$,[], $\mathbf{u}(t) - \bar{\mathbf{u}}$, ε_{SVD}). ▷ See Algorithm 2.
      Calculate Δt_Q from (14). ▷ Uses snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$.
      $t_Q \leftarrow t + \Delta t_Q$ ▷ Update snapshot query time.
    end if
    Increment t by the next ODE solver time step and compute $\mathbf{u}(t)$ using ODE solver.
    ▷ The ODE solver controls time stepping.
  end while
end function

As with the discussion of the incremental SVD in Section 2, a full discussion of a parallel implementation of the snapshot selection algorithm is out of the scope of this paper, and deferred to a future publication [30]. We briefly comment that the algorithm used for snapshot selection is related to ODE time step control methods that have been implemented scalably in ODE solver software such as SUNDIALS [35, 36] and PETSc [37–39]. Thus, algorithm 3 can also be implemented scalably because the only communication required is a reduction to compute the norm in (14) and, of course, any communication in the incremental SVD in algorithms 1 and 2, which have already been discussed in Section 2.

5. RESULTS

To demonstrate the efficacy of the error estimator presented in Section 3 when it is used in the adaptive snapshot selection algorithm in Section 4, both the error estimator and the snapshot selection algorithm were implemented in MATLAB and used to generate ROMs for a 1-D computational example. Experiments were performed using MATLAB version R2015b (8.6.0.267246) using a MacBook Air (13-inch, Mid 2011) with 4 GB of 1333 MHz DDR3 RAM and an Intel Core i5-2557M processor with two cores running at a base clock frequency of 1.7 GHz (max turbo frequency 2.7 GHz). The default setting of two threads was used in experiments.

The one-dimensional transient viscous Burgers' equation over the interval $[0, L]$ in space and $[0, T]$ in time is a common test problem in the ROM literature [23, 28, 40, 41]. This equation takes the form:

$$u_t = \mu u_{xx} - \frac{a}{2}\left(u^2\right)_x - b u_x \tag{16}$$

where μ is a momentum diffusivity (viscosity), a is the speed of momentum advection, and b is a constant speed advection term.

All results presented in this paper set $L = 1$, $T = 10$, $\mu = 10^{-3}$, $a = 0.1$, and $b = 0$, and have homogeneous Dirichlet boundary conditions $u(0, t) = u(L, t) = 0$. A shifted Gaussian initial condition is used, taking the form

$$u(x, 0) = u_G(x) = \exp\left(\frac{-(x - 0.5)^2}{2 \cdot (\frac{1}{8})^2}\right) - \exp\left(\frac{-0.5^2}{2 \cdot (\frac{1}{8})^2}\right), \tag{17}$$

where the mean of the Gaussian is 0.5, its standard deviation is $1/8$, and the vertical shift ensures that $u_G(0) = u_G(1) = 0$.

The Gaussian initial condition is chosen to illustrate behavior for a smooth ($\mathcal{C}^\infty$) solutions. It is not possible to induce a discontinuous weak solution (excluding $t = 0$) for viscous Burgers' equation for the previously given parameter values and periodic Dirichlet boundary conditions

DOI: 10.1002/nme

because any $L^\infty([0,1])$ initial condition for viscous Burgers' condition under periodic boundary conditions yields a solution u that is C^∞ with respect to x for all $t > 0$ [37, Theorem 4.3.3].

The viscous Burgers' equation is discretized uniformly in space using second order centered finite differences for the diffusive term and first order centered finite differences for the advection terms with a grid spacing of $\Delta x = 5 \cdot 10^{-3}$. This choice of discretization is also taken from the literature [28, 41]. Although this discretization is unstable in the inviscid limit ($\mu \to 0$) [43], sufficiently high viscosity stabilizes it. The linearity of this discretization makes the resulting discrete equations amenable to precomputing the coefficient matrices of the ROM via offline/online decomposition. If a nonlinear discretization were used, such as a flux-limited Lax–Wendroff scheme commonly used in solving hyperbolic PDEs, the cost of the (nonlinear, non-quadratic) POD ROM right-hand side evaluations would scale with the dimension of the FOM because the discretized FOM would be nonlinear [14, 26, 27]. In practice, to reduce the computational cost of these evaluations, an interpolation method such as the discrete interpolation method [26] or Gauss–Newton with approximated tensors [14, 44] would be required to reduce the computational cost of the ROM to the point where it scales with the ROM dimension.

After spatial semidiscretization, the viscous Burgers' equation takes the form

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u} - \frac{a}{2}\mathbf{D}\,\mathrm{diag}(\mathbf{u})\mathbf{u}, \tag{18}$$

analogous to the FOM in equation 1, where $\mathbf{u}(t) \in \mathbb{R}^{201}$ is the spatially semidiscretized velocity field, $\mathbf{L} \in \mathbb{R}^{201\times201}$ is a discrete Laplacian matrix, $\mathbf{D} \in \mathbb{R}^{201\times201}$ is a discretized form of the first order central difference operator, and $\mathbf{A} \in \mathbb{R}^{201\times201}$ contains the linear operator part of viscous Burgers' equation: $\mathbf{A} = \mu\mathbf{L} - b\mathbf{D}$.

The semidiscretized viscous Burgers equation in equation (18) is then discretized uniformly in time using a second order explicit predictor-corrector scheme. The number of time steps over the interval $[0, T]$ was calculated based on convective and diffusive stability limits using

$$n_{\text{time steps}} = \left\lceil \frac{T}{\mathrm{CFL} \cdot \min\left(\frac{\Delta x}{u_{max}}, \frac{\Delta x}{|b|}, \frac{(\Delta x)^2}{2\mu}\right)} \right\rceil \tag{19}$$

where u_{max} is a bound on the velocity $u(x,t)$, and CFL denotes the CFL number, which is taken to be 0.9. For the case studies in this paper, u_{max} was taken to be 1, because it is an upper bound on both initial conditions under consideration, and it can be shown using logic similar to [42, Lemma 4.2.3] that $\|u(x,t)\|_\infty \leqslant \|u(x,0)\|_\infty$ for all $0 \leqslant t \leqslant T$ and all $0 \leq x \leq L$.[¶] In (19), the convective stability limit determines the time step for the case studies presented, yielding 2223 time steps of size $\Delta t = T/n_{\text{time steps}} = 4.5 \cdot 10^{-3}$.

A convergence study was performed with respect to the SVD truncation tolerance ε_{SVD} under the assumption that $\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$ (i.e, the SVD truncation tolerance and the snapshot selection error tolerance are equal). In these studies, the only parameters that change are ε_{SVD} and $\varepsilon_{\text{snapshot}}$, and ROMs are generated for $\varepsilon_{\text{SVD}} = \varepsilon_{\text{snapshot}} = 10^{-i}$ for 141 equally spaced floating-point values of i between 1 and 15 inclusive. The minimum and maximum snapshot query time intervals were set to $\Delta t_{Q,\min} = \Delta t$ and $\Delta t_{Q,\max} = T = 10$, respectively. Three ROMs were generated for each case: (1) an adaptive snapshot ROM, (2) a ROM using equispaced snapshots to match the number

[¶]The result in [42, Lemma 4.2.3] applies to the case where $u(x,0)$ is a $\mathcal{C}^\infty$ function with respect to x over the interval $[0,1]$ when $a = -1$ and $b = 0$ with periodic boundary conditions. A sketch of the required extensions follows: the result in [42, Lemma 4.2.3] holds if we instead change the domain of the PDE to $[0, L]$ via trivial modifications. If $u(x,0)$ is instead an $\mathcal{L}^\infty([0,L])$ function of x, then $u(x,t)$ is a $\mathcal{C}^\infty$ function of x [42, Theorem 4.3.3], and a change of coordinates in t yields the result in the case of $\mathcal{L}^1([0,L])$ initial conditions. The case where $a = 1$ follows by noting that if v is a solution to the viscous Burgers' equation (16) when $a = -1$, then setting $w(x,t) = v(-x,t)$ yields a solution to (16) with $a = 1$ with the same boundary conditions and value for μ, assuming appropriately modified initial conditions. The case where $a > 0, a \neq 1$ follows by scaling the x coordinate so that the scaled equation resembles the $a = 1$ case. If v is a solution to the viscous Burgers' equation (16) when $b = 0$, then for $b \neq 0$, $w(x,t) = v(x - bt, t)$ is a solution to this equation with the same boundary conditions, and values for a and μ, again assuming appropriately modified initial conditions.

DOI: 10.1002/nme

of snapshots from the adaptive snapshot ROM, and (3) a ROM using equispaced snapshots to match approximately a normalized root-mean-square error (NRMSE) of the adaptive snapshot ROM. For (3), approximately matching the NRMSE was performed by calculating the number of snapshots needed to minimize the difference between the equispaced ROM NRMSE and the adaptive snapshot ROM NRMSE. This minimization was performed by bisection over the number of snapshots. In general, the optimal objective function value of this minimization will not be zero because the number of snapshots is a discrete quantity, hence the NRMSE can only be matched approximately.

The NRMSE, a scaled $\mathcal{L}^2([0, L] \times [0, T])$ function norm, is defined by:

$$\mathrm{NRMSE}(u, v) = \frac{1}{\sqrt{LT}} \left(\int_0^L \int_0^T [u(x,t) - v(x,t)]^2 \, \mathrm{d}t \, \mathrm{d}x \right)^{1/2}, \tag{20}$$

where u and v are functions in $\mathcal{L}^2([0, L] \times [0, T])$. This quantity is calculated approximately by:

$$\mathrm{NRMSE}(u, v) = \left(\frac{\Delta x \Delta t}{LT} \sum_{i,j} \big(u_i(t_j) - v_i(t_j)\big)^2 \right) \tag{21}$$

so that a unit error in $u - v$ over $[0, L] \times [0, T]$ has an NRMSE of 1. For the remainder of this article, the NRMSE will always be computed between a FOM solution and a corresponding ROM solution trained on that FOM solution, so the arguments of the NRMSE function can be inferred from context.

ROMs are generated by first capturing snapshots (one ROM per method of snapshot selection) and then calculating a POD basis using the incremental SVD algorithm described in Section 2. These bases are then used to form the ODE system (3) via offline/online decomposition using a tensorial POD approach [45], yielding the system

$$\dot{\tilde{\mathbf{u}}} = \tilde{\mathbf{A}}\tilde{\mathbf{u}} - \frac{a}{2}\mathbf{B}^{\mathrm{T}}\mathbf{D}\,\mathrm{diag}(\mathbf{B}\tilde{\mathbf{u}})\mathbf{B}\tilde{\mathbf{u}} \tag{22}$$

where

$$\tilde{\mathbf{A}} = \mathbf{B}^{\mathrm{T}}(\mathbf{A} - a\mathbf{D}\,\mathrm{diag}(\bar{\mathbf{u}}))\mathbf{B}$$

is precomputed offline,[‖] and the constant coefficients of the quadratic term in (22) make up a rank-three tensor that is also precomputed offline.

The three ROM snapshot selection methods (adaptive snapshot, equispaced snapshots matching number of adaptive snapshots, and equispaced snapshots matching approximately the adaptive snapshot NRMSE) are compared for this case study. Keeping the number of snapshots constant while comparing adaptive snapshot selection with equispaced snapshots isolates the effect of snapshot location. If adapting the location of the snapshots achieves its intended goal, then adaptive snapshot selection ROMs will have smaller NRMSEs than equispaced snapshot ROMs, at constant numbers of snapshots. Keeping the NRMSE (approximately) constant while comparing adaptive snapshot selection with equispaced snapshots isolates the effect of ROM accuracy. If adaptively selecting snapshots based on ODE-like error control achieves its intended goal, then adaptive snapshot selection ROMs will be trained using fewer snapshots than equispaced snapshot ROMs at (approximately) constant NRMSE. The goal of these two comparisons is to investigate whether adaptive snapshot selection with an incremental SVD reduces the aggregate cost of ROM training (in floating-point operations and memory) relative to the situation where we have an oracle that knows *a priori* when to select snapshots, and a POD basis from these snapshots is computed using traditional dense SVD algorithms (e.g., using LAPACK). In both cases, adding more snapshots increases number of floating-point operations to compute the SVD. When a traditional dense SVD algorithm is used to compute the basis, more snapshots also means more memory is required to store the snapshots used to compute SVD.

[‖]In the derivation, both linearity of the $\mathrm{diag}(\cdot)$ operator and the identity $\mathrm{diag}(\mathbf{A}\mathbf{x})\mathbf{y} = \mathrm{diag}(\mathbf{y})\mathbf{A}\mathbf{x}$ are used; this identity holds for all matrices $\mathbf{A}$ and all vectors $\mathbf{x}$ and $\mathbf{y}$ that can be premultiplied by $\mathbf{A}$.

DOI: 10.1002/nme

Figure 1 demonstrates the numerical convergence behavior of the ROM NRMSE as the snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$ and SVD truncation error tolerance ε_{SVD} both approach zero, with $\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$; corresponding data can be found in the first and second columns of Table I. This behavior is to be expected based on the theory by [27], which states that as the out-of-subspace

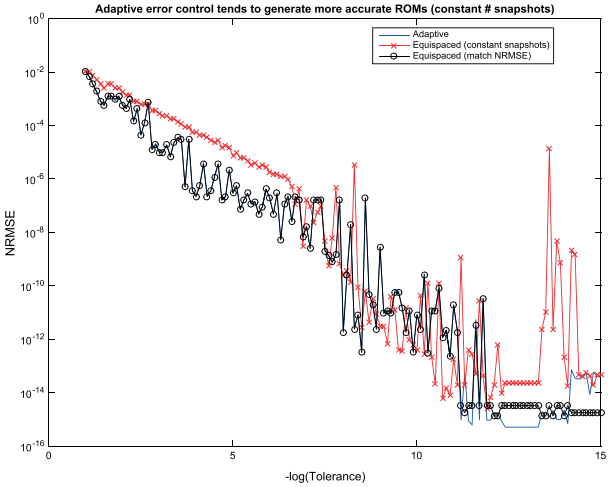

Figure 1. Plot of normalized root mean squared error (NRMSE) for reduced order models (ROMs) generated via adaptive snapshot selection, via equispaced snapshot selection while holding the number of snapshots constant, and via equispaced snapshot selection while holding the NRMSE approximately constant. Note that the NRMSE cannot truly be held constant because the number of snapshots selected for generating a ROM is a discrete quantity. When holding the number of snapshots constant, ROMs generated via adaptive snapshot selection tend to be more accurate than those generated via equispaced snapshot selection. For a snapshot selection error tolerance $\varepsilon_{\text{snapshot}} \leq 10^{-12}$, the ROM error is essentially discretization error, because the dimensions of the ROM and FOM are equal.

Table I. Tabular data of NRMSE for ROMs generated via three different snapshot selection methods.

	NRMSE for method		
		Equispaced, match	
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	snapshots	NRMSE
10^{-1}	$1.239807214007622 \cdot 10^{-2}$	$1.041443665186367 \cdot 10^{-2}$	$1.041443665186367 \cdot 10^{-2}$
10^{-2}	$6.296425416788367 \cdot 10^{-4}$	$1.848097617571954 \cdot 10^{-3}$	$5.607543713531529 \cdot 10^{-4}$
10^{-3}	$1.068645252002003 \cdot 10^{-5}$	$2.825597087719709 \cdot 10^{-4}$	$9.751951055901989 \cdot 10^{-6}$
10^{-4}	$2.536821496255930 \cdot 10^{-7}$	$5.485122993230182 \cdot 10^{-5}$	$2.157260217790364 \cdot 10^{-7}$
10^{-5}	$2.943286353118417 \cdot 10^{-7}$	$7.611039346356868 \cdot 10^{-6}$	$2.925197605369265 \cdot 10^{-7}$
10^{-6}	$1.937821517081686 \cdot 10^{-7}$	$1.806703691762856 \cdot 10^{-6}$	$1.945778328605047 \cdot 10^{-7}$
10^{-7}	$1.727708753982973 \cdot 10^{-8}$	$1.615245094755552 \cdot 10^{-7}$	$1.626985340785235 \cdot 10^{-8}$
10^{-8}	$2.111068179880942 \cdot 10^{-12}$	$2.514690293714660 \cdot 10^{-10}$	$1.847862167632617 \cdot 10^{-12}$
10^{-9}	$2.815371418185990 \cdot 10^{-9}$	$3.104189349251028 \cdot 10^{-12}$	$2.793268076260646 \cdot 10^{-9}$
10^{-10}	$8.251928175329557 \cdot 10^{-12}$	$4.019560059217289 \cdot 10^{-13}$	$8.303145841374870 \cdot 10^{-12}$
10^{-11}	$1.757797005527318 \cdot 10^{-11}$	$1.801446787941427 \cdot 10^{-13}$	$1.936521917155793 \cdot 10^{-11}$
10^{-12}	$9.376298017928967 \cdot 10^{-16}$	$6.498800727746222 \cdot 10^{-15}$	$3.266534899889506 \cdot 10^{-15}$
10^{-13}	$5.147194373348249 \cdot 10^{-16}$	$2.267367747074178 \cdot 10^{-14}$	$3.266534899889506 \cdot 10^{-15}$
10^{-14}	$2.236205734161060 \cdot 10^{-15}$	$2.063641283845553 \cdot 10^{-13}$	$1.344526095190805 \cdot 10^{-15}$
10^{-15}	$3.905071463152937 \cdot 10^{-14}$	$4.510902815043238 \cdot 10^{-14}$	$1.782006458775555 \cdot 10^{-15}$

Data corresponds to the plot in Figure 1.
NRMSE, normalized root mean squared error; ROMs, reduced order models.

DOI: 10.1002/nme

error approaches zero, the total error also approaches zero. In practice, the ROM error approaches the limit of numerical error in the FOM solution because in the limit as both $\varepsilon_{\text{snapshot}}$ and ε_{SVD} go to zero, a snapshot is collected at every time step and there is no truncation in the SVD, yielding a consistent ROM, as discussed in Section 1.3 and in both Carlberg, Bou-Mosleh, and Farhat [14] and Carlberg, *et al.* [44]. In other words, in that limit, the resulting basis **B** spans a subspace containing the FOM solution.

The snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$ controls the out-of-subspace error, because the error estimator defined in (12) is the out-of-subspace error. The SVD truncation error tolerance ε_{SVD}

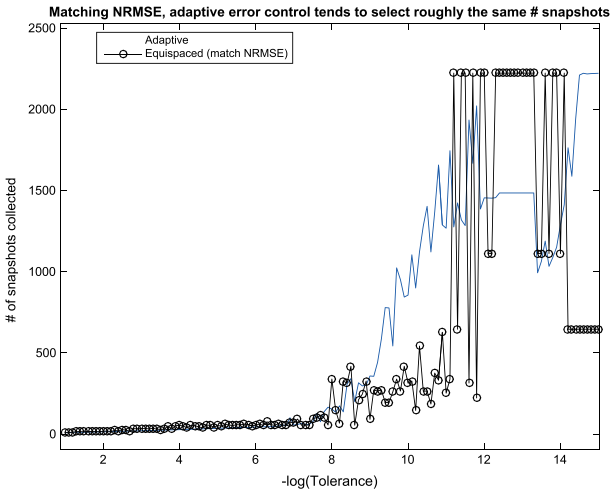

Figure 2. Plot of number of snapshots collected for reduced order models generated for the case study via adaptive snapshot selection and via equispaced snapshot selection while holding the normalized root mean squared error (NRMSE) approximately constant. Note also that for this case study, as the snapshot selection tolerance $\varepsilon_{\text{snapshot}}$ approaches zero, the number of snapshots collected should approach the total number of time steps taken in the full order model. For this case study, the total number of time steps taken is 2223.

Table II. Table of number of snapshots collected for ROMs generated for the case study via three different snapshot selection methods.

	Number of snapshots for method	
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	Equispaced, match snapshots
10^{-1}	9	9
10^{-2}	14	18
10^{-3}	21	35
10^{-4}	28	54
10^{-5}	37	53
10^{-6}	44	55
10^{-7}	56	68
10^{-8}	154	339
10^{-9}	358	910
10^{-10}	856	314
10^{-11}	1268	252
10^{-12}	1455	2223
10^{-13}	1485	2223
10^{-14}	1289	1112
10^{-15}	2222	646

Data listed here is plotted in Figure 2.
ROMs, reduced order models.

DOI: 10.1002/nme

controls the extent of truncation in the POD basis because column updates with out-of-subspace components having a norm less than ε_{SVD} do not increase the rank of the truncated SVD computed via incremental SVD. As the $\varepsilon_{\text{snapshot}}$ approaches zero, (14) implies that the query time step Δt_Q should approach the minimum query time step Δt_{min}. For this case study, the minimum query time step Δt_{min} equals the uniform time step Δt used to solve the FOM and each ROM. This behavior is illustrated in Figure 2 and Table II. As the SVD truncation tolerance ε_{SVD} approaches zero, the ROM dimension should approach the analytical rank of the snapshot matrix. In the limit of taking every time step as a snapshot, as implied by $\varepsilon_{\text{snapshot}} \to 0$, $\varepsilon_{\text{SVD}} \to 0$ should cause the ROM dimension k to approach the Kolmogorov n-width of the FOM solution. This behavior is shown in Figure 3 and Table III. When $k = n$, the ROM can be replaced by the FOM – that is, no reduction is possible for the given ε_{SVD} and $\varepsilon_{\text{snapshot}}$ – and the ROM error in that case should be the numerical error in the FOM solution because of discretization, roundoff, and so on.

Figure 1 also compares the NRMSE for the three types of ROMs mentioned earlier in Section 5, as the SVD truncation error tolerance ε_{SVD} and snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$ approach zero; Table I lists the data plotted in Figure 1. When the number of snapshots is held constant, equispaced snapshot ROMs tend to be less accurate than adaptive ROMs. This diminished accuracy is probably correlated with the trend in Figure 3 showing that when the number of snapshots is held fixed, adaptive ROMs are slightly larger in dimension than equispaced ROMs. Taken together, these results suggest that the adaptive ROM snapshots contain more (and more accurate) information on the behavior of the FOM (18) than ROMs using a comparable number of equispaced snapshots.

Figure 1 also shows that the NRMSE can only be held approximately constant when comparing an adaptive ROM to an equispaced ROM by varying the number of snapshots in the latter ROM. This behavior occurs because the number of snapshots is a discrete quantity. Figure 2 shows that to match approximately the NRMSE, equispaced ROMs require roughly the same number of snapshots as in the adaptive snapshot case up until $\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}} = 10^{-9}$. For smaller tolerances, the oscillations in Figure 2 suggest that there are cases where a POD ROM of comparable accuracy could be generated using fewer equispaced snapshots than the number selected via the adaptive procedure proposed in this work. However, determining the number of equispaced snapshots required to compute a sufficiently accurate ROM would be significantly more expensive for the general case of offline ROM training than the adaptive snapshot selection procedure. We generated the equispaced

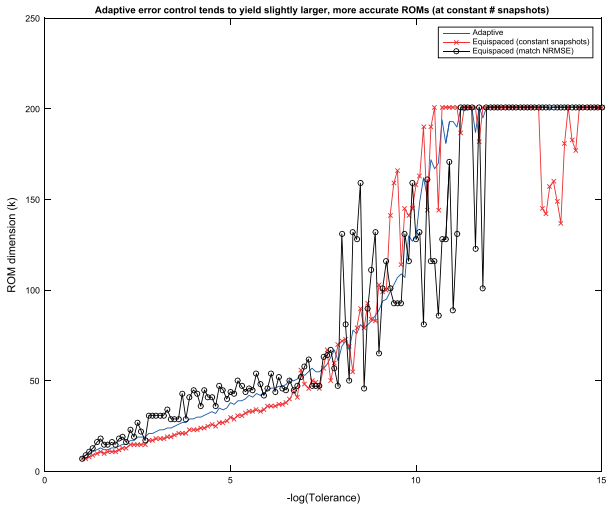

Figure 3. Plot of reduced order model (ROM) dimension (k) for ROMs generated for the case study via adaptive snapshot selection, via equispaced snapshot selection while holding the number of snapshots constant, and via equispaced snapshot selection while holding the normalized root mean squared error (NRMSE) approximately constant. Note also that for this case study, as the SVD truncation tolerance ε_{SVD} approaches zero, the ROM dimension approaches the analytical rank of the snapshot matrix. Combined with the trend that as the snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$ approaches zero, all time steps are collected, this statement implies that k should approach n. For this case study, $n = 201$.

DOI: 10.1002/nme

Table III. Table of ROM dimension k for ROMs generated for the case study via three different snapshot selection methods.

	ROM dimension (k) for method		
		Equispaced, match	
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	snapshots	NRMSE
10^{-1}	8	7	7
10^{-2}	14	12	18
10^{-3}	22	18	31
10^{-4}	29	23	45
10^{-5}	38	30	44
10^{-6}	45	36	46
10^{-7}	53	48	58
10^{-8}	69	72	131
10^{-9}	89	103	65
10^{-10}	130	158	128
10^{-11}	193	201	89
10^{-12}	201	201	201
10^{-13}	201	186	201
10^{-14}	201	181	201
10^{-15}	201	201	201

Data shown here is also plotted in Figure 3.
ROM, reduced order model; NRMSE, normalized root-mean-square error.

snapshot ROM matching a given NRMSE by first computing the FOM solution once and storing it, because there is no error control for equispaced snapshot selection procedures, and matching the NRMSE requires repeated selection of snapshots. Then, we repeatedly calculated the POD basis after the fact for different numbers of equispaced snapshots, computed the ROM solution, and then computing the NRMSE between the FOM and ROM solutions. As stated earlier in this section, a bisection procedure was used to drive this FOM-ROM simulation loop, and in practice, computing a POD basis yielding a sufficiently accurate ROM required at least 2 and as many as 13 ROM solves. If the entire FOM solution cannot be stored in memory at once, and only FOM snapshots can be stored, then there must be a FOM solve for each new POD basis generated, so for this case study, such a procedure would require between 2 and 13 FOM solves. For a computationally expensive FOM solve, the cost of repeated solves required for error-controlled equispaced snapshot ROMs could be prohibitive; this scenario was the motivation for developing the adaptive snapshot selection procedure.

Based on running convergence studies for different combinations of tolerances, we also found that ROM NRMSE for equispaced snapshots is a nonmonotonic function of the number of snapshots, which causes the oscillatory behavior and spikes in Figure 2. Bisection was used as a heuristic to control the error in ROM generated by equispaced snapshots because it limits the number of ROM solves to the base-2 logarithm of the number of time steps used in solving the FOM. As such, it selects a number of snapshots that, when used to build a POD ROM from equispaced snapshots of the FOM solution, closely matches the NRMSE of a ROM computed via adaptive snapshot selection. However, because the ROM NRMSE for equispaced snapshots is nonmonotonic, it may not select the minimal number of equispaced snapshots required to match the adaptive snapshot ROM NRMSE; it may instead select a larger number of snapshots than is actually required, yielding a spike in the data. A more accurate way to determine if a smaller number of snapshots is required to compute a sufficiently accurate ROM would be to compute the minimum number of snapshots required to build a POD basis that yields a ROM with smaller NRMSE than the ROM computed with adaptive snapshots. Due to the nonmonotonic behavior previously, the only way to find this

DOI: 10.1002/nme

minimum number would be to compute a ROM for every possible number of equispaced snapshots. For this case study, such an approach would have required at least 2223 ROM solves, which we felt was prohibitively expensive for testing because generating results would have taken days instead of hours. Furthermore, such an algorithm would be untenably expensive on production problems, because, again, if storing the full FOM solution is impractical, a FOM solve would be required for every ROM solve.

Finally, Figure 3 shows that matching approximately the NRMSE tends to yield smaller adaptive ROMs. This figure suggests that the adaptive snapshot procedure tends to select snapshots that contain more information than the equispaced snapshots selected using the bisection procedure described earlier in this section, because the adaptive ROMs have approximately the same NRMSE and tend to require fewer snapshots than equispaced ROMs (matching NRMSE).

6. CONCLUSION AND DISCUSSION

The convergence study previously in Section 5 demonstrates the efficacy of the error estimator presented in Section 3 as part of an adaptive snapshot selection algorithm in Section 4. When applied to a standard viscous Burgers' equation benchmark problem from the literature, this adaptive snapshot selection algorithm gathers snapshots in a manner that controls the error in the ROM solution. As algorithm error tolerances go to zero, this algorithm computes POD ROM bases such that their respective ROM solutions approach the FOM solution in the $\mathcal{L}^2$ sense to within discretization error, and the POD ROM dimension approaches the Kolmogorov n-width of the FOM solution. The number of snapshots required to achieve a given normalized root mean squared error (scaled $\mathcal{L}^2$ error) tends to be similar between equispaced and adaptive approaches, although the adaptive approach is considerably cheaper computationally because it does not require either multiple full-order model solves or storing the computed FOM solution (one of the two is required in the equispaced approach).

Independent of snapshot selection strategy, using an incremental SVD algorithm in concert with adaptive snapshot selection drastically reduces memory requirements of POD so that only the POD basis vectors, their corresponding singular values, and one snapshot must be stored in memory (along with an asymptotically negligible number of additional scalars). The adaptive snapshot selection algorithm reduces the computational overhead of POD by capturing fewer snapshots; this reduction in overhead is offset somewhat by calculating the error estimator for use in adaptive snapshot selection.

An obvious direction for future work is applying this algorithm to more production-scale problems to refine the parameters used, and to more thoroughly test the method by benchmarking it on a larger set of test problems. Given that POD ROMs do not effectively reduce the computational cost of general nonlinear FOMs, without incorporating hyperreduction methods (as discussed later), the method proposed here is limited to FOMs amenable to offline/online decomposition (typically, linear and quadratic models). The viscous Burgers' case study results suggest that adaptive snapshot selection could be effective for FOMs arising from parabolic PDEs. However, it is known that accurately approximating phenomena such as moving shocks [46, 47] that occur in hyperbolic PDEs is still challenging for ROMs. We are in the process of testing our adaptive snapshot selection algorithm on such problems and intend to refine it as necessary to achieve convergence in $\mathcal{L}^2$ error in the limit as algorithm error tolerances approach zero.

Another direction for future work is extending the error estimation and adaptive snapshot selection algorithms to select snapshots in parameter space as well as in time. The case of reduced order modeling applied to problems with varying parameters is a topic of great interest in the ROM community. In this context, incremental SVD approaches show promise as a limited-memory alternative to POD adaptive snapshot approaches based on full SVDs or on partial ('chunked') SVDs.

Another direction of interest would be developing adaptive snapshot selection algorithms for other data-driven model reduction methods. Of special mention are methods that generate basis vectors that could then be fed into ROM hyperreduction methods such as the discrete empirical interpolation method [26] or Gauss-Newton with approximated tensors [14, 44]. Hyperreduction methods are noteworthy for reducing the computational complexity (and thus, cost) of evaluating terms in a ROM

DOI: 10.1002/nme

originating from nonlinear terms in the FOM nonintrusively – that is, without requiring an offline/online decomposition or intimate knowledge of the right-hand side of the FOM.

Adaptive snapshot selection lends itself to the construction of localized ROM bases, which has been a topic of recent interest; a method for constructing local bases in time from the adaptive snapshot algorithm in Section 4 is straightforward and a topic for future work. Constructing local bases in parameter space is also a future research direction of interest.

Finally, the framework previously can be extended straightforwardly to include time derivative snapshots. Including time derivative snapshots [26, 29, 48–50] (or instead, difference quotients [51–53]) is a topic of recent interest because time derivative information is readily available in ODE simulations and shows promise in developing potentially more accurate ROMs, although there is some debate about how the time derivative snapshots should be scaled. Carlberg and Farhat argue for scaling sensitivity derivatives based on Taylor series in [50]. However, both Peng and Mohseni [49] and Chaturantabut and Sorensen [26, 48] do not scale time derivative snapshots. In the parameter-dependent case, analogous quantities of interest would also include sensitivity derivatives with respect to the parameters [50].

ACKNOWLEDGEMENTS

We would like to thank Bill Henshaw and Bob Anderson for helpful technical discussions. We would also like to thank Isam Alobaidi, Hiba Ghassan Fareed Fareed, and the reviewers for their comments, which greatly improved the clarity and content of this manuscript.

This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work was supported by LDRD grant 13-ERD-031 from Lawrence Livermore National Laboratory, and released under document number LLNL-TR-669265.

REFERENCES

1. Haasdonk B, Dihlmann M, Ohlberger M. A training set and multiple basis generation approach for parametrized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems* 2011; **17**:423–442.
2. Zahr MJ, Amsallem D, Farhat C. Construction of parametrically-robust CFD-based reduced-order models for PDE-constrained optimization. *in 21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2013; 1–11.
3. Antoulas AC. *Approximation of Large-Scale Dynamical Systems*, Advances in Design and Control. Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2005.
4. Sirovich L. Turbulence and the dynamics of coherent structures. Part I: coherent structures. *Quarterly of Applied Mathematics* 1987; **14**(3):561–571.
5. Berkooz G, Holmes P, Lumley JL. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics* 1993; **25**(1):539–575.
6. Golub G, Van Loan C. *Matrix Computations* (4th edn). Johns Hopkins University Press: Baltimore, MD, USA, 2013.
7. Rajamanickam S. Efficient algorithms for sparse singular value decomposition. *PhD*, University of Florida, 2009.
8. Paul-Dubois-Taine A, Amsallem D. An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *International Journal for Numerical Methods in Engineering* 2015; **102**(5):1262–1292. accepted.
9. Brand M. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision—ECCV 2002*. Springer: Berlin, Germany, 2002; 707–720.
10. Baker CG, Gallivan KA, Van Dooren P. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra and its Applications* 2012; **436**(8):2866–2888.
11. Chahlaoui Y, Van Dooren P. Model reduction of time-varying systems. In *Dimension reduction of large-scale systems*. Springer: Berlin, Germany, 2005; 131–148.
12. Washabaugh K, Amsallem D, Zahr M, Farhat C. Nonlinear model reduction for CFD problems using local reduced-order bases. In *42nd AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences*, Vol. 2686, AIAA Paper, New Orleans, LA, USA, 2012.
13. Braconnier T, Ferrier M, Jouhaud JC, Montagnac M, Sagaut P. Towards an adaptive POD/SVD surrogate model for aeronautic design. *Computers & Fluids* 2011; **40**(1):195–209.
14. Carlberg K, Bou-Mosleh C, 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* 2011; **86**(2):155–181.
15. Homescu C, Petzold LR, Serban R. Error estimation for reduced-order models of dynamical systems. *SIAM Journal on Numerical Analysis* 2005; **43**(4):1693–1714.

DOI: 10.1002/nme

16. Kunisch K, Volkwein S. Optimal snapshot location for computing POD basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis* 2010; **44**(3):509–529.
17. Lass O, Volkwein S. Adaptive POD basis computation for parametrized nonlinear systems using optimal snapshot location. *Computational Optimization and Applications* 2014; **58**(3):645–677.
18. Hoppe RHW, Liu Z. Snapshot location by error equilibration in proper orthogonal decomposition for linear and semilinear parabolic partial differential equations. *Journal of Numerical Mathematics* 2014; **22**(1):1–32.
19. Haasdonk B, Ohlberger M. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis* 2008; **42**(2):277–302.
20. Dihlmann M, Drohmann M, Haasdonk B. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. *Proc. of ADMOS* 2011; **2011**.
21. Ryckelynck D. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics* 2005; **202**(1):346–366.
22. Ryckelynck D, Missoum Benziane D. Multi-level *a priori* hyper-reduction of mechanical models involving internal variables. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(17-20):1134–1142.
23. Singler JR. New POD error expressions, error bounds, and asymptotic results for reduced order models of parabolic PDEs. *SIAM Journal on Numerical Analysis* 2014; **52**(2):852–876.
24. Grepl MA, Patera AT. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis* 2005; **39**(1):157–181.
25. Chahlaoui Y, Gallivan K, Van Dooren P. Recursive calculation of dominant singular subspaces. *SIAM Journal on Matrix Analysis and Applications* 2003; **25**(2):445–463.
26. Chaturantabut S, Sorensen DC. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing* 2010; **32**(5):2737–2764.
27. Rathinam M, Petzold LR. A new look at proper orthogonal decomposition. *SIAM Journal on Numerical Analysis* 2003; **41**(5):1893–1925.
28. Wirtz D, Sorensen DC, Haasdonk B. A posteriori error estimation for deim reduced nonlinear dynamical systems. *SIAM Journal on Scientific Computing* 2014; **36**(2):A311–A338.
29. Kostova T, Oxberry G, Chand K, Arrighi W. Error bounds and analysis of proper orthogonal decomposition model reduction methods using snapshots from the solution and the time derivatives. *arXiv preprint arXiv:1501.02004* 2015:1–26.
30. Arrighi W, Oxberry G, Vassilevska T, Chand K. libROM user guide and design. *Technical Report*, Lawrence Livermore National Laboratory, 2015.
31. Brand M. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications* 2006; **415**(1):20–30.
32. Hairer E, Wanner G, Nørsett SP. *Solving Ordinary Differential Equations I*, no. 8 in Springer Series in Computational Mathematics. Springer: Berlin, Germany, 1993.
33. Knoll DA, Keyes DE. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics* 2004; **193**(2):357–397.
34. Drohmann M, Carlberg K. The ROMES method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification* 2015; **3**(1):116–145.
35. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software* 2005; **31**(3): 363–396.
36. Cohen SD, Hindmarsh AC. CVODE, a stiff/nonstiff ODE solver in C. *Computers in Physics* 1996; **10**(2):138–143.
37. Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, Curfman McInnes L, Rupp K, Smith BF, Zampini S, Zhang H. *PETSc Web page*, 2016. URL:http://www.mcs.anl.gov/petsc.
38. Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, Curfman McInnes L, Rupp K, Smith BF, Zampini S, Zhang H. *PETSc Users Manual. Argonne National Laboratory*, 2016. Technical report ANL-95/11 - Revision 3.7. URL: http://www.mcs.anl.gov/petsc.
39. Balay S, Gropp WD, Curfman McInnes L, Smith BF. Efficient management of parallelism in object oriented numerical software libraries. In *Modern Software Tools in Scientific Computing* Arge E, Bruaset AM, Langtangen HP (eds.), Birkhäuser Press, Boston, MA, USA, 1997; 163–202.
40. Kalashnikova I, Barone MF. Stable and efficient Galerkin reduced order models for non-linear fluid flow. In *6th AIAA Theoretical Fluid Mechanics Conference*. American Institute of Aeronautics and Astronautics: Honolulu, Hawaii, 2011; 1–26.
41. Homescu C, Petzold LR, Radu S. Error estimation for reduced order models of dynamical systems. *Technical Report UCRL-TR-201494*, Lawrence Livermore National Laboratory, 2003.
42. Kreiss HO, Lorenz J. *Initial-Boundary Value Problems and the Navier-Stokes Equations*, no. 47 in SIAM Classics in Applied Mathematics. Society for Industrial and Applied Mathematics: Philadelphia, PA, 2004.
43. LeVeque RJ. *Finite Volume Methods for Hyperbolic Problems*, Vol. 31. Cambridge University Press: Cambridge, UK, 2002.
44. Carlberg K, Farhat C, Cortial J, Amsallem D. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics* 2013; **242**:623–647.

DOI: 10.1002/nme

45. Ştefănescu R, Sandu A, Navon IM. Comparison of POD reduced order strategies for the nonlinear 2D Shallow Water Equations. *International Journal for Numerical Methods in Fluids* 2014; **76**(8):497–521.
46. Lucia DJ. Reduced order modeling for high speed flows with moving shocks. *Technical Report*, DTIC Document, 2001.
47. Carlberg K. Adaptive h-refinement for reduced-order models. *International Journal for Numerical Methods in Engineering* 2015; **102**(5):1192–1210.
48. Chaturantabut S, Sorensen DC. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM Journal on Numerical Analysis* 2012; **50**(1):46–63.
49. Peng L, Mohseni K. An online manifold learning approach for model reduction of dynamical systems. *SIAM Journal on Numerical Analysis* 2014; **52**(4):1928–1952.
50. Carlberg K, Farhat C. A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. *International Journal for Numerical Methods in Engineering* 2011; **86**(3):381–402.
51. Iliescu T, Wang Z. Are the snapshot difference quotients needed in the proper orthogonal decomposition? *SIAM Journal on Scientific Computing* 2014; **36**(3):A1221–A1250.
52. Kunisch K, Volkwein S. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik* 2001; **90**(1):117–148.
53. Kunisch K, Volkwein S. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical Analysis* 2002; **40**(2):492–515.

DOI: 10.1002/nme