An enhanced algorithm for online Proper Orthogonal Decomposition and its parallelization for unsteady simulations

Li, Xiaodong; Hulshoff, Steven; Hickel, Stefan

DOI
10.1016/j.camwa.2022.09.007

Publication date
2022

Document Version
Final published version

Published in
Computers and Mathematics with Applications

Computers and Mathematics with Applications 126 (2022) 43–59

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

www.elsevier.com/locate/camwa

An enhanced algorithm for online Proper Orthogonal Decomposition and its parallelization for unsteady simulations

Xiaodong Li *, Steven Hulshoff, Stefan Hickel

Aerodynamics group, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 2, 2629 HS Delft, the Netherlands

ARTICLE INFO

Keywords:
Incremental singular value decomposition
Order reduction
Proper Orthogonal Decomposition
Low-rank representation
Parallelization
Unsteady simulations

ABSTRACT

Proper Orthogonal Decomposition (POD) plays an important role in the analysis of complex nonlinear systems governed by partial differential equations (PDEs), since it can describe the full-order system in a simplified but representative way using a handful of dominant dynamic modes. However, determining a POD from the results of complex unsteady simulations is often impractical using traditional approaches due to the need to store a large number of high-dimensional solutions. As an alternative, the incremental Singular Value Decomposition (SVD) has been developed, which can be used to avoid the storage problem by performing the POD analysis on the fly using a single-pass updating algorithm. Nevertheless, the total computing cost of incremental SVD is more than traditional approaches. In order to reduce this total cost, we incorporate POD mode truncation into the incremental procedure, leading to an enhanced algorithm for incremental SVD. Two error estimators are formulated for this enhanced incremental SVD based on an aggregated expression of the snapshot solutions, equipping the proposed algorithm with criteria for choosing the truncation number. The effectiveness of these estimators and the parallel efficiency of the enhanced algorithm are demonstrated using transient solutions from representative model problems. Numerical results show that the enhanced algorithm can significantly improve the computing efficiency for different kinds of datasets, and that the proposed algorithm is scalable in both the strong and weak sense.

1. Introduction

Modal analysis is an important tool for understanding unsteady nonlinear phenomena in complex physical systems [1]. One of the most popular techniques is the Proper Orthogonal Decomposition (POD) [2,3], also known as the Karhunen-Loeve procedure or Principal Component Analysis (PCA). This provides a set of modes ordered in terms of their contributions to the total energy of the system. By truncating this set, one can build reduced-order models (ROM) [4]. These are useful in many applications, such as fluid-structure interaction [5,6], optimization problems [7,8], uncertainty quantification [9,10], and optimal control [11–13].

Common approaches to computing the POD of a data set $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ have been discussed in several comprehensive reviews, e.g. [1,14]. These consist of eigenvalue decompositions, the method of snapshots, and singular value decomposition (SVD). POD was introduced to fluid dynamics by Lumey [15] who used the eigenvalue decomposition of the covariance matrix (i.e. $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}$). For cases when the number of degrees of freedom (DoF) (viz. the number of rows) is larger than the number of snapshots (the number of columns), the method of snapshots [16] was proposed by considering a more easily-found eigenvalue decomposition of $\boldsymbol{C} = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$, from which the POD modes can be derived. SVD [17] can also be applied to $\boldsymbol{X}$ directly to find the POD due to its relation to the eigenvalue decomposition. SVD has been shown to be robust in the presence of round-off errors [1]. All of these work on the complete snapshot matrix and are thus referred to as offline approaches. They can be applied to data sets with different structures, as shown in Fig. 1. However, datasets from high-fidelity simulations, such as large-eddy simulations, are high-dimensional, both in terms of DoF and snapshots. These are often prohibitively expensive to analyze using offline approaches.

In order to address this issue, researchers have proposed the recursive POD [18,19], randomized SVD [20,21], and the incremental SVD [22,23]. Here we focus on the latter. The incremental SVD (iSVD) can be used to perform POD online during high-dimensional large-scale simulations without storing the snapshot matrix. The iSVD is initialized using a small data set with a known SVD, which is then updated during the simulation as new

* Corresponding author.
E-mail addresses: X.Li-12@tudelft.nl (X. Li), S.J.Hulshoff@tudelft.nl (S. Hulshoff), S.Hickel@tudelft.nl (S. Hickel).

https://doi.org/10.1016/j.camwa.2022.09.007
Received 11 May 2022; Accepted 4 September 2022

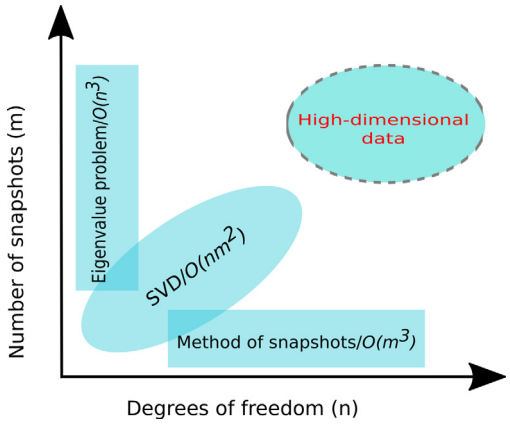


Fig. 1. Common approaches for solving POD with applications on different structures of data sets and their computing complexities, and the necessity of developing cost-effective algorithms for high-dimensional data.

snapshots become available. The incremental SVD was proposed by Brand [22] for application to incomplete datasets in computer vision and audio feature extraction. Fareed et al. [24] further extended the iSVD algorithm with a weighted inner product, and gave an error analysis [25].

Although the iSVD can be used to build a POD online for high-dimensional data, the cumulative cost over all snapshots, $\mathcal{O}(nm^3)$, can be higher than the cost of offline approaches [23]. This limitation can be overcome by either reducing the size of the snapshot matrix or improving the algorithm of iSVD. For the former, the adaptive selection of snapshots [26,27] has been proposed. For the latter, one possibility is to perform a low-rank incremental SVD using a prescribed mode truncation number. This has been applied to efficiently solve unsteady adjoint equations [28,29] in nonlinear problems, although the selection of the truncation number has been *ad hoc* or heuristic. By considering both aspects, Phalippou et al. [30] proposed an on-the-fly snapshots selection for the incremental SVD algorithm using an error estimator, which incorporates information lost due to the projection of skipped snapshots and the truncation of small singular values. Although the reduction of snapshot matrices can be effectively utilized to reduce the computational cost, a reasonable user-specified threshold must be given as a selection criterion.

Considering that one desires a low number of POD modes used to build a low-order representation, we propose an alternative method for the truncation of POD modes based on their contributions to the total energy. This method truncates the POD online as the incremental SVD proceeds, which we refer to as an enhanced iSVD. To this end, we develop an aggregated expression for the snapshot solutions and construct two estimators for evaluating the energy captured by the low-order approximation. This enables us to determine the number of POD modes based on cumulative energy. We also investigate the parallel performance of the proposed enhanced incremental SVD for large-scale datasets.

The remainder of this paper is organized as follows. In Section 2, we present the methodologies for building PODs offline and online, including an overview of three offline POD approaches and the standard incremental SVD for online PODs. We then describe the enhanced incremental SVD and its parallel implementation adapted from [31]. The enhanced algorithm is further discussed in Section 3, with the development of an aggregated expression for the snapshot matrix and the establishment of two a posteriori estimators. Serial numerical experiments are used to demonstrate the impact of the online truncation in this section. The parallel performance of the enhanced incremental SVD is then investigated for large-scale solutions generated from a synthetic data and an unsteady one-dimensional Burgers problem in Section 4. Finally, we conclude the paper with Section 5.

2. Methodologies

We begin by presenting the offline and online approaches to POD in Section 2.1 and Section 2.2, respectively. An enhanced online algorithm to improve the efficiency of the online POD is then proposed in Section 2.3, followed by the design of parallelization for this algorithm in Section 2.4.

2.1. Offline POD

The goal of POD analysis is to identify an optimal set of basis functions for representing a given dataset. Specifically, for a dataset defined by snapshots $\boldsymbol{u}^{(i)}(x) \in \mathcal{V}(\Omega), i = 1, 2, \cdots, N_\mathrm{t}, x \in \Omega$, we seek a set of functions $\boldsymbol{\varphi}(x) \in \mathcal{V}$ that represent the dataset such that the mean square projection on all snapshots is maximized, i.e.

$$\max_{\boldsymbol{\varphi}\in\mathcal{V}} \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \frac{|\langle \boldsymbol{u}^{(i)}, \boldsymbol{\varphi}\rangle|^2}{\langle \boldsymbol{\varphi}, \boldsymbol{\varphi}\rangle}. \tag{1}$$

Here $\langle\cdot,\cdot\rangle$ denotes an inner product defined on Ω, and $\langle \boldsymbol{u}^{(i)}, \boldsymbol{\varphi}\rangle$ describes a projection of the snapshot $\boldsymbol{u}^{(i)}$ onto a function $\boldsymbol{\varphi}$. This maximization problem is equivalent to the following eigenvalue problem [2,11,15]

$$\boldsymbol{K}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}, \tag{2}$$

where $\boldsymbol{\phi}$ denotes the optimal function and

$$\boldsymbol{K}\boldsymbol{\phi} = \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \boldsymbol{u}^{(i)} \langle \boldsymbol{u}^{(i)}, \boldsymbol{\phi}\rangle = \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \int_\Omega \boldsymbol{u}^{(i)}(x)\, \boldsymbol{u}^{(i)}(x')\, \boldsymbol{\phi}(x')\, \mathrm{d}x'. \tag{3}$$

It is noted that $\boldsymbol{K}$ is expressed using continuous functions over Ω. For discrete solutions $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, Equation (2) can be interpreted as finding the eigenvalue λ_j and eigenvectors ϕ_j defined by

$$\boldsymbol{R}\,\boldsymbol{\phi}_j = \lambda_j \boldsymbol{\phi}_j\,, \tag{4}$$

where $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{n\times n}$[1] is the covariance matrix based on an inner product defined by $\langle \boldsymbol{a}, \boldsymbol{b}\rangle = \boldsymbol{a}^\top \boldsymbol{b}$. There are three different approaches to computing the POD offline. First, by solving the eigenvalue problem (4), we can obtain the POD modes $\boldsymbol{\phi}^{(j)}$, where $j = 1, 2, \cdots, \min(n, m)$. In practical applications, the number of DoF (n) associated with the computational mesh is often larger than the number of snapshots (m), leading to a huge covariance matrix that is difficult to analyze directly. The method of snapshots [16] was proposed to overcome this difficulty by instead finding the eigenvalue decomposition of $\boldsymbol{C} = \boldsymbol{X}^\top \boldsymbol{X}$, leading to eigenvalues λ_j and eigenvectors $\boldsymbol{\psi}^{(j)}$, where $j = 1, 2, \cdots, m$. The POD modes are then computed as follows

$$\boldsymbol{\phi}^{(j)} = \boldsymbol{X}\boldsymbol{\psi}^{(j)} / \sqrt{\lambda_j}\,. \tag{5}$$

The eigenvalue decomposition of $\boldsymbol{X}\boldsymbol{X}^\top$ can also be related to the singular value decomposition (SVD) of $\boldsymbol{X}$. In the following, SVD will be performed using the LAPACK library [1], which gives $\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^\top$, where $\boldsymbol{\Sigma}, \boldsymbol{V}$ and $\boldsymbol{W}$ are the singular value matrix, the left singular vector matrix and the right singular vector matrix, respectively. Each column of $\boldsymbol{V}$ denotes one POD mode $\phi^{(j)}$. The associated coefficients $\alpha_j(t_i), i = 1, 2, \cdots, m$ are determined by $\boldsymbol{\Sigma}$ and $\boldsymbol{W}$. This SVD-based offline POD is utilized as a reference for the online PODs introduced in the next sections.

Using the POD, the reduced-order solution $\tilde{u}(x, t_i)$ is calculated by

$$\tilde{u}(x, t_i) = \bar{u}(x) + \sum_{j=1}^{M} \alpha_j(t_i)\,\phi^{(j)}(x)\,, \tag{6}$$

where $\phi^{(j)}, j = 1, 2, \cdots, M$, denotes the selected POD modes, and M is usually smaller than $\min(n, m)$. $\bar{u}$ represents the mean value.

2.2. Online POD using the standard incremental SVD

For high-dimensional datasets, the storage requirements for computing a SVD offline can be prohibitive. Therefore, the incremental SVD algorithm [22] was introduced to achieve POD analysis on the fly, circumventing the storage issue [24].

The standard incremental SVD starts with an initialization defined by

$$\boldsymbol{\Sigma} = \|\boldsymbol{c}_0\|_2\,, \boldsymbol{V} = \boldsymbol{c}/\boldsymbol{\Sigma}_{(1,1)}\,, \boldsymbol{W} = [1]\,, \tag{7}$$

where $\boldsymbol{c}_0$ denotes the first snapshot solution. Thus the snapshot matrix is $\boldsymbol{U} = [\boldsymbol{c}_0] = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^\top$. The SVD is completed by carrying out the standard incremental algorithm (SIA), shown in Algorithm 1, for all remaining snapshots to build the POD online. More precisely, after the projection of $\boldsymbol{c}$ onto $\boldsymbol{V}$, the matrix $\boldsymbol{Q}$ is computed, where small projections less than a threshold of **tol** [24] are neglected to prevent the impact of round-off errors. A standard SVD then is applied to $\boldsymbol{Q}$ before the updating process. It is then decided if the added column will increase the rank of the updated matrix (lines 8-13). Here the subscript denotes the index of rows and columns starting from 1 ($V_{Q_{(1:k,1:k)}}$ is a sub-matrix of V_Q with the first k-th rows and columns, for instance). The truncation of small singular values less than a prescribed threshold $\mathbf{tol}_{\mathrm{sv}}$ is used to improve efficiency. Finally, the updated modes are re-orthogonalized by the modified Gram-Schmidt process if non-orthogonality occurs among them, which improves the robustness of the algorithm.

That updated SVD used by the SIA can be proved as follows. Adding a new snapshot $\boldsymbol{c} \in \mathbb{R}^{n\times 1}$ leads to an updated matrix $\boldsymbol{U}_u = [\boldsymbol{U}\ \boldsymbol{c}]$, where $\boldsymbol{U} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^\top$. The SVD of $\boldsymbol{U}_u$ is given as

$$\boldsymbol{U}_u = \boldsymbol{V}_u\,\boldsymbol{\Sigma}_Q\,\boldsymbol{W}_u^\top\,, \tag{8}$$

where $\boldsymbol{\Sigma}_Q$, $\boldsymbol{V}_u$ and $\boldsymbol{W}_u$ are the new singular value matrix, left and right singular vector matrix, respectively, defined by

$$\boldsymbol{V}_u = [\boldsymbol{V}\ \boldsymbol{j}]\,\boldsymbol{V}_Q\,, \boldsymbol{W}_u = \begin{bmatrix} \boldsymbol{W} & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{W}_Q\,. \tag{9}$$

This follows from the fact that $\boldsymbol{V}_u$ and $\boldsymbol{W}_u$ are orthogonal matrices, which is proved as follows

$$\begin{aligned} \boldsymbol{V}^\top \boldsymbol{j} &= \boldsymbol{V}^\top \frac{\boldsymbol{c} - \boldsymbol{V}\boldsymbol{d}}{p} = \frac{1}{p}(\boldsymbol{V}^\top \boldsymbol{c} - \boldsymbol{V}^\top \boldsymbol{V}\boldsymbol{d}) = \frac{1}{p}(\boldsymbol{d} - \boldsymbol{d}) = \vec{0} \\ \boldsymbol{V}_u{}^\top \boldsymbol{V}_u &= \boldsymbol{V}_Q^\top \begin{bmatrix} \boldsymbol{V}^\top \\ \boldsymbol{j}^\top \end{bmatrix} [\boldsymbol{V}\ \boldsymbol{j}]\,\boldsymbol{V}_Q = \boldsymbol{V}_Q^\top \begin{bmatrix} \boldsymbol{V}^\top\boldsymbol{V} & \boldsymbol{V}^\top\boldsymbol{j} \\ \boldsymbol{j}^\top\boldsymbol{V} & \boldsymbol{j}^\top\boldsymbol{j} \end{bmatrix} \boldsymbol{V}_Q = \boldsymbol{V}_Q^\top \boldsymbol{V}_Q = \boldsymbol{I} \\ \boldsymbol{W}_u{}^\top \boldsymbol{W}_u &= \boldsymbol{W}_Q^\top \begin{bmatrix} \boldsymbol{W}^\top & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{W} & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{W}_Q = \boldsymbol{W}_Q^\top \begin{bmatrix} \boldsymbol{W}^\top\boldsymbol{W} & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{W}_Q = \boldsymbol{W}_Q^\top \boldsymbol{W}_Q = \boldsymbol{I}\,. \end{aligned} \tag{10}$$

2.3. Enhanced incremental SVD for online POD

Although the standard incremental SVD can achieve POD analysis online, the total computational cost is usually higher than that of the offline method. In practice, however, the number of POD modes (M) necessary for building an accurate reduced-order model is far smaller than the number of DoF (n) and snapshots (m). Thus to improve computing efficiency, we introduce the truncation of POD modes into the incremental SVD algorithm as shown in Algorithm 2 between lines 14 and line 16. This leads to an enhanced incremental algorithm (EIA), which can remarkably reduce the computing cost of the incremental SVD. This improvement does come with a sacrifice in the accuracy of POD analysis due to the influence of neglected high-order modes. However, this sacrifice is often within a reasonable range and can be controlled by adding more POD modes during the incremental process, as demonstrated in Section 3.3.

[1] Strictly speaking, $\boldsymbol{R}$ should be expressed as $\boldsymbol{X}\boldsymbol{X}^\top/N_{\mathrm{t}}$, but $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^\top$ has been widely used in literature for convenience since its eigenvectors are the same. The influence of N_{t} is thus lumped into the magnitude of eigenvalues. (See also reference [1].)

Algorithm 1 Standard incremental SVD for building POD.

Input: $V \in \mathbb{R}^{n\times k}, \Sigma \in \mathbb{R}^{k\times k}, W \in \mathbb{R}^{k\times k}, c \in \mathbb{R}^{n\times 1}, \mathbf{tol}, \mathbf{tol}_{\text{sv}}$
Output: V, Σ, W

$k = \text{nColumns}(V)$ ▷ The number of columns
$d = V^\top c, \quad p = (|(c - Vd)^\top (c - Vd)|)^{1/2}$ ▷ **Part 1: Projection**
if $p < \mathbf{tol}$ **then**
  $Q = \begin{bmatrix} \Sigma & d \\ 0 & 0 \end{bmatrix}$
else
  $Q = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}$
▷ **Part 2: SVD solution**
$V_Q, \Sigma_Q, W_Q = \text{SVD}\ (Q)$
▷ **Part 3: LSV update**
if $(p < \mathbf{tol})$ OR $(k \geq n)$ **then**
  $V = V V_{Q_{(1:k,1:k)}}, \quad \Sigma = \Sigma_{Q_{(1:k,1:k)}}, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_{Q_{(1:k+1,1:k)}}$
else
  $j = (c - Vd)/p$
  $V = [V\ j] V_Q, \quad \Sigma = \Sigma_Q, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q$
  $k = k + 1$
▷ **Part 5: Small SV truncation**
if $(\Sigma_{(k-1,k-1)} > \mathbf{tol}_{\text{sv}})$ AND $(\Sigma_{(k,k)} < \mathbf{tol}_{\text{sv}})$ **then**
  $k = k - 1$
  $\Sigma = \Sigma_{(1:k,1:k)}, \quad V = V_{(:,1,k)}, \quad W = W_{(:,1:k)}$
▷ **Part 6: Reorthogonalization**
if $|V_{(:,k)}^\top V_{(:,1)}| > \min(\mathbf{tol}, \varepsilon \times n)$ **then** ▷ ε is a double-precision machine epsilon
  $V = \text{ModifiedGramSchmidt}(V)$

Algorithm 2 Enhanced incremental SVD for building POD.

Input: $V \in \mathbb{R}^{n\times k}, \Sigma \in \mathbb{R}^{k\times k}, W \in \mathbb{R}^{k\times k}, c \in \mathbb{R}^{n\times 1}, \mathbf{tol}, \mathbf{tol}_{\text{sv}}$
Output: V, Σ, W

$k = \text{nColumns}(V)$ ▷ The number of columns
$d = V^\top c, \quad p = (|(c - Vd)^\top (c - Vd)|)^{1/2}$ ▷ **Part 1: Projection**
if $p < \mathbf{tol}$ **then**
  $Q = \begin{bmatrix} \Sigma & d \\ 0 & 0 \end{bmatrix}$
else
  $Q = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}$
▷ **Part 2: SVD solution**
$V_Q, \Sigma_Q, W_Q = \text{SVD}\ (Q)$
▷ **Part 3: LSV update**
if $(p < \mathbf{tol})$ OR $(k \geq n)$ **then**
  $V = V V_{Q_{(1:k,1:k)}}, \quad \Sigma = \Sigma_{Q_{(1:k,1:k)}}, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_{Q_{(1:k+1,1:k)}}$
else
  $j = (c - Vd)/p$
  $V = [V\ j] V_Q, \quad \Sigma = \Sigma_Q, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q$
  $k = k + 1$
▷ **Part 4: Enhanced process**
if $(k > M)$ **then**
  $\Sigma = \Sigma_{(1:M,1:M)}, \quad V = V_{(:,1:M)}, \quad W = W_{(:,1:M)}$
  $k = M$
▷ **Part 5: Small SV truncation**
if $(\Sigma_{(k-1,k-1)} > \mathbf{tol}_{\text{sv}})$ AND $(\Sigma_{(k,k)} < \mathbf{tol}_{\text{sv}})$ **then**
  $k = k - 1$
  $\Sigma = \Sigma_{(1:k,1:k)}, \quad V = V_{(:,1,k)}, \quad W = W_{(:,1:k)}$
▷ **Part 6: Reorthogonalization**
if $|V_{(:,k)}^\top V_{(:,1)}| > \min(\mathbf{tol}, \varepsilon \times n)$ **then** ▷ ε is a double-precision machine epsilon
  $V = \text{ModifiedGramSchmidt}(V)$

2.4. Parallel design of the enhanced online algorithm

There are two existing approaches to parallelizing the iSVD. Iwen and Ong [32] proposed a hierarchical approach, which involves computing a local iSVD on each processor and then performing a global agglomerative iSVD over all processors. This avoids data communication during local operations but does not produce global POD results until the terminating time step. An alternative approach is to recognize that the incremental SVD involves multiplications between vectors and matrices, and thus the parallelization of these operations can be used. This approach is adopted by Arrighi et al. [31] in the open-source library *libROM* and can offer a global POD analysis at each incremental step. Our method is developed based on this library, and we modify it to incorporate the enhanced process using the same parallel data structure.

In *libROM*, vectors are distributed among n_p processors, as illustrated in Fig. 2. The data matrix is distributed in a similar way but for each column. This ensures that each part of the dataset will be operated on by only one processor. The different types of computations using matrices

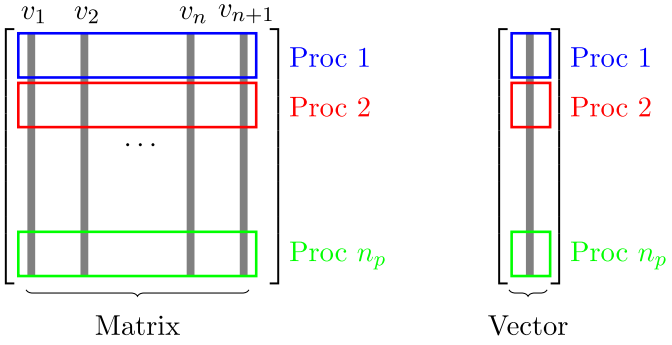


Fig. 2. Parallelization designs of matrices and vectors in incremental SVD.

Table 1
Parallelization operations in incremental SVD.

Different types of data	Local operations	Global operations
Vector^T · Vector	undis[a] · undis → undis	dis[b] · dis → undis
Matrix · Vector	undis · undis → undis	dis · undis → dis
Matrix^T · Vector	undis · undis → undis	dis · dis → undis
Matrix · Matrix	undis · undis → undis	dis · undis → dis
Matrix^T · Matrix	undis · undis → undis	dis · dis → undis

[a] denotes undistributed data.
[b] denotes distributed data.

and vectors are summarized in Table 1, where a local operation is one that employs data stored on the same processor, while a global operation employs data from other processors as well.

3. Performance of the enhanced algorithm in serial

The number of selected POD modes (M) is of key importance as it determines the accuracy of a reduced-order model. There are different methods to determine M, including heuristic and statistical methods [33]. A common method is to quantify the energy captured by the selected POD modes, namely the cumulative energy e_M, computed as

$$e_M = \frac{\sum_{i=1}^{M} \sigma_i^2}{\sum_{i=1}^{\min(m,n)} \sigma_i^2}, \tag{11}$$

where σ_i denotes the singular values, which are ordered in terms of their contributions to the total energy of the entire system. Choosing M so that e_M is larger than 90% [14] of the total energy is a standard way to ensure that the POD analysis rationally represents full-order solutions. In literature, other thresholds of the cumulative energy captured, such as 99% [6,11–13], have also been used. We define the required accuracy of the truncation using the 99% criterion.

In the following, an aggregated expression is formulated for the enhanced incremental SVD, by which the snapshot matrix is divided into the reconstructed and truncated components. Using this expression, we establish two estimators for the lower bound of the cumulative energy captured during the incremental process. The aggregated expression is described in Section 3.1. The lower-bound estimators are given in Section 3.2, followed by the investigation of their fidelity using an unsteady Burgers problem in Section 3.3.

3.1. An aggregated expression for the enhanced algorithm

In this section, we analyze the influence of the enhanced process on the incremental SVD, focusing on the accuracy of reconstructed solutions based on the selected POD modes. For M POD modes, we can express the snapshot matrix at any k-th incremental step as follows

$$\boldsymbol{U}_k = \begin{cases} \widetilde{\boldsymbol{U}}_k & k \leqslant M \\ \widetilde{\boldsymbol{U}}_k + \boldsymbol{U}'_k & k = M+1 \\ \widetilde{\boldsymbol{U}}_k + \boldsymbol{U}'_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)} & k \geqslant M+2 \end{cases}, \tag{12}$$

where $\boldsymbol{U}_k, k = 1, 2, \cdots, n$ is a sub-matrix of the entire snapshot matrix $\boldsymbol{X}$ defined by the first k columns. $\widetilde{\boldsymbol{U}}_k = \widetilde{\boldsymbol{V}}_k \widetilde{\boldsymbol{\Sigma}}_k \widetilde{\boldsymbol{W}}_k^{\mathsf{T}}$ represents the reconstructed solution after the k-th incremental step, defined as

$$\widetilde{\boldsymbol{V}}_k = \begin{cases} \boldsymbol{V}^{(k)} \\ \boldsymbol{V}^{(k)}_{(:,1:M)} \end{cases} \quad \widetilde{\boldsymbol{\Sigma}}_k = \begin{cases} \boldsymbol{\Sigma}^{(k)} \\ \boldsymbol{\Sigma}^{(k)}_{(1:M,1:M)} \end{cases} \quad \widetilde{\boldsymbol{W}}_k = \begin{cases} \boldsymbol{W}^{(k)} & , k \leqslant M \\ \boldsymbol{W}^{(k)}_{(:,1:M)} & , k \geqslant M+1 \end{cases}, \tag{13}$$

where $\boldsymbol{\Sigma}_k$ and $\boldsymbol{V}_k, \boldsymbol{W}_k$ denote the singular value matrix and singular vector matrices. $\boldsymbol{U}'_k = \boldsymbol{V}^{(k)}_{(:,M+1)} \boldsymbol{\Sigma}^{(k)}_{(M+1,M+1)} \boldsymbol{W}^{(k)\,\mathsf{T}}_{(:,M+1)}$ is the truncated solution at the k-th incremental step, and vanishes when $k \leqslant M$.

$\boldsymbol{U}_i'^{(k)} (k > i \geqslant M+1)$ represents solutions truncated at the i-th step but extended to the k-th step, and it is defined by

$$\boldsymbol{U}_i'^{(k)} = [\boldsymbol{U}'_i \quad \underbrace{\vec{\boldsymbol{0}} \quad \cdots \quad \vec{\boldsymbol{0}}}_{k-i}]. \tag{14}$$

This implies

$$\begin{aligned} \boldsymbol{U}_k'^{(k+1)} &= [\boldsymbol{U}_k' \quad \vec{\boldsymbol{0}}], \quad k \geqslant M+1 \\ \boldsymbol{U}_i'^{(k+1)} &= [\boldsymbol{U}_i'^{(k)} \quad \vec{\boldsymbol{0}}], \quad k > i \geqslant M+1 . \end{aligned} \tag{15}$$

It is worth noting that the third case in Equation (12), $k \geqslant M+2$, is the scenario we usually meet in practice. We give a proof of Equation (12) as follows.

Proof. When $k \leqslant M$, the expression is identical to the standard incremental SVD. This is to say that $\boldsymbol{U}_k = \boldsymbol{V}^{(k)}\boldsymbol{\Sigma}^{(k)}\boldsymbol{W}^{(k)\mathsf{T}} = \widetilde{\boldsymbol{U}_k}$.

When $k = M+1$, we have $\boldsymbol{U}_{M+1} = [\boldsymbol{U}_M \; \boldsymbol{c}_{M+1}] = [\widetilde{\boldsymbol{U}_M} \; \boldsymbol{c}_{M+1}]$. The SVD is utilized to obtain

$$\begin{aligned} \boldsymbol{U}_{M+1} &= \boldsymbol{V}^{(M+1)}\boldsymbol{\Sigma}^{(M+1)}\boldsymbol{W}^{(M+1)\mathsf{T}} && \triangleright\text{SVD} \\ &= \boldsymbol{V}^{(M+1)}_{(:,1:M)}\boldsymbol{\Sigma}^{(M+1)}_{(1:M,1:M)}\boldsymbol{W}^{(M+1)\;\mathsf{T}}_{(:,1:M)} + \boldsymbol{V}^{(M+1)}_{(:,M+1)}\boldsymbol{\Sigma}^{(M+1)}_{(M+1,M+1)}\boldsymbol{W}^{(M+1)\;\mathsf{T}}_{(:,M+1)} && \triangleright\text{Additivity} \\ &:= \widetilde{\boldsymbol{V}}_{M+1}\widetilde{\boldsymbol{\Sigma}}_{M+1}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{M+1} + \boldsymbol{v}^{(M+1)}_{M+1}\boldsymbol{\sigma}^{(M+1)}_{M+1}\boldsymbol{w}^{(M+1)\mathsf{T}}_{M+1} && \triangleright\text{Definition} \\ &:= \widetilde{\boldsymbol{U}_{M+1}} + \boldsymbol{U}'_{M+1}, \end{aligned} \tag{16}$$

which gives the second expression. It is noted that $\boldsymbol{U}'_{M+1}$ will be truncated since we only consider M POD modes.
The third expression can be proved using mathematical induction as follows:
Base case: When $k = M+2$, the solution data is expressed as

$$\begin{aligned} \boldsymbol{U}_{M+2} &= [\boldsymbol{U}_{M+1} \quad \boldsymbol{c}_{M+2}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} + \boldsymbol{U}'_{M+1} \quad \boldsymbol{c}_{M+2}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] + [\boldsymbol{U}'_{M+1} \quad \vec{\boldsymbol{0}}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] + \boldsymbol{U}'^{(M+2)}_{M+1}. \end{aligned} \tag{17}$$

We apply the SVD to $[\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}]$, resulting in the following statement

$$\begin{aligned} [\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] &= \boldsymbol{V}^{(M+2)}\boldsymbol{\Sigma}^{(M+2)}\boldsymbol{W}^{(M+2)\mathsf{T}} \\ &= \boldsymbol{V}^{(M+2)}_{(:,1:M)}\boldsymbol{\Sigma}^{(M+2)}_{(1:M,1:M)}\boldsymbol{W}^{(M+2)\;\mathsf{T}}_{(:,1:M)} + \boldsymbol{V}^{(M+2)}_{(:,M+1)}\boldsymbol{\Sigma}^{(M+2)}_{(M+1,M+1)}\boldsymbol{W}^{(M+2)\;\mathsf{T}}_{(:,M+1)} \\ &:= \widetilde{\boldsymbol{V}}_{M+2}\widetilde{\boldsymbol{\Sigma}}_{M+2}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{M+2} + \boldsymbol{v}^{(M+2)}_{M+1}\boldsymbol{\sigma}^{(M+2)}_{M+1}\boldsymbol{w}^{(M+2)\mathsf{T}}_{M+1} \\ &:= \widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}'_{M+2}, \end{aligned} \tag{18}$$

where $\widetilde{\boldsymbol{U}_{M+2}} = \widetilde{\boldsymbol{V}}_{M+2}\widetilde{\boldsymbol{\Sigma}}_{M+2}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{M+2}$ and $\boldsymbol{U}'_{M+2} = \boldsymbol{v}^{(M+2)}_{M+1}\boldsymbol{\sigma}^{(M+2)}_{M+1}\boldsymbol{w}^{(M+2)\mathsf{T}}_{M+1}$. Consequently, the solution is stated as $\boldsymbol{U}_{M+2} = \widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}'_{M+2} + \boldsymbol{U}'^{(M+2)}_{M+1}$. In other words, the expression of $\boldsymbol{U}_k$ is satisfied when $k = M+2$.
Inductive step: Assume that the induction hypothesis holds for a particular $k \geqslant M+2$, meaning

$$\boldsymbol{U}_k = \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}'_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)}. \tag{19}$$

When $n = k+1$, the updated snapshot matrix is expressed as

$$\begin{aligned} \boldsymbol{U}_{k+1} = [\boldsymbol{U}_k \quad c_{k+1}] &= [\widetilde{\boldsymbol{U}_k} + \boldsymbol{U}'_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)} \quad c_{k+1}] \\ &= [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + [\boldsymbol{U}'_k \quad \vec{\boldsymbol{0}}] + [\sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)} \quad \vec{\boldsymbol{0}}] \\ &= [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + \boldsymbol{U}_k'^{(k+1)} + \sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k+1)} \\ &= [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + \sum_{i=M+1}^{k} \boldsymbol{U}_i'^{(k+1)}. \end{aligned} \tag{20}$$

We apply the SVD to $[\widetilde{\boldsymbol{U}_k} \quad c_{k+1}]$ and express it as follows

$$\begin{aligned} [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] &= \boldsymbol{V}^{(k+1)}\boldsymbol{\Sigma}^{(k+1)}\boldsymbol{W}^{(k+1)\mathsf{T}} \\ &= \boldsymbol{V}^{(k+1)}_{(:,1:M)}\boldsymbol{\Sigma}^{(k+1)}_{(1:M,1:M)}\boldsymbol{W}^{(k+1)\;\mathsf{T}}_{(:,1:M)} + \boldsymbol{V}^{(k+1)}_{(:,M+1)}\boldsymbol{\Sigma}^{(k+1)}_{(M+1,M+1)}\boldsymbol{W}^{(k+1)\;\mathsf{T}}_{(:,M+1)} \\ &:= \widetilde{\boldsymbol{V}}_{k+1}\widetilde{\boldsymbol{\Sigma}}_{k+1}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{k+1} + \boldsymbol{v}^{(k+1)}_{M+1}\boldsymbol{\sigma}^{(k+1)}_{M+1}\boldsymbol{w}^{(k+1)\mathsf{T}}_{M+1} \\ &:= \widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}'_{k+1}, \end{aligned} \tag{21}$$

where $\widetilde{\boldsymbol{U}_{k+1}} = \widetilde{\boldsymbol{V}}_{k+1}\widetilde{\boldsymbol{\Sigma}}_{k+1}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{k+1}$ and $\boldsymbol{U}'_{k+1} = \boldsymbol{v}^{(k+1)}_{M+1}\boldsymbol{\sigma}^{(k+1)}_{M+1}\boldsymbol{w}^{(k+1)\mathsf{T}}_{M+1}$. Therefore, we deduce that

$$\boldsymbol{U}_{k+1} = \widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}'_{k+1} + \sum_{i=M+1}^{k} \boldsymbol{U}_i'^{(k+1)}. \tag{22}$$

Since the expression of $\boldsymbol{U}_{k+1}$ also holds true, we establish the inductive step.
Conclusion: As we have proved the base case and the inductive step, the original statement of $\boldsymbol{U}_n$ holds for every natural number $n \geqslant M+2$. □

3.2. Lower-bound estimators of cumulative energy

The enhanced process for the incremental SVD can reduce computational cost. However, the accuracy of SVD can deteriorate if the truncation number is too small. To this end, we propose error estimators that can quantify the accuracy of the enhanced incremental SVD without solving for all eigenvalues. By virtue of the above-mentioned aggregated expression, we establish two lower-bound estimators for the cumulative energy obtained by the enhanced process. Since we can evaluate the summation of singular values by the Frobenius norm (F-norm) of matrices, i.e. $\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{\mathrm{Rank}(A)} \sigma_i^2(\boldsymbol{A})}$, the energy ratio of the M-selected POD modes is computed as

$$e_M = \frac{\sum_{i=1}^{M} \sigma_i^2}{\sum_{i=1}^{\min(m,n)} \sigma_i^2} = \frac{\|\widetilde{\boldsymbol{U}}\|_{\mathrm{F}}^2}{\|\boldsymbol{U}\|_{\mathrm{F}}^2}. \tag{23}$$

Then two lower-bound estimators at the k-th incremental step are given as

$$e_{\mathrm{con}}^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\left(\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} + F_1^{(k)}\right)^2} \tag{24}$$

$$e_{\mathrm{simp}}^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_2^{(k)}}, \tag{25}$$

where $\widehat{\boldsymbol{U}_k} = \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}'_k$, and thus $\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 = \|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2$ due to the orthogonality of POD modes at k-th incremental step. $F_1^{(k)}$ and $F_2^{(k)}$ are given by

$$F_1^{(k)} = \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}} = F_1^{(k-1)} + \|\boldsymbol{U}'_{k-1}\|_{\mathrm{F}},\ k \geqslant M+2 \tag{26}$$

$$F_2^{(k)} = \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 = F_2^{(k-1)} + \|\boldsymbol{U}'_{k-1}\|_{\mathrm{F}}^2,\ k \geqslant M+2, \tag{27}$$

where $F_1^{(M+1)} = 0$ and $F_2^{(M+1)} = 0$. $F_1^{(k)}$ and $F_2^{(k)}$ are only dependent on the solution truncated in previous steps and are utilized to evaluate the error at next step. Using Equations (26) and (27) allows us to avoid storing all $\|\boldsymbol{U}_j'^{(k)}\|_{\mathrm{F}}, i = M+1, \cdots, k-1$, and to use a recursive approach for computing the two estimators.

The expression for e_{con}^k is derived using the Cauchy–Schwarz inequality for the Frobenius inner product, viz. $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_{\mathrm{F}} \leqslant \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{B}\|_{\mathrm{F}}$. We can evaluate $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ as

$$\begin{aligned}
\|\boldsymbol{U}_k\|_{\mathrm{F}}^2 &= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 + 2\sum_{i=M+1}^{k-1} \langle \widehat{\boldsymbol{U}_k}, \boldsymbol{U}_i'^{(k)} \rangle_{\mathrm{F}} \\
&= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \sum_{j=M+1}^{k-1} \langle \boldsymbol{U}_i'^{(k)}, \boldsymbol{U}_j'^{(k)} \rangle_{\mathrm{F}} + 2\sum_{i=M+1}^{k-1} \langle \widehat{\boldsymbol{U}_k}, \boldsymbol{U}_i'^{(k)} \rangle_{\mathrm{F}} \\
&\leqslant \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \sum_{j=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}} \|\boldsymbol{U}_j'^{(k)}\|_{\mathrm{F}} + 2\sum_{i=M+1}^{k-1} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}} \\
&= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_1^{(k)} F_1^{(k)} + 2F_1^{(k)} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} \\
&= \left(\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} + F_1^{(k)}\right)^2.
\end{aligned} \tag{28}$$

Then the first lower-bound estimator of e_M^k is derived as

$$e_M^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\boldsymbol{U}_k\|_{\mathrm{F}}^2} \geqslant \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_1^{(k)} F_1^{(k)} + 2F_1^{(k)} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}} := e_{\mathrm{con}}^k. \tag{29}$$

This estimator can be conservative in practice, as we estimate $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ with an upper bound. If we use this estimator to determine the number of POD modes, it will revert towards the standard incremental SVD when its value exceeds the threshold.

Based on Equation (12), however, we can have an accurate computation for $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ as stated in Proposition 1. This will be used to construct a second estimator, e_{simp}^k.

Proposition 1. *Suppose that we have the expression of $\boldsymbol{U}_k$ as shown in Equation (12), the Frobenius norm of $\boldsymbol{U}_k$ is given as*

$$\|\boldsymbol{U}_k\|_F^2 = \begin{cases} \|\widetilde{\boldsymbol{U}_k}\|_F^2 & k \leqslant M \\ \|\widetilde{\boldsymbol{U}_k}\|_F^2 + \|\boldsymbol{U}'_k\|_F^2 & k = M+1 \\ \|\widetilde{\boldsymbol{U}_k}\|_F^2 + \|\boldsymbol{U}'_k\|_F^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_F^2 & k \geqslant M+2 \end{cases}, \tag{30}$$

Proof. The F-norm for the truncated solutions $\boldsymbol{U}_i'^{(k)}$ is given before the detailed proof. It satisfies the following equation

$$\|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}} = \|\boldsymbol{U}_i'\|_{\mathrm{F}}, k \geqslant M+2, i = M+1, \cdots, k-1. \tag{31}$$

This is an inherent property of $\boldsymbol{U}_i'^{(k)}$, which can be verified using Equation (14) and the definition of the F-norm.

When $k \leqslant M+1$, the expression for $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ is satisfied, since there is no difference from the standard incremental SVD. Then we prove the third case by mathematical induction as follows.

Base case: When $k = M+2$, the solution data is expressed as

$$\begin{aligned}
\|\boldsymbol{U}_{M+2}\|_{\mathrm{F}}^2 &= \|\boldsymbol{U}_{M+1} \quad \boldsymbol{c}_{M+2}\|_{\mathrm{F}}^2 \\
&= \|\boldsymbol{U}_{M+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{M+2}\|_2^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{M+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+1}'\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{M+2}\|_2^2 && \triangleright \text{When } k = M+1 \\
&= \|\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+1}'\|_{\mathrm{F}}^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}_{M+2}'\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+1}'\|_{\mathrm{F}}^2 && \triangleright \textit{Equation (18)} \\
&= \|\widetilde{\boldsymbol{U}_{M+2}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+2}'\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+1}'\|_{\mathrm{F}}^2 && \triangleright \text{Orthogonality} \\
&= \|\widetilde{\boldsymbol{U}_{M+2}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+2}'\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{M+1}'^{(M+2)}\|_{\mathrm{F}}^2 && \triangleright \text{Equation (31)}.
\end{aligned} \tag{32}$$

Therefore, $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ is satisfied when $k = M+1$.

Inductive step: Assume that the induction hypothesis holds for a particular $k \geqslant M+2$, viz. $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ is expressed as

$$\|\boldsymbol{U}_k\|_{\mathrm{F}}^2 = \|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2, \tag{33}$$

when $n = k+1$, we can obtain the F-norm of $\boldsymbol{U}_{k+1}$ as

$$\begin{aligned}
\|\boldsymbol{U}_{k+1}\|_{\mathrm{F}}^2 &= \|\boldsymbol{U}_k \quad \boldsymbol{c}_{k+1}\|_{\mathrm{F}}^2 \\
&= \|\boldsymbol{U}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{k+1}\|_2^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{k+1}\|_2^2 && \triangleright \text{Equation (33)} \\
&= \|\widetilde{\boldsymbol{U}_k} \quad \boldsymbol{c}_{k+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}_{k+1}'\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 && \triangleright \text{Equation (21)} \\
&= \|\widetilde{\boldsymbol{U}_{k+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{k+1}'\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2 && \triangleright \text{Orthogonality} \\
&= \|\widetilde{\boldsymbol{U}_{k+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_{k+1}'\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k} \|\boldsymbol{U}_i'^{(k+1)}\|_{\mathrm{F}}^2, && \triangleright \text{Equation (31)}
\end{aligned} \tag{34}$$

which proves that the statement for $\|\boldsymbol{U}_{k+1}\|_{\mathrm{F}}^2$ is held, thereby establishing the inductive step.

Conclusion: As the base case and the inductive step have been proved, the original statement for $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ holds for every natural number $k \geqslant M+2$. □

The second estimator is thus

$$e_M^k = \frac{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2}{\|\boldsymbol{U}_k\|_{\mathrm{F}}^2} = \frac{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2}{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k'\|_{\mathrm{F}}^2 + \|\sum_{i=M+1}^{k-1} \boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}^2} = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_2^{(k)}} := e_{\mathrm{simp}}^k. \tag{35}$$

During the incremental process, it is worth noting that $\widetilde{\boldsymbol{U}_k}$ is a combination of M-selected modes for the complete solution $\boldsymbol{U}_k$ when $k \geqslant M$. Since the POD is defined by a maximization problem, the POD modes are obtained in a decreasing order of their contributions to the system. Therefore, the approximation of the snapshot matrix using first M POD modes is the most optimal combination [34]. The summation of first k singular values from POD is larger than or equal to those from any other rank-k approximation, thereby leading to $\sum_{i=1}^{M} \sigma_i^2(\widetilde{\boldsymbol{U}_k}) \leqslant \sum_{i=1}^{M} \sigma_i^2(\boldsymbol{U}_k)$. As a result, this estimator is a lower bound for the cumulative energy captured by the standard iSVD. Note that the two estimators for the evaluation of the enhanced incremental SVD have been constructed without accounting for the original errors from the standard incremental SVD, which are very small for the cases considered here.

3.3. Impact of the truncation number on the enhanced algorithm

In this section, we investigate the impact of using different M on the accuracy of the lower-bound estimators and the resulting POD modes obtained from solutions of a one-dimensional (1D) unsteady Burgers problem. The Burgers equation is often used as a mathematical model for applications that involve shock wave propagation in viscous flows or idealized turbulence [35], and it is expressed as

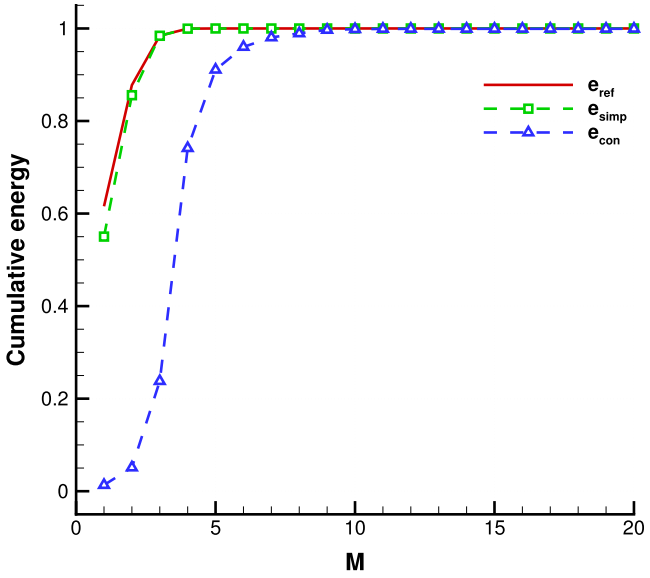
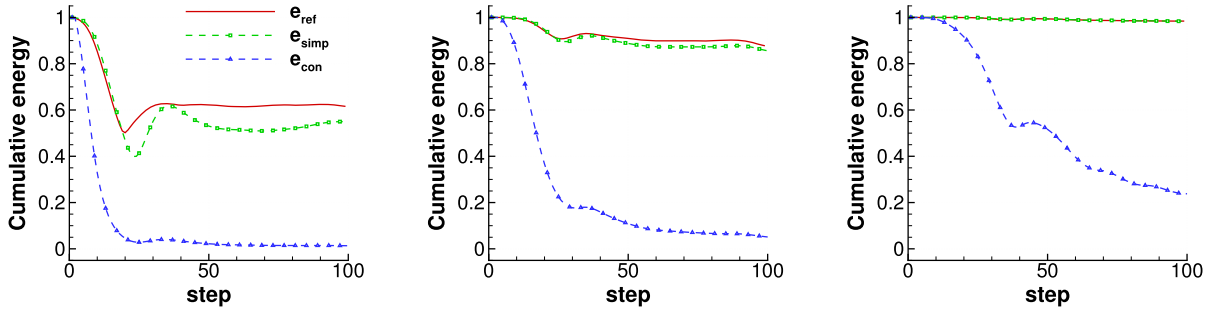


Fig. 3. Cumulative energy of two estimators, e_{con} and e_{simp} at the terminating step (100 steps), compared with the reference value from the standard incremental SVD, with different numbers (M) for the enhanced online process.



(a) $M = 1$ (b) $M = 2$ (c) $M = 3$

Fig. 4. Cumulative energy computed by estimators, e_{con} and e_{simp}, and reference value from the standard incremental SVD during incremental process for first three truncation numbers (M) of the enhanced process.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f\,, \tag{36}$$

where u is the solution, with boundary conditions $u(0,t) = u(1,t) = 0$ and an initial condition $u(x,0) = 0$. Here ν is the viscosity coefficient, chosen as 0.01, and $f \in \mathbb{R}$ is a known forcing term. We consider the Burgers problem over a space-time domain $\Omega : [0,1] \times \mathrm{I} : [0,20]$ with f defined as

$$f(x,t) = 1 + \frac{5}{30}\sum_{k=1}^{3}\sin(k\pi t)\sin(k\pi x)\,. \tag{37}$$

This forcing term produces a solution with large fluctuations and a boundary layer near the right boundary. The problem is discretized on a mesh with 256 elements, using piecewise linear basis functions in space and a variational multiscale method. We apply a Runge-Kutta time marching scheme with a time step of $\Delta t = 0.001$ to advance the solution in time. Time interval of $[0,10]$ is used for flow development before reaching a statistical steady state of the time period $[10,20]$.

The dataset is constructed by taking a snapshot every 20 time steps from $t = 10$ to $t = 12$, resulting in a 257×100 snapshot matrix. We apply the enhanced online algorithm to analyze this dataset and compare it with reference data obtained from the standard iSVD. Fig. 3 shows the cumulative energy computed based on two estimators, e_{con} and e_{simp}, and the reference value at the terminating step when different truncation numbers (M) of the enhanced process are considered. As expected, an increasing portion of the solution energy is captured as M is increased. It is noted that the estimator e_{con} lies far below the actual value when M is small, but does converge to the actual value for $M > 10$. e_{simp} is also lower than reference value but provides a much more accurate estimate than e_{con}. In this case, a difference less than 0.1% is observed when $M = 3$. For $M = 4$, the difference is only 0.04%, which is considered sufficiently accurate. The computation time is reduced by a factor of 125 with $M = 4$. Fig. 4 presents the change of cumulative energy for these two estimators during the incremental process when M is equal to 1, 2 and 3, respectively. We can observe e_{simp} is more accurate than e_{con} over the entire incremental process.

It is noted that $F_2^{(k)}$ can be related to the aforementioned $F_1^{(k)}$ as

$$\left(F_1^{(k)}\right)^2 = F_2^{(k)} + 2\sum_{i=M+1}^{k-1}\sum_{j=i+1}^{k-1}\|\boldsymbol{U}_i'^{(k)}\|_{\mathrm{F}}\,\|\boldsymbol{U}_j'^{(k)}\|_{\mathrm{F}}\,. \tag{38}$$

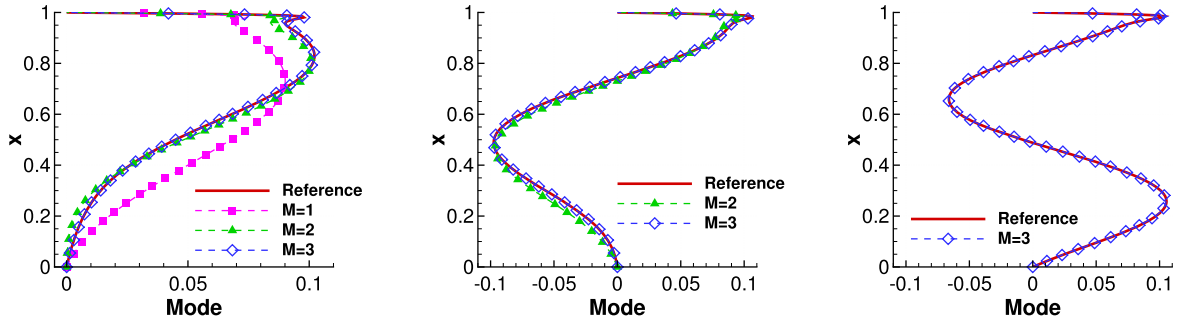


Fig. 5. Comparisons of the first three POD modes computed by enhanced and standard (reference) iSVDs, with truncations numbers of $M = 1, 2, 3$, respectively.

Table 2
First three eigenvalues computed by enhanced ($M = 1, 2, 3$) and standard (Reference value) iSVDs.

	Reference	Truncation number (M)		
		3	2	1
First eigenvalue	27.00840	27.00677	26.54177	24.13557
Second eigenvalue	11.48709	11.48611	10.99621	
Third eigenvalue	4.682270	4.678387		

Thus we have $F_2^{(k)} \leqslant (F_1^{(k)})^2$. Considering the non-negative values of $\|\widehat{U}_k\|_F$ and $F_1^{(k)}$ in Equation (24), e_{simp} is larger than e_{con}. In other words, e_{simp} provides a more accurate lower bound. Even though the performance of the enhanced iSVD depends on the problem we consider, these two estimators equip the enhanced iSVD with a posteriori knowledge of the accuracy of POD analysis. This allows us to approximate the iSVD with a desired accuracy while being much more efficient than the standard iSVD.

Fig. 5 compares the POD modes computed using different enhanced iSVDs to a standard iSVD. As shown in Fig. 5(a), the shape of first POD mode computed from the enhanced algorithm with $M = 1$ differs significantly from the reference. Increasing M reduces this discrepancy. This effect can be observed for the second and third mode as well. Table 2 shows the first three eigenvalues from these three enhanced iSVDs in comparison with the ones from standard iSVD. We can see that increasing M not only allows the POD analysis to capture more kinetic energy, but also improves the accuracy of the dominant modes. These results indicate that a truncation number that is larger than the number of selected POD modes (M) will yield more accurate versions of the selected POD modes, since this improvement is gained by interactions with other high-order modes.

4. Analyses of parallel performance

In this section, we investigate the parallel performance of the enhanced incremental SVD using both a synthetic matrix and numerical solutions from the 1D unsteady Burgers problem. For the latter case, we consider two types of data, a matrix with more DoF ($n > m$) and a matrix with more snapshots ($n < m$). All computations were performed on a 32-core node of a Linux cluster equipped with AMD Opteron(tm) Processors and 128 GB of RAM.

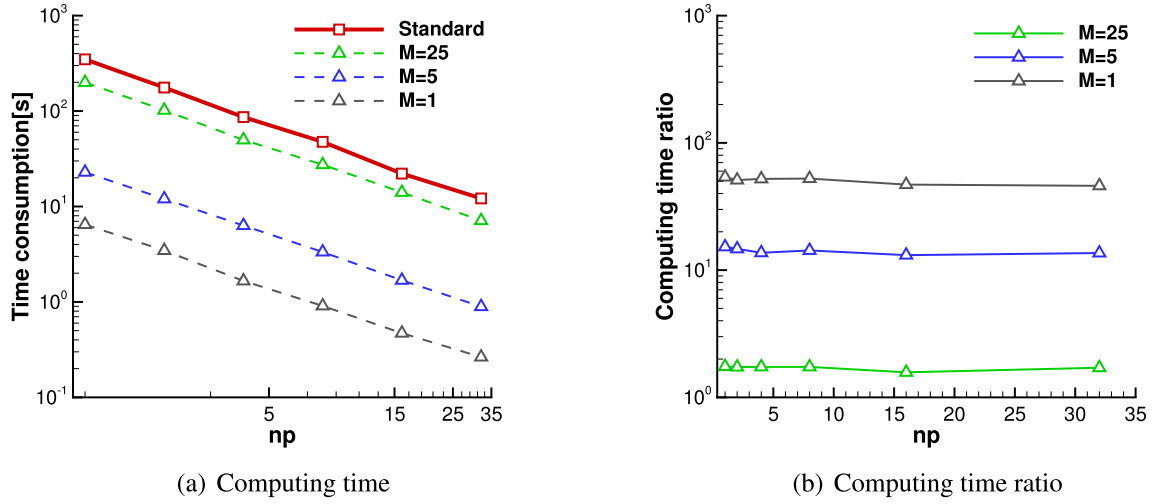
The enhanced incremental SVD algorithm is divided into six parts, as shown in right side of Algorithm 2, in order to identify the main contributions to computing cost. These consist of projection, SVD solution, LSV (left singular vectors) update, enhanced process, small SV (singular values) truncation, and reorthogonalization. In addition, we define “major parallel operations” as the summation of the projection, LSV update and reorthogonalization. The behavior of their contributions will be studied in the subsequent sections.

4.1. Synthetic matrix

Strong scaling performance was evaluated using a synthetic matrix with dimension 320000×50 formed based on random numbers with a uniform distribution. Fig. 6(a) shows the total time consumption for standard and enhanced iSVDs. The computing cost of the standard iSVD is much higher than that of the enhanced iSVDs (up to two orders higher in this case). Fig. 6(b) shows the ratio of the standard iSVD’s computing time to the enhanced iSVD’s computing time. We call this time ratio as the algorithmic speedup to distinguish it from the speedup of parallel computations, and to describe the performance improved by the enhanced algorithm. The enhanced iSVDs are able to retain their significant performance advantage as the number of processors (np) is increased.

The parallel speedups for the total incremental time, major parallel operations, and LSV update are shown in Fig. 7, for a standard iSVD and three enhanced iSVDs. All of them scale well in parallel. For the LSV update, the enhanced iSVDs show a scaling similar to the standard one, just below the ideal value (a speedup proportional to the number of processors). This is observed for the major parallel operations as well, although this is less significant. Furthermore, using a smaller number M for the enhanced process can degrade the speedup to some extent. This is because the enhanced process reduces the computing cost of the major parallel operations, and thus the contribution to the computing cost from other undistributed operations becomes more and more apparent with increased number of parallel processors.

The time ratio of the computing cost for the different contributions is given in Fig. 8. Among these, the LSV update provides the largest contribution to the total computing time for the both standard and enhanced iSVDs. However, the enhanced algorithm reduces this contribution. The smaller the truncation number M of the enhanced process, the lower the contribution of the LSV update to the total cost. The enhanced process reduces the dimension of V_Q and W_Q, and thus multiplications involving those matrices are changed to relatively light computations. In contrast,



(a) Computing time (b) Computing time ratio

Fig. 6. Time consumption of standard and enhanced incremental SVDs in parallel for a 320000×50 synthetic matrix, and the computing time ratio for these enhanced iSVDs. np denotes the number of processors.

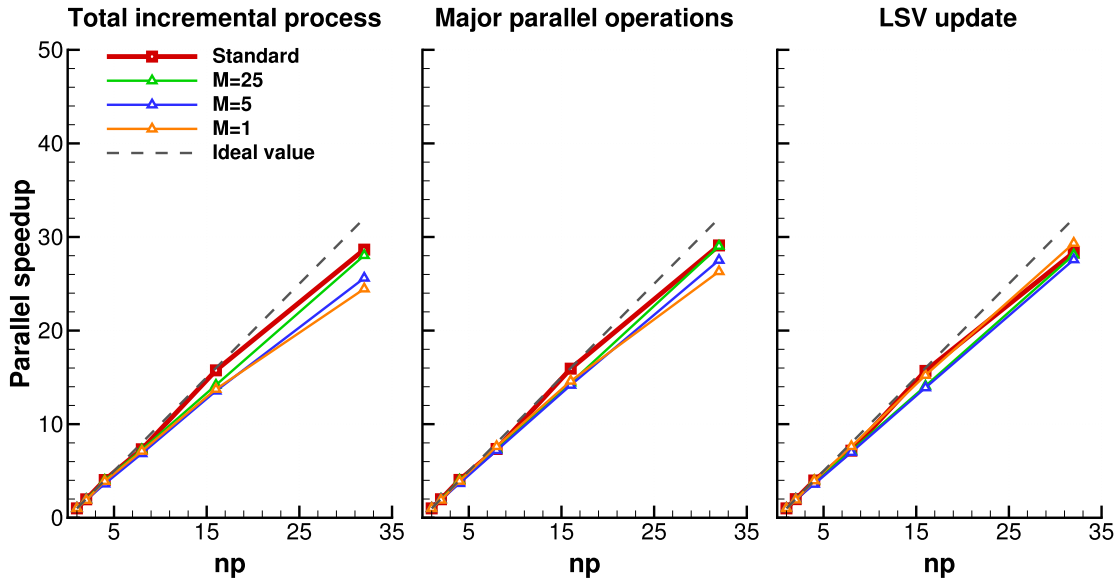


Fig. 7. The speedup of standard and enhanced parallel incremental SVDs for the total incremental time, major parallel operations and LSV update for a 320000×50 synthetic matrix. np denotes the number of processors.

for a small M, the computing cost of the projection becomes relatively significant. As the state vectors are formed from independent randomized values, reorthogonalization is not a significant contributor to this problem.

4.2. Unsteady flow problem

The Burgers problem of Section 3.3 was also used to generate large-scale matrices for parallel performance studies. We constructed two types of matrices, a matrix with more DoF and a matrix with more snapshots. We refer to the former as a deep matrix and the latter as a wide matrix. Using interpolations of the original data set, the deep matrix of 327681×200 was used to study strong scaling performance, in which each core held at least 10000 data points. The wide matrix (257×2000) was used to perform a weak scaling study.

4.2.1. A deep matrix with more DoF

In this scenario, we solve the problem with a fine mesh and assume that we only need to gather a small number of snapshots for modal analyses in order to study dominant flow structures. Such a scenario could result from the selection of snapshots described in [26,30] for example, in which linearly-dependent solutions are skipped in the POD analysis.

Fig. 9 shows the computing time of the enhanced and standard iSVDs and the algorithmic speedup of the enhanced iSVDs. We can observe that the computing time of both standard and enhanced iSVDs is consistently reduced when using more parallel processors, and that the enhanced algorithm significantly improves the computational efficiency independent of the number of processors.

Fig. 10 shows the parallel speedup of the total incremental process, major parallel operations and LSV update for the standard and enhanced incremental SVDs. For the total incremental time, good strong-scaling performance is observed, occasionally exceeding the theoretically ideal

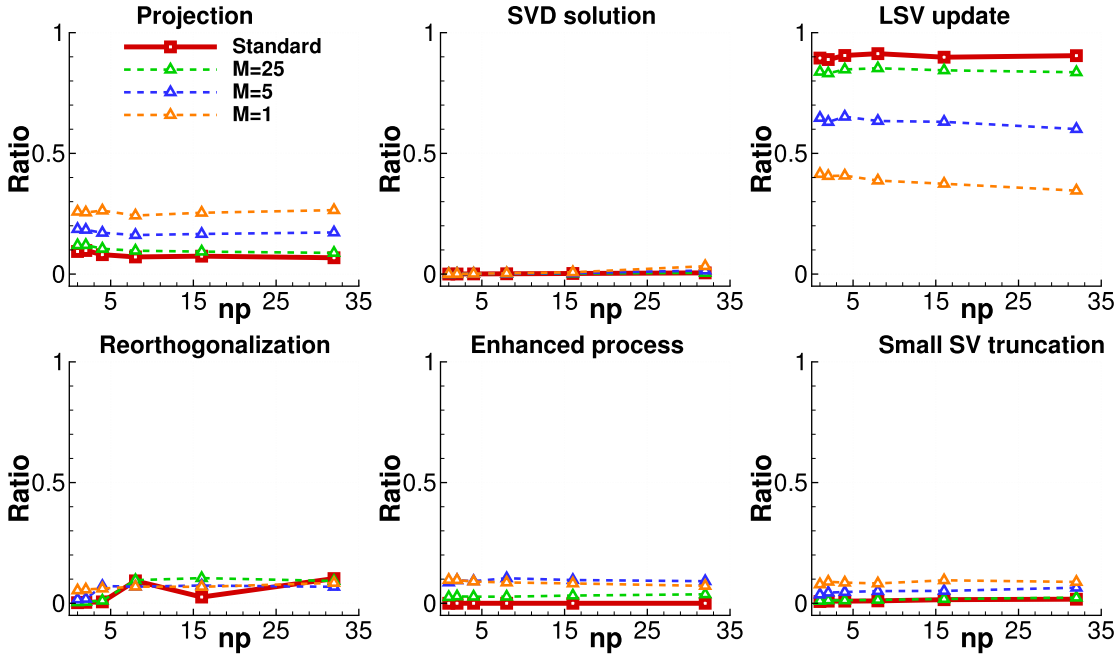
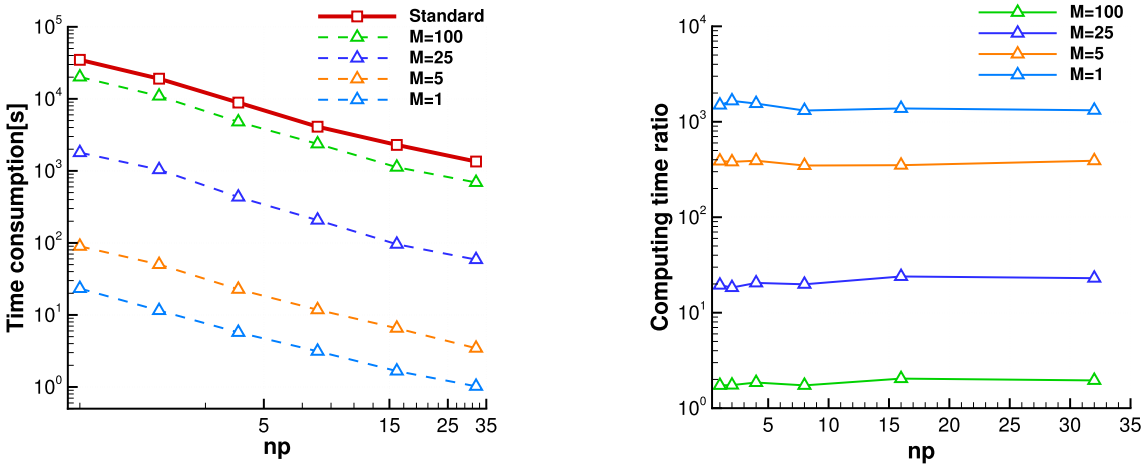


Fig. 8. The time ratio of computing cost for different parts during the standard/enhanced parallel incremental SVDs for a 320000×50 synthetic matrix, including the projection, SVD solution, LSV update, reorthogonalization, enhanced process, small SV truncation. np denotes the number of processors.



(a) Computing time (b) Computing time ratio

Fig. 9. Time consumption of standard and enhanced incremental SVDs in parallel, and the computing time ratio for the enhanced process for a deep matrix (327681×200). np denotes the number of processors.

value. The performance for major parallel operations behaves in a similar manner. However, the parallel speedup of the LSV update is well below the ideal value. The observed hyper-speedup results from the computational cost of the reorthogonalization. Fig. 11(a) presents the number of reorthogonalizations versus the number of processors. It is seen to decrease as the number of processors is increased. In this case, the computational cost of reorthogonalization benefits from reduced round-off errors when computing sums with a parallelized algorithm. Fig. 11(b) compares the non-orthogonality values computed by the standard iSVD with 1 and 32 processors respectively. The number of non-orthogonality values exceeding the threshold in 1-processor case (17) is larger than that number with 32 processors (6).

Fig. 12 shows the time ratio of each part in the standard and enhanced iSVDs. These ratios keep roughly constant for all iSVDs. The LSV update and reorthogonalization are the two major contributions to the computing cost. When using a small M, the contribution of the projection also plays an important role in the computing cost.

Normalized time consumption. Since the need for reorthogonalization is strongly affected by numerical round-off errors, the number of reorthogonalizations varies when we run the problem with different number of processors. To remove the resulting oscillations in the scaling graph, we normalize the computing time of the reorthogonalization to obtain

$$t_{\text{normal}} = t_{\text{total}} - t_{\text{reorthogonal}} + \frac{t_{\text{reorthogonal}}}{n_{\text{reorthogonal}}}, \tag{39}$$

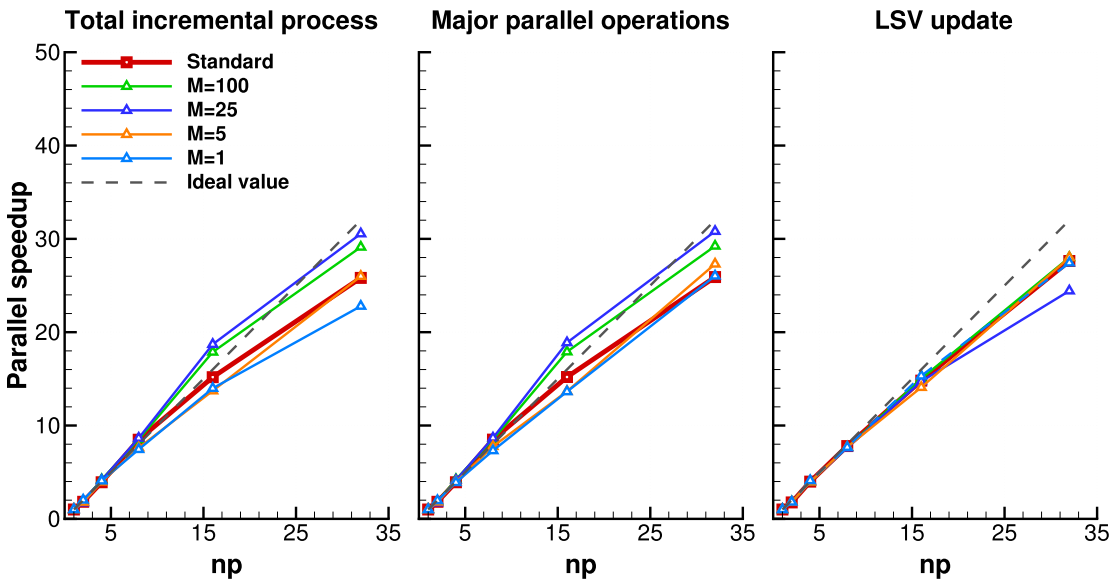
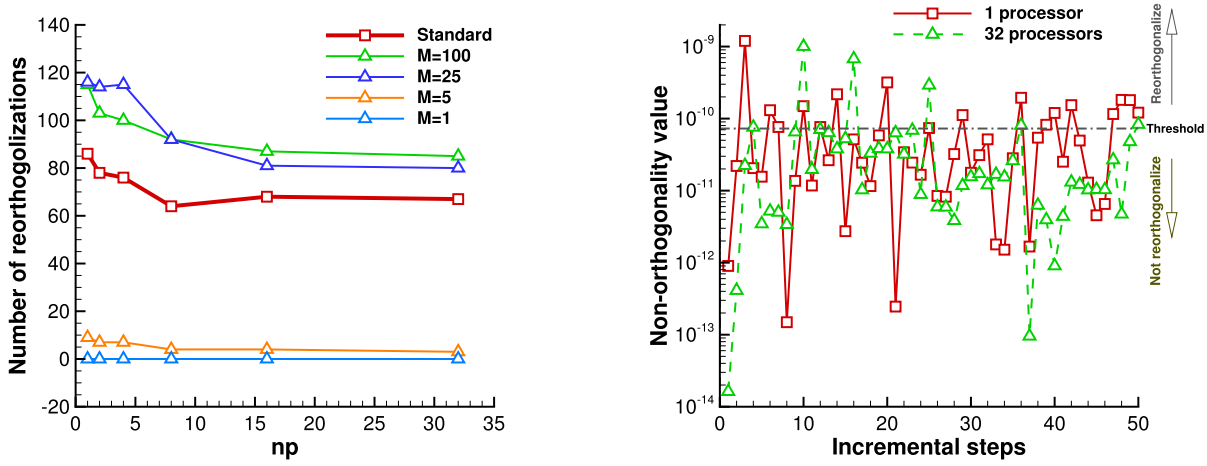


Fig. 10. Parallel speedups of standard/enhanced incremental SVDs for the total incremental time, major parallel operations and LSV update on a deep matrix (327681×200). np denotes the number of processors.



(a) Reorthogonalization number (b) Non-orthogonality values for standard iSVD

Fig. 11. The number of reorthogonalizations of the standard and enhanced incremental SVDs in parallel, and the comparison of non-orthogonality values computed for 1-processor and 32-processor cases for a deep matrix (327681×200). np denotes the number of processors.

where t_{total} and t_{normal} denote the total computing time before and after the normalization. $t_{\text{reorthogonal}}$ and $n_{\text{reorthogonal}}$ represent the computing time and the number of reorthogonalizations. When $n_{\text{reorthogonal}} = 0$, the computing time is kept the same. Otherwise the normalized value will be utilized. Fig. 13 shows the parallel speedups of normalized computing time, which now have the expected scaling, just below the ideal value.

4.2.2. *A wide matrix with more snapshots*

Another scenario may occur when we are interested in statistically steady phenomena. In this case, we consider a large number of snapshots which exceeds the number of DoF of a computational mesh, leading to a wide matrix. Fig. 14(a) presents the total computing time of the standard and enhanced iSVDs for a 257×2000 wide matrix. Here, we examine weak scaling in which the computing time ideally remains constant while the number of processors is increased. As shown in Fig. 14(b), the algorithmic speedup of the enhanced algorithm remains constant although this value is degraded for the case of $M = 1$. Fig. 15 shows the computing cost of major parallel operations and the LSV update. The cost of major parallel operations exhibits a trend similar to that of the total computing time. However, the LSV update shows a constant computing cost with increasing the number of processors. Overall, the enhanced incremental SVD shows good scaling for this wide matrix.

4.2.3. *Improvement of computational performance by the enhanced process*

The algorithmic speedup of the proposed enhanced online algorithm is shown to be constant for both deep and wide matrices. Considering that the dominant computational cost results from the LSV update, we can estimate the improvement by an a priori analysis of the computing complexity, as described in Proposition 2.

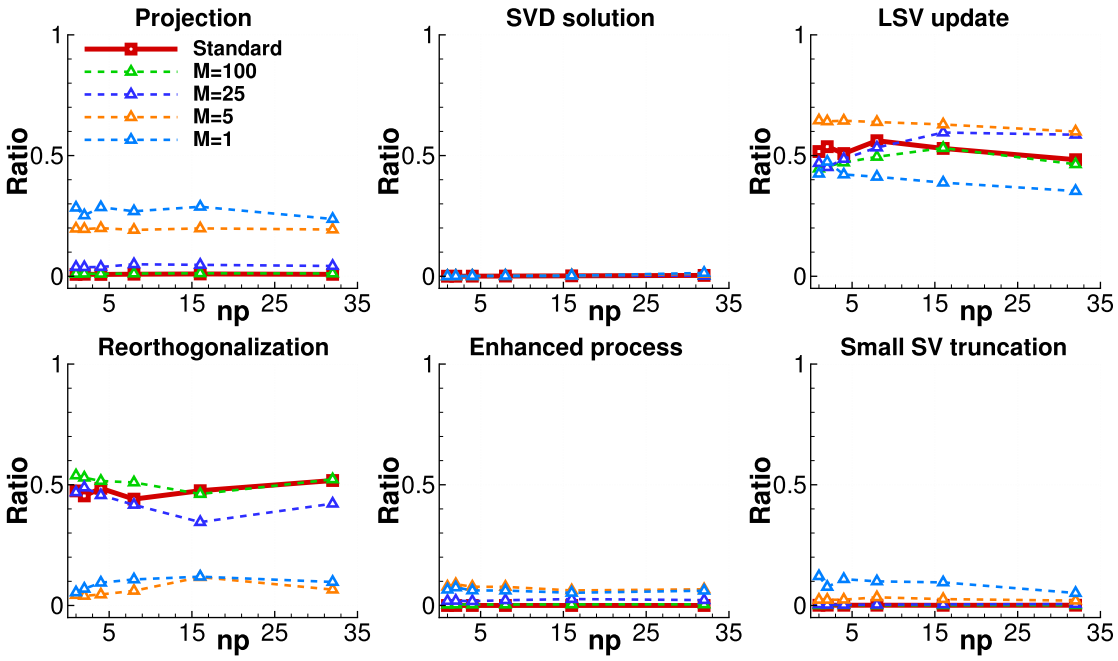


Fig. 12. The time ratio of computing cost for different parts during the standard/enhanced parallel incremental SVDs on a 327681×200 deep matrix, including the projection, SVD solution, LSV update, reorthogonalization, enhanced process, small SV truncation. np denotes the number of processors.

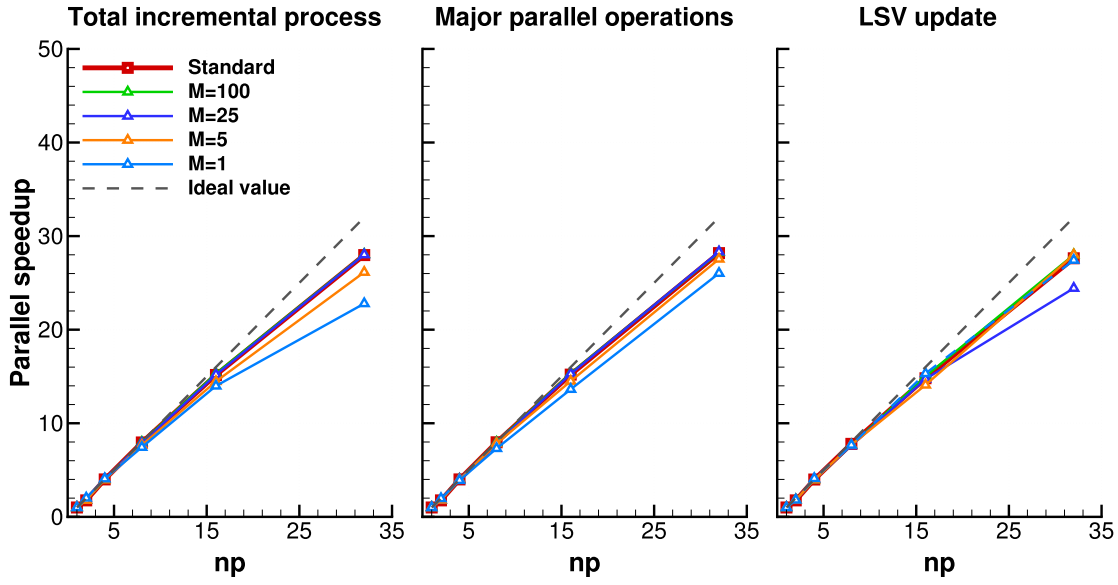


Fig. 13. Parallel speedups of standard/enhanced incremental SVDs on a deep matrix (327681×200), with the normalized computing time for the total incremental process, major parallel operations and LSV update.

Proposition 2. *Suppose we have a dense matrix* $\boldsymbol{U}^{n\times m}$. *The floating-point operations for the LSV update of the standard iSVD can be estimated to be* $\mathcal{O}(t_{\text{standard}})$,

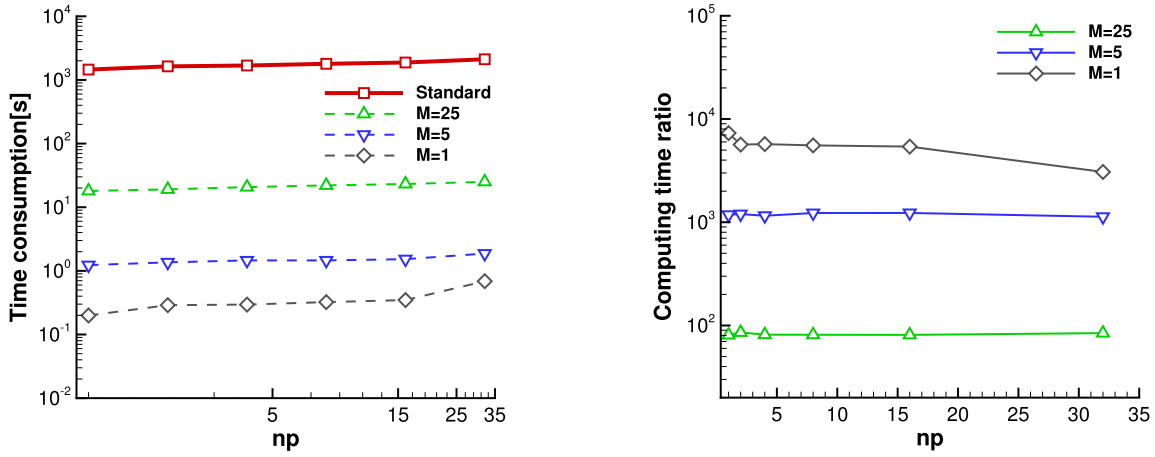
$$t_{\text{standard}} = \begin{cases} 2n[\frac{m}{6}(m+1)(2m+1)-1]+3n, & n \geqslant m \\ 2n[\frac{n}{6}(n+1)(2n+1)-1+n^2(m-n)]+3n, & n < m\,. \end{cases} \tag{40}$$

If we use $M = k_e$, $k_e < \min(n,m)$, *for the enhanced iSVD, these floating-point operations can be reduced to* $\mathcal{O}(t_{\text{enhanced}})$,

$$t_{\text{enhanced}} = 2n[\frac{k_e}{6}(k_e+1)(2k_e+1)-1+(k_e+1)^2(m-k_e)]+3n\,. \tag{41}$$

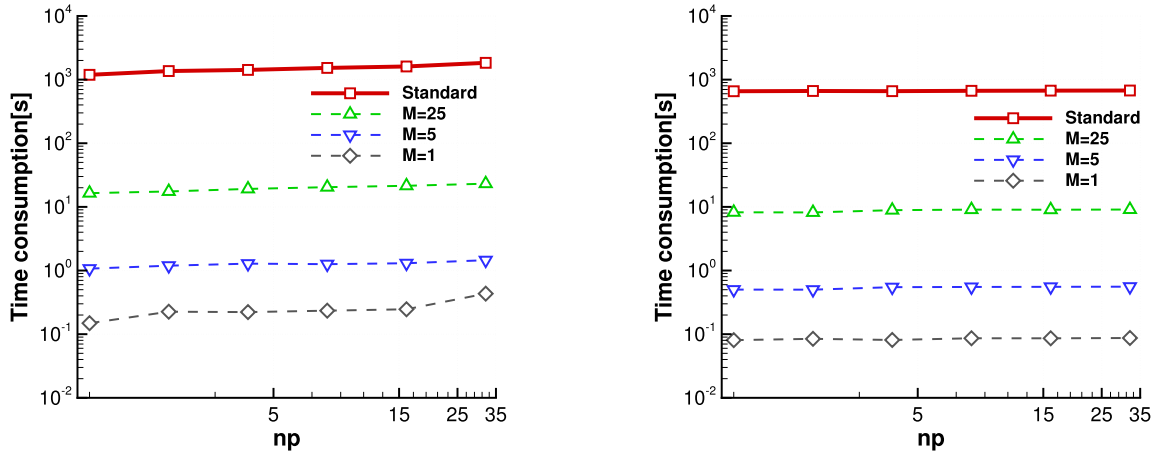
t_{enhanced} *is equivalent to* t_{standard} *when* $k_e = \min(n,m)$.

Proof. We first verify the expression for a deep matrix ($n \geqslant m$). For the standard iSVD, we need an initialization, with floating-point operations (FLOP) of $\mathcal{O}(3n)$, before proceeding to the LSV update. We can evaluate the FLOP in each LSV update as $\mathcal{O}(2nl^2)$, where l denotes the number of



(a) Time consumption (b) Computing time ratio

Fig. 14. Time consumption of standard and enhanced incremental SVDs in a weak study on a wide matrix (257×2000), and the computing time ratio for the enhanced incremental SVDs. np denotes the number of processors.



(a) Major parallel operations (b) LSV update

Fig. 15. Time consumption of the major parallel operations and LSV update with the standard and enhanced incremental SVDs during a weak scaling study for a wide matrix (257×2000). np denotes the number of processors.

snapshots. This depends on $[\boldsymbol{U}j]^{n\times l}$ and $\boldsymbol{V}_Q^{l\times l}$. The LSV update starts with the second snapshot ($l = 2$) after initialization. Therefore we can estimate the complexity of the total computation as

$$t_{\text{standard}} = 2n(2^2 + \cdots + m^2) + 3n = 2n[\frac{m}{6}(m+1)(2m+1) - 1] + 3n. \tag{42}$$

Likewise, we can compute the complexity of the enhanced iSVD by replacing the cost with $\mathcal{O}(2n(k_e+1)^2)$ after the k_e-th incremental step, leading to

$$\begin{aligned} t_{\text{enhanced}} &= 2n[2^2 + \cdots + k_e^2 + (k_e+1)^2 + (k_e+1)^2 \cdots + (k_e+1)^2] + 3n \\ &= 2n[\frac{k_e}{6}(k_e+1)(2k_e+1) - 1 + (k_e+1)^2(m-k_e)] + 3n. \end{aligned} \tag{43}$$

For a wide matrix ($n < m$), t_{enhanced} is the same as shown for the deep matrix, and t_{standard} can be computed by

$$\begin{aligned} t_{\text{standard}} &= 2n(2^2 + 3^2 + \cdots + n^2 + n^2 + \cdots + n^2) + 3n \\ &= 2n[\frac{n}{6}(n+1)(2n+1) - 1 + n^2(m-n)] + 3n. \end{aligned} \tag{44}$$

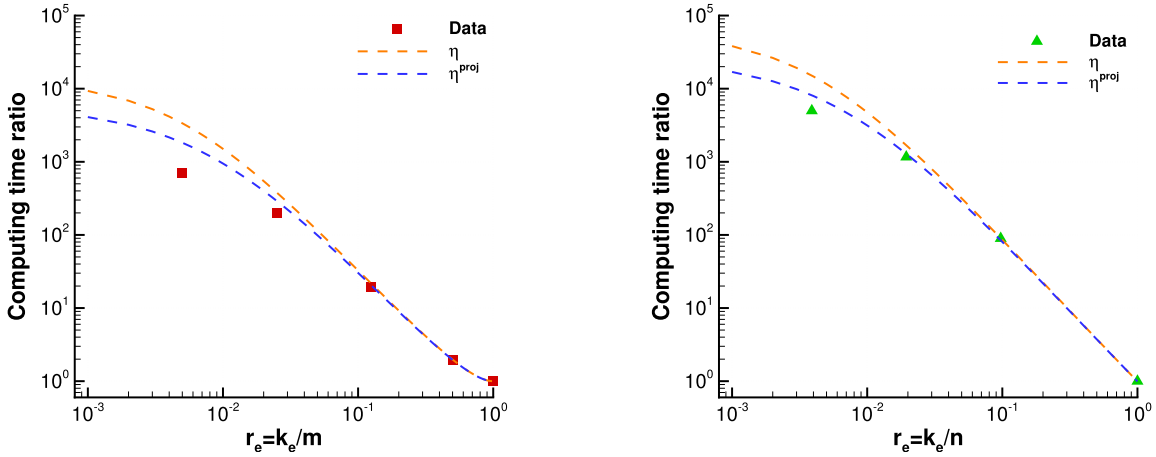
When $k_e = \min(n, m)$ for a deep or wide matrix, the enhanced iSVD is equivalent to the standard iSVD, leading to $t_{\text{enhanced}} = t_{\text{standard}}$. Consequently, Proposition 2 is confirmed. □

As the computing cost mainly results from matrix/vector multiplications, we can assume that the LSV update is the leading contribution as shown in Sections 4.1 and 4.2. Then we can estimate the algorithmic speedup provided by the enhanced online algorithm as $\mathcal{O}(\eta)$, where

$$\eta = \frac{t_{\text{standard}}}{t_{\text{enhanced}}}. \tag{45}$$

Table 3
Float-point operations in the projection part at the k-th incremental step with $\boldsymbol{V}^{n\times(k-1)}(k \geq 2)$.

	Input operation	Float-point operations
	$\boldsymbol{d} = \boldsymbol{V}^{\top}\boldsymbol{c}$	$\mathcal{O}(2n(k-1))$
	$\boldsymbol{V}\boldsymbol{d}$	$\mathcal{O}(2n(k-1))$
	$\boldsymbol{h} = \boldsymbol{c} - \boldsymbol{V}\boldsymbol{d}$	$\mathcal{O}(n)$
	$\boldsymbol{h}^{\top}\boldsymbol{h}$	$\mathcal{O}(2n)$
Total		$\mathcal{O}(4nk - n)$



(a) A deep matrix (b) A wide matrix

Fig. 16. A priori analyses on the efficiency improved by the enhanced online algorithm for incremental SVD over (a) a deep matrix (327681×200) and (b) a wide matrix (257×2000), compared with the physical data.

When the truncation number M is small, the computing cost of the LSV update is not the only dominant contribution due to the reduction of the size of $\boldsymbol{V}, \boldsymbol{V}_Q$. In such situations, the projection cost also becomes important, as shown in Figs. 8 and 12. Its computing complexity can be estimated based on the number of FLOP as summarized in Table 3.

For the standard iSVD of a deep matrix, the computing complexity of the projection over the entire incremental process is $\mathcal{O}(t_{\mathrm{s}}^{\mathrm{proj}})$, where

$$t_{\mathrm{s}}^{\mathrm{proj}} = 4n(2+3+\cdots+m) - n*(m-1) = 2n(m+2)(m-1) - n(m-1)\,. \tag{46}$$

This complexity for a wide matrix is

$$\begin{aligned} t_{\mathrm{s}}^{\mathrm{proj}} &= 4n(2+3+\cdots+n+(n+1)+\cdots+(n+1)) - n*(m-1) \\ &= 2n(n+2)(n-1) + 4n(n+1)(m-n) - n(m-1)\,, \end{aligned} \tag{47}$$

while the computing complexity for the enhanced iSVD is

$$\begin{aligned} t_{\mathrm{e}}^{\mathrm{proj}} &= 4n[2+3+\cdots+k_e+(k_e+1)+\cdots+(k_e+1)] - n(m-1) \\ &= 2n(k_e+2)(k_e-1) + 4n(k_e+1)(m-k_e) - n(m-1)\,. \end{aligned} \tag{48}$$

We can thus estimate the effective algorithmic speedup as $\mathcal{O}(\eta^{\mathrm{proj}})$,

$$\eta^{\mathrm{proj}} = \frac{t_{\mathrm{standard}} + t_{\mathrm{s}}^{\mathrm{proj}}}{t_{\mathrm{enhanced}} + t_{\mathrm{e}}^{\mathrm{proj}}}\,, \tag{49}$$

which consists of the LSV update and projection. The difference between η and η^{proj} is the impact of the projection on the incremental SVDs.

Fig. 16(a) compares the computations of η and η^{proj} using the numerical dataset from the above-mentioned deep matrix. These two estimations are similar when predicting the algorithmic speedup for a large number ratio $r_e = k_e/m$, while the η^{proj} is more accurate for small r_e values. This is because the projection cost is a more significant contribution for smaller truncation numbers k_e, i.e. a small r_e here. In addition, the performance for the wide-matrix case bears a strong resemblance to that of the deep matrix, as shown in Fig. 16(b). Note that the computing complexity is utilized to quantify the intrinsic time requirements of the algorithm, whereas the actual costs depend on the computer hardware and software implementation. Therefore the computing complexity gives only an indication of the computing time. Furthermore, the actual computing time can also be affected by other neglected operations when using a small number of r_e/k_e, as shown in Fig. 12. Overall, the estimation that includes the projection gives the most reasonable prediction for the improved efficiency. Given that $t_{\mathrm{standard}}, t_{\mathrm{enhanced}}, t_{\mathrm{s}}^{\mathrm{proj}}$ and $t_{\mathrm{e}}^{\mathrm{proj}}$ are linear functions of n for deep matrices, η^{proj} is independent of DoF (n), yielding an improvement independent of the number of processors.

5. Conclusions

We have introduced an enhanced online algorithm based on incremental SVD for modal analysis, which can efficiently perform a POD analysis on the fly. The accuracy of the method depends on the truncation number (M) of the enhanced process. Results obtained with the enhanced method converge to the results of a standard SVD for large M. Two lower-bound estimators are formulated for the a posteriori analysis of the enhanced online algorithm, with which one can estimate the accuracy of the reconstructed solution obtained from the resulting POD modes. The second estimator, e_{simp}, is shown to be more accurate for evaluating the cumulative energy captured by the selected POD modes. Numerical results demonstrate that the enhanced online algorithm has significantly better computational efficiency than the standard incremental SVD. A metric based on the number of floating-point operations has been used to estimate the speedup by the enhanced incremental SVD. Furthermore, it is shown that the proposed enhanced algorithm has good parallel scalability such that the strong reduction of computational cost is maintained in parallel computations.

The choice for truncation number, that is, the required mode number and the resulting cost reduction, is case dependent in practice. Our error estimator can guide users by detecting if the value of this user-defined parameter is appropriate. It is reasonable to assume that future users have some experience with their application field and can therefore make a reasonably conservative estimate. Even if this estimate is very conservative, the cost reduction can still be significant. Our experience with large 3D flow simulations indicates that less than 1% of the full modal basis is typically required for a physical analysis [36]. Overall, the results of this paper imply that the enhanced online algorithm will be useful for the engineering analysis of high-dimensional problems.

Acknowledgements

The authors would like to acknowledge financial support by the China Scholarship Council.

References

[1] K. Taira, S.L. Brunton, S.T.M. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordeyev, V. Theofilis, L.S. Ukeiley, Modal analysis of fluid flows: an overview, AIAA J. 55 (2017) 4013–4041.
[2] P. Holmes, J.L. Lumey, G. Berkooz, C.W. Rowley, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge Monographs on Mechanics, 2nd ed., Cambridge University Press, New York, 2012.
[3] J.N. Kutz, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, first ed., Oxford University Press, Oxford, 2013.
[4] C.W. Rowley, S.T. Dawson, Model reduction for flow analysis and control, Annu. Rev. Fluid Mech. 49 (2017) 387–417.
[5] K.C. Hall, J.P. Thomas, E.H. Dowell, Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows, AIAA J. 38 (2000).
[6] T.P. Miyanawala, R.K. Jaiman, Decomposition of wake dynamics in fluid–structure interaction via low-dimensional models, J. Fluid Mech. 867 (2019) 723–764.
[7] M.J. Zahr, C. Farhat, Progressive construction of a parametric reduced-order model for PDE-constrained optimization, Int. J. Numer. Methods Eng. 102 (2015) 1111–1135.
[8] N. Ferro, S. Micheletti, S. Perotto, POD-assisted strategies for structural topology optimization, Comput. Math. Appl. 77 (2019) 2804–2820.
[9] T. Bui-Thanh, K. Willcox, O. Ghattas, Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications, AIAA J. 46 (2008) 2520–2529.
[10] X. Sun, J.-I. Choi, Non-intrusive reduced-order modeling for uncertainty quantification of space-time-dependent parameterized problems, Comput. Math. Appl. 87 (2021).
[11] S.S. Ravindran, A reduced-order approach for optimal control of fluids using proper orthogonal decomposition, Int. J. Numer. Methods Fluids 34 (2000) 425–448.
[12] M. Bergmann, L. Cordier, J.-P. Brancher, Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model, Phys. Fluids 17 (2005) 097101.
[13] T. Akman, Local improvements to reduced-order approximations of optimal control problems governed by diffusion-convection-reaction equation, Comput. Math. Appl. 70 (2015) 104–131.
[14] G. Berkooz, P. Holmes, J.L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annu. Rev. Fluid Mech. 25 (1993) 37.
[15] J.L. Lumey, Stochastic Tools in Turbulence, Applied Mathematics and Mechanics, Academic Press, New York, 1970.
[16] L. Sirovich, Turbulence and the dynamics of coherent structures part I: coherent structures, Q. Appl. Math. 45 (1987).
[17] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, New York, 1990.
[18] B. Champagne, Adaptive eigendecomposition of data covariance matrices based on first-order perturbations, IEEE Trans. Signal Process. 42 (1994) 2758–2770.
[19] Chao Xu, Lixiang Luo, E. Schuster, On recursive proper orthogonal decomposition via perturbation theory with applications to distributed sensing in cyber-physical systems, in: Proceedings of the 2010 American Control Conference, IEEE, Baltimore, MD, 2010, pp. 4905–4910.
[20] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2011).
[21] C. Bach, D. Ceglia, L. Song, F. Duddeck, Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems, Int. J. Numer. Methods Eng. 118 (2019).
[22] M. Brand, Incremental singular value decomposition of uncertain data with missing values, in: European Conference on Computer Vision, Springer, 2002, pp. 707–720.
[23] C. Baker, K. Gallivan, P. Van Dooren, Low-rank incremental methods for computing dominant singular subspaces, Linear Algebra Appl. 436 (2012).
[24] H. Fareed, J.R. Singler, Y. Zhang, J. Shen, Incremental proper orthogonal decomposition for PDE simulation data, Comput. Math. Appl. 75 (2018) 1942–1960.
[25] H. Fareed, J.R. Singler, Error analysis of an incremental proper orthogonal decomposition algorithm for PDE simulation data, J. Comput. Appl. Math. 368 (2020) 112525.
[26] G.M. Oxberry, T. Kostova-Vassilevska, W. Arrighi, K. Chand, Limited-memory adaptive snapshot selection for proper orthogonal decomposition, Int. J. Numer. Methods Eng. 109 (2017) 198–217.
[27] K. Li, T.-Z. Huang, L. Li, S. Lanteri, POD-based model order reduction with an adaptive snapshot selection for a discontinuous Galerkin approximation of the time-domain Maxwell's equations, J. Comput. Phys. 396 (2019) 106–128.
[28] C. Vezyris, E. Papoutsis-Kiachagias, K. Giannakoglou, On the incremental singular value decomposition method to support unsteady adjoint-based optimization, Int. J. Numer. Methods Fluids 91 (2019) 315–331.
[29] X. Li, S. Hulshoff, S. Hickel, Towards adjoint-based mesh refinement for Large Eddy Simulation using reduced-order primal solutions: preliminary 1D Burgers study, Comput. Methods Appl. Mech. Eng. 379 (2021) 113733.
[30] P. Phalippou, S. Bouabdallah, P. Breitkopf, P. Villon, M. Zarroug, 'On-the-fly' snapshots selection for Proper Orthogonal Decomposition with application to nonlinear dynamics, Comput. Methods Appl. Mech. Eng. 367 (2020) 113120.
[31] W. Arrighi, G. Oxberry, K. Chand, libROM User Guide and Design, Technical Report, Lawrence Livermore National Laboratory, 2015.
[32] M.A. Iwen, B.W. Ong, A distributed and incremental SVD algorithm for agglomerative data analysis on large networks, SIAM J. Matrix Anal. Appl. 37 (2016) 1699–1718.
[33] D.A. Jackson, Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches, Ecology 74 (1993) 2204–2214.
[34] B.R. Noack, K. Afanasiev, M. Morzyński, 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.
[35] J. Caldwell, P. Wanless, A.E. Cook, A finite element approach to Burgers' equation, Appl. Math. Model. 5 (1981) 189–193.
[36] V. Pasquariello, S. Hickel, N.A. Adams, Unsteady effects of strong shock-wave/boundary-layer interaction at high Reynolds number, J. Fluid Mech. 823 (2017) 617–657.