Calcolo (2023) 60:3
https://doi.org/10.1007/s10092-022-00498-x

Convergence of a Jacobi-type method for the approximate orthogonal tensor diagonalization

Erna Begović Kovač[1]

Received: 8 September 2021 / Revised: 22 June 2022 / Accepted: 15 November 2022 /
Published online: 29 November 2022

Abstract
For a general third-order tensor $\mathcal{A} \in \mathbb{R}^{n\times n\times n}$ the paper studies two closely related problems, an SVD-like tensor decomposition and an (approximate) tensor diagonalization. We develop a Jacobi-type algorithm that works on $2 \times 2 \times 2$ subtensors and, in each iteration, maximizes the sum of squares of its diagonal entries. We show how the rotation angles are calculated and prove convergence of the algorithm. Different initializations of the algorithm are discussed, as well as the special cases of symmetric and antisymmetric tensors. The algorithm can be generalized to work on higher-order tensors.

Keywords Jacobi-type methods · Convergence · Tensor diagonalization · Tensor decompositions · SVD

Mathematics Subject Classification 15A69 · 65F25 · 65F99

1 Introduction

Singular value decomposition is arguably the most powerful tool of numerical linear algebra. It is not surprising that, when compared to the matrix SVD, the tensor generalization is significantly more complicated, see e.g. [5, 7, 8, 16]. We study the SVD-like tensor decomposition in the Tucker format,

$$\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d, \tag{1.1}$$

where $\mathcal{A}$ and $\mathcal{S}$ are tensors of order d and $U_1, U_2, \dots, U_d$ are orthogonal matrices. Here, the tensor $\mathcal{S}$ mimics the diagonal matrix of singular values from the matrix SVD. It is well known that, in the tensor case, one cannot expect to obtain a

✉ Erna Begović Kovač
ebegovic@fkit.hr

[1] Faculty of Chemical Engineering and Technology, University of Zagreb, Marulićev Trg 19, Zagreb 10000, Croatia

diagonal core tensor $\mathcal{S}$. Hence, our goal will be to get a decomposition (1.1) where $\mathcal{S}$ is “as diagonal as possible”. This SVD-like tensor decomposition problem is closely related to the tensor diagonalization problem. It has many applications in signal processing, blind source separation, and independent component analysis [3, 4, 6].

Problem (1.1) for tensors of order $d = 3$ has been studied by Moravitz Martin and Van Loan [14]. In their paper the authors use a Jacobi-type method to solve the maximization problem stated in (1.2) below. Their numerical results suggest convergence, although a convergence proof is not provided. If the tensor $\mathcal{S}$ from (1.1) is a diagonal tensor, then $\mathcal{A}$ can be diagonalized using orthogonal transformations. Since a general tensor cannot be diagonalized, we aim to achieve an approximate diagonalization. A similar problem for symmetric tensors has been studied in a series of papers by Comon, Li and Usevich [12, 13, 15] where a Jacobi-type method is also a method of choice.

In this paper we develop a Jacobi-type algorithm with the same idea as in [14], to maximize the sum of squares of the diagonal, but the algorithm itself is different from the one in [14]. Moreover, we prove the convergence of our algorithm. Our convergence results are alongside those for the symmetric case from [12, 13, 15]. We are concerned with general tensors, that is, we do not assume any tensor structure, except in Section 5, where we discuss several special cases.

One can observe the problem (1.1) either as a minimization problem where the goal is to minimize the off-diagonal norm of $\mathcal{S}$,

$$\mathrm{off}^2(\mathcal{S}) = \|\mathcal{S}\|_F^2 - \|diag(\mathcal{S})\|_F^2 \to \min,$$

or as a maximization problem,

$$\|diag(\mathcal{S})\|_F^2 \to \max, \tag{1.2}$$

where the square of the Frobenius norm of diagonal entries of $\mathcal{S}$ is maximized. We are going to work with the formulation (1.2). We mainly focus on tensors of order $d = 3$ and develop a block coordinate descent Jacobi-type algorithm for finding the decomposition

$$\mathcal{A} = \mathcal{S} \times_1 U \times_2 V \times_3 W,$$

such that

$$\|diag(\mathcal{S})\|_F^2 = \sum_{i=1}^{n} \mathcal{S}_{iii}^2$$

is maximized. We prove that the algorithm converges to a stationary point of the objective function. As it will be explained later in the paper, the algorithm can easily be generalized to tensors of order $d > 3$.

Our algorithm for an approximate tensor diagonalization can also be used for a low-rank tensor approximation. We can approximate $\mathcal{A}$ by a rank-r tensor $\tilde{\mathcal{A}}$ in the following way. Starting from the decomposition (1.1) we form a diagonal $r \times r \times \cdots \times r$ order d tensor $\mathcal{D}$ such that the diagonal elements of $\mathcal{D}$ are r diagonal elements of $\mathcal{S}$ with the highest absolute values. Moreover, for $i = 1, \dots, r$, we take $U_{i,r}$ as columns of U_i corresponding to the selected diagonal elements. Then, the low-rank approximation is obtained as

$$\tilde{\mathcal{A}} = \mathcal{D} \times_1 U_{1,r} \times_2 U_{2,r} \cdots \times_d U_{d,r}.$$

In Section 2 we describe the problem and construct the algorithm for solving the maximization problem (1.2). We prove the previously mentioned convergence results in Section 3, while in Section 4 we provide several numerical examples. Moreover, in Section 5 we study the special cases of symmetric and antisymmetric tensors.

2 Orthogonal tensor decomposition

2.1 Preliminaries and notation

We use the tensor notation from [10], which is commonly used in the papers dealing with numerical algorithms for tensors. Notation from [11] is also commonly used in multilinear algebra, but somewhat less frequently in its numerical aspects.

Tensors of order three or higher are denoted by calligraphic letters, e.g. $\mathcal{X}$. Tensor *fibers* are vectors obtained from a tensor by fixing all indices but one. For a third-order tensor, its fibers are columns, rows, and tubes. The mode-m *matricization* of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is an $n_m \times (n_1 \cdots n_{m-1} n_{m+1} \cdots n_d)$ matrix $X_{(m)}$ obtained by arranging mode-m fibers of $\mathcal{X}$ into columns of $X_{(m)}$. In this paper we mainly work with 3rd order tensors. Thus, we will have $m = 1, 2, 3$.

The mode-m *product* of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ with a matrix $A \in \mathbb{R}^{p \times n_m}$ is a tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_{m-1} \times p \times n_{m+1} \times \cdots \times n_d}$,

$$\mathcal{Y} = \mathcal{X} \times_m A, \quad \text{such that} \quad Y_{(m)} = A X_{(m)}.$$

Two important properties of the mode-m product are

$$\mathcal{X} \times_m A \times_n B = \mathcal{X} \times_n B \times_m A, \quad m \neq n, \tag{2.1}$$

$$\mathcal{X} \times_n A \times_n B = \mathcal{X} \times_n (BA). \tag{2.2}$$

The *norm* of $\mathcal{X}$ is a generalization of the matrix Frobenius norm. It is given by

$$\|\mathcal{X}\|_F = \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} x_{i_1 i_2 \dots i_d}^2}.$$

To lighten the notation throughout the paper we are going to write this norm simply as $\|\mathcal{X}\|$. The *inner product* of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is given by

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} x_{i_1 i_2 \ldots i_d} y_{i_1 i_2 \ldots i_d}.$$

It is straightforward to check that $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$.

The *Tucker decomposition* is a decomposition of a tensor $\mathcal{X}$ into a core tensor $\mathcal{S}$ multiplied by a matrix in each mode,

$$\mathcal{X} = \mathcal{S} \times_1 M_1 \times_2 M_2 \times_3 \cdots \times_d M_d. \tag{2.3}$$

Tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$ is *diagonal* when $\mathcal{X}_{ijk} \neq 0$ only if $i = j = k$, that is, if $\mathrm{off}(\mathcal{X}) = 0$.

2.2 Problem description

Let $\mathcal{A} \in \mathbb{R}^{n \times n \times n}$. We are looking for an orthogonal Tucker decomposition

$$\mathcal{A} = \mathcal{S} \times_1 U \times_2 V \times_3 W, \tag{2.4}$$

where $U, V, W \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\mathcal{S} \in \mathbb{R}^{n \times n \times n}$ is a core tensor such that

$$\|diag(\mathcal{S})\|^2 = \sum_{i=1}^{n} \mathcal{S}_{iii}^2 \to \max. \tag{2.5}$$

From relation (2.4) tensor $\mathcal{S}$ can be expressed as

$$\mathcal{S} = \mathcal{A} \times_1 U^T \times_2 V^T \times_3 W^T.$$

Hence, in order to solve the problem defined by (2.4) and (2.5) for a given tensor $\mathcal{A}$, we need to find orthogonal matrices U, V, W that maximize the objective function

$$f(U, V, W) = \|diag(\mathcal{A} \times_1 U^T \times_2 V^T \times_3 W^T)\|^2 \to \max. \tag{2.6}$$

We do this using a Jacobi-type method with a block coordinate descent approach.

For the sake of simplicity, our analysis is restricted to equal-sized modes. However, with a few technical adjustments, the same algorithm can be constructed for $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Then, in (2.4) we have $U \in \mathbb{R}^{n_1 \times n_1}$, $V \in \mathbb{R}^{n_2 \times n_2}$, $W \in \mathbb{R}^{n_3 \times n_3}$, and $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

2.3 Jacobi-type algorithm

We now describe the Jacobi-type algorithm for solving the maximization problem defined by (2.6). This is an iterative algorithm. Its kth iteration has the form

$$\mathcal{A}^{(k+1)} = \mathcal{A}^{(k)} \times_1 R_{U,k}^T \times_2 R_{V,k}^T \times_3 R_{W,k}^T, \quad k \geq 0, \quad \mathcal{A}^{(0)} = \mathcal{A}, \tag{2.7}$$

where $R_{U,k}, R_{V,k}, R_{W,k}$ are plane rotations with the following structure,

$$R(i,j,\phi) = \begin{array}{c}\left[\begin{array}{ccccc} I & & & & \\ & \cos\phi & & -\sin\phi & \\ & & I & & \\ & \sin\phi & & \cos\phi & \\ & & & & I \end{array}\right]\end{array}\begin{array}{c} \\ i \\ \\ j \\ \\ \end{array}. \tag{2.8}$$

Index pair (i, j) in the rotation matrix (2.8) is called a *pivot position*. The set of all possible pivot positions is $\{(i,j)\ :\ 1 \le i < j \le n\}$. In the k-th step, matrices $R_{U,k}, R_{V,k}, R_{W,k}$ have the same pivot position (i_k, j_k), while the rotation angle ϕ_k is, in general, different for each matrix.

Our algorithm uses a block coordinate descent approach. This means that each iteration consists of three microiterations where we hold two variables constant and vary the third one. We have

$$\mathcal{B}^{(k)} = \mathcal{A}^{(k)} \times_1 R_{U,k}^T, \tag{2.9}$$

$$\mathcal{C}^{(k)} = \mathcal{B}^{(k)} \times_2 R_{V,k}^T, \tag{2.10}$$

$$\mathcal{A}^{(k+1)} = \mathcal{C}^{(k)} \times_3 R_{W,k}^T. \tag{2.11}$$

Here, by $\mathcal{B}^{(k)}$ and $\mathcal{C}^{(k)}$ we denote the intermediate steps. Of course, if we combine all three microiterations together, using the properties of mode-m product, namely (2.1) and (2.2), we get back to the iteration step (2.7).

Let us see how the rotation angles in matrices $R_{U,k}, R_{V,k}, R_{W,k}$ are computed. For a fixed iteration step k we observe a $2 \times 2 \times 2$ subproblem. Assume that $(i_k, j_k) = (p, q)$, $1 \le p < q \le n$. A subtensor of $\mathcal{A}$ corresponding to an index pair (p, q) is denoted by $\hat{\mathcal{A}}$ and we can write it as

$$\hat{\mathcal{A}}(:,:,1) = \begin{bmatrix} a_{ppp} & a_{pqp} \\ a_{qpp} & a_{qqp} \end{bmatrix}, \quad \hat{\mathcal{A}}(:,:,2) = \begin{bmatrix} a_{ppq} & a_{pqq} \\ a_{qpq} & a_{qqq} \end{bmatrix}.$$

Then, the corresponding $2 \times 2 \times 2$ subproblem is to find 2×2 rotations $\hat{R}_U, \hat{R}_V, \hat{R}_W$ such that

$$\|diag(\hat{\mathcal{S}})\|^2 = \sigma_{ppp}^2 + \sigma_{qqq}^2 \to \max,$$

where

$$\hat{\mathcal{S}} = \hat{\mathcal{A}} \times_1 \hat{R}_U^T \times_2 \hat{R}_V^T \times_3 \hat{R}_W^T,$$

and

$$\hat{\mathcal{S}}(:,:,1) = \begin{bmatrix} \sigma_{ppp} & \sigma_{pqp} \\ \sigma_{qpp} & \sigma_{qqp} \end{bmatrix}, \quad \hat{\mathcal{S}}(:,:,2) = \begin{bmatrix} \sigma_{ppq} & \sigma_{pqq} \\ \sigma_{qpq} & \sigma_{qqq} \end{bmatrix}.$$

Taking only microiteration (2.9) we calculate rotation angles for matrix $\hat{R}_U$. We have

$$\begin{bmatrix} b_{ppp} & b_{pqp} & b_{ppq} & b_{pqq} \\ b_{qpp} & b_{qqp} & b_{qpq} & b_{qqq} \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} a_{ppp} & a_{pqp} & a_{ppq} & a_{pqq} \\ a_{qpp} & a_{qqp} & a_{qpq} & a_{qqq} \end{bmatrix}.$$

The rotation angle ϕ is chosen to maximize the function

$$\begin{aligned} g_1(\phi) &= b_{ppp}^2 + b_{qqq}^2 \\ &= (a_{ppp}\cos\phi + a_{qpp}\sin\phi)^2 + (-a_{pqq}\sin\phi + a_{qqq}\cos\phi)^2. \end{aligned} \tag{2.12}$$

Such ϕ must satisfy relation $g_1'(\phi) = 0$. Taking the derivative of g_1 we get

$$\begin{aligned} g_1'(\phi) &= 2(\cos\phi^2 - \sin\phi^2)(a_{ppp}a_{qpp} - a_{pqq}a_{qqq}) \\ &\quad + 2\cos\phi\sin\phi(a_{pqq}^2 + a_{qpp}^2 - a_{ppp}^2 - a_{qqq}^2) \\ &= 2\cos(2\phi)(a_{ppp}a_{qpp} - a_{pqq}a_{qqq}) \\ &\quad + \sin(2\phi)({a_{pqq}}^2 + a_{qpp}^2 - a_{ppp}^2 - a_{qqq}^2) \\ &= 0. \end{aligned}$$

Dividing this relation by $\cos(2\phi)$ we obtain

$$\tan(2\phi) = \frac{2(a_{ppp}a_{qpp} - a_{pqq}a_{qqq})}{a_{ppp}^2 + a_{qqq}^2 - a_{pqq}^2 - a_{qpp}^2}. \tag{2.13}$$

Similarly, we find the rotation angles for matrices $\hat{R}_V$ and $\hat{R}_W$ as

$$\tan(2\phi) = \frac{2(b_{ppp}b_{pqp} - b_{qpq}b_{qqq})}{b_{ppp}^2 + b_{qqq}^2 - b_{qpq}^2 - b_{pqp}^2} \tag{2.14}$$

and

$$\tan(2\phi) = \frac{2(c_{ppp}c_{ppq} - c_{qqp}c_{qqq})}{c_{ppp}^2 + c_{qqq}^2 - c_{ppq}^2 - c_{qqp}^2}, \tag{2.15}$$

respectively.

In the relations (2.13)–(2.15) it is possible that both the numerator and the denominator are equal to zero. If that happens for one of those relations, we can skip the rotation in the corresponding direction and move on to the next one. If this is the case for all pairs, the algorithm will be terminated and it should be restarted with preconditioning. This will be explained in Section 5 for the case of antisymmetric tensors.

Rotation angles in $\hat{R}_U, \hat{R}_V, \hat{R}_W$ do not need to be calculated explicitly. We only need the sine and the cosine of the corresponding angles. However, once we have formulas for computing $\tan(2\phi)$, there is still a problem of calculating efficiently $\sin\phi$ and $\cos\phi$. We will show how it is done for the rotation in the first mode. The procedure is the same in other modes. We go back to the relation (2.13). Denote

$$\lambda = 2(a_{ppp}a_{qpp} - a_{pqq}a_{qqq})\operatorname{sign}(a_{ppp}^2 + a_{qqq}^2 - a_{pqq}^2 - a_{qpp}^2),$$
$$\mu = |a_{ppp}^2 + a_{qqq}^2 - a_{pqq}^2 - a_{qpp}^2|.$$

Moreover, we denote $t = \tan\phi$. Using the double-angle formula for tangent,

$$\tan(2\phi) = \frac{2t}{1-t^2},$$

relation (2.13) reads

$$\frac{2t}{1-t^2} = \frac{\lambda}{\mu}.$$

This is a quadratic equation in t, $\lambda t^2 + 2\mu t - \lambda = 0$, with solutions $t_1 = \frac{-\mu+\sqrt{\mu^2+\lambda^2}}{\lambda}, t_2 = \frac{-\mu-\sqrt{\mu^2+\lambda^2}}{\lambda}$. Note that the equation for t_1 is numerically unstable because catastrophic cancellation may occur. Therefore, we multiply both numerator and denominator by $\mu + \sqrt{\mu^2 + \lambda^2}$. That way we attain a numerically stable expression $t_1 = \frac{\lambda}{\mu+\sqrt{\mu^2+\lambda^2}}$. Finally,

$$\cos\phi_i = \frac{1}{\sqrt{1+t_i^2}}, \quad \sin\phi_i = \frac{t_i}{\sqrt{1+t_i^2}} = t_i\cos\phi_i, \quad i = 1, 2.$$

We calculate both solutions and use the one that gives the bigger value of the function (2.12).

The order in which we choose pivot pairs is called *pivot strategy*. In our algorithm the pivot strategy is assumed to be cyclic. We choose an ordering of pairs (i, j), $1 \le i < j \le n$, which makes one cycle. Then we repeat the same cycle of pivot pairs until the convergence criterium is satisfied. Common examples of cyclic pivot strategies are row-wise and column-wise strategies with corresponding ordering of pivot pairs defined by

$$\mathcal{O}_r = (1,2),(1,3),\dots,(1,n),(2,3),\dots,(2,n),\dots,(n-1,n) \tag{2.16}$$

and

$$\mathcal{O}_c = (1,2),(1,3),(2,3),\dots,(1,n),(2,n),\dots,(n-1,n), \tag{2.17}$$

respectively. The convergence results from Section 3 hold for any cyclic strategy. Nevertheless, to ensure convergence of the algorithm, pivot pairs should satisfy an additional condition. We only take a pivot pair (i, j) such that (at least) one of the following inequalities is true,

$$|\langle \nabla_Q f, Q\dot{R}(i,j,0)\rangle| \ge \eta \|\nabla_Q f\|_2, \quad \text{for } Q = U, V, W, \tag{2.18}$$

where $0 < \eta \le \frac{2}{n}$, $\dot{R}(i,j,0)$ denotes $\frac{\partial}{\partial\phi}R(i,j,\phi)|_{\phi=0}$, f is defined in (2.6), and the projected gradient $\nabla_Q f$ will be defined in Subsection 3.1. If (i,j) does not satisfy any of the conditions (2.18), then it will be skipped and we move to the next pair in the cycle. It will be shown in Lemma 3.2 that for each inequality (2.18) it is always possible to find an appropriate pivot pair.

In the kth step of the algorithm, when we have $\mathcal{A}^{(k)}$, U_k, V_k, W_k, we first compute the sine and the cosine of the rotation angle in the rotation matrix $R_{U,k}$. We compute the auxiliary tensor $\mathcal{B}^{(k)}$,

$$\mathcal{B}^{(k)} = \mathcal{A}^{(k)} \times_1 R_{U,k}^T = \mathcal{A} \times_1 (R_{U,k}^T U_k^T) \times_2 V_k^T \times_3 W_k^T,$$

and

$$U_{k+1} = U_k R_{U,k}.$$

Then we repeat this procedure in the other modes. This is summarized in Algorithm 2.1.

Algorithm 2.1. Jacobi-type algorithm for the approximate tensor diagonalization

```
Input: $\mathcal{A} \in \mathbb{R}^{n\times n\times n}$.
Output: orthogonal matrices $U, V, W$
$k = 0$
$\mathcal{A}^{(0)} = \mathcal{A}$
$U_0 = V_0 = W_0 = I$
repeat
    Choose pivot pair $(i, j)$.
    if $(i, j)$ satisfies (2.18) for $Q = U$ then
        Find $\cos\phi_k$ and $\sin\phi_k$ for $R_{U,k}$ using (2.13).
        $\mathcal{B} = \mathcal{A}^{(k)} \times_1 R_{U,k}$
        $U_{k+1} = U_k R_{U,k}$
    end if
    if $(i, j)$ satisfies (2.18) for $Q = V$ then
        Find $\cos\phi_k$ and $\sin\phi_k$ for $R_{V,k}$ using (2.14).
        $\mathcal{C} = \mathcal{B} \times_2 R_{V,k}$
        $V_{k+1} = V_k R_{V,k}$
    end if
    if $(i, j)$ satisfies (2.18) for $Q = W$ then
        Find $\cos\phi_k$ and $\sin\phi_k$ for $R_{W,k}$ using (2.15).
        $\mathcal{A}^{(k+1)} = \mathcal{C} \times_3 R_{W,k}$
        $W_{k+1} = W_k R_{W,k}$
    end if
until convergence
```

We have several remarks regarding the Algorithm 2.1.

- Algorithm 2.1 employs the identity initialization $U_0 = V_0 = W_0 = I$. This is not necessarily done this way and it will be further discussed within the numerical examples in Section 4, as well as in relation with the antisymmetric tensors in Section 5.

- It is not needed to explicitly form rotation matrices and tensor matricizations in order to perform mode-n multiplications in the algorithm.
- Conditions on pivot pairs (2.18) can be simplified to lower the computational effort. This will be shown after Lemma 3.2. Moreover, the coefficient $0 < \eta \leq \frac{2}{n}$ can vary, which will be examined in Section 4.

This algorithm can be generalized for the order-d tensors where $d > 3$. In that case we need to obtain orthogonal matrices $U_1, U_2, \dots U_d$ such that maximization condition (1.2) holds. One iteration of the algorithm consists of d microiterations that are analogues of those in (2.9), (2.10), and (2.11).

3 Convergence results

3.1 Gradient of the objective function

Before we move on to the convergence of the Algorithm 2.1, let us say something about the gradient of the objective function $f : O_n \times O_n \times O_n \to \mathbb{R}$,

$$f(U,V,W) = \|diag(\mathcal{A} \times_1 U^T \times_2 V^T \times_3 W^T)\|^2, \tag{3.1}$$

where O_n stands for the group of orthogonal matrices of order n. To calculate ∇f we need an auxiliary function $\tilde{f} : \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \to \mathbb{R}$ defined by the same formula (3.1) as f. In other words, function $\tilde{f}$ is such that f is the restriction of $\tilde{f}$ to the set of triples of orthogonal matrices. Then ∇f is the projection of $\nabla\tilde{f}$ onto the tangent space at (U, V, W) to the manifold $O_n \times O_n \times O_n$. We have

$$\begin{aligned}
\nabla f(U,V,W) &= \left[\, \nabla_U f(U,V,W) \;\; \nabla_V f(U,V,W) \;\; \nabla_W f(U,V,W) \,\right] \\
&= Proj\left[\, \nabla_U \tilde{f}(U,V,W) \;\; \nabla_V \tilde{f}(U,V,W) \;\; \nabla_W \tilde{f}(U,V,W) \,\right] \\
&= \left[\, U\Lambda(U) \;\; V\Lambda(V) \;\; W\Lambda(W) \,\right],
\end{aligned} \tag{3.2}$$

where

$$\Lambda(Q) := \frac{Q^T \nabla_Q \tilde{f} - (\nabla_Q \tilde{f})^T Q}{2}. \tag{3.3}$$

To calculate $\nabla f(U,V,W)$ we write $\tilde{f}$ as

$$\tilde{f}(U,V,W) = \|diag(\mathcal{A} \times_1 U^T \times_2 V^T \times_3 W^T)\|^2 = \sum_{l=1}^{n} \left(\sum_{i,j,k=1}^{n} a_{ijk} u_{il} v_{jl} w_{kl} \right)^2.$$

Element-wise, we get

$$
\begin{aligned}
\frac{\partial \tilde{f}}{\partial u_{ml}} &= 2\left(\sum_{i,j,k=1}^{n} a_{ijk}u_{il}v_{jl}w_{kl}\right)\left(\sum_{j,k=1}^{n} a_{mjk}v_{jl}w_{kl}\right)\\
&= 2\big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{lll}\big(\mathcal{A}\times_2 V^T \times_3 W^T\big)_{mll},\\
\frac{\partial \tilde{f}}{\partial v_{ml}} &= 2\big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{lll}\big(\mathcal{A}\times_1 U^T \times_3 W^T\big)_{lml},\\
\frac{\partial \tilde{f}}{\partial w_{ml}} &= 2\big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{lll}\big(\mathcal{A}\times_1 U^T \times_2 V^T\big)_{llm}.
\end{aligned}
$$

Then, we can use the above relations together with (3.3) to get an explicit expression for $\Lambda(U)$,

$$
\begin{aligned}
(\Lambda(U))_{lp} &= \frac{1}{2}\left(\sum_{m=1}^{n} u_{ml}\frac{\partial \tilde{f}}{\partial u_{mp}} - \sum_{m=1}^{n}\frac{\partial \tilde{f}}{\partial u_{ml}}u_{mp}\right)\\
&= \big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{ppp}\big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{lpp}\\
&\quad - \big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{lll}\big(\mathcal{A}\times_1 U^T \times_2 V^T \times_3 W^T\big)_{pll}.
\end{aligned}
$$

Similarly we get the expressions for $\Lambda(V)$ and $\Lambda(W)$.

The gradient of the objective function will be needed in order to prove the following convergence result and also to check the pivot conditions (2.18).

3.2 Convergence theorem

The convergence of Algorithm 2.1 is given in Theorem 3.1.

Theorem 3.1 *Every accumulation point* (U, V, W) *obtained by Algorithm* 2.1 *is a stationary point of the function f defined by* (3.1).

The proof follows the idea from [9], which was also used in [13], as well as in [1, 2]. The major obstacle is that here we have a function of three variables, while earlier this procedure was used with single-variable functions. We prove Lemma 3.2, which is an adaptation of Lemma 3.1 from [13]. Then, using Lemma 3.2 we prove Lemma 3.4, which is an essential step in the proof of Theorem 3.1.

Lemma 3.2 *For any differentiable function* $f : O_n \times O_n \times O_n \to \mathbb{R}$, $U, V, W \in O_n$, *and* $0 < \eta \leq \frac{2}{n}$ *it is always possible to find index pairs* (i_U, j_U), (i_V, j_V), (i_W, j_W) *satisfying pivot condition* (2.18).

Proof Observe that

$$\dot{R}(i,j,0) = e_j e_i^T - e_i e_j^T.$$

From the definition of the operator Λ we see that matrix $\Lambda(U)$ is skew-symmetric. Then, from the fact that the Euclidean norm is invariant under unitary transformations and from relation (3.2) we have

$$\begin{aligned} |\langle \nabla_U f(U,V,W), U\dot{R}(i,j,0)\rangle| &= |\langle U\Lambda(U), U\dot{R}(i,j,0)\rangle| \\ &= |\langle \Lambda(U), \dot{R}(i,j,0)\rangle| = 2|\Lambda(U)_{ij}|. \end{aligned} \tag{3.4}$$

We can always find an index pair (i_U, j_U) such that

$$|\Lambda(U)_{i_U j_U}| \geq \frac{1}{n}\|\Lambda(U)\|_2.$$

Inserting this into equation (3.4) with $(i,j) = (i_U, j_U)$, we get

$$|\langle \nabla_U f(U,V,W), U\dot{R}(i_U,j_U,0)\rangle| \geq \frac{2}{n}\|\Lambda(U)\|_2 \geq \eta\|\Lambda(U)\|_2 = \eta\|\nabla_U f(U,V,W)\|_2,$$

which proves assertion (i). Since the matrices $\Lambda(V)$ and $\Lambda(W)$ are also skew-symmetric, in the same way we obtain

$$|\langle \nabla_V f(U,V,W), V\dot{R}(i_V,j_V,0)\rangle| \geq \frac{2}{n}\|\Lambda(V)\|_2 \geq \eta\|\Lambda(V)\|_2 = \eta\|\nabla_V f(U,V,W)\|_2,$$
$$|\langle \nabla_W f(U,V,W), W\dot{R}(i_W,j_W,0)\rangle| \geq \frac{2}{n}\|\Lambda(W)\|_2 \geq \eta\|\Lambda(W)\|_2 = \eta\|\nabla_W f(U,V,W)\|_2.$$

This proves assertions (ii) and (iii), respectively. □

Remark 3.3 Conditions (2.18) are equivalent to

$$2|\Lambda(Q)_{ij}| \geq \eta\|\Lambda(Q)\|_2, \quad \text{for } Q = U, V, W,$$

where $\Lambda(\cdot)$ is as in relation (3.3).

Lemma 3.4 *Let U_k, V_k, W_k, $k \geq 0$ be the sequences generated by Algorithm* 2.1. *Let $\overline{U}, \overline{V}, \overline{W}$ be a triple of orthogonal matrices satisfying $\nabla f(\overline{U}, \overline{V}, \overline{W}) \neq 0$. Then there exist $\epsilon > 0$ and $\delta > 0$ such that*

$$\|U_k - \overline{U}\|_2 < \epsilon, \quad \|V_k - \overline{V}\|_2 < \epsilon, \quad \|W_k - \overline{W}\|_2 < \epsilon$$

implies

$$f(U_{k+1}, V_{k+1}, W_{k+1}) - f(U_k, V_k, W_k) \geq \delta. \tag{3.5}$$

Proof Let us fix the iteration step k. To shorten the notation set $\phi_U = \phi_{U,k}$, $\phi_V = \phi_{V,k}$, $\phi_W = \phi_{W,k}$, and $R_{U,k} = R(i_k, j_k, \phi_U)$, $R_{V,k} = R(i_k, j_k, \phi_V)$, $R_{W,k} = R(i_k, j_k, \phi_W)$. We define three functions $h_k^{(1)}, h_k^{(2)}, h_k^{(3)} : \mathbb{R} \to \mathbb{R}$,

$$\begin{aligned} h_k^{(1)}(\phi_1) &= f(U_kR(i_k,j_k,\phi_1),V_k,W_k),\\ h_k^{(2)}(\phi_2) &= f(U_kR_{U,k},V_kR(i_k,j_k,\phi_2),W_k),\\ h_k^{(3)}(\phi_3) &= f(U_kR_{U,k},V_kR_{V,k},W_kR(i_k,j_k,\phi_3)). \end{aligned}$$

Further on, we define yet another function $h_k : \mathbb{R}\times\mathbb{R}\times\mathbb{R}\to\mathbb{R}$,

$$h_k(\phi_1,\phi_2,\phi_3) = f(U_kR(i_k,j_k,\phi_1),V_kR(i_k,j_k,\phi_2),W_kR(i_k,j_k,\phi_3)).$$

Since $R(i_k,j_k,0)=I$,

$$h_k(0,0,0) = f(U_k,V_k,W_k).$$

From the construction of Algorithm 2.1 we know that

$$\begin{aligned} \max_{\phi_1} h_k^{(1)}(\phi_1) &= h_k^{(1)}(\phi_U) = f(U_kR_{U,k},V_k,W_k),\\ \max_{\phi_2} h_k^{(2)}(\phi_2) &= h_k^{(2)}(\phi_V) = f(U_kR_{U,k},V_kR_{V,k},W_k),\\ \max_{\phi_3} h_k^{(3)}(\phi_3) &= h_k^{(3)}(\phi_W) = f(U_kR_{U,k},V_kR_{V,k},W_kR_{W,k}), \end{aligned}$$

and the kth step of the algorithm is represented by

$$h_k(\phi_U,\phi_V,\phi_W) = f(U_kR_{U,k},V_kR_{V,k},W_kR_{W,k}) = f(U_{k+1},V_{k+1},W_{k+1}).$$

Moreover, it is easy to see from the algorithm that

$$\begin{aligned} f(U_{k+1},V_{k+1},W_{k+1}) &\geq f(U_{k+1},V_{k+1},W_k)\\ &\geq f(U_{k+1},V_k,W_k)\\ &\geq f(U_k,V_k,W_k). \end{aligned} \tag{3.6}$$

In order to attain inequality (3.5) we need at least one sharp inequality in (3.6).

If $\nabla f(\overline{U},\overline{V},\overline{W}) \neq 0$, then at least one partial gradient of f is not zero, that is

$$\nabla_U f(\overline{U},\overline{V},\overline{W}) \neq 0,\ \nabla_V f(\overline{U},\overline{V},\overline{W}) \neq 0,\ \text{or}\ \nabla_W f(\overline{U},\overline{V},\overline{W}) \neq 0.$$

Let us assume that $\nabla_U f(\overline{U},\overline{V},\overline{W}) \neq 0$. Then there exists $\epsilon > 0$ such that

$$\mu_1 := \min\{\|\nabla_U f(U,V,W)\|_2 \ :\ \|U-\overline{U}\|_2 < \epsilon\} > 0. \tag{3.7}$$

We use the Taylor expansion of the function $h_k^{(1)}$ around 0,

$$h_k^{(1)}(\phi_1) = h_k^{(1)}(0) + (h_k^{(1)})'(0)\phi_1 + \frac{1}{2}(h_k^{(1)})''(\xi)\phi_1^2, \quad 0<\xi<\phi_1.$$

Set $M_1 = \max|(h_k^{(1)})''(\xi)| < \infty$. Then we have

$$h_k^{(1)}(\phi_1) - h_k^{(1)}(0) \geq (h_k^{(1)})'(0)\phi_1 - \frac{1}{2}M_1\phi_1^2. \tag{3.8}$$

The derivative of $h_k^{(1)}$ is given by

$$(h_k^{(1)})'(\phi_1) = \langle \nabla_U f(U_k R(i_k, j_k, \phi_1), V_k, W_k), U_k \dot{R}(i_k, j_k, \phi_1)\rangle.$$

In particular,

$$(h_k^{(1)})'(0) = \langle \nabla_U f(U_k, V_k, W_k), U_k \dot{R}(i_k, j_k, 0)\rangle. \tag{3.9}$$

It follows from Lemma 3.2(i) and relation (3.9) that

$$|(h_k^{(1)})'(0)| \geq \eta \|\nabla_U f(U_k, V_k, W_k)\|_2. \tag{3.10}$$

Therefore, from (3.10) and (3.7) we get

$$|(h_k^{(1)})'(0)| \geq \eta\mu_1 > 0. \tag{3.11}$$

We go back to the inequality (3.8). For $\phi_1 = \frac{1}{M_1}(h_k^{(1)})'(0)$, using the definition of the function $h_k^{(1)}$ and relations (3.6) and (3.11), we obtain

$$\begin{aligned}
&f(U_{k+1}, V_{k+1}, W_{k+1}) - f(U_k, V_k, W_k)\\
&\geq f(U_{k+1}, V_k, W_k) - f(U_k, V_k, W_k) = h_k^{(1)}(\phi_1) - h_k^{(1)}(0)\\
&\geq (h_k^{(1)})'(0)\phi_1 - \frac{1}{2}M_1\phi_1^2 = \frac{1}{M_1}((h_k^{(1)})'(0))^2 - \frac{1}{2M_1}((h_k^{(1)})'(0))^2\\
&\geq \frac{\eta^2\mu_1^2}{2M_1} = \delta > 0.
\end{aligned}$$

Now we assume that $\nabla_U f(\overline{U}, \overline{V}, \overline{W}) = 0$ and $\nabla_V f(\overline{U}, \overline{V}, \overline{W}) \neq 0$. There is an $\epsilon > 0$ such that

$$\mu_2 := \min\{\|\nabla_V f(U, V, W)\|_2 \ : \ \|V - \overline{V}\|_2 < \epsilon\} > 0. \tag{3.12}$$

In this case we use the Taylor expansion of the function $h_k^{(2)}$ around 0. We have

$$h_k^{(2)}(\phi_2) - h_k^{(2)}(0) \geq (h_k^{(2)})'(0)\phi_2 - \frac{1}{2}M_2\phi_1^2, \tag{3.13}$$

for $M_2 = \max |(h_k^{(2)})''(\xi)| < \infty$, and

$$(h_k^{(2)})'(0) = \langle \nabla_V f(U_k, V_k, W_k), V_k \dot{R}(i_k, j_k, 0)\rangle.$$

Lemma 3.2(ii) and relation (3.12) imply

$$|(h_k^{(2)})'(0)| \geq \eta \|\nabla_V f(U_k, V_k, W_k)\|_2 \geq \eta\mu_2 > 0. \tag{3.14}$$

The assertion of the lemma follows from (3.13), (3.6), and (3.14) with $\phi_2 = \frac{1}{M_2}(h_k^{(2)})'(0)$.

Finally, if $\nabla_U f(\overline{U}, \overline{V}, \overline{W}) = 0$ and $\nabla_V f(\overline{U}, \overline{V}, \overline{W}) = 0$, since $\nabla f(U, V, W)(\overline{U}, \overline{V}, \overline{W}) \neq 0$, then it must be that $\nabla_W f(\overline{U}, \overline{V}, \overline{W}) \neq 0$. Then, there is an $\epsilon > 0$ such that

$$\mu_3 := \min\{\|\nabla_W f(U, V, W)\|_2 \ : \ \|W - \overline{W}\|_2 < \epsilon\} > 0. \tag{3.15}$$

Here we need the Taylor expansion of $h_k^{(3)}$ around 0,

$$h_k^{(3)}(\phi_3) - h_k^{(3)}(0) \geq (h_k^{(3)})'(0)\phi_1 - \frac{1}{2}M_3\phi_3^2, \tag{3.16}$$

for $M_3 = \max|(h_k^{(2)})''(\xi)| < \infty$. We repeat the same steps as for the preceding two cases. We have

$$(h_k^{(3)})'(0) = \langle \nabla_W f(U_k, V_k, W_k), W_k\dot{R}(i_k, j_k, 0)\rangle,$$

and, using Lemma 3.2(iii) and the relation (3.15), it follows that

$$|(h_k^{(3)})'(0)| \geq \eta\|\nabla_W f(U_k, V_k, W_k)\|_2 \geq \eta\mu_3 > 0. \tag{3.17}$$

We attain inequality (3.5) using (3.16), (3.6), and (3.17) with $\phi_3 = \frac{1}{M_3}(h_k^{(3)})'(0)$. □

Using Lemma 3.4 we can now prove Theorem 3.1.

Proof of Theorem 3.1 Suppose that $\overline{U}$, $\overline{V}$, $\overline{W}$ are, respectively, accumulation points of the sequences $\{U_j\}_{j\geq 1}$, $\{V_j\}_{j\geq 1}$, $\{W_j\}_{j\geq 1}$ generated by Algorithm 2.1. Then there are subsequences $\{U_j\}_{j\in K_U}$, $\{V_j\}_{j\in K_V}$, $\{W_j\}_{j\in K_W}$ such that

$$\{U_j\}_{j\in K_U} \to \overline{U}, \quad \{V_j\}_{j\in K_V} \to \overline{V}, \quad \{W_j\}_{j\in K_W} \to \overline{W},$$

where $K_U, K_V, K_W \subseteq \mathbb{N}$.

Assume that $(\overline{U}, \overline{V}, \overline{W})$ is not a stationary point of the function f, that is

$$\nabla f(\overline{U}, \overline{V}, \overline{W}) \neq 0. \tag{3.18}$$

Then, for any $\epsilon > 0$, there are $k_0^{(U)} \in K_U$, $k_0^{(V)} \in K_V$, $k_0^{(W)} \in K_W$ such that

$$\|U_k - \hat{U}\|_2 < \epsilon, \quad \|V_k - \hat{V}\|_2 < \epsilon, \quad \|W_k - \hat{W}\|_2 < \epsilon,$$

for every $k > k_0$, $k_0 = \max\{k_0^{(U)}, k_0^{(V)}, k_0^{(W)}\}$. Thus, Lemma 3.4 implies $f(U_{k+1}, V_{k+1}, W_{k+1}) - f(U_k, V_k, W_k) \geq \delta > 0$. It follows that

$$f(U_k, V_k, W_k) \to \infty,$$

when $k \to \infty$. Since f is continuous, if (U_k, V_k, W_k) converges, then $f(U_k, V_k, W_k)$ should converge, too. This gives a contradiction. Therefore, assumption (3.18) cannot hold and $(\overline{U}, \overline{V}, \overline{W})$ is a stationary point of f. □

Note that all results from this section can be generalized to order-d tensors, $d > 3$.

4 Numerical examples

We illustrate the convergence of Algorithm 2.1 through several numerical examples. We observe the relative off-norm of a tensor $\mathcal{A}$, which is given as

$$\frac{\operatorname{off}(\mathcal{A})}{\|\mathcal{A}\|}. \tag{4.1}$$

For a diagonal tensor, value (4.1) is equal to zero, while for a random tensor it is, typically, close to one. Note that the off-norm is not a norm because it can be equal to zero for a nonzero input.

Figure 1 shows the change of the relative off-norm. We distinguish two different situations, one where a tensor can be completely diagonalized using orthogonal transformations, and a more general one where orthogonal diagonalization is not possible. For the first set of tensors we get $\frac{\operatorname{off}(\mathcal{A})}{\|\mathcal{A}\|} = 0$. Otherwise, we get the convergence to some value between 0 and 1. A random diagonalizable tensor is constructed by taking a diagonal tensor with random entries on the diagonal and multiplying it in each mode by orthogonal matrices obtained from QR factorizations of three random matrices. The algorithm uses row-wise cyclic pivot ordering (2.16) with different values of the parameter η.

In Figure 2 we compare five different pivot orderings. In addition to the row-wise top to bottom (2.16) and the column-wise left to right (2.17) ordering we have the row-wise bottom to top

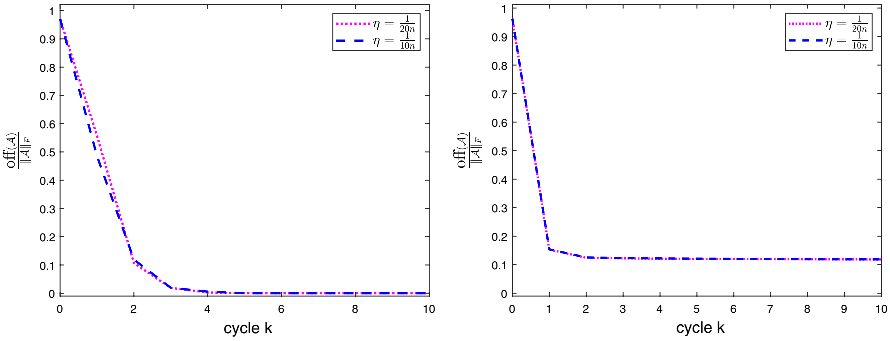

Fig. 1 Change in the relative off-norm for two 30 × 30 × 30 tensors with different values of η. Left: Diagonalizable tensor. Right: Non-diagonalizable tensor

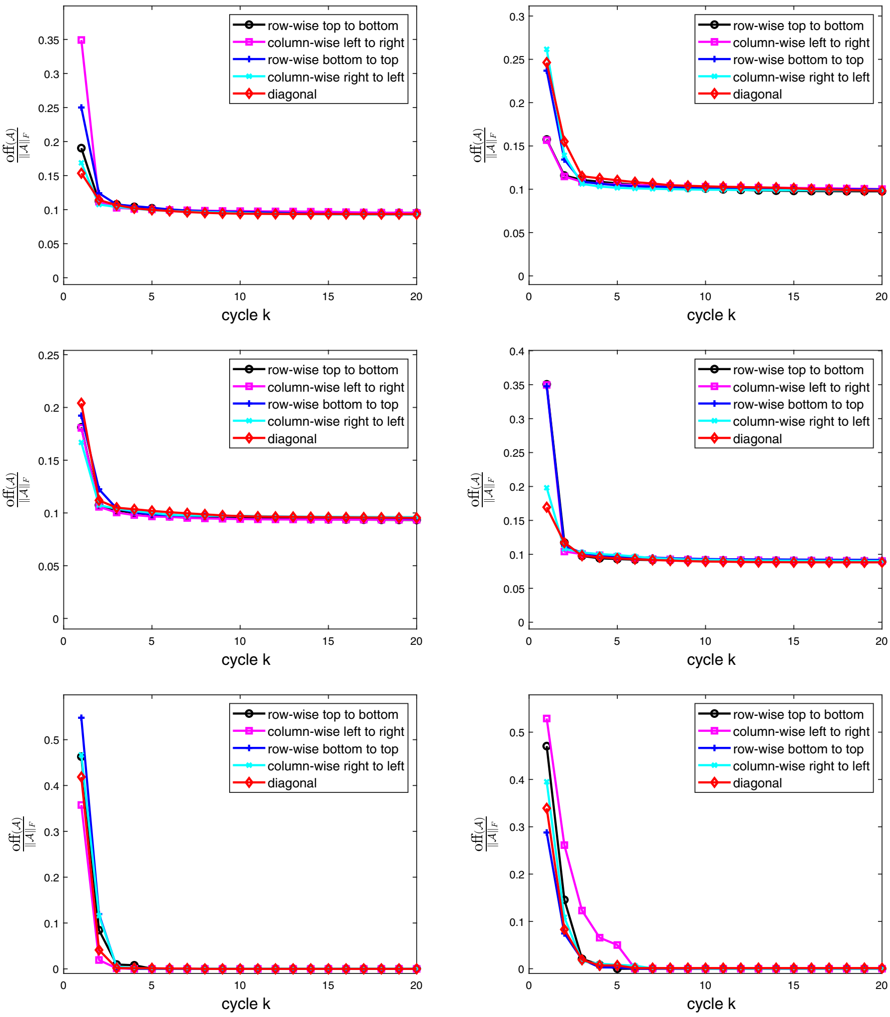

Fig. 2 Change in the relative off-norm for six $10 \times 10 \times 10$ tensors with different pivot strategies

$$\mathcal{O}'_r = (n-1,n),(n-2,n-1),(n-2,n),(n-3,n-2),\ldots$$
$$\ldots,(2,n),(1,2),\ldots,(1,n),$$

the column-wise right to left

$$\mathcal{O}'_c = (1,n),\ldots,(n-1,n),(1,n-1),\ldots,(n-2,n-1),\ldots,(1,2),$$

and the diagonal ordering of pivot pairs

$$\mathcal{O}_d = (1,2),(2,3),\ldots,(n-1,n),(1,3),(2,4),\ldots,(n-2,n),(1,4),\ldots,(1,n).$$

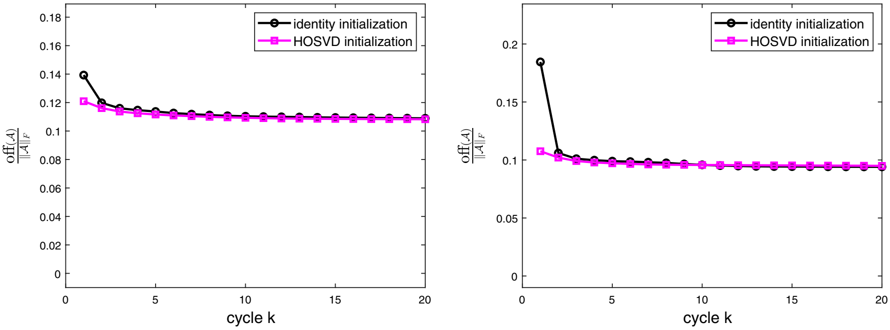

Fig. 3 Convergence of the algorithm with different initializations. Left: $20 \times 20 \times 20$ tensor. Right: $10 \times 10 \times 10$ tensor

We run the algorithm on six random tensors, four of which cannot be diagonalized by orthogonal transformations, all with $\eta = \frac{1}{20n}$. As expected, different pivot strategies are faster/slower on different tensors. However, no matter what pivot strategy we choose, the algorithm converges to the same point.

In Figure 3 we compare two different initializations for our Jacobi-type algorithm. The first one is identity initialization as it was done in Algorithm 2.1. There we have

$$\mathcal{A}^{(0)} = \mathcal{A}, \quad U_0 = V_0 = W_0 = I. \tag{4.2}$$

The other initialization that can be used is coming from the HOSVD of $\mathcal{A}$, see [5], $\mathcal{A} = \widetilde{\mathcal{S}} \times_1 \widetilde{U} \times_2 \widetilde{V} \times_3 \widetilde{W}$, where $\widetilde{U}$, $\widetilde{V}$, and $\widetilde{W}$ are matrices of left singular vectors of matricizations $A_{(1)}$, $A_{(2)}$, and $A_{(3)}$, respectively, and $\widetilde{\mathcal{S}} = \mathcal{A} \times_1 \widetilde{U}^T \times_2 \widetilde{V}^T \times_3 \widetilde{W}^T$. Then, instead of the initialization (4.2), we set $\mathcal{A}^{(0)} = \widetilde{\mathcal{S}}$, $U_0 = \widetilde{U}$, $V_0 = \widetilde{V}$, $W_0 = \widetilde{W}$. We run the algorithm with $\eta = \frac{1}{20n}$ on two random tensors. We can see that the HOSVD initialization is superior in the beginning cycles. This is the case because, compared to the starting tensor $\mathcal{A}$, the core tensor $\widetilde{\mathcal{S}}$ from the HOSVD of $\mathcal{A}$ is significantly closer to a diagonal tensor. Nevertheless, after those first cycles, both initializations are equally good.

5 Symmetric and antisymmetric tensors

We say that a tensor is symmetric if its elements remain constant under any permutation of indices. For a symmetric tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$ we have

$$x_{ijk} = x_{ikj} = x_{jik} = x_{jki} = x_{kij} = x_{kji}.$$

Symmetric tensors were studied in details in [13] and [12], where the authors also work with a Jacobi-type algorithm.

Our algorithm is not structure-preserving. In order to have a symmetry-preserving Jacobi-type algorithm, rotation matrices should be the same in all modes. Since the rotations $R_{U,k}$, $R_{V,k}$, and $R_{W,k}$ in the kth step are chosen depending on different tensors, they are not all the same. Nevertheless, we have noticed in practice that for smaller values of η from (2.18), after the convergence criterion is satisfied, the Algorithm 2.1, in most of the cases, returns mutually equal matrices U, V, W and a symmetric tensor $\mathcal{S}$. However, this is not the case for larger η. We will illustrate this behaviour on an example.

Departure from symmetry is measured as the distance in the Frobenius norm between the tensor $\mathcal{A}$ and its symmetrization sym($\mathcal{A}$),

$$\|\mathcal{A} - \mathrm{sym}(\mathcal{A})\|, \tag{5.1}$$

where, for a 3rd order tensor $\mathcal{X}$, we have

$$\mathrm{sym}(\mathcal{X}) = \frac{1}{6}(x_{ijk} + x_{ikj} + x_{jik} + x_{jki} + x_{kij} + x_{kji}).$$

It is easy to check that, if $\mathcal{X}$ is symmetric, then $\mathrm{sym}(\mathcal{X}) = \mathcal{X}$, and the expression (5.1) is equal to zero. We applied Algorithm 2.1 with $\eta = \frac{1}{2000n}$ on a randomly generated symmetric $20 \times 20 \times 20$ tensor $\mathcal{A}$. In the left picture in Figure 4 we can see that after we start with a symmetric tensor, symmetry is lost already in the first cycle, as expected, but the tensor becomes more and more symmetric through iterations and the sequence $(\mathcal{A}^{(k)})_k$ converges to a symmetric tensor. In the right picture we see that the distance between each pair of matrices U_k, V_k, W_k converges to zero, that is, U_k, V_k, W_k converge to the same matrix. This does not happen for $\eta = \frac{1}{20n}$.

One should keep in mind that for a symmetric starting tensor the solution of the maximization problem (2.5) is not necessary a symmetric tensor. One such example is tensor $\mathcal{T}$ from [12, Example 5.5] that can be given by its matricization

$$T_{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.2}$$

First, we notice that neither identity nor HOSVD initialization work on this tensor. In both cases the diagonal elements of $\mathcal{T}^{(0)}$ are zero and all rotation angles are zero, so the tensor is unchanged. Thus, here we do preconditioning with a random orthogonal matrix Q by setting $\mathcal{A}^{(0)} = \mathcal{A} \times_1 Q \times_2 Q \times_3 Q$. For the vast majority of the choices of Q, Algorithm 2.1 with $\eta = \frac{1}{2000n}$ converges to the symmetric tensor $\mathcal{S}$,

$$S_{(1)} = \begin{bmatrix} 0.8889 & -0.4444 & -0.4444 & -0.4444 & -0.4444 & -0.1111 & -0.4444 & -0.1111 & -0.4444 \\ -0.4444 & -0.4444 & -0.1111 & -0.4444 & 0.8889 & -0.4444 & -0.1111 & -0.4444 & -0.4444 \\ -0.4444 & -0.1111 & -0.4444 & -0.1111 & -0.4444 & -0.4444 & -0.4444 & -0.4444 & 0.8889 \end{bmatrix},$$

with transformation matrix $U = V = W$ depending on Q. This is a stationary point of the objective function (2.6), but not a point of its global maximum. In the other

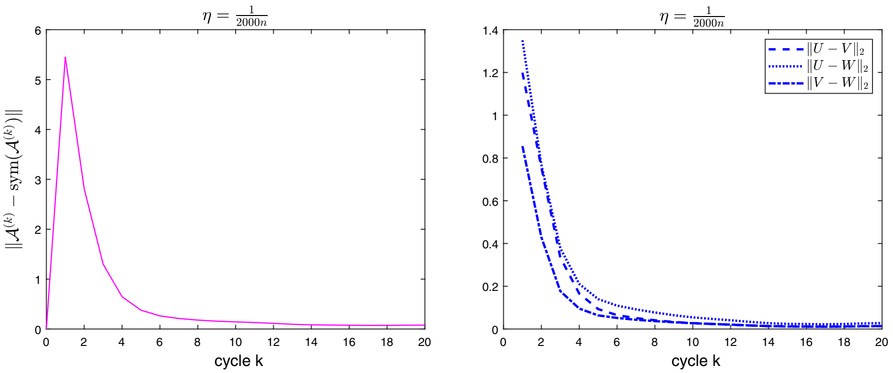

Fig. 4 Departure from the symmetry for a random symmetric 20 × 20 × 20 tensor

rare cases the algorithm converged to one of the better, but nonsymmetric, solutions of the form $\bar{S}$

$$\bar{S}_{(1)} = \begin{bmatrix} \pm 1 & 0 & 0 & 0 & 0 & 0 & 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 & 0 & \pm 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pm 1 & 0 & 0 & 0 & 0 & \pm 1 \end{bmatrix}.$$

On the other hand, a tensor is antisymmetric if its elements change sign when permuting pairs of indices. For an antisymmetric tensor $\mathcal{X} \in \mathbb{R}^{n\times n\times n}$ we have

$$x_{ijk} = x_{jki} = x_{kij} = -x_{ikj} = -x_{jik} = -x_{kji}.$$

In every antisymmetric tensor, elements on the positions where two or more indices are the same are equal to zero. Hence, all elements on the diagonal of an antisymmetric tensor are zero. This is the reason why, contrary to the symmetric case where one may be interested in a structure-preserving algorithm, we are not interested in preserving the antisymmetry. Here, by each iteration a tensor moves further from the structure. Still, antisymmetric tensors need some special attention. If we apply the algorithm directly in the form given in Algorithm 2.1, with the identity initialization, the algorithm will fail when computing the rotation angle. This happens because for an antisymmetric tensor, when computing the tangent of the double rotation angle (2.13), (2.14), and (2.15), we get both the numerator and the denominator equal to zero. We can overcome this problem with a preconditioning step — instead of the identity initialization (4.2) we use the HOSVD initialization as described in Section 4.

Acknowledgements This work has been supported in part by Croatian Science Foundation under the project UIP-2019-04-5200. The author is grateful to the anonymous referees for their detailed comments that improved the paper.

References

1. Kovač, E.B.: Finding the closest normal structured matrix. Linear Algebra Appl. **617**, 49–77 (2021)
2. Kovač, E.B., Kressner, D.: Structure-preserving low multilinear rank approximation of antisymmetric tensors. SIAM J. Matrix Anal. Appl. **38**(3), 967–983 (2017)
3. Comon, P.: Tensor diagonalization, a useful tool in signal processing. IFAC Proc. Vol. **27**(8), 77–82 (1994)
4. Comon, P., Jutten, C. (eds.): Handbook of Blind Source Separation. Academic Press, Oxford (2010)
5. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
6. De Lathauwer, L., De Moor, B., Vandewalle, J.: Independent component analysis and (simultaneous) third-order tensor diagonalization. IEEE Trans. Signal Process. **49**, 2262–2271 (2001)
7. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010)
8. Hackbusch, W.: Tensor Spaces and Numerical Tensor Calculus. Springer Series in Computational Mathematics 42 (2012)
9. Ishteva, M., Absil, P.-A., Van Dooren, P.: Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors. SIAM J. Matrix Anal. Appl. **34**(2), 651–672 (2013)
10. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
11. Landsberg, J.M.: Tensors: Geometry and Applications. American Mathematical Society, New York (2011)
12. Li, J., Usevich, K., Comon, P.: On approximate diagonalization of third order symmetric tensors by orthogonal transformations. Linear Algebra Appl. **576**(1), 324–351 (2019)
13. Li, J., Usevich, K., Comon, P.: Globally convergent Jacobi-type algorithms for simultaneous orthogonal symmetric tensor diagonalization. SIAM J. Matrix Anal. Appl. **39**(1), 1–22 (2018)
14. Moravitz Martin, C.D., Van Loan, C.F.: A Jacobi-type method for computing orthogonal tensor decompositions. SIAM J. Matrix Anal. Appl. **30**(3), 1219–11232 (2008)
15. Usevich, K., Li, J., Comon, P.: Approximate matrix and tensor diagonalization by unitary transformations: convergence of Jacobi-type algorithms. SIAM J. Optim. **30**(4), 2998–3028 (2020)
16. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Sci. Comput. **34**(2), A1027–A1052 (2012)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.