Regular Article

Three-Way Tensor Decompositions: A Generalized Minimum Noise Subspace Based Approach

Le Trung Thanh[1], **Viet-Dung Nguyen**[2], **Nguyen Linh-Trung**[1], **Karim Abed-Meraim**[3]

[1] AVITECH, VNU University of Engineering and Technology, Hanoi, Vietnam
[2] Signals and Systems Laboratory, University of Paris-Saclay, CentraleSupelec, France
[3] PRISME Laboratory, University of Orléans, Orléans, France

Correspondence: Nguyen Linh-Trung, linhtrung@vnu.edu.vn
Communication: received 8 February 2018, revised 15 April 2018, accepted 18 May 2018
Online publication: 15 June 2018, Digital Object Identifier: 10.21553/rev-jec.196
The associate editor coordinating the review of this article and recommending it for publication was Prof. Son Lam Phung.

Abstract– **Tensor decomposition has recently become a popular method of multi-dimensional data analysis in various applications. The main interest in tensor decomposition is for dimensionality reduction, approximation or subspace purposes. However, the emergence of "big data" now gives rise to increased computational complexity for performing tensor decomposition. In this paper, motivated by the advantages of the generalized minimum noise subspace (GMNS) method, recently proposed for array processing, we proposed two algorithms for principal subspace analysis (PSA) and two algorithms for tensor decomposition using parallel factor analysis (PARAFAC) and higher-order singular value decomposition (HOSVD). The proposed decomposition algorithms can preserve several desired properties of PARAFAC and HOSVD while substantially reducing the computational complexity. Performance comparisons of PSA and tensor decomposition of our proposed algorithms against the state-of-the-art ones were studied via numerical experiments. Experimental results indicated that the proposed algorithms are of practical values.**

Keywords– **Generalized minimum noise subspace (GMNS), principal subspace analysis (PSA), Tensor decomposition, parallel factor analysis (PARAFAC), Tucker decomposition, high-order singular value decomposition (HOSVD).**

1 Introduction

Over the last two decades, a number of large-scale datasets have been increasingly collected in various fields and can be smartly mined to discover new valuable information, helping us to have deeper understanding of the hidden values [1]. Various examples are seen in physical, biological, social, health and engineering science applications, wherein large-scale multi-dimensional, multi-relational and multi-model data are generated. Therefore, data analysis techniques using tensor decomposition now attract a great deal of attention from researchers and engineers.

A tensor is a multi-dimensional array and often considered as a generalization of a matrix. As a result, tensor representation gives a natural description of multi-dimensional data and hence tensor decomposition becomes a useful tool to analyze high-dimensional data. Moreover, tensor decomposition brings new opportunities for revealing hidden and new values in the data. As a result, tensor decomposition has been used in various applications. For example, in neuroscience, brain signals are inherently multi-way data in general, and spatio-temporal in particular, due to the fact that they can be monitored through different brain regions at different times. In particular, an electroencephalography (EEG) dataset can be represented by a three-way tensor with three dimensions of time, frequency and electrode, or even by multi-way tensors when extra dimensions such as condition, subject and group are also considered. Tensor decomposition can be used to detect abnormal brain activities such as epileptic seizures [2], to extract features of Alzheimer's disease [3] or other EEG applications, as reviewed in [4].

1.1 Tensor Decompositions

Two widely used decompositions for tensors are parallel factor analysis (PARAFAC) (also referred to as canonical polyadic decomposition) and Tucker decomposition. PARAFAC decomposes a given tensor into a sum of rank-1 tensors. Tucker decomposition decomposes a given tensor into a core tensor associated with a set of matrices (called factors) which are used to multiply along each mode (way to model a tensor along a particular dimension).

In the literature of tensors, many algorithms have been proposed for tensor decomposition. We can categorize them into three main approaches, respectively, based on: divide-and-conquer, compression, and optimization. The first approach aims to divide a given tensor into a number of sub-tensors, then estimate the factors of the sub-tensors and finally combine them together into the true factors. The central idea behind the second approach is to reduce the size of a given tensor until it becomes manageable before computing a specific decomposition of the compressed tensor, which

retains the main information of the original tensor. In the third approach, tensor decomposition is cast into optimization and is then solved using standard optimization tools. We refer the reader to important surveys in [5–7] for further details on the different approaches.

1.2 Objectives

In this paper, we focus on the divide-and-conquer approach for PARAFAC and Tucker decompositions of three-way tensors. With respect to the later, we focus on a specific orthonormal form which is called high-order singular value decomposition (HOSVD). Examples of three-way tensors are numerous. Image-row $\times$ image-column $\times$ time tensors are used in video surveillance, human action recognition and real-time tracking [8–10]. Spatial-row $\times$ spatial-column $\times$ wavelength tensors are used for target detection and classification in hyperspectral image applications [11, 12]. Origin $\times$ destination $\times$ time tensors are used in transportation networks to discover the spatio-temporal traffic structure [13]. Time $\times$ frequency $\times$ electrode tensors are used in EEG analysis [2].

Recently, generalized minimum noise subspace (GMNS) was proposed by Nguyen *et al.* in [14] as a good technique for subspace analysis. This method is highly beneficial in practice because it not only substantially reduces the computational complexity in finding bases for these subspaces, but also provides high estimation accuracy. Several efficient GMNS-based algorithms for principal subspace analysis (PSA), minor subspace analysis (MSA), and principal component analysis (PCA) were proposed and shown to be applicable in various applications. This motivates us to propose in this paper new implementations for tensor decomposition based on GMNS.

1.3 Contributions

The main contributions of this paper are summarized as follows. First, by expressing the *right* singular vectors obtained from singular value decomposition (SVD) in terms of principal subspace, we derive a *modified* GMNS algorithm for PSA with running time faster than the original GMNS, while still retaining the subspace estimation accuracy.

Second, we introduce a *randomized* GMNS algorithm for PSA that can deal with several matrices by performing the randomized SVD.

Third, we propose two GMNS-based algorithms for PARAFAC and HOSVD. These algorithms are highly beneficial and easily implemented in practice, thanks to its parallelized scheme with a low computational complexity. Several applications are studied to illustrate the effectiveness of the proposed algorithms.

1.4 Paper structure

The structure of the paper is organized as follows. Section 2 provides some background for our study, including two kinds of algorithms for PSA and tensor decomposition. Section 3 presents modified and randomized GMNS algorithms for PSA. Sections 4 and 5 respectively present the GMNS-based algorithms for PARAFAC and HOSVD, respectively. Section 6 carries out simulated experiments to study the effectiveness and performance of the proposed algorithms, in comparison with several state-of-the-art algorithms. Section 7 concludes the paper.

1.5 Notations

We use bold calligraphic letters to denote tensors (*e.g.*, $\boldsymbol{\mathcal{A}}$), meanwhile boldface capital letters are used for matrices (*e.g.*, $\mathbf{A}$). Vectors and scalars are denoted by boldface lowercase letters (*e.g.*, $\mathbf{a}$), and lowercase letters (*e.g.*, a) respectively. For operators on tensors, $\boldsymbol{\mathcal{A}}^{(n)}$ denotes the mode-n unfolding (fiber) of $\boldsymbol{\mathcal{A}}$ and the n-mode product of the tensor $\boldsymbol{\mathcal{A}}$ with a matrix $\mathbf{U}$ is denoted by $\boldsymbol{\mathcal{A}} \times_n \mathbf{U}$. For matrix products, $\otimes$, $\odot$ and $*$ denotes the Kronecker product, the Khatri-Rao product and the Hadamard product respectively. The $\mathbf{A}^T$, $\mathbf{A}^*$, $\mathbf{A}^H$ and $\mathbf{A}^\#$ denote the transpose, the complex conjugate, the complex conjugate transpose and the pseudo-inverse of $\mathbf{A}$ respectively. For vector's operators, $\mathbf{a} \circ \mathbf{b}$ denotes the outer product of $\mathbf{a}$ and $\mathbf{b}$. Also, $\|\cdot\|$ denotes the Frobenius norm of a vector, matrix and tensor.

2 Preliminaries

Before showing how GMNS can be used for tensor decomposition, it is of interest to first explain the central idea of the method. In addition, a divide-and-conquer algorithm for PARAFAC called alternating least-square (ALS) is also provided.

2.1 Principal Subspace Analysis based on GMNS

Consider a low rank matrix $\mathbf{X} = \mathbf{AS}$ under the conditions that $\mathbf{A} \in \mathbb{C}^{n\times p}, \mathbf{S} \in \mathbb{C}^{p\times m}$ with $p < \min(n, m)$, and $\mathbf{A} \in \mathbb{C}^{n\times m}$ is full column rank.

Under the constraint of having only a *fixed* number k of digital signal processing (DSP) units, the procedure of GMNS for PSA includes: dividing the matrix $\mathbf{X}$ into k sub-matrices $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k\}$, then estimating each principal subspace matrix $\mathbf{W}_i = \mathbf{A}_i\mathbf{Q}_i$ of $\mathbf{X}_i$, and finally combining them to obtain the overall principal subspace matrix of $\mathbf{X}$. Clearly, we should choose the number of DSP units so that the size of resulting sub-matrices $\mathbf{X}_i$ must be larger than rank of $\mathbf{X}$, $p \leq n/k$. The algorithm was proposed in [14] and summarized in Algorithm 1.

First, the principal subspace matrix $\mathbf{W}_i$ of $\mathbf{X}_i$ can be obtained from the eigenspace of its corresponding covariance matrix,

$$\mathbf{R}_{\mathbf{X}_i} = \mathrm{E}\{\mathbf{X}_i\mathbf{X}_i^H\} = \mathbf{A}_i\mathbf{R}_\mathbf{S}\mathbf{A}_i^H \overset{\mathrm{EVD}}{=} \mathbf{W}_i\boldsymbol{\Lambda}\mathbf{W}_i^H, \qquad (1)$$

where $\mathbf{W}_i = \mathbf{A}_i\mathbf{Q}_i$, with $\mathbf{Q}_i \in \mathbf{R}^{p\times p}$, is an unknown full rank matrix.

Algorithm 1: GMNS-based PSA [14]

Input: Matrix $\mathbf{X} \in \mathbb{C}^{n\times m}$, target rank p, k DSP units
Output: Principal subspace matrix $\mathbf{W}_\mathbf{X} \in \mathbb{R}^{n\times p}$ of $\mathbf{X}$

initilization
  Divide $\mathbf{X}$ into k sub-matrices $\mathbf{X}_i$
  Form covariance matrix $\mathbf{R}_{\mathbf{X}_1} = \frac{1}{m}\mathbf{X}_1\mathbf{X}_1^H$
  Extract principal subspace $\mathbf{W}_1 = \text{eig}(\mathbf{R}_{\mathbf{X}_1}, p)$
  Construct matrix $\mathbf{U}_1 = \mathbf{W}_1^{\#}\mathbf{X}_1$
main estimate PSA : *// updates can be done in parallel*
  for $i = 2 \to k$ **do**
    Form covariance matrix $\mathbf{R}_{\mathbf{X}_i} = \frac{1}{m}\mathbf{X}_i\mathbf{X}_i^H$
    Extract principal subspace $\mathbf{W}_i = \text{eig}(\mathbf{R}_{\mathbf{X}_i}, p)$
    Construct matrix $\mathbf{U}_i = \mathbf{W}_i^{\#}\mathbf{X}_i$
    Construct rotation $\mathbf{T}_i = \mathbf{U}_i\mathbf{U}_1^{\#}$
    Update $\mathbf{W}_i \leftarrow \mathbf{W}_i\mathbf{T}_i$
return $\mathbf{W}_\mathbf{X} = [\mathbf{W}_1^T\ \mathbf{W}_2^T\ \dots\ \mathbf{W}_k^T]^T$

Given the directions of $\mathbf{X}_1$, we look for $(k-1)$ rotation matrices $\mathbf{T}_i$ to align the principal axes of each $\mathbf{X}_i$ with these directions of $\mathbf{X}_1$. Specifically, let

$$\mathbf{U}_i = \mathbf{W}_i^{\#}\mathbf{X}_i, \tag{2}$$

then

$$\mathbf{U}_i = (\mathbf{A}_i\mathbf{Q}_i)^{\#}\mathbf{A}_i\mathbf{S} = \mathbf{Q}_i^{-1}\mathbf{S}. \tag{3}$$

On the other hand, combining with (1), the signal subspace can be written as

$$\mathbf{W} = \mathbf{A}\mathbf{Q} = \begin{bmatrix}\mathbf{A}_1\mathbf{Q}\\ \mathbf{A}_2\mathbf{Q}\\ \vdots\\ \mathbf{A}_k\mathbf{Q}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1\mathbf{Q}_1\mathbf{Q}_1^{-1}\mathbf{Q}\\ \mathbf{A}_2\mathbf{Q}_2\mathbf{Q}_2^{-1}\mathbf{Q}\\ \vdots\\ \mathbf{A}_k\mathbf{Q}_k\mathbf{Q}_k^{-1}\mathbf{Q}\end{bmatrix} = \begin{bmatrix}\mathbf{W}_1\mathbf{T}_1\\ \mathbf{W}_2\mathbf{T}_2\\ \vdots\\ \mathbf{W}_k\mathbf{T}_k\end{bmatrix}.$$

It then yields rotation $\mathbf{T}_i$ that can be computed as $\mathbf{T}_i = \mathbf{Q}_i^{-1}\mathbf{Q}_1$. Thus, $\mathbf{T}_i$ can be estimated, without knowing $\mathbf{Q}_1$, as

$$\mathbf{T}_i = \mathbf{Q}_i^{-1}\mathbf{Q}_1 = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{S}^{\#}\mathbf{Q}_1 = \mathbf{Q}_i^{-1}\mathbf{S}(\mathbf{Q}_1^{-1}\mathbf{S})^{\#} = \mathbf{U}_i\mathbf{U}_1^{\#}.$$

where $\mathbf{U}_i$ can be easily computed from (2).

As a result, the principal subspace matrix of $\mathbf{X}$ can be updated as

$$\mathbf{W} = \left[\mathbf{W}_1^T\ (\mathbf{W}_2\mathbf{T}_2)^T \dots (\mathbf{W}_k\mathbf{T}_k)^T\right]^T = \mathbf{A}\mathbf{Q}_1.$$

2.2 PARAFAC based on Alternating Least-Squares

Several divide-and-conquer based algorithms have been proposed for PARAFAC, such as [15–20]. The central idea of the approach is to divide a tensor $\boldsymbol{\mathcal{X}}$ into k parallel sub-tensors $\boldsymbol{\mathcal{X}}_i$, then estimate the factors (loading matrices) of the sub-tensors, and then combine them together into the overall factors of $\boldsymbol{\mathcal{X}}$. In this section, we want to describe the algorithm proposed by Nguyen *et al.* in [18], namely parallel ALS-based PARAFAC, and is summarized in Algorithm 2. This algorithm provides us with inspiration to develop other algorithms in later sections.

Algorithm 2: Parallel ALS-based PARAFAC [18]

Input: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, target rank p, k DSP units
Output: Factors $\mathbf{A} \in \mathbb{R}^{I\times p}, \mathbf{B} \in \mathbb{R}^{J\times p}, \mathbf{C} \in \mathbb{R}^{K\times p}$

function
  Divide $\boldsymbol{\mathcal{X}}$ into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \dots, \boldsymbol{\mathcal{X}}_k$
  Compute $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$ of $\boldsymbol{\mathcal{X}}_1$ using ALS
  Compute factors of sub-tensors: *// updates can be done in parallel*
  for $i = 2 \to k$ **do**
    Compute $\mathbf{A}_i, \mathbf{B}_i$ and $\mathbf{C}_i$ of $\boldsymbol{\mathcal{X}}_i$ using ALS
    Rotate $\mathbf{A}_i, \mathbf{B}_i$ and $\mathbf{C}_i$ // (7)
  Update $\mathbf{A}, \mathbf{B}, \mathbf{C}$ // (8)
return $\mathbf{A}, \mathbf{B}, \mathbf{C}$

Without loss of generality, we assume that a tensor $\boldsymbol{\mathcal{X}}$ is divided into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \dots, \boldsymbol{\mathcal{X}}_k$, by splitting the loading matrix $\mathbf{C}$ into $\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k$ so that the corresponding matrix presentation of the sub-tensor $\boldsymbol{\mathcal{X}}_i$ can be determined by

$$\mathbf{X}_i = (\mathbf{C}_i \odot \mathbf{A})\mathbf{B}^T. \tag{4}$$

Here, $\boldsymbol{\mathcal{X}}_i$ is considered as a tensor composed of frontal slices of $\boldsymbol{\mathcal{X}}$, while $\mathbf{X}_i$ is to present the sub-matrix of its matrix representation $\mathbf{X}$ of $\boldsymbol{\mathcal{X}}$.

Exploiting the fact that the two factors $\mathbf{A}$ and $\mathbf{B}$ are unique when decomposing the sub-tensors, due to the uniqueness of PARAFAC (see [6, Section IV] and [7, Section III]), gives

$$\boldsymbol{\mathcal{X}}_i = \boldsymbol{\mathcal{I}}_i \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}_i. \tag{5}$$

As a result, we need to look for an updated rule in order to concatenate the matrices $\mathbf{C}_i$ into the matrix $\mathbf{C}$, while $\mathbf{A}$ and $\mathbf{B}$ can be directly obtained from PARAFAC of $\boldsymbol{\mathcal{X}}_1$.

In particular, the algorithm can be described as follows. First, by performing PARAFAC of these sub-tensors, the factors $\mathbf{A}_i, \mathbf{B}_i,$ and $\mathbf{C}_i$ can be obtained from decomposing

$$\mathbf{X}_i = (\mathbf{C}_i \odot \mathbf{A}_i)\mathbf{B}_i^T, \tag{6}$$

using the ALS algorithm [21]. Then, $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$ are rotated in the directions of $\boldsymbol{\mathcal{X}}_1$ to yield

$$\mathbf{A}_i \leftarrow \mathbf{A}_i\mathbf{P}_i\mathbf{D}_i^{(\mathbf{A})}, \tag{7a}$$

$$\mathbf{B}_i \leftarrow \mathbf{B}_i\mathbf{P}_i\mathbf{D}_i^{(\mathbf{B})}, \tag{7b}$$

$$\mathbf{C}_i \leftarrow \mathbf{C}_i\mathbf{P}_i\mathbf{D}_i^{(\mathbf{C})}, \tag{7c}$$

where the permutation matrices $\mathbf{P}_i \in \mathbb{R}^{R\times R}$ are given by

$$\mathbf{P}_i(u,v) = \begin{cases}1, & \text{for } \max_v \dfrac{|\langle \mathbf{A}_i(:,u), \mathbf{A}_1(:,v)\rangle|}{\|\mathbf{A}_i(:,u)\|\|\mathbf{A}_1(:,v)\|},\\ 0, & \text{otherwise,}\end{cases}$$

Algorithm 3: Proposed modified GMNS-based PSA

Input: Matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, target rank p, k DSP units
Output: Principal subspace matrix $\mathbf{W}$ of $\mathbf{X}$

function
  Divide $\mathbf{X}$ into k sub-matrices: $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k$
  Compute SVD of $\mathbf{X}_1$ to obtain $[\mathbf{W}_1, \mathbf{U}_1] = \text{svd}(\mathbf{X}_1)$
  // updates can be done in parallel
  for $i = 2 \to k$ **do**
    Compute $\mathbf{W}_i = \mathbf{X}_i \mathbf{U}_1^{\#}$
return $\mathbf{W}_{\mathbf{X}} = [\mathbf{W}_1^T\ \mathbf{W}_2^T\ \ldots\ \mathbf{W}_K^T]^T$

and the scale matrices $\mathbf{D}_i^{(\cdot)} \in \mathbb{R}^{R\times R}$ are given by

$$\mathbf{D}_i^{(\mathbf{A})}(u,u) = \frac{\|\mathbf{A}_1(:,v)\|}{|\langle \mathbf{A}_i(:,u), \mathbf{A}_1(:,v)\rangle|},$$

$$\mathbf{D}_i^{(\mathbf{B})}(u,u) = \frac{\|\mathbf{B}_1(:,v)\|}{|\langle \mathbf{B}_i(:,u), \mathbf{B}_1(:,v)\rangle|},$$

$$\mathbf{D}_i^{(\mathbf{C})}(u,u) = \left(\mathbf{D}_i^{(\mathbf{A})}(u,u)\mathbf{D}_i^{(\mathbf{B})}(u,u)\right)^{-1}.$$

Finally, we obtain the factors of $\boldsymbol{\mathcal{X}}$

$$\mathbf{A} \leftarrow \mathbf{A}_1, \tag{8a}$$

$$\mathbf{B} \leftarrow \mathbf{B}_1, \tag{8b}$$

$$\mathbf{C} \leftarrow \left[\mathbf{C}_1^T\ \mathbf{C}_2^T\ \ldots\ \mathbf{C}_k^T\right]^T. \tag{8c}$$

3 Proposed Modified and Randomized GMNS-based PSA Algorithms

In this section, we introduce two modifications of the GMNS for PSA. Specifically, by expressing the right singular vectors obtained from SVD in terms of principal subspace, we derive a modified GMNS algorithm for PSA which runs faster than that of the original GMNS, while still retaining the subspace estimation accuracy. In addition, we introduce a randomized GMNS algorithm for PSA that can deal with several matrices by performing the randomized SVD [22].

3.1 Modified GMNS-based Algorithm

Consider again the low-rank data matrix $\mathbf{X} = \mathbf{AS}$, as described in Section 2.1. We first look at the true principal subspace matrix $\mathbf{W}_{\mathbf{X}}$, which is obtained via SVD of $\mathbf{X}$, that is,

$$\mathbf{X} \overset{\text{SVD}}{=} \mathbf{W}\boldsymbol{\Sigma}\mathbf{V}^H = \mathbf{W}_{\mathbf{X}}\mathbf{U}_{\mathbf{X}},$$

where $\mathbf{W}_{\mathbf{X}}$ and $\mathbf{U}_{\mathbf{X}}$ present the left singular vectors and the right singular vectors of $\mathbf{X}$ respectively.

Hence, the column space of $\mathbf{A}$ is the same as the column space of $\mathbf{W}_{\mathbf{X}}$. In particular, $\mathbf{X}$ can be expressed by

$$\mathbf{X} = \mathbf{AS} = \mathbf{AQQ}^{-1}\mathbf{S},$$

where $\mathbf{Q}$ is an unknown full rank matrix such that

$$\mathbf{W}_{\mathbf{X}} = \mathbf{AQ},$$
$$\mathbf{U}_{\mathbf{X}} = \mathbf{Q}^{-1}\mathbf{S}.$$

In GMNS, the original matrix $\mathbf{X}$ is split into $\mathbf{X}_1, \ldots, \mathbf{X}_k$ sub-matrices. Suppose that the principal subspace matrix of each sub-matrix $\mathbf{X}_i$ can be determined from

$$\mathbf{X}_i \overset{\text{SVD}}{=} \mathbf{W}_{\mathbf{X}_i}\mathbf{U}_{\mathbf{X}_i},$$

where $\mathbf{W}_{\mathbf{X}_i} = \mathbf{A}_i\mathbf{Q}_i$ and $\mathbf{U}_{\mathbf{X}_i} = \mathbf{Q}_i^{-1}\mathbf{S}$. Then, we obtain the following property:

$$\begin{aligned}\mathbf{X} &= \begin{bmatrix}\mathbf{A}_1\mathbf{S}\\ \mathbf{A}_2\mathbf{S}\\ \vdots\\ \mathbf{A}_k\mathbf{S}\end{bmatrix} = \begin{bmatrix}\mathbf{W}_{\mathbf{X}_1}\mathbf{Q}_1^{-1}\\ \mathbf{W}_{\mathbf{X}_2}\mathbf{Q}_2^{-1}\\ \vdots\\ \mathbf{W}_{\mathbf{X}_k}\mathbf{Q}_k^{-1}\end{bmatrix}\mathbf{S}\\ &= \begin{bmatrix}\mathbf{W}_{\mathbf{X}_1}\\ \mathbf{W}_{\mathbf{X}_2}\mathbf{Q}_2^{-1}\mathbf{Q}_1\\ \vdots\\ \mathbf{W}_{\mathbf{X}_k}\mathbf{Q}_k^{-1}\mathbf{Q}_1\end{bmatrix}\mathbf{Q}_1^{-1}\mathbf{S} = \begin{bmatrix}\mathbf{W}_{\mathbf{X}_1}\\ \mathbf{W}_2\\ \vdots\\ \mathbf{W}_k\end{bmatrix}\mathbf{U}_{\mathbf{X}_1}.\end{aligned} \tag{9}$$

Hence, the relationship between the sub-matrices $\mathbf{X}_i$ and their corresponding subspace matrices can be given by

$$\mathbf{X}_i = \mathbf{W}_i\mathbf{U}_{\mathbf{X}_1},$$
$$\mathbf{W}_i = \mathbf{X}_i\mathbf{U}_{\mathbf{X}_1}^{\#}.$$

As a result, we derive a new implementation of the GMNS algorithm. First, perform SVD of $\mathbf{X}_1$,

$$\mathbf{X}_1 \overset{\text{SVD}}{=} \mathbf{W}_1\mathbf{U}_1,$$

where $\mathbf{W}_1$ is the left singular vector matrix of $\mathbf{X}_1$ and $\mathbf{U}_1 = \boldsymbol{\Sigma}_1\mathbf{V}_1$ is to present its right singular vector matrix.

Next, obtain the principal subspace matrices of other sub-matrices $\mathbf{X}_i$, $i = 2, \ldots, k$, by projecting these sub-matrices onto the pseudo-inverse right singular vector matrix of $\mathbf{X}_1$, that is,

$$\mathbf{W}_i = \mathbf{X}_i\mathbf{U}_1^{\#}.$$

Finally, obtain the principal subspace matrix of $\mathbf{X}$by concatenating the principal subspace matrices of $\mathbf{X}_i$ as

$$\mathbf{W} = [\mathbf{W}_1^T\ \mathbf{W}_2^T\ \ldots\mathbf{W}_k^T]^T,$$
$$\mathbf{X} = \mathbf{WU}_1.$$

The modified GMNS algorithm for PSA can be summarized in Algorithm 3.

3.2 Randomized GMNS-based Algorithm

Although the original GMNS method in [14] provides an efficient tool for fast subspace estimation with high accuracy, it is only useful for the type of low-rank matrices addressed in Section 2.1. This motivates us to look for an improvement on GMNS that can deal with *arbitrary* matrices.

In order to apply GMNS, we want to produce a good approximation $\widehat{\mathbf{X}} = \mathbf{YZ}$ of a given matrix $\mathbf{X}$ that not only satisfies the conditions of GMNS, but also

covers the span and preserve important properties of $\mathbf{X}$. Therefore, the matrix $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$ can be a good sketch of $\mathbf{X}$ where $\mathbf{\Omega}$ is a sketching matrix like a column selection or random projection matrix. Several studies have been proposed to solve the problem so far. For example, we can apply randomized algorithms and sketching techniques in [22–24] for matrices and data to estimate $\mathbf{Y}$, hence $\mathbf{Z}$.

In this work, we are interested in Gaussian $\mathbf{\Omega}$, whose entries are i.i.d. samples generated from $\mathcal{N}(0,1)$. The Gaussian random matrix has been successfully applied in several matrix analysis methods, such as [22, 25, 26]. It is noted that the Gaussian random matrix has many desired properties, such as the following:

- For all vector $\mathbf{x}$ in the row space of $\mathbf{X}$, its length will not change much if sketching by $\mathbf{\Omega}$: $\|\mathbf{x}\|^2 \approx \|\mathbf{x}\mathbf{\Omega}\|^2$;
- In general, random vectors of $\mathbf{\Omega}$ are likely to be linear position and linearly independent;
- There is no linear combination falling in the null space of $\mathbf{X}$.

As a result, $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$ is a high quality sketch and can span the range of $\mathbf{X}$.

After finding a good sketch $\mathbf{Y}$ from the Gaussian random matrix $\mathbf{\Omega}$, the next problem is low-rank matrix approximation such that its result has to hold the Frobenius norm error bound with high probability. This leads to the following optimization problem:

$$\min_{\text{rank}(\mathbf{Z})\leq k} \|\mathbf{X} - \mathbf{Y}\mathbf{Z}\|_F^2 \leq (1+\epsilon)\|\mathbf{X} - \mathbf{X}_k\|_F^2 = (1+\epsilon)\sum_{i=k+1}^{N} \sigma_i(\mathbf{X}), \quad (10)$$

where $\sigma_i(\mathbf{X})$ is the i-th singular value of $\mathbf{X}$ and $\mathbf{X}_k$ is the best rank-k approximate of $\mathbf{X}$.

Let $\mathbf{Q}_\mathbf{Y}$ contain orthogonal bases of the sketch $\mathbf{Y}$ of $\mathbf{X}$. Clearly, since $\mathbf{Q}_\mathbf{Y}$ shares the same column space with $\mathbf{Y}$, the optimization problem of (10) can be rewritten as

$$\mathbf{Z}^\star = \arg\min_{\text{rank}(\mathbf{Z})\leq k} \|\mathbf{X} - \mathbf{Q}_\mathbf{Y}\mathbf{Z}\|_F^2. \quad (11)$$

The solution of (11) can be computed more easily as

$$\mathbf{Z}^\star = \mathbf{Q}_\mathbf{Y}^T\mathbf{X}. \quad (12)$$

Therefore, with $\mathbf{Q}_\mathbf{Y}$, the Frobenius norm error in the problem (10) can be extended to a stronger error measure, that is, the spectral norm error bound (we refer the reader to [24, Section 4.3] for further details), as follows:

$$\|\mathbf{X} - \mathbf{Q}_\mathbf{Y}\mathbf{Q}_\mathbf{Y}^T\mathbf{X}\|_2^2 \leq (1+\epsilon)\|\mathbf{X} - \mathbf{X}_k\|_2^2 = (1+\epsilon)\sigma_{k+1}(\mathbf{X}).$$

From now, we have an approximate basis for the range of $\mathbf{X}$, that is,

$$\mathbf{X} \approx \mathbf{Q}_\mathbf{Y}\mathbf{Q}_\mathbf{Y}^T\mathbf{X}.$$

Let us define $\bar{\mathbf{A}} = \mathbf{Q}_\mathbf{Y}^T\mathbf{X}$. We then have

$$\mathbf{X} \approx \mathbf{Q}_\mathbf{Y}\bar{\mathbf{A}}.$$

Accordingly, the principal subspace matrix $\mathbf{W}_{\bar{\mathbf{A}}}$ of $\bar{\mathbf{A}}$ can be computed by using the original GMNS or the modified GMNS proposed in Section 3.1. Then we can estimate the principal subspace of an arbitrary matrix $\mathbf{X}$ by

$$\mathbf{W}_\mathbf{X} \approx \mathbf{Q}_\mathbf{Y}\mathbf{W}_{\bar{\mathbf{A}}}.$$

This randomized GMNS algorithm for PSA is summarized in Algorithm 4.

Algorithm 4: Proposed randomized GMNS-based PSA

Input: Matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, target rank p, k DSP units
Output: Principal subspace matrix $\mathbf{W}$ of $\mathbf{X}$.

1 **function**
2 Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m\times l}$, $l > p$
3 Form the sketch $\mathbf{Y} \in \mathbb{R}^{n\times l}$ of $\mathbf{X}$: $\mathbf{Y} = \mathbf{X}\mathbf{\Omega}$
4 Extract principal subspace $\mathbf{Q}$ from $\mathbf{Y}$ using QR decomposition
5 Construct $\bar{\mathbf{A}} = \mathbf{Q}^T\mathbf{X}$
6 Estimate $\mathbf{W}_{\bar{\mathbf{A}}}$ of $\bar{\mathbf{A}}$ using GMNS
7 **return** $\mathbf{W}_\mathbf{X} = \mathbf{Q}\mathbf{W}_{\bar{\mathbf{A}}}$

Remark

Recall that GMNS is with a parallel computing architecture in practice. Therefore estimating the orthogonal basis of the sketch $\mathbf{Y}$ based on QR decomposition should be implemented in a parallelization scheme. In this work, we can parallelize the randomized GMNS algorithm by using a type of distributed QR decomposition, namely TSQR [27].

Specifically, we divide $\mathbf{X}$ into k sub-matrices $\mathbf{X}_i$, as in the original GMNS and the modified GMNS algorithms. First, we find all the sketch $\mathbf{Y}_i$ of the sub-matrices $\mathbf{X}_i$ under the sketching $\mathbf{\Omega}$. Next, we perform standard QR decomposition on each sub-matrix $\mathbf{Y}_i$ to obtain $\mathbf{Q}_{1,i}$ and $\mathbf{R}_{1,i}$. The resulting matrices $\mathbf{R}_{1,i}$ are then gathered into a single matrix $\mathbf{R}_1$ which is then decomposed into $\mathbf{Q}_{2,:}$ again. As a result, the original factor $\mathbf{Q}$ of $\mathbf{Y}$ can be obtained from multiplying the resulting the $\mathbf{Q}_{1,:}$ with $\mathbf{Q}_{2,:}$, which can be already distributed among the DSP units. Finally, we find the orthogonal basis of the sketch $\bar{\mathbf{A}} = \mathbf{Q}^T\mathbf{Y}$ by using the original GMNS or modified GMNS algorithms, and hence the principal subspace matrix of $\mathbf{X}$. We refer the reader to [27] for further details.

3.3 Computational Complexity

For the sake of simplicity, we assume that standard algorithms for computing matrix multiplication and different types of matrix decomposition (i.e., EVD, SVD, QR) are applied in this work, while costs of transfer and synchronization among the DSP units are ignored. Specifically, to decompose a rank-p matrix of size $n \times n$ into factors, the standard EVD requires a cost of $\mathcal{O}(n^2p)$. Considering a non-square matrix of size $n \times m$, the full Householder QR algorithm is computed in $2nm^2 - 2/3m^3$ flops, while the truncated SVD typically needs nmp flops to derive a rank-p approximation by using the partial QR decomposition. These methods are surveyed in [28]. To multiply a matrix $\mathbf{A}$ of size $n \times p$

with a matrix $\mathbf{B}$ of size $p \times m$, we consider the standard algorithm which is to perform n dot products of rows in $\mathbf{A}$ and columns in $\mathbf{B}$ whose cost is $\mathcal{O}(nmp)$.

Now, we analyse the computational complexity of the modified and randomized GMNS algorithms for PSA. The former consists of two main operations: (i) perform the truncated SVD of $\mathbf{X}_1$, which costs nmp/k flops, and (ii) perform $(k-1)$ matrix multiplications between sub-matrices $\mathbf{X}_i$ and the right-singular vector matrix of $\mathbf{X}_1$, which requires nmp/k flops. Therefore, the overall complexity is $\mathcal{O}(nmp/k)$. Meanwhile, the computational complexity of the original GMNS algorithm or PSA is $\mathcal{O}(n^2(m+p)/k^2)$. Since $m, n \gg p$, the original and the modified GMNS algorithms have lower complexity than that of the well-known method using EVD of the global covariance matrix that costs $\mathcal{O}\left(n^2(m+p)\right)$ flops.

The randomized GMNS algorithm consists of three main operations: (i) estimate a good sketch $\mathbf{Y}$ of $\mathbf{X}$, (ii) orthonormalize the columns of $\mathbf{Y}$, and (iii) update its subspace matrix. In the first operation, deriving a standard Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times l}$ and hence a good sketch $\mathbf{Y} \in \mathbb{R}^{n \times l}$ demands a cost of $\mathcal{O}(mnl)$. In the second operation, QR decomposition, used to compute the orthogonal basis of $\mathbf{Y}$, demands a cost of $2nl^2 - 2/3l^3$ flops. In the last operation, two matrix products are used to compute the matrix $\bar{\mathbf{A}}$ and update $\mathbf{W}_{\mathbf{X}}$, demanding a cost of $nl(m+p)$ flops. In addition, the algorithm uses the same order of complexity for estimating the subspace of $\bar{\mathbf{A}}$ using GMNS. Moreover, we can use the structured random matrix $\mathbf{\Omega}$ using the subsampled random FFT instead, to reduce the overall complexity. Specifically, it allows us to compute the product of $\mathbf{X}$ and $\mathbf{\Omega}$ in $nm\log(l)$ flops; and the row-extraction technique to derive $\mathbf{Q}$ with a lower cost of $\mathcal{O}(k^2(n+m))$. We refer the reader to [22, Section 4.6] for further details. In conclusion, the overall complexity of the randomized GMNS algorithm is $\mathcal{O}(nl(m+p)/k)$ using the TSQR algorithm.

4 Proposed GMNS-based PARAFAC

In this section, we derive a new implementation of PARAFAC of three-way tensors based on GMNS. Consider a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$. PARAFAC of $\boldsymbol{\mathcal{X}}$ is expressed as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{i=1}^{R} \mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i, \qquad (13)$$

where R is the rank of $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{I}}$ is an identity tensor, $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$ and $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factors (loading matrices).

Motivated by the advantages of GMNS and the ALS-based PARAFAC in Section 2.2, we are interested in finding a parallelization scheme for PARAFAC. The proposed algorithm consists of four steps:

- Step 1: Divide the tensor $\boldsymbol{\mathcal{X}}$ into k sub-tensors $\boldsymbol{\mathcal{X}}_1$, $\boldsymbol{\mathcal{X}}_2, \ldots, \boldsymbol{\mathcal{X}}_k$;
- Step 2: Estimate the principal subspace matrix of each tensors, $\mathbf{W}_i = (\mathbf{C}_i \odot \mathbf{A}_i)\mathbf{Q}_i$, using GMNS;
- Step 3: Obtain the loading matrices $\mathbf{A}, \mathbf{Q}$ and $\mathbf{B}$, using some desired properties of GMNS;
- Step 4: Update the loading matrix $\mathbf{C}$.

The main difference between the GMNS-based and ALS-based PARAFAC algorithms is in the way we compute factors $\mathbf{A}_i, \mathbf{B}_i$ and $\mathbf{C}_i$ of each sub-tensor $\boldsymbol{\mathcal{X}}_i$. Specifically, instead of applying ALS for k sub-tensors, these factors can be directly obtained from the principal subspace of each of the sub-tensors $\boldsymbol{\mathcal{X}}_i, i = 2, 3, \ldots, k$. Therefore, we only need to apply ALS for the first sub-tensor $\boldsymbol{\mathcal{X}}_1$. Now, we will describe the algorithm in details.

For the sake of simplicity, assume that the given tensor $\boldsymbol{\mathcal{X}}$ is divided into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \ldots, \boldsymbol{\mathcal{X}}_k$ by splitting the loading matrix $\mathbf{C}$ as in the ALS-based PARAFAC algorithm. The corresponding matrix representation of the sub-tensors and their subspace matrices are also given by

$$\begin{aligned} \mathbf{X}_i &= (\mathbf{C}_i \odot \mathbf{A})\mathbf{B}^T, \\ \mathbf{W}_i &= (\mathbf{C}_i \odot \mathbf{A})\mathbf{Q}_i, \end{aligned}$$

where $\mathbf{Q}_i \in \mathbb{R}^{R \times R}$ is a full rank matrix.

First, by using any specific PARAFAC algorithm, such as the ALS-based PARAFAC one, to compute the factors $\mathbf{A}_1$, $\mathbf{B}_1$, and $\mathbf{C}_1$ of $\boldsymbol{\mathcal{X}}_1$ from

$$\mathbf{X}_1 = (\mathbf{C}_1 \odot \mathbf{A}_1)\mathbf{B}_1^T,$$

we obtain the two factors $\mathbf{A} \leftarrow \mathbf{A}_1$ and $\mathbf{B} \leftarrow \mathbf{B}_1$. In addition, the principal subspace matrix $\mathbf{W}_1$ of $\mathbf{X}_1$ can also be given by

$$\mathbf{W}_1 = (\mathbf{C}_1 \odot \mathbf{A}_1)\mathbf{Q}_1.$$

Therefore, the two rotation matrices $\mathbf{Q}_1$ and $\mathbf{U}_1$ can be obtained as

$$\mathbf{Q}_1 = (\mathbf{C}_1 \odot \mathbf{A}_1)^{\#}\mathbf{W}_1, \qquad (14a)$$

$$\mathbf{U}_1 = \mathbf{W}_1^{\#}\mathbf{X}_1. \qquad (14b)$$

From now, the factors of $\boldsymbol{\mathcal{X}}_i$, $i = 2, \ldots, k$, can be derived directly from their principal subspace matrices $\mathbf{W}_i$ of $\mathbf{X}_i$ by

$$\mathbf{W}_i = \mathbf{X}_i\mathbf{U}_1^{\#}, \qquad (15a)$$

$$\mathbf{C}_i \odot \mathbf{A}_i = \mathbf{W}_i\mathbf{Q}_1^{-1}. \qquad (15b)$$

The loading matrices $\mathbf{A}_i$ and $\mathbf{C}_i$ are then easily recovered, thanks to the Khatri-Rao product. In parallel, the loading matrix $\mathbf{B}_i$ can be updated as

$$\mathbf{B}_i = \mathbf{X}_i^T(\mathbf{W}_i^{\#})^T\mathbf{Q}_1^T. \qquad (16)$$

The next step is to rotate the loading matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ according to (7). The factors of the overall PARAFAC are then obtained as

$$\begin{aligned} &\mathbf{A} \leftarrow \mathbf{A}_1, \mathbf{B} \leftarrow \mathbf{B}_1, \\ &\mathbf{C} \leftarrow \left[\mathbf{C}_1^T\ \mathbf{C}_2^T\ \ldots\ \mathbf{C}_k^T\right]^T. \end{aligned} \qquad (17)$$

The proposed GMNS-based PARAFAC algorithm is summarized in Algorithm 5.

Algorithm 5: Proposed GMNS-based PARAFAC

Input: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, target rank R, k DSP units
Output: Factors $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$ and $\mathbf{C} \in \mathbb{R}^{K\times R}$

Initilization
- Divide $\boldsymbol{\mathcal{X}}$ into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \ldots, \boldsymbol{\mathcal{X}}_k$
- Apply ALS to compute $\mathbf{A}_1, \mathbf{B}_1$ and $\mathbf{C}_1$ of $\boldsymbol{\mathcal{X}}_1$
- Extract principal subspace $\mathbf{W}_1$ of $\mathbf{X}_1$ using SVD
- Compute rotation matrix $\mathbf{Q}_1 = (\mathbf{C}_1 \odot \mathbf{A}_1)^{\#}\mathbf{W}_1$

Main Update factors of other sub-tensors
// updates can be done in parallel
- **for** $i = 2 \to k$ **do**
  - Extract principal subspace $\mathbf{W}_i$ of $\mathbf{X}_i$ using SVD
  - Compute $\mathbf{C}_i$ and $\mathbf{A}_i$ // (15)
  - Compute $\mathbf{B}_i$ // (16)
  - Rotate $\mathbf{A}_i, \mathbf{C}_i$ and $\mathbf{B}_i$ // (7)

return $\mathbf{A}, \mathbf{B}, \mathbf{C}$ // (17)

Remark

In the case of tensors with $K \gtrapprox I \times J$, the GMNS-based PARAFAC algorithm can be implemented more efficiently. Matrix representation of the overall tensor and its sub-tensors can be expressed, respectively, by

$$\mathbf{X} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T,$$
$$\mathbf{X}_i = \mathbf{C}_i(\mathbf{B} \odot \mathbf{A})^T.$$

Therefore, the factors can be computed more easily. Specifically, the principal subspace of $\mathbf{X}_i$ can be given by

$$\mathbf{W}_i = \mathbf{C}_i\mathbf{Q}_i.$$

Meanwhile, the rotation matrices are updated in a way similar to the above, as

$$\mathbf{U}_1 = \mathbf{W}_1^{\#}\mathbf{X}_1,$$
$$\mathbf{Q}_1 = \mathbf{C}_1^{\#}\mathbf{W}_1.$$

Therefore, the sub-factors $\mathbf{C}_i$ are obtained as

$$\mathbf{C}_i = \mathbf{W}_i\mathbf{Q}_1^{-1},$$

where $\mathbf{W}_i = \mathbf{X}_i\mathbf{U}_1^{\#}$. As a result, the loading matrix $\mathbf{C}$ is updated while $\mathbf{A}$ and $\mathbf{B}$ are computed from $\mathbf{X}_1$.

5 Proposed GMNS-based HOSVD

In this section, we investigate a parallelization scheme for HOSVD of three-way tensors based on GMNS. Consider again a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$. Tucker decomposition of $\boldsymbol{\mathcal{X}}$ can be expressed as

$$\begin{aligned}\boldsymbol{\mathcal{X}} &= \boldsymbol{\mathcal{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= \sum_{i=1}^{R_1}\sum_{j=1}^{R_2}\sum_{k=1}^{R_3} \mathcal{G}_{i,j,k}\mathbf{a}_i \circ \mathbf{b}_j \circ \mathbf{c}_k,\end{aligned}$$

where $\mathbf{A} \in \mathbb{R}^{I\times R_1}$, $\mathbf{B} \in \mathbb{R}^{J\times R_2}$ and $\mathbf{C} \in \mathbb{R}^{K\times R_3}$ are the loading factors, $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1\times R_2\times R_3}$ is the core of $\boldsymbol{\mathcal{X}}$ with $R_1 \le I, R_2 \le J$ and $R_3 \le K$. The decomposition can be desribed in Figure 1.

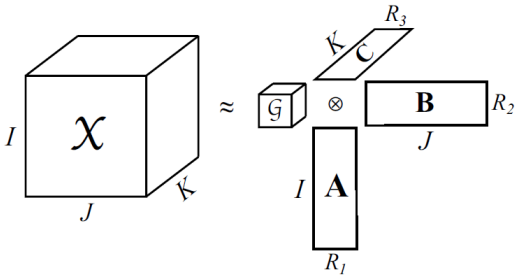


Figure 1: Higher-order singular value decomposition.

HOSVD, also called Tucker1, is a specific Tucker decomposition with orthogonal factors being derived from singular vectors of the three matrices unfolding $\boldsymbol{\mathcal{X}}$ according to the three modes of the tensor. In general, Tucker decomposition is not unique (see [6, Section V] or [7, Section IV]). Fortunately, the subspaces spanned by the factors $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are physically unique. It means that these factors can be rotated by any full rank matrix $\mathbf{Q}$. In turn, this multiplies the core tensor with its inverse. We are interested in to see if GMNS can be used to find multilinear subspaces of tensors, hence used for HOSVD.

Similarly to GMNS-based PARAFAC, we divide $\boldsymbol{\mathcal{X}}$ into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \ldots, \boldsymbol{\mathcal{X}}_k$ whose corresponding matrix representations are

$$\mathbf{X}_i = \mathbf{C}_i\mathbf{G}(\mathbf{B} \otimes \mathbf{A})^T.$$

We exploit the fact that the factors are derived from the principal components of the three modes. Thus, to estimate subspaces for $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, we can apply the following calculation of the covariance matrix of the tensor:

$$\begin{aligned}\mathbf{R}_{\mathbf{X}} &= \mathrm{E}\{\mathbf{X}\mathbf{X}^T\} \\ &= \mathrm{E}\{\mathbf{C}\mathbf{G}(\mathbf{B} \otimes \mathbf{A})^T(\mathbf{B} \otimes \mathbf{A})\mathbf{G}^T\mathbf{C}^T\} \\ &= \mathrm{E}\{\mathbf{C}\mathbf{G}\mathbf{G}^T\mathbf{C}^T\} \\ &= \mathbf{C}\mathbf{R}_{\mathbf{G}}\mathbf{C}^T \\ &\overset{\text{EVD}}{=} \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^T.\end{aligned}$$

It is therefore essential to demonstrate that the principal subspace matrix carries information of these factors, that is,

$$\mathbf{W} = \mathbf{C}\mathbf{Q},$$

where $\mathbf{Q} \in \mathbb{R}^{R_3\times R_3}$ is an unknown full rank matrix.

We can derive all these factors by using the original GMNS algorithm for PSA or the modified and randomized GMNS algorithms proposed in this paper. Here, we only illustrate this by using the proposed modified GMNS algorithm. Specifically, assume that we have already obtained the factors $\mathbf{A}_1, \mathbf{B}_1$, and $\mathbf{C}_1$ of the sub-tensor $\boldsymbol{\mathcal{X}}_1$, whether by the original HOSVD, or alternatively, such as the original higher order orthogonal iteration of tensors (HOOI) decomposition.

Then, by using GMNS to estimate the principal subspace matrices of the sub-tensors, we can obtain the

Algorithm 6: Proposed GMNS-based HOSVD

Input: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, target rank R, k DSP units
Output: Factors $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, $\mathbf{C} \in \mathbb{R}^{K\times R}$

function
  Divide $\boldsymbol{\mathcal{X}}$ into k sub-tensors $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \ldots, \boldsymbol{\mathcal{X}}_k$
  Compute factors of $\boldsymbol{\mathcal{X}}_1$'s using HOSVD
  $\{\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1\} = \text{HOSVD}(\boldsymbol{\mathcal{X}}_1)$
  Compute rotation matrix: $\mathbf{U}_1 = \mathbf{C}_1^{\#}\mathbf{X}_1$
  Update factors using modified GMNS algorithm
  // updates can be done in parallel
  for $i = 2 \to k$ **do**
    Compute $\mathbf{C}_i = \boldsymbol{\mathcal{X}}_i^{(3)}\mathbf{U}_1^{\#}$
return $\mathbf{A} \leftarrow \mathbf{A}_1, \mathbf{B} \leftarrow \mathbf{B}_1$ and $\mathbf{C} \leftarrow [\mathbf{C}_1^T, \mathbf{C}_2^T, \ldots, \mathbf{C}_k^T]^T$

decomposition. Specifically,

$$\mathbf{U}_1 = \mathbf{C}_1^{\#}\mathbf{X}_1, \tag{18}$$

where the matrix $\mathbf{U}_1$ presents the right singular vectors of $\mathbf{X}_1$. As shown in Section 4, we have to rotate the sub-factors $\mathbf{C}_i$ to follow the direction of $\mathbf{C}_1$. Instead of computing the rotation matrices $\mathbf{T}_i$, we dedicate the work to projecting matrices $\mathbf{X}_i$ onto the row space $\mathbf{U}_1$ of $\mathbf{X}_1$, that is,

$$\mathbf{C}_i = \mathbf{X}_i\mathbf{U}_1^{\#}. \tag{19}$$

As a result, the subspace generated by the loading factors $\mathbf{A}_i$ and $\mathbf{B}_i$ remains constant. The overall loading matrices can be updated as

$$\mathbf{A} \leftarrow \mathbf{A}_1, \mathbf{B} \leftarrow \mathbf{B}_1, \tag{20a}$$
$$\mathbf{C} \leftarrow [\mathbf{C}_1^T\ \mathbf{C}_2^T\ \ldots\ \mathbf{C}_k^T]^T. \tag{20b}$$

The core tensor $\mathbf{G}$ can be also computed as

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T. \tag{21}$$

The implementation of the proposed GMNS-based HOSVD is summarized in Algorithm 6.

6 Results

In this section, simulated experiments are carried out to study the performance of the proposed GMNS-based algorithms for PSA and tensor decomposition and compare them with the state-of-the-art algorithms. Some application-based scenarios are also presented to illustrate the effectiveness of the proposed algorithms.

6.1 GMNS-based PSA

We follow experiments and evaluation metrics used in [14]. Specifically, the measurement data $\mathbf{X} = \mathbf{AS}$ are generated by a random system matrix $\mathbf{A}$ and a signal matrix $\mathbf{S}$. The received data are then normalized by its Frobenius norm. The impact of noise on algorithm performance is also investigated by adding noise $\mathbf{N}$ derived from the white Gaussian noise $\mathcal{N}(0, \sigma^2)$, by

$$\mathbf{X} = \frac{\mathbf{X}}{\|\mathbf{X}\|} + \sigma\frac{\mathbf{N}}{\|\mathbf{N}\|}.$$

The signal-to-noise ratio (SNR) is then defined as

$$\text{SNR} = -10\log_{10}\sigma^2.$$

To evaluate the subspace estimation accuracy, we use the subspace estimation performance (SEP) metric

$$\text{SEP} = \frac{1}{L}\sum_1^L \frac{\text{tr}\{\mathbf{W}_i^H(\mathbf{I} - \mathbf{W}_{\text{ex}}\mathbf{W}_{\text{ex}}^H)\mathbf{W}_i\}}{\text{tr}\{\mathbf{W}_i^H(\mathbf{W}_{\text{ex}}\mathbf{W}_{\text{ex}}^H)\mathbf{W}_i\}}, \tag{22}$$

and the eigenvector estimation performance (EEP) metric, also referred to as Root Means Square Error,

$$\text{EEP} = \frac{1}{L}\sum_1^L \|\mathbf{U}_i - \mathbf{U}_{\text{ex}}\|_F^2, \tag{23}$$

where L is the number of Monte Carlo runs, $\mathbf{W}_i$ and $\mathbf{U}_i$ respectively denote the estimated subspace and eigenvector matrices at the i-th run, $\mathbf{W}_{ex}$ and $\mathbf{U}_{ex}$ denote the true subspace and eigenvector matrices. Good performance corresponds to low SEP and EEP.

We study the performance of the proposed modified GMNS and randomized GMNS algorithms for PSA by comparing them to the state-of-the-art algorithms based on SVD, the randomized SVD [22], and the original GMNS. The number of Monte Carlo run is fixed at $L = 100$.

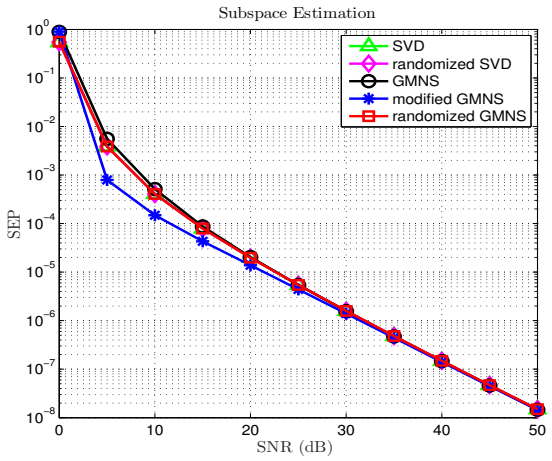
6.1.1 Effect of the number of sources, p:
To study the effect of the number of sources, p, we fixed the number of sensors, n, the number of time observations, m, and the number of DSP units, k, at 200, 500 and 2, respectively. It can be seen from Figure 2, when dealing with a specific p, the modified GMNS and randomized GMNS algorithms performed similarly to those based on the original GMNS, SVD and the randomized SVD, in terms of SEP and EEP. In particular, at low SNRs ($\leq$ 10 dB), the SVD-based algorithm yielded slightly better subspace estimation accuracy than the GMNS-based ones. Meanwhile, at high SNRs ($>$ 10 dB), all algorithms performed similarly.

As shown in Figure 3, no significant difference of subspace estimation accuracy among the original, modified and randomized GMNS-based algorithms when changing the number of sources p, excepting the case of the modified GMNS-based one with small p at SNR $=$ 10 dB. However, the result is still reasonable when compared to the conventional SVD-based algorithms.

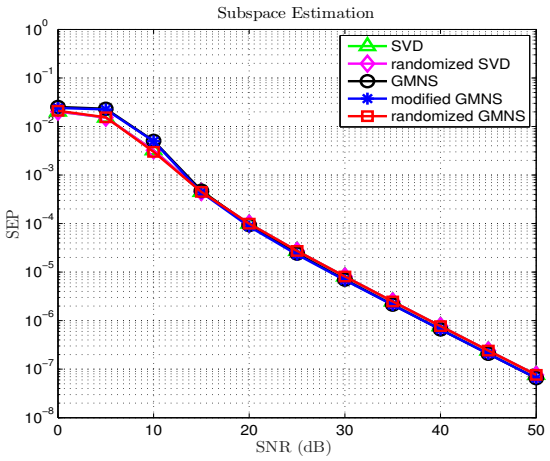
6.1.2 Effect of the number of DSP units, k:
We fixed $n = 240$, $m = 600$, and $p = 2$, while varying k. The experimental results indicated that increasing k yielded slightly reduced SEP. More specifically, when the system $\mathbf{A}$ is divided into a small number of subsystems ($k < 10$), all algorithms provided almost same subspace estimation accuracy, as shown in Figure 4.

When k is large, the randomized GMNS algorithm performed similarly to the SVD-based and the randomized SVD-based algorithms, and slightly better than the original GMNS and the modified GMNS algorithms, as shown in Figure 5.

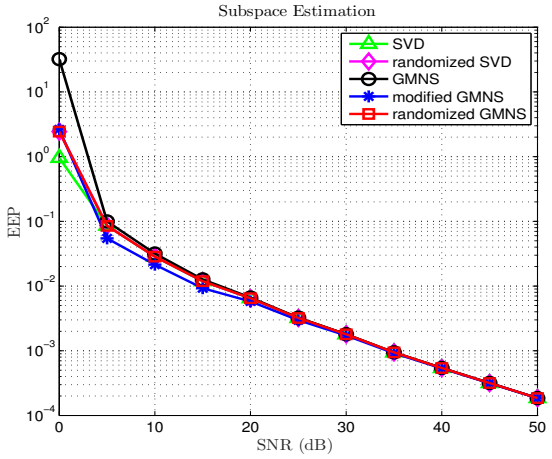
6.1.3 Effect of the number of sensors, n, and time observations, m:
We fixed $k = 2$ and $p = 2$, while varying the size



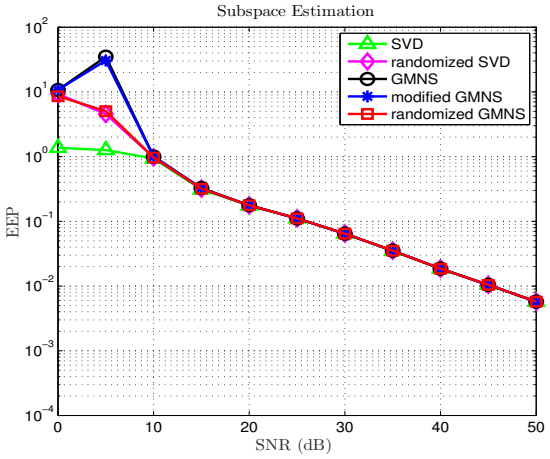
(a) $p = 2$



(b) $p = 50$



(c) $p = 2$



(d) $p = 50$

Figure 2: Effect of number of sources, p, on performance of PSA algorithms; $n = 200$, $m = 500$, $k = 2$.

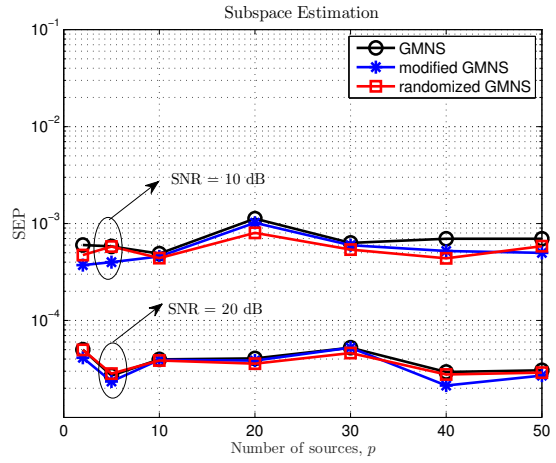


Figure 3: Performance of the proposed GMNS algorithms for PSA versus the number of sources, p, with $n = 200$, $m = 500$ and $k = 2$.

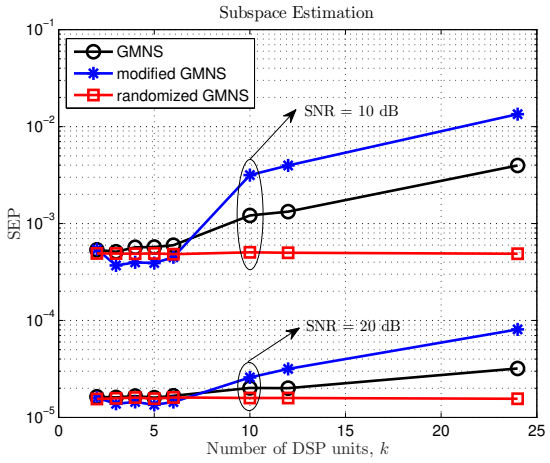
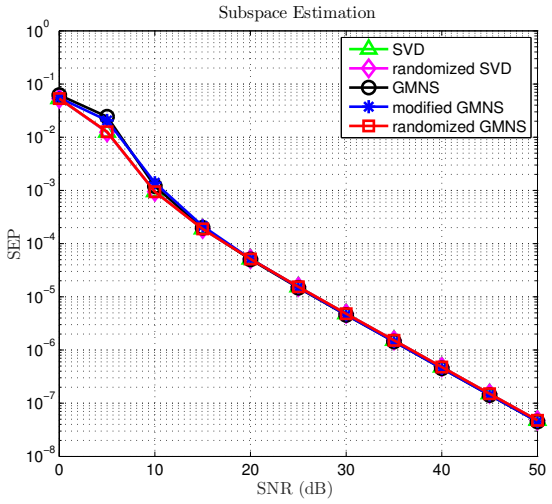
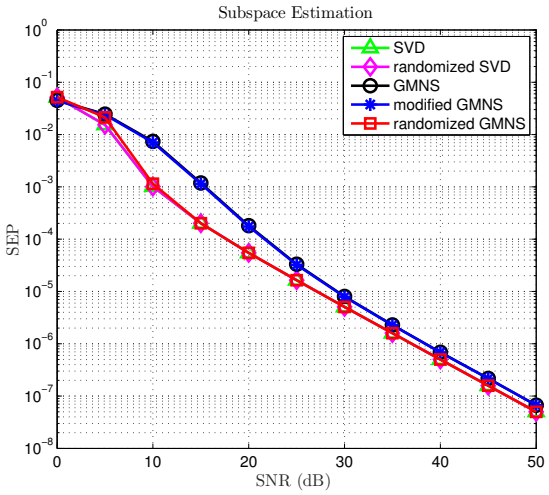


Figure 4: Performance of the proposed GMNS algorithms for PSA versus the number of DSP units, k, with $n = 240$, $m = 600$ and $p = 2$.

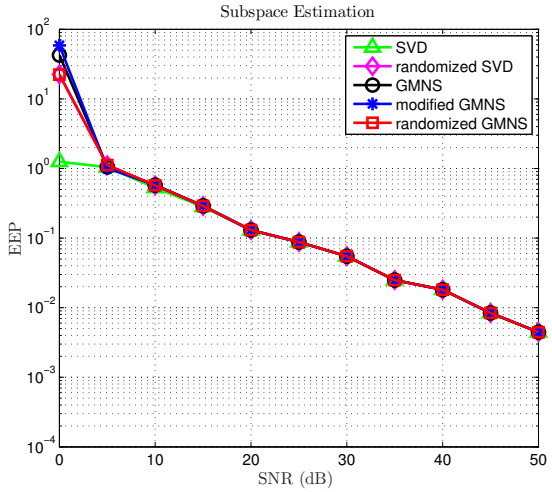
of the data matrix, (m, n). The results, as shown in Figure 6, indicated that all methods yielded similar subspace estimation accuracy. However, in terms of run time, Figure 7 indicated that, when the data matrix is small ($n, m \leq 1000$), all GMNS algorithms took a similar amount time to obtain the same accuracy. When dealing with matrices of higher dimension, the modified GMNS algorithm was faster.



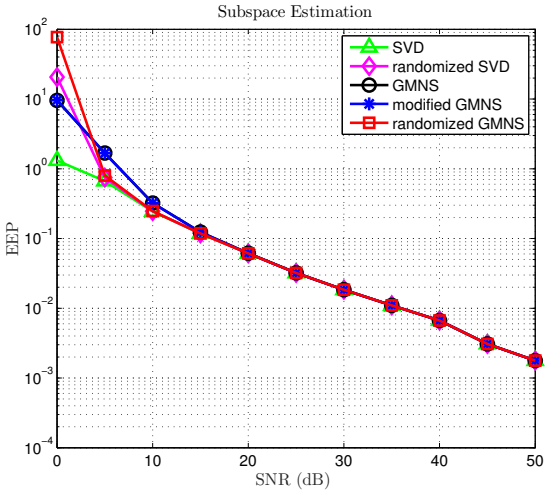
(a) $k = 2$



(b) $k = 10$



(c) $k = 2$



(d) $k = 10$

Figure 5: Effect of number of DSP units, k, on performance of PSA algorithms; $n = 240, m = 600, p = 2$.

6.1.4 Effect of the relationship between the number of sensors, sources and the number of DSP units:
As mentioned above, the original GMNS and the modified GMNS algorithms for PSA may be useful only for data measurements which satisfy the condition $p < n/k$, meanwhile the randomized GMNS algorithm was proposed to handle the rest. The key idea is (i) to choose the number of random vectors so that $n < kp \leq l$, so the problem will return the original setup, or (ii) to structure the random matrix using the subsampled random fast Fourier transform, thanks to advantages of the spectral domain. We fixed the size of the data matrix at $n = 150, m = 500$, and fixed $k = 2$. The number of random vectors was $l = 2p$. As shown in Figure 8, the randomized GMNS algorithm can be useful for the problem, as shown via the green line.

6.2 GMNS-based PARAFAC

We simulated tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, derived from the Gaussian distribution $\mathcal{N}(0,1)$. The tensors were then normalized and added by a random noise $\boldsymbol{\mathcal{N}}$ with a parameter σ to control the noise level

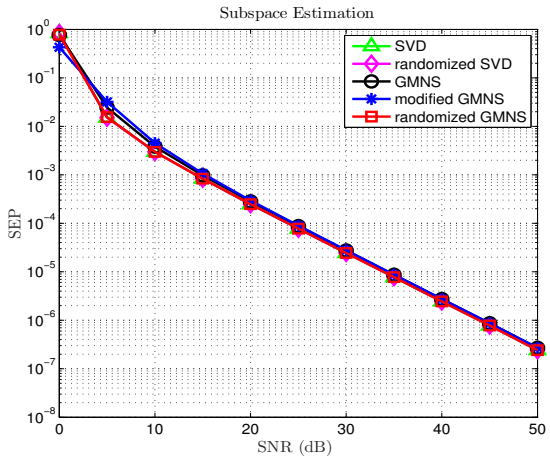
$$\boldsymbol{\mathcal{Y}} = \frac{\boldsymbol{\mathcal{X}}}{\|\boldsymbol{\mathcal{X}}\|} + \sigma \frac{\boldsymbol{\mathcal{N}}}{\|\boldsymbol{\mathcal{N}}\|}.$$

To assess the estimated factors, we use the metric of relative error, ρ, as given by

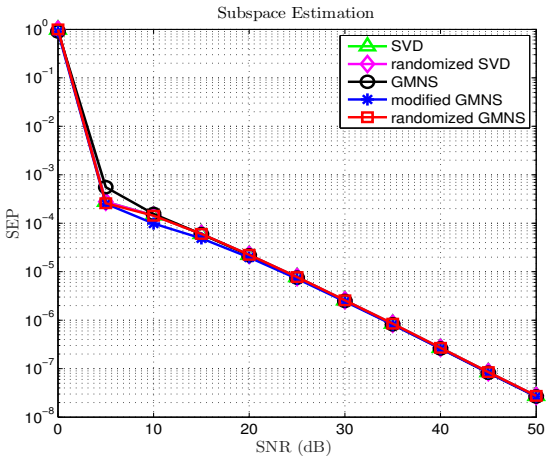
$$\rho(\mathbf{H}) = \frac{1}{L} \sum_{i=1}^{L} \frac{\|\mathbf{H}_i - \mathbf{H}_{ex}\|}{\|\mathbf{H}_{ex}\|}, \tag{24}$$

where L is the Monte Carlo run, $\mathbf{H}_i$ and $\mathbf{H}_{ex}$ are estimated and true factors respectively.

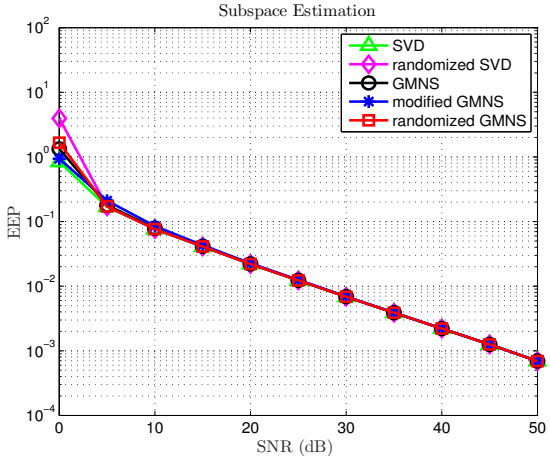
Our experiments were implemented in MATLAB 2015b on Intel core i7 processor and 8G RAM machine using the tensor toolbox [29]. Four kinds of PARAFAC algorithms were compared: simultaneous diagonalization computed by QR iteration-based PARAFAC, namely SDQZ-based PARAFAC [30], original ALS-based PARAFAC [21], parallel ALS-based PARAFAC developed in [18] and described in Section 2.2 and the proposed GMNS-based PARAFAC. The number of Monte Carlo run was fixed at $L = 100$.



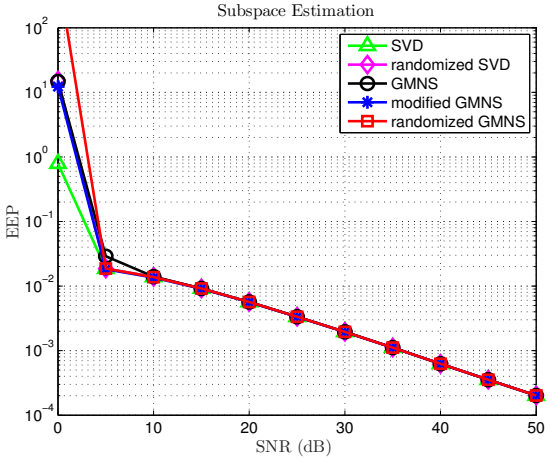
(a) $n = 50$, $m = 100$



(b) $n = 200$, $m = 1000$



(c) $n = 50$, $m = 100$



(d) $n = 200$, $m = 1000$

Figure 6: Effect of matrix size, (m, n), on performance of PSA algorithms; $p = 2$, $k = 2$.

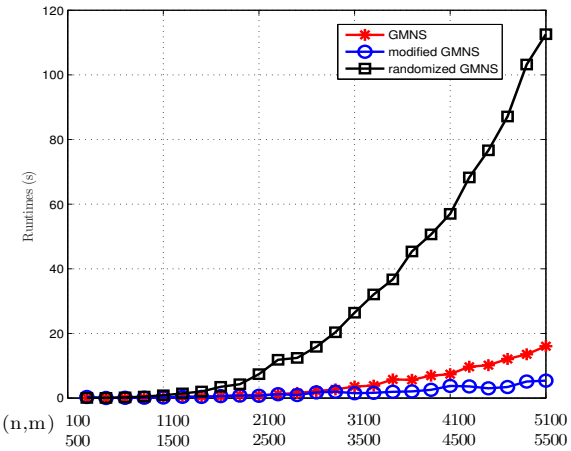


Figure 7: Effect of data matrix size, (n, m), on runtime of GMNS-based PSA algorithms; $p = 20$, $k = 5$.

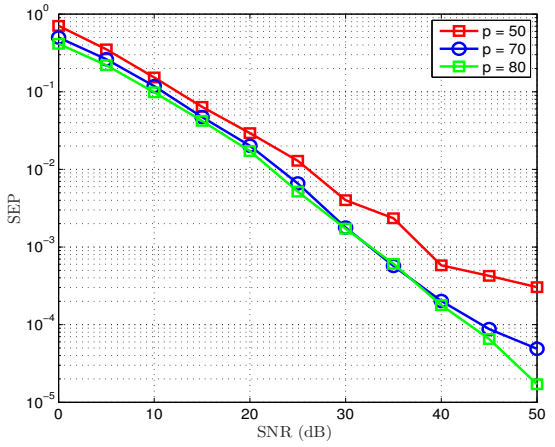
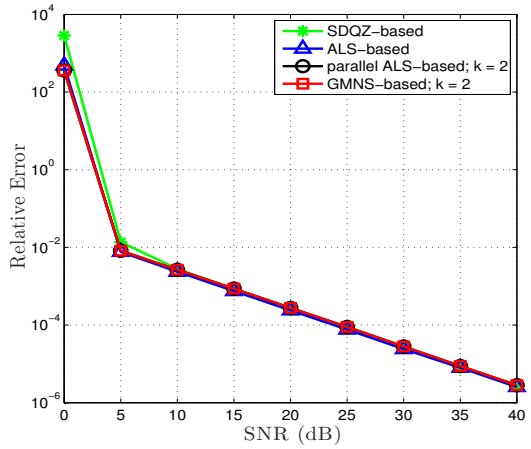


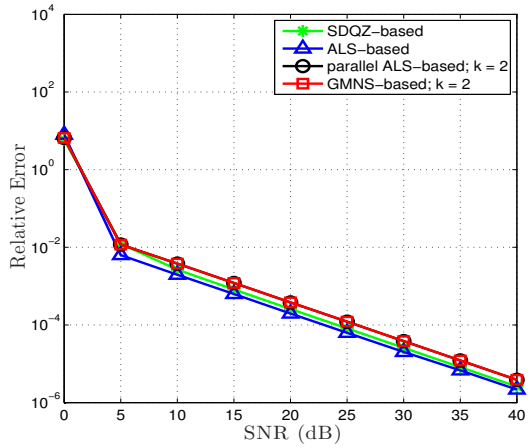
Figure 8: Performance of randomized GMNS algorithm on data matrices with $kp > n$; $k = 2$, $n = 150$, $m = 500$.

6.2.1 Effect of noise:

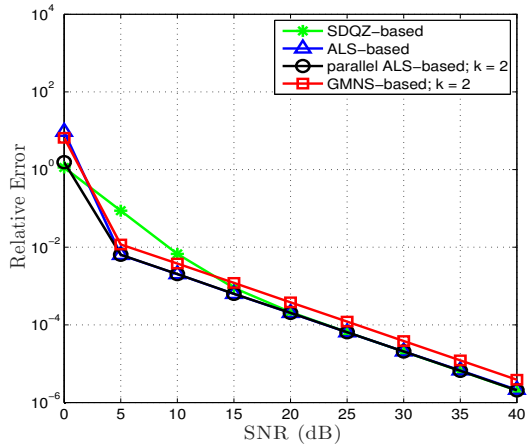
We study the effect of noise on the performance of the PARAFAC algorithms at different values of SNR. The tested tensor has size of $100 \times 100 \times 120$ and rank of 10. As shown in Figure 9, the proposed GMNS-based PARAFAC algorithm performed similarly to the other ALS-based PARAFAC algorithms. At low SNR ($\leq$ 15dB), they were all better than the SDQZ-based PARAFAC. At high SNR, all algorithms yielded almost the same results in terms of relative estimation error.



(a) Loading matrix **A**



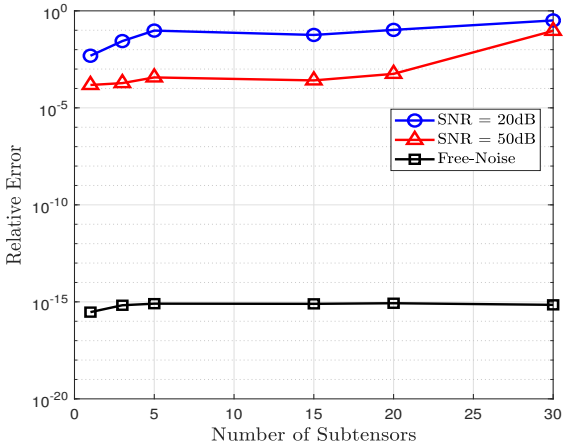
(b) Loading matrix **B**



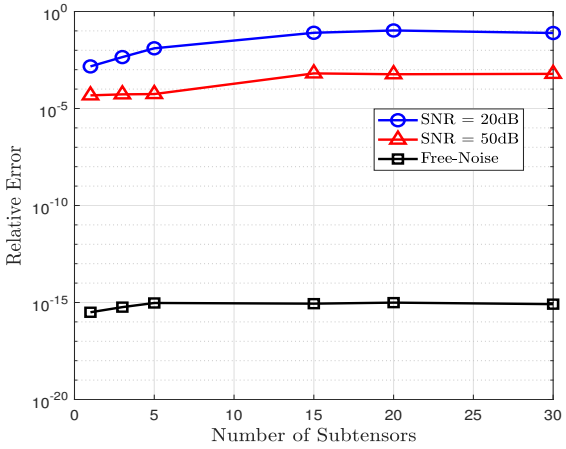
(c) Loading matrix **C**

Figure 9: Effect of noise on performance of PARAFAC algorithms; tensor size $= 50 \times 50 \times 60$, rank $R = 5$.

6.2.2 Effect of the number of sub-tensors, k:
Consider two tensors with size of $50 \times 50 \times 60$ and $100 \times 100 \times 120$. The SNRs were fixed at 20 dB and 50 dB. Assume that the tensors are divided into sub-tensors by splitting the loading matrix **C**. The number of sub-tensors varied in the range $[1, k/\text{rank}(\boldsymbol{\mathcal{X}})]$,



(a) Loading matrix **C** for $\boldsymbol{\mathcal{X}}_1$ of size $50 \times 50 \times 60$

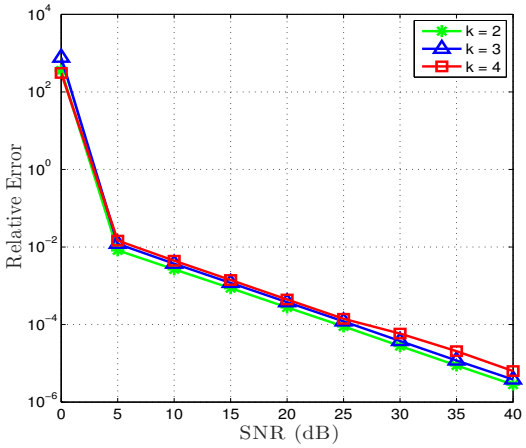


(b) Loading matrix **C** for $\boldsymbol{\mathcal{X}}_2$ of size $100 \times 100 \times 120$

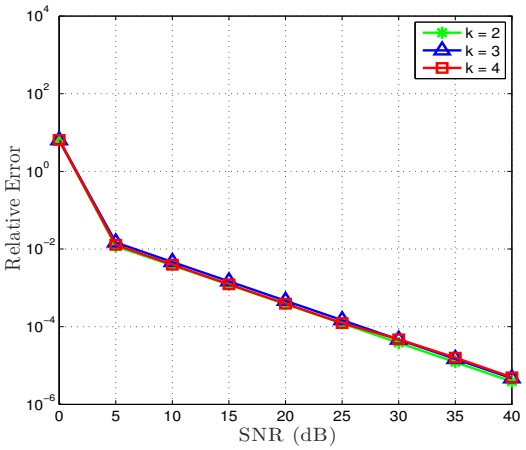
Figure 10: Effect of number of sub-tensors on performance of GMNS-based PARAFAC algorithm; tensor rank $R = 5$.

while still being required to maintain the conditions of uniqueness of PARAFAC. The experimental results are shown in Figures 10 and 11. It can be seen that, in general, the higher the number of sub-tensors was, the lower the performance of the GMNS-based PARAFAC algorithm, with or without noise. Intuitively, this is a trade-off between complexity and accuracy over the number of DSP units. However, the difference was little.

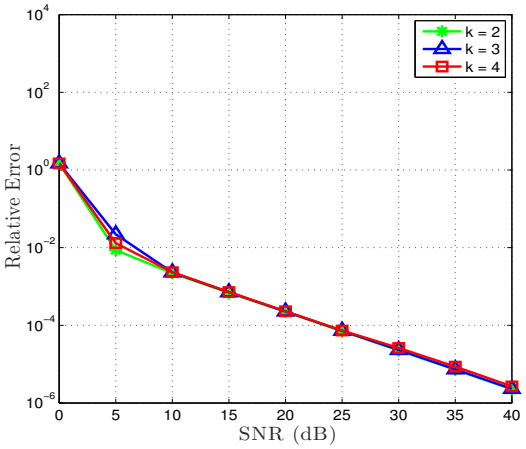
6.2.3 Effect of the tensor rank, R:
Consider two tensors with size of $50 \times 50 \times 60$ and $100 \times 100 \times 120$. The number of sub-tensors was fixed at $k = 2$. The results are shown in Figure 12. Generally, the higher the rank of the tensor was, the lower the performance of the GMNS-based PARAFAC algorithm. Under the effect of noise, the algorithm still yielded a reasonable estimation accuracy for tensors of small rank; $R(\boldsymbol{\mathcal{X}}_1) < 30$ or $R(\boldsymbol{\mathcal{X}}_2) < 50$. However, there was an unprecedented rise in error if the tensor rank became greater than a specific threshold of n/k. Therefore, choosing k plays a vital role in decomposing a tensor with a given rank.



(a) Loading matrix **A**

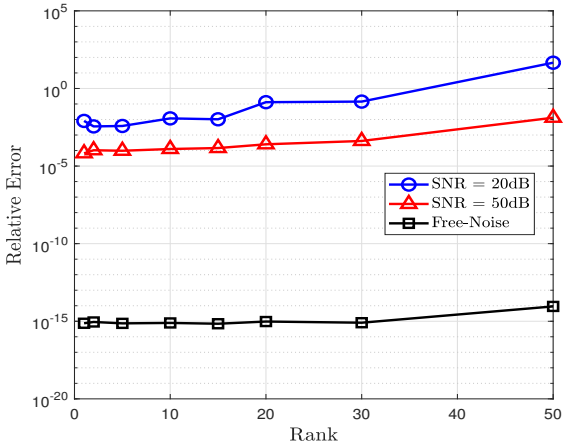


(b) Loading matrix **B**

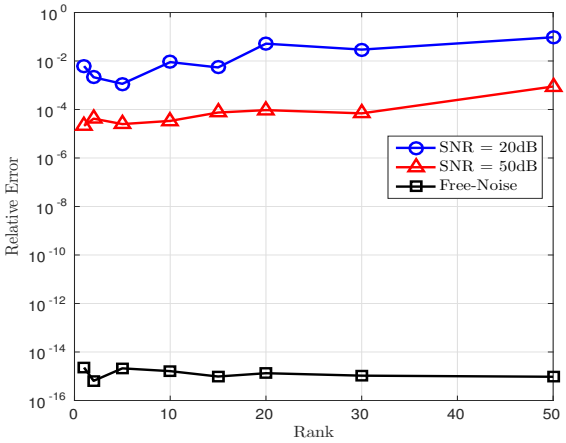


(c) Loading matrix **C**

Figure 11: Effect of number of sub-tensors on performance of GMNS-based PARAFAC algorithm; tensor size $= 50 \times 50 \times 60$, rank $R = 5$.



(a) Loading matrix **C** for $\boldsymbol{\mathcal{X}}_1$ of size $50 \times 50 \times 60$



(b) Loading matrix **C** for $\boldsymbol{\mathcal{X}}_2$ of size $100 \times 100 \times 120$

Figure 12: Effect of tensor rank, R, on performance of GMNS-based PARAFAC algorithm.

6.3 GMNS-based HOSVD

To study the performance of the proposed GMNS-based HOSVD, we investigate three main application-based scenarios:

- Best low-rank tensor approximation;
- Tensor-based principal subspace estimation;
- Tensor-based dimensionality reduction.

6.3.1 Best low-rank tensor approximation:

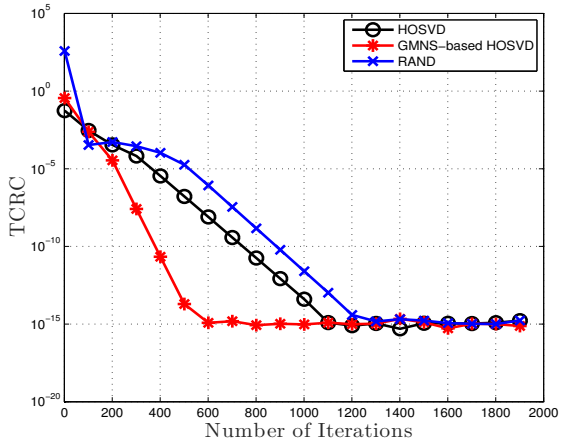
A performance comparison of Tucker decomposition with different initialization methods is provided via simulation study. In particular, we consider three algorithms to initialize loading factors: original HOSVD, GMNS-based HOSVD and a method that factors are chosen randomly (legend = RAND) [7]. After that, the alternating least square (ALS) algorithm is applied to obtain the best low-rank approximation of tensors.

Two performance metrics are used: tensor core relative change (TCRC) and subspace relative change (SRC). They are defined as

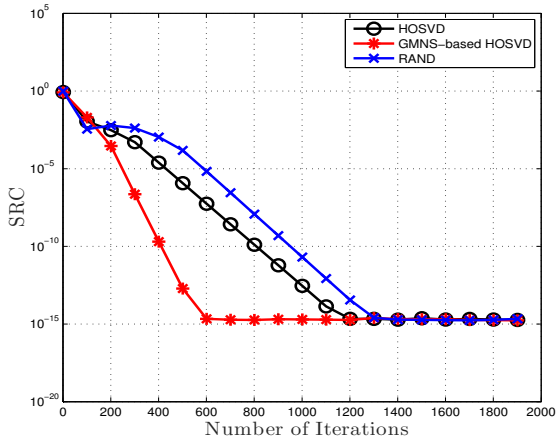
$$\text{TCRC}(k) = \frac{\|\boldsymbol{\mathcal{G}}^{(k)} - \boldsymbol{\mathcal{G}}^{(k-1)}\|}{\|\boldsymbol{\mathcal{G}}^{(k-1)}\|}, \tag{25}$$

$$\text{SRC}(k) = \frac{\sum_{i=1}^{N} \|\mathbf{U}_i^{(k)}(\mathbf{U}_i^{(k)})^T - \mathbf{U}_i^{(k-1)}(\mathbf{U}_i^{(k-1)})^T\|}{\sum_{i=1}^{N} \|\mathbf{U}_i^{(k-1)}(\mathbf{U}_i^{(k-1)})^T\|}, \tag{26}$$

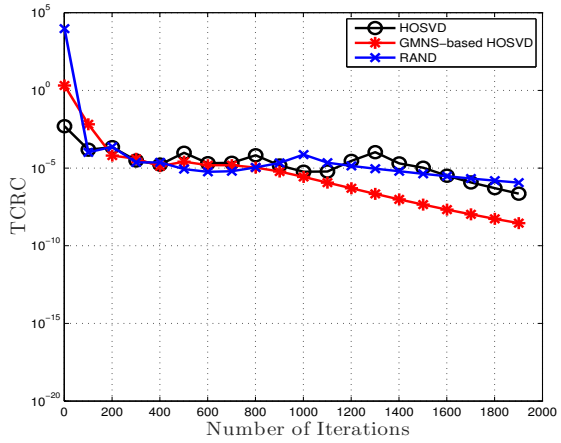
where N is the number of modes (fibers), $\boldsymbol{\mathcal{G}}_{(k)}$ and $\mathbf{U}_i^{(k)}$ are the estimated tensor core and the factors at the k-th iteration step.



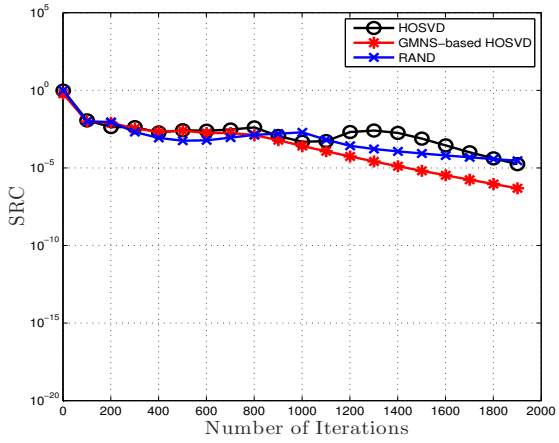
(a) TCRC for $\mathcal{X}_1$ of size $50 \times 50 \times 50$



(b) SRC for $\mathcal{X}_1$ of size $50 \times 50 \times 50$

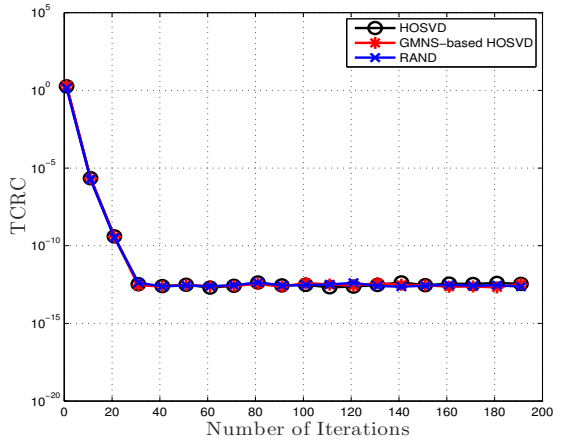


(c) TCRC for $\mathcal{X}_2$ of size $400 \times 400 \times 400$

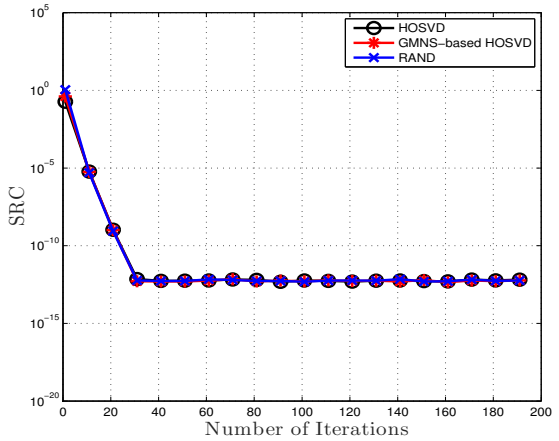


(d) SRC for $\mathcal{X}_2$ of size $400 \times 400 \times 400$

Figure 13: TCRC and SRC performance of Tucker decomposition algorithms on random tensors, $\mathcal{X}_1$ and $\mathcal{X}_2$, associated with a core tensor $\mathcal{G}_1$ size of $5 \times 5 \times 5$.



(a) TCRC for $\mathcal{X}_3$ of size $128 \times 128 \times 648$



(b) SRC for $\mathcal{X}_3$ of size $128 \times 128 \times 648$

Figure 14: TCRC and SRC performance of Tucker decomposition algorithms on real tensor obtained from Coil20 database [31]; $\mathcal{X}_3$ of size $128 \times 128 \times 648$ associated with tensor core $\mathcal{G}_2$ of size $64 \times 64 \times 100$.

We used three tensors to assess algorithm performance: two synthetic tensors and one real tensor from the Coil20 database [31]. The two synthetic tensors, $\mathcal{X}_1$ of size $50 \times 50 \times 50$ and $\mathcal{X}_2$ of size $400 \times 400 \times 400$, were randomly generated from the zero-mean and unit-variance Gaussian distribution. They were then com-

pressed into a tensor core $\mathcal{G}_1$ of $5 \times 5 \times 5$. The Coil20 database is composed of 9 subjects with 72 different images. We formed a real tensor $\mathcal{X}_3$ of size $128 \times 128 \times 648$, associated with a tensor core $\mathcal{G}_2$ of size $64 \times 64 \times 100$.

The convergence results are shown in Figures 13 and 14. It can be seen that, for the small synthetic tensor, the GMNS-based HOSVD algorithm converged fastest, while still yielding a good performance, in terms of TCRC and SRC ($\approx 10^{-15}$). For the big synthetic tensor, all algorithms yielded similar performance, but the GMNS-based algorithm was faster in terms of convergence as compared to the original HOSVD and the random-based algorithms, as clearly shown in Figure 13(c)-(d). In the case of the real data, all algorithms yielded the same performance in terms of TCRC and SRC with fast convergence.

6.3.2 Tensor-based principal subspace estimation:
Tensor-based subspace estimation was introduced in [32], wherein it was proved that the HOSVD-based approach improved subspace estimation accuracy and was better than conventional methods, like SVD, if the steering matrix $\mathbf{A}$ satisfies some specific conditions. Inspired by this work, we wanted to see how the proposed GMNS-based HOSVD algorithm works for principal subspace estimation. We also compared with the original HOSVD, the modified GMNS, and SVD.

For the sake of simplicity, we assume that the measurement $\mathbf{X}$ can be expressed by matrix and tensor representations as

$$\mathbf{X} = \mathbf{A}\mathbf{S} + \sigma\mathbf{N},$$
$$\mathcal{X} = \mathcal{A} \times_{R+1} \mathbf{S}^T + \sigma\mathcal{N},$$

where the steering matrix $\mathbf{A}$ and the tensor $\mathcal{A}$ can be expressed by two sub-systems $\mathbf{A}_1$ and $\mathbf{A}_2$ as

$$\mathbf{A} = \mathbf{A}_1 \odot \mathbf{A}_2,$$
$$\mathcal{A} = \mathcal{I} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2.$$

The multidimensional version of the true subspace $\mathbf{W}$ in the matrix case can be defined as

$$\mathcal{U} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2, \tag{27}$$

where $\mathcal{G}$ denotes the core of tensor $\mathcal{X}$, $\mathbf{U}_1$ and $\mathbf{U}_2$ are two (truncated) loading factors derived by a specific algorithm for Tucker decomposition, such as the original HOSVD, the GMNS-based HOSVD, and the HOOI algorithms.

In this work, we follow the experiment set up in [32]. The array steering tensor $\mathcal{A}$ and the signal $\mathbf{S}$ were derived from the random zeros-mean and unit-variance Gaussian distribution, similarly in Section 6.1. The experimental results are shown in Figure 15. It can be seen that the GMNS-based HOSVD algorithm for PSA yielded almost the same subspace estimation accuracy in terms of SEP as the HOSVD-based, SVD-based and GMNS-based algorithms. Thus, the proposed GMNS-based HOSVD algorithm can be useful for subspace-based parameter estimation.

6.3.3 Tensor based dimensionality reduction:
We investigated the use of GMNS-based HOSVD, the truncated HOSVD (T-HOSVD), another truncated

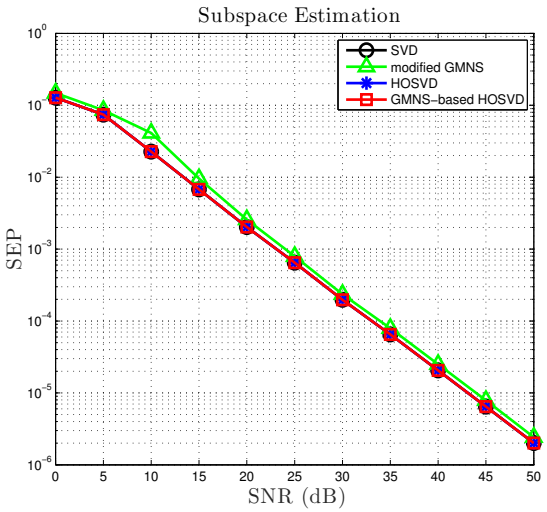


Figure 15: HOSVD for PSA.

HOSVD [33] (ST-HOSVD), and SVD for compression of an image tensor with a fixed rank. The image tensor was obtained from the Coil20 database.

The root mean square error (RMSR) is used as the performance metric and is defined as

$$\text{RMSE} = \frac{\|\mathbf{A}_{\text{re}} - \mathbf{A}_{\text{ex}}\|}{\|\mathbf{A}_{\text{ex}}\|}, \tag{28}$$

where $\mathbf{A}_{\text{ex}}$ and $\mathbf{A}_{\text{re}}$ are the true and reconstructed images, respectively.

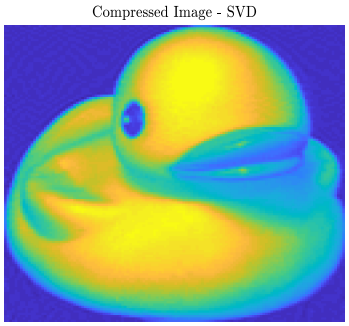
The results are shown in Figure 16. Clearly, GMNS-based HOSVD yielded the same performance as the truncated HOSVD but slightly worse ($\backsimeq 0.2\%$ in terms of RMSE) than ST-HOSVD. The tensor-based approach for dimensionality reduction was much worse than the SVD-based approach on each single image.

7 Conclusions

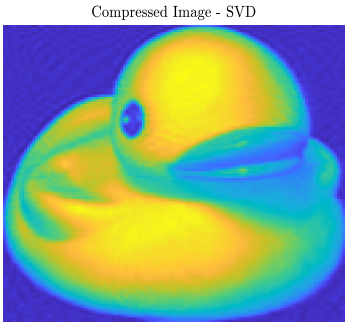
In this paper, motivated by the advantages of the GMNS method, we proposed several new algorithms for principal subspace analysis and tensor decomposition. We first introduced the modified and the randomized GMNS-based algorithms for PSA with reasonable subspace estimation accuracy. Based on these, we proposed two GMNS-based algorithms for PARAFAC and HOSVD. Numerical experiments indicated that our proposed algorithms can be a suitable alternative to their counterparts, as they can significantly reduce the computational complexity while yielding reasonable performance.

8 Acknowledgments

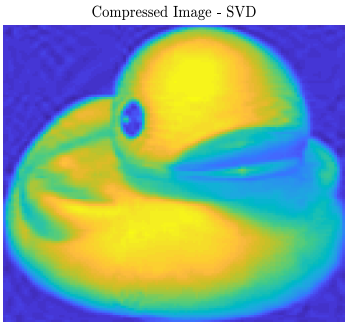
This research was funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2015.32.



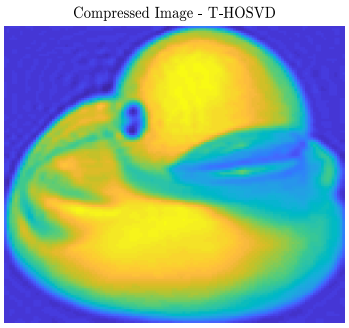
(a) SVD: $n = 40$, RMSE $= 0.0036$



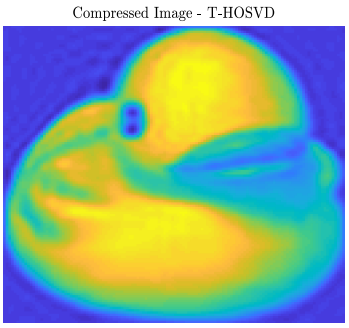
(b) SVD: $n = 30$, RMSE $= 0.0059$



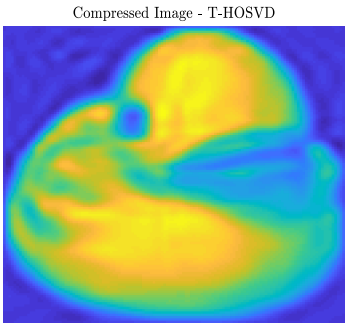
(c) SVD: $n = 20$, RMSE $= 0.01$



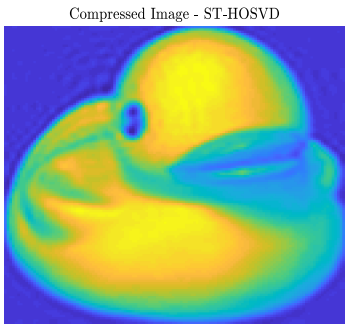
(d) T-HOSVD: $n = 40$, RMSE $= 0.0129$



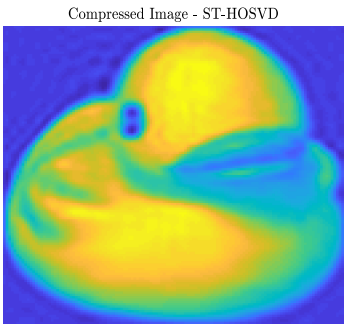
(e) T-HOSVD: $n = 30$, RMSE $= 0.0166$



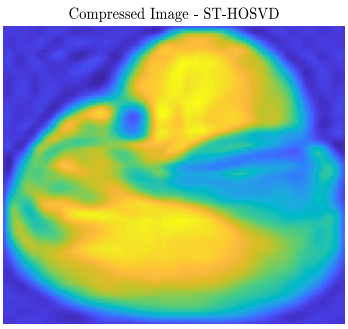
(f) T-HOSVD: $n = 20$, RMSE $= 0.0328$



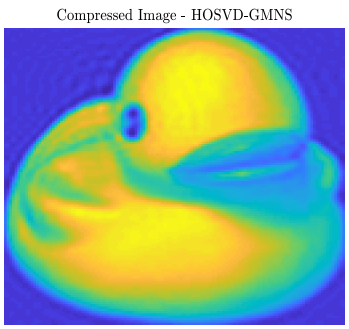
(g) ST-HOSVD: $n = 40$, RMSE $= 0.0127$



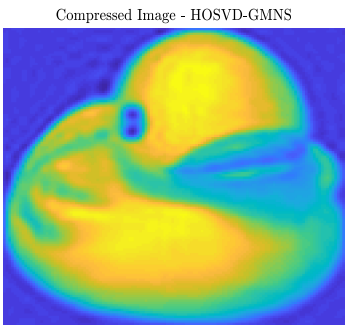
(h) ST-HOSVD: $n = 30$, RMSE $= 0.0164$



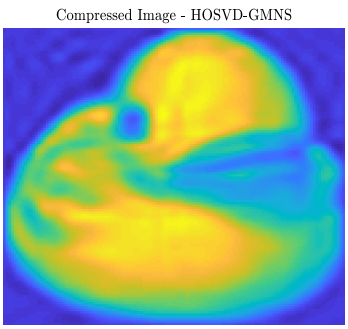
(i) ST-HOSVD: $n = 20$, RMSE $= 0.0326$



(j) GMNS-based HOSVD: $n = 40$, RMSE $= 0.0129$



(k) GMNS-based HOSVD: $n = 30$, RMSE $= 0.0166$



(l) GMNS-based HOSVD: $n = 20$, RMSE $= 0.0328$

Figure 16: Image compression using SVD and different Tucker decomposition algorithms.

References

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile networks and applications*, vol. 19, no. 2, pp. 171–209, 2014.

[2] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

[3] C.-F. V. Latchoumane, F.-B. Vialatte, J. Solé-Casals, M. Maurice, S. R. Wimalaratna, N. Hudson, J. Jeong, and A. Cichocki, "Multiway array decomposition analysis of EEGs in Alzheimer's disease," *Journal of neuroscience methods*, vol. 207, no. 1, pp. 41–50, 2012.

[4] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of EEG signals: a brief review," *Journal of neuroscience methods*, vol. 248, pp. 59–69, 2015.

[5] V. D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast tensor decompositions for big data processing," in *2016 International Conference on Advanced Technologies for Communications (ATC)*, Oct 2016, pp. 215–221.
[6] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.
[7] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
[8] L. Tran, C. Navasca, and J. Luo, "Video detection anomaly via low-rank and sparse decompositions," in *2012 Western New York Image Processing Workshop (WNYIPW)*. IEEE, 2012, pp. 17–20.
[9] X. Zhang, X. Shi, W. Hu, X. Li, and S. Maybank, "Visual tracking via dynamic tensor analysis with mean update," *Neurocomputing*, vol. 74, no. 17, pp. 3277–3285, 2011.
[10] H. Li, Y. Wei, L. Li, and Y. Y. Tang, "Infrared moving target detection and tracking based on tensor locality preserving projection," *Infrared Physics & Technology*, vol. 53, no. 2, pp. 77–83, 2010.
[11] S. Bourennane, C. Fossati, and A. Cailly, "Improvement of classification for hyperspectral images based on tensor modeling," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 4, pp. 801–805, 2010.
[12] N. Renard and S. Bourennane, "Dimensionality reduction based on tensor modeling for classification methods," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1123–1131, 2009.
[13] H. Fanaee-T and J. Gama, "Event detection from traffic tensors: A hybrid model," *Neurocomputing*, vol. 203, pp. 22–33, 2016.
[14] V. D. Nguyen, K. Abed-Meraim, N. Linh-Trung, and R. Weber, "Generalized minimum noise subspace for array processing," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3789–3802, July 2017.
[15] A. H. Phan and A. Cichocki, "PARAFAC algorithms for large-scale problems," *Neurocomputing*, vol. 74, no. 11, pp. 1970–1984, 2011.
[16] A. L. de Almeida and A. Y. Kibangou, "Distributed computation of tensor decompositions in collaborative networks," in *2013 IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2013, pp. 232–235.
[17] A. L. De Almeida and A. Y. Kibangou, "Distributed large-scale tensor decomposition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 26–30.
[18] V. D. Nguyen, K. Abed-Meraim, and L. T. Nguyen, "Parallelizable PARAFAC decomposition of 3-way tensors," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 5505–5509.
[19] K. Shin, L. Sael, and U. Kang, "Fully scalable methods for distributed tensor factorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 100–113, Jan 2017.
[20] D. Chen, Y. Hu, L. Wang, A. Y. Zomaya, and X. Li, "H-PARAFAC: Hierarchical parallel factor analysis of multidimensional big data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1091–1104, April 2017.
[21] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.
[22] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.
[23] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.
[24] D. P. Woodruff, "Sketching as a Tool for Numerical Linear Algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.
[25] V. Rokhlin, A. Szlam, and M. Tygert, "A randomized algorithm for principal component analysis," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1100–1124, 2009.
[26] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 687–717, 2014.
[27] A. R. Benson, D. F. Gleich, and J. Demmel, "Direct QR factorizations for tall-and-skinny matrices in MapReduce architectures," in *2013 IEEE International Conference on Big Data*, Oct 2013, pp. 264–272.
[28] N. Kishore Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear and Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, 2017.
[29] B. W. Bader, T. G. Kolda *et al.*, "MATLAB Tensor Toolbox Version 2.6," Available online, February 2015. [Online]. Available: http://www. sandia.gov/ tgkolda/TensorToolbox/
[30] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, 2006.
[31] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia University Image Library (COIL-20)," 1996. [Online]. Available: http://www.cs.columbia.edu/ CAVE/ software/ softlib/coil-20.php
[32] M. Haardt, F. Roemer, and G. Del Galdo, "Higher-order SVD-based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3198–3213, 2008.
[33] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, "A new truncation strategy for the higher-order singular value decomposition," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A1027–A1052, 2012.

Le Trung Thanh received his B.Eng. degree in Electronics and Telecommunications from VNU University of Engineering and Technology (VNU-UET), Vietnam National University, Hanoi (VNU), Vietnam in 2016. He is now a master student at VNU-UET and a research assistant at the Advanced Institute of Engineering and Technology (AVITECH) within VNU-UET. His research interests include graph signal processing, subspace tracking and tensor analysis.

Viet-Dung Nguyen received the B.Sc. degree from the University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam, in 2009, the M.Sc. degree in network and telecommunication from the Université Paris XI and the Ecole Normale Supérieure, Cachan, France, in 2012, and the Ph.D. degree from the Université d'Orléans, Orléans, France, in 2016, in the field of signal processing. Currently, he is a Postdoctoral Research Fellow in the Signals and Systems Laboratory, University of Paris-Saclay, Paris, France. His current research interests include: wireless communication for Internet-of-Things, matrix and tensor decompositions, adaptive signal processing, blind source separation, array signal processing, and statistical performance analysis.

Nguyen Linh-Trung obtained his B.Eng. and Ph.D. degrees, both in Electrical Engineering, from Queensland University of Technology, Brisbane, Australia. Since 2006, he has been on the faculty of VNU University of Engineering and Technology (VNU-UET), a member university of Vietnam National University, Hanoi (VNU), where he is currently an associate professor of electronic engineering in the Faculty of Electronics and Telecommunications and director of the Advanced Institute of Engineering and Technology (AVITECH) within VNU-UET. He is interested in signal processing methods, including time-frequency signal analysis, blind source separation, compressive sampling, tensor-based signal analysis, graph signal processing, and apply them to wireless communication and networking, biomedical engineering, with a current focus on large-scale processing.

Karim Abed-Meraim was born in 1967. He received the State Engineering Degree from Ecole Polytechnique, Paris, France, in 1990, the State Engineering Degree from Ecole Nationale Supérieure des Télécommunications (ENST), Paris, France, in 1992, the M.Sc. degree from Paris XI University, Orsay, France, in 1992, and the Ph.D. degree from the ENST in 1995 (in the field of signal processing and communications). From 1995 to 1998, he took a position as a Research Fellow at the Electrical Engineering Department, University of Melbourne, where he worked on research project related to "Blind System Identification for Wireless Communications" and "Array Processing for Communications." From 1998 to 2012, he has been an Assistant then an Associate Professor at the Signal and Image Processing Department, Telecom ParisTech. In September 2012, he joined the University of Orléans (PRISME Laboratory) as a Full Professor. He has also been a visiting scholar at the Centre of Wireless Communications (National University of Singapore) in 1999, at the EEE Department of Nanyang Technological University (Singapore) in 2001, at Telecom Malaysia Research and Development Centre in 2004, at the School of Engineering and Mathematics of Edith Cowan University (Perth, Australia) in 2004, at the EEE Department of the National University of Singapore in 2006, at Sharjah University (UAE) in 2008-2009, and at King Abdullah University of Science and Technology (KSA) in 2013 and 2014. He is the author of about 400 scientific publications including book chapters, international journal, and conference papers and patents. His research interests are related to statistical signal processing with application to communications, system identification, adaptive filtering and tracking, radar and array processing, biomedical signal processing, and statistical performance analysis.