# A Randomized Singular Value Decomposition for Third-Order Oriented Tensors

*Minghui Ding[†]        Yimin Wei[‡]        Pengpeng Xie[§]

**Abstract**

The oriented singular value decomposition (O-SVD) proposed by Zeng and Ng provides a hybrid approach to the t-product based third-order tensor singular value decomposition with the transformation matrix being a factor matrix of the higher order singular value decomposition. Continuing along this vein, this paper explores realizing the O-SVD efficiently by drawing a connection to the tensor-train rank-1 decomposition and gives a truncated O-SVD. Motivated by the success of probabilistic algorithms, we develop a randomized version of the O-SVD and present its detailed error analysis. The new algorithm has advantages in efficiency while keeping good accuracy compared with the current tensor decompositions. Our claims are supported by numerical experiments on several oriented tensors from real applications.

**Keywords:** Oriented tensor · Singular value decomposition · Truncation  · Randomized algorithm

**Mathematics Subject Classification:** 65F30, 65F99, 15A69

---

[*]Published in Journal of Optimization Theory and Applications.

[†]School of Mathematical Sciences, Ocean University of China, Qingdao 266100, China. E-Mail: `dingminghui@stu.ouc.edu.cn`

[‡]School of Mathematical Sciences and and Key Laboratory of Mathematics for Nonlinear Sciences, Fudan University, Shanghai 200433, China. E-Mail: `ymwei@fudan.edu.cn`

[§]Corresponding author (P. Xie). School of Mathematical Sciences, Ocean University of China, Qingdao 266100, China. E-Mail: `xie@ouc.edu.cn`.

# 1 Introduction

Tensors [21, 28, 29] are multidimensional arrays that have been used in diverse fields of applications, including psychometrics [4], image/video and signal processing [10], machine learning [33], and web link analysis [22]. The compression, sort, analysis, and many other processing of tensor data rely on the tensor decomposition. Various tensor decompositions under different tensor products such as the CANDECOMP/PARAFAC [4, 16], higher order singular value decomposition (HOSVD) [11, 35], T-SVD [19, 20], T-CUR [5], tensor-train [27] and tensor-train rank-1 (TTr1) SVD (TTr1SVD) [2] have been investigated to extend linear algebra methods to the multilinear context. Among these decompositions, the HOSVD algorithm is not orientation dependent and can achieve high compression ratios if the target rank for the Tucker approximation is small compared to the original dimensions. By contrast, since the Fourier matrix is independent of the tensor, the T-SVD cannot embody the data feature and is not suitable for all orientation-dependent data. However, we usually confront orientation-dependent tensors which have high correlation among frontal slices in applications. Recently, Zeng and Ng [40] proposed a new decomposition for third-order tensors based on the HOSVD and T-SVD, which is named oriented singular value decomposition (O-SVD). Like the T-SVD, the O-SVD also aims at tensors with fixed orientations and has been demonstrated to be useful in approximation and data compression.

While the O-SVD combines the ideas of the HOSVD and T-SVD, it can still be expressed with the sum of outer product terms. This property is also reminiscent of the TTr1SVD which decomposes an arbitrary tensor into a finite sum of orthogonal rank-1 outer products. Unlike the O-SVD, one needs to progressively reshape and compute the SVD of each singular vector to produce the TTr1 decomposition [2]. Therefore, in this paper, we first consider the acceleration of the computation of the O-SVD by means of the constructive approach for the TTr1SVD. Then we turn to the numerical approximation of the O-SVD. A straightforward $r$-term approximation of the O-SVD has been proposed by keeping $r$ terms for which singular values are the largest in magnitude and discarding the other terms. We propose an alternative truncation strategy for better preserving the original structure inherited from the HOSVD and T-SVD.

In recent years, randomized matrix methods have been used to efficiently and accurately compute approximate low-rank matrix decompositions and the least squares problem (see [12, 15, 24, 32, 38, 39]). These algorithms are easy to implement, and have been extended to the singular value decomposition of tensors based on different tensor products [6–9, 26, 41]. Specially, Zhang $et$ $al.$ [41] proposed an algorithm that extends a well-known randomized matrix method to the T-SVD that was called the RT-SVD, which is more computationally efficient on large data sets. Che and Wei [6] designed randomized algorithms for computing the Tucker and tensor train approximations of tensors with unknown multilinear rank and analyzed their probabilistic error bounds under certain assumptions. Minster $et$ $al.$ [26] presented randomized algorithms of the HOSVD (RHOSVD) and sequentially truncated HOSVD (RSTHOSVD) in the Tucker representation and gave a detailed probabilistic error analysis for both algorithms. They also applied the adaptive randomized algorithm to find a low-rank representation satisfying a given tolerance and proposed a structure-preserving decomposition where the core tensor retains favorable properties of the original tensor. However, the randomized algorithms mentioned above are not useful for tensors with a fixed orientation involving time series or other ordered data that are highly correlated among slices. Hence, we are motivated to design a new type of randomization strategy corresponding to the O-SVD, and hopefully, this technique can greatly reduce the computational cost while maintaining the accuracy.

The rest of this paper is organized as follows. In Section 2, we introduce some basic definitions and preliminaries. In Section 3, we briefly introduce the O-SVD, discuss its connection to TTr1SVD and present a new truncation strategy for the O-SVD. Thereafter, in Section 4, a randomized tensor algorithm based on the O-SVD is proposed. We also give an expected error bound and compare the

computational and memory cost with the RT-SVD and RHOSVD. Section 5 presents numerical results on the approximation error and the computational complexity of the algorithm, and compares it with some existing methods. Some conclusions are presented in Section 6.

## 2    Preliminaries

In this section, we introduce definitions and notation used throughout the paper. Scalars are denoted by lowercase letters, e.g. $a$, vectors are denoted by bold-face lowercase letters, e.g. $\boldsymbol{a}$, matrices are denoted by bold-face capitals, e.g. $\boldsymbol{A}$, and tensors are written as calligraphic letters, e.g. $\mathcal{A}$. The $i$th entry of a vector $\boldsymbol{a}$ is denoted by $a_i$, and the $(i, j, k)$th element of a third-order tensor $\mathcal{A}$ is denoted by $a_{ijk}$. For convenience, we sometimes use the MATLAB notation to denote subportions of a matrix or tensor, (e.g., $\mathcal{A}(:,:,k)$ denotes the $k$th frontal slice of a tensor and $\boldsymbol{A}(i,:)$ the $i$th row of a matrix).

A mode-$n$ fiber is a column vector defined by fixing every index but the $n$th index, and a mode-$(m, n)$ slice is a matrix defined by fixing every index but the $m$th index and the $n$th index. A mode-$(1, 2)$ slice is also called a frontal slice. The mode-$n$ unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is denoted by $\boldsymbol{A}_{(n)}$ and arranges the mode-$n$ fibers to be the columns of the resulting matrix.

**Definition 2.1** ( Mode-$n$ product [21]). *The mode-$n$ product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $\boldsymbol{B} \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n \boldsymbol{B}$, is a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$ with*

$$c_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} b_{j i_n},$$

*where $n = 1, 2, \ldots, N$.*

**Definition 2.2** ( Inner product [21]). *The inner product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1, i_2, \ldots, i_N} \overline{a}_{i_1 i_2 \cdots i_N} b_{i_1 i_2 \ldots i_N}.$$

We call two tensors orthogonal if their inner product is 0. The norm of a tensor is taken to be the Frobenius norm $\|\mathcal{A}\|_F = \langle \mathcal{A}, \mathcal{A} \rangle^{1/2}$.

**Definition 2.3** ( Outer product [2]). *A third-order rank-1 tensor $\mathcal{A}$ can always be written as the outer product*

$$\sigma(\boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}) \quad \text{with components } a_{ijk} = \sigma a_i b_j c_k$$

*with $\sigma \in \mathbb{R}$, whereas $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are vectors of arbitrary lengths. Using the mode-$n$ multiplication, this outer product can also be written as $\sigma_{\times_1} \boldsymbol{a}_{\times_2} \boldsymbol{b}_{\times_3} \boldsymbol{c}$, where $\sigma$ is now regarded as a $1 \times 1 \times 1$ tensor.*

**Definition 2.4** ( Tensor-tensor product [40]). *The three-mode product of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathcal{B} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$ denoted by $\mathcal{A} *_3 \mathcal{B}$, is of size $I_1 \times I_4 \times I_3$, which is given by*

$$(\mathcal{A} *_3 \mathcal{B})(:,:,k) = \mathcal{A}(:,:,k)\mathcal{B}(:,:,k), \quad k = 1, \ldots, I_3.$$

**Definition 2.5** ( Transpose). *If $\mathcal{A}$ is an $I_1 \times I_2 \times I_3$ tensor, then $\mathcal{A}^T$ is an $I_2 \times I_1 \times I_3$ tensor obtained by transposing each of the frontal slices, i.e., $\mathcal{A}^T(:,:,i) = \mathcal{A}(:,:,i)^T$, for $i = 1, \ldots, I_3$.*

We should note that the definition here of a transpose operation for tensors is different from that in [20]. The following lemma introduces some properties of the mode-$n$ product.

3

**Lemma 2.1** ( [21, 40] ). *Let* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}, \boldsymbol{M} \in \mathbb{R}^{J_n \times I_n}, \mathcal{B} = \mathcal{A} \times_n \boldsymbol{M}, n = 1, 2, 3.$ *Then*
(1) $\boldsymbol{B}_{(n)} = \boldsymbol{M} \boldsymbol{A}_{(n)};$
(2) *If* $\boldsymbol{M}^{(n)}$ *is orthonormal for* $J_n = I_n$, *then* $\mathcal{A} = \mathcal{B} \times_n \boldsymbol{M}^T;$
(3) *For* $\boldsymbol{M}_1 \in \mathbb{R}^{J_n \times I_n}, \boldsymbol{M}_2 \in \mathbb{R}^{J_n' \times J_n}$, *then* $(\mathcal{A} \times_n \boldsymbol{M}_1) \times_n \boldsymbol{M}_2 = \mathcal{A} \times_n (\boldsymbol{M}_2 \boldsymbol{M}_1).$

**Definition 2.6** ( Tensor rank [11, 19]). *Let* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$.
(1) *The n-rank of* $\mathcal{A}$, *denoted by* $\mathrm{rank}_n(\mathcal{A})$, *is the dimension of the vector space spanned by all mode-n fibers. For example,* $\mathrm{rank}_3(\mathcal{A}) = \mathrm{rank}(\boldsymbol{A}_{(3)})$.
(2) *The multirank of* $\mathcal{A}$ *is a mode-3 fiber* $\mathrm{rank}_m(\mathcal{A}) \in \mathbb{R}^{I_3}$ *such that* $\mathrm{rank}_m(\mathcal{A})(i)$ *is the rank of* $(\mathcal{A} \times_3 \boldsymbol{M})(:, :, i)$ *where* $\boldsymbol{M}$ *represents different meanings under different tensor products.*

The Tucker rank of tensor $\mathcal{A}$ is a vector with its elements being the ranks of matrix unfoldings with respect to the corresponding modes, i.e., $(\mathrm{rank}_1(\mathcal{A}), \mathrm{rank}_2(\mathcal{A}), \mathrm{rank}_3(\mathcal{A}))$. If $\boldsymbol{M}$ is the left singular matrix of $\boldsymbol{A}_{(3)}$ with $\boldsymbol{A}_{(3)} = \boldsymbol{U}^{(3)} \boldsymbol{S}^{(3)} \boldsymbol{V}^{(3)T}$, it is shown in [40] that

$$\max \mathrm{rank}_m(\mathcal{A}) = \max \{ \mathrm{rank}((\mathcal{A} \times_3 \boldsymbol{U}^{(3)T})(:, :, i)) \} \leq \min \{ \mathrm{rank}_1(\mathcal{A}), \mathrm{rank}_2(\mathcal{A}) \}.$$

# 3 O-SVD

We first review the O-SVD developed by Zeng and Ng [40], which is built on the operations of tensors introduced in Section 2. In order to better understand the O-SVD and improve its numerical realization, we show that the outer product form of the O-SVD is in fact the TTr1SVD introduced in [2]. We then proceed to a type of truncated O-SVD (TO-SVD) in an analogous manner to the truncated T-SVD (TT-SVD) [20] and develop a rigorous error analysis.

**Theorem 3.1** (O-SVD [40]). *Let* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *and* $R_3 = \mathrm{rank}_3(\mathcal{A})$. *There exists an orthogonal matrix* $\boldsymbol{U}^{(3)} \in \mathbb{R}^{I_3 \times I_3}$, *three tensors* $\mathcal{U} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$, $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathcal{V} \in \mathbb{R}^{I_2 \times I_2 \times I_3}$ *such that*

$$\mathcal{A} = (\mathcal{U} *_3 \mathcal{S} *_3 \mathcal{V}) \times_3 \boldsymbol{U}^{(3)}, \tag{3.1}$$

*where*
(1) $\mathcal{U}(:, :, i), \mathcal{V}(:, :, i)$ *are orthogonal and* $\mathcal{S}(:, :, i)$ *is a nonnegative diagonal matrix for* $i = 1, 2, \ldots, R_3;$
(2) $\mathcal{U}(:, :, i), \mathcal{V}(:, :, i)$ *and* $\mathcal{S}(:, :, i)$ *are all zero matrices for* $i = R_3 + 1, \ldots, I_3$.

The diagonal elements $s_{jji}$ of each frontal slice of $\mathcal{S}$ are called the singular values of the pair $(\mathcal{A}, \mathcal{S})$.

**Theorem 3.2** ( [40] ). *Let the core tensor corresponding to the O-SVD of* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *be* $\mathcal{S}$. *Then each slice* $\mathcal{S}(:, :, i) \in \mathbb{R}^{I_1 \times I_2}$ *has the following property:*

$$\|\mathcal{S}(:, :, 1)\|_F \geq \|\mathcal{S}(:, :, 2)\|_F \geq \cdots \geq \|S(:, :, I_3)\|_F \geq 0,$$

*where* $\|\mathcal{S}(:, :, i)\|_F = \sigma_i$ *with* $\sigma_i$ *being the ith largest singular value of* $\boldsymbol{A}_{(3)}$ *and*

$$\sigma_i \geq \mathcal{S}(1, 1, i) \geq \mathcal{S}(2, 2, i) \geq \cdots \geq \mathcal{S}(r_2, r_2, i) \geq 0$$

*for* $i = 1, 2, \ldots, r_1$, *where* $r_1 = \min \{I_3, I_1 I_2\}$, $r_2 = \min \{I_1, I_2\}$.

## 3.1 O-SVD and TTr1SVD

It follows from Theorem 3.1 that there are two steps in the computational procedure of the O-SVD. The first step is to find a basis of the space spanned by the frontal slices of $\mathcal{A}$. Specifically, one needs to conduct the "economical" SVD of the $I_3 \times I_1 I_2$ matrix $\boldsymbol{A}_{(3)}$

$$\boldsymbol{A}_{(3)} = \boldsymbol{U}^{(3)} \boldsymbol{S}^{(3)} \boldsymbol{V}^{(3)T},$$

where the number of non-zero singular values obtained is equal to the number of the desired basis. Then, the frontal slices of $\widetilde{\mathcal{A}} = \mathcal{A} \times_3 \boldsymbol{U}^{(3)T}$ are the basis we are looking for. Motivated by the observation that

$$\widetilde{\mathcal{A}}(:,:,i) = \sigma_i \widetilde{\boldsymbol{V}}_i = \sigma_i \texttt{reshape}(\boldsymbol{V}^{(3)}(:,i), [I_1, I_2]), \tag{3.2}$$

where the operator $\texttt{reshape}$ returns the $I_1$-by-$I_2$ matrix $\widetilde{\boldsymbol{V}}_i$ whose elements are taken columnwise from $\boldsymbol{V}^{(3)}(:,i)$, we can compute the SVD directly for each matrix $\widetilde{\boldsymbol{V}}_i$

$$\widetilde{\boldsymbol{V}}_i = \boldsymbol{U}_i \boldsymbol{S}_i \boldsymbol{V}_i^T \tag{3.3}$$

instead of forming $\widetilde{\mathcal{A}}$ first. This procedure is directly inspired by the algorithm of TTr1 decomposition [2], which requires recursively reshaping the right singular vectors $\boldsymbol{V}^{(3)}(:,i)$, and computing their SVDs. This algorithm is called TTr1SVD and gives rise to the formation of a tree. Since the first step of the TTr1SVD algorithm is to expand the tensor along the selected mode, the O-SVD is in fact the TTr1SVD for the third-order oriented tensors with the processing order $\rho = [3, 1, 2]$. Let $\sigma_{ij}$ denote the $j$th largest singular value of $\widetilde{\boldsymbol{V}}_i$ where $i = 1, 2, \ldots, r_1$, $j = 1, 2, \ldots, r_2$. Substituting (3.3) into (3.2), it is easy to derive that

$$s_{jji} = \sigma_i \sigma_{ij}, \, \mathcal{U}(:,:,i) = \boldsymbol{U}_i, \, \mathcal{V}(:,:,i) = \boldsymbol{V}_i^T. \tag{3.4}$$

Since any matrix can be written as a sum of rank-1 terms, we can also rewrite $\mathcal{A}$ as

$$\mathcal{A} = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} s_{jji} \times_1 \boldsymbol{u}_{ij} \times_2 \boldsymbol{v}_{ij} \times_3 \boldsymbol{u}_i, \tag{3.5}$$

where $\boldsymbol{u}_i$, $\boldsymbol{u}_{ij}$, $\boldsymbol{v}_{ij}$ are the column vectors of $\boldsymbol{U}^{(3)}$, $\boldsymbol{U}_i$, $\boldsymbol{V}_i$ respectively. Consequently, the O-SVD also has three main features that render it similar to the matrix SVD.

**Corollary 3.1** ( [2]). *Let (3.5) be the outer product form of the O-SVD of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and the number of rank-1 terms be $R = r_1 r_2$. Then,*
*(1) the scalars $s_{jji}$ are the weights of the outer products in the decomposition,*
*(2) the outer products affiliated with each singular value are tensors of unit Frobenius norm, since each product vector (or mode vector) is a unit vector, and*
*(3) each outer product in the decomposition is orthogonal to all the others.*

Furthermore, we can obtain a more economical expression, similar to the form of the $(L_r, L_r, 1)$-term decomposition [34].

**Corollary 3.2.** *Let (3.1) be the O-SVD of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Then*

$$\mathcal{A} = \sum_{r=1}^{R_3} \boldsymbol{H}_r \otimes \boldsymbol{u}_r, \quad \texttt{rank}\,(\boldsymbol{H}_r) = L_r > 0 \ \textit{for } 1 \leq r \leq R_3, \tag{3.6}$$

*where $\boldsymbol{H}_r = \widetilde{\mathcal{A}}(:,:,r) = \mathcal{U}(:,:,r)\mathcal{S}(:,:,r)\mathcal{V}(:,:,r)$, $\boldsymbol{u}_r = \boldsymbol{U}^{(3)}(:,r)$, and $\otimes$ is the tensor product defined by $(H \otimes \boldsymbol{u})(i,j,k) = h_{ij}u_k$.*

## 3.2 TO-SVD and its Error Analysis

For an orientation-dependent tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the numerical rank of $\boldsymbol{A}_{(3)}$ is usually much smaller than $I_3$. In this circumstance, we can utilize the truncation strategy to efficiently compute an approximate O-SVD. The Eckart-Young theorem [13] states that an optimal rank-$k$ approximation to a matrix can be constructed using the rank-$k$ truncated SVD. Similarly, an $r$-term approximation for the O-SVD can be obtained by truncating (3.5) to the first $r$ terms.

**Lemma 3.1** (Approximation [2, 40]). *Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and denote $\widetilde{\sigma}_i$ as the ith singular value of the pair $(\mathcal{A}, \mathcal{S})$ in descending order. Denote by $\mathcal{A}_r$ the r-term approximation by the O-SVD. Then we have*

$$\|\mathcal{A} - \mathcal{A}_r\|_F^2 = \sum_{i=r+1}^{r_1 r_2} \widetilde{\sigma}_i^2. \tag{3.7}$$

Notice that the $r$-term approximation of $\mathcal{A}$ requires reordering all the singular values, finding the outer product corresponding to each singular value and adding them one by one, which is undoubtedly laborious and time-consuming. At the same time, the original structures of Theorem 3.1 and Corollary 3.2 cannot be maintained.

To overcome this drawback, we adopt an alternative truncation strategy. Recall that the factor matrix of the truncated HOSVD (THOSVD) [36] is obtained from a truncated SVD of the mode-$k$ unfolding of the tensor. For the TT-SVD, it consists of transforming the tensor to the Fourier domain and applying the truncated SVD to each frontal slice of the tensor. Following the procedure of the O-SVD, we consider a truncation method combining the ideas of TT-SVD and THOSVD. We first perform a truncated SVD of $\boldsymbol{A}_{(3)}$ to get the approximate matrix $\boldsymbol{U}_{k_1}^{(3)}$ of the left singular matrix $\boldsymbol{U}^{(3)}$, where $k_1$ is the target truncation rank. Secondly, for each frontal slice of $\mathcal{A} \times_3 (\boldsymbol{U}_{k_1}^{(3)})^T$, we conduct the economical SVD with different target truncation terms. Let $\boldsymbol{k}_2 = [k_{21}, \ldots, k_{2k_1}]^T$ be the target multirank of the second step. Obviously, this kind of truncation leads to different nonzero blocks in each frontal slice of $\mathcal{S}$, and so do $\mathcal{U}$ and $\mathcal{V}$. For the convenience of description, we set $k_2 = \max\{k_{21}, \ldots, k_{2k_1}\}$. Now we are ready to summarize the above discussion in the following definition.

**Definition 3.1** ($\boldsymbol{k}$-term TO-SVD). *Given a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, define the truncation of the O-SVD to $\boldsymbol{k}$ terms of $\mathcal{A}$ as*

$$\mathcal{A}_{\boldsymbol{k}} = (\mathcal{U}_{k_2} *_3 \mathcal{S}_{k_2} *_3 \mathcal{V}_{k_2}) \times_3 \boldsymbol{U}_{k_1}^{(3)}, \tag{3.8}$$

*where*
*(1) $\boldsymbol{k} = [k_1; \boldsymbol{k}_2] \in \mathbb{R}^{k_1+1}$ is the target rank vector;*
*(2) $\mathcal{U}_{k_2}(:,:,i) \in \mathbb{R}^{I_1 \times k_2}$ and $\mathcal{V}_{k_2}^T(:,:,i) \in \mathbb{R}^{I_2 \times k_2}$ have $k_{2i}$ orthogonal columns for $i = 1, 2, \ldots, k_1$. $\mathcal{S}_{k_2}(:,:,i) \in \mathbb{R}^{k_2 \times k_2}$ is a nonnegative diagonal matrix for $i = 1, 2, \ldots, k_1$;*
*(3) $\boldsymbol{U}_{k_1}^{(3)} = \boldsymbol{U}^{(3)}(:, 1 : k_1) \in \mathbb{R}^{I_3 \times k_1}$.*

In Algorithm 1, we show how the TO-SVD can be implemented in combination with the improved methods mentioned in Subsection 3.1. The error of the TO-SVD is presented in Theorem 3.3.

---

**Algorithm 1** $\boldsymbol{k}$-term TO-SVD

---

**Input:** $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, target truncation vector $\boldsymbol{k} = [k_1; \boldsymbol{k}_2]$, $\boldsymbol{k}_2 = [k_{21}, \ldots, k_{2k_1}]^T$,
**Output:** $\boldsymbol{U}_{k_1}^{(3)} \in \mathbb{R}^{I_3 \times k_1}$, $\mathcal{U}_{k_2} \in \mathbb{R}^{I_1 \times k_2 \times k_1}$, $\mathcal{S}_{k_2} \in \mathbb{R}^{k_2 \times k_2 \times k_1}$, $\mathcal{V}_{k_2} \in \mathbb{R}^{k_2 \times I_2 \times k_1}$
1: Initialization: $\mathcal{U}_{k_2}$, $\mathcal{S}_{k_2}$, $\mathcal{V}_{k_2}$ are zero tensors of appropriate size;
2: $\left[\boldsymbol{U}_{k_1}^{(3)}, \boldsymbol{S}_{k_1}^{(3)}, \boldsymbol{V}_{k_1}^{(3)}\right] = \text{svds}\left(\boldsymbol{A}_{(3)}, k_1\right)$;
3: **for** $i = 1, 2, \ldots, k_1$, **do**
4: $\quad \widetilde{\boldsymbol{V}}_i = \texttt{reshape}(\boldsymbol{V}_{k_1}^{(3)}(:, i), [I_1, I_2])$;
5: $\quad [\boldsymbol{U}_i, \boldsymbol{S}_i, \boldsymbol{V}_i] = \text{svds}(\widetilde{\boldsymbol{V}}_i, k_{2i})$;
6: $\quad \mathcal{U}_{k_2}(:, 1 : k_{2i}, i) = \boldsymbol{U}_i$, $\mathcal{S}_{k_2}(1 : k_{2i}, 1 : k_{2i}, i) = \boldsymbol{S}_{k_1}^{(3)}(i, i)\boldsymbol{S}_i$, $\mathcal{V}_{k_2}(1 : k_{2i}, :, i) = \boldsymbol{V}_i^T$;
7: **end for**

---

**Theorem 3.3.** *Let $\mathcal{A}_{\boldsymbol{k}}$ be the truncation of the O-SVD to $\boldsymbol{k}$ terms of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Then*

$$\|\mathcal{A} - \mathcal{A}_{\boldsymbol{k}}\|_F^2 = \sum_{i=1}^{k_1} \sum_{j=k_{2i}+1}^{r_2} s_{jji}^2 + \sum_{i=k_1+1}^{I_3} \sum_{j=1}^{r_2} s_{jji}^2. \tag{3.9}$$

*Proof.* Since the Frobenius norm is unitarily invariant, by (3.1) and (3.8), we have

$$\|\mathcal{A} - \mathcal{A}_{\boldsymbol{k}}\|_F = \left\| \mathcal{A} \times_3 \boldsymbol{U}^{(3)T} - \mathcal{A}_{\boldsymbol{k}} \times_3 \boldsymbol{U}^{(3)T} \right\|_F$$
$$= \left\| \mathcal{U} *_3 \mathcal{S} *_3 \mathcal{V} - (\mathcal{U}_{k_2} *_3 \mathcal{S}_{k_2} *_3 \mathcal{V}_{k_2}) \times_3 \left( \boldsymbol{U}^{(3)T} \boldsymbol{U}_{k_1}^{(3)} \right) \right\|_F.$$

Noting that $\boldsymbol{U}^{(3)T} \boldsymbol{U}_{k_1}^{(3)} = \begin{bmatrix} \boldsymbol{I}_{k_1} \\ \boldsymbol{0} \end{bmatrix}$, now let $\widetilde{\mathcal{U}}_{k_2}(:,:,i) = \mathcal{U}_{k_2}(:,:,i)$, $\widetilde{\mathcal{S}}_{k_2}(:,:,i) = \mathcal{S}_{k_2}(:,:,i)$, $\widetilde{\mathcal{V}}_{k_2}(:,:,i) = \mathcal{V}_{k_2}(:,:,i)$, for $i = 1, 2, \ldots, k_1$ and $\widetilde{\mathcal{U}}_{k_2}(:,:,i) = \boldsymbol{0}$, $\widetilde{\mathcal{S}}_{k_2}(:,:,i) = \boldsymbol{0}$, $\widetilde{\mathcal{V}}_{k_2}(:,:,i) = \boldsymbol{0}$ of the corresponding dimension for $i = k_1 + 1, \ldots, I_3$, from which we can obtain

$$\|\mathcal{A} - \mathcal{A}_{\boldsymbol{k}}\|_F^2 = \left\| \mathcal{U} *_3 \mathcal{S} *_3 \mathcal{V} - \widetilde{\mathcal{U}}_{k_2} *_3 \widetilde{\mathcal{S}}_{k_2} *_3 \widetilde{\mathcal{V}}_{k_2} \right\|_F^2$$
$$= \sum_{i=1}^{I_3} \left\| \mathcal{U}(:,:,i)\mathcal{S}(:,:,i)\mathcal{V}(:,:,i) - \widetilde{\mathcal{U}}_{k_2}(:,:,i)\widetilde{\mathcal{S}}_{k_2}(:,:,i)\widetilde{\mathcal{V}}_{k_2}(:,:,i) \right\|_F^2$$
$$= \sum_{i=1}^{I_3} \left\| \mathcal{S}(:,:,i) - \begin{bmatrix} \widetilde{\mathcal{S}}_{k_2}(1:k_{2i}, 1:k_{2i}, i) & \\ & \boldsymbol{0} \end{bmatrix} \right\|_F^2$$
$$= \sum_{i=1}^{k_1} \sum_{j=k_{2i}+1}^{r_2} s_{jji}^2 + \sum_{i=k_1+1}^{I_3} \sum_{j=1}^{r_2} s_{jji}^2,$$

where we used $\mathcal{U}(:,:,i)^T \widetilde{\mathcal{U}}_{k_2}(:,:,i) = \begin{bmatrix} \boldsymbol{I}_{k_{2i}} \\ \boldsymbol{0} \end{bmatrix}$ and $\widetilde{\mathcal{V}}_{k_2}(:,:,i)\mathcal{V}(:,:,i)^T = \begin{bmatrix} \boldsymbol{I}_{k_{2i}} & \boldsymbol{0} \end{bmatrix}$ in the third equality. Finally, observing that $\widetilde{\mathcal{S}}_{k_2}(1:k_{2i}, 1:k_{2i}, i) = \mathcal{S}(1:k_{2i}, 1:k_{2i}, i)$ for $i = 1, 2, \ldots, k_1$ and $s_{jji}$ is the $j$th singular value of $\mathcal{S}(:,:,i)$, we get the desired result. $\qquad \square$

The error will serve as an important reference to compare the accuracy of the randomized O-SVD with the deterministic one. Comparing Theorem 3.3 with Lemma 3.1, we have

$$\|\mathcal{A} - \mathcal{A}_k\|_F \geq \|\mathcal{A} - \mathcal{A}_r\|_F,$$

where $r = \sum_{i=1}^{k_1} k_{2i}$. However, the TO-SVD is cheaper to compute and retains the original structure of the O-SVD.

# 4 Randomized O-SVD

Randomized algorithms play a key role in low-rank approximations of large matrices. In this section, the scheme of the matrix randomized SVD (R-SVD) [15, Section 4] is extended to a randomized O-SVD algorithm (RO-SVD). We first review the matrix R-SVD method and its expected error estimate that we will use later to analyze the error in the RO-SVD method. We also discuss the computational and memory costs of the proposed randomized algorithm.

## 4.1 Randomized SVD

Randomized SVD, popularized by [15], is a computationally efficient way to compute a low-rank approximation of a matrix. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}(m \leq n)$, a target rank $k$, and an oversampling parameter $p$, we first multiply the matrix $\boldsymbol{A}$ by a Gaussian random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n \times (k+p)}$. The matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ thus contains random linear combinations of the columns of $\boldsymbol{A}$. A thin QR of $\boldsymbol{Y}$ is then computed, so that range($\boldsymbol{Y}$) = range($\boldsymbol{Q}$). The idea is if $\boldsymbol{A}$ has rapidly decaying singular values, the dominant part of the range of $\boldsymbol{A}$ is marked by the first $k$ or so the left singular vectors, that is, $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{A} = \widehat{\boldsymbol{A}}$. Then, we compute a thin SVD of much smaller matrix $\boldsymbol{Q}^T\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}_k^T$, truncate down to the target rank $k$, and compute $\boldsymbol{U}_k = \boldsymbol{Q}\boldsymbol{U}$ to obtain the low-rank approximation $\widehat{\boldsymbol{A}} = \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T$.

The techniques described above work well for matrices whose singular values exhibit some decay, but they may produce a poor basis when the input matrix has a flat singular spectrum or when the input matrix is very large. A modified scheme originally proposed in [30], makes use of power iteration to improve the accuracy of randomized algorithms in these situations. Specifically, the projection step $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ is replaced with $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^T)^q\boldsymbol{A}\boldsymbol{\Omega}$ for small integer $q$, where $q = 1$ or $q = 2$ usually suffices in practice. In particular, when $q = 0$, the algorithm is equivalent to the basic randomized SVD [15]. In our paper, we adopt Algorithm 2, a numerically stable version, which is available in [15, Algorithm 4.4] that alternates the QR factorization with the matrix-matrix products and assume that it can be invoked as $[\boldsymbol{U}_k, \boldsymbol{S}_k, \boldsymbol{V}_k] = \mathrm{rsvd}(\boldsymbol{A}, \boldsymbol{\Omega}, k, p, q)$.

**Remark 4.1.** *For Gaussian test matrices, it is adequate to choose the oversampling parameter to be a small constant, such as $p = 5$ or $p = 10$. There is rarely any advantage to select $p > k$. This observation, first presented in [25], demonstrates that a Gaussian test matrix results in a negligible amount of extra computation.*

---

**Algorithm 2** R-SVD method with power iteration [15]

---

**Input:** $A \in \mathbb{R}^{m \times n}$, Gaussian random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n \times (k+p)}$, target truncation term $k$, a parameter $q$, and oversampling parameter $p$
**Output:** $\boldsymbol{U}_k \in \mathbb{R}^{n \times k}$, $\boldsymbol{S}_k \in \mathbb{R}^{k \times k}$, $\boldsymbol{V}_k \in \mathbb{R}^{n \times k}$
1: Form $\boldsymbol{Y}_0 = \boldsymbol{A}\boldsymbol{\Omega}$ and compute its QR factorization $\boldsymbol{Y}_0 = \boldsymbol{Q}_0\boldsymbol{R}_0$;
2: for $j = 1, 2, \ldots, q$, do
3:     Form $\widehat{\boldsymbol{Y}}_j = \boldsymbol{A}^T\boldsymbol{Q}_{j-1}$ and compute its QR factorization $\widehat{\boldsymbol{Y}}_j = \widehat{\boldsymbol{Q}}_j\widehat{\boldsymbol{R}}_j$;
4:     Form $\boldsymbol{Y}_j = \boldsymbol{A}\widehat{\boldsymbol{Q}}_j$ and compute its QR factorization $\boldsymbol{Y}_j = \boldsymbol{Q}_j\boldsymbol{R}_j$;
5: end
6: $\boldsymbol{Q} = \boldsymbol{Q}_q$;
7: Form $\boldsymbol{B} = \boldsymbol{Q}^T\boldsymbol{A} \in \mathbb{R}^{(k+p) \times n}$;
8: $[\boldsymbol{U}, \boldsymbol{S}_k, \boldsymbol{V}_k] = \mathrm{svds}(\boldsymbol{B}, k)$;
9: Form $\boldsymbol{U}_k = \boldsymbol{Q}\boldsymbol{U}$.

---

When $\boldsymbol{A}$ is dense and of size $n \times n$, the basic randomized SVD takes $\mathcal{O}(kn^2)$ flops and Algorithm 2 requires $2q + 1$ times as many matrix-matrix multiplications as the basic randomized SVD. An error bound for Algorithm 2 in the Frobenius norm is presented below, which can be found in [41].

**Theorem 4.1** (Average Frobenius Error for Algorithm 2). *Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\Omega} \in \mathbb{R}^{n \times (k+p)}$ be a Gaussian random matrix with $p \geq 2$ being the oversampling parameter. Suppose that $\boldsymbol{Q}$ is obtained from Algorithm 2 and $\boldsymbol{B}_k$ is the rank-k truncated SVD of $\boldsymbol{Q}^T\boldsymbol{A}$, then*

$$\mathbb{E}_{\boldsymbol{\Omega}} \left\| \boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{A} \right\|_F^2 \leq \mathbb{E}_{\boldsymbol{\Omega}} \left\| \boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}_k \right\|_F^2 \leq \left( 1 + \frac{k}{p-1}\tau_k^{4q} \right) \left( \sum_{j>k}^{\min\{m,n\}} \sigma_j^2 \right), \qquad (4.1)$$

where $k$ is a target truncation term, $q$ is the number of iterations, $\sigma_j$ is the $j$th singular value of $\boldsymbol{A}$, and $\tau_k = \sigma_{k+1}/\sigma_k \ll 1$ is the singular value gap.

**Remark 4.2.** *Instead of $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}_k\|_F^2$, we will use $\|\boldsymbol{A} - \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{A}\|_F^2$. It is straightforward to show the equivalence between the two forms [31, Section 5.3].*

## 4.2 RO-SVD and its Error Analysis

The goal of the RO-SVD method is to find a good approximate O-SVD of tensor $\mathcal{A}$ with less storage and time. There are two stages in producing the approximation, which are summarized in Algorithm 3. The basic RO-SVD method is a specific case of Algorithm 3 that all iteration parameters are chosen to be 0. In the first stage, setting $k_1$ as the first target rank, a full SVD of $\boldsymbol{A}_{(3)}$ is replaced with a randomized SVD to find an orthonormal matrix $\boldsymbol{U}_{k_1}^{(3)}$ such that

$$\widehat{\mathcal{A}} = \mathcal{A} \times_3 \boldsymbol{U}_{k_1}^{(3)T}. \tag{4.2}$$

This allows us to express $\mathcal{A} \approx \widehat{\mathcal{A}} \times_3 \boldsymbol{U}_{k_1}^{(3)}$. Then, the second stage is to connect this low 3-rank tensor $\widehat{\mathcal{A}}$ representation to a randomized tensor SVD, where we apply the randomized SVD to each frontal slice of $\widehat{\mathcal{A}}$ with different target truncation term $k_{2i}$ for $i = 1, 2, \ldots, k_1$. This means that the target multirank is a vector $\boldsymbol{k}_2$ whose elements are $k_{2i}$. For the convenience of notation, we further define an iteration vector as $\boldsymbol{q} = (q_1, q_2, \ldots, q_{k_1})^T$ with employing different iteration count $q_i$ and denote $\boldsymbol{k} = [k_1; \boldsymbol{k}_2]$. Thus we find the tensor $\mathcal{U}_{k_2}$ such that

$$\widehat{\mathcal{A}} \approx \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} = \widehat{\mathcal{A}}_{k_2}. \tag{4.3}$$

Obviously, the procedure to compute the core tensor $\mathcal{S}_{k_2}$ is similar to the algorithm proposed in Algorithm 1. Now, the rank-$\boldsymbol{k}$ representation can be written as

$$\mathcal{A}_{\boldsymbol{k}} = (\mathcal{U}_{k_2} *_3 \mathcal{S}_{k_2} *_3 \mathcal{V}_{k_2}) \times_3 \boldsymbol{U}_{k_1}^{(3)}. \tag{4.4}$$

If $I_3 = 1$, then Algorithm 3 reduces to Algorithm 2.

---

**Algorithm 3** RO-SVD with power iterations

---

**Input:** $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, target truncation vector $\boldsymbol{k} = [k_1; \boldsymbol{k}_2]$, $\boldsymbol{k}_2 = [k_{21}, \ldots k_{2k_1}]^T$, oversampling parameter $p$, the first iteration parameter $q_0$, and the iteration vector $\boldsymbol{q} = (q_1, q_2, \ldots, q_{k_1})$

**Output:** $\boldsymbol{U}_{k_1}^{(3)} \in \mathbb{R}^{I_3 \times k_1}$, $\mathcal{U}_{k_2} \in \mathbb{R}^{I_1 \times k_2 \times k_1}$, $\mathcal{S}_{k_2} \in \mathbb{R}^{k_2 \times k_2 \times k_1}$, $\mathcal{V}_{k_2} \in \mathbb{R}^{k_2 \times I_2 \times k_1}$

1: Generate $k_1 + 1$ Gaussian random matrices $\boldsymbol{\Omega}_1 \in \mathbb{R}^{I_1 I_2 \times (k_1+p)}$ and $\boldsymbol{\Omega}_{2i} \in \mathbb{R}^{I_2 \times (k_{2i}+p)}$;
2: Initialization: $\mathcal{U}_{k_2}$, $\mathcal{S}_{k_2}$, $\mathcal{V}_{k_2}$ are zero tensors of appropriate size;
3: $\left[ \boldsymbol{U}_{k_1}^{(3)}, \boldsymbol{S}_{k_1}^{(3)}, \boldsymbol{V}_{k_1}^{(3)} \right] = \text{rsvd} \left[ \boldsymbol{A}_{(3)}, \boldsymbol{\Omega}_1, k_1, p, q_0 \right]$;
4: **for** $i = 1, 2, \ldots, k_1$, **do**
5:    $\widehat{\boldsymbol{V}}_i = \texttt{reshape}(\boldsymbol{V}_{k_1}^{(3)}(:, i), [I_1, I_2])$;
6:    $[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = \text{rsvd}(\widehat{\boldsymbol{V}}_i, \boldsymbol{\Omega}_{2i}, k_{2i}, p, q_i)$;
7:    $\mathcal{U}_{k_2}(:, 1:k_{2i}, i) = \boldsymbol{U}$, $\mathcal{S}_{k_2}(1:k_{2i}, 1:k_{2i}, i) = \boldsymbol{S}_{k_1}^{(3)}(i,i)\boldsymbol{S}$, $\mathcal{V}_{k_2}(1:k_{2i}, :, i) = \boldsymbol{V}^T$;
8: **end for**

---

We now present the error analysis for Algorithm 3. There are two major difficulties here in extending the proofs of Theorem 3.3. For the probabilistic error analysis, it is important to note that at each step of the cycle, the partially truncated $\widehat{\boldsymbol{V}}_i$ is a random matrix. The elements on the diagonal of $\boldsymbol{S}_{k_1}^{(3)}$ are also derived from the R-SVD. Second, since the orthonormal matrix $\boldsymbol{U}_{k_1}^{(3)}$ here is generated

by a randomized method, it no longer has the property $\boldsymbol{U}^{(3)T}\boldsymbol{U}_{k_1}^{(3)} = \begin{bmatrix} \boldsymbol{I}_{k_1} \\ \boldsymbol{0} \end{bmatrix}$ as $\boldsymbol{U}^{(3)}$ in Theorem 3.1.

As a consequence, using the orthogonal invariance to deal with the Frobenius norm directly does not work in deriving the expected error. In fact, we can provide an expected error bound by splitting the error into two parts.

**Theorem 4.2.** *Let $\mathcal{A}_{\boldsymbol{k}}$ be the output of Algorithm 3 with target truncation parameter $\boldsymbol{k} = [k_1; \boldsymbol{k}_2]$ satisfying $k_1 \leq r_1$, $k_2 \leq r_2$, oversampling parameter $p \geq 2$, iteration count $q_0$, iteration vector $\boldsymbol{q}$ and Gaussian random matrices set $\boldsymbol{\Omega}_2 = \{\boldsymbol{\Omega}_{21}, \ldots, \boldsymbol{\Omega}_{2k_1}\}$. Suppose $\mathcal{U}_{k_2}$, $\boldsymbol{U}_{k_1}^{(3)}$ are obtained from Algorithm 3. Then, the approximation error in expectation satisfies*

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2} \|\mathcal{A} - \mathcal{A}_{\boldsymbol{k}}\|_F &\leq \left[ \left( 1 + \frac{k_1}{p-1} \tau_{k_1}^{4q_0} \right) \left( \sum_{i>k_1}^{r_1} \sum_{j \geq 1}^{r_2} s_{jji}^2 \right) \right]^{\frac{1}{2}} \\
&\quad + \left[ \sum_{i=1}^{k_1} \left( 1 + \frac{k_{2i}}{p-1} \left( \tau_{k_{2i}}^{(i)} \right)^{4q_i} \right) \left( \sum_{j \geq k_{2i}}^{r_2} s_{jji}^2 \right) \right]^{\frac{1}{2}},
\end{aligned}
\tag{4.5}
$$

*where $\tau_{k_1}$ is the singular value gap of $\boldsymbol{A}_{(3)}$ and $\tau_{k_{2i}}^{(i)}$ is the singular value gap of $\widetilde{\mathcal{A}}(:,:,i)$.*

*Proof.* It is straightforward to show that

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2} \|\mathcal{A} - \mathcal{A}_{\boldsymbol{k}}\|_F &= \mathbb{E}_{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2} \left\| \mathcal{A} - \left[ \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \left( \mathcal{A} \times_3 \boldsymbol{U}_{k_1}^{(3)T} \right) \right] \times_3 \boldsymbol{U}_{k_1}^{(3)} \right\|_F \\
&= \mathbb{E}_{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2} \left\| \mathcal{A} - \mathcal{A} \times_3 \boldsymbol{U}_{k_1}^{(3)T} \times_3 \boldsymbol{U}_{k_1}^{(3)} + \mathcal{A} \times_3 \boldsymbol{U}_{k_1}^{(3)T} \times_3 \boldsymbol{U}_{k_1}^{(3)} - \left( \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right) \times_3 \boldsymbol{U}_{k_1}^{(3)} \right\|_F \\
&\leq \mathbb{E}_{\boldsymbol{\Omega}_1} \left\| \mathcal{A} - \mathcal{A} \times_3 \left( \boldsymbol{U}_{k_1}^{(3)} \boldsymbol{U}_{k_1}^{(3)T} \right) \right\|_F + \mathbb{E}_{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2} \left\| \widehat{\mathcal{A}} \times_3 \boldsymbol{U}_{k_1}^{(3)} - \left( \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right) \times_3 \boldsymbol{U}_{k_1}^{(3)} \right\|_F,
\end{aligned}
$$

where we have used the fact that the first part does not depend on the second random matrix $\boldsymbol{\Omega}_2$. We tackle two parts separately.

**Part I** : Using Theorem 4.1 and Hölder's inequality [18, Theorem 23.10], we can write the expected error of the first part directly

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\Omega}_1} \left\| \mathcal{A} - \mathcal{A} \times_3 \left( \boldsymbol{U}_{k_1}^{(3)} \boldsymbol{U}_{k_1}^{(3)T} \right) \right\|_F &\leq \left( \mathbb{E}_{\boldsymbol{\Omega}_1} \left\| \mathcal{A} \times_3 \left( \boldsymbol{I}_{I_3} - \boldsymbol{U}_{k_1}^{(3)} \boldsymbol{U}_{k_1}^{(3)T} \right) \right\|_F^2 \right)^{\frac{1}{2}} \\
&= \left( \mathbb{E}_{\boldsymbol{\Omega}_1} \left\| \left( \boldsymbol{I}_{I_3} - \boldsymbol{U}_{k_1}^{(3)} \boldsymbol{U}_{k_1}^{(3)T} \right) \boldsymbol{A}_{(3)} \right\|_F^2 \right)^{\frac{1}{2}} \\
&\leq \left[ \left( 1 + \frac{k_1}{p-1} \tau_{k_1}^{4q_0} \right) \left( \sum_{i>k_1} \sigma_i^2 \right) \right]^{\frac{1}{2}}.
\end{aligned}
\tag{4.6}
$$

By Theorem 3.2, we can replace $\sigma_i^2$ by $\sum_{j \geq 1}^{r_2} s_{jji}^2$.

**Part II** : As for the second part, since $\boldsymbol{U}_{k_1}^{(3)}$ has orthonormal columns,

$$
\left\| \widehat{\mathcal{A}} \times_3 \boldsymbol{U}_{k_1}^{(3)} - \left( \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right) \times_3 \boldsymbol{U}_{k_1}^{(3)} \right\|_F \leq \left\| \widehat{\mathcal{A}} - \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right\|_F^2.
$$

Then, using Theorem 4.1 (keeping $\boldsymbol{\Omega}_1$ fixed) and the linearity of expectation, we have

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\Omega}_2} \left\| \widehat{\mathcal{A}} - \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right\|_F^2 &= \sum_{i=1}^{k_1} \mathbb{E}_{\boldsymbol{\Omega}_{2i}} \left\| \left( \boldsymbol{I}_{I_1} - \mathcal{U}_{k_2}(:,:,i) \mathcal{U}_{k_2}^T(:,:,i) \right) \widehat{\sigma}_i \widehat{\boldsymbol{V}}_i \right\|_F^2 \\
&\leq \sum_{i=1}^{k_1} \left( 1 + \frac{k_2}{p-1} \left( \tau_{k_{2i}}^{(i)} \right)^{4q_i} \right) \left( \sum_{j>k_{2i}}^{r_2} (\widehat{\sigma}_i \sigma_{ij})^2 \right),
\end{aligned}
$$

10

where $\widehat{\sigma}_i$ is the $i$th singular value of the $\widehat{\boldsymbol{A}}_{(3)}$.

We recall the definition of Löwner partial ordering [17, Section 7.7]. Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ be Hermitian; $\boldsymbol{A} \preceq \boldsymbol{B}$ means $\boldsymbol{B} - \boldsymbol{A}$ is positive semi-definite. Furthermore, $\lambda_i(\boldsymbol{A}) \leq \lambda_i(\boldsymbol{B})$ for $i = 1, 2, \ldots, n$, where $\lambda$ is the eigenvalue of the matrix. Notice $\boldsymbol{U}_{k_1}^{(3)} \boldsymbol{U}_{k_1}^{(3)T}$ is a projector so that

$$\widehat{\boldsymbol{A}}_{(3)}^T \widehat{\boldsymbol{A}}_{(3)} = \left(\boldsymbol{U}_{k_1}^{(3)T} \boldsymbol{A}_{(3)}\right)^T \left(\boldsymbol{U}_{k_1}^{(3)T} \boldsymbol{A}_{(3)}\right) \preceq \boldsymbol{A}_{(3)}^T \boldsymbol{A}_{(3)},$$

and the singular values of $\widehat{\boldsymbol{A}}_{(3)}$ satisfy

$$\widehat{\sigma}_i \leq \sigma_i, \quad \text{for} \quad i = 1, 2, \ldots, k_1.$$

Applying Hölder's inequality gives

$$\mathbb{E}_{\boldsymbol{\Omega}_2} \left\| \widehat{\mathcal{A}} - \mathcal{U}_{k_2} *_3 \mathcal{U}_{k_2}^T *_3 \widehat{\mathcal{A}} \right\|_F \leq \left[ \sum_{i=1}^{k_1} \left( 1 + \frac{k_{2i}}{p-1} \left(\tau_{k_{2i}}^{(i)}\right)^{4q_i} \right) \left( \sum_{j > k_{2i}}^{r_2} (\sigma_i \sigma_{ij})^2 \right) \right]^{\frac{1}{2}}. \tag{4.7}$$

Combining (4.6), (4.7) and $s_{jji} = \sigma_i \sigma_{ij}$ gives the conclusion. $\qquad\square$

## 4.3 Computational Complexity and Memory Cost

We now discuss the computational cost of Algorithm 1 and Algorithm 3 , and compare them against the O-SVD, the RT-SVD proposed by Zhang *et al.* [41, Algorithm 6] and the R-HOSVD proposed by Minster *et al.* [26, Algorithm 3.1]. We assume that the tensors are dense and the target truncation terms $k_1$ and $k_2$ are sufficiently small, i.e., $k_1 \ll r_1$, $k_2 \ll r_2$, so that we can neglect the computational cost of the QR factorization and the truncation steps of the R-SVD algorithm. The dominant cost of Algorithm 3 lies in computing a total of $k_1 + 1$ R-SVD, while the TO-SVD algorithm requires to compute the full SVD, which results in an expensive computational cost.

Recall that the RT-SVD algorithm consists of transforming the tensor to the Fourier domain and applying the R-SVD to each frontal slice of the tensor. The R-HOSVD algorithm has three main steps including multiplying each mode unfolding with a Gaussian random matrix, computing an approximation to the column space and then forming the core tensor. The storage and computational cost of the TO-SVD, RO-SVD, RT-SVD and RHOSVD algorithms are summarized in Table 1. Each algorithm takes a core tensor $\mathcal{S}$ of the same size except the RT-SVD. The table includes the costs for a general third-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with the first iteration parameter $q_0$ and the iteration vector $\boldsymbol{q} = (q_1, q_2, \ldots q_{k_1})$ ($k_1 = I_3$ for the RT-SVD, target rank $\boldsymbol{k}_3 = (k_2, k_2, k_1)$ for the R-HOSVD). For a more intuitive comparison of the computational cost we assume that $k_{21} = \cdots = k_{2k_1} = k_2$. Since by assumption $k_1 \ll r_1$, $k_2 \ll r_2$, the RO-SVD is expected to be much faster than all four other algorithms.

Table 1: Comparison of O-SVD, TO-SVD, RO-SVD, RT-SVD, R-HOSVD

| Algorithm | Computational Cost | Storage Cost |
|---|---|---|
| O-SVD | $\mathcal{O}\left(I_1 I_2 I_3^2 + R_3 I_1 I_2 I_3 + R_3 I_1 I_2^2\right)$ | $R_3(I_2 I_1 + I_2 I_2 + I_3 + I_2)$ |
| TO-SVD | $\mathcal{O}\left(I_1 I_2 I_3^2 + k_1 I_1 I_2^2\right)$ | $k_1 k_2 I_1 + k_1 k_2 I_2 + k_1 I_3 + k_1 k_2$ |
| RO-SVD | $\mathcal{O}\left((2q_0 + 1)k_1 I_1 I_2 I_3 + \sum_{i=1}^{k_1}(2q_i + 1)k_2 I_1 I_2\right)$ | $k_1 k_2 I_1 + k_1 k_2 I_2 + k_1 I_3 + k_1 k_2$ |
| RT-SVD | $\mathcal{O}\left(I_1 I_2 I_3 \log I_3 + \sum_{i=1}^{I_3}(2q_i + 1)k_2 I_1 I_2\right)$ | $k_2(I_1 I_3 + I_2 I_3 + I_3)$ |
| R-HOSVD | $\mathcal{O}(\sum_{i=1}^{3}(2q_i + 1)k_{3i} I_1 I_2 I_3 + k_1 I_1 I_2 + k_1 k_2 I_2 + k_1 k_2^2)$ | $k_1 k_2^2 + k_2(I_1 + I_2) + I_3 k_1$ |

# 5 Numerical Examples

In order to evaluate Algorithm 1 and Algorithm 3, we present the numerical results by comparing them with truncation methods such as TT-SVD, THOSVD and the above mentioned randomized algorithm of tensor singular value decomposition: RT-SVD, RHOSVD. All the computations are based on the Matlab Tensor Toolbox [1] and Tensor-Tensor Product Toolbox [23]. Our results were run in Matlab R2020b on a Lenovo computer with AMD Ryzen 5 3500U processor and 12 GB RAM. We use the HOSVD algorithm to show that the tensor is orientation-dependent. To the best of our knowledge, there is no specific approach for the selection of all truncation parameters in fixed-rank random tensor algorithms. The oriented tensors have high correlation among frontal slices and we can estimate the rank of the orthonormal matrix in the RO-SVD in advance. We could set $k_1$ much smaller than $I_3$. The optimal value of $k_1$ is the number of a basis in the space spanned by all frontal slices of the third-order oriented tensor. Since both the TT-SVD and the RT-SVD use the same truncation parameters $k_2$ for each frontal slice, we also use the same truncation parameters $k_{21} = \cdots = k_{2k_1} = k_2$ for TO-SVD and RO-SVD in the experiments. In addition, the computational time of each method was measured in seconds. The results of each experiment were averaged three times.

We employ four indices, i.e., the relative error, the compression ratio, the peak-signal-to-noise ratio (PSNR) and the structural similarity index (SSIM), to evaluate the performance of image compression algorithm. If $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is the original tensor and $\widehat{\mathcal{A}}$ is a low-rank approximation of $\mathcal{A}$, then the relative approximation error and the compression ratio are respectively given by

$$\mathrm{Err} = \frac{\left\|\mathcal{A} - \widehat{\mathcal{A}}\right\|_F}{\|\mathcal{A}\|_F} \text{ and Ratio} = \frac{I_1 \times I_2 \times I_3}{\text{storage cost}}.$$

The PSNR is given by:

$$\mathrm{PSNR} = 20 \log_{10} \frac{\max(\mathcal{A})}{\sqrt{\mathrm{MSE}}},$$

where the MSE of tensor $\mathcal{A}$ is as follows:

$$\mathrm{MSE} = \frac{\|\mathcal{A} - \hat{\mathcal{A}}\|_F^2}{I_1 \times I_2 \times I_3}.$$

According to research, a PSNR value above 40 for the pixel component of an image is an indication of very good quality (i.e., the restored frame is very close to the original frame). If the PSNR is between 30 and 40, then the image quality is usually good (i.e., the distortion in the restored image is noticeable but still acceptable). If it is between 20 and 30 then the image quality is poor, and finally, images with a PSNR below 20 are not acceptable. SSIM [37] measures the similarity between the original image and the reconstructed image on structural consistency. The equation of SSIM is below:

$$\mathrm{SSIM}(a, \hat{a}) = \frac{(2\mu_a \mu_{\hat{a}} + C_1)(2\sigma_{a\hat{a}} + C_2)}{\left(\mu_a^2 + \mu_{\hat{a}}^2 + C_1\right)\left(\sigma_a^2 + \sigma_{\hat{a}}^2 + C_2\right)},$$

where $\mu_a$ and $\mu_{\hat{a}}$ are mean intensities, $\sigma_a$ and $\sigma_{\hat{a}}$ are standard deviations, $C_1$ and $C_2$ are default values. Covariance $\sigma_{a\hat{a}}$ is calculated as follows:

$$\sigma_{a\hat{a}} = \frac{1}{I_1 I_2 I_3 - 1} \sum_{ijk} \left(a_{ijk} - \mu_a\right)\left(\hat{a}_{ijk} - \mu_{\hat{a}}\right).$$

Obviously, the SSIM is a number between 0 and 1. The larger the SSIM, the smaller the difference between the two images.

## 5.1 Hyperspectral Image

In this subsection, we test a hyperspectral image— Salinas [14]. This scene was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The area covered comprises 512 lines by 217 samples 224 available spectral reflectance bands in the wavelength range. Hence, the size of the resulting tensor is $512 \times 217 \times 224$. Denote by $\mathcal{A}$ the tensor of the testing data. Under the relative error tolerance 0.005, (Matlab command: hosvd $(\mathcal{A}, 0.005)$), the size of the nonzero part of the core tensor is $423 \times 203 \times 32$. Hence, this is a well-oriented tensor. And the running time of a full O-SVD is 6.2432s.

We take the first iteration parameter $q_0 = 1$, the target truncation term $k_2$ fixed at 80, the oversampling parameter $p = 5$ and the second iteration parameters $q_i = 1$. The relative errors for varying $k_1$ between 5 and 40 are plotted in Figure 1, where we can see the errors of the RO-SVD are quite close to the TO-SVD. Since $\mathcal{A}$ is a well-oriented tensor, the errors level off after $k_1 = 30$. The errors of TO-SVD and TT-SVD are very close, indicating that it is possible to compress the third dimension of $\mathcal{A}$ to $k_1$ without being affected. The error of THOSVD is larger for the same size of core tensor.

Then, we choose $k_1$ to be 35 and allow $k_2$ to vary between 30 and 100 for simplicity. We track the relative errors of the RO-SVD for the case that the iteration parameters $q_i$ $(i = 1, 2, \ldots, k_1)$ are equal, i.e., $q_1 = \cdots = q_{k_1} = q$. Figure 2 shows that the errors for varying $q$ have similar convergence trajectories and $q = 1$ is enough for practical use. Moreover, as $q$ increases, a much more accurate approximation is yielded. In addition, we give error curves for TT-SVD, THOSVD and their corresponding randomized algorithms with $q = 1$. It is shown that under the same conditions, the tensor singular value decomposition based on the Tucker product has a much larger error than the other two tensor decompositions, both for the exact and the randomized algorithms.

Figure 3 shows the time, relative error, PSNR and SSIM of the three randomized algorithms at different compression ratios. We choose $k_1$ to be 35, the oversampling parameter $p = 5$, all the iteration parameters $q = 1$ and $k_2$ varies with the compression ratio. Figure 3 shows that the RO-SVD outperforms the RT-SVD in all four indices. In Table 2, we record the time, relative error, PSNR and SSIM of four algorithms (TO-SVD, RO-SVD, RT-SVD, and RHOSVD) at $Ratio = 40$. From Figure 3 and Table 2, we find that the proposed randomized algorithm has similar results to the exact algorithm, illustrating the effectiveness of the RO-SVD. As for the running time, the RO-SVD is usually much faster than TO-SVD and O-SVD respectively. Compared with other randomized algorithms, the PSNR and SSIM of RHOSVD and RO-SVD are within the acceptable range, but the former about 6 times longer than the latter when $Ratio = 40$. As both the time and relative error of the RO-SVD are reduced by half compared to the RT-SVD, the RO-SVD is superior to the RT-SVD at the same compression ratio. This gives us a suggestion for choosing $k_2$: in addition to setting the value of $k_2$ directly, we can also set the parameter $k_2$ according to the desired compression ratio.

Table 2: Comparison of four algorithms with $Ratio = 40$

| Algorithm | time | Err | PSNR | SSIM |
|-----------|--------|--------|---------|--------|
| RT-SVD | 1.6819 | 0.1926 | 29.7846 | 0.9994 |
| RHOSVD | 5.1130 | 0.0322 | 45.3095 | 0.9999 |
| TOSVD | 2.6429 | 0.0737 | 38.1286 | 0.9999 |
| RO-SVD | 0.8886 | 0.0758 | 37.8799 | 0.9999 |

Figure 1: Relative errors of four algorithms with $k_2 = 80$



Figure 2: Relative errors of four algorithms with $k_1 = 35$



(a) Time



(b) Relative errors



(c) PSNR



(d) SSIM

Figure 3: Comparison results of four algorithms (TO-SVD, RO-SVD, RT-SVD, and RHOSVD) on Salinas.

(a) Time

(b) Relative errors

(c) PSNR

(d) SSIM

Figure 4: Comparison results of three tensor randomized methods (RO-SVD, RT-SVD, and RHOSVD) on the video.

Table 3: Comparison of RT-SVD , RHOSVD and RO-SVD with $k_1 = 70$ , $k_2 = 90$

| Algorithm | time | Err | PSNR | SSIM |
|-----------|---------|--------|---------|--------|
| RT-SVD | 49.1988 | 0.0139 | 45.5254 | 0.9993 |
| RHOSVD | 23.8538 | 0.0424 | 36.5753 | 0.9953 |
| RO-SVD | 8.9908 | 0.0144 | 45.0915 | 0.9993 |

(a) Original video



(b) RT-SVD



(c) RHOSVD



(d) RO-SVD

Figure 5: Comparison results of three tensor randomized methods (RO-SVD, RT-SVD, and RHOSVD) with the original image

## 5.2 Video

The second example is a video. The dataset $\mathcal{B}$ is a video from Tencent[1], with size $424 \times 726 \times 500$ (height$\times$width$\times$frames). See Figure 5(a). Most regions of the frames are stable, while a radish is growing delicately. Under the relative error tolerance 0.01, the size of the nonzero part of the core tensor is $198 \times 351 \times 71$, which is a well-oriented tensor. And the running time of a full O-SVD is 58.6212s.

We run three tensor randomized methods (RO-SVD, RT-SVD, and RHOSVD) with increasing the same target truncation term $k_2$ and oversampling parameter $p = 5$ while setting the target rank $k_1 = 70$. Here we only show the results with the power parameter $q = 1$, as for many applications this already achieves sufficient accuracy. We compare the time, relative error, PSNR and SSIM of the RT-SVD, the RHOSVD and the RO-SVD algorithms with different target truncation term $k_2$, which are shown in Figure 4. In Table 3, we record the results of three algorithms at $k_2 = 90$ from Figure 4. From the results in Figure 4(a) and (b), we see that the RO-SVD algorithms is far superior to them in terms of time cost while keeping the accuracy and its advantage becomes more apparent with the increase of $k_2$. Both the RT-SVD and the RO-SVD algorithm perform better than RHOSVD, as shown in Figure 4(c) and (d). In Table 3, we can see the RO-SVD is about 6 times faster than the implementation of RT-SVD, and the compression ratio of the former is 7.1 times (21.12 vs. 2.97) better than that of the latter. Figures 5(b), 5(c) and 5(d) show the results of the RT-SVD, the RHOSVD and the RO-SVD respectively when the target truncation term $k_2 = 90$. We can see the performance of the RO-SVD is the best with the same target truncation term $k_2$ while it has significantly lower computational cost.

## 5.3 Synthetic Oriented Tensor

For further reflection on the characteristics of the RO-SVD for large-scale oriented tensors, two kinds of synthetic tensors are tested standing for different distribution patterns of singular values:
• Tensor 1 (slow decay): $\mathcal{A} = (\mathcal{U} *_3 \mathcal{S} *_3 \mathcal{V}) \times_3 \boldsymbol{U}^{(3)}$, where $\boldsymbol{U}^{(3)}$, $\mathcal{U}(:,:,i)$ and $\mathcal{V}(:,:,i)^T$ are randomly drawn matrices with orthonormal columns, and the diagonal matrix $\mathcal{S}(:,:,i)$ has diagonal elements $\sigma_{jji} = 1/(i+j)^2$. Furthermore, if $\boldsymbol{U}^{(3)}$ is a matrix with far fewer columns than rows, the resulting tensor $\mathcal{A}$ has to be a well-oriented tensor.
• Tensor 2 (fast decay): $\mathcal{A}$ is formed just like Tensor 1, but the diagonal elements of $\mathcal{S}(:,:,i)$ are given by $\sigma_{jji} = e^{-j-i/7}$. It reflects a fast decay of singular values.

For each kind, we first generate a $1000 \times 1000 \times 300$ oriented tensor with $\mathtt{rank}_3(\mathcal{A}) = 30$, for which we compare the relative errors and time of the proposed techniques for different $\boldsymbol{k}_2$ and $q$. Let $k_1 = 30$, $p = 5$ and $k_{2i}$ be randomly generated positive integers in $[0, 20], [20, 40], \ldots, [180, 200]$, respectively. The results are plotted in Figure 6. We observe that the RO-SVD runs at approximately three times the speed of the TO-SVD within an acceptable error margin. By the power scheme, the errors of the RO-SVD can be remarkably reduced. We notice that $q = 1$ suffices in practice since it produces indistinguishable results with $q = 2$.

Then, we compare the performance of the RT-SVD, RHOSVD and RO-SVD algorithms with equal $k_{2i}$, oversampling parameter $p = 5$ and power parameter $q = 1$ while setting the target rank $k_1 = 30$, which are shown in Figure 7. It is obvious that the RO-SVD shows advantages in both accuracy and time cost. Specifically, our algorithm takes roughly one-sixth of the time required for the other two fixed-rank randomized algorithms. Moreover, the RO-SVD achieves better compression ratio compared to the RT-SVD with $k_{2i} = 80$ (62.35 vs 6.24).

Finally, we conduct an experiment to compare the efficiency of the proposed algorithms with

---

[1]https://v.qq.com/x/page/v3237ztwzs7.html

varying sizes. We generate oriented tensors with $\mathrm{rank}_3(\mathcal{A}) = I_3/10$ and summarize the running time and relative errors of the five algorithms for different dimensions in Table 4. The RO-SVD shows overwhelming advantages in computational efficiency. Specially, when the tensor size comes to $1000 \times 1000 \times 400$, all the other existing algorithms lose competitiveness because of timeouts and memory limits.

Table 4: Comparison of five algorithms for different $\mathcal{A}$
with $k_1 = I_3/10$, $k_2 = 200$, $q = 1$ and $p = 5$

| Algorithm | O-SVD | TO-SVD | | RO-SVD | | RT-SVD | | RHOSVD | |
|---|---|---|---|---|---|---|---|---|---|
| $I_1 \times I_2 \times I_3$ | time | time | Err | time | Err | time | Err | time | Err |
| $1000 \times 1000 \times 100$ | 11.66 | 9.60 | 0.0086 | 3.17 | 0.0092 | 45.25 | 0.0242 | 16.24 | 0.0441 |
| $1500 \times 1500 \times 100$ | 31.48 | 26.47 | 0.0108 | 7.45 | 0.0115 | 102.04 | 0.0265 | 41.08 | 0.0452 |
| $2000 \times 2000 \times 100$ | — | 67.67 | 0.0105 | 27.89 | 0.0111 | 168.73 | 0.0271 | 95.35 | 0.0456 |
| $1000 \times 1000 \times 300$ | 63.44 | 53.80 | 0.0125 | 23.98 | 0.0133 | 142.16 | 0.0337 | 75.33 | 0.0923 |
| $1000 \times 1000 \times 400$ | — | — | — | 30.33 | 0.0146 | — | — | — | — |

# 6    Conclusion

The contributions of this paper are twofold: we revisit the O-SVD and refine its process from the viewpoint of the TTr1SVD and a truncated version for the O-SVD is given. Based on recent results on the randomized SVD template, we design a randomized algorithm for the O-SVD for third-order oriented tensors. In addition, we give the corresponding probabilistic error analysis of the RO-SVD. The performance of the RO-SVD is better than the RT-SVD for the same compression ratio and better than the RHOSVD for the same size core tensor and shows great advantages in time cost if the tensor is well-oriented.

In the future, we will continue a further study of the adaptive randomized approach for the case where the target rank is unknown or cannot be estimated in advance. One more potential research direction is the use of updated SVD algorithms [3] to study dynamic video streaming.

# Acknowledgments

# References

[1] B. W. BADER, G. K. TAMARA, ET AL., *Matlab Tensor Toolbox, version 3.2.1.* `https://www.tensortoolbox.org`, April 2017.

[2] K. BATSELIER, H. LIU, AND N. WONG, *A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1315–1337.

[3] M. BRAND, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra Appl., 415 (2006), pp. 20–30.

(a) Time of Tensor 1

(b) Relative errors of Tensor 1

(c) Time of Tensor 2

(d) Relative errors of Tensor 2

Figure 6: Comparison results of the TO-SVD and RO-SVD on the test tensors ($k_1 = 30$).

(a) Time of Tensor 1



(b) Relative errors of Tensor 1



(c) Time of Tensor 2



(d) Relative errors of Tensor 2

Figure 7: Comparison results of three tensor randomized methods (RO-SVD, RT-SVD, and RHOSVD) on the test tensors ($k_1 = 30$).

[4] J. D. CARROLL AND J. J. CHANG, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[5] M. CHE, J. CHEN, AND Y. WEI, *Perturbations of the TCUR decomposition for tensor valued data in the Tucker format*, J. Optim. Theory Appl., 194 (2022), pp. 852–877.

[6] M. CHE AND Y. WEI, *Randomized algorithms for the approximations of Tucker and the tensor train decompositions*, Adv. Comput. Math., 45 (2019), pp. 395–428.

[7] M. CHE, Y. WEI, AND H. YAN, *The computation of low multilinear rank approximations of tensors via power scheme and random projection*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 605–636.

[8] M. CHE, Y. WEI, AND H. YAN, *An efficient randomized algorithm for computing the approximate Tucker decomposition*, J. Sci. Comput., 88 (2021). Paper No. 32.

[9] M. CHE, Y. WEI, AND H. YAN, *Randomized algorithms for the low multilinear rank approximations of tensors*, J. Comput. Appl. Math., 390 (2021). Paper No. 113380.

[10] A. CICHOCKI, D. MANDIC, L. DE LATHAUWER, G. ZHOU, Q. ZHAO, C. CAIAFA, AND H. A. PHAN, *Tensor decompositions for signal processing applications: From two-way to multiway component analysis*, IEEE Signal Process Mag., 32 (2015), pp. 145–163.

[11] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[12] P. DRINEAS AND M. W. MAHONEY, *RandNLA: Randomized numerical linear algebra*, Commun. ACM, 59 (2016), pp. 80–90.

[13] C. ECKART AND G. YOUNG, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[14] V. EISAVI, *Hyperspectral Remote Sensing Scenes - gic php.* http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes, March 2017.

[15] N. HALKO, P. G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[16] R. A. HARSHMAN, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-mode factor analysis*, UCLA Working Papers in phonetics., 16 (1969), pp. 1–84.

[17] R. A. HORN AND C. R. JOHNSON, *Matrix Analysis*, Cambridge University Press, Cambridge, Second ed., 2013.

[18] J. JACOD AND P. PROTTER, *Probability Essentials*, Springer, Berlin, 2012.

[19] M. E. KILMER, K. BRAMAN, N. HAO, AND R. C. HOOVER, *Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 148–172.

[20] M. E. KILMER AND C. D. MARTIN, *Factorization strategies for third-order tensors*, Linear Algebra Appl., 435 (2011), pp. 641–658.

[21] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[22] T. G. KOLDA, B. W. BADER, AND J. P. KENNY, *Higher-order web link analysis using multilinear algebra*, in Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM '05, USA, 2005, IEEE Computer Society, pp. 242—-249.

[23] C. LU, *Tensor-Tensor Product Toolbox.* https://github.com/canyilu/tproduct, June 2018.

[24] M. W. MAHONEY, *Randomized algorithms for matrices and data*, Foundations and Trends in Machine Learning, 3 (2011), pp. 123–224.

[25] P.-G. MARTINSSON, V. ROKHLIN, AND M. TYGERT, *A randomized algorithm for the decomposition of matrices*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 47–68.

[26] R. MINSTER, A. K. SAIBABA, AND M. E. KILMER, *Randomized algorithms for low-rank tensor decompositions in the Tucker format*, SIAM J. Math. Data Sci., 2 (2020), pp. 189–215.

[27] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[28] L. Qi, H. Chen, and Y. Chen, *Tensor Eigenvalues and Their Applications*, vol. 39 of Advances in Mechanics and Mathematics, Springer, Singapore, 2018.

[29] L. Qi and Z. Luo, *Tensor Analysis: Spectral Theory and Special Tensors*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[30] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124.

[31] A. K. Saibaba, *Randomized subspace iteration: analysis of canonical angles and unitarily invariant norms*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 23–48.

[32] G. Shabat, Y. Shmueli, Y. Aizenbud, and A. Averbuch, *Randomized LU decomposition*, Appl. Comput. Harmon. Anal., 44 (2018), pp. 246–272.

[33] M. Signoretto, Q. Tran Dinh, L. De Lathauwer, and J. A. K. Suykens, *Learning with tensors: A framework based on convex optimization and spectral regularization*, Mach. Learn., 94 (2014), pp. 303–351.

[34] L. Sorber, M. Van Barel, and L. De Lathauwer, *Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization*, SIAM J. Optimiz., 23 (2013), pp. 695–720.

[35] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[36] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A new truncation strategy for the higher-order singular value decomposition*, SIAM J. Sci. Comput., 34 (2012), pp. A1027–A1052.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Trans. Image Proc., 13 (2004), pp. 600–612.

[38] Y. Wei, P. Xie, and L. Zhang, *Tikhonov regularization and randomized GSVD*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 649–675.

[39] P. Xie, H. Xiang, and Y. Wei, *Randomized algorithms for total least squares problems*, Numer. Linear Algebra Appl., 26 (2019). e2219.

[40] C. Zeng and M. K. Ng, *Decompositions of third-order tensors: HOSVD, T-SVD, and beyond*, Numer. Linear Algebra Appl., 27 (2020). e2290.

[41] J. Zhang, A. K. Saibaba, M. E. Kilmer, and S. Aeron, *A randomized tensor singular value decomposition based on the t-product*, Numer. Linear Algebra Appl., 25 (2018). e2179.