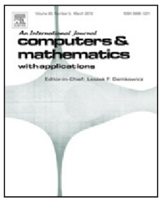
Computers and Mathematics with Applications 75 (2018) 1942–1960

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa



Incremental proper orthogonal decomposition for PDE simulation data

Hiba Fareed [a], John R. Singler [a,*], Yangwen Zhang [a], Jiguang Shen [b]

[a] Department of Mathematics and Statistics, Missouri University of Science and Technology, Rolla, MO 65409, USA
[b] School of Mathematics, University of Minnesota, Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Available online 6 October 2017

Keywords:
Proper orthogonal decomposition
Incremental algorithm
Weighted norm
Finite element method

ABSTRACT

We propose an incremental algorithm to compute the proper orthogonal decomposition (POD) of simulation data for a partial differential equation. Specifically, we modify an incremental matrix SVD algorithm of Brand to accommodate data arising from Galerkin-type simulation methods for time dependent PDEs. The algorithm is applicable to data generated by many numerical methods for PDEs, including finite element and discontinuous Galerkin methods. The algorithm initializes and efficiently updates the dominant POD eigenvalues and modes during the time stepping in a PDE solver without storing the simulation data. We prove that the algorithm without truncation updates the POD exactly. We demonstrate the effectiveness of the algorithm using finite element computations for a 1D Burgers' equation and a 2D Navier–Stokes problem.

1. Introduction

Proper orthogonal decomposition (POD) has been widely used in many applications involving partial differential equations such as aeroelasticity [1], fluid dynamics [2], feedback control [3], PDE constrained optimization and optimal control [4,5], uncertainty quantification [6], and data assimilation [7,8]. In the most basic form, POD is an optimal data approximation method: the POD of a dataset produces a basis (called POD modes) that can be used to optimally reconstruct the data. There are many methods to compute the POD of a dataset; the most basic approaches rely on computing the eigenvalue decomposition or singular value decomposition (SVD) of a matrix constructed using the dataset. The method of snapshots introduced by Sirovich [9] is commonly used to find POD eigenvalues and modes. For more information, see, e.g., [10–12].

For very large datasets, such as datasets arising from simulations of partial differential equations (PDEs), the basic approaches to POD computations become computationally expensive and require a large amount of data storage. Researchers have proposed various methods to deal with the computational complexity (see, e.g., [13–16]), and both the computational complexity and data storage (see, e.g., [17] and the references therein). The latter class of methods are *incremental SVD algorithms*; specifically, the SVD is initialized on a small amount of data and then updated as new data becomes available. This type of incremental algorithm can be easily used in conjunction with a time stepping code for simulating a time dependent PDE; the POD eigenvalues and modes are updated during the time stepping without storing any of the simulation data. For examples of this approach, see, e.g., [18–20].

In this work, we focus on computing the POD of data arising from a Galerkin-type simulation of a PDE. Specifically, the data lies in a Hilbert space and is expressed using a collection of basis functions; therefore, the data can be generated using many

* Corresponding author.
E-mail addresses: hf3n3@mst.edu (H. Fareed), singlerj@mst.edu (J.R. Singler), ywzfg4@mst.edu (Y. Zhang), shenx179@umn.edu (J. Shen).

https://doi.org/10.1016/j.camwa.2017.09.012
0898-1221/© 2017 Elsevier Ltd. All rights reserved.

numerical methods for PDEs, including finite element and discontinuous Galerkin methods. We extend Brand's incremental matrix SVD algorithm [21] to accommodate data of this type. Specifically, we show that the POD of the Hilbert space data expressed in terms of a basis is equivalent to the POD of the vectors in $\mathbb{R}^m$ of the coefficient data with respect to a weighted inner product on $\mathbb{R}^m$ (see Appendix A.1). We consider two approaches to compute the incremental SVD with respect to the weighted inner product: (1) using a Cholesky factor of the weight matrix (Section 3), and (2) without using a Cholesky factor (Section 4). The first approach follows standard POD ideas for weighted inner products (see, e.g., [12]), but requires the computation of the Cholesky factor and also solving linear systems involving the Cholesky factor. In the second approach, we avoid these extra computations by directly extending Brand's incremental matrix SVD algorithm [21] to work with a weighted inner product on $\mathbb{R}^m$. We analyze an idealized version of the second approach that does not involve truncation and prove that it produces the exact SVD with respect to the weighted inner product.

We link the second approach together with (approximate) POD of time varying functions in Section 5, and then present numerical results in Section 6. For the numerical results, we consider computing the POD of finite element simulation data for a 1D Burgers' equation and a 2D Navier–Stokes equation. For the 1D problem and a coarse discretization of the 2D flow problem, we compare the result of the second incremental SVD approach to the true results and find excellent agreement. For a large-scale simulation of the 2D flow problem, we compute the POD without storing all the data by calculating the incremental SVD only. We present conclusions in Section 7.

2. Basic definitions and concepts

We begin by introducing many important definitions and concepts needed throughout this work.

For notational convenience, we adopt Matlab notation herein. Given a vector $u \in \mathbb{R}^n$ and $r \leq n$, let $u(1:r)$ denote the vector of the first r components of u. Similarly, for a matrix $A \in \mathbb{R}^{m\times n}$, we let $A(p:q, r:s)$ denote the submatrix of A consisting of the entries of A from rows $p, \dots, q$ and columns $r, \dots, s$.

2.1. The SVD with respect to a weighted inner product

Let $M \in \mathbb{R}^{m\times m}$ be a symmetric positive definite square matrix. Let $\mathbb{R}^m_M$ denote the Hilbert space $\mathbb{R}^m$ with M-weighted inner product, i.e.,

$$(x, y)_M = y^T M x \quad \text{for all } x, y \in \mathbb{R}^m.$$

Throughout this work, $\mathbb{R}^k$ without a subscript denotes the space with the standard inner product $(x, y) = y^T x$ (i.e., the Euclidean inner product or dot product).

We require two functional analytic concepts for a matrix $P \in \mathbb{R}^{m\times n}$ considered as a mapping $P : \mathbb{R}^n \to \mathbb{R}^m_M$: the Hilbert adjoint operator and the singular value decomposition.

First, the Hilbert adjoint operator of the matrix $P : \mathbb{R}^n \to \mathbb{R}^m_M$ is a matrix $P^* : \mathbb{R}^m_M \to \mathbb{R}^n$ satisfying

$$(Px, y)_M = (x, P^*y) \quad \text{for all } x \in \mathbb{R}^n \text{ and } y \in \mathbb{R}^m_M.$$

We can show $P^* = P^T M$ as follows:

$$(Px, y)_M = (x, P^*y) \quad \Rightarrow \quad y^T M\, Px = (P^*y)^T x \quad \Rightarrow \quad y^T M\, Px = y^T (P^*)^T x.$$

Since this holds for all x, y, we have $M\,P = (P^*)^T$ and therefore $P^* = P^T M$.

Second, since the matrix $P : \mathbb{R}^n \to \mathbb{R}^m_M$ is a compact linear operator, it has a singular value decomposition: the nonzero eigenvalues of $PP^* : \mathbb{R}^m_M \to \mathbb{R}^m_M$ and $P^*P : \mathbb{R}^n \to \mathbb{R}^n$ are equal, and the nonzero singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ of P equal the square roots of those eigenvalues. Also, zero is a singular value of P if either PP^* or P^*P has a zero eigenvalue. Therefore, there are $\max\{m, n\}$ singular values, counting multiplicities. The corresponding orthonormal bases of eigenvectors, $\{\phi_j\}_{j=1}^m \subset \mathbb{R}^m_M$ and $\{\psi_j\}_{j=1}^n \subset \mathbb{R}^n$, are the singular vectors. This gives

$$P\psi_j = \sigma_j \phi_j, \quad P^*\phi_j = \sigma_j \psi_j \quad \text{if } \sigma_j > 0. \tag{1}$$

Note that $\{\phi_j\}_{j=1}^m$ being orthonormal in $\mathbb{R}^m_M$ means

$$(\phi_i, \phi_j)_M = \phi_j^T M \phi_i = \begin{cases} 0, & \text{if } i \neq j, \\ 1, & \text{if } i = j. \end{cases}$$

For more information about the singular value decomposition of operators acting on Hilbert spaces, see, e.g., [22, Chapters VI–VIII], [23, Chapter 30], [24, Sections VI.5–VI.6].

In POD applications, it is typical to only need information about singular vectors corresponding to nonzero singular values. Let $k = \operatorname{rank}(P)$, i.e., P has exactly k positive singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. Let $V = [\phi_1, \phi_2, \dots, \phi_k] \in \mathbb{R}^{m\times k}$ be the matrix of the first k orthonormal eigenvectors of PP^*, and let $W = [\psi_1, \psi_2, \dots, \psi_k] \in \mathbb{R}^{n\times k}$ be the matrix of the first k orthonormal eigenvectors of P^*P. Then (1) gives

$$PW = V\Sigma, \quad P^*V = W\Sigma, \quad \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_k). \tag{2}$$

Since $\{\psi_j\}_{j=1}^k$ is orthonormal in $\mathbb{R}^n$, we have $W^TW = I$. Also, since $\{\phi_j\}_{j=1}^k$ is orthonormal in $\mathbb{R}^m_M$, we have $V^TMV = I$ or $V^*V = I$, where $V^* = V^TM$. Therefore, (2) is equivalent to

$$P = V\Sigma W^T. \tag{3}$$

Since we are primarily interested in the nonzero singular values and corresponding singular vectors, we primarily consider the decomposition (3) for our theoretical results. For the standard matrix case (i.e., without weighted norms), the decomposition (3) is called by various names in the literature and sometimes definitions in different references conflict.[1] In order to potentially avoid confusion, we call this decomposition the *core SVD*.

Definition 2.1. For a matrix $P : \mathbb{R}^n \to \mathbb{R}^m_M$ with exactly k positive singular values, a *core SVD* of P is given by $P = V\Sigma W^T$, where $V \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, and $W \in \mathbb{R}^{n\times k}$ are defined above.

Just like the regular SVD, the core SVD is not unique. For example, the columns of V and W can change sign and $P = V\Sigma W^T$ is still a core SVD of P.

Also, we sometimes consider a core SVD of a matrix in the standard sense, i.e., all inner products are unweighted. To be clear, we call this the standard core SVD.

Below, we give some basic properties of the core SVD.

Proposition 2.2. *Suppose $V \in \mathbb{R}^{m\times k}$ has M-orthonormal columns, $W \in \mathbb{R}^{n\times k}$ has orthonormal columns, and $\Sigma \in \mathbb{R}^{k\times k}$ is a positive diagonal matrix with $\Sigma_{11} \geq \Sigma_{22} \geq \cdots \Sigma_{kk} > 0$. If $P : \mathbb{R}^n \to \mathbb{R}^m_M$ satisfies $P = V\Sigma W^T$, then V, Σ, W give a core SVD of P.*

Proof. First, it is clear that $\text{rank}(P) \leq k$, and therefore P has at most k positive singular values. It is straightforward to check that (2) holds. This implies the k columns of V are eigenvectors of PP^* and the k columns of W are eigenvectors of P^*P, and the corresponding k eigenvalues are the nonzero diagonal entries of Σ. Thus, $P = V\Sigma W^T$ is a core SVD of P. □

We use the next basic result frequently in this work.

Proposition 2.3. *Suppose $V_u \in \mathbb{R}^{m\times k}$ has M-orthonormal columns and $W_u \in \mathbb{R}^{n\times k}$ has orthonormal columns. If $Q \in \mathbb{R}^{k\times k}$ has standard core SVD $Q = V_Q\Sigma_Q W_Q^T$ and $P : \mathbb{R}^n \to \mathbb{R}^m_M$ is defined by $P = V_uQW_u^T$, then*

$$P = V\Sigma_Q W^T, \quad V = V_uV_Q, \quad W = W_uW_Q, \tag{4}$$

is a core SVD of P.

Proof. First, it is clear P has the representation (4). By the above proposition, we only need to show V has M-orthonormal columns and W also has orthonormal columns. This follows directly since $V_u^TMV_u = I$, $W_u^TW_u = I$, $V_Q^TV_Q = I$, and $W_Q^TW_Q = I$. □

2.2. Computing the exact SVD with respect to a weighted inner product

Next, we briefly outline how to compute the exact SVD of a matrix with respect to a weighted inner product using a Cholesky factorization of the weight matrix. In our numerical results in Section 6, we compare the incremental SVD approach in Section 4 to the exact SVD computed using the Cholesky factorization approach discussed here. Note that this Cholesky approach requires storing all of the data, which incremental approaches do not require.

Let $U \in \mathbb{R}^{m\times n}$ be a matrix considered as a mapping $U : \mathbb{R}^n \to \mathbb{R}^m_M$, where $M \in \mathbb{R}^{m\times m}$ is a symmetric positive definite weight matrix. We want to compute the SVD of U as defined in Section 2.1.

Let

$$M = R_M^TR_M$$

be the Cholesky factorization of M, where $R_M \in \mathbb{R}^{m\times m}$ is upper triangular and invertible. Transform the matrix U using the Cholesky factor by defining

$$\tilde{U} = R_MU \in \mathbb{R}^{m\times n}.$$

The standard SVD of $\tilde{U}$ (i.e., the SVD with unweighted inner products) gives the SVD of U with respect to the weighted inner product.

[1] For example, Horn and Johnson [25] call this decomposition the *thin SVD*, but this contradicts with the definition of *thin SVD* in Golub and Van Loan [26]. Furthermore, this definition is called compact SVD in [27]; however, we do not use this terminology to avoid confusion with the SVD of a compact operator.

Proposition 2.4. *Let $U \in \mathbb{R}^{m\times n}$, and suppose $M \in \mathbb{R}^{m\times m}$ is symmetric positive definite with Cholesky factorization $M = R_M^T R_M$ as above. If $\tilde{U} = \tilde{V}\tilde{\Sigma}\tilde{W}^T$ is the standard core SVD of $\tilde{U} = R_M U \in \mathbb{R}^{m\times n}$, then*

$$U = V\Sigma W^T, \quad V = R_M^{-1}\tilde{V}, \quad \Sigma = \tilde{\Sigma}, \quad W = \tilde{W} \tag{5}$$

is the core SVD of $U : \mathbb{R}^n \to \mathbb{R}^m_M$.

Proof. We have

$$U = R_M^{-1}\tilde{U} = V\tilde{\Sigma}\tilde{W}^T,$$

and

$$V^*V = V^T M V = \tilde{V}^T R_M^{-T} M R_M^{-1}\tilde{V} = \tilde{V}^T R_M^{-T} R_M^T R_M R_M^{-1}\tilde{V} = \tilde{V}^T\tilde{V} = I.$$

The result follows directly from Proposition 2.2. □

Algorithm 1 Exact SVD via Cholesky factorization

```
Input: $U \in \mathbb{R}^{m\times n}$ and $M \in \mathbb{R}^{m\times m}$ (symmetric positive definite)
1: $R_M = \mathrm{chol}(M)$
2: $\tilde{U} = R_M U$
3: $[\,\tilde{V}, \Sigma, W\,] = \mathrm{svd}(\tilde{U})$
4: Solve for $V$: $R_M V = \tilde{V}$
5: return $V, \Sigma, W$
```

3. Brand's incremental SVD

Next, we briefly review Brand's incremental SVD algorithm from [21]. The algorithm updates the SVD of a matrix when one or more columns are added to the matrix. A basic implementation of his algorithm has been used for POD computations in [18–20].

Below, we present the modified version of Brand's incremental SVD algorithm from [20] using single column updates. We first consider the standard inner product, and then a weighted inner product using the Cholesky factorization of the weight matrix. We also briefly discuss why it is beneficial to avoid the Cholesky factorization. Then, in Section 4, we propose an extension of Brand's algorithm for a weighted inner product that avoids the Cholesky factorization entirely.

3.1. Standard inner product

Suppose we already have the rank-k truncated SVD of a matrix $U \in \mathbb{R}^{m\times n}$ denoted by

$$U = V\Sigma W^T, \tag{6}$$

where $\Sigma \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the k (ordered) singular values of U on the diagonal, $V \in \mathbb{R}^{m\times k}$ is the matrix containing the corresponding k left singular vectors of U, and $W \in \mathbb{R}^{n\times k}$ is the matrix of the corresponding k right singular vectors of U.

Let $c \in \mathbb{R}^m$ be the single column to be added to U. Our goal is to update the above SVD, i.e., we want to find the SVD of $[\,U\ c\,]$. Furthermore, we want to update (Σ, V, W) without forming the matrices U or $[\,U\ c\,]$.

To do this, let $h \in \mathbb{R}^m$ be the projection of c onto the orthogonal complement of the space spanned by the columns of V, i.e.,

$$h = c - VV^T c.$$

Also, let p be the magnitude of h and let j be the unit vector in the direction of h, i.e.,

$$p = \|h\|, \quad j = h/p.$$

If $p > 0$, we have the fundamental identity

$$\begin{bmatrix} U & c \end{bmatrix} = \begin{bmatrix} V\Sigma W^T & c \end{bmatrix} = \begin{bmatrix} V & j \end{bmatrix} \begin{bmatrix} \Sigma & V^T c \\ 0 & p \end{bmatrix} \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix}^T.$$

As in Proposition 2.3, we can find the SVD of the updated matrix $[\,U\ c\,]$ by finding the SVD of the middle matrix Q in the right hand side of the above identity. Specifically, if

$$Q := \begin{bmatrix} \Sigma & V^T c \\ 0 & p \end{bmatrix} = V_Q \Sigma_Q W_Q^T$$

is the standard core SVD of Q, then the standard core SVD of $[\,U\ c\,]$ is given by

$$\begin{bmatrix} U & c \end{bmatrix} = \left(\begin{bmatrix} V & j \end{bmatrix} V_Q \right) \Sigma_Q \left(\begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q \right)^T. \tag{7}$$

In practice, truncation is performed when p is very small, and also reorthogonalization must be performed on the columns of the updated V and W. We discuss these implementation steps in detail in Section 4.2. We also correct an error in the truncation formulas for the right singular vectors in [20].

Furthermore, as is discussed in [20], we note that only the singular values and left singular vectors need to be computed for many POD applications. However, if one wants to retain an approximation to the data without storing the data, then it is necessary to also compute the right singular vectors.

3.2. *Weighted inner product via Cholesky factorization*

Next, we discuss incrementally computing the SVD of $[\,U\ c\,]$ with respect to a weighted inner product using a Cholesky factorization of the weight matrix. Specifically, consider $U \in \mathbb{R}^{m\times n}$ as a mapping $U : \mathbb{R}^n \to \mathbb{R}^m_M$, where $M \in \mathbb{R}^{m\times m}$ is symmetric positive definite. Let $c \in \mathbb{R}^m_M$ be the column to be added to U. Our goal is to find the SVD of the updated matrix $[\,U\ c\,]$ considered as a mapping $[\,U\ c\,] : \mathbb{R}^{n+1} \to \mathbb{R}^m_M$.

Suppose we already have the rank-k SVD of U with respect to the weighted inner product given by $U = V\Sigma W^T$. As in Section 2.1, let $M = R_M^T R_M$ be the Cholesky factorization of M, and transform the data: Let

$$\tilde{U} = R_M U = \tilde{V}\Sigma W^T, \quad \tilde{V} = R_M V.$$

Next, we update the standard SVD of $\tilde{U}$ by applying Brand's algorithm as outlined above in Section 3.1 to the transformed updated matrix $R_M[\,U\ c\,] = [\,\tilde{U}\ R_M c\,]$. This gives

$$[\,\tilde{U}\ R_M c\,] = R_M[\,U\ c\,] = \tilde{V}_{\text{new}}\Sigma_{\text{new}}W^T_{\text{new}}. \tag{8}$$

We undo the transformation by multiplying (8) on the left by R_M^{-1}. Therefore, in order to find the updated SVD of $[\,U\ c\,]$ we need to rescale the left singular vectors for $\tilde{U}$ as follows:

$$V_{\text{new}} = R_M^{-1}\tilde{V}_{\text{new}}. \tag{9}$$

This gives the updated SVD: $[\,U\ c\,] = V_{\text{new}}\Sigma_{\text{new}}W^T_{\text{new}}$.

The above approach gives an incremental algorithm for the SVD with respect to a weighted inner product; however, the algorithm has a few drawbacks if m is very large:

- A Cholesky factorization of $M \in \mathbb{R}^{m\times m}$ is required.
- The Cholesky factor R_M of M may not be as sparse as M. (However, it may be possible to avoid a significant loss of sparsity by using ordering methods; see, e.g., [28,29].)
- Solving the linear system $R_M V_{\text{new}} = \tilde{V}_{\text{new}}$ is required.

We avoid all of these drawbacks in the next section by modifying Brand's algorithm to deal with the weighted inner product directly.

4. Brand's incremental SVD with respect to a weighted inner product

In this section, we avoid the Cholesky factorization of the weight matrix M and modify Brand's algorithm to treat the weighted inner product case. In the modified algorithm, we do not need to solve any linear systems; we only need to multiply by the weight matrix M. In large-scale applications involving partial differential equations, it is common for M to be sparse; therefore, multiplying by M is a minor computational cost. The modified algorithm has a similar computational cost to the standard algorithm if M is sparse or if multiplying M by a vector can be computed quickly.

We begin in Section 4.1 by describing an idealized version of the incremental SVD algorithm, and we prove it produces the exact core SVD. Then we discuss implementation details in Section 4.2.

4.1. *Idealized algorithm without truncation*

Suppose we have an exact core SVD of a matrix $U : \mathbb{R}^n \longrightarrow \mathbb{R}^m_M$, and our goal is to update the core SVD when we add a column $c \in \mathbb{R}^m_M$ to U. Furthermore, we want to update the core SVD without forming U or $[\,U\ c\,]$.

First, we propose an idealized version of the algorithm by modifying Brand's algorithm to work in the Hilbert space structure of the weighted inner product space $\mathbb{R}^m_M$ from Section 2. We use Hilbert adjoint operators of matrices, and also the weighted norm $\|x\|_M = x^T M x^{1/2}$ for $x \in \mathbb{R}^m_M$.

The idealized algorithm is given in the following theorem. Again, we present implementation details in Section 4.2.

Theorem 4.1. *Let $U : \mathbb{R}^n \longrightarrow \mathbb{R}^m_M$, and suppose $U = V\Sigma W^T$ is the exact core SVD of U, where $V^T MV = I$ for $V \in \mathbb{R}^{m\times k}$, $W^T W = I$ for $W \in \mathbb{R}^{n\times k}$, and $\Sigma \in \mathbb{R}^{k\times k}$. Let $c \in \mathbb{R}^m_M$ and define*

$$h = c - VV^*c, \quad p = \|h\|_M, \quad Q = \begin{bmatrix} \Sigma & V^*c \\ 0 & p \end{bmatrix},$$

where $V^* = V^T M$. If $p > 0$ and the standard core SVD of $Q \in \mathbb{R}^{k+1\times k+1}$ is given by

$$Q = V_Q\, \Sigma_Q\, W_Q^T, \tag{10}$$

then the core SVD of $[\,U\,c\,] : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^m_M$ is given by

$$[\,U\,c\,] = V_u \Sigma_Q W_u^T,$$

where

$$V_u = [\,V\,j\,]\,V_Q, \quad j = h/p, \quad W_u = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q.$$

Proof. Since $j = h/p = (c - VV^*c)/p$, we have $c = VV^*c + jp$. This gives

$$\begin{aligned}
[\,U\,c\,] &= [\,V\Sigma W^T\ c\,] \\
&= [\,V\Sigma W^T\ \ VV^*c + jp\,] \\
&= [\,V\,j\,] \begin{bmatrix} \Sigma W^T & V^*c \\ 0 & p \end{bmatrix} \\
&= [\,V\,j\,] \begin{bmatrix} \Sigma & V^*c \\ 0 & p \end{bmatrix} \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix}^T.
\end{aligned}$$

Next, note

$$[\,V\,j\,]^T M [\,V\,j\,] = \begin{bmatrix} V^T MV & V^T Mj \\ (V^T Mj)^T & j^T Mj \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

since $V^T MV = V^*V = I$ by assumption,

$$V^T Mj = V^*j = V^*(c - VV^*c)/p = (V^*c - V^*c)/p = 0,$$

and

$$j^T Mj = \frac{\|h\|_M^2}{p^2} = \frac{\|h\|_M^2}{\|h\|_M^2} = 1.$$

Also, since $W^T W = I$,

$$\begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}.$$

Proposition 2.3 gives the result. □

4.2. Algorithm details: initialization, truncation, and orthogonalization

In Section 4.1, we demonstrated an idealized approach to computing the SVD with respect to a weighted inner product incrementally, i.e., by adding one column at a time. Next, we give implementation details concerning initialization, truncation, and orthogonalization.

Initialization. We initialize the SVD with a single column of data c by setting

$$\Sigma = \|\, c \,\|_M = (c^T Mc)^{1/2}, \quad V = c\Sigma^{-1}, \quad W = 1.$$

We note that although the matrix M may be positive definite in theory, small round off errors may cause $c^T Mc$ to be very small and negative in practice. Therefore, throughout this work, when computing the weighted norm we use absolute values under the square root. For example, we actually set $\Sigma = (|c^T Mc|)^{1/2}$. We also note that c should be nonzero to initialize. The procedure is given in Algorithm 2.

Truncation part 1. The exact SVD update result in Theorem 4.1 requires $p = \|c - VV^*c\|_M > 0$. When p is small enough, i.e., $p <$ tol for a given tolerance tol, we extend the truncation update approach of Brand [21] to the current weighted norm framework.

Algorithm 2 Initialize incremental SVD with respect to weighted inner product

```
Input: $c \in \mathbb{R}^{m\times 1}$, $c \neq 0$, $M \in \mathbb{R}^{m\times m}$ (symmetric positive definite)
1: $\Sigma = (|c^T M c|)^{1/2}$
2: $V = c\Sigma^{-1}$
3: $W = 1$
4: return $V, \Sigma, W$
```

If $p <$ tol, we approximate and set $p = 0$. Since $p = \|c - VV^*c\|_M$, this implies $c = VV^*c$. This gives

$$\begin{aligned}[\,U\,c\,] &= [\,V\Sigma W^T\ c\,]\\ &= [\,V\Sigma W^T\ VV^*c\,]\\ &= [\,V\,0\,]\begin{bmatrix}\Sigma W^T & V^*c\\ 0 & 0\end{bmatrix}\\ &= [\,V\,0\,]\begin{bmatrix}\Sigma & V^*c\\ 0 & 0\end{bmatrix}\begin{bmatrix}W & 0\\ 0 & 1\end{bmatrix}^T.\end{aligned}$$

Similarly to Section 4.1, define $Q \in \mathbb{R}^{k+1\times k+1}$ by

$$Q = \begin{bmatrix}\Sigma & V^*c\\ 0 & 0\end{bmatrix}.$$

If the full standard SVD of Q is given by $Q = V_Q\Sigma_Q W_Q^T$, where $V_Q, \Sigma_Q, W_Q \in \mathbb{R}^{k+1\times k+1}$, then

$$\begin{aligned}Q &= V_Q\begin{bmatrix}\Sigma_{Q_{(1:k,1:k)}} & 0\\ 0 & 0\end{bmatrix}W_Q^T\\ &= V_Q\begin{bmatrix}\Sigma_{Q_{(1:k,1:k)}}\left(W_{Q_{(:,1:k)}}\right)^T\\ 0\end{bmatrix}\\ &= V_{Q_{(:,1:k)}}\Sigma_{Q_{(1:k,1:k)}}\left(W_{Q_{(:,1:k)}}\right)^T.\end{aligned}$$

This gives

$$\begin{aligned}[\,U\,c\,] &= [\,V\,0\,]Q\begin{bmatrix}W & 0\\ 0 & 1\end{bmatrix}^T\\ &= [\,V\,0\,]V_{Q_{(:,1:k)}}\Sigma_{Q_{(1:k,1:k)}}\left(W_{Q_{(:,1:k)}}\right)^T\begin{bmatrix}W & 0\\ 0 & 1\end{bmatrix}^T\\ &= VV_{Q_{(1:k,1:k)}}\Sigma_{Q_{(1:k,1:k)}}\left(\begin{bmatrix}W & 0\\ 0 & 1\end{bmatrix}W_{Q_{(:,1:k)}}\right)^T.\end{aligned}$$

This suggests the following update

$$V \longrightarrow VV_{Q_{(1:k,1:k)}},\quad \Sigma \longrightarrow \Sigma_{Q_{(1:k,1:k)}},\quad W \longrightarrow \begin{bmatrix}W & 0\\ 0 & 1\end{bmatrix}W_{Q_{(:,1:k)}}.$$

We note that the rank of the SVD is not increased even though we added a column. Furthermore, the formula for the update of W given here corrects an error in [21] (the matrix $[W, 0;\ 0, 1]$ is missing from the update formula).

Orthogonalization. In the idealized algorithm in Section 4.1, the SVD update yields orthonormal left and right singular vectors. However, in practice, small numerical errors cause a loss of orthogonality. Following [20], we reorthogonalize when the weighted inner product between the first and last left singular vectors is greater than some tolerance. Specifically, we apply a modified M-weighted Gram–Schmidt procedure with reorthogonalization to the columns of V; the columns of the resulting matrix are orthonormal with respect to the M-weighted inner product. See Algorithm 3, which is a modification for the weighted inner product of the Gram–Schmidt code in [30, Algorithm 6.11, page 307, Section 6.5.6].

Truncation part 2. The orthogonalization step described above is a large part of the computational cost of the incremental SVD algorithm. If the incremental SVD update is to be repeated for a large number of added columns, the number of nonzero singular values can increase quickly and the computational cost of the orthogonalization steps will be large. In such a case, it is important to keep only the singular values of interest to the application. Usually, singular values very near zero (and their corresponding singular vectors) are not required for POD applications. Therefore, during an incremental SVD update, we keep only the singular values (and corresponding singular vectors) above a user specified tolerance.

Algorithm 3 Modified M-weighted Gram–Schmidt with reorthogonalization

```
Input: $V \in \mathbb{R}^{m\times r}, M \in \mathbb{R}^{m\times m}$
 1: $Q = V$
 2: for $k = 1$ to $m$ do
 3:   for $i = 1$ to $k-1$ do
 4:     for $t = 1$ to 2 (reorthogonalize) do
 5:       $E = Q(:,i)^T MQ(:,k)$
 6:       $Q(:,k) = Q(:,k) - EQ(:,i)$
 7:       $R(i,k) = R(i,k) + E$
 8:     end for
 9:   end for
10:   $R(k,k) = \mathrm{sqrt}(Q(:,k)^T MQ(:,k))$
11:   $Q(:,k) = Q(:,k)/R(k,k)$
12: end for
13: return $Q$
```

Complete implementation. Our implementation of the incremental SVD update algorithm for a weighted inner product is given in Algorithm 4. Our implementation is modeled after the algorithm in [20] (without a weighted inner product). As is noted in [20], for many POD applications only the singular values and left singular vectors need to be updated and stored. However, if one desires to be able to approximately reconstruct the entire dataset (without storing the data), then one must update and store the singular values and both left and right singular vectors.

Algorithm 4 Incremental SVD with weighted inner product

```
Input: $V \in \mathbb{R}^{m\times k}, \Sigma \in \mathbb{R}^{k\times k}, W \in \mathbb{R}^{n\times k}, c \in \mathbb{R}^m, M \in \mathbb{R}^{m\times m}$, tol, $\mathrm{tol}_{\mathrm{sv}}$
    % Prepare for SVD update
 1: $d = V^T Mc, p = \mathrm{sqrt}(|(c - Vd)^T M(c - Vd)|)$
 2: if $(p < \mathrm{tol})$ then
 3:   $Q = \begin{bmatrix} \Sigma & d \\ 0 & 0 \end{bmatrix}$
 4: else
 5:   $Q = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}$
 6: end if
 7: $[\, V_Q, \Sigma_Q, W_Q \,] = \mathrm{svd}(Q)$
    % SVD update
 8: if $(p < \mathrm{tol})$ or $(k \geq m)$ then
 9:   $V = VV_{Q(1:k,1:k)}, \Sigma = \Sigma_{Q(1:k,1:k)}, W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_{Q(:,1:k)}$
10: else
11:   $j = (c - Vd)/p$
12:   $V = [V\, j]V_Q, \Sigma = \Sigma_Q, W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q$
13:   $k = k + 1$
14: end if
    % Orthogonalize if necessary
15: if ( $|V^T_{(:,\mathrm{end})} MV_{(:,1)}| > \min(\mathrm{tol}, \mathrm{tol} \times m)$) then
16:   $V = \mathrm{modifiedGSweighted}(V, M)$                                % Algorithm 3
17: end if
    % Neglect small singular values: truncation
18: if $(\Sigma_{(r,r)} > \mathrm{tol}_{\mathrm{sv}})$ and $(\Sigma_{(r+1,r+1)} < \mathrm{tol}_{\mathrm{sv}})$ then
19:   $\Sigma = \Sigma_{(1:r,1:r)}, \quad V = V_{(:,1:r)}, \quad W = W_{(:,1:r)}$
20: end if
21: return $V, \Sigma, W$
```

5. Incremental POD for time varying functions

POD is often used to extract mode shapes or basis functions from solutions of time dependent PDEs. In this case, the dataset consists of a time varying function taking values in a Hilbert space X with inner product $(\cdot,\cdot)$ and corresponding norm $\|\cdot\|$. More information about POD for this type of data can be found in, e.g., [11,12,31,32].

In this section, we show how to compute the POD of the data in this setting using the modified incremental SVD algorithm proposed in Section 4. We focus on approximating continuous POD in Section 5.1, and then consider data expanded in basis functions for X in Section 5.2. We also briefly consider data generated by a finite difference method in Section 5.3.

5.1. Approximate continuous POD

Let w be in $L^2(0,T;X)$, i.e., roughly, $\int_0^T \|w(t)\|^2\,dt < \infty$. Define the continuous POD operator $Z : L^2(0,T) \to X$ by

$$Zg = \int_0^T w(t)g(t)\,dt.$$

In practice, we typically only have access to the data at discrete points in time $\{t_j\}_{j=1}^{s+1}$, where $0 \le t_1 < t_2 < \cdots < t_{s+1} \le T$. For $j = 1,\ldots,s$, let $\delta_j = t_{j+1} - t_j$ denote the jth time step. As is well-known (see, e.g., [11,12]), we rescale the data below by $\delta_j^{1/2}$ in order to arrive at a discrete POD operator.

Approximate the time integral in the definition of the POD operator Z by a Riemann sum with left hand endpoint:

$$\begin{aligned} Zg &\approx \sum_{j=1}^{s} w(t_j)\,\delta_j\,g(t_j) \\ &= \sum_{j=1}^{s} u_j f_j, \quad u_j = \delta_j^{1/2} w(t_j), \quad f_j = \delta_j^{1/2} g(t_j). \end{aligned}$$

Therefore, define the (discrete) POD operator $K : \mathbb{R}^s \to X$ by

$$Kf = \sum_{j=1}^{s} u_j f_j, \quad f = [f_1,\ldots,f_s]^T.$$

5.2. Data expanded in basis functions

In POD applications, a common way to collect data from a time dependent PDE is via numerical simulation. Many approximate solution methods for PDEs (such as finite element and discontinuous Galerkin methods) are Galerkin-type, i.e., the approximate solution data is expressed in terms of a basis of a finite dimensional subspace of X. Let $\{\phi_k\}_{k=1}^m \subset X$ be a collection of linearly independent basis functions, and assume the rescaled data $u_j = \delta_j^{1/2} w(t_j) \in X$ can be expressed as

$$u_j = \delta_j^{1/2} w(t_j) = \sum_{k=1}^{m} U_{k,j}\phi_k, \quad \text{for } j = 1,\ldots,s.$$

In Appendix A.1, we show that the SVD of $K : \mathbb{R}^s \to X$ can be computed using the SVD of the coefficient data matrix $U : \mathbb{R}^s \to \mathbb{R}^m_M$, where the weight matrix $M \in \mathbb{R}^{m\times m}$ has entries $M_{j,k} = (\phi_j,\phi_k)$. We leave the precise statement of the result to Appendix, but we note that if $\{\sigma_i, f_i, c_i\}$ are the nonzero singular values and corresponding singular vectors of $U : \mathbb{R}^s \to \mathbb{R}^m_M$, then $\{\sigma_i, f_i, x_i\}$ are the nonzero singular values and corresponding singular vectors of K, where

$$x_i = \sum_{k=1}^{m} c_{i,k}\phi_k$$

and $c_{i,k}$ is the kth entry of the vector c_i.

Let U_j denote the jth column of the coefficient data matrix U. Algorithm 5 gives the incremental POD algorithm for time varying data. As mentioned previously, for many POD applications only the singular values and left singular vectors (the POD modes) need to be updated and stored. However, if one also updates and stores the right singular vectors then it is possible to approximately reconstruct the entire dataset without storing the data. Furthermore, we note that the POD eigenvalues are the squares of the POD singular values.

Remark 5.1. Many researchers remove the average of the data, and then compute the POD of this new data. Such a computation can be performed incrementally without storing the data. We give a brief overview of the algorithm with a weighted inner product in Appendix A.2.

Algorithm 5 Time varying incremental POD

```
Input: $\{\delta_j\}, \{U_j\}, M \in \mathbb{R}^{m\times m}$, tol, tol$_{\text{sv}}$          % $\delta_j, U_j, M$ as described above
1: $[V, \Sigma, W] = \text{initializeSVD}(U_1, M)$          % Algorithm 2
2: for $j = 2, \ldots, s$ do
3:     $[V, \Sigma, W] = \text{incrementalSVD}(V, \Sigma, W, U_j, \text{tol}, \text{tol}_{\text{sv}}, M)$          % Algorithm 4
4: end for
5: $W = \text{diag}(\delta)^{-1/2} W$          % undo rescaling, $\delta = [\delta_1, \ldots, \delta_s]$
6: return $V, \Sigma, W$
```

In some of our numerical results, we compare the time varying incremental SVD with the exact time varying SVD computed using the Cholesky factorization of M as in Section 2.2. We must modify Algorithm 1 for the exact SVD to account for the rescaling by the square roots of the time steps. Let $D = \text{diag}(\delta)$, where $\delta = [\delta_1, \ldots, \delta_s]$. The modified exact SVD algorithm is shown in Algorithm 6. Again, note that this exact algorithm requires storing all of the data, which incremental algorithms do not require.

Algorithm 6 Exact time varying SVD via Cholesky factorization

```
Input: $U = [U_1, \ldots, U_s], M, D$          % $U_j, M, D$ as described above
1: $R_M = \text{chol}(M)$
2: $\tilde{U} = R_M U$
3: $[\tilde{V}, \Sigma, \tilde{W}] = \text{svd}(\tilde{U})$
4: Solve for $V$: $R_M V = \tilde{V}$
5: $W = D^{-1/2} \tilde{W}$
6: return $V, \Sigma, W$
```

5.3. Data from a finite difference method

Although we focus on Galerkin-type simulation methods for PDEs in this work, we briefly consider incremental POD for data generated by one type of numerical method for PDEs that is not of Galerkin-type: finite difference methods. The key is to focus on the Hilbert space inner product and its approximation.

Suppose a finite difference method is used to approximate the solution of a scalar time dependent PDE on a bounded domain $\Omega \subset \mathbb{R}^d$, and the goal is to approximate the POD of the data with respect to the $L^2(\Omega)$ inner product. For a function u, let $u_f \in \mathbb{R}^m$ denote the vector of approximations to the function u evaluated at the m finite difference nodes. Then

$$(u, v)_{L^2(\Omega)} = \int_\Omega u(x)\, v(x)\, dx \approx \sum_{i=1}^{m} \eta_i u_{f,i} v_{f,i} = v_f^T M u_f,$$

where $\{\eta_i\}_{i=1}^m$ are positive quadrature weights and $M = \text{diag}(\eta_1, \ldots, \eta_m)$. It is possible to apply the modified incremental SVD algorithm in this work with the weight matrix M. However in this case the data can be rescaled by the square roots of the quadrature weights and Brand's incremental SVD algorithm can be used without a weighted inner product. Once the POD modes are computed, the modes must be multiplied by the diagonal matrix $M^{-1/2}$ so that they are orthonormal with respect to the M weighted inner product.

If instead the goal is to approximate the POD of the data with respect to the $H^1(\Omega)$ inner product, then we have

$$(u, v)_{H^1(\Omega)} = \int_\Omega u\, v + \nabla u \cdot \nabla v\, dx \approx v_f^T M u_f.$$

Here, the matrix M is obtained by approximating the integral using quadrature with positive weights and approximating the gradients by finite difference approximations. In this case, the weight matrix M is not diagonal and so the rescaling idea described above is not applicable; however, the incremental SVD algorithm with weight matrix M can be applied.

6. Numerical results

In this section, we present numerical results for the incremental POD algorithm applied to time varying finite element solution data for two PDEs: (i) a 1D Burgers' equation, and (ii) a 2D Navier–Stokes equation. The first problem serves as a small test problem with varying time steps. For the second problem, we consider fixed time steps and both small-scale and large-scale computations. For the small-scale problems, we stored all of the simulation data to compare the standard SVD (computed using Algorithm 6) with the incremental SVD (computed using Algorithm 5). For the large-scale problem, we did not store the simulation data and only computed the incremental SVD using Algorithm 5.

For all of the examples reported here, we used the standard L^2 inner product for the POD computations. This corresponds to the matrix SVD with respect to a weighted inner product, where the weight matrix M is the standard finite element mass

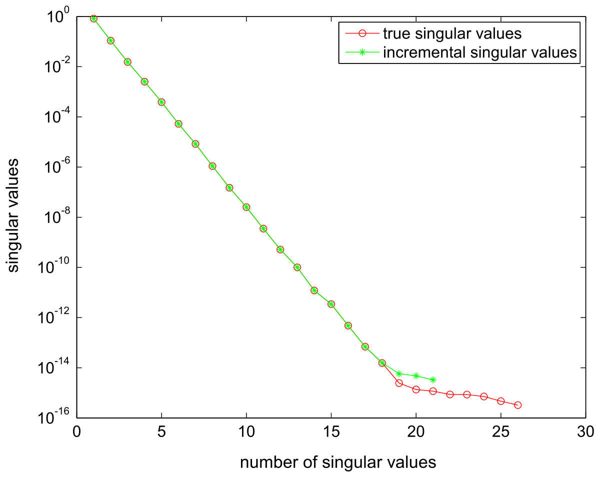


Fig. 1. Example 1, $Re = 20$: Exact versus incremental singular values.

matrix. For the 1D Burgers' equation example, we also tested the incremental POD using the standard H^1 inner product, which yields a different matrix M. We do not report these results here; we found the algorithm performance is similar in this case to the L^2 performance. We also tested the incremental approach to POD for the removed average data for the 1D Burgers' equation example (as outlined in Appendix A.2). Again, we found that the performance of the algorithm is similar to the other cases, and so we do not report the results here.

6.1. Example 1: 1D Burgers' equation

We begin with a small test problem. Consider 1D Burgers' equation with zero Dirichlet boundary conditions

$$\frac{\partial w}{\partial t}(t,x) + w(t,x)\,\frac{\partial w}{\partial x}(t,x) = \frac{1}{Re}\,\frac{\partial^2 w}{\partial x^2}(t,x), \qquad -1 < x < 1.$$

We used piecewise linear finite elements with 1000 equally spaced notes to approximate the solution of this PDE with $Re = 20$ and initial condition $w(0,x) = \sin(\pi x)$. We used Matlab's `ode23s` to approximate the solution of the resulting nonlinear ODE system on the time interval $0 \leq t \leq 2$. The solver returned the approximate solution at 26 points in time in that interval; the time steps were not equally spaced. At each time point, the finite element coefficient vector had length 998.

For the incremental POD algorithm, we set $\text{tol} = 10^{-14}$ and $\text{tol}_{\text{sv}} = 10^{-15}$. Recall, the first tolerance tol is the truncation tolerance for the incremental algorithm, while the second tolerance tol_{sv} is the truncation tolerance for the singular values. In this example and in the examples below, we set tol_{sv} very small in order to test the accuracy of the very small singular values and the corresponding singular vectors; in practice, a very small singular value tolerance is likely rarely needed.

Fig. 1 shows the exact versus the incrementally computed singular values. We see excellent agreement for all singular values down to near the singular value tolerance (10^{-14}). Note that the incremental SVD algorithm only returns 21 singular values due to the singular value truncation. A few of the exact and incrementally computed right singular vectors and POD modes are shown in Figs. 2 and 3. Again, we see excellent agreement.

Next, Fig. 4 shows the weighted norm error between the exact and incrementally computed POD modes. The errors for the dominant POD modes (corresponding to the largest singular values) are extremely small. The errors in the POD modes increase slowly and monotonically as the corresponding singular values approach zero. The number of highly accurate POD modes is quite large; the first 12 modes are computed to an accuracy level of at least 10^{-5}. The 12th singular value is $O(10^{-9})$. In many POD applications, POD modes are not required for POD singular values that are this small. (Recall, the POD eigenvalues are the squares of the POD singular values.) The incremental POD algorithm works very well for this problem.

6.2. Example 2: 2D Navier–Stokes equation

For our second example, we consider a 2D laminar flow around a cylinder with circular cross-section [33]. The flow is governed by the time dependent incompressible Navier–Stokes equations with Reynolds number $Re = 100$, and we consider a rectangular spatial domain of length 2.2 and width 0.41. The diameter of the cylinder is 0.1, and it is centered at the point (0.2, 0.2). For the initial condition, we take the steady state solution of the same problem with Reynolds number 40 (instead of 100). On the right boundary of the rectangle (the outlet), we consider stress free boundary conditions. The boundary

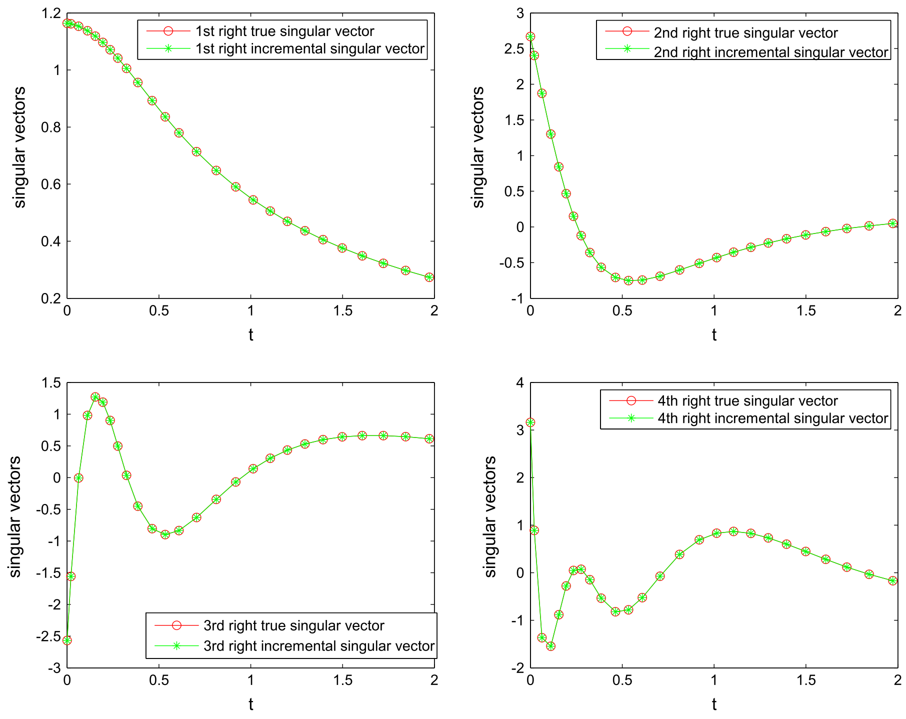


Fig. 2. Example 1, $Re = 20$: Exact versus incremental right singular vectors.

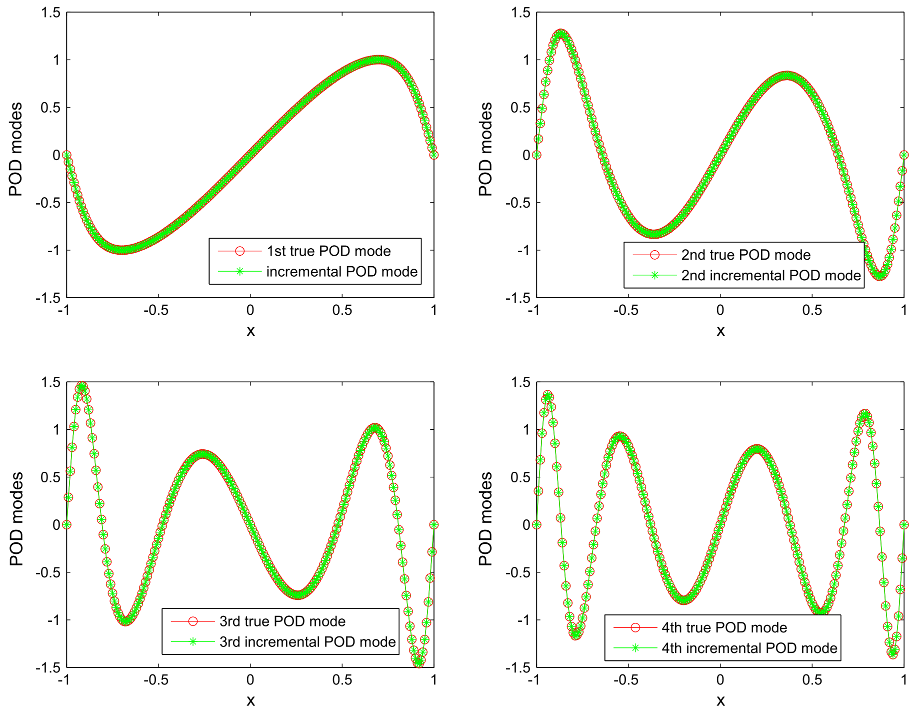


Fig. 3. Example 1, $Re = 20$: Exact versus incremental POD modes.

conditions on all other walls are Dirichlet boundary conditions. The Dirichlet velocity data on the left wall of the rectangle (the inlet) is $(\frac{6y(0.41-y)}{0.41^2}, 0)$. The Dirichlet data on all other boundaries is zero.

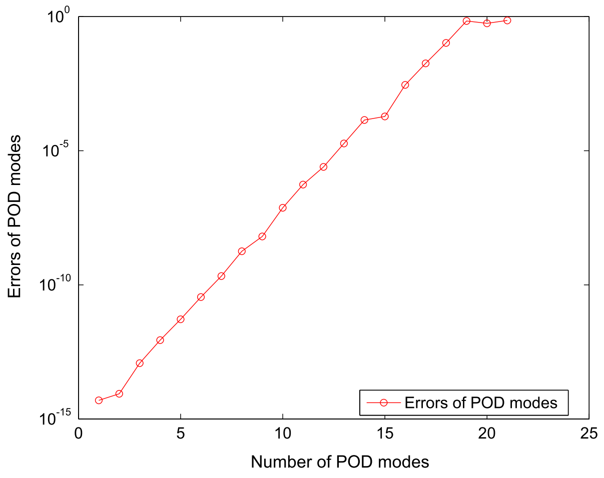


Fig. 4. Example 1, $Re = 20$: Errors between true and incremental POD modes.

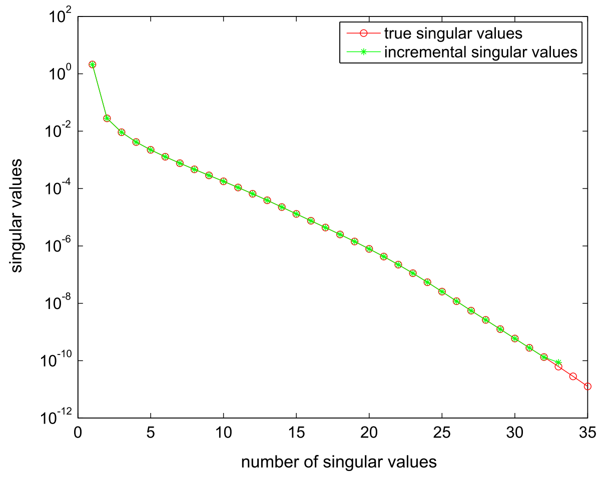


Fig. 5. Example 2, $Re = 100$: Exact versus incremental singular values.

The primary goal of this example is to test the incremental POD algorithm on a problem with more complex solution behavior than the first example (the 1D Burgers' equation). First, we use a coarse grid and a relatively small number of time steps over a short time interval in order to compute the exact errors compared to the exact SVD (with respect to the weighted inner product). We do not attempt to simulate over a longer time period in order to obtain similar numerical results to POD works in the literature (see, e.g., [34,35]).

For the simulation, we consider the time interval $0 \leq t \leq 1$ and time step 0.01. The finite element mesh is generated by Triangle [36,37] with local refinement near the cylinder; also, the mesh is polygonal and only approximately fits the circular boundary of the cylinder. We used standard Taylor–Hood elements, and backward Euler for the time stepping for simplicity.

We first consider a coarse mesh. At each time point, the velocity finite element coefficients are vectors of length 55552. We have 101 total solution snapshots. For the incremental SVD computation, we take $\text{tol} = 10^{-10}$ and the singular value tolerance $\text{tol}_{\text{sv}} = 10^{-12}$.

The incremental SVD algorithm returns 33 singular values and corresponding singular vectors. Fig. 5 shows the exact versus the incremental singular values. We see excellent agreement for all singular values down to near the singular value tolerance (10^{-10}). The first four exact and incrementally computed right singular vectors are shown in Fig. 6, and the agreement is again excellent. Fig. 7 shows the horizontal and vertical components of the 1st, 5th, and 10th velocity POD modes.

Fig. 8 shows the weighted norm error between the exact and incrementally computed velocity POD modes. The error behaves in a similar fashion to the POD mode error in the first example (Fig. 4). Again, the errors for the dominant POD modes are extremely small, and the errors increase slowly and monotonically as the corresponding singular values approach zero.

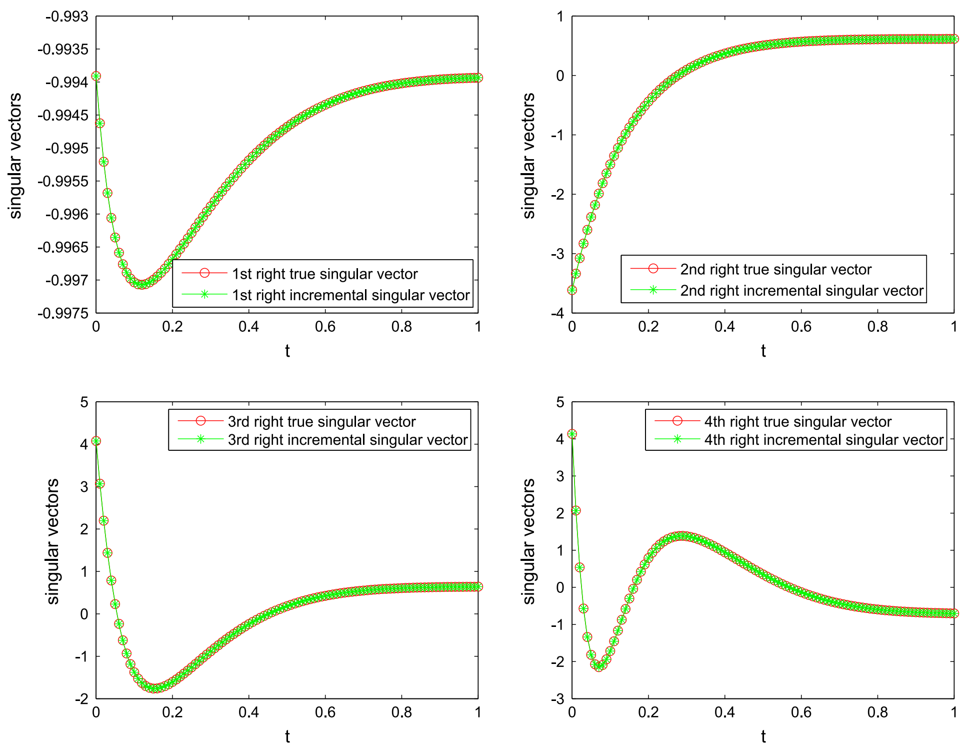


Fig. 6. Example 2, $Re = 100$: Exact versus incremental right singular vectors.

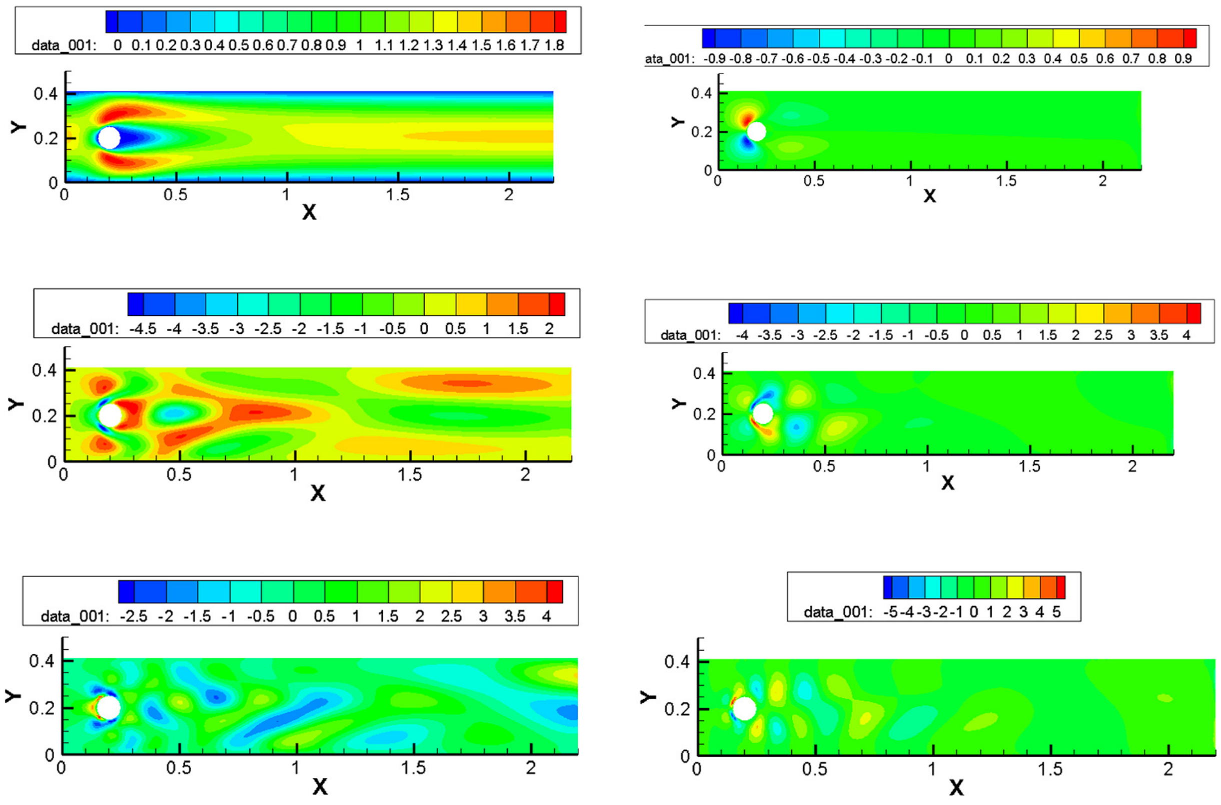


Fig. 7. Example 2, $Re = 100$: 1st, 5th, and 10th incremental velocity POD modes (from top to bottom); horizontal components are on the left, and vertical components are on the right.

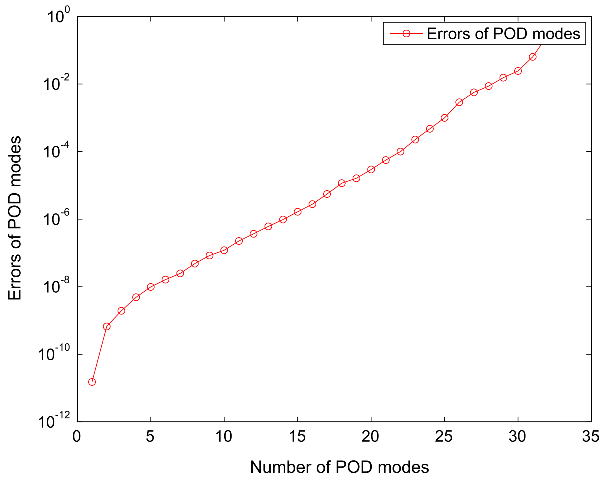


Fig. 8. Example 2, $Re = 100$: Errors between true and incremental POD modes.

Furthermore, there are a large number of highly accurate POD modes. Again, the incremental POD algorithm is very accurate for this problem.

We also tested the same problem with a smaller time step of 0.001 (instead of 0.01); this gives 1001 solution snapshots. We also reduced the algorithm tolerance to $\text{tol} = 10^{-12}$. Using the same the singular value tolerance $\text{tol}_{\text{sv}} = 10^{-12}$, the algorithm returned 97 singular values and corresponding singular vectors. We again found that the incremental approach gave accurate results (not shown).

Next, we return to the larger time step 0.01, but now use a fine mesh for the finite element discretization. Each of the 101 flow velocity snapshots has a finite element coefficient vector of length nearly 2 million (1 978 904). In this case, we did not store the solution data or compute the exact SVD; we only performed the POD computations incrementally. Also, we compared the incremental SVD to the incremental SVD computed on the coarse finite element mesh (with the same time step). We found both incremental SVD computations gave similar singular values and singular vectors (not shown), as we would expect from POD theory. Specifically, the finite element solution should converge to the solution of the PDE as the mesh is refined; therefore, the POD eigenvalues and modes must converge (see, e.g., [11,12,32]).

7. Conclusion

We extended Brand's incremental SVD algorithm [21] to treat data expanded in basis functions from a Hilbert space. Many numerical methods for PDEs generate data of this form. Specifically, we reformulated Brand's matrix algorithm in a weighted norm setting using functional analytic techniques. We proved that an idealized version of the algorithm exactly updates the SVD when a new column is added to the data. We also considered time varying data by incorporating the quadrature on the time integral into the incremental approach.

We used the left singular vectors to compute the POD modes for the collected data. Standard methods for computing the POD modes require storing the whole large dataset; in contrast, using an incremental SVD algorithm only requires storing one snapshot of the data at a time. Therefore, the incremental approach drastically reduces the memory requirement for computing the POD of the data. Furthermore, the computational cost of the incremental approach is also much lower than standard approaches. Moreover, by truncating small singular values (and corresponding singular vectors) during the incremental update, we reduce the computational cost of orthgonalizing the stored singular vectors.

We tested our approach on finite element simulation data with the L^2 inner product for a 1D Burgers' equation and a 2D Navier–Stokes equation. For the small-scale computational cases, we compared the incremental SVD results with the exact SVD and found excellent agreement. We also found that the incremental algorithm worked very well using a different inner product and also if we removed the average from the data (again with an incremental approach without storing the data). We also tested the algorithm on a fine mesh for the Navier–Stokes problem with nearly 2 million velocity unknowns.

Our implementation of the incremental SVD algorithm with respect to a weighted inner product is based on the implementation of the unweighted incremental SVD algorithm in [20], and a parallel implementation of this algorithm has been developed [38]. It is possible that implementation ideas from [38] can be used to develop a parallel implementation of the proposed algorithm; we leave this to be explored elsewhere.

Although we showed the proposed algorithm is exact in an idealized case, we did not perform an error analysis of the algorithm with truncation in this work. In our numerical experiments, we found that we obtained very accurate results for many choices of the truncation tolerances, as long as the tolerances were chosen relatively small (such as 10^{-8} and smaller). An analysis may provide more insight into the accuracy of the algorithm with truncation and the choices of the tolerances; we leave this for future work.

Acknowledgments

J. Singler and Y. Zhang were supported in part by National Science Foundation grant DMS-1217122. J. Singler and Y. Zhang thank the Institute for Mathematics and its Applications at the University of Minnesota for funding research visits, during which some of this work was completed. The authors thank the referees for their comments, which helped to improve the manuscript.

Appendix

A.1. POD in a Hilbert space and the matrix SVD with a weighted norm

Let X be a Hilbert space with inner product $(\cdot,\cdot)$, and suppose $\{u_j\}_{j=1}^s \subset X$. Define the POD operator $K : \mathbb{R}^s \to X$ by

$$Kf = \sum_{j=1}^{s} u_j f_j, \quad f = [f_1, \ldots, f_s]^T. \tag{11}$$

The Hilbert adjoint operator $K^* : X \to \mathbb{R}^s$ satisfies $(Kf, x) = (f, K^*x)_{\mathbb{R}^s}$ for all $f \in \mathbb{R}^s$ and $x \in X$. It can be checked that

$$K^*x = \big[(x, u_1), (x, u_2), \ldots, (x, u_s)\big]^T. \tag{12}$$

Since K has rank at most s, K is compact and has a singular value decomposition. Let $\{\sigma_i, f_i, x_i\}$ be the core singular values and singular vectors of K, i.e., the nonzero singular values and corresponding singular vectors of K. Then

$$Kf_i = \sigma_i x_i, \tag{13}$$
$$K^*x_i = \sigma_i f_i. \tag{14}$$

In the proposition below, we consider the case where each u_j is expressed in terms of a finite set of basis functions. We show that the core singular values and singular vectors of K can be computed by finding the core SVD of a coefficient matrix with respect to a weighted inner product.

Proposition A.1. *Suppose $\{\phi_k\}_{k=1}^m \subset X$ are linearly independent, and assume $u_j \in X$ is given by*

$$u_j = \sum_{k=1}^{m} U_{k,j}\phi_k, \quad \text{for } j = 1, \ldots, s. \tag{15}$$

Let the matrices $M \in \mathbb{R}^{m\times m}$ and $U \in \mathbb{R}^{m\times s}$ have entries $M_{j,k} := (\phi_j, \phi_k)$ and $U_{k,l}$, for $j, k = 1, \ldots, m$ and $l = 1, \ldots, s$. Then $\{\sigma_i, f_i, c_i\} \subset \mathbb{R} \times \mathbb{R}^s \times \mathbb{R}^m_M$ are the core singular values and singular vectors of $U : \mathbb{R}^s \to \mathbb{R}^m_M$ if and only if $\{\sigma_i, f_i, x_i\} \subset \mathbb{R} \times \mathbb{R}^s \times X$ are the core singular values and singular vectors of $K : \mathbb{R}^s \to X$, where c_i and x_i are related by

$$x_i = \sum_{k=1}^{m} c_{i,k}\phi_k \quad \text{for all } i.$$

Proof. First, since $\{\phi_k\}_{k=1}^m \subset X$ is a linearly independent set, we know M is symmetric positive definite. Next, assume $Kf_i = \sigma_i x_i$ (13) is satisfied with $\sigma_i > 0$. Substitute in the expansion for u_j (15) and use the definition of K in (11) to obtain

$$\sum_{j=1}^{s}\sum_{k=1}^{m} U_{k,j} f_{i,j}\phi_k = \sigma_i x_i, \tag{16}$$

where $f_{i,j}$ denotes the jth entry of the vector f_i. Therefore,

$$x_i = \sum_{l=1}^{m} c_{i,l}\phi_l, \quad c_{i,l} = \frac{1}{\sigma_i}\sum_{j=1}^{s} f_{i,j} U_{l,j}. \tag{17}$$

Let $c_i \in \mathbb{R}^m_M$ denote the vector with entries $c_{i,k}$. Then we have

$$Uf_i = \sigma_i c_i \quad \text{for all } i. \tag{18}$$

Note the above argument is reversible, i.e., if we assume $Uf_i = \sigma_i c_i$ with $\sigma_i > 0$ as in (18), then we obtain $Kf_i = \sigma_i x_i$, where x_i is defined in (17).

Next, we proceed similarly with $K^*x_i = \sigma_i f_i$ (14) and $\sigma_i > 0$. Using the definition of K^* in (12), the expansion for u_j in (15), and the expansion for x_i in (17) gives

$$\begin{aligned}\sigma_i f_i &= \sum_{l=1}^{m}\sum_{k=1}^{m}\left[\left(c_{i,l}\phi_l, U_{k,1}\phi_k\right), \ldots, \left(c_{i,l}\phi_l, U_{k,s}\phi_k\right)\right]^T\\ &= \sum_{l=1}^{m}\sum_{k=1}^{m}\left[c_{i,l}M_{l,k}U_{k,1}, \ldots, c_{i,l}M_{l,k}U_{k,s}\right]^T\\ &= \left[c_i^T MU_1, \ldots, c_i^T MU_s\right]^T\\ &= \left(c_i^T MU\right)^T,\end{aligned}$$

where U_j denotes the jth column of the matrix U. Since $U^* = U^T M$, we have

$$U^*c_i = \sigma_i f_i \quad \text{for all } i. \tag{19}$$

Again, this argument is reversible, i.e., $U^*c_i = \sigma_i f_i$ in (19) with $\sigma_i > 0$ implies $K^*x_i = \sigma_i f_i$, where x_i is defined in (17).

Therefore, we have

$$Uf_i = \sigma_i c_i, \quad U^*c_i = \sigma_i f_i \quad \text{for all } i$$

if and only if

$$Kf_i = \sigma_i x_i, \quad K^*x_i = \sigma_i f_i \quad \text{for all } i.$$

Next, suppose $\{\sigma_i, f_i, x_i\} \subset \mathbb{R} \times \mathbb{R}^s \times X$ are the core singular values and singular vectors of $K : \mathbb{R}^s \to X$. To show $\{\sigma_i, f_i, c_i\} \subset \mathbb{R} \times \mathbb{R}^s \times \mathbb{R}^m_M$ are the core singular values and singular vectors of $U : \mathbb{R}^s \to \mathbb{R}^m_M$, where $c_i = \sigma_i^{-1}Uf_i$, we only need to show $\{c_i\} \subset \mathbb{R}^m_M$ is orthonormal. We show this as follows. Using $c_j = \sigma_j^{-1}Uf_j$, $U^*c_i = \sigma_i f_i$ (19), and $\{f_i\} \subset \mathbb{R}^s$ is orthonormal gives

$$\begin{aligned}\left(c_i, c_j\right)_M &= \frac{1}{\sigma_j}\left(c_i, Uf_j\right)_M\\ &= \frac{1}{\sigma_j}\left(U^*c_i, f_j\right)_{\mathbb{R}^s}\\ &= \frac{\sigma_i}{\sigma_j}\left(f_i, f_j\right)_{\mathbb{R}^s}\\ &= \frac{\sigma_i}{\sigma_j}\delta_{ij} = \delta_{ij}.\end{aligned}$$

Therefore, $\{c_i\} \subset \mathbb{R}^m_M$ is orthonormal.

Finally, suppose $\{\sigma_i, f_i, c_i\} \subset \mathbb{R} \times \mathbb{R}^s \times \mathbb{R}^m_M$ are the core singular values and singular vectors of $U : \mathbb{R}^s \to \mathbb{R}^m_M$. To show $\{\sigma_i, f_i, x_i\} \subset \mathbb{R} \times \mathbb{R}^s \times X$ are the core singular values and singular vectors of $K : \mathbb{R}^s \to X$, where x_i is defined in (17), we only need to show $\{x_i\} \subset X$ is orthonormal. This follows directly from $\{c_i\} \subset \mathbb{R}^m_M$ being an orthonormal set:

$$\left(x_i, x_j\right) = c_j^T Mc_i = \delta_{ij}.$$

This completes the proof. □

A.2. Incremental SVD after removing the average

Some authors apply POD on the data after removing the average of the data. Such a computation has recently been performed incrementally in [19] by applying an algorithm for the additive modification of an SVD [39]. A similar procedure can be done for time varying data with a weighted norm (as considered in Section 5). We do not give the details of the procedure here; however, we show how Brand's algorithm for the additive modification of the SVD [39] can be extended to the case of a weighted norm.

Theorem A.2. *Let $M \in \mathbb{R}^{m\times m}$ be symmetric positive definite, and let $a \in \mathbb{R}^m_M$ and $b \in \mathbb{R}^n$. Suppose $U : \mathbb{R}^n \longrightarrow \mathbb{R}^m_M$ has core SVD given by $U = V\Sigma W^T$, where $V^TMV = I$ for $V \in \mathbb{R}^{m\times k}$, $W^TW = I$ for $W \in \mathbb{R}^{n\times k}$, and $\Sigma \in \mathbb{R}^{k\times k}$. Define*

$$m = V^*a, \quad p = a - Vm, \quad p_a = \|p\|_M, \tag{20}$$

$$n = W^Tb, \quad d = b - Wn, \quad d_b = \|d\|_{\mathbb{R}^n}, \tag{21}$$

where $V^* = V^TM$ and

$$K = \begin{bmatrix}\Sigma + mn^T & d_b m\\ p_a n^T & p_a d_b\end{bmatrix}.$$

If $p_a, d_b > 0$ and the standard core SVD of $K \in \mathbb{R}^{k+1\times k+1}$ is given by

$$K = V_K \Sigma_K W_K^T, \tag{22}$$

then the core SVD of $U + ab^T$ is given by

$$U + ab^T = V_u \Sigma_K W_u^T,$$

where

$$V_u = [\, V\, r\,]\, V_K, \quad r = p_a^{-1} p, \quad W_u = [\, W\, q\,]\, W_K, \quad q = d_b^{-1} d.$$

Proof. Rewrite $U + ab^T$ as

$$U + ab^T = V\Sigma W^T + ab^T = [\, V\, a\,] \begin{bmatrix} \Sigma & 0 \\ 0 & 1 \end{bmatrix} [\, W\, b\,]^T. \tag{23}$$

Next, use the definitions in (20) and (21), respectively, to obtain

$$[\, V\, a\,] = [\, V\, r\,] \begin{bmatrix} I & V^* a \\ 0 & p_a \end{bmatrix},$$
$$[\, W\, b\,] = [\, W\, q\,] \begin{bmatrix} I & W^T b \\ 0 & d_b \end{bmatrix}.$$

Substituting these results into (23) gives

$$\begin{aligned} U + ab^T &= [\, V\, r\,] \left(\begin{bmatrix} I & m \\ 0 & p_a \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & n \\ 0 & d_b \end{bmatrix}^T \right) [\, W\, q\,]^T \\ &= [\, V\, r\,] \begin{bmatrix} \Sigma + mn^T & d_b m \\ p_a n^T & p_a d_b \end{bmatrix} [\, W\, q\,]^T. \end{aligned}$$

Next, note

$$[\, V\, r\,]^T M [\, V\, r\,] = \begin{bmatrix} V^T M V & V^T M r \\ (V^T M r)^T & r^T M r \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

since $V^T M V = V^* V = I$ by assumption,

$$V^T M r = V^* r = V^*(a - Vm)/p_a = (m - m)/p_a = 0,$$

and

$$r^T M r = \frac{\|p\|_M^2}{p_a^2} = \frac{\|p\|_M^2}{\|p\|_M^2} = 1.$$

Also, we have

$$[\, W\, q\,]^T [\, W\, q\,] = \begin{bmatrix} W^T W & W^T q \\ (W^T q)^T & q^T q \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

since $W^T W = I$,

$$W^T q = W^T (b - Wn)/d_b = (n - n)/d_b = 0,$$

and

$$q^T q = \frac{\|d\|_M^2}{d_b^2} = \frac{\|d\|_M^2}{\|d\|_M^2} = 1.$$

Proposition 2.3 gives the result. □

References

[1] D. Amsallem, C. Farhat, Interpolation method for adapting reduced-order models and application to aeroelasticity, AIAA J. 46 (7) (2008) 1803–1813. http://dx.doi.org/10.2514/1.35374.

[2] P. Holmes, J.L. Lumley, G. Berkooz, C.W. Rowley, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, second ed., Cambridge University Press, Cambridge, 2012, p. xvi+386. http://dx.doi.org/10.1017/CBO9780511919701.

[3] H.T. Banks, R.C.H. del Rosario, R.C. Smith, Reduced-order model feedback control design: numerical implementation in a thin shell model, IEEE Trans. Automat. Control 45 (7) (2000) 1312–1324. http://dx.doi.org/10.1109/9.867024.

[4] P. Benner, E. Sachs, S. Volkwein, Model order reduction for PDE constrained optimization, in: Trends in PDE Constrained Optimization, in: Internat. Ser. Numer. Math., vol. 165, 2014, pp. 303–326. http://dx.doi.org/10.1007/978-3-319-05083-6_19.
[5] M. Gubisch, S. Volkwein, POD for linear-quadratic optimal control, in: P. Benner, A. Cohen, M. Ohlberger, K. Willcox (Eds.), Model Reduction and Approximation: Theory and Algorithms, SIAM, Philadelphia, PA, 2017.
[6] M. Gunzburger, N. Jiang, M. Schneier, An ensemble-proper orthogonal decomposition method for the nonstationary Navier-Stokes equations, SIAM J. Numer. Anal. 55 (1) (2017) 286–304. http://dx.doi.org/10.1137/16M1056444.
[7] R. Ştefănescu, A. Sandu, I.M. Navon, POD/DEIM reduced-order strategies for efficient four dimensional variational data assimilation, J. Comput. Phys. 295 (2015) 569–595. http://dx.doi.org/10.1016/j.jcp.2015.04.030.
[8] S. Qian, X. Lv, Y. Cao, F. Shao, Parameter estimation for a 2D tidal model with POD 4D VAR data assimilation, Math. Probl. Eng. 2016 (2016). http://dx.doi.org/10.1155/2016/6751537. Article ID 6751537.
[9] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, Quart. Appl. Math. 45 (3) (1987) 561–571. http://dx.doi.org/10.1090/qam/910462.
[10] R. Pinnau, Model reduction via proper orthogonal decomposition, in: Model Order Reduction: Theory, Research Aspects and Applications, in: Math. Ind., vol. 13, Springer, Berlin, 2008, pp. 95–109. http://dx.doi.org/10.1007/978-3-540-78841-6_5.
[11] J.R. Singler, Convergent snapshot algorithms for infinite-dimensional Lyapunov equations, IMA J. Numer. Anal. 31 (4) (2011) 1468–1496. http://dx.doi.org/10.1093/imanum/drq028.
[12] S. Volkwein, Proper orthogonal decomposition: Theory and reduced-order modelling (lecture notes), 2013. URL http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf.
[13] M. Fahl, Computation of POD basis functions for fluid flows with Lanczos methods, Math. Comput. Modelling 34 (1–2) (2001) 91–107. http://dx.doi.org/10.1016/S0895-7177(01)00051-6.
[14] C.A. Beattie, J. Borggaard, S. Gugercin, T. Iliescu, A domain decomposition approach to POD, in: Proceedings of the IEEE Conference on Decision and Control, 2006, pp. 6750–6756. http://dx.doi.org/10.1109/CDC.2006.377642, ISSN: 0191-2216.
[15] Z. Wang, B. McBee, T. Iliescu, Approximate partitioned method of snapshots for POD, J. Comput. Appl. Math. 307 (2016) 374–384. http://dx.doi.org/10.1016/j.cam.2015.11.023.
[16] C. Himpe, T. Leibner, S. Rave, Hierarchical Approximate Proper Orthogonal Decomposition, 2016. arXiv:1607.05210.
[17] C.G. Baker, K.A. Gallivan, P. Van Dooren, Low-rank incremental methods for computing dominant singular subspaces, Linear Algebra Appl. 436 (8) (2012) 2866–2888. http://dx.doi.org/10.1016/j.laa.2011.07.018.
[18] B. Peherstorfer, K. Willcox, Dynamic data-driven reduced-order models, Comput. Methods Appl. Mech. Engrg. 291 (2015) 21–41. http://dx.doi.org/10.1016/j.cma.2015.03.018.
[19] M.J. Zahr, C. Farhat, Progressive construction of a parametric reduced-order model for PDE-constrained optimization, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 1111–1135. http://dx.doi.org/10.1002/nme.4770.
[20] G.M. Oxberry, T. Kostova-Vassilevska, W. Arrighi, K. Chand, Limited-memory adaptive snapshot selection for proper orthogonal decomposition, Internat. J. Numer. Methods Engrg. 109 (2) (2017) 198–217. http://dx.doi.org/10.1002/nme.5283.
[21] M. Brand, Incremental Singular Value Decomposition of Uncertain Data with Missing Values, in: A. Heyden, G. Sparr, M. Nielsen, P. Johansen (Eds.), Computer Vision — ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 707–720. http://dx.doi.org/10.1007/3-540-47969-4_47.
[22] I. Gohberg, S. Goldberg, M.A. Kaashoek, Classes of Linear Operators. Vol. I, in: Operator Theory: Advances and Applications, vol. 49, Birkhäuser Verlag, Basel, 1990, p. xiv+468. http://dx.doi.org/10.1007/978-3-0348-7509-7.
[23] P.D. Lax, Functional Analysis, in: Pure and Applied Mathematics (New York), Wiley-Interscience [John Wiley & Sons], New York, 2002, p. xx+580.
[24] M. Reed, B. Simon, Methods of Modern Mathematical Physics I: Functional Analysis, second ed., Academic Press, Inc., New York, 1980, p. xv+400.
[25] R.A. Horn, C.R. Johnson, Matrix Analysis, second ed., Cambridge University Press, Cambridge, 2013, p. xviii+643.
[26] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., in: Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013, p. xiv+756.
[27] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000, p. xxx+410. http://dx.doi.org/10.1137/1.9780898719581.
[28] T.A. Davis, Direct Methods for Sparse Linear Systems, in: Fundamentals of Algorithms, vol. 2, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006, p. xii+217. http://dx.doi.org/10.1137/1.9780898718881.
[29] T.A. Davis, S. Rajamanickam, W.M. Sid-Lakhdar, A survey of direct methods for sparse linear systems, Acta Numer. 25 (2016) 383–566. http://dx.doi.org/10.1017/S0962492916000076.
[30] W. Gander, M.J. Gander, F. Kwok, Scientific Computing, in: Texts in Computational Science and Engineering, vol. 11, Springer, Cham, 2014, p. xviii+905. http://dx.doi.org/10.1007/978-3-319-04325-8.
[31] K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for parabolic problems, Numer. Math. 90 (1) (2001) 117–148. http://dx.doi.org/10.1007/s002110100282.
[32] K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics, SIAM J. Numer. Anal. 40 (2) (2002) 492–515. http://dx.doi.org/10.1137/S0036142900382612.
[33] M. Schäfer, S. Turek, F. Durst, E. Krause, R. Rannacher, Benchmark computations of laminar flow around a cylinder, in: E.H. Hirschel (Ed.), Flow Simulation with High-Performance Computers II: DFG Priority Research Programme Results 1993–1995, Vieweg+Teubner Verlag, Wiesbaden, 1996, pp. 547–566. http://dx.doi.org/10.1007/978-3-322-89849-4_39.
[34] I. Akhtar, J. Borggaard, J.A. Burns, H. Imtiaz, L. Zietsman, Using functional gains for effective sensor location in flow control: A reduced-order modelling approach, J. Fluid Mech. 781 (2015) 622–656. http://dx.doi.org/10.1017/jfm.2015.509.
[35] B.R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, F. Thiele, A hierarchy of low-dimensional models for the transient and post-transient cylinder wake, J. Fluid Mech. 497 (2003) 335–363. http://dx.doi.org/10.1017/S0022112003006694.
[36] J.R. Shewchuk, Triangle: Engineering a 2D quality mesh generator and delaunay triangulator, in: M.C. Lin, D. Manocha (Eds.), Applied Computational Geometry: Towards Geometric Engineering, in: Lecture Notes in Computer Science, vol. 1148, Springer-Verlag, 1996, pp. 203–222.
[37] J.R. Shewchuk, Triangle: A two-dimensional quality mesh generator and Delaunay triangulator, version 1.6, 2005. URL https://www.cs.cmu.edu/~quake/triangle.html.
[38] W. Arrighi, G. Oxberry, T. Vassilevska, K. Chand, libROM, https://github.com/LLNL/libROM.
[39] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear Algebra Appl. 415 (1) (2006) 20–30. http://dx.doi.org/10.1016/j.laa.2005.07.021.