Math. Commun. **31** (2026), 1–23
DOI: 10.1000/100

Mathematical Communications

# Randomized algorithms for coupled decompositions

Erna Begović Kovač[1,*], Anita Carević[2], and Ivana Šain Glibić[3]

[1] University of Zagreb, Faculty of Chemical Engineering and Technology, Marulićev trg 19, 10 000 Zagreb, Croatia.
[2] University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, Ruđera Boškovića 32, 21 000 Split, Croatia.
[3] University of Zagreb, Faculty of Science, Department of Mathematics, Bijenička 30, 10 000 Zagreb, Croatia.

**Abstract.** Coupled decompositions are a widely used tool for data fusion. As the volume of data increases, so does the dimensionality of matrices and tensors, highlighting the need for more efficient coupled decomposition algorithms. This paper studies the problem of coupled matrix factorization (CMF), where two matrices represented in low-rank form share a common factor. Additionally, it explores coupled matrix and tensor factorization (CMTF), where a matrix and a tensor are represented in low-rank form, also sharing a common factor matrix. We show that these problems can be solved using a direct approach with singular value decomposition (SVD), rather than relying on an iterative method. Knowing that matrices coming from real-world applications are often very large, the computational cost can be substantial. To address this issue and improve the efficiency, we propose new techniques for randomizing these algorithms. This includes a novel strategy for selecting a projection subspace that takes into account the contribution from both matrices involved in the decomposition equally. We present extensive results of numerical tests that confirm the efficiency of our algorithms. Furthermore, as a novel approach and with a high success rate, we apply our randomized algorithms to the face recognition problem.

**AMS subject classifications**: 68W20, 65F55
**Keywords**: coupled decompositions; randomized algorithms; randomized SVD; randomized subspace iteration; randomized block Krylov iteration; face recognition
Received April 8, 2025; accepted April 7, 2026

## 1. Introduction

Coupled decompositions of multiple data sets are broadly used in different engineering disciplines as a tool for data fusion. They are utilized in the analysis of data coming from different sources, for example, to better describe data obtained by different technologies or methods. To name a few applications, coupled decompositions appear in chemometrics [24, 3, 2, 30], signal processing [31, 27], bioinformatics [4, 5], metabolimics [1, 36, 13], chromatography [26], etc. In this paper, we apply them to the problem of face recognition. To the best of our knowledge, this is a novel approach. A standard procedure for the face recognition algorithm is to calculate the mean face image, derive the covariance matrix, extract its principal components, and use them to project the images onto a lower-dimensional subspace, as explained in [37, 10, 35], and generalized to tensors in [32, 8, 7]. However, the coupled decomposition bypasses this procedure and achieves dimensionality reduction by extracting the common part of the images.

Coupled matrix factorization (CMF) [23] decomposes a set of matrices in a way that they are represented in a low-rank format sharing one common factor. For two matrices $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$, their coupled rank-$k$ approximation is given by

$$X \approx UV^T \quad \text{and} \quad Y \approx UW^T, \tag{1}$$

where $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$.

Big and complex data sets are often represented by tensors rather than matrices. Therefore, in addition to the matrix-matrix case, it is also important to consider tensor-matrix factorization, generalizing problem (1). There are various algorithms for coupled matrix and tensor factorization (CMTF). An often-used approach is the alternating least squares (ALS) method [24, 2, 3, 27, 25, 21, 22], which is an iterative method

*Email addresses:* ebegovic@fkit.unizg.hr (E. Begović Kovač), carevica@fesb.hr (A. Carević), ivanasai@math.hr (I. Šain Glibić)
*Corresponding author.

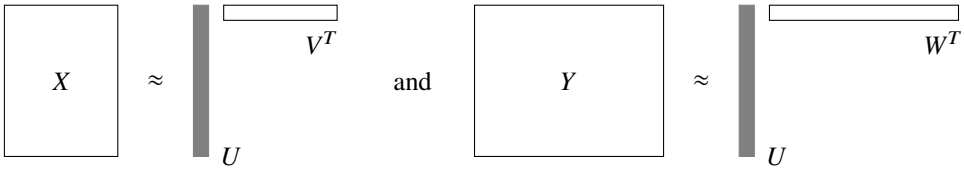


Figure 1: Graphical description of CMF.

that may encounter convergence issues. We are going to show how CMTF of the tensor $\mathcal{X} \in \mathbb{R}^{m\times n_2\times n_3}$ and matrix $Y \in \mathbb{R}^{m\times n}$ can be expressed in terms of CMF of tensor matricization $X_{(1)} \in \mathbb{R}^{m\times n_2 n_3}$ and matrix $Y$. Consequently, CMTF is simplified and solved using the singular value decomposition of the matrix $\begin{bmatrix} X_{(1)} & Y \end{bmatrix} \in \mathbb{R}^{m\times(n_2n_3+n)}$, instead of the alternating least squares technique. In this way, we do not obtain the full decomposition, as is the case with ALS, but we achieve the shared subspace faster and avoid potential convergence problems. Since the aforementioned factorizations can be time-consuming for large-scale matrices and tensors, we look into a way to overcome this issue.

In recent years, randomized algorithms have experienced a breakthrough in numerical linear algebra [9, 34, 6, 14, 15, 17]. Randomized algorithms are known to be significantly faster than deterministic algorithms, and they are reliable in many applications. In the context of coupled decompositions, randomized projections as an option for dealing with large data sets were briefly mentioned in [25]. However, this paper thoroughly analyzes the potential of randomization in such decompositions, emphasizing the requirement of considering all matrices equally. We develop a randomized algorithm for coupled decomposition, inspired by the randomized SVD [9], which has proven effective for a single matrix. Furthermore, we study more refined forms of randomization, similar to randomized subspace iteration [9, 28] and randomized block Krylov iteration [28], for the case of coupled decompositions. These approaches are tested on different examples, for both CMF and CMTF, and the results are compared mutually, as well as with non-randomized algorithms. In addition to synthetic numerical examples, the algorithms are also tested on the face recognition problem. Given a database of faces, our randomized algorithms for coupled decompositions match a new face with a person from the database. Although this is not a traditional approach to face recognition, our algorithms demonstrated a very good success rate.

In summary, the key contributions of this paper are as follows:

- A direct approach based on SVD to solve the coupled rank-$k$ approximation problem for two matrices and for matrix-tensor pairs is proposed.
- A new strategy for constructing randomized projections in problems involving two matrices is developed. Based on this strategy, randomized algorithms for CMF and CMTF are created.
- As a novel approach, our randomized algorithms for coupled decompositions are applied to the face recognition problem.

The paper is organized as follows. In Section 2, we study coupled matrix factorization and introduce our basic algorithm (Algorithm 1). This algorithm is randomized in Subsection 2.1, while numerical examples are given in Section 3. Then, we analyse coupled matrix and tensor factorization in Section 4. The Tucker tensor decomposition is examined in Subsection 4.2 and CP tensor decomposition in Subsection 4.3. The corresponding numerical examples are presented in Section 5. In Section 6, we apply our algorithms to the problem of face recognition. We end the paper with a short conclusion in Section 7.

## 2. Coupled matrix factorization

We start with the coupled matrix factorization (CMF). Let $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$. The goal of CMF is to find a coupled rank-$k$ approximation of $X$ and $Y$ in the form given in relation (1). To solve this problem, we need to minimize the objective function

$$f(U,V,W) = \|X - UV^T\|_F^2 + \|Y - UW^T\|_F^2 \to \min. \tag{2}$$

Before we construct an algorithm for solving the minimization problem (2), we first show that the approximation problem (1) is equivalent to the low-rank approximation of one matrix.

**Theorem 1.** *Let $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$. Let $\Sigma_k \in \mathbb{R}^{k\times k}$ be a diagonal matrix of the largest $k$ singular values of the matrix $\begin{bmatrix} X & Y \end{bmatrix} \in \mathbb{R}^{m\times(n_1+n_2)}$, and let $U_k \in \mathbb{R}^{m\times k}$ and $V_k \in \mathbb{R}^{(n_1+n_2)\times k}$ contain left and right singular vectors of $\begin{bmatrix} X & Y \end{bmatrix}$, respectively, corresponding to those singular values. Then, a solution to the minimization problem (2) is defined by $U_k$, $V_k$ and $\Sigma_k$.*

*Precisely, the objective function $f$ defined in (2) attains its minimum for $U = U_k$, $V$ equal to the first $n_1$ rows of $V_k\Sigma_k$, and $W$ equal to the remaining $n_2$ rows of $V_k\Sigma_k$.*

*Proof.* Let $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$ be arbitrary matrices. For fixed $X \in \mathbb{R}^{m\times n_1}$, $Y \in \mathbb{R}^{m\times n_2}$, using the properties of the Frobenius norm, we get

$$\|X - UV^T\|_F^2 + \|Y - UW^T\|_F^2 = \| \begin{bmatrix} X & Y \end{bmatrix} - \begin{bmatrix} UV^T & UW^T \end{bmatrix} \|_F^2 = \| \begin{bmatrix} X & Y \end{bmatrix} - UZ^T\|_F^2,$$

where $Z = \begin{bmatrix} V \\ W \end{bmatrix}$. Hence,

$$\min_{U,V,W} \left\{\|X - UV^T\|_F^2 + \|Y - UW^T\|_F^2\right\} = \min_{U,Z} \| \begin{bmatrix} X & Y \end{bmatrix} - UZ^T\|_F^2. \tag{3}$$

The left-hand side in (3) is the best coupled rank-$k$ approximation of $X$ and $Y$, while the right-hand side corresponds to the best rank-$k$ approximation of the matrix $\begin{bmatrix} X & Y \end{bmatrix}$.

It is well known that a solution to the minimization problem on the right-hand side of (3) is given by the truncated SVD,

$$\begin{bmatrix} X & Y \end{bmatrix} \approx U_k\Sigma_k V_k^T,$$

where $U_k, V_k, \Sigma_k$ are as in the statement of the theorem. Then,

$$\min_{U,Z} \| \begin{bmatrix} X & Y \end{bmatrix} - UZ^T\|_F^2 = \| \begin{bmatrix} X & Y \end{bmatrix} - \bar{U}\bar{Z}^T\|_F^2,$$

where $\bar{U} = U_k$ and $\bar{Z}^T = \Sigma_k V_k^T$, that is, $\bar{Z} = V_k\Sigma_k$.
Therefore, it follows from (3) that

$$\begin{aligned}\min_{U,V,W} \left\{\|X - UV^T\|_F^2 + \|Y - UW^T\|_F^2\right\} &= \| \begin{bmatrix} X & Y \end{bmatrix} - \bar{U}\bar{Z}^T\|_F^2 \\ &= \|X - \bar{U}\bar{V}^T\|_F^2 + \|Y - \bar{U}\bar{W}^T\|_F^2,\end{aligned}$$

for matrices $\bar{V}$ and $\bar{W}$ obtained by splitting the matrix $\bar{Z}$ into two parts, such that $\bar{V}$ contains the first $n_1$ rows of $Z$, while $\bar{W}$ contains the remaining $n_2$ rows of $\bar{Z}$. □

**Corollary 1.** *The best coupled rank-$k$ approximation of two matrices $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$ is equivalent to the best rank-$k$ approximation of the matrix $\begin{bmatrix} X & Y \end{bmatrix} \in \mathbb{R}^{m\times(n_1+n_2)}$.*

*Proof.* The proof follows directly from Theorem 1 and relation (3). □

Now, using Theorem 1 we can write the algorithm for solving the minimization problem (2). Algorithm 1 is our baseline for the coupled matrix factorization. In the following subsection, we are going to incorporate different randomization techniques.

## 2.1. Randomized CMF

The core of an efficient randomization technique for CMF determines the projections of matrices $X$ and $Y$ onto the same subspace, as the accuracy of the result depends on it, i.e., how well it approximates the joint subspace of the direct sum $\text{range}(X) + \text{range}(Y)$. Given Theorem 1, a straightforward approach to selecting the projection matrix $Q$ is to compute the basis for the subspace $\text{range}\left(\begin{bmatrix} X & Y \end{bmatrix}\right)$. However, we propose a more refined method. Specifically, we first determine the bases $Q_1$ and $Q_2$ for the subspaces $\text{range}(X)$ and $\text{range}(Y)$, respectively. Then, $Q$ is obtained by reorthogonalizing the columns of the matrix $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$.

**Algorithm 1** CMF — Basic algorithm

**Input:** $X \in \mathbb{R}^{m\times n_1}$, $Y \in \mathbb{R}^{m\times n_2}$, $k < \min\{n_1, n_2\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$
$[U_{XY}, \Sigma, V_{XY}] = \texttt{svd}([X\ Y])$
$U = U_{XY}(:, 1:k)$
$Z = V_{XY}(:, 1:k)\Sigma(1:k, 1:k)$
$V = Z(1:n_1, :)$
$W = Z(n_1+1:n_1+n_2, :)$

The approach involves generating projections of $X$ and $Y$ onto the same subspace using a random Gaussian matrix $\Omega$:

$$\widehat{X} = \Pi_{X\Omega}X, \quad \widehat{Y} = \Pi_{Y\Omega}Y. \tag{4}$$

Subsequently, the coupled matrix factorization (CMF) of the reduced matrices $\widehat{X}$ and $\widehat{Y}$, which are significantly smaller than $X$ and $Y$, is computed. The detailed procedure is outlined below.

For $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$, we generate random Gaussian matrices $\Omega_1 \in \mathbb{R}^{n_1\times k}$ and $\Omega_2 \in \mathbb{R}^{n_2\times k}$. Then, we find the thin QR decompositions

$$Q_1R_1 = X\Omega_1, \quad Q_2R_2 = Y\Omega_2, \tag{5}$$

such that $Q_1, Q_2 \in \mathbb{R}^{m\times k}$. The matrices $Q_1$ and $Q_2$ form orthogonal bases of subspaces of range$(X)$ and range$(Y)$, respectively. To obtain a joint subspace of the direct sum range$(X)$+range$(Y)$, we reorthogonalize the columns of the matrix $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$. In this way, we get a matrix $Q$ whose columns form an orthogonal basis of a subspace of range$(X)$ + range$(Y)$. Matrix $Q$ has $m$ rows and $k + p$ columns, where $0 \le p \le k$. Notice that $p < k$ if the subspaces of range$(X)$ and range$(Y)$ intersect. We can think of $p$ as an oversampling parameter.

When we have $Q$, we apply Algorithm 1 to $\widehat{X} = Q^TX \in \mathbb{R}^{(k+p)\times n_1}$ and $\widehat{Y} = Q^TY \in \mathbb{R}^{(k+p)\times n_2}$. It returns $\widehat{U} \in \mathbb{R}^{(k+p)\times k}$, $V = \widehat{V} \in \mathbb{R}^{n_1\times k}$, $W = \widehat{W} \in \mathbb{R}^{n_2\times k}$. Finally, we set $U = Q\widehat{U}$. Randomized CMF is presented in Algorithm 2. In our code, we truncate $Q$ by reducing the number of columns from $2k$ to $k + p$, if possible, by examining the diagonal elements of the triangular matrix from the QR decomposition.

**Algorithm 2** Randomized CMF

**Input:** $X \in \mathbb{R}^{m\times n_1}$, $Y \in \mathbb{R}^{m\times n_2}$, $k < \min\{n_1, n_2\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$
Generate random matrices $\Omega_1 \in \mathbb{R}^{n_1\times k}$ and $\Omega_2 \in \mathbb{R}^{n_2\times k}$.
$[Q_1, \sim] = \texttt{qr}(X\Omega_1, 0)$
$[Q_2, \sim] = \texttt{qr}(Y\Omega_2, 0)$
$[Q, \sim] = \texttt{qr}([Q_1, Q_2], 0)$
$[\widehat{U}, V, W] = \texttt{CMF}(Q^TX, Q^TY, k)$ ▷ Algorithm 1
$U = Q\widehat{U}$

As can be seen from the above discussion and relations (4) and (5), our randomization technique differs from the randomization of other matrix pair factorizations such as the generalized singular value decomposition [33]. Since we search for the joint subspace of a direct sum, our procedure includes both matrices to obtain the projection matrix $Q$.

We compare two approaches for generating projection matrices and compute

$$\|X - UV^T\|_F^2 + \|Y - UW^T\|_F^2,$$

which is the obtained minimum of the objective function (2). The first approach is as in Algorithm 2. In the second approach, the projection matrix $\bar{Q}$ is the orthogonal factor in the thin QR decomposition

$$\bar{Q}R = \begin{bmatrix} X & Y \end{bmatrix}\bar{\Omega}, \tag{6}$$

where $\bar{\Omega}$ is an $(n_1 + n_2) \times k$ random Gaussian matrix, and CMF is performed on $\bar{Q}^TX$ and $\bar{Q}^TY$.

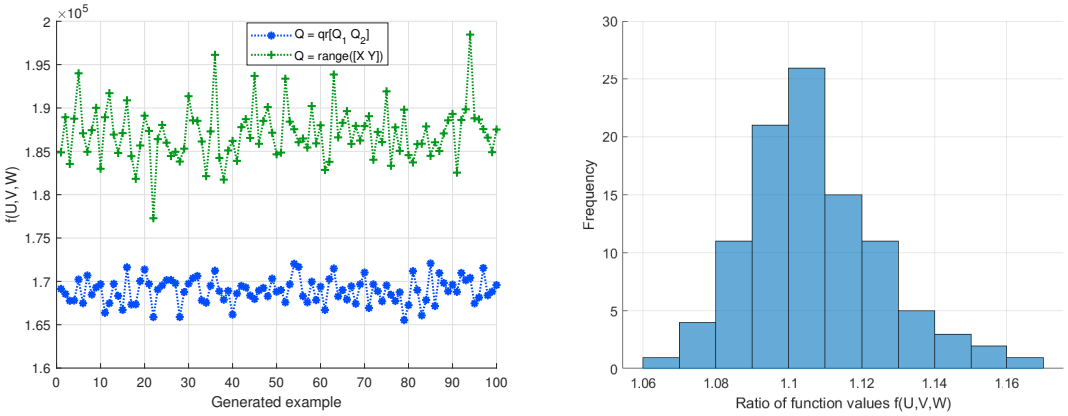


Figure 2: Comparison of two different approaches to computing the projection matrix $Q$ on 100 randomly generated examples.

The following set of tests demonstrates that our approach is superior and can significantly impact the outcome of the randomization algorithms. We generate low-rank matrices $X \in \mathbb{R}^{500\times 200}$ and $Y \in \mathbb{R}^{500\times 300}$ without imposing any specific structure on their singular values:

- `X = rand(500,100)*rand(100,200);`
- `Y = rand(500,150)*rand(150,300);`.

The target rank for the coupled decomposition is fixed at $k = 30$. We perform 100 experiments on different, randomly generated pairs of matrices $X$ and $Y$. The results are presented in Figure 2. It is clear that for every generated pair of matrices, our approach to choosing the projection matrix performs significantly better. In the majority of experiments, when the projection strategy is not applied, the value of the objective function $f(U, V, W)$ is around 10% or much larger than in the case when our projection strategy is used.

We are also going to explain this behavior from the theoretical point of view. Firstly, note that the approximation error obtained by randomization can be estimated using the result for the randomized SVD from [9].

**Theorem 2.** *[9, Theorem 10.5] Suppose that $A$ is a real $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}}$. Choose a target rank $k \geq 2$ and an oversampling parameter $p \geq 2$, where $k + p \leq \min\{m, n\}$. Set $S_k = \sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2$. Draw an $n \times (k + p)$ standard Gaussian matrix $\Omega$ and construct the sample matrix $A\Omega$. Then, for the projection $\Pi_{A\Omega}A$, the expected error satisfies the inequality*

$$\mathbb{E}\|A - \Pi_{A\Omega}A\|_F \leq \left(1 + \frac{k}{p-1}\right)^{\frac{1}{2}} S_k^{\frac{1}{2}}.$$

We have to be careful when using this result because neither of the two observed approaches completely satisfies the assumptions of Theorem 2. Precisely, our Gaussian matrices have $k$ columns, not $k + p$, while the effect of oversampling comes from the fact that the projection matrix $Q$ in Algorithm 2 contains more than $k$ columns. However, in order to explain the difference in these two approaches, in this discussion, we will make a minor modification. We are going to assume that $\Omega_1$, $Q_1$, $\Omega_2$, and $Q_2$ from (5), as well as $\bar{\Omega}$ and $\bar{Q}$ from (6), consist of $k + p$, $p \geq 2$, columns.

Then, the approximation of $X$ obtained via $\Omega_1$ and $Q_1$, denoted by $\Pi_{Q_1}X$, satisfies the assumptions of Theorem 2 and we have

$$\mathbb{E}\|X - \Pi_{Q_1}X\|_F \leq \left(1 + \frac{k}{p-1}\right)^{\frac{1}{2}} S_{X,k}^{\frac{1}{2}}, \tag{7}$$

where $S_{X,k}$ is the sum of the $\min\{m, n_1\} - k$ smallest singular values of $X$. Furthermore, for the orthogonal matrix $Q$ obtained by the thin QR decomposition, $QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$, the subspace spanned by the columns of $Q_1$ is a subset of the subspace spanned by the columns of $Q$. Thus, for the projection obtained via $Q$ and denoted by $\Pi_Q$, inequality

$$\|X - \Pi_Q X\|_F \leq \|X - \Pi_{Q_1}X\|_F \tag{8}$$

holds. Hence, it follows from relations (7) and (8) that

$$\mathbb{E}\|X - \Pi_Q X\|_F \leq \left(1 + \frac{k}{p-1}\right)^{\frac{1}{2}} S_{X,k}^{\frac{1}{2}}, \tag{9}$$

In the same way we get

$$\mathbb{E}\|Y - \Pi_Q Y\|_F \leq \left(1 + \frac{k}{p-1}\right)^{\frac{1}{2}} S_{Y,k}^{\frac{1}{2}}, \tag{10}$$

where $S_{Y,k}$ is the sum of the $\min\{m, n_2\} - k$ smallest singular values of $Y$. In other words, in the joint approximation obtained via $Q$, the approximation error in $X$ depends on its singular values, not on the singular values of $Y$, and vice versa.

On the other hand, if one took the projection $\Pi_{\bar{Q}}$ obtained via $\bar{Q}$, Theorem (2) would imply

$$\mathbb{E}\| \begin{bmatrix} X & Y \end{bmatrix} - \Pi_{\bar{Q}} \begin{bmatrix} X & Y \end{bmatrix} \|_F \leq \left(1 + \frac{k}{p-1}\right)^{\frac{1}{2}} \bar{S}_k^{\frac{1}{2}}, \tag{11}$$

for $\bar{S}_k = \sum_{j=k+1}^{\min\{m, n_1+n_2\}} \bar{\sigma}_j^2$, where $\bar{\sigma}_j$, $1 \leq j \leq \min\{m, n_1 + n_2\}$, are the singular values of $\begin{bmatrix} X & Y \end{bmatrix}$. That is, the approximation error in $X$, as well as in $Y$, does not depend directly on the singular values of the matrix in question, but on the singular values of $\begin{bmatrix} X & Y \end{bmatrix}$.

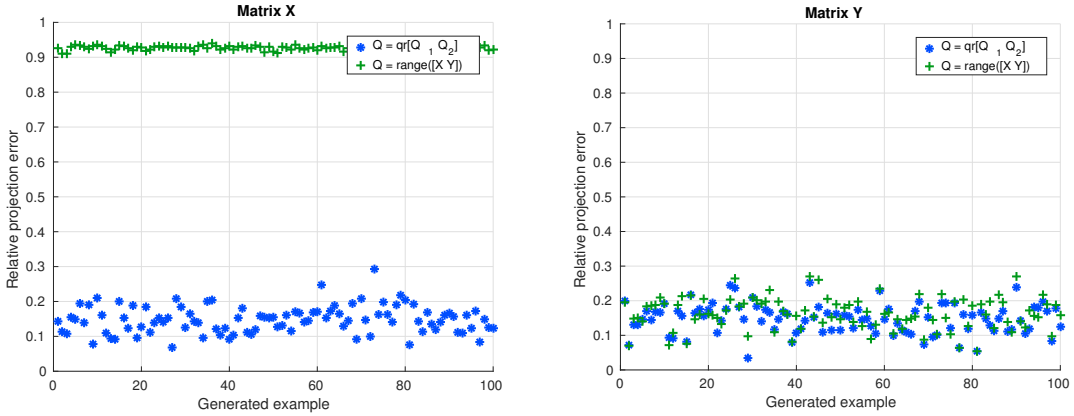


Figure 3: Relative projection errors for matrices $X$ and $Y$ on 100 randomly generated examples.

We have observed in our numerical examples that if the singular values of $X$ and $Y$ are significantly different in size, the approach described by (6) results in a much worse relative projection error for the matrix with smaller singular values, compared to the matrix with larger singular values. In contrast, the approach used in Algorithm 2 results in relative projection errors that are of the same order of magnitude. We performed the experiments on 100 pairs of matrices constructed as follows:

- `X = orth(rand(500,35))*diag(sort(rand(35,1)))*(orth(rand(200,35)))';`
- `Y = orth(rand(500,35))*diag(100*sort(rand(35,1)))*(orth(rand(300,35)))';`.

By construction, both matrices have rank 35. However, the singular values of `Y` are substantially larger in magnitude. The target rank for a coupled decomposition was set to $k = 30$. The results given in relations (9), (10), and (11) are illustrated in Figure 3. Our approach achieves a much better projection of the matrix $X$ compared to the method based on range($\begin{bmatrix} X & Y \end{bmatrix}$). As demonstrated earlier, this advantage stems from the fact that the singular values of $Y$ are much larger than the singular values of $X$, and they affect the error in $X$. When oversampling is incorporated into the algorithms for randomization, the overall performance improves slightly. However, the relative relationship between the errors remains unchanged. We must emphasize that the results shown in Figure 3 only show that we obtain better projections; they do not guarantee better final approximations.

Additionally, since Algorithm 2 corresponds to randomized SVD with $\Omega = \begin{bmatrix} \Omega_1 & 0 \\ 0 & \Omega_2 \end{bmatrix} \in \mathbb{R}^{(n_1+n_2)\times 2k}$, one could find it interesting to compare our randomization approach with randomized SVD of the form (6), but with $\bar{\Omega}$ of size $(n_1 + n_2) \times 2k$. In that case, the obtained approximation errors are very similar to those obtained by Algorithm 2, that is, the gap observed in Figure 2 disappears. In order to get matrix $\bar{\Omega}$, one has to perform a QR factorization on a matrix with $2k$ columns, while we do two QR factorizations on the matrices with $k$ columns and reorthogonalization of a matrix with $2k$ columns. Although our approach requires more computation in the case of the simple randomization from Algorithm 2, it becomes faster when the advanced randomization techniques are used. As will be explained in the next subsection, randomized subspace iterations and randomized block Krylov iterations use multiple QR decompositions. In our Algorithms 3 and 5, those are the decompositions of the matrices with only $k$, instead of $2k$, columns. Hence, our approach becomes faster, because computational cost of the QR decomposition scales quadratically with the number of columns.

## 2.2. Refinements of randomized CMF

In the case of a single matrix, it has been shown that, depending on the singular value distribution of the matrix, more sophisticated randomization techniques can yield greater efficiency. Therefore, we are going to generalize the algorithms for the randomized subspace iteration and the randomized block Krylov iteration from [28], so they can be used with CMF.

The first refinement of Algorithm 2 uses *randomized subspace iteration (RSI)*. Here, instead of obtaining the orthogonal basis $Q_1$ of range$(X)$ and $Q_2$ of range$(Y)$ like in (5), we have

$$Q_1 R_1 = (XX^T)^{q-1} X\Omega_1, \quad Q_2 R_2 = (YY^T)^{q-1} Y\Omega_2,$$

where $\Omega_1 \in \mathbb{R}^{n_1\times k}$ and $\Omega_2 \in \mathbb{R}^{n_2\times k}$ are random Gaussian matrices and $q$ is a depth parameter. Typically, $2 \le q \le 5$, see [28]. After we get $Q_1$ and $Q_2$, we proceed in the same way as in Algorithm 2. The procedure for CMF using RSI is given in Algorithm 3.

**Algorithm 3** RSI CMF

**Input:** $X \in \mathbb{R}^{m\times n_1}$, $Y \in \mathbb{R}^{m\times n_2}$, $k < \min\{n_1, n_2\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$

```
Generate random matrices Ω1 ∈ R^{n1×k} and Ω2 ∈ R^{n2×k}.
for 1 ≤ i ≤ q do
    [Q1, ~] = qr(XΩ1, 0)
    Ω1 = X^T Q1
    [Q2, ~] = qr(YΩ2, 0)
    Ω2 = Y^T Q2
end for
[Q, ~] = qr([Q1, Q2], 0)
[Û, V, W] = CMF(Q^T X, Q^T Y, k)        ▷ Algorithm 1
U = QÛ
```

In the *randomized block Krylov iteration* (RBKI), the idea is to project $X$ and $Y$ onto corresponding Krylov subspaces $\mathcal{K}_q(XX^T; X\Omega_1)$ and $\mathcal{K}_q(YY^T; Y\Omega_2)$ defined as

$$\mathcal{K}_q(XX^T; X\Omega_1) = \text{range}\begin{bmatrix} X\Omega_1 & (XX^T)X\Omega_1 & \dots & (XX^T)^{q-1}X\Omega_1 \end{bmatrix},$$
$$\mathcal{K}_q(YY^T; X\Omega_2) = \text{range}\begin{bmatrix} Y\Omega_2 & (YY^T)Y\Omega_2 & \dots & (YY^T)^{q-1}Y\Omega_2 \end{bmatrix}.$$

Matrices $\Omega_1 \in \mathbb{R}^{n_1\times \ell}$ and $\Omega_2 \in \mathbb{R}^{n_2\times \ell}$ are random Gaussian matrices. Parameter $\ell$ represents the block dimension, and $q$ is the order of the Krylov subspace. As the result, the RBKI method constructs the orthogonal bases $Q_1 \in \mathbb{R}^{m\times \ell q}$ of $\mathcal{K}_q(XX^T; X\Omega_1)$ and $Q_2 \in \mathbb{R}^{m\times \ell q}$ of $\mathcal{K}_q(YY^T; Y\Omega_2)$. In the literature [9, 28, 16], $\ell$ is usually $1, 2, k$ or $k + 4$. The parameter $q$ can vary, and we are going to discuss it further within the numerical examples. To define the projections onto range$(X)$ + range$(Y)$, we use a rank-revealing

QR decomposition and continue in the same way as described for Algorithm 2. Note that the projection defined for RBKI is usually much larger than the one in the RSI method, due to variability in $\ell$ and $q$. As a consequence, we expect a much more accurate approximation. We present the algorithm from [28] for finding the basis for the Krylov subspaces of order $q$ (Algorithm 4), together with our CMF algorithm using RBKI projection (Algorithm 5).

**Algorithm 4** RBKI [28]

**Input:** $A \in \mathbb{R}^{m\times n}$, block size $\ell$, iteration count $q$
**Output:** Matrix $Q \in \mathbb{R}^{m\times \ell q}$ representing basis for Krylov subspace $\mathcal{K}_q(AA^T; A\Omega_0)$

```
Generate random Ω_0 ∈ R^{n×ℓ}
for i = 1, ..., q do
    Q_i = AΩ_{i-1}
    Q_i = Q_i − Σ_{j<i} Q_j (Q_j^T Q_i)
    Q_i = Q_i − Σ_{j<i} Q_j (Q_j^T Q_i)        ▷ Reorthogonalization
    [Q_i, ~] = qr(Q_i, 0)
    Ω_i = A^T Q_i
end for
Q = [Q_1 ... Q_q]
```

**Algorithm 5** RBKI CMF

**Input:** $X \in \mathbb{R}^{m\times n_1}$, $Y \in \mathbb{R}^{m\times n_2}$, $\ell, q, k < \min\{n_1, n_2\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n_1\times k}$, $W \in \mathbb{R}^{n_2\times k}$

```
Q_1 = RBKI(X, ℓ, q)                ▷ Algorithm 4
Q_2 = RBKI(Y, ℓ, q)
[Q, ~] =qr([Q_1, Q_2], 0)          ▷ Q ∈ R^{n×2kq}
[U, V, W] =CMF(Q^T X, Q^T Y, k)    ▷ Algorithm 1
U = QU
```

Algorithms 1, 2, 3, and 5 will be compared by using numerical results will be presented in the next section.

## 3. Numerical examples for CMF

In this section, we construct numerical experiments to test the efficiency of our algorithms and compare the proposed randomized methods, i.e., Algorithm 2, Algorithm 3, and Algorithm 5. We use MATLAB 9.0.0.341360 (R2016a) in double precision (IEEE Standard 754).

The accuracy of our algorithms is determined by comparing the corresponding relative errors,

$$err_X = \frac{\|X - UV^T\|_F}{\|X\|_F}, \quad err_Y = \frac{\|Y - UW^T\|_F}{\|Y\|_F}. \tag{12}$$

The results are presented in the following way. The relative error for the basic algorithm (Algorithm 1) serves as the benchmark. We vary parameters for both the RSI and RBKI algorithms. In our tables, $k$ is the approximation rank, $p$ is the oversampling parameter, $\ell$ is the block dimension for RBKI, and $q$ is the depth parameter in RSI, that is, the order of the Krylov subspace in RBKI. For RSI, we report one selected result. We test the RBKI algorithm using the blocks of dimension $\ell = 1, 2, k$ and $k + 4$. This results in different oversampling parameters $p$. For each block dimension, we highlight one result that is close enough to the benchmark result, along with the corresponding oversampling parameter. Furthermore, for selected problems, for the block dimensions $\ell = 1$ and $\ell = 2$, we present the plots showing the relative errors as a function of the oversampling parameter. These plots demonstrate that even with a smaller oversampling parameter than the one shown in the table, the error remains close to that of the basic algorithm.

Let us present the test matrices and then analyze each example individually.

1. `SyntheticTest1`: We construct low rank matrices $X \in \mathbb{R}^{m\times n_1}, Y \in \mathbb{R}^{m\times n_2}$ with no special structure for singular values as in Section 2.1:
   - `X = rand(m,`$r_1$`)*rand(`$r_1$`,`$n_1$`);`
   - `Y = rand(m,`$r_2$`)*rand(`$r_2$`,`$n_2$`)`.
2. `SyntheticTest2`: Matrix $X \in \mathbb{R}^{n\times n}$ is constructed as in [20], i.e., for $d \in \mathbb{N}$:
   - $\Sigma_X = \mathrm{diag}(1, \ldots, 1, 2^{-d}, 3^{-d}, \ldots, (n-r+1)^{-d})$;
   - `UX = orth(rand(n)); VX = orth(rand(n));`
   - `X = UX*`$\Sigma_X$`*VX'`.

   Notice that $X$ is designed to have quickly decaying singular values. Next, the matrix $Y \in \mathbb{R}^{n\times n}$ is constructed so that parts of range$(X)$ and range$(Y)$ intersect. This is controlled with the parameter $c$:
   - $\Sigma_Y = \mathrm{diag}(1, \ldots, 1, 2^{-1}, 3^{-1}, \ldots, (n-2r+1)^{-1})$;
   - `UY = ([UX(:,1:c) orth(rand(n,n-c))];`
   - `VY = ([VX(:,1:c) orth(rand(n,n-c))];`
   - `Y = UY*`$\Sigma_Y$`*VY'`.
3. `SynthethicTest3`: Matrix $X \in \mathbb{R}^{m\times n}$ is constructed as in [20]. More precisely,

$$X = \sum_{j=1}^{r} \frac{10}{j} x_j^X \left(y_j^X\right)^T + \sum_{j=r+1}^{\min\{m,n\}} \frac{1}{j} x_j^X \left(y_j^X\right)^T,$$

   where $x_j^X \in \mathbb{R}^m$ and $y_j^X \in \mathbb{R}^n$ are sparse random vectors with density 0.25, created using MATLAB command `sprand`. We construct the matrix $Y \in \mathbb{R}^{m\times n}$ in a similar way, with the first $r$ random vectors $x_j^X \in \mathbb{R}^m$, $y_j^X \in \mathbb{R}^n$ the same as for $X$,

$$Y = \sum_{j=1}^{r} \frac{10}{j} x_j^X \left(y_j^X\right)^T + \sum_{j=r+1}^{\min\{m,n\}} \frac{1}{j} x_j^Y \left(y_j^Y\right)^T.$$

4. `SyntheticTest4`: Matrix $X \in \mathbb{R}^{m\times n_1}$ is defined as:
   - $\Sigma_X = \mathrm{diag}(2^{-i})_{i=1}^{n_1}$;
   - `X=orth(rand(m))*`$\Sigma_X$`;`

   while $Y \in \mathbb{R}^{m\times n_2}$ is a low-rank matrix with no special structure:
   - `Y = rand(m,`$r_2$`)*rand(`$r_2$`,`$n_2$`)`.
5. `SyntheticTest5`: Here, both $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$ are ill-conditioned with fast decaying singular values:
   - `A` = diag$(1, 2^{-1}, 2^{-2}, \ldots, 2^{-n_1})$;
   - `UA = orth(rand(m));`
   - `A = UA*A;`
   - `B` = diag$(1, 2^{-1}, 2^{-2}, \ldots, 2^{-n_2})$;
   - `UB = [UA(1:10) orth(rand(m,m-10))];`
   - `B = UB*B`.

## 3.1. SyntheticTest1

We consider `SyntheticTest1` with $m = 500$, $n_1 = 200$, $n_2 = 300$, $r_1 = 100$ and $r_2 = 150$. We are looking for the low-rank approximation of order $k = 30$. We present the performance of Algorithms 3 and 5. The selected results are given in Table 1. Figure 4 shows the relative errors for different oversampling parameters in RBKI. Notice that the relative error for $X$ initially decreases below the benchmark, then increases until it reaches the benchmark error. In contrast, the error for $Y$ consistently decreases. We assume that this happens because $X$ and $Y$ do not share any common features and it takes a number of iterations for the relative errors to balance.

| Algorithm | $p$ | $\ell$ | $q$ | $err_X$ | $err_Y$ |
|---|---|---|---|---|---|
| Basic CMF | - | - | - | $2.88218893 \cdot 10^{-2}$ | $2.03454946 \cdot 10^{-2}$ |
| RSI | 30 | - | 5 | $2.86340153 \cdot 10^{-2}$ | $2.06624393 \cdot 10^{-2}$ |
| RBKI | 112 | 1 | 71 | $2.88153938 \cdot 10^{-2}$ | $2.04048432 \cdot 10^{-2}$ |
| RBKI | 122 | 2 | 38 | $2.89287460 \cdot 10^{-2}$ | $2.03281104 \cdot 10^{-2}$ |
| RBKI | 90 | 30 | 2 | $2.83115828 \cdot 10^{-2}$ | $2.11079710 \cdot 10^{-2}$ |
| RBKI | 106 | 34 | 3 | $2.83986717 \cdot 10^{-2}$ | $2.09275944 \cdot 10^{-2}$ |

Table 1: Selected result for `SyntheticTest1` and $k = 30$. Oversampling parameters and relative errors for RSI with $q = 5$ and RBKI with $\ell = 1, 2, k, k + 4$.

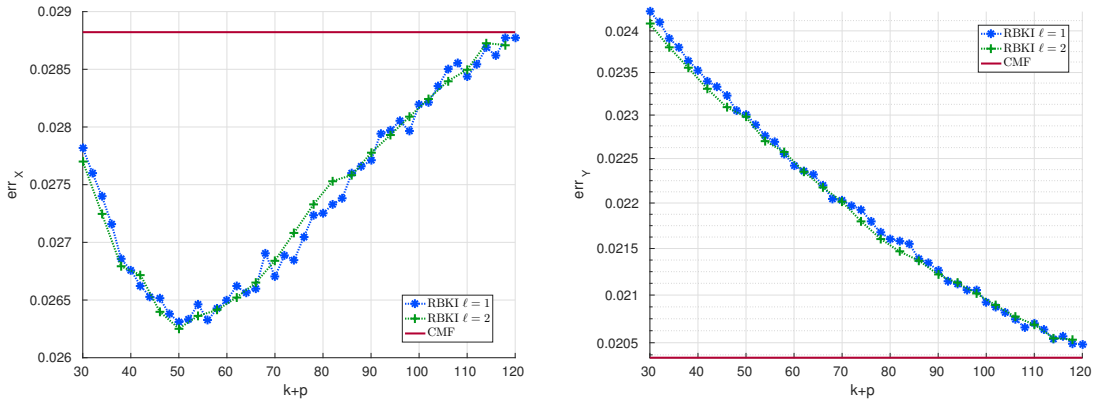


Figure 4: Relative errors with the respect to the dimension of the projection subspace in Algorithm 5 for `SyntheticTest1`.

## 3.2. SyntheticTest2

We consider $n = 1000$, $r = 15$, $d = 2$ and $c = 50$. We are looking for the low-rank approximation of order $k = 50$.

Selected results for Algorithms 2, 3, and 5 are presented in Table 2. We also give the errors obtained by the basic algorithm for the purpose of comparison. For this and the following example, we document the running time for each algorithm using MATLAB function `tic toc`. For Algorithms 2, 3, and 5 we present the total time, i.e., the time needed for constructing the projection matrix and running CMF on a smaller set of matrices. Separately, we present the time for the CMF algorithm itself. For the basic CMF, total time and CMF time are the same, since there is no preprocessing, the matrices are not projected. We can see that, as expected, the randomized algorithms are always faster than the basic CMF.

The oversampling parameter for RSI is $p = k = 50$, but we need $q = 4$ iterations to bring the relative error close to the one of the basic algorithm. The errors are slightly better for the RBKI algorithm. The oversampling parameters for RBKI are higher because of the way we construct the projection matrices. However, even a smaller oversampling parameter provides a good enough approximation, which is presented in Figure 5. Simple randomization, i.e., Algorithm 2, performed the worst, which is anticipated.

| Algorithm | $p$ | $\ell$ | $q$ | $err_X$ | $err_Y$ | total time (s) | CMF time (s) |
|---|---|---|---|---|---|---|---|
| Basic CMF | - | - | - | $3.25997151 \cdot 10^{-3}$ | $3.57247002 \cdot 10^{-2}$ | 0.288542 | 0.288542 |
| Randomized | 50 | - | - | $2.13494569 \cdot 10^{-3}$ | $4.19216919 \cdot 10^{-2}$ | 0.017870 | 0.009305 |
| RSI | 50 | - | 4 | $3.25992287 \cdot 10^{-3}$ | $3.57585534 \cdot 10^{-2}$ | 0.025313 | 0.008786 |
| RBKI | 84 | 1 | 67 | $3.25997444 \cdot 10^{-3}$ | $3.57247002 \cdot 10^{-2}$ | 0.125757 | 0.012734 |
| RBKI | 86 | 2 | 34 | $3.25997518 \cdot 10^{-3}$ | $3.57247006 \cdot 10^{-2}$ | 0.077220 | 0.012798 |
| RBKI | 150 | 50 | 2 | $3.25979677 \cdot 10^{-3}$ | $3.58010584 \cdot 10^{-2}$ | 0.044432 | 0.020858 |
| RBKI | 166 | 54 | 2 | $3.25980127 \cdot 10^{-3}$ | $3.57555301 \cdot 10^{-2}$ | 0.050801 | 0.024738 |

Table 2: Selected results for `SyntheticTest2` and $k = 50$. Oversampling parameters, relative errors for RSI with $q = 4$ and RBKI with $\ell = 1, 2, k, k + 4$, and running times.

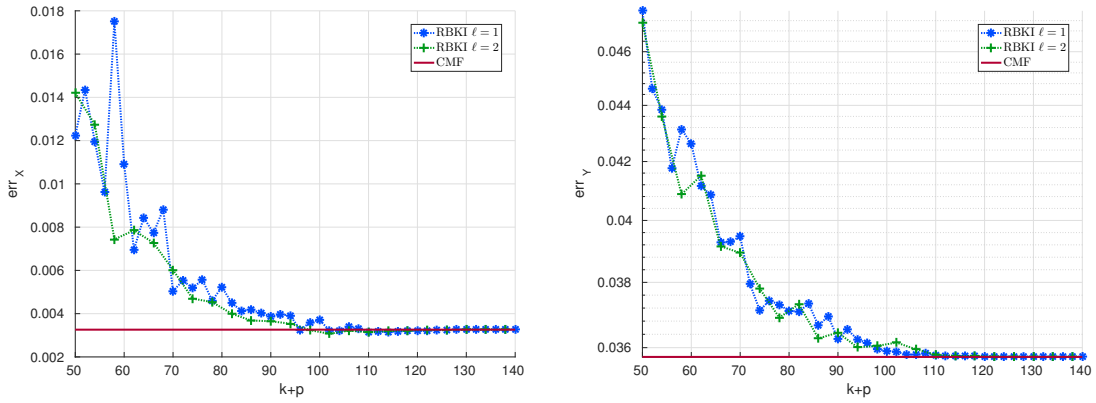


Figure 5: Relative errors with the respect to the dimension of the projection subspace in Algorithm 5 for `SyntheticTest2`.

## 3.3. SyntheticTest3

For this experiment, we consider $m = 10000$, $n = 500$ and $r = 50$. We are looking for the rank $k = 30$ approximation.

Selected results are presented in Table 3. Regarding the execution time, the difference between the randomized algorithms and Algorithm 1 is noticeable. This is because the randomized algorithms carry out CMF of much smaller matrices. For the RSI algorithm, we needed $q = 5$ iterations to match the benchmark error. The oversampling parameter in the RBKI method is greater. However, by examining the result, we can conclude that, like in the previous example, even a smaller oversampling parameter leads to a satisfactory result. We can also conclude that a small $\ell$ in RBKI gives as good a result as a larger $\ell$. An advantage of a smaller block size is a smaller oversampling parameter, resulting in smaller matrices on which CMF is performed. However, the total execution time is shorter for the larger blocks because the RBKI algorithm demands only two iterations. Again, Algorithm 2 has the worst performance.

| Algorithm | $p$ | $\ell$ | $q$ | $err_X$ | $err_Y$ | total time (s) | CMF time (s) |
|---|---|---|---|---|---|---|---|
| Basic CMF | - | - | - | $6.03560458 \cdot 10^{-2}$ | $6.03576711 \cdot 10^{-2}$ | 0.527464 | 0.527464 |
| Randomized | 30 | - | - | $6.15615406 \cdot 10^{-2}$ | $6.15000752 \cdot 10^{-2}$ | 0.054951 | 0.008305 |
| RSI | 30 | - | 5 | $6.03862191 \cdot 10^{-2}$ | $6.03867712 \cdot 10^{-2}$ | 0.127206 | 0.008582 |
| RBKI | 64 | 1 | 47 | $6.03560471 \cdot 10^{-2}$ | $6.03576722 \cdot 10^{-2}$ | 0.562032 | 0.006410 |
| RBKI | 74 | 2 | 26 | $6.03560459 \cdot 10^{-2}$ | $6.03576711 \cdot 10^{-2}$ | 0.426180 | 0.006848 |
| RBKI | 90 | 30 | 2 | $6.03685901 \cdot 10^{-2}$ | $6.03696584 \cdot 10^{-2}$ | 0.146570 | 0.008841 |
| RBKI | 106 | 34 | 2 | $6.03610827 \cdot 10^{-2}$ | $6.03625993 \cdot 10^{-2}$ | 0.127120 | 0.009377 |

Table 3: Selected results for `SyntheticTest3` and $k = 30$. Oversampling parameters, relative errors for RSI with $q = 5$ and RBKI with $\ell = 1, 2, k, k + 4$, and running times.

### 3.4. SyntheticTest4

Consider $m = 500$, $n_1 = 300$, $n_2 = 200$ and $r_2 = 100$. We want to find the rank $k = 30$ approximation.

Table 4 represents the selected results. Similarly to `SynthethicTest1`, matrices $X$ and $Y$ do not have anything in common; thus, the behavior is very much alike. The difference here is that it takes a larger oversampling parameter to obtain a good error for $Y$.

| Algorithm | $p$ | $\ell$ | $q$ | $err_X$ | $err_Y$ |
|---|---|---|---|---|---|
| Basic CMF | - | - | - | $4.86320647 \cdot 10^{-1}$ | $2.02998694 \cdot 10^{-2}$ |
| RSI | 30 | - | 8 | $4.86654497 \cdot 10^{-1}$ | $2.04024796 \cdot 10^{-2}$ |
| RBKI | 118 | 1 | 74 | $4.86320519 \cdot 10^{-1}$ | $2.02998694 \cdot 10^{-2}$ |
| RBKI | 118 | 2 | 37 | $4.86315021 \cdot 10^{-1}$ | $2.02998748 \cdot 10^{-2}$ |
| RBKI | 60 | 30 | 2 | $4.85925751 \cdot 10^{-1}$ | $2.09976041 \cdot 10^{-2}$ |
| RBKI | 68 | 34 | 2 | $4.85065718 \cdot 10^{-1}$ | $2.08038065 \cdot 10^{-2}$ |

Table 4: Selected results for `SyntheticTest4` and $k = 30$. Oversampling parameters and relative errors for RSI with $q = 8$ and RBKI with $\ell = 1, 2, k, k + 4$, and running times.

### 3.5. SyntheticTest5. Range intersection.

This example is constructed to demonstrate the importance of using a rank-revealing QR decomposition when constructing the projection matrix $Q$ in the randomized algorithms. This results in a possibly smaller oversampling parameter $p$, as discussed in Subsection 2.1.

We consider $m = 500, n_1 = 300, n_2 = 200$, and we want to find the approximation of rank $k = 30$. We separately analyze the influence of rank determination when using RBKI with block dimensions $\ell = 1$ and $\ell = 2$.

For $\ell = 1$, we iterate $q$ from 15 to 30 to get the oversampling parameter $p \leq (2q\ell - k)$. For $\ell = 2$, we iterate $q$ from 8 to 16 to get the oversampling parameter $p \leq (4q\ell - k)$. Figure 6 demonstrates the difference between the rank of the matrix $Q$ and the maximal dimension of the matrix $Q$. The impact of the rank-revealing QR should be more noticeable on large-dimensional matrices. Randomization itself contributes to the reduction of dimensionality and the CMF algorithm is executed on smaller matrices. QR with pivoting can further reduce that dimensionality.

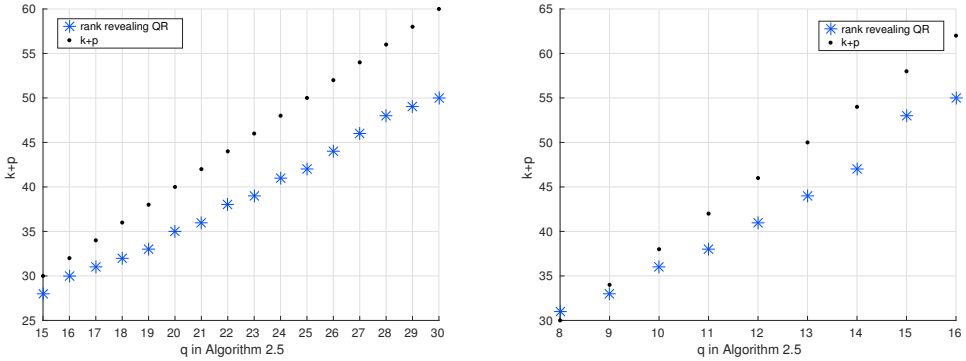


Figure 6: RBKI with $\ell = 1$ (on the left) and $\ell = 2$ (on the right). Rank-revealing QR leads to a smaller oversampling parameter.

Let us end this section with a remark on another well-known randomization technique—Generalized Nystrom (GN), [18].

*Remark* 1. Generalized Nystrom is a method that produces a rank-$k$ approximation of any matrix $X \in \mathbb{R}^{m \times n}$ using random matrices. We combined stabilized GN with CMF and tested it on a few examples. The results were compared with other methods presented in this paper. The comparison showed that CMF with GN is slower and less accurate. Other methods are faster because they first approximate matrices $X$ and

$Y$ with $QQ^TX$ and $QQ^TY$, respectively. By determining the matrix $Q$, we have already found one part of the matrix $U$ (which is a common part of matrices X and Y). What remains is to find CMF of smaller matrices $Q^TX$ and $Q^TY$. On the other hand, GN does not have this feature, which makes its combination with CMF slower. In addition, all methods were tested with Gaussian random matrices, which in the case of GN produces high errors. The results improved for GN when we replaced those matrices with a random subset of columns of the identity matrix, but they were still less accurate when compared to other methods. There is a possibility that more carefully chosen random matrices would produce more accurate results, but the question of algorithm speed remains.

## 4. Coupled matrix and tensor factorization

Coupled matrix and tensor factorization (CMTF) can be approached differently based on the tensor format. Here we consider two: the Tucker format in Subsection 4.2 and the CP format in Subsection 4.3. For the sake of simplicity, we assume that the tensors are of order three. Generalization on order-$d$, $d > 3$, tensors is direct but requires more complicated computations.

### 4.1. Tensor preliminaries

Throughout the text, we denote the tensors by calligraphic letters, e.g., $\mathcal{X}$. The *order* of a tensor is its dimension. Thus, $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is an order-3 tensor. Vectors obtained from a tensor by fixing all indices but the $n^{\text{th}}$ one are called *mode-n fibers*. *Mode-n matricization* (unfolding) is a matrix representation of a tensor acquired by arranging mode-$n$ fibers into a matrix. Mode-$n$ matricization of a tensor $\mathcal{X}$ is denoted by $X_{(n)}$. Mode-1 matricization of $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is an $n_1 \times n_2 n_3$ matrix. *Mode-n product* of a tensor $\mathcal{X}$ and a matrix $M$ is a tensor $\mathcal{T}$,

$$\mathcal{T} = \mathcal{X} \times_n M,$$

such that

$$T_{(n)} = MX_{(n)}. \tag{13}$$

For any matrices $M_1$ and $M_2$ of appropriate size, equality

$$\mathcal{X} \times_m M_1 \times_n M_2 = \mathcal{X} \times_n M_2 \times_m M_1, \quad \text{if } m \neq n, \tag{14}$$

holds. *Tensor norm* is a generalization of the Frobenius matrix norm,

$$\|\mathcal{X}\| = \sqrt{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} x_{ijk}^2}, \quad \mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}.$$

For this paper, we need two tensor decompositions. The first one is the *Tucker decomposition* [12, 29]. It is a decomposition of a tensor into a core tensor multiplied by a matrix in each mode. The Tucker decomposition of $\mathcal{X}$ takes the form

$$\mathcal{X} = \mathcal{S} \times_1 M_1 \times_2 M_2 \times_3 M_3.$$

*The multilinear rank* of $\mathcal{X}$ is a triplet $(r_1, r_2, r_3)$, where $r_i = \operatorname{rank}(X_{(i)})$, $i = 1, 2, 3$. If

$$\mathcal{X} \approx \mathcal{S} \times_1 V_1 \times_2 V_2 \times_3 V_3 \tag{15}$$

and $\mathcal{S} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, we say that (15) is a multilinear rank-$(r_1, r_2, r_3)$ approximation of $\mathcal{X}$.

The other decomposition of our interest is *CP decomposition* [12, 11]. It is a decomposition of a tensor $\mathcal{X}$ into a sum of rank-one tensors. A rank-one order-3 tensor is a tensor that can be written as an outer product of three vectors,

$$\mathcal{T} = a \circ b \circ c,$$

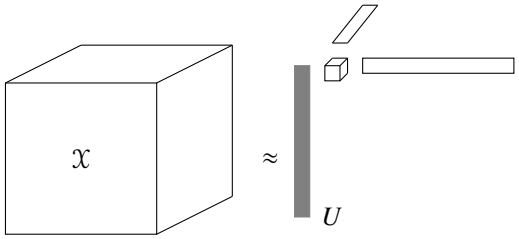


Figure 7: Graphical description of the tensor $\mathcal{X}$ from CMTF (19).

where $\circ$ stands for the outer product. Then, we write

$$\mathcal{X} = \sum_{i=1}^{R} a_i \circ b_i \circ c_i \equiv [[A, B, C]], \tag{16}$$

for $A = \begin{bmatrix} a_1 & a_2 & \cdots & a_R \end{bmatrix}$, $B = \begin{bmatrix} b_1 & b_2 & \cdots & b_R \end{bmatrix}$, $C = \begin{bmatrix} c_1 & c_2 & \cdots & c_R \end{bmatrix}$. *Tensor rank* is the smallest number $R$ in the CP decomposition (16) and

$$\mathcal{X} \approx \sum_{i=1}^{r} \tilde{a}_i \circ \tilde{b}_i \circ \tilde{c}_i \equiv [[\tilde{A}, \tilde{B}, \tilde{C}]] \tag{17}$$

is a rank-$r$ approximation of $\mathcal{X}$.

Before we proceed with CMTF, let us also define two matrix products that will be used in this section. The *Khatri-Rao product*, denoted by $\odot$, is a product of two matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times n}$,

$$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \cdots & a_n \otimes b_n \end{bmatrix} \in \mathbb{R}^{(mp)\times n},$$

where $a_k$ and $b_k$ denote the $k$th column of $A$ and $B$, respectively. An important property of CP decomposition is that if (16) holds, then

$$X_{(1)} = A(C \odot B)^T, \quad X_{(2)} = B(C \odot A)^T, \quad X_{(3)} = C(B \odot A)^T. \tag{18}$$

Moreover, the *Hadamard product*, denoted by $*$, of two matrices $A, B \in \mathbb{R}^{m\times n}$, is their element-wise product,

$$A * B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}.$$

## 4.2. Tensor in Tucker format

Let $\mathcal{X} \in \mathbb{R}^{m\times n_2\times n_3}$ and $Y \in \mathbb{R}^{m\times n}$. Joint rank-$k$ approximation of $\mathcal{X}$ and $Y$, coupled in the first mode and taking $\mathcal{X}$ in the Tucker format, as shown in Figure 7, is given by

$$\mathcal{X} \approx \mathcal{S} \times_1 U \times_2 V_2 \times_3 V_3, \quad Y \approx UW^T, \tag{19}$$

where $U \in \mathbb{R}^{m\times k}$, $V_2 \in \mathbb{R}^{n_2\times k}$, $V_3 \in \mathbb{R}^{n_3\times k}$, $W \in \mathbb{R}^{n\times k}$, $\mathcal{S} \in \mathbb{R}^{k\times k\times k}$, and a multilinear rank of $\mathcal{X}$ equals $(k, k, k)$.

Using the properties of mode-$n$ product (13) and (14), tensor $\mathcal{X}$ from (19) can be represented as

$$X_{(1)} \approx U(\mathcal{S} \times_2 V_2 \times_3 V_3)_{(1)}. \tag{20}$$

Thus, to get a solution to CMTF (19), we can observe CMF of tensor matricization $X_{(1)}$ and matrix $Y$. We get the approximations

$$X_{(1)} \approx UV^T, \quad Y \approx UW^T. \tag{21}$$

The problem of finding matrices $U$, $V$, and $W$ from (21) is the same as problem (1) and Algorithm 1 can be used. Now, the approximation $\widetilde{X}$ can be formed by folding $UV^T$ into a tensor. The procedure is given in Algorithm 6.

If there are $\mathcal{S}$, $V_2$, and $V_3$ of appropriate size, such that $V \in \mathbb{R}^{n_2 n_3 \times k}$ can be written as $V^T = (\mathcal{S} \times_2 V_2 \times_3 V_3)_{(1)}$, approximation reached in Algorithm 6 would be the best possible rank-$k$ approximation of the form (19). Although this cannot be guaranteed, a simple procedure in Algorithm 6 extracts the joint factor $U$, and returns a competitive approximation. Its application is presented in Section 6.

**Algorithm 6** CMTF — Basic algorithm for the Tucker format

**Input:** $\mathcal{X} \in \mathbb{R}^{m \times n_2 \times n_3}$, $Y \in \mathbb{R}^{m \times n}$, $k < \min\{n_2, n_2, n\}$
**Output:** $\widetilde{\mathcal{X}} \in \mathbb{R}^{m \times n_2 \times n_3}$, $U \in \mathbb{R}^{m \times k}$, $W \in \mathbb{R}^{n \times k}$
$[U, V, W] = \texttt{CMF}(X_{(1)}, Y, k)$ ▷ Algorithm 1
$\widetilde{X}_{(1)} = UV^T$
Fold $\widetilde{X}_{(1)}$ into $\widetilde{\mathcal{X}}$.

In order to randomize Algorithm 6, one should use one of the Algorithms 2, 3, or 5 in place of Algorithm 1 (within Algorithm 6). Note that, compared to CMF, the sizes of the random Gaussian matrices created for the randomized CMTF will be larger. Instead of projection (4), we have

$$\widehat{X}_{(1)} = \Pi_{X_{(1)}\Omega} X_{(1)}, \quad \widehat{Y} = \Pi_{Y\Omega} Y.$$

That is, for $\mathcal{X} \in \mathbb{R}^{m \times n_2 \times n_3}$ and $Y \in \mathbb{R}^{m \times n}$, $\Omega_1$ is an $n_2 n_3 \times k$ and $\Omega_2$ is an $n \times k$ matrix. Except for this difference, the randomization process is the same. Numerical results of randomized algorithms are discussed in Section 5.

We have only considered joint approximation coupled in the first mode. Instead, one can be interested in approximation coupled in a different way. In that case, relations (19) and (20) are modified accordingly. For example, if we are looking for an approximation of the form

$$\mathcal{X} \approx \mathcal{S} \times_1 V_1 \times_2 U \times_3 V_3, \quad Y \approx UW^T,$$

we need to represent the tensor $\mathcal{X}$ using mode-2 matricization,

$$X_{(2)} \approx U(\mathcal{S} \times_1 V_1 \times_3 V_3)_{(2)}.$$

This can be written as CMF of $X_{(2)}$ and $Y$, very similar to (21),

$$X_{(2)} \approx UV^T, \quad Y \approx UW^T,$$

which is then used in Algorithm 6. For the coupling in the third mode, we proceed in the same way, but using the mode-3 matricization

$$X_{(3)} \approx U(\mathcal{S} \times_1 V_1 \times_2 V_2)_{(3)}$$

and CMF of $X_{(3)}$ and $Y$. The coupling mode depends on the particular application and nature of the observed problem.

## 4.3. Tensor in CP format

Next, we describe the CMTF problem where a tensor is represented in CP format. Again, let $\mathcal{X} \in \mathbb{R}^{m \times n_2 \times n_3}$ and $Y \in \mathbb{R}^{m \times n}$. Then,

$$\mathcal{X} \approx [[U, B, C]], \quad Y \approx UW^T, \tag{22}$$

where $U \in \mathbb{R}^{m \times k}$, $B \in \mathbb{R}^{n_2 \times k}$, $C \in \mathbb{R}^{n_3 \times k}$, $W \in \mathbb{R}^{n \times k}$, is the joint rank-$k$ factorization of $\mathcal{X}$ and $Y$ coupled in the first mode and taking $\mathcal{X}$ in CP format, as shown in Figure 8. Note that for CP format we consider the tensor rank as given in relation (17).

In order to find the coupled decomposition (22), we first define the objective function

$$f(U, B, C, W) = \|\mathcal{X} - [[U, B, C]]\|^2 + \|Y - UW^T\|_F^2 \to \min. \tag{23}$$

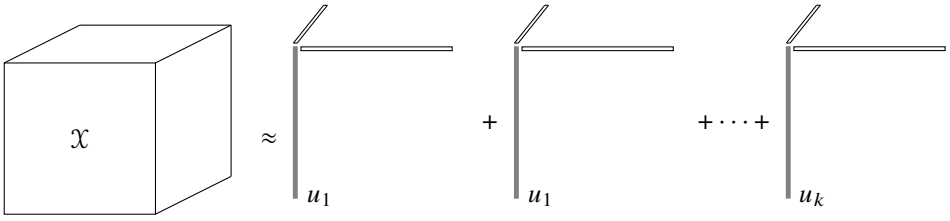


Figure 8: Graphical description of the tensor $\mathcal{X}$ from CMTF (22).

A common way for solving the optimization problem (23) is the alternating least squares (ALS) method. That is an iterative method where one iteration is made of several, in this case four, microiterations. In each microiteration, optimization is done over one of the matrices $U, B, C, W$, while the others are fixed. Let us see how the ALS algorithm for solving (23) is developed.

We can write the function $f$ as

$$f(U,B,C,W) = f_1(U,B,C,W) + f_2(U,B,C,W),$$

where

$$\begin{aligned} f_1(U,B,C,W) &= \|\mathcal{X} - [[U,B,C]]\|^2, \\ f_2(U,B,C,W) &= \|Y - UW^T\|_F^2. \end{aligned}$$

It follows from [2, Corollary 4.2] that

$$\begin{aligned} \frac{\partial f_1}{\partial U} &= -2X_{(1)}(C \odot B) + 2U\Gamma^{(1)}, \\ \frac{\partial f_1}{\partial B} &= -2X_{(2)}(C \odot U) + 2B\Gamma^{(2)}, \\ \frac{\partial f_1}{\partial C} &= -2X_{(3)}(B \odot U) + 2C\Gamma^{(3)}, \end{aligned}$$

for

$$\Gamma^{(1)} = (B^TB) * (C^TC), \quad \Gamma^{(2)} = (U^TU) * (C^TC), \quad \Gamma^{(3)} = (U^TU) * (B^TB).$$

Obviously, $\frac{\partial f_1}{\partial W} = 0$. For the function $f_2$, after a short calculation, we get

$$\frac{\partial f_2}{\partial U} = -2YW + 2UW^TW, \quad \frac{\partial f_2}{\partial W} = -2Y^TU + 2WU^TU,$$

while $\frac{\partial f_2}{\partial B} = \frac{\partial f_2}{\partial C} = 0$. Furthermore, by setting the partial derivatives of $f$ to zero, we get

$$\begin{aligned} \frac{\partial f}{\partial U} &= -2X_{(1)}(C \odot B) + 2U\Gamma^{(1)} - 2YW + 2UW^TW = 0, \\ \frac{\partial f}{\partial B} &= -2X_{(2)}(C \odot U) + 2B\Gamma^{(2)} = 0, \\ \frac{\partial f}{\partial C} &= -2X_{(3)}(B \odot U) + 2C\Gamma^{(3)} = 0, \\ \frac{\partial f}{\partial W} &= -2Y^TU + 2WU^TU = 0. \end{aligned}$$

That is,

$$\begin{aligned} U &= (X_{(1)}(C \odot B) + YW)(\Gamma^{(1)} + W^TW)^{-1}, \\ B &= X_{(2)}(C \odot U)(\Gamma^{(2)})^{-1}, \\ C &= X_{(3)}(B \odot U)(\Gamma^{(3)})^{-1}, \\ W &= Y^TU(U^TU)^{-1}. \end{aligned}$$

**Algorithm 7** CMTF-CP ALS

**Input:** $X \in \mathbb{R}^{m\times n_2\times n_3}$, $Y \in \mathbb{R}^{m\times n}$, $k < \min\{n_2, n_3, n\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{n_2\times k}$, $C \in \mathbb{R}^{n_3\times k}$, $W \in \mathbb{R}^{n\times k}$
Initialize $U, B, C, W$.
**repeat**
$U = X_{(1)}(C \odot B) + YW)(\Gamma^{(1)} + W^T W)^{-1}$
$B = X_{(2)}(C \odot U)(\Gamma^{(2)})^{-1}$
$C = X_{(3)}(B \odot U)(\Gamma^{(3)})^{-1}$
$W = Y^T U(U^T U)^{-1}$
**until** convergence

Now we can write the ALS algorithm for CMTF.

Randomization of Algorithm 7 is more complex than for Algorithm 1. Let $\mathcal{X} = [[A, B, C]]$ be a CP decomposition of $\mathcal{X} \in \mathbb{R}^{m\times n_2\times n_3}$ and let $Y \in \mathbb{R}^{m\times n}$. First, we generate random Gaussian matrices $\Omega_1 \in \mathbb{R}^{(n_2 n_3)\times k}$ and $\Omega_2 \in \mathbb{R}^{n\times k}$. We compute thin QR decompositions $Q_1 R_1 = X_{(1)}\Omega_1$ and $Q_2 R_2 = Y\Omega_2$, such that $Q_1, Q_2 \in \mathbb{R}^{m\times k}$, where $X_{(1)}$ is mode-1 matricization of $\mathcal{X}$. Then, as explained in Subsection 2.1, we reorthogonalize the columns of the matrix $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ to obtain $Q \in \mathbb{R}^{m\times(k+p)}$, where $p$ $(0 \le p \le k)$ is an oversampling parameter. Randomized CMTF-CP ALS is presented in Algorithm 8. Randomization using RSI and RBKI is done accordingly.

**Algorithm 8** Randomized CMTF-CP ALS

**Input:** $X \in \mathbb{R}^{m\times n_2\times n_3}$, $Y \in \mathbb{R}^{m\times n}$, $k < \min\{n_2, n_3, n\}$
**Output:** $U \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{n_2\times k}$, $C \in \mathbb{R}^{n_3\times k}$, $W \in \mathbb{R}^{n\times k}$
Generate random matrices $\Omega_1 \in \mathbb{R}^{(n_2\cdot n_3)\times k}$ and $\Omega_2 \in \mathbb{R}^{n\times k}$.
$[Q_1, \sim] = \texttt{qr}(X_{(1)}\Omega_1, 0)$
$[Q_2, \sim] = \texttt{qr}(Y\Omega_2, 0)$
$[Q, \sim] = \texttt{qr}([Q_1, Q_2], 0)$
$[\widehat{U}, B, C, W] = \texttt{CMTF}(\mathcal{X} \times_1 Q^T, Q^T Y)$ ▷ Algorithm 7
$U = Q\widehat{U}$

Since Algorithm 7 is an iterative algorithm, convergence issues may occur. Therefore, it is worth observing other options. Using relation (18), tensor $\mathcal{X}$ from (22) can be written as

$$X_{(1)} \approx U(C \odot B)^T.$$

Therefore, it makes sense to try to apply CMF to $X_{(1)}$ and $Y$, just as we do for the Tucker tensor format in (21). Hence, we can use Algorithm 6 and we get

$$X_{(1)} \approx UV^T, \quad Y \approx UW^T.$$

Again, we cannot be sure that there are $B$ and $C$ of appropriate sizes such that $V = C \odot B$. However, we get a joint factor $U$ and a numerical advantage of the approximation compared to Algorithm 7 is given in the next section.

Similarly to the Tucker tensor format, we have only examined coupling in the first mode. Results for the couplings in other modes follow the same reasoning, but use different matricizations of $\mathcal{X}$. In the case of the second mode, we need

$$\mathcal{X} \approx [[A, U, C]], \quad Y \approx UW^T,$$

and we use the relation

$$X_{(2)} = U(C \odot A)^T$$

to justify the use of CMF of $X_{(2)}$ and $Y$. On the other hand, if we need

$$\mathcal{X} \approx [[A, B, U]], \quad Y \approx UW^T,$$

the use of CMF of $X_{(3)}$ and $Y$ is a consequence of the relation

$$X_{(3)} = U(B \odot A)^T.$$

## 5. Numerical examples for CMTF

Here, we consider two numerical examples comparing our CMTF algorithms. In the first example, we are going to compare two approaches to CMTF algorithms without randomization, one using the ALS method and the other using SVD. Namely, we compare Algorithm 7 with Algorithm 6. The second example is analogous to Examples 1–5 from Section 3 and we compare randomized versions of Algorithm 6.

### 5.1. First example: Comparing the basic algorithms

Let $m = 100$, $n_2 = 50$, $n_3 = 20$, $n = 30$ and $r = 3$ and define:

- `U = randn(m,r); B = randn(n2,r); C=randn(n3,r); W=randn(n,r);`
- $\mathcal{X} = [[U, B, C]]$;
- `Y=U*W'.`

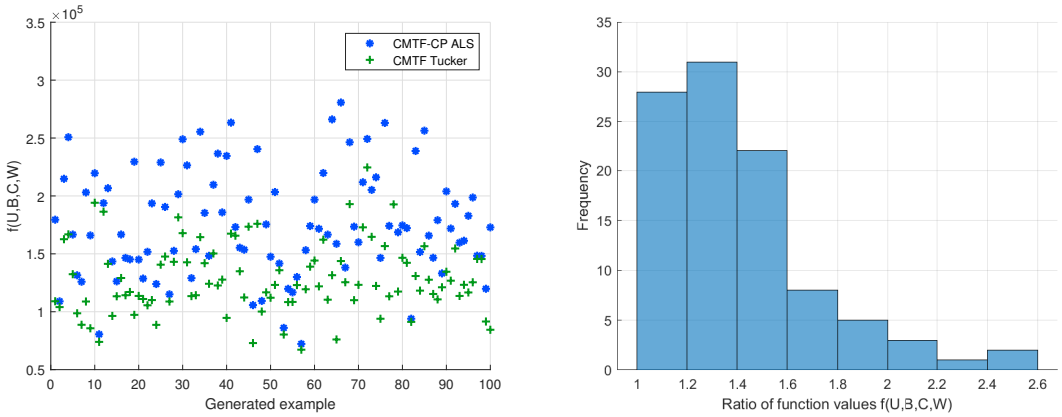


Figure 9: Comparison of Algorithms 6 and 7 for CMTF for 100 randomly generated examples.

We use this example to test the algorithms without randomization, i.e., we compare Algorithm 6 for a tensor in the Tucker format using CMF of $X_{(1)}$ and $Y$ and Algorithm 7 for a tensor in CP format using the ALS method. We generate 100 random pairs $(\mathcal{X}, Y)$, as described above. The approximation rank is $k = 3$. For the values obtained by the algorithms, we compute

$$\|\mathcal{X} - \mathcal{S} \times_1 U \times_1 V_2 \times_3 V_3\|^2 + \|Y - UW^T\|_F^2,$$

for $\mathcal{X}$ in the Tucker format, and

$$\|\mathcal{X} - [[U, B, C]]\|^2 + \|Y - UW^T\|_F^2,$$

for $\mathcal{X}$ in CP format. These expressions are the values of the corresponding objective functions.

The results are represented in Figure 9. It is clear that Algorithm 6, that is, the method which uses CMF of tensor matricization and matrix $Y$, performs equally well and almost always better than Algorithm 7, which is based on ALS. In most experiments, the objective function value $f(U, B, C, W)$ obtained with ALS is at least 20% higher, and in some cases, it is up to twice as high, compared to our approach. This is because convergence of the ALS method to the global minimum is not guaranteed, i.e., the method could converge to another stationary point. Therefore, in the next example, we do not use the ALS approach.

### 5.2. Second example: Synthetic tensor and matrix

This example is inspired by the matrix $X$ from `TestSynthetic2`. We define a diagonal matrix

$$S = \operatorname{diag}(1, \ldots, 1, d^{-2}, d^{-3}, \ldots, d^{-(n-r+1)}) \in \mathbb{R}^{n \times n}$$

and construct matrix $Y \in \mathbb{R}^{n \times n}$ and tensor $X \in \mathbb{R}^{n \times n \times n}$ in the following way:

- `UY=orth(rand(n)); VY=orth(rand(n));`
- `Y = UY*S*VY';`
- `X(:,:,1)=[UY(:,1:r1) orth(rand(n,n-r1)]*S*[VY(:,1:r1) orth(rand(n,n-r1)]';`
- `X(:,:,2)=[UY(:,1:r2) orth(rand(n,n-r2)]*S*[VY(:,1:r2) orth(rand(n,n-r2)]';`
- `X(:,:,3)=[UY(:,1:r3) orth(rand(n,n-r3)]*S*[VY(:,1:r3) orth(rand(n,n-r3)]'.`

For our experiment, we choose $n = 100$, $r = 5$, $d = 2$, $r_1 = 5$, $r_2 = 10$, and $r_3 = 7$. We are looking for the rank $k = 10$ approximation.

| Algorithm | $p$ | $\ell$ | $q$ | $err_{\mathcal{X}}$ | $err_Y$ | total time (s) | CMF time (s) |
|---|---|---|---|---|---|---|---|
| Basic CMTF | - | - | - | $2.51650855 \cdot 10^{-2}$ | $2.52792977 \cdot 10^{-2}$ | 0.444682 | 0.444682 |
| Randomized | 10 | - | - | $2.96659479 \cdot 10^{-2}$ | $2.84378062 \cdot 10^{-2}$ | 0.234335 | 0.022847 |
| RSI | 10 | - | 5 | $2.51760209 \cdot 10^{-2}$ | $2.52882608 \cdot 10^{-2}$ | 0.235490 | 0.022847 |
| RBKI | 26 | 1 | 18 | $2.51648934 \cdot 10^{-2}$ | $2.52806919 \cdot 10^{-2}$ | 0.247642 | 0.012539 |
| RBKI | 22 | 2 | 8 | $2.51646244 \cdot 10^{-2}$ | $2.52820672 \cdot 10^{-2}$ | 0.244876 | 0.022847 |

Table 5: Selected results for CMTF where the tensor is in the Tucker format and $k = 10$. Oversampling parameters, relative errors for RSI with $q = 5$ and RBKI with $\ell = 1, 2$, and running times.

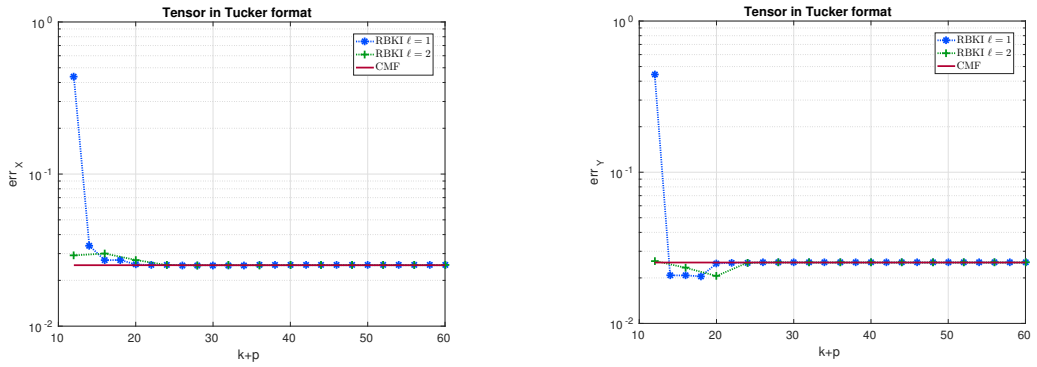


Figure 10: Relative errors with respect to the number of iterations in the RBKI algorithm for Matrix-Tensor decomposition in the Tucker format.

Following the results from the previous example, here we do the tests only for the Tucker format. We use the result obtained by the basic Algorithm 6 as a benchmark. In the randomized versions, we use Algorithm 3 for RSI and Algorithm 4 for RBKI, instead of Algorithm 1. The accuracy is determined by the relative errors

$$err_{\mathcal{X}} = \frac{\|\mathcal{X} - \mathcal{S} \times_1 U \times_2 V_2 \times_3 V_3\|}{\|\mathcal{X}\|} \tag{24}$$

and $err_Y$, as given in (12). The selected results, together with the running times, are presented in Table 5. We can see that randomized algorithms are always faster, both in total time and CMF time. Figure 10 shows relative errors for the RBKI method with $\ell = 1$ and $\ell = 2$, with various $q$.

## 6. Face recognition

So far, we have introduced CMF and CMTF algorithms with different randomization techniques. Since these algorithms extract a common part of two matrices (or a matrix and a tensor), we created new face recognition algorithms with CM(T)F as their main feature.

We performed the tests on the image collection from the widely used Georgia Tech database [19]. This database contains a total of 750 images of 50 different people. Each person is represented by 15 face images taken with different facial expressions. An example of a set of images for one person is given in Figure 11. We used the first ten as the training images and the last five as the test images. Developing a new optimal face recognition algorithm is out of scope of this paper. Therefore, to expedite testing, we selected 75 images from five different individuals. Those are presented in Figure 12. For now, we limit our tests to grayscale images. All images are uniformly cropped to the same size, with each image represented by a single matrix. Although the CMF algorithm requires only the same number of rows, the best test results were achieved when we also set the same number of columns.

Figure 11: An example of 15 images per person from the Georgia Tech database.

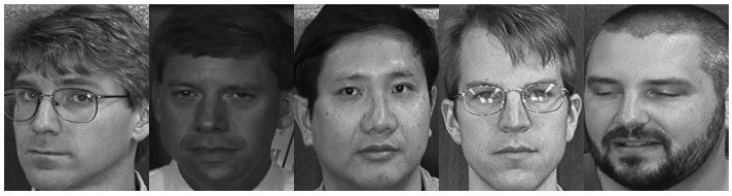

Figure 12: Images of five people used in the tests of our face recognition algorithms.

We constructed two versions of the face recognition algorithm. The first one employs a coupled factorization of two matrices, while the second one uses coupled matrix and tensor factorization. The approximation rank in CMF was set to $k = 5$. Randomization parameters are $q = 2$ for RSI and $\ell = 5$, $q = 2$ for RBKI. The algorithm steps are as follows.

(1) The first 10 images per person were put into our database, where it is known which person is in which image. We denoted the matrices representing those images by $X^{(1)}, X^{(2)}, \ldots, X^{(50)}$. Matrices $X^{(1)}$, $X^{(2)}, \ldots, X^{(10)}$ correspond to the first person, $X^{(11)}, X^{(12)}, \ldots, X^{(20)}$ to the second person, etc. We used the remaining 25 images (five per person) to test the algorithms. Matrix $Y$ represents the image of the person we want to recognize.

(2 CMF) We calculate

$$\mathrm{CMF}(X^{(1)}, Y),\ \mathrm{CMF}(X^{(2)}, Y),\ \ldots,\ \mathrm{CMF}(X^{(50)}, Y).$$

The corresponding errors are calculated as

$$err^{(i)} = err_X^{(i)} + err_Y^{(i)}, \quad \text{for all } i = 1, \ldots, 50,$$

where $err_X$ and $err_Y$ are the relative errors defined in relation (12).

(2 CMTF) We created five $m \times n \times 10$ tensors, one per person, each of them containing ten images of the same person. Then, we calculate

$$\mathrm{CMTF}(\mathcal{X}^{(1)}, Y),\ \mathrm{CMTF}(\mathcal{X}^{(2)}, Y),\ \ldots,\ \mathrm{CMTF}(\mathcal{X}^{(5)}, Y).$$

The relative errors are determined by

$$err^{(i)} = err_{\mathcal{X}}^{(i)} + err_Y^{(i)}, \quad \text{for all } i = 1, \ldots, 5,$$

where $err_{\mathcal{X}}$ is like in relation (24), for tensors in the Tucker format, or

$$err_{\mathcal{X}} = \frac{\|\mathcal{X} - [[U, B, C]]\|}{\|\mathcal{X}\|},$$

in CP format, and $err_Y$ is as in relation (12).

(3) The index $i$ for which the value $err^{(i)}$ is minimal determines which person is in the image.

We compared 12 different versions of the algorithm, depending on whether CMF or CMTF is used and also on the randomization type (the basic algorithm without randomization, simple randomization, RSI, or RBKI). For each tested case, we calculated the success rate, that is, the number of correctly recognized images divided by the total number of tested images. The results are presented in Table 6.

| Algorithm | Person 1 (%) | Person 2 (%) | Person 3 (%) | Person 4 (%) | Person 5 (%) | Total (%) |
|---|---|---|---|---|---|---|
| CMF | 80 | 100 | 20 | 100 | 100 | 80 |
| Randomized CMF | 80 | 100 | 40 | 80 | 80 | 76 |
| RSI CMF | 100 | 100 | 20 | 80 | 100 | 80 |
| RBKI CMF | 80 | 100 | 20 | 100 | 100 | 80 |
| CMTF-Tucker | 60 | 100 | 100 | 100 | 80 | 88 |
| Randomized CMTF-Tucker | 40 | 80 | 20 | 80 | 20 | 48 |
| RSI CMTF-Tucker | 60 | 100 | 80 | 100 | 80 | 84 |
| RBKI CMTF-Tucker | 60 | 100 | 100 | 100 | 80 | 88 |
| CMTF-CP ALS | 80 | 100 | 0 | 40 | 20 | 48 |
| Randomized CMTF-CP ALS | 40 | 100 | 0 | 100 | 0 | 48 |
| RSI CMTF-CP ALS | 0 | 100 | 0 | 100 | 60 | 52 |
| RBKI CMTF-CP ALS | 20 | 100 | 40 | 100 | 60 | 64 |

Table 6: The success rate for different versions of the face recognition algorithm.

We can see that CMTF-Tucker shows the best performance. The success rate goes up to 88%. As expected, for the randomized algorithms, the best results are achieved when RBKI is used. For CMF and CMTF-Tucker, these are the same as for the basic Algorithms 1 and 6. With CMTF-Tucker, the success rate is more evenly distributed among individuals than with CMF. The worst results were obtained with CMTF-CP ALS. Here, we can also observe that, unexpectedly, randomization improved the basic algorithm. We attribute this to the convergence problems mentioned earlier.

## 7. Conclusion

In this paper, we analyze coupled factorization of two matrices (CMF), as well as coupled factorization of a matrix and a tensor (CMTF), and the possibilities for their randomization. We showed that the problem of a rank-$k$ coupled matrix factorization for matrices $X \in \mathbb{R}^{m\times n_1}$ and $Y \in \mathbb{R}^{m\times n_2}$ is equivalent to the problem of rank-$k$ approximation of the matrix $\begin{bmatrix} X & Y \end{bmatrix} \in \mathbb{R}^{m\times(n_1+n_2)}$. Moreover, we showed that a related problem of a rank-$k$ coupled factorization of tensor $\mathcal{X} \in \mathbb{R}^{m\times n_2\times n_3}$ and matrix $Y \in \mathbb{R}^{m\times n}$ is equivalent to the problem of rank-$k$ approximation of the matrix $\begin{bmatrix} X_{(1)} & Y \end{bmatrix} \in \mathbb{R}^{m\times(n_2n_3+n)}$, where $X_{(1)} \in \mathbb{R}^{m\times(n_2n_3)}$ is mode-1 matricization of $\mathcal{X}$. The main part of the paper is concerned with randomization techniques that can be used with CMF and CMTF algorithms. First, we argue for the importance of choosing a good projection matrix for the dimension reduction of the original problem. We propose a method where we use the QR decomposition of both terms in factorization, as opposed to just computing the QR decomposition of an augmented matrix. This produces more accurate results. Next, using our strategy for finding the projection matrix, and motivated by three randomization techniques, i.e., randomization based on the randomized SVD algorithm, randomized subspace iteration (RSI), and randomized block Krylov iteration (RBKI), we created and tested different versions of randomized CM(T)F algorithms. Finally, we applied our algorithms to the face recognition problem, resulting in new competitive algorithms for this type of problem.

Matlab codes used to generate numerical examples are available at:
https://github.com/anitacarevic/Randomized-coupled-decompositions.

## Acknowledgements

This work has been supported by the Croatian Science Foundation under the project UIP-2019-04-5200. The authors are grateful to Daniel Kressner for his useful comments on the early version of this work. The authors also thank the anonymous referees for their helpful suggestions that improved the paper.

## References

[1] E. Acar, R. Bro, and A. K. Smilde, Data fusion in metabolomics using coupled matrix and tensor factorizations, *Proceedings of the IEEE* **103** (2015), no. 9, 1602–1620.

[2] E. Acar, D. M. Dunlavy, and T. G. Kolda, A scalable optimization approach for fitting canonical tensor decompositions, *Journal of Chemometrics* **25** (2011), no. 2, 67–86.

[3] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, Scalable tensor factorizations for incomplete data, *Chemometrics and Intelligent Laboratory Systems* **106** (2011), no. 1, 41–56, Multiway and Multiset Data Analysis.

[4] M. Bagherian, R. B. Kim, C. Jiang, M. A. Sartor, H. Derksen, and K. Najarian, Coupled matrix–matrix and coupled tensor–matrix completion methods for predicting drug–target interactions, *Briefings in Bioinformatics* **22** (2020), no. 2, 2161–2171.

[5] R. A. Borsoi, I. Lehmann, M. A. B. S. Akhonda, V. D. Calhoun, K. Usevich, D. Brie, and T. Adali, Coupled CP tensor decomposition with shared and distinct components for multi-task fMRI data fusion, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[6] P. Drineas and M. W. Mahoney, RandNLA: randomized numerical linear algebra, *Commun. ACM* **59** (2016), no. 6, 80–90.

[7] A. El Hachimi, K. Jbilou, A. Ratnani, and L. Reichel, A tensor bidiagonalization method for higher-order singular value decomposition with applications, *Numerical Linear Algebra with Applications* **31** (2024), no. 2, e2530.

[8] M. Hached, K. Jbilou, C. Koukouvinos, and M. Mitrouli, A multidimensional principal component analysis via the C-product Golub–Kahan–SVD for classification and face recognition, *Mathematics* **9** (2021), no. 11, 1249.

[9] N. Halko, P. G. Martinsson, and J. A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Rev.* **53** (2011), no. 2, 217–288.

[10] Z.-G. Jia, S.-T. Ling, and M.-X. Zhao, Color two-dimensional principal component analysis for face recognition based on quaternion model, in *Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part I 13*, Springer, 2017, pp. 177–189.

[11] H. A. L. Kiers, Towards a standardized notation and terminology in multiway analysis, *Journal of Chemometrics* **14** (2000), no. 3, 105–122.

[12] T. G. Kolda and B. W. Bader, Tensor decompositions and applications, *SIAM Rev.* **51** (2009), no. 3, 455–500.

[13] L. Li, S. Yan, D. Horner, M. Rasmussen, A. Smilde, and E. Acar Ataman, Revealing static and dynamic biomarkers from postprandial metabolomics data through coupled matrix and tensor factorizations, *Metabolomics* **20** (2024), no. 86.

[14] P.-G. Martinsson, Randomized methods for matrix computations, *IAS/Park City Mathematics Series* (2016).

[15] P.-G. Martinsson and J. A. Tropp, Randomized numerical linear algebra: Foundations and algorithms, *Acta Numerica* **29** (2020), 403–572.

[16] R. Meyer, C. Musco, and C. Musco, On the unreasonable effectiveness of single vector Krylov methods for low-rank approximation, in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SIAM, 2024, pp. 811–845.

[17] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra, Randomized numerical linear algebra: A perspective on the field with an eye to software, *arXiv:2302.11474 [math.NA]* (2023).

[18] Y. Nakatsukasa, Fast and stable randomized low-rank matrix approximation, *arXiv:2009.11392 [math.NA]* (2020).

[19] A. V. NEFIAN, Georgia Tech face database, http://www.anefian.com/research/face_reco.htm.

[20] A. K. SAIBABA and A. MIĘDLAR, Randomized low-rank approximations beyond Gaussian random matrices, *SIAM Journal on Mathematics of Data Science* **7** (2025), no. 1, 136–162.

[21] C. SCHENKER, J. E. COHEN, and E. ACAR, A flexible optimization framework for regularized matrix-tensor factorizations with linear couplings, *IEEE Journal of Selected Topics in Signal Processing* **15** (2021), no. 3, 506–521.

[22] C. SCHENKER, J. E. COHEN, and E. ACAR, An optimization framework for regularized linearly coupled matrix-tensor factorization, in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 985–989.

[23] A. P. SINGH and G. J. GORDON, Relational learning via collective matrix factorization, in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *KDD '08*, Association for Computing Machinery, New York, NY, USA, 2008, p. 650–658.

[24] A. K. SMILDE, J. A. WESTERHUIS, and R. BOQUÉ, Multiway multiblock component and covariates regression models, *Journal of Chemometrics* **14** (2000), no. 3, 301–331.

[25] M. SØRENSEN, I. DOMANOV, and L. DE LATHAUWER, Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-$(L_{r,n}, L_{r,n}, 1)$ terms—Part II: Algorithms, *SIAM J. Matrix Anal. Appl.* **36** (2015), no. 3, 1015–1045.

[26] M. D. SOROCHAN ARMSTRONG, J. L. HINRICH, A. P. DE LA MATA, and J. J. HARYNUK, PARAFAC2×N: Coupled decomposition of multi-modal data with drift in N modes, *Analytica Chimica Acta* **1249** (2023), 340909.

[27] M. SØRENSEN and L. DE LATHAUWER, Coupled tensor decompositions for applications in array signal processing, in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2013, pp. 228–231.

[28] J. A. TROPP and R. J. WEBBER, Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications, *arXiv:2306.12418 [math.NA]* (2023).

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

[30] F. VAN EEGHEM and L. DE LATHAUWER, Tensor similarity in chemometrics, in *Comprehensive Chemometrics (Second Edition)* (S. BROWN, R. TAULER, and B. WALCZAK, eds.), Elsevier, Oxford, 2020, pp. 337–354.

[31] I. VAN MECHELEN and A. K. SMILDE, A generic linked-mode decomposition model for data fusion, *Chemometrics and Intelligent Laboratory Systems* **104** (2010), no. 1, 83–94, OMICS.

[32] M. A. O. VASILESCU and D. TERZOPOULOS, Multilinear analysis of image ensembles: Tensorfaces, in *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*, Springer, 2002, pp. 447–460.

[33] Y. WEI, P. XIE, and L. ZHANG, Tikhonov regularization and randomized GSVD, *SIAM Journal on Matrix Analysis and Applications* **37** (2016), no. 2, 649–675.

[34] D. P. WOODRUFF, Sketching as a tool for numerical linear algebra, *Found. Trends Theor. Comput. Sci.* **10** (2014), no. 1–2, 1–157.

[35] X. XIAO and Y. ZHOU, Two-dimensional quaternion PCA and sparse PCA, *IEEE Transactions on Neural Networks and Learning Systems* **30** (2018), no. 7, 2028–2042.

[36] S. YAN, L. LI, D. HORNER, P. EBRAHIMI, B. CHAWES, L. DRAGSTED, M. RASMUSSEN, A. SMILDE, and E. ACAR ATAMAN, Characterizing human postprandial metabolic response using multiway data analysis, *Metabolomics* **20** (2024), no. 50.

[37] J. YANG, D. ZHANG, A. F. FRANGI, and J.-Y. YANG, Two-dimensional PCA: A new approach to appearance-based face representation and recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** (2004), no. 1, 131–137.