Numerical CP Decomposition of Some Difficult Tensors

Petr Tichavský[a,*], Anh Huy Phan[b] and Andrzej Cichocki[b]

[a] *Institute of Information Theory and Automation, Prague 182 08, Czech Republic*
[b] *Brain Science Institute, RIKEN, Wakoshi, Japan.*

Abstract

In this paper, a numerical method is proposed for canonical polyadic (CP) decomposition of small size tensors. The focus is primarily on decomposition of tensors that correspond to small matrix multiplications. Here, rank of the tensors is equal to the smallest number of scalar multiplications that are necessary to accomplish the matrix multiplication. The proposed method is based on a constrained Levenberg-Marquardt optimization. Numerical results indicate the rank and border ranks of tensors that correspond to multiplication of matrices of the size 2×3 and 3×2, 3×3 and 3×2, 3×3 and 3×3, and 3×4 and 4×3. The ranks are 11, 15, 23 and 29, respectively. In particular, a novel algorithm for computing product of matrices of the sizes 3×4 and 4×3 using 29 multiplications is presented.

Keywords: Small matrix multiplication, canonical polyadic tensor decomposition, Levenberg-Marquardt method

1. Introduction

The problem of determining the complexity of matrix multiplication became a well studied topic since the discovery of the Strassen's algorithm [1]. The Strassen's algorithm allows multiplying 2×2 matrices using seven multiplications. A consequence of this algorithm is that $n \times n$ matrices can be multiplied by performing of the order $n^{2.81}$ operations. More recent advances have brought the number of operations needed even closer to the n^2 operations. The current record is $O(n^{2.373})$ operations due to Williams [2].

The problem of the matrix multiplication can be rephrased as a problem of decomposing a particular tensor according to its rank [3]. In short, consider an order-3 tensor $\mathcal{T}$ of the size $I \times J \times K$ having elements $\mathcal{T}_{ijk}$ that admit a canonical polyadic (CP) decomposition

$$\mathcal{T}_{ijk} = \sum_{r=1}^{R} A_{ir} B_{jr} C_{kr} \tag{1}$$

where A_{ir}, B_{jr}, C_{kr} are elements of so-called factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, respectively. We shall use the symbolic notations of Kolda [4], $\mathcal{T} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$. Then, the smallest R such that a CP decomposition (1) exists, is called the tensor rank. A border rank $\underline{R}$ is defined as the smallest integer such that the given tensor $\mathcal{T}$ can be approximated to arbitrary precision by tensors of rank $\underline{R}$.

The lowest number of the scalar multiplications needed to compute the matrix product corresponds to the ranks of certain tensors. It is equivalent to solution to the so-called Brent equation [5]. The focus of this paper is not on improving the above asymptotic results of [2], but on numerical decomposition of tensors that correspond to multiplication of small matrices and determining their rank [6]. Although the problem is quite old, only partial results are known so far.

The matrix multiplication tensor for the 2×2 matrices is already completely clear [7]. It has rank 7 and border rank 7. For the 3×3 case, an algorithm using 23 scalar multiplications was found by Laderman [8]. It means that the rank is at most 23. For multiplying two 4×4 matrices, one can use twice the Strassen's algorithm, and therefore the rank is at most 49. Multiplication of 5×5 matrices was studied by Makarov [9] with the result of 100 multiplications (rank 100).

In this paper we present a numerical decomposition of the matrix multiplication tensors. For now, we are not able to improve the known results of Strassen, Laderman and Makarov, instead we show a method of the decomposition with these ranks and numerical results indicating that further improvements are probably not possible. Moreover, the numerical methods allow to guess the border rank of the tensors. As a new result, we have derived a novel algorithm for multiplying two matrices of the size 3×4 and 4×3 through 29 multiplications.

Traditional numerical tensor decomposition methods include the alternating least squares method (ALS) [10], improved ALS through the enhanced line search (ELS) [11], damped Gauss-Newton method, also known as Levenberg- Marquardt (LM) method [12], and different nonlinear optimization methods, e.g. [13]. For decomposition of the multiplication tensors we have developed a special variant of the constrained LM method. Once an exact representation is found, we propose a method of seeking another representation such that the factor matrices only contain nulls, ones and minus ones.

The rest of the paper is organized as follows. The tensors of the matrix multiplication are introduced in Section 2. The numerical method of their decomposition is presented in Section 3. Section 4 presents numerical results and Section 5 concludes the paper.

2. Tensor of Matrix Multiplication

Consider two matrices $\mathbf{E}$ and $\mathbf{F}$ of the sizes $P \times Q$ and $Q \times S$, respectively, and their matrix product $\mathbf{G} = \mathbf{EF}$ of the size $P \times S$. The operation of the matrix multiplication can be represented by a tensor $\mathcal{T}_{PQS}$ of the size $PQ \times QS \times PS$ which is filled with nulls and ones only, such that

$$\mathrm{vec}(\mathbf{G}) = \mathcal{T}_{PQS} \times_1 \mathrm{vec}(\mathbf{E}^T)^T \times_2 \mathrm{vec}(\mathbf{F}^T)^T \tag{2}$$

regardless of the elements values of $\mathbf{E}$ and $\mathbf{F}$. Here, $\times_i$ denotes a tensor-matrix multiplication along the dimension i, and the operator "vec" stacks all elements of a matrix or tensor in one long column vector.

Note that the number of ones in the tensor $\mathcal{T}_{PQS}$ is PQS; it is the number of scalar multiplications needed for evaluating the matrix product by a conventional matrix multiplication algorithm.

The tensor $\mathcal{T}_{PQS}$ has the elements

$$(\mathcal{T}_{PQS})_{\alpha\beta\gamma} = \delta_{in}\delta_{jk}\delta_{\ell m} \tag{3}$$

where $\alpha = (i-1)Q + j$; $\beta = (k-1)S + \ell$; $\gamma = (m-1)P + n$; $i, n = 1, \ldots, P$; $j, k = 1, \ldots, Q$; $\ell, m = 1, \ldots, S$.

For example,

$$\mathcal{T}_{222} = \left(\begin{array}{cccc|cccc|cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) . \tag{4}$$

This tensor has the size $4 \times 4 \times 4$, and the vertical lines separate the four frontal slices of the tensor.

A canonical polyadic decomposition of the tensor $\mathcal{T}_{PQS}$ is a representation of the tensor as in (1), $\mathcal{T}_{PQS} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$. For example, a CP decomposition of the tensor $\mathcal{T}_{222}$ in (4) corresponding to the Strassen algorithm [2] is $\mathcal{T}_{222} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$ with

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The multiplication tensors have the following properties:

1. Ranks of these tensors exceed the tensors' dimensions.
2. The CP decompositions are not unique.
3. The border ranks of the tensors might be strictly lower than their true ranks.
4. Tensors $\mathcal{T}_{NNN}$ are invariant with respect to some permutations of indices. Using the Matlab notation we can write $\mathcal{T}_{NNN} = \text{permute}(\mathcal{T}_{NNN}, [2,3,1]) = \text{permute}(\mathcal{T}_{NNN}, [3,1,2])$
5. Tensors $\mathcal{T}_{PQS}$ are invariant with respect to certain tensor-matrix multiplications [1].

Let us explain the last item in more details. Since it holds $\mathbf{G} = \mathbf{EF} = (\mathbf{EX})(\mathbf{X}^{-1}\mathbf{F})$ for any invertible matrix $\mathbf{X}$, we have

$$\text{vec}(\mathbf{G}) = \mathcal{T}_{PQS} \times_1 \text{vec}(\mathbf{E}^T)^T \times_2 \text{vec}(\mathbf{F}^T)^T = \mathcal{T}_{PQS} \times_1 \text{vec}(\mathbf{X}^T\mathbf{E}^T)^T \times_2 \text{vec}(\mathbf{F}^T\mathbf{X}^{-T})^T \,.$$

The multiplication with $\mathbf{X}$ and $\mathbf{X}^{-1}$ can be absorbed into $\mathcal{T}_{PQS}$, because

$$\begin{aligned} \text{vec}(\mathbf{X}^T\mathbf{E}^T) &= (\mathbf{I} \otimes \mathbf{X}^T)\text{vec}(\mathbf{E}^T) \\ \text{vec}(\mathbf{F}^T\mathbf{X}^{-T}) &= (\mathbf{X}^{-1} \otimes \mathbf{I})\text{vec}(\mathbf{F}^T) \end{aligned}$$

where $\mathbf{I}$ is an identity matrix of an appropriate size. Therefore

$$\mathcal{T}_{PQS} = \mathcal{T}_{PQS} \times_1 \mathbf{S}_1(\mathbf{X}) \times_2 \mathbf{S}_2(\mathbf{X})$$

where $\mathbf{S}_1(\mathbf{X}) = \mathbf{I} \otimes \mathbf{X}^T$ and $\mathbf{S}_2(\mathbf{X}) = \mathbf{X}^{-1} \otimes \mathbf{I}$.

3. Numerical CP Decomposition

For numerical CP decomposition of the multiplication tensors we propose a three–step procedure.

1. Finding an "exact fit" solution, if it exists.
2. Finding another solution which would be as much sparse as possible.
3. Finding a solution with factor matrices containing only integer (or rational) entries.

Step 1: Finding an Exact Fit

We seek a vector of parameters $\theta_R = [\mathrm{vec}(\mathbf{A})^T, \mathrm{vec}(\mathbf{B})^T, \mathrm{vec}(\mathbf{C})^T]^T$ of the size $3N^2R \times 1$ that minimizes the cost function $\varphi(\theta) = \|\mathcal{T}_N - \hat{\mathcal{T}}(\theta)\|_F^2$ in the compact set $\{\theta \in \mathbb{R}^{3N^2R}; \|\theta\|^2 = c\}$, where c is a suitable constant.

The ordinary (unconstrained) LM algorithm updates θ as

$$\theta \leftarrow \theta - (\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{g} \tag{5}$$

where

$$\mathbf{H} = \mathbf{J}^T\mathbf{J}, \qquad \mathbf{J} = \frac{\partial \mathrm{vec}(\hat{\mathcal{T}}(\theta))}{\partial \theta}, \qquad \mathbf{g} = \mathbf{J}^T \mathrm{vec}(\mathcal{T}_N - \hat{\mathcal{T}}(\theta)) \tag{6}$$

and μ is a damping parameter, which is sequentially updated according to a rule described in [14]. Closed-form expressions for the Hessian $\mathbf{H}$ and gradient $\mathbf{g}$ can be found in [12].

Optimization constrained to the ball is performed by minimizing the cost function in the tangent plane $\{\theta, (\theta - \theta_0)^T\theta_0 = 0\}$ first, where θ_0 is the latest available estimate of θ which obeys the constraint. Let θ_1' be the minimizer in the tangent plane. Then, θ_1' is projected on the ball by an appropriate scale change, $\theta_1 = \theta_1'\sqrt{c}/\|\theta_1'\|$.

Towards computing θ_1', let the following second-order approximation of the cost function be minimized,

$$\varphi(\theta) \approx \varphi(\theta_0) + \mathbf{g}^T(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T\mathbf{H}(\theta - \theta_0) \tag{7}$$

under the linear constraint $(\theta - \theta_0)^T\theta_0 = 0$. We use the method of Lagrange multiplier to get

$$\theta_1' = \theta_0 - \mathbf{H}^{-1}\mathbf{g} + \frac{\theta_0^T\mathbf{H}^{-1}\mathbf{g}}{\|\theta_0\|^2}\,\mathbf{H}^{-1}\theta_0 \ . \tag{8}$$

Instead of using (4) directly, we propose replacing $\mathbf{H}^{-1}$ by $(\mathbf{H} + \mu\mathbf{I})^{-1}$ as in the LM method.

We need to do multiple random initializations to get close to the global minimum of the cost function; in the optimum case it is an exact fit solution, i.e., with $\varphi(\theta) = 0$. The method works well for small matrices. For example, for decomposition of the $\mathcal{T}_{333}$ and constraint $c = 150$ we need only a few random trials to obtain an exact representation. On the other hand, for tensor $\mathcal{T}_{444}$ the false local minima are so numerous that it is almost impossible

to get an exact fit decomposition when the algorithm is started from random initial conditions.

Step 2: Finding a Sparse Representation

For simplicity, we describe a method of finding a sparse CP decomposition in the case of tensors $\mathcal{T}_{NNN}$. Let $\mathcal{T}_{NNN} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$ be an exact CP decomposition with certain $\mathbf{A}, \mathbf{B}, \mathbf{C}$. We have

$$\mathcal{T}_{NNN} = [[\mathbf{S}_1(\mathbf{X})\mathbf{A}, \mathbf{S}_2(\mathbf{X})\mathbf{B}, \mathbf{C}]] \tag{9}$$

where $\mathbf{X}$ is an arbitrary invertible matrix of size $N \times N$.

First, we seek a matrix $\mathbf{X}$ of determinant 1 such that $\|\mathbf{S}_1(\mathbf{X})\mathbf{A}\|_1 + \|\mathbf{S}_2(\mathbf{X})\mathbf{B}\|_1$ is minimized, and update $\mathbf{A}, \mathbf{B}$ as $\mathbf{A} \leftarrow \mathbf{S}_1(\mathbf{X})\mathbf{A}$, $\mathbf{B} \leftarrow \mathbf{S}_2(\mathbf{X})\mathbf{B}$. Here, $\|\cdot\|_1$ means the ℓ_1 norm. We use the Nelder-Mead algorithm [15] for the minimization; In Matlab^{TM} it is implemented as the function "fminsearch". The constraint $\det(\mathbf{X}) = 1$ is attained by considering all elements of $\mathbf{X}$ but one (say the right-lower corner one, X_{NN}) as free variables; The last element of $\mathbf{X}$ is computed from the previous ones as

$$X_{NN} = (1 - \det \mathbf{X}_0) / \det(\mathbf{X}_{1:N-1,1:N-1})$$

where $\mathbf{X}_0$ is obtained by inserting 0 in $\mathbf{X}$ for X_{NN}, and $\det(\mathbf{X}_{1:N-1,1:N-1})$ is the algebraic complement of X_{NN} in $\mathbf{X}$. Then, it can be shown that $\det(\mathbf{X}) = 1$, and the optimization proceeds in the space of dimension $N^2 - 1$.

Similarly, we seek another $\mathbf{X}$ such that $\|\mathbf{S}_1(\mathbf{X})\mathbf{B}\|_1 + \|\mathbf{S}_2(\mathbf{X})\mathbf{C}\|_1$ is minimized, and update $\mathbf{B}$ and $\mathbf{C}$. Next, we seek still another $\mathbf{X}$ such that $\|\mathbf{S}_1(\mathbf{X})\mathbf{C}\|_1 + \|\mathbf{S}_2(\mathbf{X})\mathbf{A}\|_1$ is minimized, and update $\mathbf{C}$ and $\mathbf{A}$.

The sequence of three partial optimizations is repeated until convergence is obtained.

As a result, we obtain $\mathcal{T}_{NNN} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$ where many elements of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are nulls.

Step 3: Finding a Rational Representation

We continue adjusting the exact representation obtained in the previous step by constraining some nonzero elements of θ_R to be 1 or -1. To this end, we sequentially increase the number of elements of θ_R belonging to the set $\{0, 1, -1\}$. In each step, the function $\varphi(\theta_R)$ is minimized, starting from the latest available solution with some other free element of θ_R changed and fixed to 1 or -1. If an exact representation cannot be achieved, another free

TABLE I
Upper bounds for ranks and border ranks of multiplication tensors

acronym	matrix sizes	# of 1's	rank	border rank
222	$2 \times 2, \quad 2 \times 2$	8	7	7
232	$2 \times 3, \quad 3 \times 2$	12	11	10
322	$3 \times 2, \quad 2 \times 2$	12	11	10
332	$3 \times 3, \quad 3 \times 2$	18	15	14
333	$3 \times 3, \quad 3 \times 3$	27	23	21
343	$3 \times 4, \quad 4 \times 3$	36	29	28
443	$4 \times 4, \quad 4 \times 3$	48	40	39
444	$4 \times 4, \quad 4 \times 4$	64	49	49

element is tried instead. At the very end, it might happen that none of the free elements of θ_R can be set to 1 or -1. In that case, we suggest to try the values 2 or -2 or higher. Some other elements of θ_R may become $1/2$ or $-1/2$, or some other rational number.

4. Experiments

4.1. Estimating the tensor rank

For the multiplication tensor $\mathcal{T}_{333}$ it holds $\min \varphi(\theta_{23}) = 0$ under the constraint $\|\theta_{23}\|^2 = 150$. An exact representation can be obtained quite quickly. For a rank–22 approximation of $\mathcal{T}_{333}$, even with a more relaxed constraint $\|\theta_{22}\|^2 = 594$, the lowest possible value of the fit that we were able to find was $\min \varphi(\theta_{22}) = 6.766 \cdot 10^{-5} > 0$. Note that 594 is the total number of elements in the factor matrices for rank 22. These observations indicate that the rank of the tensor $\mathcal{T}_{333}$ is 23. Similarly, if we attempt to decompose the tensor to 21 terms, $\min \varphi(\theta)$ converges to zero for $c \to \infty$. However, for decomposition to 20 terms, $\min \varphi(\theta)$ does not converge to zero. Therefore we make the conjecture that the border rank of the tensor is 21.

A more complete table of numerical results obtained by the above described procedure is as follows. The table shows rank of the exact-fit solutions to the CP decomposition of the tensors obtained by the constrained optimization. The numerical border rank was determined as the minimum rank for which $\min \varphi(\theta)$ constrained by $\|\theta\| = c$ converges to zero for $c \to \infty$. It is not a mathematical proof that the border rank has the displayed values, but an empirical observation. The true border ranks can be theoretically smaller.

The results in the table are rather discouraging for multiplication of the matrices 2×3 with 3×2 and 3×2 with 2×2. Our experiment results

indicate that the necessary number of multiplications is 11 in these two cases. Corresponding algorithm can be obtained by applying the Strassen algorithm to the 2×2 blocks. However, it is not interesting from the computational point of view.

The only novel results are obtained for the cases 3×3 with 3×2 and 3×4 with 4×3. The former case is studied in the next subsection in more details.

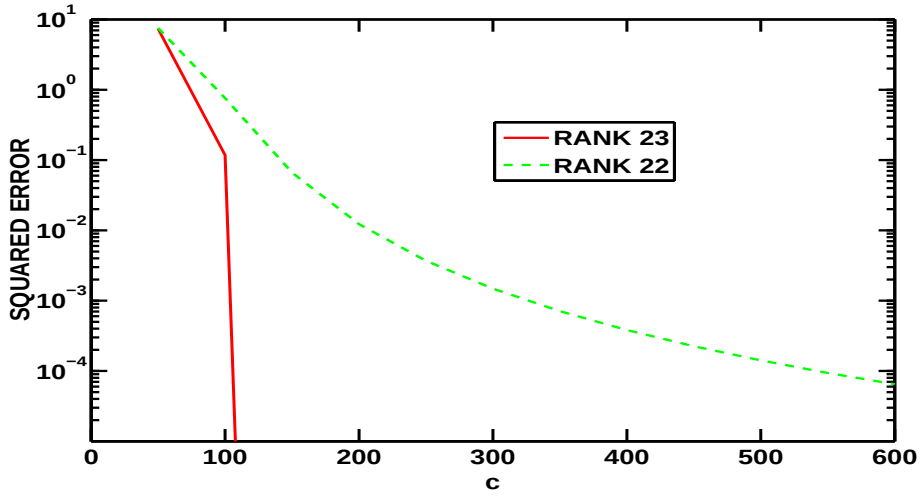

Fig. 1: Reconstruction error as a function of parameter c in the constraint $\|\theta_R\|^2 = c$ for $N = 3$, $R = 22$ and $R = 23$.

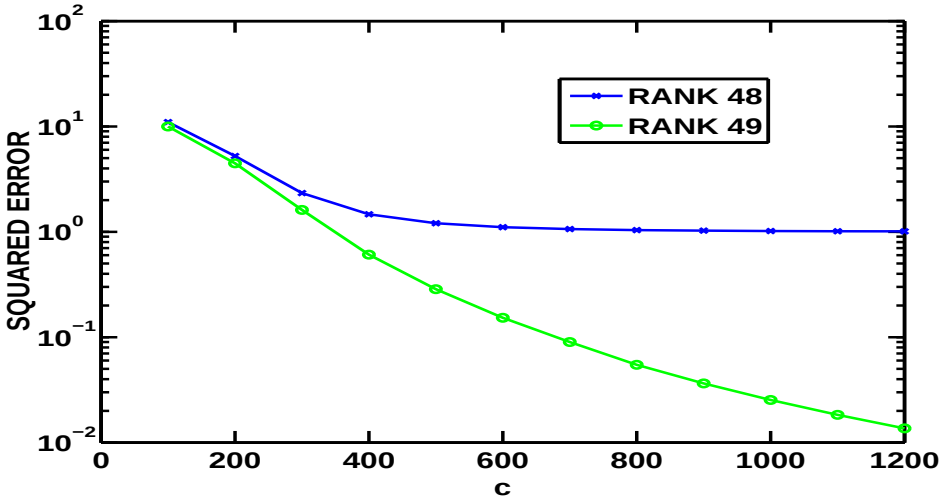

Fig. 2: Reconstruction error as a function of parameter c in the constraint $\|\theta_R\|^2 = c$ for $N = 4$, $R = 48$ and $R = 49$.

4.2. Multiplication of Matrices of the sizes 3×3 *and* 3×2

Consider multiplication of matrices of the sizes 3×3 and 3×2 of the form

$$\begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ f_{31} & f_{32} \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \\ g_{31} & g_{32} \end{pmatrix} \tag{10}$$

Standard algorithm for computing $g_{11}, \ldots, g_{32}$ from $\{e_{ij}\}$ and $\{f_{ij}\}$ requires 18 scalar multiplications. We show that the computation can be accomplished through 15 scalar multiplications only.

The tensor representing the multiplication has dimension $9 \times 6 \times 6$. A CP decomposition of this tensor obtained by applying the proposed algorithm is $\mathcal{T}_{332} = [[A, B, C]]$, where

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \end{pmatrix}$$

The 15 scalar multiplications are

$$\begin{aligned} m_1 &= -(f_{11} - f_{12})(e_{11} + e_{12} - e_{31}) \\ m_2 &= -(f_{11} + f_{32})(e_{13} - e_{31}) \\ m_3 &= -e_{21}(f_{12} - f_{22}) \end{aligned}$$

$$
\begin{aligned}
m_4 &= -(e_{12} - e_{21})(f_{11} - f_{12} + f_{22}) \\
m_5 &= (f_{31} - f_{32})(e_{32} - e_{13} + e_{33}) \\
m_6 &= f_{11}(e_{11} - e_{13} + e_{21}) \\
m_7 &= e_{31}(f_{12} + f_{32}) \\
m_8 &= -e_{12}(f_{11} - f_{12} - f_{21} + f_{22}) \\
m_9 &= e_{23}(f_{21} - f_{31}) \\
m_{10} &= -f_{32}(e_{23} + e_{31} - e_{33}) \\
m_{11} &= f_{21}(e_{12} + e_{22} + e_{23}) \\
m_{12} &= f_{22}(e_{21} + e_{22} - e_{32}) \\
m_{13} &= (e_{23} + e_{32})(f_{21} - f_{31} + f_{32}) \\
m_{14} &= e_{32}(f_{21} - f_{22} - f_{31} + f_{32}) \\
m_{15} &= e_{13}(f_{11} + f_{31})
\end{aligned}
$$

Having computed these products, the elements g_{ij} in (11) can be written as

$$
\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \\ g_{31} & g_{32} \end{pmatrix} = \begin{pmatrix} m_3 - m_4 + m_6 + m_8 + m_{15} & m_1 - m_2 + m_3 - m_4 + m_6 + m_7 \\ -m_3 + m_4 - m_8 - m_9 + m_{11} & -m_3 - m_9 + m_{12} + m_{13} - m_{14} \\ m_2 + m_5 - m_9 + m_{10} + m_{13} + m_{15} & m_7 - m_9 + m_{10} + m_{13} - m_{14} \end{pmatrix} .
$$

A similar but not exactly identical solution to the problem was proposed in [16].

4.3. Multiplication of Matrices of the sizes 3×4 *and* 4×3

Consider multiplication of matrices of the sizes 3×4 and 4×3 of the form

$$
\begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \\ f_{41} & f_{42} & f_{43} \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \qquad (11)
$$

Standard algorithm for computing $g_{11}, \ldots, g_{33}$ from $\{e_{ij}\}$ and $\{f_{ij}\}$ requires 36 scalar multiplications. We show that the computation can be accomplished through 29 scalar multiplications only.
The tensor representing the multiplication has dimension $12 \times 12 \times 9$. A CP decomposition of this tensor obtained by applying the proposed algorithm is

$\mathcal{T}_{343} = [[A, B, C]]$, where

$A =$

$$\left(\begin{array}{cccccccccccccccc}
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}\right.$$

$$\left.\begin{array}{cccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 \\
0 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0
\end{array}\right)$$

$B =$

$$\left(\begin{array}{cccccccccccccccc}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0
\end{array}\right.$$

$$\left.\begin{array}{cccccccccccccccc}
1 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0
\end{array}\right)$$

$C =$

$$\left(\begin{array}{cccccccccccccccc}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & -1 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & -1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right.$$

$$\left.\begin{array}{cccccccccccccccc}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right)$$

4.4. Multiplication of Matrices of the sizes 4×4 *and* 4×3

Standard algorithm for computing the matrix product requires 48 scalar multiplications, see Table 1. We applied the proposed algorithm to CP decomposition of the corresponding tensor $\mathcal{T}_{443}$, which has the size $16 \times 12 \times 12$. One decomposition of the tensor with the rank $R = 40$ is $\mathcal{T}_{443} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$ with

$\mathbf{A} =$

$$\begin{pmatrix}
0 & 1 & 0 & 4/3 & 0 & 0 & 0 & 8/9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
2/3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 4/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 4/9 & 1 & 0 & 0 & 0 & 2/3 & 0 & 0 & 0 & 0 & 3/4 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 2 & 3/8 & 0 & 0 & 4/3 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3/2 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$\begin{pmatrix}
0 & -2/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4/3 & 0 & 0 & 0 & -4/3 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
4/27 & 0 & -4/9 & 4/9 & 2/3 & 0 & 0 & 0 & -4/3 & 0 & 0 & 2/3 & 2 & 0 & 1 & 0 & 2/3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 \\
1/2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2/9 & 0 & 1/3 & 2/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2/3 & 0 & 0 & 1/2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 4 & 0 & 2 & 1/2 & -1 & 0 & 0 \\
1/9 & 0 & -1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & -4 & 0 & -2 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -3/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
2/9 & 0 & -2/3 & 2/3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$\mathbf{B} =$

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1/6 & 0 & 1 & -2/9 & 0 & 0 & -1/3 & 4/3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1/6 & 0 & 0 & 1/4 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & -3/4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 2/9 & 0 & 0 & 1/3 & -1 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & -1/6 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & -1/3 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$\begin{pmatrix}
0 & -1 & 0 & 0 & 0 & 0 & -2/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 4 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 1/3 & 0 & 0 & 0 & 0 & 1/3 & 0 & -1 & 0 & 0 & 0 & -1/12 & 1/3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1/4 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
2/3 & 0 & 0 & -2/3 & 0 & 0 & 2/3 & 0 & 0 & 0 & 0 & 4/9 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\
-1 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & -2/3 & 0 & 0 & 0 & 0 & 0 & 0 & -1/2 & 0
\end{pmatrix}$$

$\mathbf{C} =$

$$\left(\begin{array}{cccccccccccccccccccc}
0 & 0 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9/8 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
-2/3 & 0 & 0 & 0 & 0 & 6 & 0 & 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 0 & -2 & 0 & -2 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & -3/4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 4/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2/9 & -4/3 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & -2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/9 & 0 & -2/3 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1/4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2/3 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2/3 & 0 & 0 & -1
\end{array}\right.$$

$$\left.\begin{array}{cccccccccccccccccccc}
0 & 3/2 & 0 & 0 & 9/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & -1 & 1 & 0 & 0 & 0 & 3/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & -4/3 & 0 & 1 & 0 & 0 & 0 & 3/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1/3 & 1/3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -2 & 0 & 0 & 1 & -1/3 \\
0 & 0 & 1/3 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 3/4 & 0 & 3/4 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1/4 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1/2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1/2 & 0 & 0
\end{array}\right)$$

In this example, we tried several CP decompositions followed by sparsifications and search for integer solutions, unfortunately, we have not yet found a solution with 0's and 1's only, but solutions with rational entries instead. We still believe that a solution with 0's and 1's only exists, but we do not have yet an algorithm to obtain it.

5. Conclusions

The constrained LM algorithm may serve for decomposition of difficult tensors that have the border rank lower than the true rank and when uniqueness is not required. Numerical decomposition of tensors larger than $\mathcal{T}_{443}$, e.g., $\mathcal{T}_{444}$, is still a challenging task. We have provided a decomposition of $\mathcal{T}_{343}$ tensor to 29 rank-one terms, i.e., we showed that product of the matrices 3×4 and 4×3 can be computed through 29 scalar multiplications. Matlab codes of the proposed algorithms are available on the web page of the first author.

Acknowledgements

The work of Petr Tichavský was supported by Czech Science Foundation through project 14-13713S.

References

[1] V. Strassen, Gaussian elimination is not optimal, Numer. Math. 13 (1969) 354–356.

[2] V.V. Williams, Multiplying matrices faster than Coppersmith-Winograd, in Proc. of the 44th Symposium on Theory of Computing, STOC 12, New York, NY, USA, (2012) 887-898.

[3] J. M. Landsberg, Tensors: Geometry and Applications, AMS 2012.

[4] T.G. Kolda and B.W. Bader, Tensor decompositions and applications, SIAM Review, 51 (2009), pp. 455–500.

[5] R. Brent, Algorithms for matrix multiplication, Tech. Report Report TRCS-70-157, Department of Computer Science, Stanford, 52 pages, March 1970. Available at http://maths.anu.edu.au/ brent/pd/rpb002i.pdf

[6] C.-E. Drevet, M.N. Islam, and E. Schost, Optimization techniques for small matrix multiplications, Theoretical Computer Science 412 (2011), 2219-2236.

[7] S. Winograd, On Multiplication of 2×2 Matrices, Linear Algebra and Appl. 4 (1971) 381-388.

[8] J.D. Laderman, A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications, Bul. Amer. Math. Soc. 82 (1976) 126–128.

[9] O.M. Makarov, A noncommutative algorithm for multiplying $5 \times 5-$ matrices using one hundred multiplications, U.S.S.R. Comput. Maths. Math. Phys. 27 (1987) 311–315.

[10] P. Comon, X. Luciani and A. L. F. de Almeida, Tensor decompositions, alternating least squares and other tales, Chemometrics 23 (2009) 393-405.

[11] M. Rajih, P. Comon, and R. A. Harshman, Enhanced line search: A novel method to accelerate PARAFAC, SIAM Journal on Matrix Analysis Appl. 30 (2008) 1148–1171.

[12] A.H. Phan, P. Tichavský and A. Cichocki, Low Complexity Damped Gauss-Newton Algorithms for Parallel Factor Analysis, SIAM J. Matrix Anal. and Appl. 34 (2013) 126–147.

[13] L. Sorber, M. Van Barel, and L. De Lathauwer, Structured data fusion, IEEE J. Selected Topics in Signal Processing 9, (2015) 586–600.

[14] K. Madsen, H. B. Nielsen, O. Tingleff, Methods for nonlinear least squares problems, second ed., Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2004.

[15] J.C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, SIAM J. of Optimization 9 (1998) 112-147.

[16] A.E. Hopcroft and J. Musinski, Duality applied to the complexity of matrix multiplication and other bilinear form, SIAM J. Comput. **2** (1973), 159–173.