Received: 17 November 2017 | Revised: 22 January 2019 | Accepted: 18 April 2019

DOI: 10.1002/nla.2246

RESEARCH ARTICLE

WILEY

Fast approximate truncated SVD

Serge L. Shishkin[1] | Arkadi Shalaginov[2] | Shaunak D. Bopardikar[3]

[1]United Technologies Research Center, East Hartford, Connecticut

[2]Qcentive, Boston, Massachusetts

[3]Electrical and Computer Engineering Department, Michigan State University, East Lansing, Michigan

Correspondence
Serge L. Shishkin, United Technologies Research Center, 411 Silver Lane, MS 129-15, East Hartford, CT 06447.
Email: shishks@utrc.utc.com

Funding information
Office of Naval Research, Grant/Award Number: N00014-16-C-2009

Summary

This paper presents a new method for the computation of truncated singular value decomposition (SVD) of an arbitrary matrix. The method can be qualified as *deterministic* because it does not use randomized schemes. The number of operations required is asymptotically lower than that using conventional methods for nonsymmetric matrices and is at a par with the best existing deterministic methods for unstructured symmetric ones. It slightly exceeds the asymptotical computational cost of SVD methods based on randomization; however, the error estimate for such methods is significantly higher than for the presented one. The method is one-pass, that is, each value of the matrix is used just once. It is also readily parallelizable. In the case of full SVD decomposition, it is exact. In addition, it can be modified for a case when data are obtained sequentially rather than being available all at once. Numerical simulations confirm accuracy of the method.

KEYWORDS

incremental computation, matrix factorization, scalability, singular value decomposition

1 | INTRODUCTION

Truncated (partial) singular value decomposition (SVD) is a basic linear algebraic operation; therefore, any acceleration of that procedure has a significant impact in many application areas. Currently, there are two approaches to the problem: "stochastic" and "deterministic"; the former are based on randomization schemes, whereas the latter do not use them. The most efficient (in terms of flops) deterministic methods require $O(rnN)$ operations for a matrix $\boldsymbol{X}$ of dimension $n \times N$ ($n \leq N$) and rank r of the approximation (see the works of Demmel et al.[1] and Gu et al.[2]). Better estimates are available only for $r = n$ or for matrices of some special structure. In contrast, the truncated SVD of a symmetric matrix requires $O(r^{\rho}n^2)$ operations (see the work of Demmel et al.[1]), where the numerical value of ρ is equal to either 0.807 (if Strassen matrix multiplication algorithm[3] is used) or 0.376 (if the work of Coppersmith–Winograd[4] is used). It is desirable to bring the complexity of nonsymmetric SVD at par with that of symmetric ones.

Several stochastic methods were proposed during last decade (see the papers[5–7] and references within). The best one known to the authors (referred, henceforth, as *Randomized algorithm*) requires $O(nN \log r + Nr^2)$ operations.[6] However, due to their nature, these methods are not exact. The most accurate one has an approximation error as high as $11\sqrt{rn}\,\sigma_{r+1}$, where σ_{r+1} stands for the $(r+1)$-th singular value of the matrix. Besides, any error control they can offer is probabilistic.

In this paper, we propose a deterministic method with rigorous error control. The number of operations required is either $O(r^{0.807}nN)$ or $O(r^{0.376}nN)$, depending on the matrix multiplication method used. It is the same asymptotic estimate

* Shaunak D. Bopardikar was with UTRC when this work was carried out.

as the one for existing SVD methods for symmetric matrices ($n = N$). The Frobenius and L_2 norms of the approximation error of our method are both upper bounded by

$$3\sqrt{n-r}(\log_2 n - \log_2 r + 1)\sigma_{r+1}$$

(for a more exact estimate, see Theorem 2 and its corollaries). Note that the estimate does not depend on the larger dimension N of the matrix $\boldsymbol{X}$. To the best of our knowledge, such accuracy betters any error estimate for existing methods other than *exact SVD*. Moreover, the method has several attractive features: it is *one-pass* (i.e., each entry of the matrix is used just once) and is *readily parallelizable*. In the case of streaming data, when the matrix columns or rows are obtained successively over time, the factorization can be computed as new data become available, similar to the model used in other works.[7,8] The total asymptotical cost is the same as if all the data are available from the beginning.

The general scheme of the proposed method is similar to the one used in the papers.[9,10] However, the key difference is that our method does not use randomization as shown in the work of Ambikasaran et al.[9] or any special structure of the matrix as shown in the work of Vogel et al.[10] Even more important, it has two new features. The first is a *reverse* order of computing the orthogonal matrix of singular vectors of $\boldsymbol{X}$. The second is the use of an intermediate factorization of rank higher than r. The first feature is crucial to achieve $O(Nnr^{\rho})$ asymptotical computational cost of the algorithm. The second enables a significant improvement of approximation accuracy because less information is truncated off during intermediate steps, as compared to the conventional case of constant factorization rank.

The emphasis of this paper is an introduction and theoretical justification of a new numerical scheme rather than the development of highly optimized ready-to-use software. The latter should follow; however, it must be considered as a separate task because it constitutes significant work and requires different skills. For this reason, numerical simulations in this paper are focused on algorithm accuracy, with brief comments on runtime complexity.

The notation in this paper is as per the following convention: Matrices are denoted by bold capital letters, vectors by bold small letters, and scalars by nonbold letters.

This paper has the following structure. The general scheme of the algorithm is laid out in Section 2. The estimate of the asymptotical computational cost is performed in Section 3. Section 4 presents the accuracy estimate. Results of numerical simulations are discussed in Section 5. Section 6 contains conclusions.

2 | METHOD DESCRIPTION

The problem of truncated SVD can be formulated as follows. For any given matrix $\boldsymbol{X} \in \mathbb{R}^{n\times N}$ and a number r such that $r \le \operatorname{rank}(\boldsymbol{X}) \le n \le N$, we need to construct matrices $\boldsymbol{V} \in \mathbb{R}^{N\times r}$, $\boldsymbol{U} \in \mathbb{R}^{n\times r}$, $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$ to make the approximation

$$\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T \tag{1}$$

be as accurate as possible. The matrix $\boldsymbol{Q}$ is diagonal and the column orthogonality condition (further to be referred just as orthogonality, for brevity)

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I} \tag{2}$$

holds. Here and below, $\boldsymbol{I}$ stands for identity matrix. Accuracy of approximation is measured in either L_2 or Frobenius norm:

$$\|\boldsymbol{A}\|_2 = \max_{\boldsymbol{x}:|\boldsymbol{x}|=1} |\boldsymbol{A}\boldsymbol{x}|, \qquad \|\boldsymbol{A}\|_F = \sqrt{\operatorname{Tr}(\boldsymbol{A}^T\boldsymbol{A})}.$$

Optimal solution of (1) is achieved when the diagonal of $\boldsymbol{Q}$ contains r largest singular values of $\boldsymbol{X}$, and matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ hold corresponding right and left singular vectors. It is easy to verify[11] that the optimal solution of (1) must satisfy the condition

$$\boldsymbol{V}^T(\boldsymbol{X}^T\boldsymbol{X})\,\boldsymbol{V} = \boldsymbol{Q}^2, \qquad \boldsymbol{Q} > \boldsymbol{0}. \tag{3}$$

Fast factorization constructed in this paper is suboptimal. Still, it is reasonable to require that it satisfies the optimality condition (3). Hence, we impose (3) as a requirement.

We reduce the problem defined by (1)–(3) to another one, important in its own right: construct matrices $\boldsymbol{V} \in \mathbb{R}^{N\times r}$, $\boldsymbol{D} \in \mathbb{R}^{r\times r}$ such that the approximation

$$\boldsymbol{C} = \boldsymbol{X}^T\boldsymbol{X} \approx \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T \tag{4}$$

is as accurate as possible, where matrix $\boldsymbol{D}$ is diagonal positive semidefinite and $\boldsymbol{V}$ satisfies orthogonality condition (2). The pair $(\boldsymbol{V}, \boldsymbol{D})$ will be called r-truncated factorization of the correlation matrix $\boldsymbol{C}$. It will be shown in Section 4 that, once

such a factorization is constructed, the solution matrices $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{Q}$ of (1) can be computed as follows:

$$\boldsymbol{Q} = \boldsymbol{D}^{1/2}, \qquad \boldsymbol{U} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{Q}^{-1} \tag{5}$$

with $\boldsymbol{V}$ being the same for both problems.

Below, we will need a procedure for matrix multiplication. Two most used numerical schemes at the moment are Strassen[3] and Coppersmith–Winograd[4] algorithms. For multiplication of matrices of dimensions m_1, m_2, m_3 (e.g., $m_1 \times m_2$ and $m_2 \times m_3$ or $m_2 \times m_1$ and $m_1 \times m_3$, etc.) such that $m_1, m_2 \geq m_3$, Strassen algorithm requires $O(m_1 m_2 m_3^{0.807})$ operations, and Coppersmith–Winograd one — $O(m_1 m_2 m_3^{0.374})$ operations. While the second asymptotic estimate is better, in practice, the method by Strassen outperforms the one by Coppersmith–Winograd unless the matrices are extremely big. In this paper, we do not specify which matrix multiplication method must be used; we just assume that it is fixed and requires $O(m_1 m_2 m_3^{\rho})$ operations, where $\rho = 0.807$ or $\rho = 0.376$ depending on the method used. If other methods are developed in the future, our approach will accommodate it with the corresponding value of the term ρ.

We also rely on a method for exact r-truncated factorization of any symmetric matrix $\boldsymbol{A} = \boldsymbol{A}^T \in \mathbb{R}^{m \times m}$, that is, computation of the diagonal matrix $\boldsymbol{D}$ containing r maximal eigenvalues of $\boldsymbol{A}$ and the matrix $\boldsymbol{U}$ containing corresponding eigenvectors. In particular, these matrices satisfy the following statement:

$$\boldsymbol{D} = \boldsymbol{U}^T \boldsymbol{A} \boldsymbol{U}\,, \qquad \boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}\,. \tag{6}$$

One way of doing it is by using a modification of full-eigenvalue decomposition method[12] presented in the work of Demmel et al.[1] As shown in the work of Demmel et al.,[1] that method has complexity $O(m^{2+\rho})$. We do not assume though that the method[1,12] is optimal for our needs because it performs full rather than partial decomposition and does not exploit special structure and nonnegative definiteness of the matrix. Any truncated eigenvalue decomposition method is applicable here as long as its complexity is $O(m^{2+\rho})$ or better. Without further specification, we will refer to it as *standard eigenvalue decomposition method*.

We propose to factorize $\boldsymbol{C}$ in a *divide-and-conquer* manner. We start with blocks of small fixed dimension q and double the size of factorized matrix at each step of the method. After the factorization of intermediate blocks, we update only the matrix $\boldsymbol{D}$. Orthogonal matrices $\boldsymbol{V}$ are not updated. Instead, the matrix factors are stored to recover the matrix $\boldsymbol{V}$ later, after the factorization of the whole correlation matrix $\boldsymbol{C}$ is complete.

For simplicity and without loss of generality, we assume that $N = 2^K q$ for some $K \in \mathbb{N}$. Let us define block sizes $k_j = 2^j q, j = 0, \ldots, K$. For each j, the matrix $\boldsymbol{X}$ is split into submatrices $\boldsymbol{X}_{i,j}$ of size $n \times k_j$ (see Figure 1):

$$\begin{aligned} \boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_{1,j} & \ldots & \boldsymbol{X}_{2^{K-j},j} \end{bmatrix}, \\ \boldsymbol{X}_{i,j} &= \begin{bmatrix} \boldsymbol{x}_{((i-1)k_j+1)} & \cdots & \boldsymbol{x}_{(ik_j)} \end{bmatrix}, \qquad i = 1, \ldots, 2^{K-j};\ j = 0, \ldots, K. \end{aligned} \tag{7}$$

Let us consider the diagonal blocks of the matrix $\boldsymbol{X}^T\boldsymbol{X}$ (see Figure 2):

$$\boldsymbol{C}_{i,j} = \boldsymbol{X}_{i,j}^T \boldsymbol{X}_{i,j} \in \mathbb{R}^{k_j \times k_j}, \qquad i = 1, \ldots, 2^{K-j};\ j = 0, \ldots, K.$$

The second index j will be called the level of the submatrix.

The main idea of the algorithm can be described as follows. We will construct truncated factorization for all matrices $\boldsymbol{C}_{i,j}$ starting from $j = 0$ and then increasing j until we reach $j = K$ and thus obtain factorization of whole $\boldsymbol{X}^T\boldsymbol{X}$. For each $j > 0$, the matrices $\boldsymbol{C}_{i,j}$ are not constructed in explicit form. Rather, previously computed factorizations of the matrices $\boldsymbol{C}_{2i-1,j-1}, \boldsymbol{C}_{2i,j-1}$ and some auxiliary matrices are used for the *reduction* of the matrix $\boldsymbol{C}_{i,j}$ to a smaller size. Then, this *reduced* matrix is factorized using the standard eigenvalue factorization; auxiliary matrix $\widetilde{\boldsymbol{X}}_{i,j}$ is computed and the process iterates.

One of the key features of algorithm is gradual adjustment of the rank of truncated factorization as the hierarchical process progresses. Indeed, that rank is not necessarily equal to r. We rather assume that it depends on the level j, which

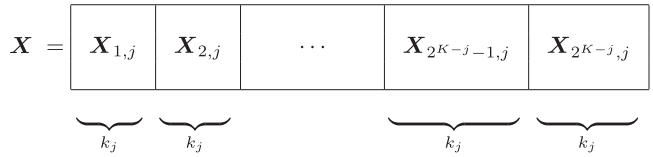

FIGURE 1 Splitting of the matrix $\boldsymbol{X}$ on level j

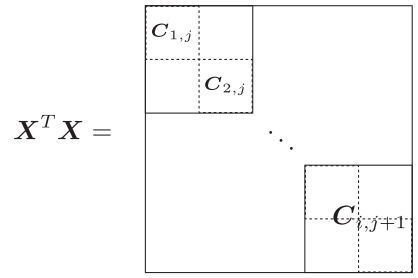

FIGURE 2 Hierarchical construction of the blocks $\boldsymbol{C}_{i,j}$

is denoted by r_j and is treated as an algorithm parameter, with the following constraints:

$$r_K = r; \qquad r_j \geq \max(r, r_{j+1}/2), \qquad \text{for any } j < K.$$

In particular, we consider the assignment

$$r_j = \min\left(n, \max\left(r, \lfloor c k_j^{\alpha} \rfloor\right)\right), \qquad j = 0, \ldots, K-1, \tag{8}$$

where c, α are parameters set a priori, such that

$$c > 0, \qquad 0 \leq \alpha < (2+\rho)^{-1}. \tag{9}$$

Note that the value of these parameters may depend on n, N, r. The cases $\alpha = 0$ (i.e., $r_j \equiv r$) and $\alpha = \frac{1}{2+\rho}\left(1 - \frac{(1-\rho)}{3}\frac{r}{n}\right) \approx \frac{1}{3}$ are of special interest for the following reason:

- $\alpha = \mathbf{0}$: This case represents the simplest option: The ranks of factorization of working blocks and the corresponding dimensions of auxiliary submatrices are constant;
- $\alpha \approx \mathbf{1/3}$: Such parameter value appears to be the optimal as can be seen below.

For this reason, most considerations below, including the key theorems, are focused on these two cases.
The proposed method can be formally described as follows.

I. At the level $j = 0$, compute the matrices $\boldsymbol{C}_{1,0}, \ldots, \boldsymbol{C}_{2^K,0}$ and perform their exact r_0-truncated factorization:

$$\boldsymbol{C}_{i,0} \approx \boldsymbol{U}_{i,0}\boldsymbol{D}_{i,0}\boldsymbol{U}_{i,0}^T, \qquad i = 1, \ldots, 2^K,$$

using standard eigenvalue decomposition method (i.e., the works of Demmel et al.[1] and Bai et al.[12]). Compute matrices

$$\widetilde{\boldsymbol{X}}_{i,0} = \boldsymbol{X}_{i,0}\boldsymbol{U}_{i,0} \in \mathbb{R}^{n \times r_0}, \qquad i = 1, \ldots, 2^K. \tag{10}$$

Matrices $\boldsymbol{U}_{i,0}, \boldsymbol{D}_{i,0}, \widetilde{\boldsymbol{X}}_{i,0}$ are stored for future use.

II. For level $j > 0$, and for any $i = 1, \ldots, 2^{K-j}$, compute r_j-truncated decomposition of $\boldsymbol{C}_{i,j}$ only after blocks $\boldsymbol{C}_{2i-1,j-1}$ and $\boldsymbol{C}_{2i,j-1}$ are processed and the matrices $\boldsymbol{D}_{s,j-1}, \widetilde{\boldsymbol{X}}_{s,j-1}$ are constructed:

$$\boldsymbol{C}_{s,j-1} \approx \boldsymbol{U}_{s,j-1}\boldsymbol{D}_{s,j-1}\boldsymbol{U}_{s,j-1}^T, \qquad \widetilde{\boldsymbol{X}}_{s,j-1} = \boldsymbol{X}_{s,j-1}\boldsymbol{U}_{s,j-1}, \qquad s = 2i-1, 2i. \tag{11}$$

First, let us construct the $(2r_{j-1} \times 2r_{j-1})$ matrix

$$\boldsymbol{P}_{i,j} = \begin{bmatrix} \boldsymbol{D}_{2i-1,j-1} & \widetilde{\boldsymbol{X}}_{2i-1,j-1}^T\widetilde{\boldsymbol{X}}_{2i,j-1} \\ \widetilde{\boldsymbol{X}}_{2i,j-1}^T\widetilde{\boldsymbol{X}}_{2i-1,j-1} & \boldsymbol{D}_{2i,j-1} \end{bmatrix}, \tag{12}$$

and perform its exact r_j-truncated factorization using the standard eigenvalue decomposition method:

$$\boldsymbol{P}_{i,j} \approx \boldsymbol{U}_{i,j}\boldsymbol{D}_{i,j}\boldsymbol{U}_{i,j}^T, \qquad \boldsymbol{U}_{i,j}^T\boldsymbol{U}_{i,j} = \boldsymbol{I}, \qquad \boldsymbol{U}_{i,j} \in \mathbb{R}^{2r_{j-1} \times r_j}, \boldsymbol{D}_{i,j} \in \mathbb{R}^{r_j \times r_j}. \tag{13}$$

Then, we compute

$$\widetilde{\boldsymbol{X}}_{i,j} = \left[\widetilde{\boldsymbol{X}}_{2i-1,j-1}\ \widetilde{\boldsymbol{X}}_{2i,j-1}\right]\boldsymbol{U}_{i,j}. \tag{14}$$

Thereafter, the matrices $\boldsymbol{D}_{2i-1,j-1}, \widetilde{\boldsymbol{X}}_{2i-1,j-1}, \boldsymbol{D}_{2i,j-1}, \widetilde{\boldsymbol{X}}_{2i,j-1}$ need not be stored in memory. Instead, matrices $\boldsymbol{U}_{i,j}, \boldsymbol{D}_{i,j}, \widetilde{\boldsymbol{X}}_{i,j}$ (but not $\boldsymbol{P}_{i,j}$) are stored for future use. Matrices $\boldsymbol{U}_{2i-1,j-1}, \boldsymbol{U}_{2i,j-1}$ are stored in memory as well.

III. Step **II** is repeated for increasing values of i, j until $j = K$ is processed. Thereafter, we assign $\boldsymbol{D} = \boldsymbol{D}_{1,K}$ and save the matrix $\widetilde{\boldsymbol{X}}_{1,K}$ for future use. We also introduce a matrix $\boldsymbol{V}_K$ and assign $\boldsymbol{V}_K = \boldsymbol{U}_{1,K}$.

IV. Iteratively decreasing j from $K-1$ to 0, we construct matrices

$$\boldsymbol{V}_j = \begin{bmatrix} \boldsymbol{U}_{1,j} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{U}_{2^{K-j},j} \end{bmatrix} \boldsymbol{V}_{j+1}, \qquad j = K-1, \ \dots, 0. \tag{15}$$

During operation (15), we exploit the block-diagonal, sparse matrix structure on the right-hand side: Each $\boldsymbol{U}_{i,j}$ is multiplied on corresponding submatrix (a strip) of $\boldsymbol{V}_{j+1}$. Once $j = 0$ is processed, assign $\boldsymbol{V} = \boldsymbol{V}_0$. The factorization $(\boldsymbol{V}, \boldsymbol{D})$ of the matrix $\boldsymbol{C}$ is constructed. To complete the factorization of $\boldsymbol{X}$, we define matrices $\boldsymbol{U}, \boldsymbol{Q}$ as

$$\boldsymbol{Q} = \boldsymbol{D}^{1/2}, \qquad \boldsymbol{U} = \widetilde{\boldsymbol{X}}_{1,K} \boldsymbol{Q}^{-1}. \tag{16}$$

Remark 1.

- In (16), the invertibility of the matrix Q follows from its positive-definiteness proven below in Theorems 2 and 3.
- One can observe that each entry of the matrix $\boldsymbol{X}$ is used just once, at Step 1 of the algorithm, in computation of the small $q \times q$ matrix $\boldsymbol{C}_{i,0}$.
- The matrices $\boldsymbol{P}_{ij}$ are nonnegative definite and have a special structure: Their upper left and lower right blocks are diagonal. Potentially, these features may be useful for designing special fast factorization algorithms for these matrices.

3 | ASYMPTOTICAL COMPUTATIONAL COST OF THE ALGORITHM

The algorithm parameters q, c, α can be chosen freely as long as constraints (9) and $r \le q$ hold. We recommend choosing

$$q = c_q r, \qquad c = c_0 r^{1-\alpha}, \qquad \alpha = \frac{1}{2+\rho}\left(1 - \frac{(1-\rho)}{3}\frac{r}{n}\right) \ge \frac{1}{3}, \tag{17}$$

where c_q, c_0 are constants; $c_0, c_q \ge 1$. This choice is dictated by the fact that higher values of c, α lead to a higher rank of truncated factorization of submatrices $\boldsymbol{C}_{i,j}$. That results in better accuracy of the overall factorization. On the other hand, the increase of these parameters beyond the values in (17) destroys the optimal estimate of the computational cost, as can be seen in the proof of Theorem 1.

Note that the values (17) are not the only possible ones; the algorithm still works with $\alpha < 1/3$. In particular, the case $\alpha = 0$ is proven in Theorem 3 below.

The asymptotical computational cost of the algorithm is characterized in the following theorem.

Theorem 1. *Let the parameters r_j are defined by (8), with either $\alpha = 0, c = r \le q, q = O(r)$ or α, c, q as defined by (17). Then, the computation cost of the algorithm is $O(r^\rho nN)$.*

First, let us obtain an estimate for the general case, without imposing (17). Define

$$\beta = 1 - \alpha(1+\rho) > 0. \tag{18}$$

Lemma 1. *Let the parameters r_j be defined by (8), (9), and $q \ge r$. Then, the total computational cost of the algorithm is as following.*

$$O\left(q^\rho (q+n)N\right), \qquad \text{if } \alpha = 0, \ c \le r, \tag{19}$$

$$O\left(\left(q^{1+\rho} + \frac{c^{2+\rho} q^{\alpha-\beta}}{\beta-\alpha} + \frac{c^2 q^{-(1-\rho-2\alpha)}}{1-2\alpha} + \left(q^\rho + \frac{c^{1+\rho} q^{-\beta}}{\beta}\right) n\right) N\right), \quad \text{otherwise}. \tag{20}$$

Proof. At Step **I**, computation of the matrix $\widetilde{\boldsymbol{X}}_{i,0}$ takes $O(nq^{1+\rho})$ operations if $r_0 \le q$ and $O(nr_0^{1+\rho})$ operations, otherwise. The latter case is possible only if $r_0 = \lfloor cq^{\alpha} \rfloor$. In this case, the estimate simplifies to the form

$$O\left(nc^{1+\rho}q^{\alpha(1+\rho)}\right) = O\left(nc^{1+\rho}q^{1-\beta}\right). \tag{21}$$

Computing and factorization of the matrix $\boldsymbol{C}_{i,0}$ bear computational costs of $O(nq^{1+\rho})$ and $O(q^{2+\rho})$, respectively, for any i. Given that there are $\frac{N}{q}$ submatrices at level 0, the total computational cost of Step **I** is

$$O((n+q)q^{\rho}N) \quad \text{if } \alpha = 0,\ c \le r \quad \text{and} \quad O\left((nq^{\rho} + q^{1+\rho} + nc^{1+\rho}q^{-\beta})N\right), \quad \text{otherwise}. \tag{22}$$

At Step **II**, the cost of forming of the matrix $\boldsymbol{P}_{i,j}$ and its factorization is $O(nr_{j-1}^{1+\rho} + r_{j-1}^{2+\rho})$ (see Demmel et al.[1]). The cost of computing $\widetilde{\boldsymbol{X}}_{i,j}$ is $O(nr_{j-1}r_j^{\rho})$. Note that $r_{j-1} \le r_j$ by (8). Thus, the total cost for one submatrix does not exceed

$$O\left(r_j^{1+\rho}(n + r_j)\right). \tag{23}$$

By definition of q, k_j,

$$r \le q = 2^{-j}k_j, \qquad j = 0, \ldots, K. \tag{24}$$

If $r_j = r$, then the total cost (23) for one submatrix can be bounded by $O(2^{-j}k_j r^{\rho}(n + r))$. Thus, the total cost for all $\frac{N}{k_j}$ submatrices of the level j is

$$O\left(2^{-j}r^{\rho}(n + r)N\right). \tag{25}$$

Now, let us consider the case

$$r_j = \lfloor ck_j^{\alpha} \rfloor = \lfloor c2^{j\alpha}q^{\alpha} \rfloor. \tag{26}$$

Substituting (18), (24) into (26) yields

$$\begin{aligned} r_j^{1+\rho} &\le c^{1+\rho}2^{j\alpha(1+\rho)}q^{\alpha(1+\rho)} = k_j 2^{-j(1-\alpha(1+\rho))}c^{1+\rho}q^{-(1-\alpha(1+\rho))} \\ &= k_j 2^{-j\beta}c^{1+\rho}q^{-\beta}, \\ r_j^{2+\rho} &\le k_j 2^{-j(1-\alpha(2+\rho))}c^{2+\rho}q^{-(1-\alpha(2+\rho))} = k_j 2^{j(\alpha-\beta)}c^{2+\rho}q^{\alpha-\beta}, \end{aligned}$$

and thus, the total cost (23) for one submatrix satisfies the estimate

$$O\left(k_j\left(2^{j(\alpha-\beta)}c^{2+\rho}q^{\alpha-\beta} + 2^{-j\beta}c^{1+\rho}q^{-\beta}n\right)\right).$$

Thus, the total cost for all $\frac{N}{k_j}$ submatrices of the level j is bounded by

$$O\left(\left(2^{j(\alpha-\beta)}c^{2+\rho}q^{\alpha-\beta} + 2^{-j\beta}c^{1+\rho}q^{-\beta}n\right)N\right). \tag{27}$$

Note that, in the case $c \le r, \alpha = 0$, the estimate (27) transforms into (25).

Step **III** has negligible computational cost. In Step **IV**, the matrices $\boldsymbol{U}_{i,j}$ have size $2r_{j-1} \times r_j$ for $j \ge 1$ and $q \times r$ for $j = 0$. The second dimension of the matrix $\boldsymbol{V}_j$ is equal to r for all j. Thus, for $j > 0$, the computation of the matrix $\boldsymbol{V}_j$ requires $\frac{N}{k_j} = 2^{-j}\frac{N}{q}$ matrix multiplications at the cost of $O(r_{j-1}r_j r^{\rho})$ each, for a total of

$$O\left(2^{-j}r_j^2 r^{\rho}N/q\right) = \begin{cases} O\left(2^{-j}r^{1+\rho}N\right), & \text{if } r_j = r, \\ O\left(2^{-j(1-2\alpha)}c^2q^{-(1-2\alpha)}r^{\rho}N\right), & \text{otherwise.} \end{cases} \tag{28}$$

For $j = 0$, it requires $\frac{N}{q}$ matrix multiplications at the cost of $O(r^{1+\rho}q)$ each, for a total of

$$O(r^{1+\rho}N). \tag{29}$$

Summing up the estimates (22), (25), (27)–(29) for all four steps and for all iterations, we obtain the following estimate for the following total cost of the algorithm:

$$O\left(N\left(q^{\rho}(n+q) + \sum_{j=1}^{K} 2^{-j}r^{\rho}(n+r)\right)\right), \qquad \text{if } \alpha = 0,\ c \le r, \tag{30}$$

$$O\left(N\sum_{j=0}^{K}\left(2^{-j}q^{1+\rho} + 2^{-j(1-2\alpha)}c^2q^{-(1-\rho-2\alpha)}r^{\rho} + 2^{-j(\beta-\alpha)}c^{2+\rho}q^{\alpha-\beta} + \left(2^{-j}q^{\rho} + 2^{-j\beta}c^{1+\rho}q^{-\beta}\right)n\right)\right), \qquad \text{otherwise}. \tag{31}$$

Note that $\alpha < (2+\rho)^{-1}$ implies

$$\beta - \alpha = 1 - \alpha(2+\rho) > 1 - (2+\rho)/(2+\rho) = 0\,,$$
$$(1-2\alpha) > 0\,;$$

therefore, the trivial bound

$$\sum_{j=1}^{K} 2^{-j\varepsilon} < \frac{2^{-\varepsilon}}{1-2^{-\varepsilon}} = \frac{1}{2^{\varepsilon}-1} < \frac{1}{\varepsilon \log 2}\,, \qquad \text{for any } \varepsilon > 0 \tag{32}$$

applies to $\varepsilon = \beta$, $\varepsilon = \beta - \alpha$ and $\varepsilon = 1 - 2\alpha$.

Computation of matrices $\boldsymbol{Q}, \boldsymbol{U}$ requires $O(r^{2+\rho})$ and $O(Nnr^{\rho})$ operations, respectively. Summing these estimates and the estimates (30), (31) for all levels $j = 0, \ldots, K$, then applying (32), we obtain the statement of Lemma 1. □

Now, the statement of Theorem 1 follows from Lemma 1.

Proof of Theorem 1. In the case when $\alpha = 0, c = r \le q, q = O(r)$, the statement of Theorem follows from (19). Let us consider the other case.

Applying definitions (18) and $c = r^{1-\alpha}$, we obtain for any α:

$$c^{2+\rho} r^{\alpha-\beta} = r^{(1-\alpha)(2+\rho)+\alpha-1+\alpha(1+\rho)} = r^{1+\rho}\,, \tag{33}$$

$$c^{2} r^{-(1-\rho-2\alpha)} = r^{2(1-\alpha)-(1-\rho-2\alpha)} = r^{1+\rho}\,, \tag{34}$$

$$c^{1+\rho} r^{-\beta} = r^{(1-\alpha)(1+\rho)-1+\alpha(1+\rho)} = r^{\rho}\,. \tag{35}$$

Setting $\alpha = \frac{1}{2+\rho}\left(1 - \frac{(1-\rho)}{3}\frac{r}{n}\right)$ implies

$$(\beta - a)^{-1} = (1 - \alpha(2+\rho))^{-1} = 3(1-\rho)^{-1} n/r\,, \tag{36}$$

$$(1-2\alpha)^{-1} < (1 - 2/(2+\rho))^{-1} = (2+\rho)/\rho\,, \tag{37}$$

$$\beta^{-1} = (1 - \alpha(1+\rho))^{-1} < (2+\rho)\,. \tag{38}$$

Substituting (33)–(38) into (20) and recalling that $q = c_q r, r \le n, \rho < 1$, we obtain the statement of the theorem. □

4 | ACCURACY

Definition 1. For any matrix $\boldsymbol{A}$ with singular values being $\sigma_i(\boldsymbol{A})$ sorted in descending order (if $\boldsymbol{A} = \boldsymbol{A}^T$, they are eigenvalues), we define $\tau_i(\boldsymbol{A})$ as

$$\tau_i(\boldsymbol{A}) = \left(\sum_{s=i}^{\operatorname{rank}(\boldsymbol{A})} \sigma_s^2(\boldsymbol{A}) \right)^{1/2}. \tag{39}$$

By basic properties of singular values (see the work of Stewart[13]), for any matrix $\boldsymbol{A}$, the following relationships hold:

$$\sigma_{i+1}(\boldsymbol{A}) = \min_{\boldsymbol{B}:\ \operatorname{rank}\boldsymbol{B} \le i} \|\boldsymbol{A} - \boldsymbol{B}\|_2\,, \qquad i = 0, \ldots, \operatorname{rank}(\boldsymbol{A})\,, \tag{40}$$

$$\tau_{i+1}(\boldsymbol{A}) = \min_{\boldsymbol{B}:\ \operatorname{rank}\boldsymbol{B} \le i} \|\boldsymbol{A} - \boldsymbol{B}\|_F\,, \qquad i = 0, \ldots, \operatorname{rank}(\boldsymbol{A})\,, \tag{41}$$

$$\sigma_i(\boldsymbol{A}) \le \tau_i(\boldsymbol{A}) \ \le\ \sqrt{\operatorname{rank}(\boldsymbol{A}) - (i-1)}\,\sigma_i(\boldsymbol{A})\,, \qquad i = 1, \ldots, \operatorname{rank}(\boldsymbol{A})\,. \tag{42}$$

In particular, for $i = 1$, we obtain from (40)–(42)

$$\|\boldsymbol{A}\|_2 = \sigma_1(\boldsymbol{A}) \ \le\ \tau_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_F \ \le\ \sqrt{\operatorname{rank}(\boldsymbol{A})}\,\|\boldsymbol{A}\|_2\,. \tag{43}$$

In view of (40), (41), it is reasonable to estimate the accuracy of r-truncated SVD in terms of $\sigma_{r+1}(\boldsymbol{X})$, $\tau_{r+1}(\boldsymbol{X})$. Thus, we have the following result.

Theorem 2. *Let the parameters r_j and q be defined by (8) and (17), and let the matrices $\boldsymbol{Q}$ and $\boldsymbol{U}$ be defined by (16). Then, (3) and (5) hold, and*

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}, \qquad \boldsymbol{Q} = \boldsymbol{Q}^T > \boldsymbol{0}\,. \tag{44}$$

$$\textit{If } \boldsymbol{X} = \boldsymbol{X}^T, \quad \textit{then } \boldsymbol{U} = \boldsymbol{V}\,. \tag{45}$$

$$\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_F = \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_F \;\le\; \tau_{r+1}(\boldsymbol{X}) + \sum_{j=0}^{J-1}\tau_{r_j+1}(\boldsymbol{X})\,, \tag{46}$$

where

$$J = \left\lceil \alpha^{-1}(\log_2 n - \log_2 r - \log_2 c_0 - \alpha\log_2 c_q)\right\rceil\,. \tag{47}$$

Before presenting the proof of Theorem 2, let us formulate several important corollaries. First, applying (42), (43), we immediately obtain an estimate for L_2 norm.

Corollary 1. *Under conditions of Theorem 2,*

$$\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_2 = \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_2 \;\le\; \sqrt{n-r}\,\sigma_{r+1}(\boldsymbol{X}) + \sum_{j=0}^{J-1}\sqrt{n-r_j}\,\sigma_{r_j+1}(\boldsymbol{X})\,. \tag{48}$$

Because $\alpha \ge \frac{1}{3},\ c_0, c_q \ge 1$, the estimates (46), (48) can be simplified as follows.

Corollary 2. *Under conditions of Theorem 2,*

$$\begin{aligned}
\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_F &= \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_F \;\le\; 3(\log_2 n - \log_2 r + 1)\,\tau_{r+1}(\boldsymbol{X})\,,\\
\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_2 &= \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_2 \;\le\; 3(\log_2 n - \log_2 r + 1)\,\sqrt{n-r}\,\sigma_{r+1}(\boldsymbol{X})\,.
\end{aligned}$$

For comparison, we estimate also the approximation error in the trivial case $r_j \equiv r$.

Theorem 3. *Let the parameters r_j be chosen to be equal to r for all $j = 0, \dots, K$, and the matrices $\boldsymbol{Q}, \boldsymbol{U}$ be defined by (16). Then, (3), (5), (44), (45) hold, and*

$$\begin{aligned}
\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_F &= \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_F \;\le (n-r)\,\tau_{r+1}\,,\\
\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T\|_2 &= \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_2 \;\le (n-r)^{3/2}\,\sigma_{r+1}\,.
\end{aligned}$$

Now, let us proceed to the proof of Theorem 2. However, first, we need to introduce some notation and establish a few simple facts. For convenience, we define $r_{-1} = q/2$. Let us define the following matrices:

$$\widetilde{\boldsymbol{U}}_j = \operatorname{diag}\left(\boldsymbol{U}_{1,j}, \dots, \boldsymbol{U}_{2^{K-j},j}\right) \in \mathbb{R}^{2^{K-j+1}r_{j-1}\times 2^{K-j}r_j}, \qquad j = 0, \dots, K, \tag{49}$$

$$\widetilde{\boldsymbol{V}}_{-1} = \boldsymbol{I}; \qquad \widetilde{\boldsymbol{V}}_j = \widetilde{\boldsymbol{U}}_0\widetilde{\boldsymbol{U}}_1 \dots \widetilde{\boldsymbol{U}}_j \in \mathbb{R}^{N\times 2^{K-j}r_j}, \qquad j = 0, \dots, K. \tag{50}$$

Let us define matrices $\widetilde{\boldsymbol{V}}_{i,j} \in \mathbb{R}^{k_j\times r_j}, \boldsymbol{W}_{i,j} \in \mathbb{R}^{k_j\times 2r_{j-1}}$ by the following recurrence rule:

$$\widetilde{\boldsymbol{V}}_{i,0} = \boldsymbol{U}_{i,0}, \qquad i = 1, \dots, 2^K, \tag{51}$$

$$\boldsymbol{W}_{i,j} = \begin{bmatrix} \widetilde{\boldsymbol{V}}_{2i-1,j-1} & \boldsymbol{0} \\ \boldsymbol{0} & \widetilde{\boldsymbol{V}}_{2i,j-1} \end{bmatrix}, \qquad i = 1, \dots, 2^{K-j};\ j = 1, \dots, K, \tag{52}$$

$$\widetilde{\boldsymbol{V}}_{i,j} = \boldsymbol{W}_{i,j}\boldsymbol{U}_{i,j}, \qquad i = 1, \dots, 2^{K-j};\ j = 1, \dots, K. \tag{53}$$

It follows immediately from (49)–(53) that

$$\widetilde{\boldsymbol{V}}_j = \operatorname{diag}\left(\widetilde{\boldsymbol{V}}_{1,j}, \dots, \widetilde{\boldsymbol{V}}_{2^{K-j},j}\right), \qquad j = 0, \dots, K. \tag{54}$$

In addition, let us denote $\boldsymbol{P}_{i,0} = \boldsymbol{C}_{i,0}, i = 1, \dots, 2^K$ (so far, $\boldsymbol{P}_{i,j}$ were defined only for $j > 0$). Then, we have the following result.

Lemma 2.

$$\boldsymbol{P}_{i,j} = \boldsymbol{W}_{i,j}^T \left(\boldsymbol{X}_{i,j}^T \boldsymbol{X}_{i,j} \right) \boldsymbol{W}_{i,j}, \qquad i = 1, \dots, 2^{K-j};\ j = 1, \dots, K, \tag{55}$$

$$\boldsymbol{D}_{i,j} = \widetilde{\boldsymbol{V}}_{i,j}^T \boldsymbol{X}_{i,j}^T \boldsymbol{X}_{i,j} \widetilde{\boldsymbol{V}}_{i,j}, \qquad i = 1, \dots, 2^{K-j};\ j = 0, \dots, K, \tag{56}$$

$$\widetilde{\boldsymbol{X}}_{i,j} = \boldsymbol{X}_{i,j} \widetilde{\boldsymbol{V}}_{i,j}, \qquad i = 1, \dots, 2^{K-j};\ j = 0, \dots, K, \tag{57}$$

$$\boldsymbol{W}_{i,j}^T \boldsymbol{W}_{i,j} = \boldsymbol{I}, \qquad i = 1, \dots, 2^{K-j};\ j = 1, \dots, K, \tag{58}$$

$$\widetilde{\boldsymbol{V}}_{i,j}^T \widetilde{\boldsymbol{V}}_{i,j} = \boldsymbol{I}, \qquad i = 1, \dots, 2^{K-j};\ j = 0, \dots, K. \tag{59}$$

Proof. The statements (55)–(59) will be proven by induction w.r.t. level j. For $j = 0$, the statements (55), (58) are trivial, (57) is identical to the definition (10), and (56), (59) follow from the property (6) of exact factorization performed at Step **I**. Let (55)–(59) be true for $j-1$. Then, (57) at level j holds due to (14), (52), (53) as follows:

$$\begin{aligned} \widetilde{\boldsymbol{X}}_{i,j} &= \left[\widetilde{\boldsymbol{X}}_{2i-1,j-1}\ \widetilde{\boldsymbol{X}}_{2i,j-1} \right] \boldsymbol{U}_{i,j} = \left[\boldsymbol{X}_{2i-1,j-1}\ \boldsymbol{X}_{2i,j-1} \right] \boldsymbol{W}_{i,j} \boldsymbol{U}_{i,j} \\ &= \boldsymbol{X}_{i,j} \widetilde{\boldsymbol{V}}_{i,j}, \qquad i = 1, \dots, 2^{K-j};\ j = 1, \dots, K. \end{aligned}$$

Expressing $\boldsymbol{D}_{s,j-1}, \widetilde{\boldsymbol{X}}_{s,j-1}$ via (56), (57), respectively, and substituting the result into (12), we conclude that for any $i = 1, \dots, 2^{K-j}; j = 1, \dots, K$:

$$\begin{aligned} \boldsymbol{P}_{i,j} &= \begin{bmatrix} \boldsymbol{D}_{2i-1,j-1} & \widetilde{\boldsymbol{X}}_{2i-1,j-1}^T \widetilde{\boldsymbol{X}}_{2i,j-1} \\ \widetilde{\boldsymbol{X}}_{2i,j-1}^T \widetilde{\boldsymbol{X}}_{2i-1,j-1} & \boldsymbol{D}_{2i,j-1} \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{\boldsymbol{V}}_{2i-1,j-1}^T \boldsymbol{X}_{2i-1,j-1}^T \boldsymbol{X}_{2i-1,j-1} \widetilde{\boldsymbol{V}}_{2i-1,j-1} & \widetilde{\boldsymbol{V}}_{2i-1,j-1}^T \boldsymbol{X}_{2i-1,j-1}^T \boldsymbol{X}_{2i,j-1} \widetilde{\boldsymbol{V}}_{2i,j-1} \\ \widetilde{\boldsymbol{V}}_{2i,j-1}^T \boldsymbol{X}_{2i,j-1}^T \boldsymbol{X}_{2i-1,j-1} \widetilde{\boldsymbol{V}}_{2i-1,j-1} & \widetilde{\boldsymbol{V}}_{2i,j-1}^T \boldsymbol{X}_{2i,j-1}^T \boldsymbol{X}_{2i,j-1} \widetilde{\boldsymbol{V}}_{2i,j-1} \end{bmatrix} \\ &= \boldsymbol{W}_{i,j}^T \begin{bmatrix} \boldsymbol{X}_{2i-1,j-1}^T \\ \boldsymbol{X}_{2i,j-1}^T \end{bmatrix} \left[\boldsymbol{X}_{2i-1,j-1}\ \boldsymbol{X}_{2i,j-1} \right] \boldsymbol{W}_{i,j} \\ &= \boldsymbol{W}_{i,j}^T \left(\boldsymbol{X}_{i,j}^T \boldsymbol{X}_{i,j} \right) \boldsymbol{W}_{i,j}\ , \end{aligned}$$

which proves (55). The statement (56) now follows from (52), (53), (55) and property (6) of an exact truncated factorization at Step **II**. The statements (58), (59) are trivially verified as follows:

$$\begin{aligned} \boldsymbol{W}_{i,j}^T \boldsymbol{W}_{i,j} &= \begin{bmatrix} \widetilde{\boldsymbol{V}}_{2i-1,j-1}^T \widetilde{\boldsymbol{V}}_{2i-1,j-1} & \boldsymbol{0} \\ \boldsymbol{0} & \widetilde{\boldsymbol{V}}_{2i,j-1}^T \widetilde{\boldsymbol{V}}_{2i,j-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \boldsymbol{I} \\ \widetilde{\boldsymbol{V}}_{i,j}^T \widetilde{\boldsymbol{V}}_{i,j} &= \boldsymbol{U}_{i,j}^T \boldsymbol{W}_{i,j}^T \boldsymbol{W}_{i,j} \boldsymbol{U}_{i,j} = \boldsymbol{U}_{i,j}^T \boldsymbol{U}_{i,j} = \boldsymbol{I}. \end{aligned}$$

□

Lemma 3.

$$\left\| \boldsymbol{X} \widetilde{\boldsymbol{V}}_{j-1} \widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X} \widetilde{\boldsymbol{V}}_j \widetilde{\boldsymbol{V}}_j^T \right\|_F \le \left(\sum_{i=1}^{2^{K-j}} \sum_{s=r_j+1}^{2r_{j-1}} \sigma_s(\boldsymbol{P}_{i,j}) \right)^{1/2} \le \tau_{r_j+1}(\boldsymbol{X})\,, \qquad \textit{for any } j = 0, \dots, K. \tag{60}$$

Proof. Let us fix the level $j = 0, \dots, K$. For any number $i = 1, \dots, 2^{K-j}$, let us consider the following eigenvalue decomposition of $\boldsymbol{P}_{i,j}$:

$$\boldsymbol{P}_{i,j} = \sum_{s=1}^{2r_{j-1}} \sigma_s(\boldsymbol{P}_{i,j})\ \boldsymbol{a}_i \boldsymbol{a}_i^T\ ,$$

where $\boldsymbol{a}_i$ is the eigenvector of $\boldsymbol{P}_{i,j}$ corresponding to i^{th} eigenvalue. By construction, the matrix $\boldsymbol{U}_{i,j}$ consists of first r_j eigenvectors of $\boldsymbol{P}_{i,j}$. Therefore, the matrix $(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T)$ is the projector on the linear span of the vectors $\boldsymbol{a}_{r_j+1},\ \dots,\boldsymbol{a}_{2r_{j-1}}$. It follows that

$$\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{P}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)=\sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j})\,\boldsymbol{a}_i\boldsymbol{a}_i^T\,;$$

hence,

$$\operatorname{Tr}\left(\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{P}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right)=\sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j}),\qquad\text{for any } i=1,\ \dots,2^{K-j}.$$

With (53) and Lemma 2, it follows that

$$\begin{aligned}
\left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T-\boldsymbol{X}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}^T\right\|_F^2&=\left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\right\|_F^2\\
&=\operatorname{Tr}\left(\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\right)\\
&=\operatorname{Tr}\left(\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\right)\\
&=\left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right\|_F^2\\
&=\operatorname{Tr}\left(\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right)\\
&=\operatorname{Tr}\left(\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{P}_{i,j}\left(\boldsymbol{I}-\boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right)\\
&=\sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j}),\qquad\text{for any } i=1,\ \dots,2^{K-j}.
\end{aligned}\tag{61}$$

Let us construct the best approximation of $\boldsymbol{X}$ by a matrix of size $n\times N$ and rank r_j and denote it by $\boldsymbol{Z}$:

$$\operatorname{rank}\boldsymbol{Z}=r_j,\qquad\|\boldsymbol{X}-\boldsymbol{Z}\|_F=\tau_{r_j+1}(\boldsymbol{X})\,.\tag{62}$$

Furthermore, let us split the matrix $\boldsymbol{Z}$ onto 2^{K-j} submatrices of the size $n\times k_j$ in the same way as the matrix $\boldsymbol{X}$ was split on submatrices $\boldsymbol{X}_j$:

$$\boldsymbol{Z}=\left[\,\boldsymbol{Z}_1\ \cdots\ \boldsymbol{Z}_{2^{K-j}}\,\right].\tag{63}$$

By construction,

$$\operatorname{rank}\boldsymbol{Z}_i\le\operatorname{rank}\boldsymbol{Z}=r_j\qquad\text{for any } i=1,\ \dots,2^{K-j}.$$

Hence, by (41),

$$\|\boldsymbol{X}_{i,j}-\boldsymbol{Z}_i\|_F\ge\tau_{r_j+1}(\boldsymbol{X}_{i,j})\qquad\text{for any } i=1,\ \dots,2^{K-j}.\tag{64}$$

Using (55) and orthogonality of the matrices $\boldsymbol{W}_{i,j}$, we can write

$$\begin{gathered}
\sigma_s^2(\boldsymbol{X}_{i,j})\ \ge\ \sigma_s^2(\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j})=\sigma_s\left(\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\right)=\ \sigma_s(\boldsymbol{P}_{i,j})\,,\\
\text{for any } s=1,\ \dots,2r_{j-1},\ i=1,\ \dots,2^{K-j},
\end{gathered}\tag{65}$$

which implies that

$$\tau_{t_j+1}^2(\boldsymbol{X}_{i,j})=\sum_{s=r_j+1}^{\min(n,k_j)}\sigma_s^2(\boldsymbol{X}_{i,j})\ \ge\ \sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j})\qquad\text{for any } i=1,\ \dots,2^{K-j}.\tag{66}$$

Stacking matrices $(\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T - \boldsymbol{X}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}^T)$ in a row yields the matrix $(\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T)$. Inequality (61) implies that

$$\left\|\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T\right\|_F^2 = \sum_{i=1}^{2^{K-j}}\left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T - \boldsymbol{X}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}\widetilde{\boldsymbol{V}}_{i,j}^T\right\|_F^2 \leq \sum_{i=1}^{2^{K-j}}\sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j})\,. \tag{67}$$

On the other hand, combining the statements (63), (64), (66) yields

$$\sum_{i=1}^{2^{K-j}}\sum_{s=r_j+1}^{2r_{j-1}}\sigma_s(\boldsymbol{P}_{i,j}) \leq \sum_{i=1}^{2^{K-j}}\tau_{r_j+1}^2(\boldsymbol{X}_{i,j}) \leq \sum_{i=1}^{2^{K-j}}\|\boldsymbol{X}_{i,j} - \boldsymbol{Z}_i\|_F^2 = \|\boldsymbol{X} - \boldsymbol{Z}\|_F^2 \quad = \tau_{r_j+1}^2(\boldsymbol{X})\,. \tag{68}$$

Together, (67) and (68) yield the statement of the lemma. □

Corollary 3. *If $r_j = n$ for some level j, then*

$$\left\|\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T\right\|_2 = \left\|\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T\right\|_F = 0\,.$$

Lemma 4.

$$\operatorname{rank}\boldsymbol{D} = r\,. \tag{69}$$

Proof. First, let us fix the number $j = 1, \ldots, K$ and prove that

$$\operatorname{rank}\boldsymbol{X}\widetilde{\boldsymbol{V}}_j \geq \min\left(r_j, \operatorname{rank}\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\right)\,. \tag{70}$$

Repeating the steps at the beginning of the proof of Lemma 3, we easily obtain

$$\left\|\boldsymbol{P}_{i,j}\left(\boldsymbol{I} - \boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right\|_F^2 = \tau_{r_{j+1}}^2(\boldsymbol{P}_{i,j}) \qquad \text{for any } i = 1, \ldots, 2^{K-j};$$

hence, by (55):

$$\begin{aligned}\left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T - \boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j}\boldsymbol{V}_{i,j}^T\boldsymbol{X}_{i,j}^T\right\|_F &= \left\|\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I} - \boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\right\|_F \\ &= \left\|\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\left(\boldsymbol{I} - \boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right\|_F = \left\|\boldsymbol{P}_{i,j}\left(\boldsymbol{I} - \boldsymbol{U}_{i,j}\boldsymbol{U}_{i,j}^T\right)\right\|_F \\ &= \tau_{r_{j+1}}(\boldsymbol{P}_{i,j})\,, \qquad \text{for any } i = 1, \ldots, 2^{K-j}.\end{aligned} \tag{71}$$

Let us denote

$$\boldsymbol{B}_{ij} = \boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T, \qquad i = 1, \ldots, 2^{K-j}$$

and consider the following alternative cases.

(1) There exists at least one i such that $\boldsymbol{B}_{ij} \geq r_j$. Hence, $\sigma_{r_j}(B_{i,j}) \neq 0$. By (55), the matrix $\boldsymbol{B}_{ij}$ has the same nonzero eigenvalues as $\boldsymbol{P}_{ij} = (\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j})^T(\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j})$. The minimal Frobenius distance from $\boldsymbol{B}_{ij}$ to a matrix of rank $r_j - 1$ is equal to $\tau_{r_j}(\boldsymbol{B}_{i,j})$. Because

$$\tau_{r_j}(\boldsymbol{B}_{i,j}) = \sqrt{\sigma_{r_j}^2(\boldsymbol{B}_{i,j}) + \tau_{r_{j+1}}^2(\boldsymbol{B}_{i,j})} > \tau_{r_{j+1}}(\boldsymbol{B}_{i,j}) = \tau_{r_{j+1}}(\boldsymbol{P}_{i,j}),$$

this distance exceeds the distance in (71). Hence, $\operatorname{rank}\boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j}\boldsymbol{V}_{i,j}^T\boldsymbol{X}_{i,j}^T \geq r_j$; therefore,

$$\operatorname{rank}\boldsymbol{X}\widetilde{\boldsymbol{V}}_j \geq \operatorname{rank}\boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j} = \operatorname{rank}\boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j}\boldsymbol{V}_{i,j}^T\boldsymbol{X}_{i,j}^T \geq r_j\,.$$

Thus, we had proven that in this case (70) holds.

(2) $\operatorname{rank} \boldsymbol{B}_{ij} < r_j$ for all $i = 1, \dots, 2^{K-j}$. Then, $\tau_{r_{j+1}}(\boldsymbol{P}_{i,j}) = \tau_{r_{j+1}}(\boldsymbol{B}_{i,j}) = 0$, and by (71), we obtain

$$\boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T = \boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j}\boldsymbol{V}_{i,j}^T\boldsymbol{X}_{i,j}^T, \qquad \text{for any } i = 1, \dots, 2^{K-j}.$$

Hence,

$$\begin{aligned}\operatorname{rank} \boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1} &= \operatorname{rank} \boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T\boldsymbol{X}^T = \operatorname{rank} \sum_{i=1}^{2^{K-j}} \boldsymbol{X}_{i,j}\boldsymbol{W}_{i,j}\boldsymbol{W}_{i,j}^T\boldsymbol{X}_{i,j}^T \\ &= \operatorname{rank} \sum_{i=1}^{2^{K-j}} \boldsymbol{X}_{i,j}\boldsymbol{V}_{i,j}\boldsymbol{V}_{i,j}^T\boldsymbol{X}_{i,j}^T = \operatorname{rank} \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T\boldsymbol{X}^T = \operatorname{rank} \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\,,\end{aligned}$$

which implies (70).

Thus, in either case (70) holds. Applying induction w.r.t. j to (70) and recalling that $r \le \operatorname{rank}\boldsymbol{X}, r \le r_0, \dots, r_K$, we obtain

$$\operatorname{rank} \boldsymbol{X}\widetilde{\boldsymbol{V}}_K \ge r\,. \tag{72}$$

Combining (72) with (56), (57) and recalling that $\boldsymbol{D} = \boldsymbol{D}_{1,K}$ has dimensions $r \times r$, we obtain the claim (69). □

Now, we are ready to prove our main theorem.

Proof of Theorem 2. It follows from the statement (56) of Lemma 2 that

$$\boldsymbol{D} = \boldsymbol{D}_{1,K} = \widetilde{\boldsymbol{V}}_{1,K}^T\boldsymbol{X}_{1,K}^T\boldsymbol{X}_{1,K}\widetilde{\boldsymbol{V}}_{1,K}\,. \tag{73}$$

By definitions (7), (15), (49), (50) and statements (54), (57), we obtain

$$\widetilde{\boldsymbol{V}}_{1,K} = \boldsymbol{V}, \qquad \boldsymbol{X}_{1,K} = \boldsymbol{X}, \qquad \widetilde{\boldsymbol{X}}_{1,K} = \boldsymbol{X}\boldsymbol{V}\,. \tag{74}$$

Together, (73), (74), (16), and (69) yield (5) and (3).

The matrix $\boldsymbol{Q} = \boldsymbol{D}^{1/2}$ is symmetric and nonnegative definite. Hence, by Lemma 4, it is positive definite. Orthogonality of $\boldsymbol{U}$ is implied by (3), (5) as follows:

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{Q}^{-1}(\boldsymbol{V}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{V})\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{-1}(\boldsymbol{Q}^2)\boldsymbol{Q}^{-1} = \boldsymbol{I}\,.$$

That proves statement (44).

If $\boldsymbol{X} = \boldsymbol{X}^T$, then the matrix $\boldsymbol{V}^T\boldsymbol{X}\boldsymbol{V}$ is symmetric. Then, by (3)

$$(\boldsymbol{V}^T\boldsymbol{X}\boldsymbol{V})^2 = \boldsymbol{V}^T\boldsymbol{X}\boldsymbol{X}\boldsymbol{V} = \boldsymbol{Q}^2\,.$$

Because both matrices are symmetric, it follows that $\boldsymbol{Q} = \boldsymbol{V}^T\boldsymbol{X}\boldsymbol{V}$. Therefore,

$$\boldsymbol{U}^T\boldsymbol{V} = \boldsymbol{Q}^{-1}\boldsymbol{V}^T\boldsymbol{X}\boldsymbol{V} = \boldsymbol{I}\,.$$

With orthogonality of $\boldsymbol{V}$ and $\boldsymbol{U}$, it yields

$$(\boldsymbol{V}-\boldsymbol{U})^T(\boldsymbol{V}-\boldsymbol{U}) = \boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{U}^T\boldsymbol{U} - \boldsymbol{V}^T\boldsymbol{U} - \boldsymbol{U}^T\boldsymbol{V} = \boldsymbol{I} + \boldsymbol{I} - \boldsymbol{I} - \boldsymbol{I} = \boldsymbol{0}\,.$$

Statement (45) follows.

It follows immediately from definition (5) that

$$\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{V}^T = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{Q}^{-1}\boldsymbol{Q}\boldsymbol{V}^T = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\,. \tag{75}$$

Applying Lemma 3 yields

$$\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_F \le \sum_{j=0}^{K} \left\|\boldsymbol{X}\widetilde{\boldsymbol{V}}_{j-1}\widetilde{\boldsymbol{V}}_{j-1}^T - \boldsymbol{X}\widetilde{\boldsymbol{V}}_j\widetilde{\boldsymbol{V}}_j^T\right\|_F \le \sum_{j=0}^{K} \tau_{r_j+1}(\boldsymbol{X})\,. \tag{76}$$

Recall the value of the parameter J in (47). By (8), (17),

$$\begin{aligned} r_j &\ge \min\left(n, \lfloor c2^{j\alpha}q^\alpha \rfloor\right) \ge \min\left(n, \lfloor c_0 r^{1-\alpha} 2^{J\alpha} c_q^\alpha r^\alpha \rfloor\right) \\ &\ge \min\left(n, \left\lfloor c_0 c_q^\alpha r \frac{n}{r c_0 c_q^\alpha} \right\rfloor\right) = n \qquad \text{for any } j = J, \dots, K-1.\end{aligned}$$

Hence, by Corollary 3,

$$\sum_{j=J}^{K-1} \tau_{r_j+1}(\boldsymbol{X}) = 0\,. \tag{77}$$

Combining (75)–(77) and recalling that $r_k = r$ yields the statement (46) of Theorem 2. □

Proof of Theorem 3 repeats all the steps of the proof above in simplified form.

5 | NUMERICAL SIMULATION

In this section, we summarize the numerical performance of our approach and compare it with other approaches in literature. For numerical tests of our method, we use the following matrices:

(A) **A practical example:** the matrix *sarcos_inv* from Rasmussen et al.,[14] slightly cut for programming convenience from the size $28 \times 44,484$ to the size $28 \times 40,960$.

(B) **Uniformly random matrix:** the matrices of the size $100 \times 10,240$ with each entry being a random number independently and uniformly generated in the interval $[0, 1]$. Similar experiments were done using the standard normal distribution but because the results for both distributions were almost identical, we present results for the uniform distribution only.

(C) **Random matrix with exponentially decaying singular values and less than full rank**: the matrices of the size $100 \times 10,240$ and the structure

$$\boldsymbol{X} = \boldsymbol{UHV}, \qquad \boldsymbol{H} = \left[\,\mathrm{diag}(\boldsymbol{h})\ \boldsymbol{O}\,\right], \tag{78}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are random orthogonal matrices of the size 100 and 10,240, respectively, $\boldsymbol{O}$ is a matrix of zeros of size $100 \times 10,140$, and

$$\boldsymbol{h} = \mathrm{col}(1, e^{-0.1}, e^{-0.2}, \ldots, e^{-6}, 0, \ldots, 0) \in \mathbb{R}^{100}.$$

(D) **Random matrix with slowly decreasing singular values and less than full rank**: the matrices of the size $100 \times 10,240$ and the structure (78) with the same $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{O}$ as in the case (3) but with different $\boldsymbol{h}$:

$$\boldsymbol{h} = \mathrm{col}(2, 1 + e^{-0.1}, 1 + e^{-0.2}, \ldots, 1 + e^{-6}, 0, \ldots, 0) \in \mathbb{R}^{100}.$$

All experiments with random matrices are averaged over 30 trials.

The approximation error was captured in the following three quantities:

Relative L_2 approximation error	ε_1	$= \frac{\|\boldsymbol{X}-\boldsymbol{XVV}^T\|_2}{\sigma_{r+1}(\boldsymbol{X})} - 1$
Relative Frobenius approximation error	ε_2	$= \frac{\|\boldsymbol{X}-\boldsymbol{XVV}^T\|_F}{\tau_{r+1}(\boldsymbol{X})} - 1$
Maximum relative error for singular values	ε_3	$= \max_{i=1,\ldots,r} \left\lvert \frac{\sigma_i(\boldsymbol{Q})}{\sigma_i(\boldsymbol{X})} - 1 \right\rvert$

where matrices $\boldsymbol{V}$ and $\boldsymbol{Q}$ are the results of factorization (1).

Parameter q is chosen as

$$q = \min\left(q_0, 5 \cdot 2^{\lceil \log_2(4r/3) \rceil}\right), \qquad q_0 = \begin{cases} 20, & \text{for the matrix (A)}, \\ 80, & \text{otherwise.} \end{cases}$$

The parameter q_0 is introduced to avoid q growing to the value n, which leads to the algorithm performing full SVD factorization rather than partial one. The value q_0 is by the following criterion: the gap $(n - q_0)$ must be big enough to make a difference between full and partial SVD but small enough to not interfere with nominal setting (17) for majority of the values of r. This is how q_0 was chosen equal to 20 for $n = 28$ and equal to 80 for $n = 100$.

Note that, when r grows above q_0, the definition of q violates the condition $q \geq r$ in (17). Thus, for high values of r, the proposed method is being used outside the parameter regimes analyzed in this paper. However, we observe that no significant changes in the algorithm performance were observed.

All experiments were performed for the values of $r = 1, \ldots, n$. For the matrices (C), (D), as r grew above 61, it violated the condition $r \leq \mathrm{rank}X$. Nevertheless, in all these cases, the algorithm demonstrated the accuracy in the order of MATLAB numerical error (10^{-10}).

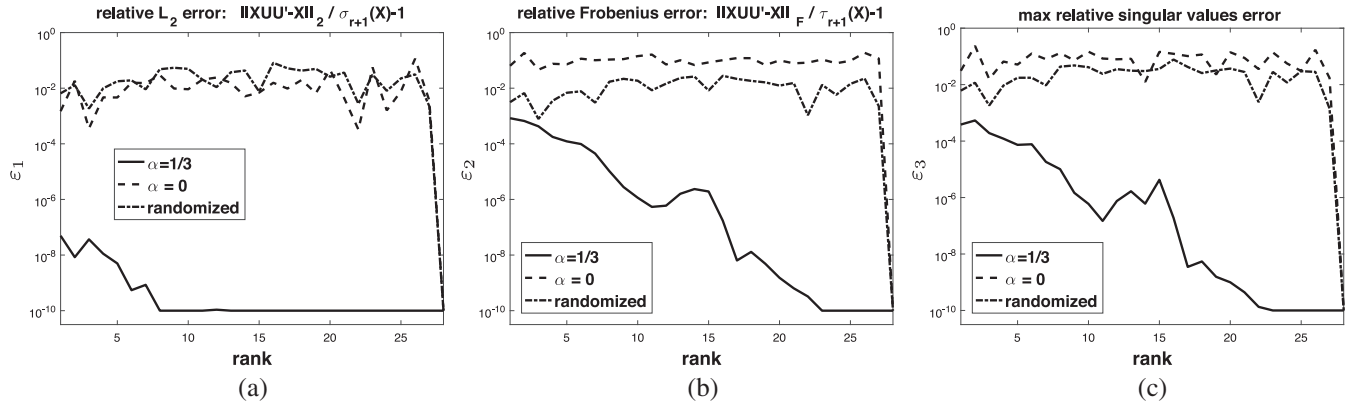

FIGURE 3 Comparison of accuracy of singular value decomposition (SVD) algorithms for the matrix (A)

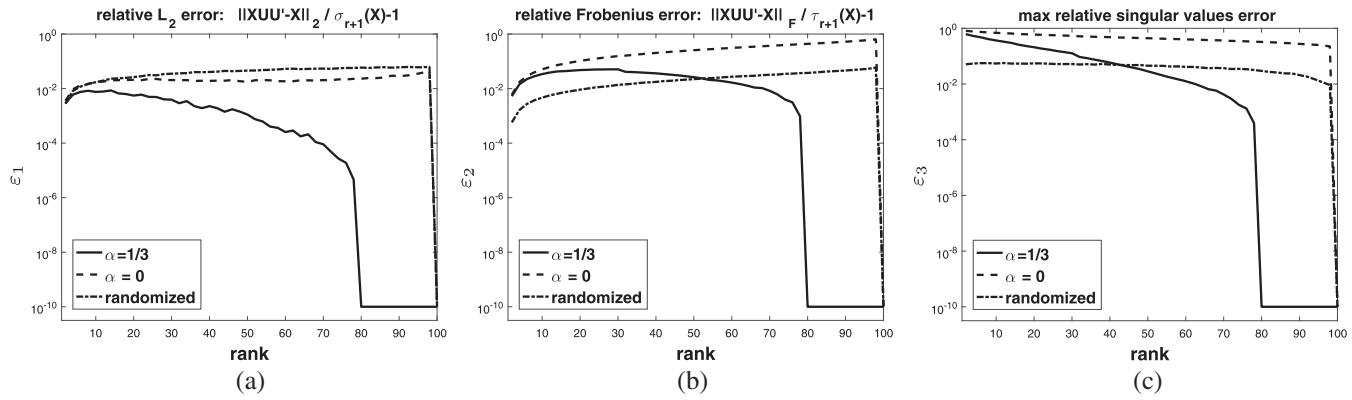

FIGURE 4 Comparison of accuracy of singular value decomposition (SVD) algorithms for the matrix (B)

The following algorithms were compared:
$\alpha = \mathbf{0}$: The algorithm presented in this paper with settings $\alpha = 0, c = r, r_j \equiv r$;
$\alpha = \mathbf{1/3}$: The algorithm presented in this paper with the settings

$$\alpha = \frac{1}{3}, \quad c = r^{1-\alpha}, \quad r_j = \min\left(n, \max\left(r, \lfloor ck_j^{\alpha} \rfloor\right)\right) .$$

We use $\alpha = 1/3$ instead of the exact value defined in (17) for simplicity because the difference between $1/3$ and the value of α in (17) appears negligible.

Randomized algorithm: The method[6] that has the best accuracy and complexity estimates among randomization-based algorithms. The number of power iterations is set to 4.

The results of numerical experiments for the matrices (A)–(D) are presented on Figures 3–6, respectively. Approximation errors are plotted as a function of the rank r on logarithmic scale. If any error is less than 10^{-10} (machine accuracy in our case), it is set to that value. If $\text{rank}(\boldsymbol{X}) < n$ (matrices (C) and (D)), errors are shown only for $r \leq \text{rank}(\boldsymbol{X})$ because it is impossible to compute relative error when $\sigma_i(\boldsymbol{X}) = \tau_i(\boldsymbol{X}) = 0$. Note that, in all such cases, approximation error of our method for $r > \text{rank}(\boldsymbol{X})$ was of the order of machine accuracy.

The results of numerical experiments lead us to the following observations:

- If singular values of the matrix decay exponentially (the matrices (A) and (C)), the method with $\alpha = \mathbf{1/3}$ has a clear advantage. All approximation errors decrease exponentially as the rank r increases. Even for small r, the L_2 error ε_1 is of the order 10^{-6} and other two errors — of the order 10^{-3}, which is two orders of magnitude better than accuracy of other two methods.
- If singular values of the matrix decrease slowly or stay almost constant (the matrices (B) and (D)), the method with $\alpha = \mathbf{1/3}$ produces smaller L_2 error ε_1 than the two other methods. As r grows, it also produces the smallest Frobenius and singular value errors $\varepsilon_2, \varepsilon_3$. However, for smaller r, these errors are not as small as the ones produced by the Randomized algorithm.[6]

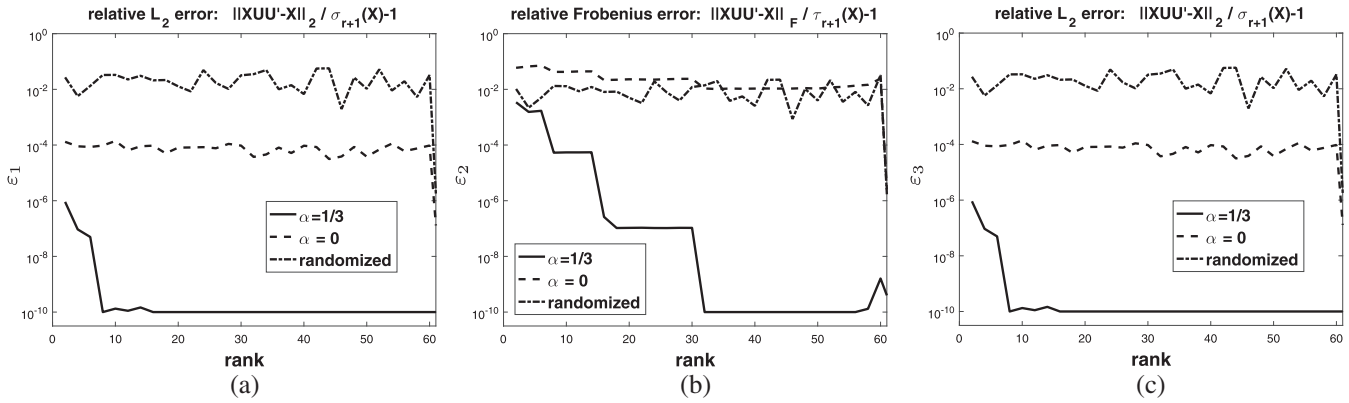

FIGURE 5 Comparison of accuracy of singular value decomposition (SVD) algorithms for the matrix (C)

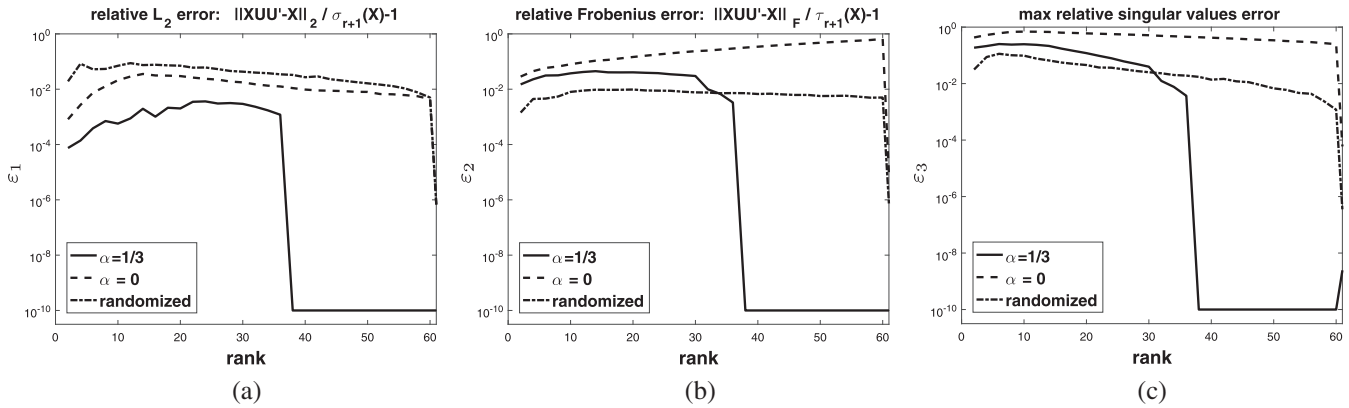

FIGURE 6 Comparison of accuracy of singular value decomposition (SVD) algorithms for the matrix (D)

- Well, in advance of r approaching rank($\boldsymbol{X}$), the factorization obtained by the method with $\alpha = \mathbf{1/3}$ becomes exact. The other two competing methods do not have this feature.
- Upward trend on the bottom right of the maximum singular value error chart for the matrix (C) (Figure 5) can be easily explained by the fact that $\sigma_i(X)$ is vanishing exponentially as i approaches 61. Thus, the observed relative error 10^{-6} corresponds to the absolute error 10^{-12}.
- Switching to a higher value of the parameter q is easily observable on charts; implying that fine-tuning of this parameter may have significant impact on the method performance.

An average runtime performance of all three methods on the matrices (B) is shown on Figure 7. For other matrices, it is very similar. One can observe that all three methods have comparable runtimes. As can be expected, the method with $\alpha = \mathbf{0}$ is always faster than the method with $\alpha = \mathbf{1/3}$. The Randomized method is insignificantly faster than the method with $\alpha = \mathbf{0}$ for small r but the slowest for large r.

In addition, note that performance of the method with $\alpha = \mathbf{1/3}$ grows sublinearly w.r.t. r, in perfect accordance to Theorem 1. As r approaches n, speed of that method increases and finally matches that of the method with $\alpha = \mathbf{0}$. For comparison, MATLAB internal function *svds* working on the $100 \times 10,240$ matrix (B) takes approximately 1.4 s independently of rank $r \leq 33$; if $r > 33$, it switches to computation of full factorization, which takes ≈ 0.15 s.

Because all methods were implemented in MATLAB, the runtime was heavily affected by internal MATLAB features, and thus all the conclusions about method runtime are preliminary at best. Moreover, it makes a fair comparison with performance of efficiently coded methods impossible. There are multiple operations, such as memory allocation and data transfer, that must be optimized and are not accounted for in our computational cost analysis. Efficient coding that could be run against LAPACK and similar packages constitutes separate and implementation focused task and is not part of this paper.

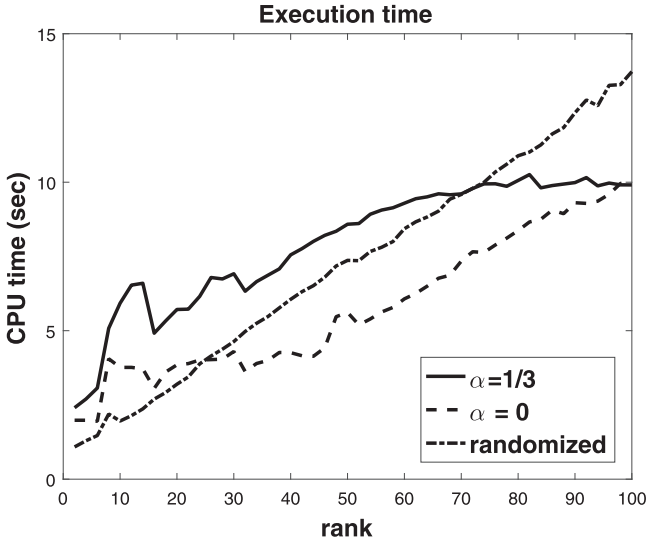

FIGURE 7 Runtime performance of the methods

6 | CONCLUSION

We now summarize the properties of the method presented.

Asymptotical computational cost: The method requires either $O(Nnr^{0.807})$ or $O(Nnr^{0.376})$ operations depending on the matrix multiplication method used. This estimate is asymptotically lower than the cost of the best deterministic truncated SVD methods for nonsymmetric matrices.[1] It exceeds the cost $O(Nn\log r + Nr^2)$ of the Randomized algorithm[6] in the cases when both r and n/r are large. Potentially, the computational cost of the method can be further decreased if more efficient methods are developed for the matrices $\boldsymbol{P}_{ij}$ exploiting features of these matrices (nonnegative definiteness, diagonal submatrices).

Accuracy: In simplified form, the estimate for both Frobenius and L_2 norms of approximation error is $3(\log_2 n - \log_2 r)\,\tau_{r+1}(\boldsymbol{X})$ (see Theorem 2). It is worse but comparable with minimal possible errors $\tau_{r+1}(\boldsymbol{X}), \sigma_{r+1}(\boldsymbol{X})$ in Frobenius and L_2 norms, respectively. It is significantly better than accuracy estimate of $11\sqrt{nr}\sigma_{r+1}(\boldsymbol{X})$ for Randomized algorithms.[6] If the trailing singular values of $\boldsymbol{X}$ decline fast (which is often the case) and thus $\tau_i(\boldsymbol{X}) \approx \sigma_i(\boldsymbol{X})$, $i = r+1, \ldots, n$, the approximation error of this method approaches the best possible one in both norms:

$$\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_2 \le \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{V}\boldsymbol{V}^T\|_F \le \sum_{j=0}^{J-1} \tau_{r_j+1}(\boldsymbol{X}) \approx \sigma_{r+1}\,.$$

In particular, in the case of full factorization $r = \operatorname{rank}\boldsymbol{X}$, the approximation error of the method is theoretically zero and is limited only by computer accuracy. Moreover, the numerical simulations show that the approximation error vanishes as r grows, long before it reaches rank $\boldsymbol{X}$. Finding a combination of the algorithm parameters q, c, α that provides the best trade-off between accuracy and computational cost is a topic of future research.

Parallel implementation: The algorithm has a binary tree structure. Any two mutually independent nodes of the tree (i.e., nonintersecting submatrices of $\boldsymbol{X}$) can be processed simultaneously. Thus, a high degree of parallelization for the algorithm can be achieved, especially at low levels. At the same time, the processing of the blocks on low levels takes the bulk of computational load of the algorithm. For higher levels, parallelization must rely on parallel implementation of matrix multiplication. Another factor that contributes to possible parallelization is simplicity of the method.

Performance on streaming data: In many applications (e.g., Gaussian Processes[8] and other correlation analysis[7]), it is typical that the entries of the matrix $\boldsymbol{X}$ are not available all at once, but come one column a time. In such a case, our scheme allows one to process data as they come, effectively *scanning* the binary tree bottom-to-top along the branches and releasing intermediate results of factorization each time when the number of columns reaches $2^i q, i = 1, \ldots$. The most complicated part of such modification seem to be efficient in handling of the matrices $\boldsymbol{U}_{ij}$; it may lead to slight increase of computational cost.

One-pass processing: Note that the computational scheme is one-pass — each entry of $\boldsymbol{X}$ is addressed just once, at Step 1 of the algorithm, in computation of small $q \times q$ matrix $\boldsymbol{C}_{i,0} = \boldsymbol{X}_{i,0}^T \boldsymbol{X}_{i,0}$. Thus, there is no need of keeping the entries of $\boldsymbol{X}$, which are already processed.

Very big matrices: It is implicitly assumed throughout this paper that the smaller dimension n of the matrix $\boldsymbol{X}$ is small enough so that submatrices of the size $n \times r_j$ can be handled without difficulty. However, it is possible that n is so large that such submatrices do not fit into memory. In such a case, the matrix $\boldsymbol{X}$ can be first split into $\frac{n}{r}$ "bands" of the dimension $r \times N$, and exact SVD factorization is performed on each band, with setting $r_j \equiv r$. Then, the factorizations of the whole matrix $\boldsymbol{X}$ can be performed by combination of blocks similar to Step II. On that last stage, optimal setting of the intermediate ranks r_j cannot be used as they will quickly grow to n; instead, their values must be bounded by some constant depending on the available memory. Due to this limitation, the optimal accuracy of SVD will not be achieved. The asymptotical computational cost of such modification will be the same as $O(Nnr^{\rho})$.

Error-bound factorization: This paper is devoted to the problem of rank-bound factorization. However, the algorithm can be redesigned for error-bound factorization. Lemma 3 provides an efficient tool for the control of approximation error during factorization, because the values $\sigma_s(\boldsymbol{P}_{ij})$ are computed on the Step **II**. Lemma 3 may also be useful for accuracy analysis of other divide-and-conquer SVD methods.

Memory efficiency: The memory required by the method can be significantly reduced if the matrices $\widetilde{\boldsymbol{V}}_j$ are computed explicitly, instead of storing $\boldsymbol{U}_{ij}$ for future construction of $\boldsymbol{V}$. Unfortunately, such modification will destroy the computational cost estimate, which is the main feature of the method. Nevertheless, such modification may be useful for parallel processing or streaming data.

There are several possible ways to speed up our method further. In particular, one can try to exploit special structure of the matrix $\boldsymbol{P}_{i,j}$: its major blocks are diagonal. Another, more promising modification is to combine the main ideas of the proposed method with randomized schemes as shown in the papers.[6,15]

As follows from the theoretical analysis, the presented method claims to possess the best asymptotical estimates of speed and accuracy for truncated SVD. An efficient implementation and more numerical experiments on large matrices will help make the proposed technique more mature.

ACKNOWLEDGEMENT

Research of the first and third authors was supported by the Office of Naval Research (ONR) under the contract N00014-16-C-2009.

CONFLICT OF INTEREST

There are no conflicts of interest to this work.

ORCID

Serge L. Shishkin https://orcid.org/0000-0002-9864-030X

REFERENCES

1. Demmel J, Dumitriu I, Holtz O. Fast linear algebra is stable. Numerische Mathematik. 2007;108(1):59–91.
2. Gu M, Eisenstat SC. Efficient algorithms for computing a strong rank-revealing QR factorization. SIAM J Sci Comput. 1996;17:848–869.
3. Skiena SS. Matrix multiplication. In: The algorithm design manual matrix multiplication. New York, NY: Springer-Verlag; 1998.
4. Coppersmith D, Winograd S. Matrix multiplication via arithmetic progressions. J Symb Comput. 1990;9(3):251–280.
5. Bosner N. Fast methods for large scale singular value decomposition. PhD [thesis]. Zagreb, Croatia: University of Zagreb; 2006.
6. Halko N, Martinsson PG, Tropp JA. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review. 2011;53(2):217–288.
7. Tzeng J. Fast singular value decomposition for large-scale growing data. Proceedings of the 5th International Conference on Digital Society (ICDS); 2011 Feb 23–28; Guadeloupe, France. Wilmington, DE: International Academy, Research, and Industry Association (IARIA); 2011. p. 193–198.
8. Bopardikar SD, Ekladious GSE. Sequential randomized matrix factorization for Gaussian processes. Proceedings of the IEEE International Conference on Big Data; 2016 Dec 5–8; Washington, DC. Piscataway, NJ: IEEE; 2016. p. 3957–3959.

9. Ambikasaran S, Foreman-Mackey D, Greengard L, Hogg DW, O'Neil M. Fast direct methods for Gaussian processes. IEEE Trans Pattern Anal Mach Intell. 2016;38(2):252–265.
10. Vogel J, Xia J, Cauley S, Balakrishnan V. Superfast divide-and-conquer method and perturbation analysis for structured Eigenvalue solutions. SIAM J Sci Comput. 2016;38(3):A1358–A1382.
11. Gentle JE. Singular value factorization. In: Numerical linear algebra for applications in statistics. New York, NY: Springer-Verlag; 1998.
12. Bai Z, Demmel J, Gu M. Inverse free parallel spectral divide and conquer algorithms for nonsymmetric eigenproblems. Numerische Mathematik. 1997;76:279–308.
13. Stewart GW. Matrix algorithms, volume 2: Eigensystems. Philadelphia, PA: SIAM; 2001.
14. Rasmussen CE, Williams CKI. Gaussian processes for machine learning. Cambridge, MA: The MIT Press; 2006. http://www.gaussianprocess.org/gpml/data/
15. Bopardikar SD. Randomized matrix factorization for Kalman filtering. Proceedings of the American Control Conference; 2017 May 24–26; Seattle, WA. Piscataway, NJ: IEEE; 2017. p. 5795–5800.

How to cite this article: Shishkin SL, Shalaginov A, Bopardikar SD. Fast approximate truncated SVD. *Numer Linear Algebra Appl*. 2019;26:e2246. https://doi.org/10.1002/nla.2246