LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-669265

Limited-memory adaptive snapshot selection for proper orthogonal decomposition

G. M. Oxberry, T. Kostova-Vassilevska, B. Arrighi, K. Chand

April 2, 2015

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LIMITED-MEMORY ADAPTIVE SNAPSHOT SELECTION FOR PROPER ORTHOGONAL DECOMPOSITION*

GEOFFREY M. OXBERRY†, TANYA KOSTOVA-VASSILEVSKA‡, BILL ARRIGHI§, AND KYLE CHAND¶

Abstract. 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 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 reduced order models, 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. The resulting method can be used on supercomputers to generate proper orthogonal decomposition-based reduced order models, or as a subroutine within hyperreduction algorithms that require taking snapshots in time, or within greedy algorithms for sampling parameter space.

Key words. proper orthogonal decomposition, reduced order model, snapshot, incremental singular value decomposition

AMS subject classifications. 15A21, 65F30, 65L05

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 intended to be used in constructing ROMs from large-scale FOM simulations run on supercomputers. As such, it minimizes memory requirements for POD algorithms. It can be used as a component of a procedure for selecting snapshots from simulations of transient FOMs with parameters (for example, the greedy approach in [23]) in order to build POD reduced order models (ROMs) for applications such as optimization [52], 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 [2, 48, 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

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-TR-669265). This work was supported by LDRD grant 13-ERD-031 from Lawrence Livermore National Laboratory.

†Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA, 94550 (oxberry1@llnl.gov, corresponding author).

‡Center for Applied Scientific Computing, Lawrence Livermore National Lab, Livermore, CA, 94550 (vassilevska1@llnl.gov).

§Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, 94550 (arrighi2@llnl.gov).

¶Center for Applied Scientific Computing, Lawrence Livermore National Lab, Livermore, CA, 94550 (chand1@llnl.gov).

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 2-norm of a vector[1] 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 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

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

1.2. Proper Orthogonal Decomposition. Consider the ordinary differential equation (ODE)

$$\dot{\mathbf{u}}(t) = \mathbf{f}(t, \mathbf{u}(t)); \quad \mathbf{u}(0) = \mathbf{u}^0 \in \mathbb{R}^m \tag{1.1}$$

defining the FOM. The discrete version of POD[2] [48] takes as input a matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$ defined by

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^1 - \bar{\mathbf{u}} & \mathbf{u}^2 - \bar{\mathbf{u}} & \ldots & \mathbf{u}^n - \bar{\mathbf{u}} \end{bmatrix}, \tag{1.2}$$

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 (1.1)[3] and that the inner product used in POD is the standard Euclidean inner product on $\mathbb{R}^n$.[4]

Let $\mathbf{U} = \mathbf{V}\operatorname{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}}$ 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

$$\dot{\tilde{\mathbf{u}}}(t) = \mathbf{B}^{\mathrm{T}}\mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}); \quad \tilde{\mathbf{u}}(0) = \mathbf{B}^{\mathrm{T}}(\mathbf{u}^0 - \bar{\mathbf{u}}). \tag{1.3}$$

[1] Here, the 2-norm is used because we also use the Euclidean inner product when calculating the POD basis. When using an alternate inner product, the norm defined by that inner product should be used instead. The use of the infinity-norm is for ROM error control, and is unrelated to the choice of inner product for POD.

[2] The continuous version of POD is out of the scope of this article; consult [5] for a description.

[3] These snapshots need not come from simulation data; experimental data has been used, for instance, in fluid mechanics [1]. However, the focus of this work is on gathering data from numerical solutions, as this case is common.

[4] For a development of POD that relaxes this assumption on the inner product, see [46]. In essence, instead of calculating the eigenvalues and eigenvectors of $\mathbf{U}\mathbf{U}^{\mathrm{T}}$ via the singular value decomposition (SVD), changing the inner product means calculating the eigenvalues and eigenvectors of $\mathbf{\Sigma}^{1/2}\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{\Sigma}^{1/2}$, where $\mathbf{\Sigma}$ is a symmetric positive definite matrix induced by the inner product; in other words, we would compute the SVD of $\mathbf{\Sigma}^{1/2}\mathbf{U}$ instead, and the methods proposed here would still apply.

Upon solving equation (1.3), an approximation to the solution $\mathbf{u}$ of equation (1.1) is recovered via lifting (or interpolation, or prolongation)[5]:

$$\mathbf{v}(t) = \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}. \tag{1.4}$$

The motivation for using a ROM such as (1.3) 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. In general (though not always [46]), increasing the basis size decreases the error and increases both the number of arithmetic operations and amount of memory required to calculate a solution to the ROM. A value of k is chosen to balance the trade-off between accuracy and computational costs; typical choices made to preserve accuracy involve arguments based on an energy criterion (e.g., sum of largest k singular values makes up 95% of the sum of all singular values) or an error criterion (e.g., the $(k+1)$th singular value is less than 10^{-5}). In contrast, typical choices of k based on computational requirements set k so that solving the ROM is possible within limits on time and memory (e.g., set $k = 100$, based on these limits). 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 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, resources for the offline training phase are constrained. There are strict limitations on the amount of memory and computational time (or, as a proxy, number of floating-point operations) available for ROM training. 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.[6] Typical approaches for calculating the SVD as part of POD use conventional dense algorithms (see [21] for a comprehensive bibliography). Since these approaches require storing the whole snapshot matrix $\mathbf{U}$ in memory, these approaches will fail in the limit of large numbers of snapshots due to insufficient memory. Out-of-core dense SVD variants are undesirable due to increased latency of hard disks. Sparse direct[7] approaches to calculating the SVD [44]

[5] Terminology varies. Interpolation and prolongation follow by analogy to multigrid methods; see also the discrete empirical interpolation method [15]. Lifting follows by analogy to methods in optimization that re-express constraints in higher-dimensional spaces.

[6] Similar considerations apply when calculating the POD basis vectors using algorithms for calculating eigendecompositions; although this paper focuses on incremental SVD methods, incremental eigenspace methods [26, 27] could be used when eigendecompositions are preferred.

[7] Here, "direct" – as opposed to iterative – refers to the process used to generate a bidiagonal matrix; in any SVD, the reduction of this bidiagonal matrix to diagonal form to calculate the singular values is iterative.

are inapplicable, since the snapshot matrix $\mathbf{U}$ is not typically sparse. Iterative approaches to calculating the SVD typically calculate the largest k' singular values and the corresponding singular vectors (again, see [21] for a comprehensive bibliography); to use these methods to generate a POD basis, the largest $k' = k$ singular values and their corresponding left singular vectors must be calculated. However, k is not necessarily known *a priori*, so iterative approaches cannot be used in many practical cases.[8] Furthermore, iterative approaches would still require storing all snapshots in the snapshot matrix simultaneously, since it is unlikely that the action of the snapshot matrix (and its transpose) could be calculated in matrix-free fashion.

Recently, Paul-Dubois-Taine and Amsallem ([42], Section 3.2.5) propose two different methods for compressing the SVD by partitioning the matrix $\mathbf{U}$ into blocks of columns of $\mathbf{U}$, computing a[9] truncated SVD of $\mathbf{U}$, and then combining these truncated SVDs to obtain an approximate truncated SVD for each of these blocks. They apply their SVD compression method to transient, parameter-dependent problems such that each block of columns of $\mathbf{U}$ corresponds to snapshots for each time step of a simulation for a given parameter realization, one parameter per block. The number of modes used in the truncated SVD of each block is set *a priori*, rather than using an error criterion. In principle, instead of using parameter-based blocks, these blocking methods could use time-step-based blocks. These blocking methods require storing only a fraction of the total number of snapshots in memory, but do not update directly the SVD of the original snapshot matrix; bounds on the relative projection error incurred by their approximations are provided.

Single-pass incremental SVD methods [7, 4, 14] 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 [50, 6]. Over a single pass of the FOM solution data, incremental SVD algorithms require the least memory (in an asymptotic sense) of all deterministic algorithms available for computing the POD basis. This property is particularly notable in light of memory limitations per core on large supercomputers. A more detailed analysis of this point will be presented in Section 2.

Having discussed memory costs, there are two main contributions to the floating-point operation costs of offline ROM training. First, the cost of computing the POD ROM basis from the currently selected snapshots typically scales at least linearly with the number of snapshots selected. In summary: over a single pass of the FOM solution data, incremental SVD algorithms are the most efficient deterministic algorithms for calculating the POD basis as each snapshot is selected; a more detailed analysis of this point will be presented in Section 2. Second, the manner in which snapshots are selected also contributes to ROM training costs; for all but the simplest methods used, this dominates the cost of the computing the POD basis, and will be the focus of this subsection.

When gathering snapshots for transient models (ignoring parameter dependence), such as the FOM in equation (1.1), 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 by Carlberg, Bou-Mosleh, and Farhat [10], and by Carlberg, et al. [12].[10] This approach has zero

[8] As discussed above, a common approach is to determine k based on examining all singular values of the snapshots matrix $\mathbf{U}$; this information is usually unavailable in iterative approaches.

[9] For any matrix $\mathbf{A}$, there exist multiple SVDs. Excluding the case of degenerate singular values, complex-valued singular vectors are only unique up to multiplication by $e^{i\theta}$ for $\theta \in [0, 2\pi)$ (in the real-valued case, up to sign).

[10] This description oversimplifies the recommendations of Carlberg, Bou-Mosleh, and Farhat in

floating-point operation cost associated with snapshot selection, since the only operations involved are data movement from the FOM solution to the snapshot matrix[11]. This approach also has useful numerical properties, the most important being *consistency*, so that as k approaches m, the approximation error in the ROM approaches zero. A chief drawback of this approach is that it gathers a large number of snapshots, which could exhaust available memory when using conventional dense SVD approaches to compute the basis. To circumvent this memory limitation, more sophisticated approaches to snapshot gathering are needed to decrease the number of snapshots gathered. None of these approaches can have a smaller cost in floating-point operations than collecting a snapshot every time step. All subsequent snapshot selection algorithms discussed here aim to reduce the memory and computational costs of computing the POD basis by selecting fewer snapshots.

The simplest way to select fewer snapshots in time is to collect snapshots at every j simulation time steps for some natural number j. (See, for instance, the case studies in [29].) This approach also has zero floating-point operation cost because the only operations performed involve data movement.

More sophisticated approaches select snapshots by solving an ODE (or PDE) constrained optimization problem. However, this approach increases substantially the number of floating-point operations required due to the additional overhead beyond calculating the SVD of the snapshot matrix. Approaches such as those by Lass and Volkwein [39], Kunisch and Volkwein [38], and by Hoppe and Liu [30] set a given number n of snapshots and then find the times $t_1, \ldots, t_n$ at which snapshots are to be collected by solving a large nonlinear program that controls or minimizes some estimate of error in the ROM. If the number m of state variables is large, this approach requires a prohibitively large number of floating-point operations and a large amount of memory, due to the large ODE- (or PDE-) constrained nonlinear program that must be solved.

Greedy approaches control the error in the ROM solution by selecting snapshots that maximize a ROM error indicator over a predefined set of candidate snapshots. These approaches do not incorporate the constraint that the FOM solution can only be calculated forward in time[12]. Consequently, assuming that the candidate set of snapshots is the numerical solution of the FOM, these snapshot selection approaches either require that the whole numerical solution of the FOM be in memory [24] at each training iteration so that the error maximizing snapshot can be selected, or that the FOM solution be recalculated for each training iteration so that the error maximizing snapshot can be selected on the fly. The former approach has large memory requirements in the limit of large numbers of FOM solution time steps; the latter approach is prohibitively expensive when solving the FOM is expensive, which is usually the case when training a ROM. More recent, state-of-the-art greedy approaches reduce memory requirements somewhat by using time-partitioning methods that construct a basis for each time interval in the partition [17]. This adaptive time-partitioning

[10], Carlberg, et al. in [12] for the sake of streamlining the argument. They recommend instead to take a snapshot at every iteration of a nonlinear solve, which requires even more memory than merely collecting a snapshot at every time step. The distinction is important when employing implicit ODE methods, as opposed to the explicit ODE methods used in this work, and does not change the broader point that such an approach potentially requires a large amount of memory to store snapshots.

[11] These operations are integer operations.

[12] Or only backward in time, for adjoint problems. In this work, we assume time steps are calculated sequentially either forward or backward in time. We are not yet aware of any approaches that compute snapshots via time-parallel integration methods; for a brief overview of these methods, see the recent review by Gander [20] or recent work by Falgout, et al. [19].

variant requires some recomputation of the FOM solution while controlling the size of time intervals to trade off between accuracy and maximum basis size.

1.4. Error control. Snapshot selection for POD is a fundamental issue in determining both the accuracy of the ROM (e.g., how well $\mathbf{v}$ approximates $\mathbf{u}$, or $g(\mathbf{v})$ approximates $g(\mathbf{u})$ for some functional g) and the computational costs of the POD algorithm. By definition, values of $\mathbf{v} - \bar{\mathbf{u}}$ must be in the range of $\mathbf{B}$, which in turn must be a subset of the range of the snapshot matrix $\mathbf{U}$. Thus, the ROM solution cannot represent exactly the values of $\mathbf{u}$ for any time such that $\mathbf{u}(t) - \bar{\mathbf{u}}$ is not in the span of the snapshots; the distance $\mathbf{u}(t) - \bar{\mathbf{u}}$ to that subspace is a lower bound for the distance between $\mathbf{u}(t) - \bar{\mathbf{u}}$ and $\mathbf{v}(t) - \bar{\mathbf{u}}$.[13]

Empirically, it has been observed that certain snapshots, such as those depicting the location of shock fronts in compressible flow simulations, and more generally, snapshots depicting discontinuities in the (weak) solutions of PDEs, are of particular importance [41]. These snapshots must be selected in order to observe similar discontinuities in ROM solutions. Including discontinuous ROM solution snapshots is important to preserve the physical fidelity of ROMs because the discontinuities are not in the span of a finite number of continuous snapshots depicting discrete representations of continuous functions.

The need for accuracy is balanced by the offline and online computational costs of POD ROMs. The offline cost of POD ROMs was discussed in the previous two subsections. The online cost of POD ROMs scales with the size k of the POD basis. As noted in Section 1.2, an upper limit on k may be dictated by computational resource constraints; a lower limit on k is dictated by accuracy requirements via controlling the ROM approximation error, assuming a fixed collection of snapshots.

Almost all of the methods described above control ROM error with respect to an error tolerance via by selection the locations of snapshots in time. The notable exceptions are approaches that select every jth snapshot. As noted earlier, the consistency property of ROMs computed via collecting snapshots at every time step ensures that error control can be achieved by selecting the size k of the POD basis appropriately [12]. This property does not necessarily hold when selecting every jth snapshot for $j > 1$ and setting $j = 1$ can be impractical when memory is constrained. Also, note that current methods for selecting every jth snapshot have no error tolerances. Recent theory on bounding ROM error in terms of snapshot spacing in time [34] supports the intuition that reducing j (equivalently, reducing the time interval between snapshots) will reduce ROM error, and will yield a method for error control.

1.5. 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, this approach is the first to exploit the property that transient FOM solutions can only be calculated unidirectionally in time, without storing the whole solution in memory at once, or recomputing the solution multiple times.[14] Similar directionality considerations do not necessarily exist if, for instance, snapshots are selected in parameter space for system of algebraic equations with one parameter. In the single-parameter case, is not necessarily more expensive to compute snapshots moving back and forth in parameter space, whereas computing earlier solutions in time (assuming we solve

[13] This argument is a weaker version of the proof of [45, Proposition 4.2]; also see the diagram in [45, Figure 4.1].

[14] Again, assuming sequential calculation of the ODE or PDE solution; use of parallel-in-time approaches is an open research question.

the full-model ODE forward in time) is more expensive because it requires restarting integration from either given initial conditions or a checkpointed solution.

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, in contrast to existing methods that assume the FOM is a semi-discretized PDE using a specific spatial discretization (e.g., finite element [47, 22], finite volume [24]). As each snapshot is selected, the POD basis is updated using a single-pass incremental SVD algorithm developed for streaming data analysis [7]. 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 $\mathbf{B}$.

As suggested in [13] and [4], 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 blocking the SVD as in [42], but uses less memory because the "blocks" in the proposed approach are of size 1. The proposed approach also employs no intermediate compression steps. Instead, truncation of the SVD occurs on the fly, and after every added snapshot, the truncated incremental SVD approximates the SVD that would be obtained from conventional dense SVD algorithms.

The proposed basis updating approach is also related to the basis extension method proposed by Haasdonk and Ohlberger for use in POD-Greedy approaches [24], 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 [50]; the approach in this paper differs from that work in that the incremental SVD is applied to appending columns of the snapshot matrix rather than changing the offset $\bar{\mathbf{u}}$ of the snapshot matrix via a rank-1 update [8]. Furthermore, [50] assumes a given database of FOM solutions from which snapshots are selected for local ROMs via k-means clustering algorithms; this work instead focuses on how to select FOM solutions for such a database to build a global ROM. 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 [6] on steady-state parameter-dependent problems; time-dependent problems are not considered. ROM approximation error is estimated using "leave-one-out" approach, where the error at a given parameter sample is calculated as the difference between the current ROM solution at that parameter realization and the solution of a lower-fidelity ROM constructed by leaving out the snapshot at that parameter sample (here, snapshots are calculated by solving the FOM at parameter samples). Such an approach is undesirable for time-dependent problems without parameter dependence, because it would require multiple passes over the data, and does not leverage any theoretical work done on error bounding and error estimation for ROMs on this class of problems.

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, 45, 22, 29, 47, 51, 34]). 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, [30] but without resorting to solving an expensive optimization problem in both memory and arithmetic operations.

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 a FOM solution. 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. To that end, an upcoming publication [3] describes a parallel version of the algorithm presented in this paper.

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 to 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.

2. Incremental SVD algorithm. An incremental SVD algorithm from Brand [7] 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, since we assume that the full order model 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 [4] for a recent review. Brand's algorithm was selected due to ease of implementation and the simplicity of describing the algorithm. It is also used in other ROM algorithms (for instance, [50, 6]).

Brand's algorithm[15] takes as input an existing rank-k SVD of a matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$ defined by

$$\mathbf{M} = \mathbf{V}\,\mathrm{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}} + \mathbf{E}, \tag{2.1}$$

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 due to 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 (2.1) to a truncated SVD of $[\mathbf{M}\ \mathbf{c}]$.

Set

$$p = \|\mathbf{c} - \mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{c}\|. \tag{2.2}$$

This incremental SVD algorithm arises from the identity

$$\begin{aligned}\begin{bmatrix} \mathbf{V}\,\mathrm{diag}(\mathbf{s})\mathbf{W}^{\mathrm{T}} & \mathbf{c} \end{bmatrix} &= \begin{bmatrix} \mathbf{V} & (\mathbf{I}-\mathbf{V}\mathbf{V}^{\mathrm{T}})\mathbf{c}/p \end{bmatrix} \begin{bmatrix} \mathrm{diag}(\mathbf{s}) & \mathbf{V}^{\mathrm{T}}\mathbf{c} \\ \mathbf{0} & p \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \mathbf{V} & \mathbf{j} \end{bmatrix} \begin{bmatrix} \mathrm{diag}(\mathbf{s}) & \boldsymbol{\ell} \\ \mathbf{0} & p \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{2.3}$$

[15] 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.

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 ε_{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.

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

```
1: function EmptyIncSVD(c, ε_SVD)
2:     if ‖c‖ > ε_SVD then                ▷ Truncated SVD is numerical rank k = 1.
3:         s ← [‖c‖]
4:         V ← c/‖s‖
5:         W ← [1]
6:     else                               ▷ Truncated SVD is numerical rank k = 0.
7:         s ← [ ]
8:         V ← [ ]
9:         W ← [ ]
10:    end if
11: return V, s, W
12: end function
```

In algorithm 2, $\mathbf{Q}$ is a 1-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 [7, 4].

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 non-increasing. 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[16], and are required 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 as 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

[16] Specifically, Brand recommends modified Gram-Schmidt orthogonalization [7]; thin QR works well in practice.

Algorithm 2 Modified version of Brand's incremental SVD

function IncSVD($\mathbf{V}, \mathbf{s}, \mathbf{W}, \varepsilon_{\text{SVD}}$)
  if $k = 0$ **then** ▷ Initialize empty SVD; see Algorithm 1.
    $[\mathbf{V}, \mathbf{s}, \mathbf{W}] \leftarrow \text{EmptyIncSVD}(\mathbf{c}, \varepsilon_{\text{SVD}})$
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
  $[\mathbf{V}', \text{diag}(\mathbf{s}'), \mathbf{W}'^{\mathrm{T}}] \leftarrow \text{SVD}(\mathbf{Q})$.

  ▷ 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.
    $\mathbf{s} \leftarrow \mathbf{s}'_{1:k}$
    $\mathbf{W} \leftarrow \mathbf{W}\mathbf{W}'_{:,\,1:k}$ ▷ Rotate right singular vectors.
  else ▷ $\mathbf{c}$ is not in the span of $\mathbf{M}$; SVD rank increases.
    $\mathbf{V} \leftarrow [\mathbf{V}\ \mathbf{j}]\mathbf{V}'$
    $\mathbf{s} \leftarrow \mathbf{s}'$
    $\mathbf{W} \leftarrow \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{W}'$
    $k \leftarrow k + 1$
  end if ▷ In next line, ε is unit roundoff (`eps` in MATLAB).
  if $|\mathbf{V}^{\mathrm{T}}{}_{1:,1}\, \mathbf{V}_{1:,\text{end}}| > \min(\varepsilon_{\text{SVD}}, \varepsilon \cdot m)$ **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.
end function

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 [13]. 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 [21]. An iterative[17]

[17] Again, with respect to the SVD, "direct" and "iterative" are only being used to describe the algorithm for generating a bidiagonal matrix within the SVD algorithm.

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 [7, 21]. Since there is no way of obtaining the action of the snapshot matrix (or its transpose) in matrix-free fashion in general, the storage requirements for Krylov-type iterative methods are still $\Omega(mn)$ (i.e., the full snapshot matrix), and larger than necessary, since $n \geq k$, and generally, $n \gg k$. The snapshot matrix is not sparse in general, so sparse direct SVD methods offer no memory savings [44]. 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 [8]; error estimates will not reflect this rank-1 update.

3. Error estimation. Controlling the error in 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 [25]. 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 (1.4) that $\mathbf{v}$ approximates the solution $\mathbf{u}$ to equation (1.1). Define the error in that approximation by

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{3.1}$$

so that $\mathbf{v} = \mathbf{u} - \mathbf{e}$. An ODE governing the error follows by substituting the definition in equation (1.4) into equation (3.1), differentiating both sides with respect to time, then substituting in the right-hand sides from the ODEs in equations (1.1) and (1.3), yielding

$$\dot{\mathbf{e}}(t) = \mathbf{f}(t, \mathbf{u}(t)) - \mathbf{B}\mathbf{B}^{\mathrm{T}}\mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}}); \quad \mathbf{e}(0) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}) \tag{3.2}$$

Rewriting equation (3.2) as

$$\begin{aligned} \dot{\mathbf{e}}(t) &= (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{f}(t, \mathbf{u}(t)) + \mathbf{B}\mathbf{B}^{\mathrm{T}}[\mathbf{f}(t, \mathbf{u}(t)) - \mathbf{f}(t, \mathbf{B}\tilde{\mathbf{u}}(t) + \bar{\mathbf{u}})] \\ \mathbf{e}(0) &= (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}) \end{aligned} \tag{3.3}$$

shows that there are two components to the error: (a) an out-of-subspace component (the first term on the right-hand side of equation (3.3)), and (b) an in-subspace component (the second term on the right-hand side of equation (3.3)) [45].

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

$$\dot{\boldsymbol{\eta}}(t) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{f}(t, \mathbf{u}(t)); \quad \boldsymbol{\eta}(0) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}^0 - \bar{\mathbf{u}}) \tag{3.4}$$

One motivation for using this error comes from trading off lower accuracy in the error estimate for lower computational cost. Note from equation (3.4) that integrating with respect to time yields

$$\boldsymbol{\eta}(t) = (\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})(\mathbf{u}(t) - \bar{\mathbf{u}}). \tag{3.5}$$

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[18]. Computing the right-hand side of equation (3.4) only requires evaluating the right-hand side of equation (1.1) – 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 is computed.

Another motivation for using only the out-of-subspace contributions to the error estimates in (3.4) and (3.5) 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 [45] 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 equations (3.4) and (3.5). Additional terms could be added to the right-hand side of equation (3.4) 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 (1.1) [29, 45, 51]. 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, since 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. Since fewer snapshots are preferred, all other things equal, slightly underestimating the error might be advantageous, 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 [18] 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 above 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 (1.1). First, the algorithm is described 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 $\varepsilon_{\mathrm{SVD}}$, and a snapshot selection error tolerance $\varepsilon_{\mathrm{snapshot}}$. Concurrent to initializing the FOM ODE solution loop, adaptive snapshot selection

[18] In this algorithm, the matrix of left singular vectors $\mathbf{V}$ is the POD basis $\mathbf{B}$, and the added column $\mathbf{c}$ would be a snapshot of the form $\mathbf{u}^i - \bar{\mathbf{u}}$, as in equation (1.2).

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

$$\boldsymbol{\eta}(t_Q + \Delta t_Q) = \boldsymbol{\eta}(t_Q) + \Delta t_Q \dot{\boldsymbol{\eta}}(t_Q) + \text{h.o.t}, \tag{4.1}$$

where $\boldsymbol{\eta}(t_Q)$ is calculated via (3.5), $\dot{\boldsymbol{\eta}}(t_Q)$ is calculated via (3.4), 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

$$\Delta t_Q \leftarrow \text{mid}\left(\Delta t_{Q,\min}, \Delta t_{Q,\max}, \Delta t_Q \cdot \text{mid}\left(\varphi_{\min}, \varphi_{\max}, \varphi \cdot \left(\frac{1}{\|\boldsymbol{\eta}(t + \Delta t_Q)\|_e}\right)\right)\right), \tag{4.2}$$

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

$$\|\mathbf{x}\|_e = \|\mathbf{x}/\varepsilon_{\text{snapshot}}\|_\infty. \tag{4.3}$$

The update formula (4.2) 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 ([25], Section II.4). Values of $\varphi_{\min} = 0.1$, $\varphi_{\max} = 5$, $\varphi = 0.8$ 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 [25]; here, simple scaling by an absolute tolerance was chosen for simplicity. The error norm quotient term in step size formulas such as (4.2) 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 (4.2), the error norm quotient term has an exponent of 1, corresponding to $q = 0$, since 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, since 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 by Algorithm 3.

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$(\mathbf{B}, \mathbf{s}, \mathbf{u}(t_Q) - \bar{\mathbf{u}}, \varepsilon_{\text{SVD}})$. ▷ See Algorithm 2.
      Calculate Δt_Q from (4.2). ▷ Uses snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$.
      $t_Q \leftarrow t_Q + \Delta t_Q$ ▷ 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

Once $\mathbf{u}(t_Q)$ is added as a snapshot, $\mathbf{u}(t_Q)$ is in the range of $\mathbf{B}$, so $(\mathbf{I} - \mathbf{B}\mathbf{B}^{\mathrm{T}})\mathbf{u}(t_Q)$ 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.

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.

The 1-D 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 [51, 47, 32, 28]. This equation takes the form:

$$u_t = \mu u_{xx} - \frac{a}{2}(u^2)_x - b u_x \tag{5.1}$$

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 $L = 1$, $T = 10$, $\mu = 0.01$, $a = 0.1$, 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

$$u(x, 0) = u_G(x) = \exp\left(\frac{-(x - 0.5)^2}{2 \cdot (\frac{1}{8})^2}\right) - \exp\left(\frac{-0.5^2}{2 \cdot (\frac{1}{8})^2}\right), \tag{5.2}$$

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. Since any $L^\infty([0,1])$ initial condition for viscous Burgers' equation under periodic boundary conditions yields a solution u that is $\mathcal{C}^\infty$ with respect to x for all $t > 0$ [35, Theorem 4.3.3], it is not possible to induce a discontinuous weak solution (excluding $t = 0$) for viscous Burgers' equation for the parameter values and periodic Dirichlet boundary conditions given above.

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 [51, 28]. Although this discretization is unstable in the inviscid limit ($\mu \to 0$) [40], 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.[19] 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 [45, 10, 15]. In practice, to reduce the computational cost of these evaluation, an interpolation method such as the discrete interpolation method (DEIM) [15] or Gauss-Newton with approximated tensors (GNAT) [10, 12] 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

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u} - \frac{a}{2}\mathbf{D}\,\mathrm{diag}(\mathbf{u})\mathbf{u}, \tag{5.3}$$

analogous to the FOM in equation (1.1), where $\mathbf{u}(t) \in \mathbb{R}^{201}$ is the spatially semidiscretized velocity field, $\mathbf{L} \in \mathbb{R}^{201\times 201}$ is a discrete Laplacian matrix, $\mathbf{D} \in \mathbb{R}^{201\times 201}$ is a discretized form of the first order central difference operator, and $\mathbf{A} \in \mathbb{R}^{201\times 201}$ contains the linear operator part of viscous Burgers' equation: $\mathbf{A} = \mu\mathbf{L} - b\mathbf{D}$.

The semidiscretized viscous Burgers equation in equation (5.3) 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

$$n_{\text{time steps}} = \left\lceil \frac{T}{\mathrm{CFL} \cdot \min\left(\frac{\Delta x}{u_{max}}, \frac{(\Delta x)^2}{2\mu}\right)} \right\rceil \tag{5.4}$$

[19] The basic idea of tensorial POD is that for full-order models that consist of explicit ODEs that have right-hand sides that are polynomials of degree p or less, the resulting POD ROM ODEs have right-hand sides that are also polynomials of degree p or less. The coefficient matrices and tensors of these ROMs can be precomputed for a one-time cost, and doing so is advantageous because these precomputed coefficient matrices can be used to evaluate the ROM right-hand sides at a cost proportional to the pth power of the ROM dimension. In practice, this tensorial approach is efficient only for $p \leq 2$ [49]. For full-order models with more general nonlinear right-hand sides, evaluating POD ROMs requires computing the full-order model right-hand side and two matrix-vector multiplies that scale with the dimension of the full-order model [45, 10, 15]. This scaling motivates hyperreduction methods that reduce the cost of ROM right-hand side evaluations.

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 [35, Lemma 4.2.3] that $\|u(x,t)\|_\infty \leq \|u(x,0)\|_\infty$ for all $0 \leq t \leq T$.[20] In (5.4), the diffusive stability limit determines the time step for the case studies presented, yielding 8901 time steps of size $\Delta t = T/n_{\text{time steps}} = 1.1 \cdot 10^{-3}$.

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 $i = 1, \ldots, 15$. 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 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. Since the number of snapshots is a discrete quantity, in general, the optimal objective function value of this minimization will not be zero, hence the NRMSE can only be matched approximately.

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

$$\text{NRMSE}(u,v) = \frac{1}{\sqrt{LT}} \left(\int_0^L \int_0^T [u(x,t) - v(x,t)]^2 \, \mathrm{d}t \, \mathrm{d}x \right)^{1/2}, \tag{5.5}$$

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

$$\text{NRMSE}(u,v) = \left(\frac{\Delta x \Delta t}{LT} \sum_{i,j} (u_i(t_j) - v_i(t_j))^2 \right) \tag{5.6}$$

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 capturing snapshots (one ROM per method of snapshot selection) and calculating a POD basis using the incremental SVD algorithm described

[20] The result in [35, 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 [35, 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 [35, 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 $a > 0$, $a \neq 1$ 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 5.1 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.

in Section 2. These bases are then used to form the ODE system (1.3) via offline/online decomposition using a tensorial POD approach [49], yielding the system

$$\dot{\tilde{\mathbf{u}}} = \tilde{\mathbf{A}}\tilde{\mathbf{u}} - \frac{a}{2}\mathbf{B}^{\mathrm{T}}\mathbf{D}\,\mathrm{diag}(\mathbf{B}\tilde{\mathbf{u}})\mathbf{B}\tilde{\mathbf{u}} \tag{5.7}$$

where

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

is precomputed offline[21], and the constant coefficients of the quadratic term in (5.7) 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. A comparison of adaptive snapshot selection to equispaced snapshots matching the number of adaptive snapshots isolates the effect of snapshot location, keeping the number of snapshots constant. 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. A comparison of adaptive snapshot selection to equispaced snapshots matching approximately the adaptive snapshot NRSME instead keeps the NRMSE approximately constant to study the differences in the number of snapshots between two methods for ROMs of approximately the same quality, where NRMSE is used as the quality figure of merit. 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. Taken together, these comparisons investigate whether adaptive snapshot selection saves memory by selecting fewer snapshots and whether it yields more accurate ROMs than comparable naive approaches that are designed to save memory by crudely reducing the number of snapshots gathered.

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 these methods compute snapshot locations that minimize the error between the ROM and FOM solutions, so the ROM error from these methods must necessarily 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.

Figure 5.1 demonstrates the numerical convergence behavior of the ROM error 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 5.1. This behavior is to be expected based on the theory by [45], which states that as the out-of-subspace error approaches zero, the

[21] 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 $\mathbf{A}$.

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 [10] and Carlberg, et al. [12]. In cases where gathering every snapshot yields a full-rank snapshot matrix, the resulting basis $\mathbf{B}$ is the identity, and there is no approximation error due to model reduction.

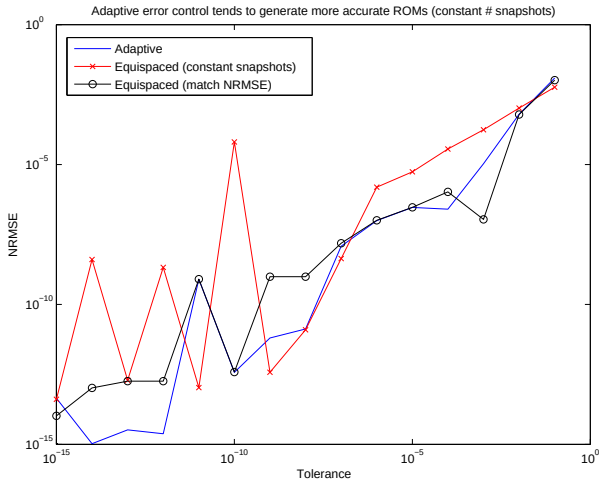

FIG. 5.1. *Plot of normalized root mean squared error (NRMSE) for ROMs generated via adaptive snapshot selection, via equispaced snapshot selection while holding the number of snapshots constant, and via equispaced snapshot selection while holding the NRMSE approximately constant. Note that the NRMSE 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* $\varepsilon_{snapshot} \leq 10^{-12}$*, the ROM error is essentially discretization error, because the dimensions of the ROM and FOM are equal.*

The snapshot selection error tolerance $\varepsilon_{\text{snapshot}}$ controls the out-of-subspace error, because the error estimator defined in (3.5) is the out-of-subspace error. The SVD truncation error tolerance ε_{SVD} controls the extent of truncation in the POD basis because column updates with out-of-subspace components having a norm[22] less than ε_{SVD} do not increase the rank of the truncated SVD computed via incremental SVD. As the $\varepsilon_{\text{snapshot}}$ approaches zero, (4.2) implies that the query time step Δt_Q should approach the minimum query time step $\Delta t_{\min}$. For this case study, the minimum query time step $\Delta t_{\min}$ equals the uniform time step Δt used to solve the FOM and each ROM. This behavior is illustrated in Figure 5.2 and Table 5.2. 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 FOM dimension n. This behavior is shown in Figure 5.3 and Table 5.3. When $k = n$, the ROM can be replaced by the FOM – that is, no reduction is

[22]This norm is induced by the inner product used in POD. In this work, the standard inner product and 2-norm are used; again, see [46] for a development of POD that relaxes this assumption.

TABLE 5.1
Tabular data of normalized root mean squared error (NRMSE) for ROMs generated via three different snapshot selection methods. Data corresponds to the plot in Figure 5.1.

	NRMSE for method		
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	Equispaced, match	
		snapshots	NRMSE
10^{-1}	$1.239807214007622 \cdot 10^{-2}$	$5.828677008862002 \cdot 10^{-3}$	$5.828677008862002 \cdot 10^{-3}$
10^{-2}	$6.296425416789283 \cdot 10^{-4}$	$1.039243702024760 \cdot 10^{-3}$	$6.129920029249113 \cdot 10^{-4}$
10^{-3}	$1.068645251991729 \cdot 10^{-5}$	$1.746999648199025 \cdot 10^{-4}$	$2.080226537527587 \cdot 10^{-5}$
10^{-4}	$2.536821485132733 \cdot 10^{-7}$	$3.604437259962754 \cdot 10^{-5}$	$1.050635123434055 \cdot 10^{-6}$
10^{-5}	$2.943286117122302 \cdot 10^{-7}$	$5.484497075282365 \cdot 10^{-6}$	$2.961321194657043 \cdot 10^{-7}$
10^{-6}	$9.927282323640713 \cdot 10^{-8}$	$1.550798690655468 \cdot 10^{-6}$	$1.017037973103686 \cdot 10^{-7}$
10^{-7}	$1.205197614116022 \cdot 10^{-8}$	$4.341741792173280 \cdot 10^{-9}$	$2.123073561620740 \cdot 10^{-8}$
10^{-8}	$1.329186522489154 \cdot 10^{-11}$	$1.236299527261632 \cdot 10^{-11}$	$1.354559042199227 \cdot 10^{-11}$
10^{-9}	$6.352783184073289 \cdot 10^{-12}$	$3.757421939779895 \cdot 10^{-13}$	$1.662813259533936 \cdot 10^{-13}$
10^{-10}	$3.742506310600856 \cdot 10^{-13}$	$6.411223998569631 \cdot 10^{-5}$	$6.287748822147332 \cdot 10^{-14}$
10^{-11}	$7.914026183371563 \cdot 10^{-10}$	$1.064007225742505 \cdot 10^{-13}$	$3.521887390284772 \cdot 10^{-13}$
10^{-12}	$2.395510913210610 \cdot 10^{-15}$	$2.101736530712850 \cdot 10^{-9}$	$1.813148794351629 \cdot 10^{-13}$
10^{-13}	$3.304287743723571 \cdot 10^{-15}$	$1.957486145353330 \cdot 10^{-13}$	$1.813148794351629 \cdot 10^{-13}$
10^{-14}	$1.059201105169324 \cdot 10^{-15}$	$4.027588919626010 \cdot 10^{-9}$	$1.038062487184869 \cdot 10^{-13}$
10^{-15}	$4.425995829865038 \cdot 10^{-14}$	$4.064600244343451 \cdot 10^{-14}$	$2.809341957484505 \cdot 10^{-14}$

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 due to discretization, roundoff, and so on.[23]

Figure 5.1 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 5.1 lists the data plotted in Figure 5.1. 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 5.3 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 (5.3) than ROMs using a comparable number of equispaced snapshots.

Figure 5.1 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 results because the number of snapshots is a discrete quantity. Figure 5.2 shows that to match approximately the NRMSE, equispaced ROMs tend to require significantly more snapshots (in extreme cases, over 300 times as many), likely because equispaced snapshots are chosen without regard to controlling the error. Figure 5.3 also shows that matching approximately the NRMSE tends to yield smaller adaptive ROMs than equispaced ROMs. Cross-referencing Figure 5.3 with Figure 5.1 suggests that this trend holds because matching the NRMSE of the adaptive ROM with an equispaced ROM at constant SVD truncation ε_{SVD} tends to result in the equispaced ROM being more accurate, although for $\varepsilon_{\text{SVD}} \in \{10^{-i}\}_{i=1}^{5}$,

[23] If the FOM were being solved with implicit methods, the errors in the linear and nonlinear solvers would also contribute.

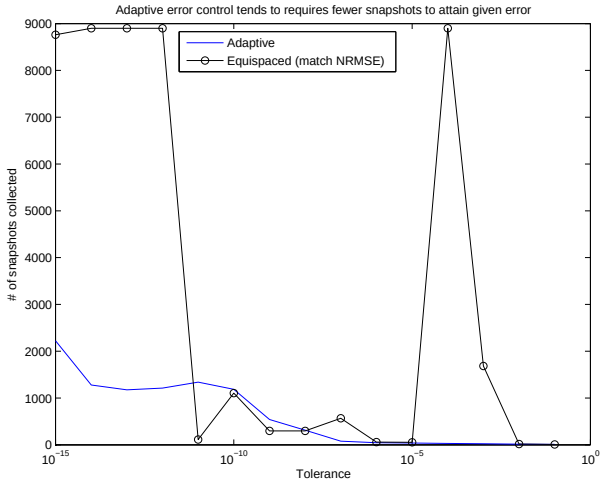

FIG. 5.2. *Plot of number of snapshots collected for ROMs generated for the case study via adaptive snapshot selection and via equispaced snapshot selection while holding the normalized root mean squared error (NRMSE) approximately constant. Note also that for this case study, as the snapshot selection tolerance $\varepsilon_{snapshot}$ 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 8901.*

the equispaced ROM (matching NRMSE) is both less accurate and larger than the corresponding adaptive ROM. These findings for equispaced models that match approximately the NRMSE suggest that adaptive ROM snapshots also contain more information per snapshot.

6. Conclusion and discussion. The convergence study above in Section 5 demonstrate 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 no more snapshots than commonly used equispaced snapshot selection algorithms for a given normalized root-mean-square error, at the cost of a slightly larger ROM basis. 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 the reduce computational cost of general nonlinear full-order models, without incorporating hyperreduction methods (as discussed below), the method proposed here is limited to linear and quadratic full-order models. The viscous Burgers' case study results suggest that adaptive snapshot selection could be effective for full-order

TABLE 5.2
Table of number of snapshots collected for ROMs generated for the case study via three different snapshot selection methods. Data listed here is plotted in Figure 5.2.

	# snapshots for method	
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	Equispaced, match snapshots
10^{-1}	10	10
10^{-2}	15	17
10^{-3}	22	1595
10^{-4}	29	8900
10^{-5}	38	53
10^{-6}	44	58
10^{-7}	77	1180
10^{-8}	312	344
10^{-9}	540	1654
10^{-10}	1186	1716
10^{-11}	1340	1341
10^{-12}	1212	8901
10^{-13}	1174	8901
10^{-14}	1278	8901
10^{-15}	2224	3881

models arising from parabolic PDEs. However, it is known that accurately approximating phenomena such as moving shocks [41, 9] 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 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 (DEIM) [15] or Gauss-Newton with approximated tensors (GNAT) [10, 12]. Hyperreduction methods are noteworthy for reducing the computational complexity (and thus, cost) of evaluating terms in a ROM 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 above can be extended straightforwardly to include time

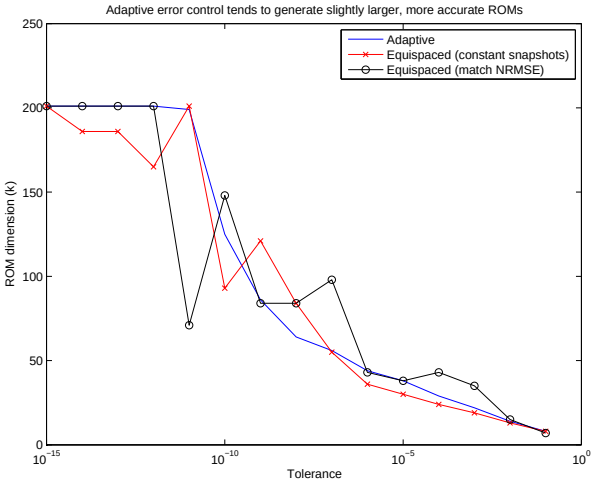

FIG. 5.3. *Plot of ROM dimension (k) for ROMs generated for the case study via adaptive snapshot selection, via equispaced snapshot selection while holding the number of snapshots constant, and via equispaced snapshot selection while holding the normalized root mean squared error (NRMSE) approximately 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* $\varepsilon_{snapshot}$ *approaches zero, all time steps are collected, this statement implies that* k *should approach* n*. For this case study,* $n = 201$*.*

derivative snapshots. Including time derivative snapshots [34, 15, 16, 43, 11] (or instead, difference quotients [31, 36, 37]) 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 [11]. However, both Peng and Mohseni [43] and Chaturantabut and Sorensen [15, 16] 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 [11].

REFERENCES

[1] R. J. ADRIAN, K. T. CHRISTENSEN, AND Z.-C. LIU, *Analysis and interpretation of instantaneous turbulent velocity fields*, Experiments in fluids, 29 (2000), pp. 275–290.

[2] ATHANASIOS C. ANTOULAS, *Approximation of Large-Scale Dynamical Systems*, Advances in Design and Control, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[3] WILLIAM ARRIGHI, GEOFFREY OXBERRY, TANYA VASSILEVSKA, AND KYLE CHAND, *libROM User Guide and Design*, tech. report, Lawrence Livermore National Laboratory, 2015.

[4] C.G. BAKER, K.A. GALLIVAN, AND P. VAN DOOREN, *Low-rank incremental methods for computing dominant singular subspaces*, Linear Algebra and its Applications, 436 (2012), pp. 2866–2888.

[5] GAL BERKOOZ, PHILIP HOLMES, AND JOHN L. LUMLEY, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annual review of fluid mechanics, 25 (1993), pp. 539–575.

[6] T. BRACONNIER, M. FERRIER, J.-C. JOUHAUD, M. MONTAGNAC, AND P. SAGAUT, *Towards an adaptive POD/SVD surrogate model for aeronautic design*, Computers & Fluids, 40 (2011), pp. 195–209.

[7] MATTHEW BRAND, *Incremental singular value decomposition of uncertain data with missing*

TABLE 5.3
Table of ROM dimension k for ROMs generated for the case study via three different snapshot selection methods. Data shown here is also plotted in Figure 5.3.

	ROM dimension (k) for method		
$\varepsilon_{\text{snapshot}} = \varepsilon_{\text{SVD}}$	Adaptive	Equispaced, match	
		snapshots	NRMSE
10^{-1}	8	8	8
10^{-2}	14	13	15
10^{-3}	22	19	34
10^{-4}	29	24	43
10^{-5}	38	30	38
10^{-6}	44	36	43
10^{-7}	56	55	120
10^{-8}	64	84	84
10^{-9}	86	121	199
10^{-10}	125	93	201
10^{-11}	199	201	177
10^{-12}	201	165	201
10^{-13}	201	186	201
10^{-14}	201	186	201
10^{-15}	201	201	201

values, in Computer Vision—ECCV 2002, Springer, 2002, pp. 707–720.

[8] ———, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra and its Applications, 415 (2006), pp. 20–30.

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

[10] KEVIN CARLBERG, CHARBEL BOU-MOSLEH, AND CHARBEL FARHAT, *Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations*, International Journal for Numerical Methods in Engineering, 86 (2011), pp. 155–181.

[11] KEVIN CARLBERG AND CHARBEL FARHAT, *A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems*, International Journal for Numerical Methods in Engineering, 86 (2011), pp. 381–402.

[12] KEVIN CARLBERG, CHARBEL FARHAT, JULIEN CORTIAL, AND DAVID AMSALLEM, *The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows*, Journal of Computational Physics, 242 (2013), pp. 623–647.

[13] Y. CHAHLAOUI, K. GALLIVAN, AND P. VAN DOOREN, *Recursive Calculation of Dominant Singular Subspaces*, SIAM Journal on Matrix Analysis and Applications, 25 (2003), pp. 445–463.

[14] YOUNES CHAHLAOUI AND PAUL VAN DOOREN, *Model reduction of time-varying systems*, in Dimension reduction of large-scale systems, Springer, 2005, pp. 131–148.

[15] SAIFON CHATURANTABUT AND DANNY C. SORENSEN, *Nonlinear Model Reduction via Discrete Empirical Interpolation*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2737–2764.

[16] ———, *A State Space Error Estimate for POD-DEIM Nonlinear Model Reduction*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 46–63.

[17] M. DIHLMANN, M. DROHMANN, AND B. HAASDONK, *Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning*, Proc. of ADMOS, 2011 (2011).

[18] MARTIN DROHMANN AND KEVIN CARLBERG, *The ROMES method for statistical modeling of reduced-order-model error*, arXiv preprint arXiv:1405.5170, (2014).

[19] R. D. FALGOUT, S. FRIEDHOFF, TZ. V. KOLEV, S. P. MACLACHLAN, AND J. B. SCHRODER, *Parallel Time Integration with Multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp. C635–C661.

[20] MARTIN J. GANDER, *50 Years of Time Parallel Time Integration*, in Householder Symposium

XIX June 8-13, Spa Belgium, p. 81.
[21] Gene Golub and Charles Van Loan, *Matrix Computations*, Society for Industrial and Applied Mathematics, 4th ed., 2013.
[22] Martin A. Grepl and Anthony T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, ESAIM: Mathematical Modelling and Numerical Analysis, 39 (2005), pp. 157–181.
[23] Bernard Haasdonk, Markus Dihlmann, and Mario Ohlberger, *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, 17 (2011), pp. 423–442.
[24] Bernard Haasdonk and Mario Ohlberger, *Reduced basis method for finite volume approximations of parametrized linear evolution equations*, ESAIM: Mathematical Modelling and Numerical Analysis, 42 (2008), pp. 277–302.
[25] Ernst Hairer, Gerhard Wanner, and Syvert P. Nørsett, *Solving Ordinary Differential Equations I*, no. 8 in Springer Series in Computational Mathematics, Springer, 1993.
[26] Peter Hall, David Marshall, and Ralph Martin, *Merging and splitting eigenspace models*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22 (2000), pp. 1042–1049.
[27] ———, *Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition*, Image and Vision Computing, 20 (2002), pp. 1009–1016.
[28] Chris Homescu, Linda R. Petzold, and Serban Radu, *Error Estimation for Reduced Order Models of Dynamical Systems*, Technical Report UCRL-TR-201494, Lawrence Livermore National Laboratory, Dec. 2003.
[29] Chris Homescu, Linda R. Petzold, and Radu Serban, *Error Estimation for Reduced-Order Models of Dynamical Systems*, SIAM Journal on Numerical Analysis, 43 (2005), pp. 1693–1714.
[30] Ronald HW Hoppe and Zhiheng Liu, *Snapshot location by error equilibration in proper orthogonal decomposition for linear and semilinear parabolic partial differential equations*, Journal of Numerical Mathematics, 22 (2014), pp. 1–32.
[31] Traian Iliescu and Zhu Wang, *Are the Snapshot Difference Quotients Needed in the Proper Orthogonal Decomposition?*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1221–A1250.
[32] Irina Kalashnikova and Mathew F. Barone, *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, June 2011.
[33] D.A. Knoll and D.E. Keyes, *Jacobian-free Newton–Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.
[34] Tanya Kostova, Geoffrey Oxberry, Kyle Chand, and William Arrighi, *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).
[35] Heinz-Otto Kreiss and Jens Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, no. 47 in SIAM Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 2004.
[36] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for parabolic problems*, Numerische Mathematik, 90 (2001), pp. 117–148.
[37] K. Kunisch and Stefan Volkwein, *Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics*, SIAM Journal on Numerical Analysis, 40 (2002), pp. 492–515.
[38] Karl Kunisch and Stefan Volkwein, *Optimal snapshot location for computing POD basis functions*, ESAIM: Mathematical Modelling and Numerical Analysis, 44 (2010), pp. 509–529.
[39] Oliver Lass and Stefan Volkwein, *Adaptive POD basis computation for parametrized nonlinear systems using optimal snapshot location*, Computational Optimization and Applications, 58 (2014), pp. 645–677.
[40] Randall J. LeVeque, *Finite volume methods for hyperbolic problems*, vol. 31, Cambridge university press, 2002.
[41] David J. Lucia, *Reduced order modeling for high speed flows with moving shocks*, tech. report, DTIC Document, 2001.
[42] A Paul-Dubois-Taine and David Amsallem, *An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models*, International Journal for Numerical Methods in Engineering, accepted (2014), pp. 1–32.
[43] Liqian Peng and Kamran Mohseni, *An Online Manifold Learning Approach for Model Re-*

duction of Dynamical Systems, SIAM Journal on Numerical Analysis, 52 (2014), pp. 1928–1952.

[44] Sivasankaran Rajamanickam, *Efficient Algorithms for Sparse Singular Value Decomposition*, PhD, University of Florida, 2009.

[45] Muruhan Rathinam and Linda R. Petzold, *A New Look at Proper Orthogonal Decomposition*, SIAM Journal on Numerical Analysis, 41 (2003), pp. 1893–1925.

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

[47] John R. Singler, *New POD error expressions, error bounds, and asymptotic results for reduced order models of parabolic PDEs*, SIAM Journal on Numerical Analysis, 52 (2014), pp. 852–876.

[48] Sirovich, Lawrence, *Turbulence and the dynamics of coherent structures. Part I: coherent structures*, Quarterly of Applied Mathematics, 14 (1987), pp. 561–571.

[49] Răzvan Ştefănescu, Adrian Sandu, and Ionel M. Navon, *Comparison of POD reduced order strategies for the nonlinear 2d shallow water equations*, International Journal for Numerical Methods in Fluids, 76 (2014), pp. 497–521.

[50] Kyle Washabaugh, David Amsallem, Matthew Zahr, and Charbel Farhat, *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, AIAA Paper, vol. 2686, 2012.

[51] D. Wirtz, D. C. Sorensen, and B. Haasdonk, *A Posteriori Error Estimation for DEIM Reduced Nonlinear Dynamical Systems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A311–A338.

[52] Matthew J. Zahr, David Amsallem, and Charbel Farhat, *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, pp. 1–11.