Fast low rank approximations of matrices and tensors

S. Friedland [*] V. Mehrmann[†] A. Miedlar[‡] M. Nkengla[*]

7.2.2008

Abstract

In many applications such as data compression, imaging or genomic data analysis, it is important to approximate a given $m \times n$ matrix A by a matrix B of rank at most k which is much smaller than m and n. The best rank k approximation can be determined via the singular value decomposition which, however, has prohibitively high computational complexity and storage requirements for very large m and n.

We present an optimal least squares algorithm for computing a rank k approximation to an $m \times n$ matrix A by reading only a limited number of rows and columns of A. The algorithm has complexity $\mathcal{O}(k^2 \max(m, n))$ and allows to iteratively improve given rank k approximations by reading additional rows and columns of A. We also show how this approach can be extended to tensors and present numerical results.

2000 Mathematics Subject Classification. 15A18, 65F15, 62-07, 15A69
Key words. singular value decomposition, CUR decomposition, rank k approximation, least squares, Tucker decomposition

1 Introduction

Let $A = [a_{i,j}] \in \mathbb{R}^{m\times n}$, i.e. A is a real valued $m \times n$ matrix, where m, n are very large, e.g. $m, n \approx 10^6$. We may think of A as a full matrix which describes a noisy image [15], micro-array [1], genetic marker or HAP SNP data [22]. In order to compress or denoise the matrix, often a low rank (say rank k) approximation $B = [b_{i,j}] \in \mathbb{R}^{m\times n}$ of A is determined and used instead of the original data. Such a decomposition can be written as

$$B = \mathbf{x}_1\mathbf{y}_1^\top + \ldots + \mathbf{x}_k\mathbf{y}_k^\top, \quad \mathbf{x}_i \in \mathbb{R}^m,\ \mathbf{y}_i \in \mathbb{R}^n,\ i = 1, \ldots, k. \tag{1.1}$$

It requires only an amount of storage given by $(m+n)k$ instead of mn floating point numbers for the full A. The best rank k approximation of A with respect to the Frobenius norm can be determined via the singular value decomposition (SVD) $A = W\Sigma V^T$, with orthogonal matrices $W \in \mathbb{R}^{m\times m}$ $V \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{R}^{m\times n}$

[*]Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, Illinois 60607-7045, USA, `friedlan@uic.edu`, `nkengla@msn.com`. Partially supported by *Berlin Mathematical School*, Berlin, Germany.

[†]Institut für Mathematik, TU Berlin, Str. des 17. Juni 136, D-10623 Berlin, FRG. `mehrmann@math.tu-berlin.de`. Partially supported by *Deutsche Forschungsgemeinschaft*, through the DFG Research Center MATHEON *Mathematics for Key Technologies* in Berlin

[‡]Institut für Mathematik, TU Berlin, Str. des 17. Juni 136, D-10623 Berlin, FRG. `miedlar@math.tu-berlin.de`. Supported by *Berlin Mathematical School*, Berlin, Germany.

with $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0$. Here r is the rank of A and the best rank k approximation is given by $B = W\tilde{\Sigma}V^T$, where $\tilde{\Sigma}$ is obtained from Σ by replacing $\sigma_{k+1}, \ldots, \sigma_r$ with zeroes [10].

The standard procedures for computing the SVD have a computational complexity,of $\mathcal{O}(\min(m,n)^2 \max(m,n))$ floating point operations (flops), see e.g. [2, 10]. This makes the SVD unsuitable for most large scale applications. If one is only interested in a low rank approximation then one can use Arnoldi or Lanczos methods [10, 16] to compute the singular values and singular vectors associated with the k largest singular values. This requires a substantial number of matrix-vector multiplications with the matrix A and thus a complexity of at least $\mathcal{O}(mnk)$ which still may be too large. Due to its high demand in applications, the topic of deriving faster methods to compute rank k approximation has therefore recently received a lot of attention.

Let us briefly recall some of the recent literature. In the context of data-sparse approximations for the numerical solution of large linear systems, *pseudo-skeleton approximation* algorithms were suggested [11, 12, 13]. These procedures use block Gaussian elimination with a suitable $k \times k$ block matrix to determine a rank k approximation via a so-called CUR decomposition. They have complexity $\mathcal{O}(k^2(m+n))$ once an appropriate block is chosen.

Other methods with complexity $\mathcal{O}(mnk)$ that are also based on *CUR-decompositions* and subspace sampling were suggested in [23, 24] and later improved and analyzed in [3, 4]. An improvement of the sampling idea was suggested in [7], which uses an iterative algorithm that reads several rows of A at random, determines from these data a rank k approximation and then iteratively improves the approximation by reading additional rows. Each update gives a better rank k approximation to A and the algorithm is terminated if the approximation is not *improved* any further, or if an allocated number of computational work is exceeded. The complexity of this algorithm, however, is still $\mathcal{O}(mnk)$. The first approaches for sampling methods that go below this complexity were presented in [14, 24].

In this paper we describe a method that combines several of these ideas to an iterative method that allows on the one hand adaptive improvement of current rank k approximations and on the other hand has small computational complexity.

Let us briefly introduce the main idea of our algorithm for matrices. Assume that we read at random p columns and q rows of A, indexed by the sets $J \subset \{1, \ldots, n\}, I \subset \{1, \ldots, m\}$. This information corresponds to $m \times p$ and $q \times n$ matrices C and R respectively. Let $A_{I,J}$ be the submatrix of A formed by the rows in I and columns in J. We look for a matrix $F = CUR \in \mathbb{R}^{m \times n}$, with $U \in \mathbb{R}^{p \times q}$ still to be determined. We determine U_{opt} as a solution to the least square problem of minimizing $\sum_{(i,j)\in\mathcal{S}}(a_{i,j} - (CUR)_{i,j})^2$, i.e.,

$$U_{\text{opt}} = \arg \min_{U \in \mathbb{R}^{p \times q}} \sum_{(i,j)\in\mathcal{S}} (a_{i,j} - (CUR)_{i,j})^2. \tag{1.2}$$

Here $\mathcal{S} \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$ combines the entries of A which are contained in C and R. We show that a U_{opt} with minimal Frobenius norm is given by $A_{I,J}^{\dagger}$, the *Moore-Penrose* inverse of $A_{I,J}$. See [10] for the definition and the properties of the Moore-Penrose inverse. Assume that $p = q$ and the matrix $A_{I,J}$ is invertible. Then $U_{\text{opt}} = A_{I,J}^{-1}$ is the unique solution to (1.2), and the rows and columns of A indexed by I, J, respectively, coincide with $CA_{I,J}^{-1}R$. If $A_{I,J}$ is not a square matrix, or if $A_{I,J}$ is ill-conditioned, then we modify the CUR approximation to be of the form $C\tilde{U}_{\text{opt}}R$, where $\tilde{U}_{\text{opt}}$ is the Moore-Penrose inverse of a best rank ℓ approximation of $A_{I,J}$. To ensure that $A_{I,J}$ was not poorly chosen, we will pick several choices of $A_{I,J}$ and we will use the one which has the maximal number of significant singular values and among those the maximal product of significant singular values. This concept is an alternative to the approach suggested in [11] to use a square submatrix $A_{I,J}$ of maximal absolute determinant.

The paper is organized as follows. We present the mathematical formulation of our method in Section 2 and in Section 3 we sketch how to extend our ideas to tensors. In Section 4 we present some computational results obtained with our algorithm on real and synthetic data and we compare our results with the best rank k approximation and with a full *least squares approximation*.

We suggest that our new approach is very useful for real life applications with noisy data, and that it will have many applications.

2 Mathematical formulation

In this section we describe the mathematical basis of our method.

Let $A = [a_{i,j}]_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ and let $||A||_F := (\sum_{i,j=1}^{m,n} a_{i,j}^2)^{1/2}$ denote the Frobenius norm of A. We use the notation $\langle m\rangle := \{1,\ldots,m\}, \langle n\rangle := \{1,\ldots,n\}$ and let

$$I = \{1 \le \alpha_1 < \ldots < \alpha_q \le m\} \subset \langle m\rangle, \quad J = \{1 < \beta_1 < \ldots < \beta_p \le n\} \subset \langle n\rangle$$

be two nonempty sets of cardinality q, p respectively. Using the indices in I, J, we consider the submatrices

$$\begin{aligned} A_{I,J} &= [a_{\alpha_k,\beta_l}]_{k,l=1}^{q,p} \in \mathbb{R}^{q\times p}, \\ R = A_{I,\langle n\rangle} &= [a_{\alpha_k,j}]_{k,j=1}^{q,n} \in \mathbb{R}^{q\times n}, \\ C = A_{\langle m\rangle,J} &= [a_{i,\beta_l}]_{i,l=1}^{m,p} \in \mathbb{R}^{m\times p}, \end{aligned} \tag{2.1}$$

with index set $\mathcal{S} := \langle m\rangle \times \langle n\rangle \backslash ((\langle m\rangle\backslash I) \times (\langle n\rangle\backslash J))$, of cardinality $\#\mathcal{S} = mp + qn - pq$. Thus, $C = A_{\langle m\rangle,J}$ and $R = A_{I,\langle n\rangle}$ are composed of the columns in J and the rows I of A, respectively.

A commonly used approximation [1, 4, 5, 6] of A based on C, R is of the form CUR, for some $U \in \mathbb{R}^{p\times q}$. If we know all the entries of A then the optimal choice of U is given by U_b satisfying

$$||A - CU_bR||_F = \min_{U\in\mathbb{R}^{p\times q}} ||A - CUR||_F. \tag{2.2}$$

It is well known that $U_b = C^\dagger AR^\dagger$.

Let $\mathcal{C}_r(p,q) \subset \mathbb{R}^{p\times q}$ denote the set of all real $p\times q$ matrices of rank at most r. Then, more generally, for $r \le \min(p,q)$ the optimal choice for U is given by $U_{b,r}$ satisfying

$$||A - CU_{b,r}R||_F = \min_{U\in\mathcal{C}_r(p,q)} ||A - CUR||_F. \tag{2.3}$$

An explicit formula for $U_{b,r}$ is given in [8]. For $r = \min(\text{rank } C, \text{rank } R) \le \min(p,q)$, one can choose $U_b = U_{b,r}$. The computational complexity to compute U_b is $\mathcal{O}(mnp^2q^2)$ (or roughly $\mathcal{O}(mn)$ if $p << m$ and $q << n$), which is again prohibitively large if m and n are very large. Note that the iterative algorithm suggested in [7] to compute a rank r approximation of A is of order $\mathcal{O}(mnr)$.

We have seen that to compute the best approximation to A of the form $F = CUR$ using only the entries of A given by C and R, we have to determine U_{opt} by solving the minimization problem (1.2). The optimal solution can be determined by the *least squares* solution, via a reformulation as linear system

$$T\mathbf{u} = \mathbf{a}, \quad T \in \mathbb{R}^{(mp+qn-pq)\times pq}, \mathbf{u} \in \mathbb{R}^{pq}, \mathbf{a} \in \mathbb{R}^{mp+qn-pq}, \tag{2.4}$$

where $\mathbf{u}, \mathbf{a}$ are vectors, whose coordinates are the entries of U and those entries of A, which are either in C or R, respectively. Moreover T is a submatrix of the Kronecker product $A \otimes A^\top$. This least squares approach is nice for the analysis of the problem but it cannot be used for efficient computation in this way.

Instead we can actually give in the following two theorems an explicit solution that can even be computed fast. If $A_{I,J}$ is a square and invertible, then the overdetermined system (2.4) has a unique solution but even if $A_{I,J}$ is not square we can make the solution unique by requiring norm minimization in Frobenius norm.

Theorem 2.1 *Let $A \in \mathbb{R}^{m\times n}$, and let $I \subset \langle m\rangle$, $J \subset \langle n\rangle$ have cardinality q and p, respectively. Let $C = A_{\langle m\rangle,J} \in \mathbb{R}^{m\times p}$, and $R = A_{I,\langle n\rangle} \in \mathbb{R}^{p\times n}$ be as in (2.1) and suppose that $A_{I,J}$ is invertible. Then the overdetermined system (2.4) has a unique solution $U = A_{I,J}^{-1}$, i.e., the rows in I and the columns in J of the matrix $CA_{I,J}^{-1}R$ are equal to the corresponding rows and columns of A, respectively.*

Proof. For any $I \subset \langle m\rangle$, $J \subset \langle n\rangle$, with $\#I = q$, $\#J = p$, and $U \in \mathbb{R}^{m\times n}$ we have the identity

$$(A_{\langle m\rangle,J}UA_{I,\langle n\rangle})_{I,J} = A_{I,J}UA_{I,J}. \tag{2.5}$$

Hence the part of the system (2.4) corresponding to $(CUR)_{I,J} = A_{I,J}$ reduces to the equation

$$A_{I,J}UA_{I,J} = A_{I,J} \tag{2.6}$$

If $A_{I,J}$ is a square matrix and invertible, then the unique solution to this matrix equation is $U = A_{I,J}^{-1}$. Furthermore

$$(A_{\langle m\rangle,J}A_{I,J}^{-1}A_{I,\langle n\rangle})_{I,\langle n\rangle} = A_{I,J}A_{I,J}^{-1}A_{I,\langle n\rangle} = A_{I,\langle n\rangle},$$
$$(A_{\langle m\rangle,J}A_{I,J}^{-1}A_{I,\langle n\rangle})_{\langle m\rangle,J} = A_{\langle m\rangle,J}A_{I,J}^{-1}A_{I,J} = A_{\langle m\rangle,J}.$$

□

This results extends to the general nonsquare case.

Theorem 2.2 *Let $A \in \mathbb{R}^{m\times n}$, and let $I \subset \langle m\rangle$, $J \subset \langle n\rangle$ have cardinality q and p, respectively. Let $C = A_{\langle m\rangle,J} \in \mathbb{R}^{m\times p}$, and $R = A_{I,\langle n\rangle} \in \mathbb{R}^{p\times n}$ be as in (2.1). Then $U = A_{I,J}^\dagger$ is the minimal solution (in Frobenius norm) of (1.2).*

Proof. Using the SVD of $A_{I,J}$ we may assume w.l.o.g. that $A_{I,J} = \Sigma_r \oplus 0_{(q-r)\times(p-r)}$, where $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ and $r = \mathrm{rank}\, A_{I,J}$. For $U \in \mathbb{R}^{p\times q}$ denote by $U_r \in \mathbb{R}^{p\times q}$ the matrix obtained from U by replacing the last $p-r$ rows and $q-r$ columns by rows and columns of zeroes, respectively. Note that then $CUR = CU_rR$ and $\|U_r\|_F \le \|U\|_F$, and equality holds if and only if $U = U_r$. Hence, the minimal Frobenius norm least squares solution U of is given by $U = U_r$. Using the fact that the rows $r+1, \ldots, q$ and columns $r+1, \ldots, p$ of CUR are zero it follows that the minimum in (1.2) is reduced to the minimum on $\mathcal{S}' = \langle m\rangle \times \langle r\rangle \cup \langle r\rangle \times \langle n\rangle$.

Then, by Theorem 2.1 the solution to the minimal Frobenius norm least square problem is given by $\Sigma^\dagger$. $\square$

To make use of these ideas in finite precision arithmetic we have to work with the *numerical rank* rather than with the rank. For $A \in \mathbb{R}^{m\times n}$ the numerical rank $\mathrm{rank}_{\mathrm{num}}\, A$ of A is defined to be the number of singular values σ_j of A that are above a certain threshold, which is usually taken to be $\epsilon \cdot c(n,m)\cdot \sigma_1$, where ϵ is the machine precision, $c(n,m)$ is a low degree polynomial in the dimensions n, m and σ_1 is the largest singular value of A, see [2, 10] for details.

Assume that we have selected at random sets of q rows I and p columns J, and set $r_{\mathrm{num}} := \mathrm{rank}_{\mathrm{num}}\, A_{I,J}$. If the rank of $A_{I,J}$ equals its numerical rank, then we can use Theorem 2.2 to choose $U_{\mathrm{opt}} = A^\dagger_{I,J}$. If rank $_{\mathrm{num}} <$ rank $A_{I,J}$, then we choose as approximation

$$B = A_{\langle m\rangle, J} A^\dagger_{I,J,r_{\mathrm{num}}} A_{J,\langle n\rangle}, \quad \tilde{U}_{\mathrm{opt}} = A^\dagger_{I,J,r_{\mathrm{num}}}, \tag{2.7}$$

where $A_{I,J,r_{\mathrm{num}}}$ is the best rank r_{num} approximation of $A_{I,J}$.

We summarize the presented procedure in the following algorithm.

Algorithm 2.3 FSVD *Let $A \in \mathbb{R}^{m\times n}$. Fix integers $p, t_{\max} \geq 1$ and set $q = p$.*

1. *Choose two sets $I \subset \langle m\rangle, J \subset \langle n\rangle$ of cardinality p at random. Determine the numerical rank $k(I,J) = r_{\mathrm{num}} A_(I,J)$ and the product of the first $k(I,J)$ singular values of $A_{I,J}$, which is denoted by $\pi(I,J)$. Repeat this procedure $t_{\max}$ times and keep the rows and columns I, J with the maximal $k := k(I,J)$ and of those, keep that with maximal $\pi(I,J)$.*

2. *Compute the best rank k approximation of $A_{I,J}$ denoted by $A_{I,J,k}$.*

3. *Use $B = CA^\dagger_{I,J,k}R$ as rank k approximation of A.*

Based on the information that we read, the S-average error (SAE) of our approximation is

$$\mathrm{Error}_{\mathrm{av}}(B) = \frac{\sum_{(i,j)\in\mathcal{S}}(a_{i,j} - b_{i,j})^2}{\sum_{(i,j)\in\mathcal{S}} a^2_{i,j}} \tag{2.8}$$

and typically this error is small. But if we are not satisfied with the average approximation error, then we can adaptively improve the approximation by adding more rows and columns to the matrices C, R, i.e. by increasing the index sets I, J and by computing a new $A^\dagger_{I,J,k}$ for the extended sets.

Since B and B_1 are of low rank, we can efficiently, (in complexity $\mathcal{O}(k^2(n+m))$), compute the average distance

$$\frac{||B - B_1||^2_F}{||B||_F ||B_1||_F}. \tag{2.9}$$

This is done as follows. Write explicitly each matrix X in the above expression as a sum of the corresponding rank one matrices, and recall that $\|X\|^2_F = \mathrm{tr}\, XX^\top$, where tr denotes the trace. Finally note that $\mathrm{tr}\, \mathbf{x}\mathbf{y}^\top \mathbf{u}\mathbf{v}^\top = (\mathbf{y}^\top\mathbf{u})(\mathbf{v}^\top\mathbf{u})$.

We adaptively increase the sets I, J until either the distance (2.9) has converged, or we have exceeded the allowed amount of computational work. In this iterative improvement we employ updating the computation of $A^\dagger_{I,J,k}$. Since updating of the singular value decomposition is usually not possible, we can reduce computation work (in particular for larger k) by replacing the singular value decomposition with the ULV decomposition [26] which can be updated when rows and columns are added.

Since the complexity of computing $A^\dagger_{I,J,k}$ is $\mathcal{O}(\min(p,q)^2 \max(p,q))$, it follows that for small $p,q << m,n$ the computational complexity of this method is dominated by the reading of the data and the computation of the average error (2.8) and thus we obtain a complexity of $\mathcal{O}(k^2 \max(m,n))$.

After presenting the basic concept of our method for matrices, in the next section we show how these ideas can be extended to tensors.

3 Extensions to tensors

In this section we discuss how our idea of computing least squares optimal CUR decompositions can be extended to 3 and 4 tensors, see also [17].

3.1 3-tensors

Let $\mathcal{A} = [a_{i_1,i_2,i_3}] \in \mathbb{R}^{l_1 \times l_2 \times l_3}$ be a 3-tensor, where the dimensions l_1, l_2, l_3, are large. For each $j = 1,2,3$ we read subtensors of $\mathcal{A}$ denoted by $\mathcal{C}_j = [c^{(j)}_{i_{1,j} i_{2,j} i_{3,j}}] \in \mathbb{R}^{l_{1,j} \times l_{2,j} \times l_{3,j}}$. We assume that $\mathcal{C}_j$ has the same number of coordinates as $\mathcal{A}$ in j-th direction, and a small number of coordinates in the other two directions. That is, $l_{j,j} = l_j$ and the other two indices $l_{s,j},\ s \in \{1,2,3\}\backslash\{j\}$ are of order $\mathcal{O}(k)$, for $j = 1,2,3$. So $\mathcal{C}_j$ corresponds to the *j-section* of the tensor $\mathcal{A}$. The *small* dimensions of $\mathcal{C}_j$ are $(l_{s_j,j}, l_{t_j,j})$ where $\{s_j, t_j\} = \{1,2,3\}\backslash\{j\}$ for $j = 1,2,3$. Let $m_j := l_{s_j,j} l_{t_j,j}$ for $j = 1,2,3$.

To determine an approximation, we then look for a 6-tensor

$$\mathcal{V} = [v_{q_1,q_2,q_3,q_4,q_5,q_6}] \in \mathbb{R}^{l_{2,1} \times l_{3,1} \times l_{1,2} \times l_{3,2} \times l_{1,3} \times l_{2,3}}$$

and approximate the given tensor $\mathcal{A}$ by a tensor

$$\mathcal{B} = [b_{i_1,i_2,i_3}] := \mathcal{V} \cdot \mathcal{C}_1 \cdot \mathcal{C}_2 \cdot \mathcal{C}_3 \in \mathbb{R}^{\ell_1 \times \ell_2 \times \ell_3},$$

where we contract the 6 indices in $\mathcal{V}$ and the corresponding two indices $\{1,2,3\}\backslash\{j\}$ in $\mathcal{C}_j$ for $j = 1,2,3$, i.e., our approximation has the entries

$$b_{i_1,i_2,i_3} = \sum_{q_1=1}^{\ell_{2,1}} \sum_{q_2=1}^{\ell_{3,1}} \sum_{q_3=1}^{\ell_{1,2}} \sum_{q_4=1}^{\ell_{3,2}} \sum_{q_5=1}^{\ell_{1,3}} \sum_{q_6=1}^{\ell_{2,3}} v_{q_1,q_2,q_3,q_4,q_5,q_6} c^{(1)}_{i_1,q_1,q_2} c^{(2)}_{q_3,i_2,q_4} c^{(3)}_{q_5,q_6,i_3}. \tag{3.1}$$

This approximation is equivalent to a so-called *Tucker approximation* [27]. Indeed, if we represent each tensor $\mathcal{C}_j$ by a matrix $C_j \in \mathbb{R}^{m_j \times l_j}$ that has the same number of columns as the range of the j-th index of the tensor $\mathcal{A}$ and as number of rows the product of the ranges of the remaining two *small* indices of $\mathcal{C}_j$, i.e. $C_j = [c^{(j)}_{r,i_j}]^{m_j \cdot \ell_j}_{r,i_j=1}$. Then $c^{(j)}_{r,i_j}$ is equal to the corresponding entry $c^{(j)}_{i_1,i_2,i_3}$, where the value of r corresponds to the double index (i_s, i_t) for $\{s,t\} = \{1,2,3\}\backslash\{j\}$.

Now with $\mathcal{U} = [u_{j_1,j_2,j_3}] \in \mathbb{R}^{m_1 \times m_2 \times m_3}$, the equivalent Tucker representation of $\mathcal{B} = [b_{i_1,i_2,i_3}]$ is given by the entries

$$b_{i_1,i_2,i_3} = \sum_{j_1=1}^{m_1} \sum_{j_2=1}^{m_2} \sum_{j_3=1}^{m_3} u_{j_1,j_2,j_3} c^{(1)}_{j_1,i_1} c^{(2)}_{j_2,i_2} c^{(3)}_{j_3,i_3}, \quad (i_1,i_2,i_3) \in \langle \ell_1 \rangle \times \langle \ell_2 \rangle \times \langle \ell_3 \rangle. \tag{3.2}$$

This formula is expressed commonly as

$$\mathcal{B} = \mathcal{U} \times_1 C_1 \times_2 C_2 \times_3 C_3. \tag{3.3}$$

We now choose three subsets of the rows, columns and heights of $\mathcal{A}$

$$I \subset \langle \ell_1 \rangle,\ \#I = p, \quad J \subset \langle \ell_2 \rangle,\ \#J = q, \quad K \subset \langle \ell_3 \rangle,\ \#K = r.$$

Let

$$\begin{aligned}
&\mathcal{C}_1 = \mathcal{A}_{\langle \ell_1 \rangle, J, K} := [a_{i,j,k}],\ i \in \langle \ell_1 \rangle, j \in J, k \in K,\\
&\mathcal{C}_2 = \mathcal{A}_{I, \langle \ell_2 \rangle, K} := [a_{i,j,k}],\ i \in I, j \in \langle \ell_2 \rangle, k \in K,\\
&\mathcal{C}_3 = \mathcal{A}_{I, J, \langle \ell_3 \rangle} := [a_{i,j,k}],\ i \in I, j \in J, k \in \langle \ell_3 \rangle,\\
&\mathcal{S} = (\langle \ell_1 \rangle \times J \times K) \cup (I \times \langle \ell_2 \rangle \times K) \cup (I \times J \times \langle \ell_3 \rangle).
\end{aligned} \tag{3.4}$$

Then we define $\mathcal{U}_{\text{opt}}$ similarly to (1.2) as solution of a least squares problem, i.e.

$$\mathcal{U}_{\text{opt}} = \arg \min_{\mathcal{U} \in \mathbb{R}^{m_1 \times m_2 \times m_3}} \sum_{(i,j,k) \in \mathcal{S}} (a_{i,j,k} - (\mathcal{U} \times_1 C_1 \times_2 C_2 \times_3 C_3)_{i,j,k})^2. \tag{3.5}$$

A tensor $\mathcal{A}$ is called *generic* if its entries are not the set of zeroes of any finite number of given polynomial equations in the entries of $\mathcal{A}$.

Theorem 3.1 *For a given generic tensor $\mathcal{A} \in \mathbb{R}^{\ell_1 \times \ell_2 \times \ell_3}$ let $I \subset \langle \ell_1 \rangle$, $J \subset \langle \ell_2 \rangle$, and $K \subset \langle \ell_3 \rangle$ be three sets of integers with the particular cardinalities $\#I = p$, $\#J = p$, $\#K = p^2$ (for a given integer p). Let $\mathcal{C}_1 = \mathcal{A}_{\langle \ell_1 \rangle, J, K} \in \mathbb{R}^{\ell_1 \times p \times p^2}$, $\mathcal{C}_2 = \mathcal{A}_{I, \langle \ell_2 \rangle, K} \mathbf{i} \mathbb{R}^{p \times \ell_2 \times p^2}$, and $\mathcal{C}_3 = \mathcal{A}_{I,J,\langle \ell_3 \rangle} \in \mathbb{R}^{p \times p \times \langle \ell_3 \rangle}$ be the subtensors defined as in (3.4). Assume that $C_j \in \mathbb{R}^{m_j \times \ell_j}$ is the matrix induced by the tensor $\mathcal{C}_j$ for $j = 1, 2, 3$.*

Then the minimum given in (3.5) is zero, i.e. there exists $\mathcal{U} \in \mathbb{R}^{p^3 \times p^3 \times p^2}$ such that the tensor $\mathcal{B}$ given by (3.3) has the same entries as $\mathcal{A}$ for $(i, j, k) \in \mathcal{S}$.

Proof. We can represent the tensor $\mathcal{A} = [a_{i,j,k}]$ as a matrix $E = [e_{s,k}] \in \mathbb{R}^{(\ell_1 \cdot \ell_2) \times \ell_3}$. So $e_{s,k} = a_{i,j,k}$ for the corresponding pair of indices $(i, j) \in \langle \ell_1 \rangle \times \langle \ell_2 \rangle$. Then the set of indices $(i, j) \in I \times J$ corresponds to the set of indices $L \subset \langle \ell_1 \cdot \ell_2 \rangle$, where $\#L = p^2$. Since $\mathcal{A}$ is generic it follows that the submatrix $E_{L,K}$ that has row indices in L and column indices in K is invertible. Hence the matrix $\tilde{E} = E_{\langle \ell_1 \cdot \ell_2 \rangle, K} E_{L,K}^{-1} E_{L, \langle \ell_3 \rangle}$ has the same entries as E in the places $\langle \ell_1 \cdot \ell_2 \rangle \times K \cup L \times \langle \ell_3 \rangle$. Equivalently, one can represent $\tilde{E}$ as

$$\mathcal{A}_{\langle \ell_1 \rangle, \langle \ell_2 \rangle, K} E_{L,K}^{-1} \mathcal{A}_{I,J,\langle \ell_3 \rangle}. \tag{3.6}$$

For each $k \in K$ consider the matrix

$$F_k := \mathcal{A}_{\langle \ell_1 \rangle, \langle \ell_2 \rangle, k} = [a_{i,j,k}]_{i,j=1}^{\ell_1, \ell_2} \in \mathbb{R}^{\ell_1 \times \ell_2}.$$

Since $\mathcal{A}$ is generic, $\det(F_k)_{I,J} \neq 0$ for each $k \in K$. Hence F_k can be approximated by $G_k := (F_k)_{\langle \ell_1 \rangle, J} (F_k)_{I,J}^{-1} (F_k)_{I, \langle \ell_2 \rangle}$. Moreover, G_k has the same entries as F_k in the positions $\langle \ell_1 \rangle \times J \cup I \times \langle \ell_2 \rangle$.

Equivalently, we have that

$$\mathcal{A}_{\langle \ell_1 \rangle, J, k} \mathcal{A}_{I,J,k}^{-1} \mathcal{A}_{I, \langle \ell_2 \rangle, k}, \tag{3.7}$$

is an approximation of $\mathcal{A}_{\langle \ell_1\rangle,\langle \ell_2\rangle,k}$, which has the same entries as $\mathcal{A}_{\langle \ell_1\rangle,\langle \ell_2\rangle,k}$ in the positions $\langle \ell_1\rangle \times J \cup I \times \langle \ell_2\rangle$ for $k \in K$. Replacing $\mathcal{A}_{\langle \ell_1\rangle,\langle \ell_2\rangle,k}$ appearing in (3.6) with the expression that appears in (3.7) gives rise to the approximation $\mathcal{B}$ of the form (3.3). Furthermore $\mathcal{B}$ has the same entries as $\mathcal{A}$ for $(i,j,k) \in \mathcal{S}$. $\square$

Note that the Tucker decomposition outlined in Theorem 3.1 is not symmetric with respect to the three coordinates of the 3-tensor $\mathcal{A}$. In certain applications this can be useful, where one coordinate dimension is significantly larger than the other two, see e.g. [25] and the references therein.

Using the construction suggested in the proof of Theorem 3.1, one can derive an algorithm for computing $\mathcal{U}$ which is similar to the approximation algorithm for matrices outlined in Section 2. The complexity of this algorithm would be of order

$$\mathcal{O}(\max(p^3\ell_1, p^3\ell_2, p^2\ell_3) + p^8),$$

namely, the memory needed for the matrices C_1, C_2, C_3 and the tensor $\mathcal{U}$. The numerical properties of this method are currently under investigation.

3.2 4-tensors

The ideas that we have developed in the previous subsection for 3-tensors can be extended easily to 4-tensors. However, in this case we even obtain a result that is symmetric in all coordinates.

Theorem 3.2 *For a given generic tensor $\mathcal{A} \in \mathbb{R}^{\ell_1\times\ell_2\times\ell_3\times\ell_4}$ let $I \subset \langle \ell_1\rangle$, $J \subset \langle \ell_2\rangle$, $K \subset \langle \ell_3\rangle$, and $L \subset \ell_4$ be four sets of integers all of cardinality p and let*

$$\begin{aligned}\mathcal{C}_1 &= \mathcal{A}_{\langle \ell_1\rangle,J,K,L} \in \mathbb{R}^{\ell_1\times p\times p\times p},\\ \mathcal{C}_2 &= \mathcal{A}_{I,\langle \ell_2\rangle,K,L} \in \mathbb{R}^{p\times \ell_2\times p\times p},\\ \mathcal{C}_3 &= \mathcal{A}_{I,J,\langle \ell_3\rangle,K} \in \mathbb{R}^{p\times p\times \langle \ell_3\rangle\times p},\\ \mathcal{C}_4 &= \mathcal{A}_{I,J,K,\langle \ell_4\rangle} \in \mathbb{R}^{p\times p\times p\times \ell_4}\end{aligned}$$

be four sections of $\mathcal{A}$. Let $C_1 \in \mathbb{R}^{p^3\times\ell_1}$, $C_2 \in \mathbb{R}^{p^3\times\ell_2}$, $C_3 \in \mathbb{R}^{p^3\times\ell_3}$, $C_4 \in \mathbb{R}^{p^3\times\ell_4}$ be matrix representations corresponding to the fours sections $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ of $\mathcal{A}$.

Then there exists a tensor $\mathcal{U} \in \mathbb{R}^{p^3\times p^3\times p^3\times p^3}$ such that the entries of the tensor

$$\mathcal{B} = \mathcal{U} \times_1 C_1 \times_2 C_2 \times_3 C_3 \times_4 C_4 \in \mathbb{R}^{\ell_1\times\ell_2\times\ell_3\times\ell_4}. \tag{3.8}$$

coincide with the entries of $\mathcal{A}$ for $(i,j,k,l) \in \mathcal{S}$, where

$$\mathcal{S} := (\langle \ell_1\rangle \times J \times K \times L) \cup (I \times \langle \ell_2\rangle \times K \times L) \cup (I \times J \times \langle \ell_3\rangle \times L) \cup (I \times J \times K \times \langle \ell_4\rangle).$$

Proof. The proof can be obtained in a similar way as that of the 3-tensor case. $\square$

Note that usually the tensor $\mathcal{U}$ will not be unique, since we can *flatten* the given 4 tensor $\mathcal{A}$ to a matrix E in 6 different ways.

4 Experimental Results

In this section, we present several experimental results that demonstrates the usefulness of our algorithm when applied to some synthetic and real image data. The numerical results were run in Matlab [19] on a PC with an Intel(R) Pentium(R) 4 CPU 3.20 GHz processor with 1 GB RAM memory, machine precision $2.2204 \cdot 10^{-16}$ and operating system SuSE Linux 10.2.

For the given matrix A and the computed rank k approximation B, we present the *total relative error (TRE) of the approximation*, defined as

$$\|A - B\|_F / \|A\|_F, \tag{4.1}$$

and the S-average error (SAE) as given in (2.8).

In order to demonstrate the statements made in Theorem 2.2, we compute U_{opt} in two ways. First we solve the least squares problem (2.4) (we denote the solution by U_{opt_1}), then we compute $U_{opt_2} = A^{\dagger}_{I,J}$ as in Theorem 2.2. As before, we denote by $t_{\max}$ the number of trials to find a well conditioned matrix $A_{I,J}$. We compare these results with $\tilde{U}_{opt}$ defined in (2.7) and with the solution of the complete least squares problem (denoted by CLS), given by $U_b = C^{\dagger} A R^{\dagger}$.

First of all, we observe that our algorithm tends to perform better with p and q chosen closer to the matrix rank.

Figure 1 portrays the original image of the Tire picture from the Image Processing Toolbox of Matlab [19], given by a matrix $A \in \mathbb{R}^{205 \times 232}$ of rank 205, the image compression given by the SVD (using the Matlab function svds) [18] of rank 30 and the image compression given by $B_b = C U_b R$.

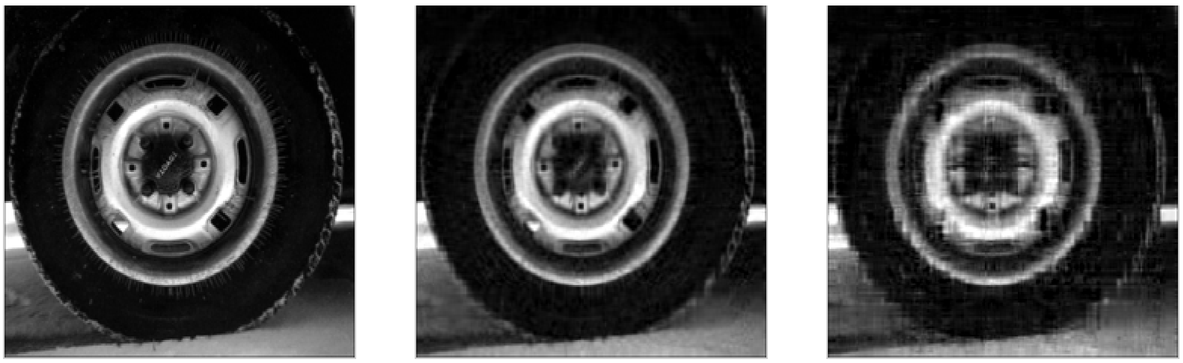

Figure 1: Tire image compression (a) original, (b) SVD approximation, (c) CLS approximation, $t_{\max} = 100$.

The corresponding image compressions given by the approximations B_{opt_1}, B_{opt_2} and $\tilde{B}_{opt}$ are displayed respectively in Figure 2. Here, $t_{\max} = 100$ and $p = q = 30$. Note that the number of trials $t_{\max}$ is set to the large value of 100 for all simulations in order to be able to compare results for different (small and large) matrices.

In Table 1 we present the S-average and total relative errors of the image data compression. Here, $B_b = C U_b R$, $B_{opt_2} = C U_{opt_2} R$ and $\tilde{B}_{opt} = C \tilde{U}_{opt} R$. Table 1 indicates that the less computationally costly FSVD with B_{opt_1}, B_{opt_2} and $\tilde{B}_{opt}$ obtains a smaller S-average error

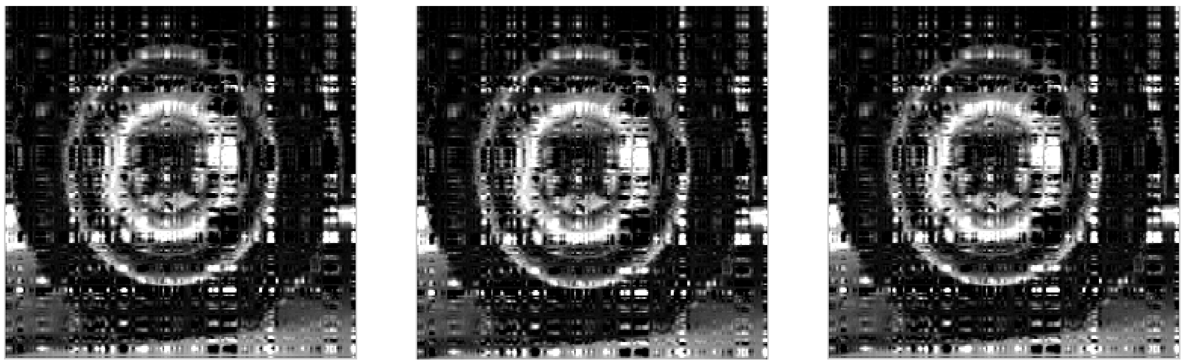

Figure 2: Tire image compression with (a) B_{opt_1}, (b) B_{opt_2}, (c) $\tilde{B}_{opt}$, $t_{\max} = 100$.

	rank	SAE	TRE
B_{svd}	30	0.0072	0.0851
B_b	30	0.0162	0.1920
B_{opt_1}	30	$1.6613 \cdot 10^{-26}$	0.8274
B_{opt_2}	30	$3.2886 \cdot 10^{-29}$	0.8274
$\tilde{B}_{opt}$	30	$1.9317 \cdot 10^{-29}$	0.8274

Table 1: Comparison of rank, S-average error and total relative error.

than the more expensive complete least squares solution CLS and the SVD. On the other hand, CLS and the SVD yield better results in terms of the total relative error. However, it should be noted that CLS is very costly and cannot be applied to very large matrices.

Figure 3 shows the results for the compression of the data for the original image of a camera man from the Image Processing Toolbox of MATLAB [19]. This data is a matrix $A \in \mathbb{R}^{256\times256}$ of rank 253 and the resulting image compression of rank 69 is derived using the SVD and the complete least square approximation CLS given by $B_b = CU_bR$. Notice that there is no discernible difference to the eye in the first two pictures. Figure 4 displays estimates given by

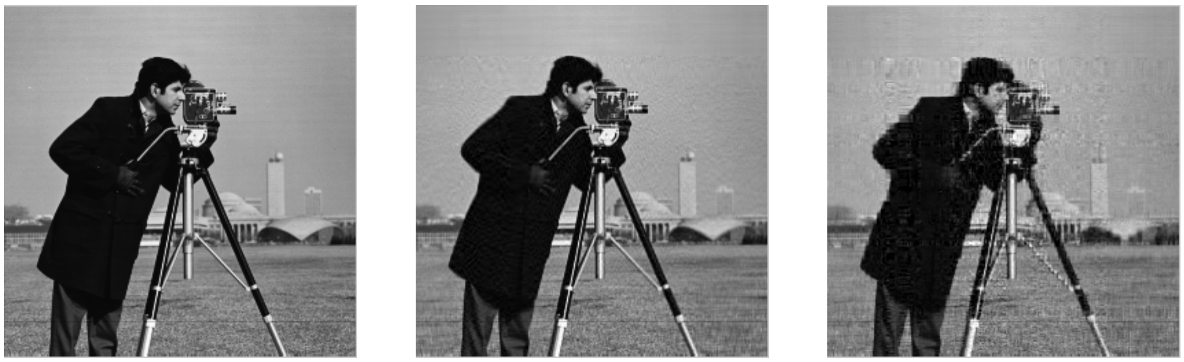

Figure 3: Camera man image compression (a) original, (b) SVD approximation, (c) CLS approximation, $t_{\max} = 100$.

the FSVD approximation $B_{opt_2} = CU_{opt_2}R$ and $\tilde{B}_{opt} = C\tilde{U}_{opt}R$, respectively. Here, we chose $t_{\max} = 100$ and $p = q = 80$. Correspondingly, in Table 2 we provide the resulting S-average and total relative errors. Due to the inability of the Matlab least squares solver to deal with large systems of equations, B_{opt_1} cannot be computed for large data. We see that the FSVD approximation performs well considering the comparison of the computational cost compared with the CLS approximation. Generally we want the TRE to be small, however, since we are not able to compute it for 'real world' applications, we use the SAE as an indicator for our error, since it can be computed effectively.

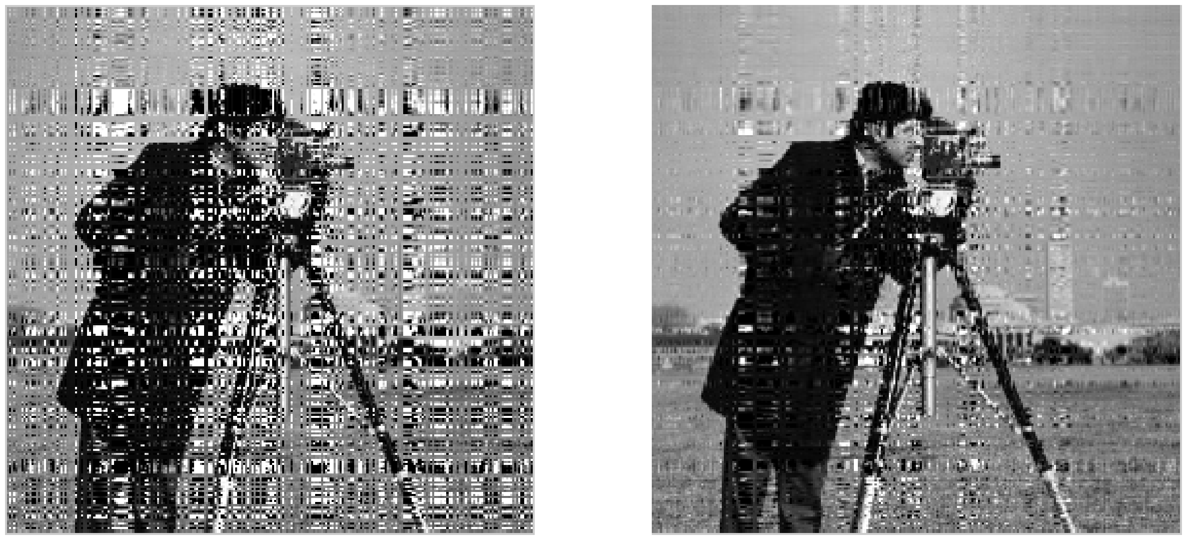

Figure 4: Camera man image compression. FSVD approximation with (a) $B_{opt_2} = CU_{opt_2}R$, (b) $\tilde{B}_{opt} = C\tilde{U}_{opt}R$. $t_{\max} = 100$.

	rank	SAE	TRE
B_{svd}	69	0.0020	0.0426
B_b	80	0.0049	0.0954
B_{opt_1}	–	–	–
B_{opt_2}	80	$3.7614 \cdot 10^{-27}$	1.5154
$\tilde{B}_{opt}$	69	$7.0114 \cdot 10^{-4}$	0.2175

Table 2: Comparison of rank, S-average error and total relative error.

Figure 5 portrays the plots of total relative errors versus number of selected rows and columns for the camera man. The fact that the error stabilizes as the number of rows and columns increases is quite perceptible. This can be attributed to the fact that this increase implies that the rank of the submatrix $A_{I,J}$ is getting closer to the rank of the original matrix and as such will result in a decreasing total relative error. Although, the FSVD converges reasonably well, the SVD and CLS exhibit comparatively faster convergence.

Figure 6 portrays the plot of S-average error versus the number of chosen rows and columns for the camera man. The theory suggests that better choices of p and q will result in better approximations and result in smaller average errors. This logic can be substantiated using the basis that choosing specific singular values from the SVD of $A_{I,J}$. Larger singular values lead to to better approximations. In addition, we would expect the graph to exhibit a decrease in error as the p and q values get larger because the resulting a larger submatrix

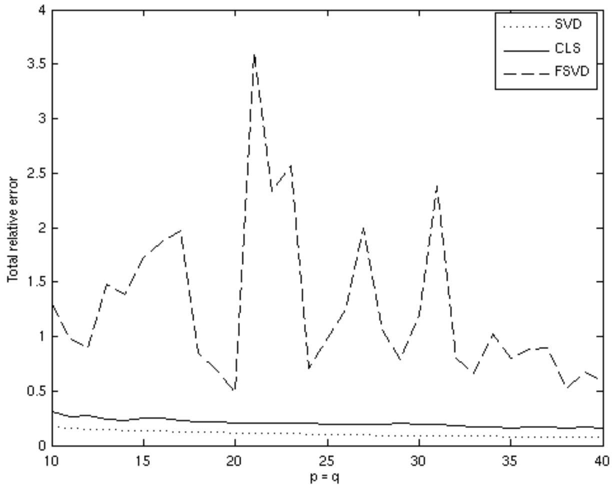

Figure 5: Camera man: total relative errors versus number of selected rows and columns, $t_{\max} = 100$.

$A_{I,J}$ has a greater probability of being well conditioned. This in turn will provide us with a better Moore-Penrose inverse. Correspondingly, observe that the errors converge as p and q approach the matrix rank.

Figures 7 and 8 exhibit the result of the same analysis applied to an original high resolution image of 'Canal at Night' (an image from Max Lyons Digital Image Gallery, Washington DC) [20] given by a matrix $A \in \mathbb{R}^{812\times1200}$ of rank 812. Figure 7 shows the original and the SVD approximation image, of rank 159, which show no 'eye-recognizable' difference while Figure 8 shows the approximations given by CLS and FSVD (with $\tilde{B}_{opt} = C\tilde{U}_{opt}R$) respectively. The parameters here are $t_{\max} = 100$ and $p = q = 200$.

In addition, Table 3 presents the corresponding S-average and total relative errors of the algorithms when performed with the ranks as stated. The results can be seen to be in accordance with the previous results. Hence it can be concluded that the algorithm performs similar for images of different resolutions.

	rank	SAE	TRE
B_{svd}	159	0.0091	0.0951
B_b	200	0.0109	0.1492
B_{opt_1}	–	–	–
B_{opt_2}	200	$4.6677 \cdot 10^{-28}$	2.0919
$\tilde{B}_{opt}$	159	0.0031	0.2246

Table 3: Comparison of rank, S-average error and total relative error.

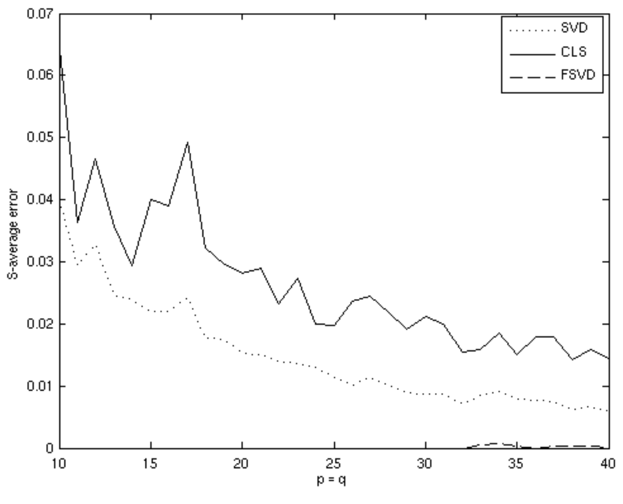

Figure 6: Camera man: S-average errors versus number of selected rows and columns. $t_{\max} = 100$.

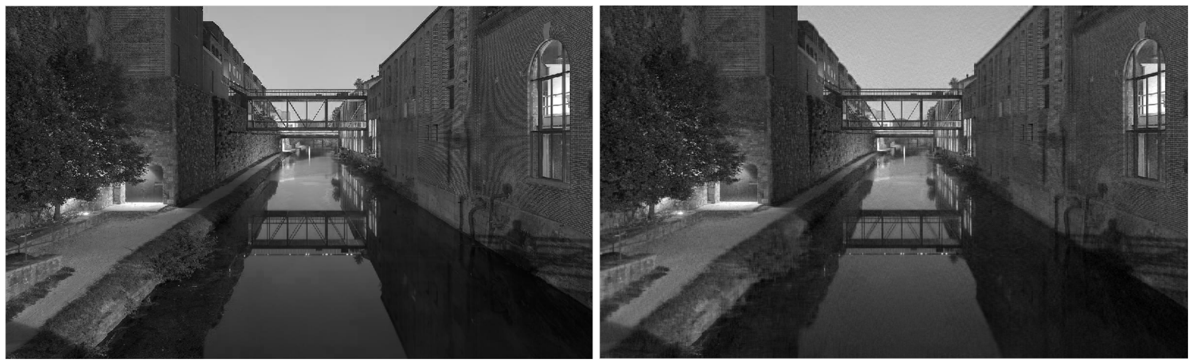

Figure 7: Canal image (a) original, (b) SVD approximation, $t_{\max} = 100$.

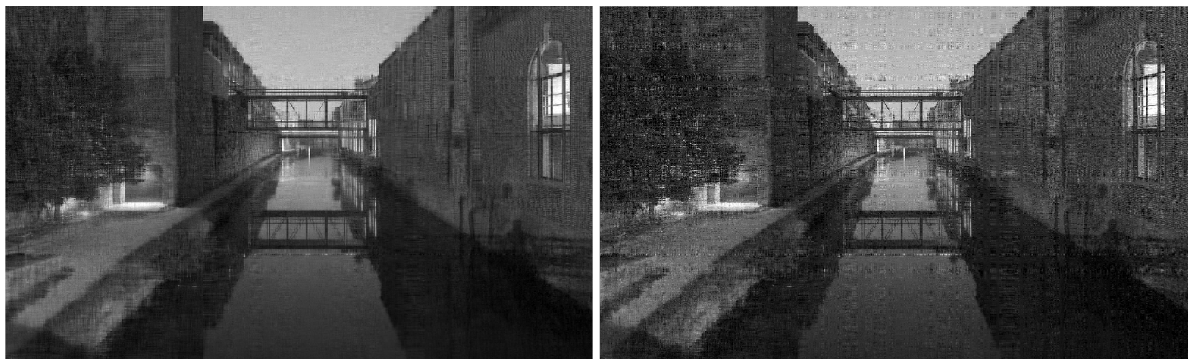

Figure 8: Canal image compression (a) CLS approximation, (b) FSVD with $\tilde{B}_{opt}$, $t_{\max} = 100$.

5 Conclusions

We have introduced an iterative updating procedure for the fast computation of rank k approximations to large dense matrices. We have demonstrated the properties of the methods with real and synthetic data and we have shown how the ideas can be extended to tensors.

References

[1] O. Alter, P. Brown, and D. Botstein. Processing and Modeling Genome-Wide Expression Data Using Singular Value Decomposition, *Microarrays: Optical Technologies and Informatics, Proc. of SPIE* 4266 (2001), 171 – 186.

[2] J. Demmel and W. Kahan. Accurate singular values od bidiagonal matrices, *SIAM J. Sci. Comput.* 11 (1990), 873 – 912.

[3] A. Deshpande and S. Vempala, Adaptive sampling and fast low-rank matrix approximation, *Technical Report TR06-042, Electronic Colloquim on Computational Complexity*, March 2006.

[4] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices I-III: computing a compressed approximate matrix decompositon, *SIAM J. Comput.* 36 (2006), 132 – 206.

[5] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, Polynomial Time Algorithm for Column-Row-Based Relative-Error Low-Rank Matrix Approximation, *Technical Report, DIMACS TR 2006-04*, March 2006.

[6] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, Subspace Sampling and Relative-Error Matrix Approximation: Column-Row-Based Methods, *Proc. 14-th Annual ESA* (2006), 304 – 314.

[7] S. Friedland, M. Kaveh, A. Niknejad and H. Zare. Fast Monte-Carlo low rank approximations for matrices, *Proc. IEEE SoSE* (2006), 218 – 223.

[8] S. Friedland and A. Torokhti. Generalized rank-constrained matrix approximations, *SIAM J. Matrix Anal. Appl.* 29 (2007), 656 – 659.

[9] A. Frieze, R. Kannan, and S. Vempala, Fast Monte-Carlo algorithms for finding low rank approximations, *Journal of the ACM*, 51(6) (2004), 1025 – 1041.

[10] G.H. Golub and C.F. Van Loan. Matrix Computation, *John Hopkins Univ. Press, 3rd Ed.*, Baltimore, 1996.

[11] S.A. Goreinov and E.E. Tyrtyshnikov, The maximum-volume concept in approximation by low-rank matrices, *Contemporary Mathematics* 280 (2001), 47 – 51.

[12] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin, Pseudo-skeleton approximations of matrices, *Reports of the Russian Academy of Sciences* 343(2) (1995), 151 – 152.

[13] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin, A theory of pseudo-skeleton approximations of matrices, *Linear Algebra Appl.* 261 (1997), 1 – 21.

[14] S. Har-Peled. Low rank matrix approximation in linear time. *Manuscript. http://valis.cs.uiuc.edu/ sariel/papers/05/lrank/lrank.pdf*, January 2006.

[15] L. Kamm and J. G. Nagy. Kronecker product and SVD approximations in image restoration, *Linear Algebra Appl.* 284 (1998), 177 – 192.

[16] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack User's Guide : Solution of Large-Scale Eigenvalue Problems With Implicityly Restarted Arnoldi Methods (Software, Environments, Tools), *SIAM Publications*, April 1998.

[17] M. W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data, *Proc. 12-th Annual SIGKDD* (2006), 327 – 336.

[18] Matlab 7.4.0.336 (R2007a), The Math Works Inc..

[19] Matlab Image Processing Toolbox Version 5.4 (R2007a), The Math Works Inc..

[20] M. Lyons. Max Lyons Digital Image Gallery, *http://www.tawbaware.com/maxlyons/index.html*.

[21] I.V. Osedelets, D.V. Savostianov, and E.E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time, *www.inm.ras.ru/library/oseledets/t-cross.pdf*, 2007.

[22] P. Paschou, M. W. Mahoney, A. Javed, J. R. Kidd, A. J. Pakstis, S. Gu, K. K. Kidd, and P. Drineas. Intra-and interpopulation genotype reconstruction from tagging SNPs. *Genome Research.* 17(1) (2007), 96 – 107.

[23] M. Rudelson and R. Vershynin, Sampling from large matrices: an approach through geometric functional analysis, *J. ACM* 54 (2007), no. 4, Art. 21, 19 pp.

[24] T. Sarlos. Improved approximation algorithms for large matrices via random projections. *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006), 143 – 152.

[25] A. Stegeman, J.M.F. ten Berge and L. De Lathauwer. Sufficient conditions for uniqueness in Candecomp/Parafac and Indscal with random component matrices, *Psychometrika* 71 (2006), 219 – 229.

[26] G.W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.* 14 (1993), 494 – 499.

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