BIT Numerical Mathematics (2023) 63:48
https://doi.org/10.1007/s10543-023-00990-y

BIT

Paige's Algorithm for solving a class of tensor least squares problem

Xue-Feng Duan[1] · **Yong-Shen Zhang**[1] · **Qing-Wen Wang**[2] · **Chun-Mei Li**[1]

Received: 12 October 2022 / Accepted: 29 August 2023 / Published online: 20 September 2023

Abstract

In this paper, we consider a class of tensor least squares problem with an invertible linear transform, which arises in image restoration. Based on the operator-bidiagonal procedure, two Paige's algorithms are designed to solve it. The convergence theorems of the new methods are derived. Numerical experiments are performed to illustrate the feasibility and efficiency of the new methods, including when the algorithm is tested with the synthetic data and on some image restoration problems. Comparisons with some previous methods are also given.

Keywords Tensor least squares problem · Paige's Algorithm · Convergence analysis · Color image restoration

Mathematics Subject Classification 15A69 · 15B33 · 65F10

This work was supported by the National Natural Science Foundation of China (Grant No. 12361079; 12201149; 12261026; 12371023), the Natural Science Foundation of Guangxi Province (Grant No. 2023GXNSFAA026067), and the Innovation Project of GUET Graduate Education (Grant No. 2022YCXS147; 2022YCXS140).

✉ Xue-Feng Duan
guidian520@126.com

Yong-Shen Zhang
981725633@qq.com

Qing-Wen Wang
wqw@t.shu.edu.cn

Chun-Mei Li
lengyue123@126.com

[1] College of Mathematics and Computational Science, Center for Applied Mathematics of Guangxi (GUET), Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin 541004, Guangxi, People's Republic of China

[2] Department of Mathematics, Shanghai University, Shanghai 200444, Guangxi, People's Republic of China

1 Introduction

Throughout this paper, vectors are written in italic lowercase letters such as u, v, matrices correspond to uppercase letters, e.g., A, B, and tensors are denoted by calligraphic capital letters such as $\mathscr{A}$, $\mathscr{B}$. Let $\mathbb{R}^{I_1\times I_2\times I_3}$ denote the set of $I_1 \times I_2 \times I_3$ tensors over the real field $\mathbb{R}$. The zero tensor $\mathscr{O}$ is the one with all entries being zero. A slice is a two-dimensional section of a tensor, defined by fixing all but two indices. The kth frontal slice $\mathbf{X}_{::k}$ of the third-order tensor $\mathscr{X} \in \mathbb{R}^{I_1\times I_2\times I_3}$ is denoted as $\mathscr{X}^{(k)}$. The tensor Frobenius norm $||\mathscr{A}||_F = \sqrt{\langle\mathscr{A}, \mathscr{A}\rangle}$ is induced by the tensor inner product

$$\langle\mathscr{A}, \mathscr{B}\rangle = \sum_{i}^{I_1}\sum_{j}^{I_2}\sum_{k}^{I_3} a_{ijk}b_{ijk},$$

where $\mathscr{A} = (a_{ijk}) \in \mathbb{R}^{I_1\times I_2\times I_3}$, $\mathscr{B} = (b_{ijk}) \in \mathbb{R}^{I_1\times I_2\times I_3}$.

In this paper, we consider the following tensor least squares problem.

Problem 1.1 Given tensors $\mathscr{A} \in \mathbb{R}^{m\times l\times n}$, $\mathscr{B} \in \mathbb{R}^{p\times q\times n}$, and $\mathscr{C} \in \mathbb{R}^{m\times q\times n}$, find a tensor $\tilde{\mathscr{X}} \in \mathbb{R}^{l\times p\times n}$ such that

$$||\mathscr{A} *_L \tilde{\mathscr{X}} *_L \mathscr{B} - \mathscr{C}||_F^2 = \min_{\mathscr{X}\in\mathbb{R}^{l\times p\times n}} ||\mathscr{A} *_L \mathscr{X} *_L \mathscr{B} - \mathscr{C}||_F^2. \tag{1.1}$$

Here we set the linear operator $\mathscr{M} : \mathbb{R}^{l\times p\times n} \to \mathbb{R}^{m\times q\times n}$ be

$$\mathscr{M}(\mathscr{X}) = \mathscr{A} *_L \mathscr{X} *_L \mathscr{B}.$$

The operator $*_L$ in Problem 1.1 is a tensor-tensor product with an invertible linear transform, which was first proposed by Kernfeld, Kilmer and Aeron [27]. It is a generalization of the t-product [28] and the cosine transform product (*c product). The definitions and connections between the t-product and *c product are described in Section 2. The $*_L$ product is also referred to as the $\star_M$-product family in the literature [29]. This kind of tensor-tensor product has been applied to processing of seismic data [13], color image restoration [16, 18], facial recognition [21] and data compression [41].

Problem 1.1 often arises in the image restoration. As described in [16], the 3D model of color image restoration can be expressed as the tensor expression

$$\mathscr{B} = \mathscr{A} *_L \mathscr{X} + \mathscr{E}, \tag{1.2}$$

where $\mathscr{B}$, $\mathscr{A}$ and $\mathscr{E}$ are the observation image, the blurring operator and the noise, respectively. And $\mathscr{X}$ is the restored image to be sought. Obviously, Problem 1.1 is a generalized form of (1.2), which consider both horizontal and vertical blurring of the image. Compared with the previous image restoration models, the matrix product model [5]

$$g = Hx + n, \tag{1.3}$$

and the Einstein product model [10, 31, 40]

$$\mathscr{M} = \mathscr{T} *_3 \mathscr{Q} + \mathscr{N}, \tag{1.4}$$

the advantage of (1.2) is that it avoids the lack of information inherent in the expansion process of the tensor and takes into account the orderliness and correlation between the data.

The tensor equations and their least squares problems have been investigated by many scholars. Some mathematical tools, for example, the tensor generalized inverse, have been used in the literature [2, 3, 8, 11, 22, 37, 38]. The methods for solving the tensor equations and their least squares problems can be generally divided into two classes. The first class is the so-called direct methods, such as the elimination method [7] and the tensor generalized inverse methods [2, 3, 6, 8]. However, these methods do not perform well in large-scale data. The second class is the iterative methods. Wang, Xu and Duan [39] proposed the conjugate gradient method to solve the quaternion Sylvester tensor equation. Huang and Ma [23] and Xie et al. [40] solved the tensor least squares problem in Einstein product by conjugate gradient method and preprocessed conjugate gradient method, respectively. Ding and Wei [12] proposed the Newton method to solve the $m-1$ degree homogeneous tensor equation with the symmetric M-tensor coefficient. Guide and Ichi et al. [18] proposed the GMRES-type methods for solving the image processing problems with the t-product. Similar algorithms can also be found in the literature [9, 16, 17, 24, 39]. For the ill-posed problem, Reichel and Ugwu [34, 35] use the Golub-Kahan bidiagonalization and the Arnoldi upper Hessenberg preprocessors to solve the Tikhonov regularization problem in the image restoration model. Some other algorithms [19, 36] have been designed by involving matrixization of tensors, thus they are difficult to be used for the large-scale tensor least squares problem. However, the research results of Problem 1.1 are very few as far as we know.

In this paper, we consider the tensor least squares Problem 1.1 arising in image restoration. We first give the operator-bidiagonal procedures of $\mathscr{M}(\mathscr{X})$, and then use them to design two Paige's Algorithms for solving Problem 1.1. Two convergence theorems of the new methods are also given. A numerical example shows the feasibility of the Paige's methods for solving Problem 1.1. The simulation experiments of the image restoration illustrate the feasibility and effectiveness of the new methods. Specifically, the algorithms in this paper differ from the ones described in [34, 35] in the following two ways:

- Our proposed operator bidiagonalization process is distinguished from GG-tGKB process [35] because it provides a unified iterative framework for different linear operator bidiagonalization.
- We derive a coefficient relation between the minimal Frobenius norm solution of Problem 1.1 and the orthogonal tensor bases of the Krylov subspace

$$\mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{V}_1) = span\{\mathscr{M}^*(\mathscr{U}_1), (\mathscr{M}^*\mathscr{M})\mathscr{M}^*(\mathscr{U}_1), \cdots, (\mathscr{M}^*\mathscr{M})^{k-1}\mathscr{M}^*(\mathscr{U}_1)\},$$

which allows our algorithms to use only the tensor generated by the current kth iteration. We do not need additional memory for the orthogonal bases as described in [34, 35], and this approach reduces the space complexity of the algorithm.

This paper is organized as follows. In Sect. 2, we introduce some notations and definitions. And we also summarize the connection between $*_L$ product, t-product and $*_c$ product. In Sect. 3, we design the Paige's algorithms for solving Problem 1.1 and give the convergence theorems. In Sect. 4, some numerical experiments are given to illustrate that the algorithms are feasible and effective.

2 Preliminaries

In this section, we give some preliminaries, and then recall the $*_L$ product proposed by Kernfeld, Kilmer and Aeron [27] and analyze the connection between t-product, $*_c$ product, and $*_L$ product.

Given $\mathscr{A} \in \mathbb{R}^{m\times l\times n}$ and its frontal slice $\mathscr{A}^{(i)}$, $i = 1, 2, \cdots, n$, the operators bcirc unfold and fold can be defined as [28]

$$bcirc(\mathscr{A}) := \begin{bmatrix} \mathscr{A}^{(1)} & \mathscr{A}^{(n)} & \dots & \mathscr{A}^{(2)} \\ \mathscr{A}^{(2)} & \mathscr{A}^{(1)} & \dots & \mathscr{A}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathscr{A}^{(n)} & \mathscr{A}^{(n-1)} & \dots & \mathscr{A}^{(1)} \end{bmatrix}, unfold(\mathscr{A}) := \begin{bmatrix} \mathscr{A}^{(1)} \\ \mathscr{A}^{(2)} \\ \vdots \\ \mathscr{A}^{(n)} \end{bmatrix},$$

and $fold(unfold(\mathscr{A})) := \mathscr{A}$. Then the t-product can be defined as follows.

Definition 2.1 (t-product [28]) Let the t-product be the operator $*$: $\mathbb{R}^{m\times l\times n} \times \mathbb{R}^{l\times p\times n} \to \mathbb{R}^{m\times p\times n}$ between two tensors $\mathscr{A} \in \mathbb{R}^{m\times l\times n}$ and $\mathscr{B} \in \mathbb{R}^{l\times p\times n}$, which produces a tensor of $\mathbb{R}^{m\times p\times n}$ as follows

$$\mathscr{A} * \mathscr{B} = fold(bcirc(\mathscr{A})unfold(\mathscr{B})).$$

Notice that the block matrix $bcirc(\mathscr{A})$ can be block diagonalized by the discrete Fourier transform (DFT), i.e.,

$$A = blockdiag(\hat{\mathscr{A}}^{(1)}, \hat{\mathscr{A}}^{(2)}, \cdots \hat{\mathscr{A}}^{(n)}) = (F_n \otimes I_l)bcirc(\mathscr{A})(F_n^* \otimes I_m),$$

where $\otimes$ is the Kronecker product, F_n is the unitary DFT matrix satisfing

$$F_n F_n^* = F_n^* F_n = I_n.$$

Therefore, $\mathscr{A} * \mathscr{B}$ with the t-product can be efficiently implemented by the fast Fourier transform.

Considering the discrete cosine transform instead of the discrete Fourier transform, we obtain the $*_c$ product similar to the t-product. The block Toeplitz-plus-Hankel

matrix and the operator ten can be defined as [27]

$$mat(\mathscr{A}) := \begin{bmatrix} \mathscr{A}^{(1)} & \mathscr{A}^{(2)} & \dots & \mathscr{A}^{(n)} \\ \mathscr{A}^{(2)} & \mathscr{A}^{(1)} & \dots & \mathscr{A}^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathscr{A}^{(n)} & \mathscr{A}^{(n-1)} & \dots & \mathscr{A}^{(1)} \end{bmatrix} + \begin{bmatrix} \mathscr{A}^{(2)} & \dots & \mathscr{A}^{(n)} & 0 \\ \vdots & \iddots & \iddots & \mathscr{A}^{(3)} \\ \mathscr{A}^{(n)} & 0 & \iddots & \vdots \\ 0 & \mathscr{A}^{(n)} & \dots & \mathscr{A}^{(2)} \end{bmatrix},$$

$$ten(mat(\mathscr{A})) = fold(Tmat(\mathscr{A})E_l) = \mathscr{A},$$

where $E_l = \begin{bmatrix} I_l \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{nl \times l}$ is the first l columns of the identity matrix I_{nl} and

$$T = \begin{bmatrix} I_m & -I_m & \cdots & (-I_m)^{n-1} \\ 0 & I_m & \cdots & (-I_m)^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & I_m \end{bmatrix} \in \mathbb{R}^{nm \times nm}.$$

Definition 2.2 ($*_c$ product [27]) Set $\mathscr{A} \in \mathbb{R}^{m \times l \times n}$ and $\mathscr{B} \in \mathbb{R}^{l \times p \times n}$ be two real tensors. Then the $*_c$ product $\mathscr{A} *_c \mathscr{B}$ is an $m \times p \times n$ real tensor defined by

$$\mathscr{A} *_c \mathscr{B} = ten(mat(\mathscr{A})mat(\mathscr{B})).$$

Both the t-product and the $*_c$ product use linear transformations that block diagonalize the block matrix. If we consider each diagonal block as each frontal slice of the tensor, we can redefine the t-product and $*_c$ product using the n-mode product.

Definition 2.3 ($*_L$ product [27]) Let L be an invertible linear operator (see Definition 2.4), and set $\mathscr{A} \in \mathbb{R}^{m \times l \times n}$ and $\mathscr{B} \in \mathbb{R}^{l \times p \times n}$. Then the $*_L$ product $\mathscr{A} *_L \mathscr{B}$ is an $m \times p \times n$ tensor defined by

$$\mathscr{A} *_L \mathscr{B} = L^{-1}(L(\mathscr{A}) \triangle L(\mathscr{B})),$$

where the face-wise product $\mathscr{A} \triangle \mathscr{B}$ is defined by

$$(\mathscr{A} \triangle \mathscr{B})^{(i)} = \mathscr{A}^{(i)}\mathscr{B}^{(i)}.$$

Definition 2.4 ([30]) Let $M \in \mathbb{R}^{n \times n}$ be an invertible matrix. Then the invertible linear operator $L: \mathbb{R}^{m \times l \times n} \to \mathbb{R}^{m \times l \times n}$ is defined as

$$L(\mathscr{A}) = \mathscr{A} \times_3 M,$$

where $\mathscr{A} \in \mathbb{R}^{m \times l \times n}$.

As described by Kernfeld, Kilmer and Aeron [27], the $*_c$ product is an example of the $*_L$ product, where $M = W_c^{-1} C_n(I + Z)$ in the invertible linear operator L. Here C_n denote the $n \times n$ orthogonal DCT matrix, $W_c = diag(C_n(:, 1))$ and Z is the $n \times n$ (singular) circulant upshift matrix. The connection between the t-product and the $*_L$ product is

$$M = W_t^{-1} F_n,$$

where $W_t = diag(F_n(:, 1))$.

Using the invertible linear operator L, we can give the definition of a tensor transpose which satisfies the t-product and $*_c$ product.

Definition 2.5 ([27]) Set $\mathscr{A} \in \mathbb{R}^{m\times l\times n}$, then transpose $\mathscr{A}^T \in \mathbb{R}^{l\times m\times n}$ satisfies

$$L(\mathscr{A}^T)^{(i)} = (L(\mathscr{A})^{(i)})^T,\ i = 1, \cdots, n.$$

We also need the following definitions for the paper.

Definition 2.6 ([18]) Let $\mathbb{V} = [\mathscr{V}_1, \mathscr{V}_2, \cdots, \mathscr{V}_k] \in \mathbb{R}^{m\times kp\times n}$, where $\mathscr{V}_i \in \mathbb{R}^{m\times p\times n}$, $i = 1, 2, \cdots, k$ and $y = [y_1, y_2, \cdots, y_k]^T \in \mathbb{R}^k$, $Z = [z_1, z_2, \cdots, z_k] \in \mathbb{R}^{k\times k}$. Then the product $\mathbb{V} \circledast y$ and $\mathbb{V} \circledast Z$ are defined as

$$\mathbb{V} \circledast y = \sum_{i=1}^{k} y_i \mathscr{V}_i,\ \mathbb{V} \circledast Z = [\mathscr{V}_1 \circledast z_1, \mathscr{V}_2 \circledast z_2, \cdots, \mathscr{V}_k \circledast z_k].$$

Definition 2.7 ([18]) Let $\mathbb{A} = [\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_k] \in \mathbb{R}^{m\times kl\times n}$ and $\mathbb{B} = [\mathscr{B}_1, \mathscr{B}_2, \cdots, \mathscr{B}_p] \in \mathbb{R}^{m\times pl\times n}$, where $\mathscr{A}_i, \mathscr{B}_i \in \mathbb{R}^{m\times l\times n}$, $i = 1, 2, \cdots$. Then the product $\mathbb{A}^T \Diamond \mathbb{B}$ is the $k \times p$ matrix defined by

$$(\mathbb{A}^T \Diamond \mathbb{B})_{ij} = \left\langle \mathscr{A}_i, \mathscr{B}_j \right\rangle.$$

Notice that if $\mathbb{B} = \mathscr{B} \in \mathbb{R}^{m\times l\times n}$, then the product $\mathbb{A}^T \Diamond \mathscr{B}$ yields a column vector.

Definition 2.8 ([26]) Let $\mathscr{M}$: $\mathbb{R}^{l\times p\times n} \to \mathbb{R}^{l\times q\times n}$ be the linear operator. If the linear operator $\mathscr{M}^*$: $\mathbb{R}^{l\times q\times n} \to \mathbb{R}^{l\times p\times n}$ satisfies

$$\langle \mathscr{M}(\mathscr{X}), \mathscr{Y} \rangle = \left\langle \mathscr{X}, \mathscr{M}^*(\mathscr{Y}) \right\rangle,\ \forall \mathscr{X} \in \mathbb{R}^{l\times p\times n},\ \mathscr{Y} \in \mathbb{R}^{l\times q\times n},$$

then it is called the adjoint of $\mathscr{M}$.

3 Paige's Algorithms for solving problem 1.1

In this section, we first give the operator-bidiagonal procedure of $\mathscr{M}(\mathscr{X})$, and then use it to design the Paige's Algorithms for solving Problem 1.1. The convergence of the new methods are also given.

3.1 The operator-bidiagonal procedure

This subsection gives the operator-bidiagonal procedure. Firstly, the adjoint operator $\mathscr{M}^*: \mathbb{R}^{m\times q\times n} \to \mathbb{R}^{l\times p\times n}$ of the linear operator $\mathscr{M} : \mathbb{R}^{l\times p\times n} \to \mathbb{R}^{m\times q\times n}$ in Problem 1.1 is derived.

Lemma 3.1 *Set $\mathscr{M}(\mathscr{X}) = \mathscr{A} *_L \mathscr{X} *_L \mathscr{B}$. The adjoint of the linear operator $\mathscr{M}$ is*

$$\mathscr{M}^*(\mathscr{X}) = \mathscr{A}^T *_L \mathscr{X} *_L \mathscr{B}^T.$$

Proof Here we first prove that

$$\langle \mathscr{A} *_L \mathscr{X}, \mathscr{Y} \rangle = \left\langle \mathscr{X}, \mathscr{A}^T *_L \mathscr{Y} \right\rangle, \ \forall \mathscr{X} \in \mathbb{R}^{l\times p\times n}, \ \mathscr{Y} \in \mathbb{R}^{m\times p\times n}.$$

Set $M = (m_1, m_2, \cdots, m_n)$, $M^{-1} = (\hat{m}_1, \hat{m}_2, \cdots, \hat{m}_n)$, $\mathscr{A} = (a_{ijk}) \in \mathbb{R}^{m\times l\times n}$, $\mathscr{X} = (x_{ijk}) \in \mathbb{R}^{l\times p\times n}$, $\mathscr{Y} = (y_{ijk}) \in \mathbb{R}^{m\times p\times n}$, and set

$$a_{ij}^{(i)} := \mathbf{a}_{ij:}^T m_i, \ i = 1, 2, \cdots, n.$$

Combining Definition 2.4, we have

$$L(\mathscr{A})^{(i)} = (\mathscr{A} \times_3 M)^{(i)} = \begin{bmatrix} a_{11}^{(i)} & a_{12}^{(i)} & \cdots & a_{1l}^{(i)} \\ a_{21}^{(i)} & a_{22}^{(i)} & \cdots & a_{2l}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}^{(i)} & a_{m2}^{(i)} & \cdots & a_{ml}^{(i)} \end{bmatrix}.$$

Similarly, we have

$$L(\mathscr{X})^{(i)} = (\mathscr{X} \times_3 M)^{(i)} = \begin{bmatrix} x_{11}^{(i)} & x_{12}^{(i)} & \cdots & x_{1p}^{(i)} \\ x_{21}^{(i)} & x_{22}^{(i)} & \cdots & x_{2p}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{l1}^{(i)} & x_{l2}^{(i)} & \cdots & x_{lp}^{(i)} \end{bmatrix},$$

$$L(\mathscr{Y})^{(i)} = (\mathscr{Y} \times_3 M)^{(i)} = \begin{bmatrix} y_{11}^{(i)} & y_{12}^{(i)} & \cdots & y_{1p}^{(i)} \\ y_{21}^{(i)} & y_{22}^{(i)} & \cdots & y_{2p}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1}^{(i)} & y_{m2}^{(i)} & \cdots & y_{mp}^{(i)} \end{bmatrix},$$

where $x_{ij}^{(i)} := \mathbf{x}_{ij:}^T m_i$, $y_{ij}^{(i)} := \mathbf{y}_{ij:}^T m_i$, $i = 1, 2, \cdots, n$. Then the expression for any element in $\mathscr{A} *_L \mathscr{X}$ can be given as

$$(\mathscr{A} *_L \mathscr{X})_{ijk} = \left(\sum_{t=1}^{l} a_{it}^{(1)} x_{tj}^{(1)} \ \sum_{t=1}^{l} a_{it}^{(2)} x_{tj}^{(2)} \ \cdots \ \sum_{t=1}^{l} a_{it}^{(n)} x_{tj}^{(n)} \right) \hat{m}_i.$$

Noting that

$$\langle \mathscr{A} *_L \mathscr{X}, \mathscr{Y} \rangle = \sum_{i=1}^{n} \left\langle (\mathscr{A} *_L \mathscr{X})^{(i)}, \mathscr{Y}^{(i)} \right\rangle,$$

and by making use of the matrix inner product of each frontal slice of the tensor $\mathscr{A} *_L \mathscr{X}$ and tensor $\mathscr{Y}$, we get

$$\begin{aligned}
&\sum_{i=1}^{n} ((\mathscr{A} *_L \mathscr{X})^{(i)T} \mathscr{Y}^{(i)})_{rr} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{t=1}^{l} a_{jt}^{(1)} x_{tr}^{(1)} \sum_{t=1}^{l} a_{jt}^{(2)} x_{tr}^{(2)} \cdots \sum_{t=1}^{l} a_{jt}^{(n)} x_{tr}^{(n)} \right) \hat{m}_i y_{jr} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{t=1}^{l} a_{jt}^{(1)} x_{tr}^{(1)} y_{jr} \sum_{t=1}^{l} a_{jt}^{(2)} x_{tr}^{(2)} y_{jr} \cdots \sum_{t=1}^{l} a_{jt}^{(n)} x_{tr}^{(n)} y_{jr} \right) \hat{m}_i \\
&= \sum_{i=1}^{n} \sum_{t=1}^{l} \left(\sum_{j=1}^{m} a_{jt}^{(1)} y_{jr}^{(1)} x_{tr} \sum_{j=1}^{m} a_{jt}^{(2)} y_{jr}^{(2)} x_{tr} \cdots \sum_{j=1}^{m} a_{jt}^{(n)} y_{jr}^{(n)} x_{tr} \right) \hat{m}_i \\
&= \sum_{i=1}^{n} \sum_{t=1}^{l} \left(\sum_{j=1}^{m} a_{jt}^{(1)} y_{jr}^{(1)} \sum_{j=1}^{m} a_{jt}^{(2)} y_{jr}^{(2)} \cdots \sum_{j=1}^{m} a_{jt}^{(n)} y_{jr}^{(n)} \right) \hat{m}_i x_{tr} \\
&= \sum_{i=1}^{n} (\mathscr{X}^{(i)T} (\mathscr{A}^T *_L \mathscr{Y})^{(i)})_{rr}, \ r = 1, 2, \cdots, p.
\end{aligned}$$

This implies that

$$\langle \mathscr{A} *_L \mathscr{X}, \mathscr{Y} \rangle = \left\langle \mathscr{X}, \mathscr{A}^T *_L \mathscr{Y} \right\rangle. \tag{3.1}$$

In a similar way, we have

$$\langle \mathscr{X} *_L \mathscr{B}, \mathscr{Y} \rangle = \left\langle \mathscr{X}, \mathscr{Y} *_L \mathscr{B}^T \right\rangle. \tag{3.2}$$

Combining (3.1) and (3.2), we can obtain that

$$\langle \mathscr{A} *_L \mathscr{X} *_L \mathscr{B}, \mathscr{Y} \rangle = \left\langle \mathscr{X}, \mathscr{A}^T *_L \mathscr{Y} *_L \mathscr{B}^T \right\rangle,$$

which means that

$$\mathscr{M}^*(\mathscr{X}) = \mathscr{A}^T *_L \mathscr{X} *_L \mathscr{B}^T.$$

The proof is completed. □

Combining Definition 2.8, we simplify the operator $\mathscr{M}$ in Problem 1.1 using the operator-bidiagonal procedure, which is similar to the Golub-Kahan bidiagonalization technique [14] as follows.

Algorithm 1: The lower operator-bidiagonal procedure of $\mathscr{M}$

Set the initial tensor $\mathscr{U}_1 \in \mathbb{R}^{m\times q\times n}$, which satisfies $||\mathscr{U}_1||_F = 1$;
Calculate $\alpha_1\mathscr{V}_1 = \mathscr{M}^*(\mathscr{U}_1)$ and let $k = 1$;
for $k = 1, 2, \cdots, m$ **do**
  $\beta_{k+1}\mathscr{U}_{k+1} = \mathscr{M}(\mathscr{V}_k) - \alpha_k\mathscr{U}_k$;
  $\alpha_{k+1}\mathscr{V}_{k+1} = \mathscr{M}^*(\mathscr{U}_{k+1}) - \beta_{k+1}\mathscr{V}_k$;
end

In Algorithm 1, the scalars α_k and β_k are chosen so that $||\mathscr{U}_k||_F = ||\mathscr{V}_k||_F = 1$. It is not difficult to find that after m iterations of Algorithm 1, we have the following results

$$\begin{aligned}\mathscr{V}_k &\in span\{\mathscr{M}^*(\mathscr{U}_1), (\mathscr{M}^*\mathscr{M})\mathscr{M}^*(\mathscr{U}_1), \cdots, (\mathscr{M}^*\mathscr{M})^{k-1}\mathscr{M}^*(\mathscr{U}_1)\}\\ &\triangleq \mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{M}^*(\mathscr{U}_1)) = \mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{V}_1),\\ \mathscr{U}_k &\in span\{\mathscr{U}_1, (\mathscr{M}^*\mathscr{M})(\mathscr{U}_1), \cdots, (\mathscr{M}^*\mathscr{M})^{k-1}(\mathscr{U}_1)\} \triangleq \mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{U}_1).\end{aligned}$$

By using the definitions

$$\begin{aligned}&\mathbb{V}_k := [\mathscr{V}_1, \mathscr{V}_2, \cdots, \mathscr{V}_k] \in \mathbb{R}^{l\times pk\times n},\ \mathbb{U}_k := [\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k] \in \mathbb{R}^{m\times qk\times n},\\ &L_k = \begin{bmatrix}\alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k\end{bmatrix},\ \tilde{L}_k = \begin{bmatrix}\alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1}\end{bmatrix},\\ &\mathscr{M}^*(\mathbb{U}_k) := [\mathscr{M}^*(\mathscr{U}_1), \mathscr{M}^*(\mathscr{U}_2), \cdots, \mathscr{M}^*(\mathscr{U}_k)] \in \mathbb{R}^{l\times pk\times n},\\ &\mathscr{M}(\mathbb{V}_k) := [\mathscr{M}(\mathscr{V}_1), \mathscr{M}(\mathscr{V}_2), \cdots, \mathscr{M}(\mathscr{V}_k)] \in \mathbb{R}^{m\times qk\times n},\end{aligned}$$

the recurrence relations of Algorithm 1 can be rewritten as

$$\mathscr{M}^*(\mathbb{U}_k) = \mathbb{V}_k \circledast L_k^T,\ \mathscr{M}(\mathbb{V}_k) = \mathbb{U}_{k+1} \circledast \tilde{L}_k. \tag{3.3}$$

Theorem 3.1 *The tensors sequences $\{\mathscr{V}_k\}$ and $\{\mathscr{U}_k\}$ generated by Algorithm 1 are the orthonormal basis of the Krylov space $\mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{V}_1)$ and $\mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{U}_1)$, i.e.*

$$\mathbb{U}_k^T \Diamond \mathbb{U}_k = \mathbb{V}_k^T \Diamond \mathbb{V}_k = I_k. \tag{3.4}$$

Proof We prove this theorem by the mathematical induction. When $m = 1$, we have

$$\begin{aligned}\langle \mathscr{U}_1, \mathscr{U}_2\rangle &= \langle \mathscr{U}_1, (\mathscr{M}(\mathscr{V}_1) - \alpha_1\mathscr{U}_1)/\beta_2\rangle\\ &= \frac{\langle \mathscr{M}^*(\mathscr{U}_1), \mathscr{V}_1\rangle - \alpha_1\langle \mathscr{U}_1, \mathscr{U}_1\rangle}{\beta_2}\\ &= \frac{\langle \alpha_1\mathscr{V}_1, \mathscr{V}_1\rangle - \alpha_1\langle \mathscr{U}_1, \mathscr{U}_1\rangle}{\beta_2} = 0.\end{aligned}$$

Assume that the result is true for $m = k$, then for $k + 1$, we have

$$\begin{aligned}
\langle \mathscr{U}_i, \mathscr{U}_{k+2} \rangle &= \langle \mathscr{U}_i, (\mathscr{M}(\mathscr{V}_{k+1}) - \alpha_{k+1}\mathscr{U}_{k+1})/\beta_{k+2} \rangle \\
&= \frac{\langle \mathscr{U}_i, \mathscr{M}(\mathscr{V}_{k+1}) \rangle - \alpha_{k+1} \langle \mathscr{U}_i, \mathscr{U}_{k+1} \rangle}{\beta_{k+2}} \\
&= \frac{\langle \mathscr{M}^*(\mathscr{U}_i), \mathscr{V}_{k+1} \rangle - \alpha_{k+1} \langle \mathscr{U}_i, \mathscr{U}_{k+1} \rangle}{\beta_{k+2}} \\
&= \frac{\langle \alpha_i \mathscr{V}_i + \beta_i \mathscr{V}_{i-1}, \mathscr{V}_{k+1} \rangle - \alpha_{k+1} \langle \mathscr{U}_i, \mathscr{U}_{k+1} \rangle}{\beta_{k+2}} \\
&= \begin{cases} \dfrac{\langle \alpha_i \mathscr{V}_i + \beta_i \mathscr{V}_{i-1}, \mathscr{V}_{k+1} \rangle - \alpha_{k+1} \langle \mathscr{U}_i, \mathscr{U}_{k+1} \rangle}{\beta_{k+2}}, & i = 1, 2, \cdots, k \\ \dfrac{\langle \alpha_{k+1} \mathscr{V}_{k+1} + \beta_{k+1} \mathscr{V}_k, \mathscr{V}_{k+1} \rangle - \alpha_{k+1} \langle \mathscr{U}_{k+1}, \mathscr{U}_{k+1} \rangle}{\beta_{k+2}}, & i = k+1 \end{cases} \\
&= 0.
\end{aligned}$$

Similarly, we also have

$$\begin{aligned}
\langle \mathscr{V}_i, \mathscr{V}_{k+1} \rangle &= \left\langle \mathscr{V}_i, (\mathscr{M}^*(\mathscr{U}_{k+1}) - \beta_{k+1}\mathscr{V}_k)/\alpha_{k+1} \right\rangle \\
&= \frac{\langle \mathscr{V}_i, \mathscr{M}^*(\mathscr{U}_{k+1}) \rangle - \beta_{k+1} \langle \mathscr{V}_i, \mathscr{V}_k \rangle}{\alpha_{k+1}} \\
&= \frac{\langle \mathscr{M}(\mathscr{V}_i), \mathscr{U}_{k+1} \rangle - \beta_{k+1} \langle \mathscr{V}_i, \mathscr{V}_k \rangle}{\alpha_{k+1}} \\
&= \frac{\langle \alpha_i \mathscr{U}_i + \beta_{i+1} \mathscr{U}_{i+1}, \mathscr{U}_{k+1} \rangle - \beta_{k+1} \langle \mathscr{V}_i, \mathscr{V}_k \rangle}{\alpha_{k+1}} \\
&= \begin{cases} \dfrac{\langle \alpha_i \mathscr{U}_i + \beta_{i+1} \mathscr{U}_{i+1}, \mathscr{U}_{k+1} \rangle - \beta_{k+1} \langle \mathscr{V}_i, \mathscr{V}_k \rangle}{\alpha_{k+1}}, & i = 1, 2, \cdots, k-1 \\ \dfrac{\langle \alpha_k \mathscr{U}_k + \beta_{k+1} \mathscr{U}_{k+1}, \mathscr{U}_{k+1} \rangle - \beta_{k+1} \langle \mathscr{V}_k, \mathscr{V}_k \rangle}{\alpha_{k+1}}, & i = k \end{cases} \\
&= 0.
\end{aligned}$$

It is easy to obtain that $||\mathscr{V}_{k+1}||_F = ||\mathscr{U}_{k+1}||_F = 1$, which means

$$\mathbb{U}_k^T \Diamond \mathbb{U}_k = \mathbb{V}_k^T \Diamond \mathbb{V}_k = I_k.$$

The proof is completed. □

In a similar way we can get the upper operator-bidiagonal procedure of the linear operator $\mathscr{T}: \mathbb{R}^{m \times q \times n} \to \mathbb{R}^{l \times p \times n}$

$$\mathscr{T}(\mathscr{X}) = \hat{\mathscr{A}} *_L \mathscr{X} *_L \hat{\mathscr{B}},$$

where $\hat{\mathscr{A}} \in \mathbb{R}^{l \times m \times n}$ and $\hat{\mathscr{B}} \in \mathbb{R}^{q \times p \times n}$, which can be seen in the following Algorithm 2.

Algorithm 2: The upper operator-bidiagonal procedure of $\mathscr{T}$

Set the initial tensor $\mathscr{V}_1 \in \mathbb{R}^{l\times q\times n}$, which satisfies $||\mathscr{V}_1||_F = 1$;
Calculate $\rho_1\mathscr{P}_1 = \mathscr{T}^*(\mathscr{V}_1)$ and let $k = 1$;
for $k = 1, 2, \cdots, m$ **do**
  $\theta_{k+1}\mathscr{V}_{k+1} = \mathscr{T}(\mathscr{P}_k) - \rho_k\mathscr{V}_k$;
  $\rho_{k+1}\mathscr{P}_{k+1} = \mathscr{T}^*(\mathscr{V}_{k+1}) - \theta_{k+1}\mathscr{P}_k$;
end

Similar to Algorithm 1, assume that

$$\mathbb{V}_k := [\mathscr{V}_1, \mathscr{V}_2, \cdots, \mathscr{V}_k] \in \mathbb{R}^{l\times pk\times n},\ \mathbb{P}_k := [\mathscr{P}_1, \mathscr{P}_2, \cdots, \mathscr{P}_k] \in \mathbb{R}^{m\times qk\times n},$$

$$B_k = \begin{bmatrix} \rho_1 & \theta_2 & & \\ & \rho_2 & \ddots & \\ & & \ddots & \theta_k \\ & & & \rho_k \end{bmatrix}, \tilde{B}_k = \begin{bmatrix} \rho_1 & \theta_2 & & \\ & \rho_2 & \ddots & \\ & & \ddots & \theta_k \\ & & & \rho_k \ \theta_{k+1} \end{bmatrix},$$

the recurrence relations of Algorithm 2 can be rewritten as

$$\mathscr{T}^*(\mathbb{V}_k) = \mathbb{P}_k \circledast B_k^T,\ \mathscr{T}(\mathbb{P}_k) = \mathbb{V}_{k+1} \circledast \tilde{B}_k. \tag{3.5}$$

Theorem 3.2 *The tensors sequences* $\{\mathscr{V}_k\}$ *and* $\{\mathscr{P}_k\}$ *generated by Algorithm 2 are the orthonormal basis of the Krylov space* $\mathscr{K}_k(\mathscr{T}^*\mathscr{T}, \mathscr{V}_1)$ *and* $\mathscr{K}_k(\mathscr{T}^*\mathscr{T}, \mathscr{P}_1)$*, i.e.*

$$\mathbb{P}_k^T \Diamond \mathbb{P}_k = \mathbb{V}_k^T \Diamond \mathbb{V}_k = I_k.$$

Proof It is similar with Theorem 3.1 and is omitted here. □

Comparing (3.3) with (3.5), it can be found that the lower operator-bidiagonal procedure of the adjoint operator $\mathscr{M}^*$ is equivalent to upper operator-bidiagonal procedure of the linear operator $\mathscr{M}$.

3.2 New algorithms for Problem 1.1 based on the operator-bidiagonal procedure

In this subsection, we present two Paige's Algorithms (denoted as Paige1-TTP and Paige2-TTP) based on Algorithms 1 and 2.

Set the initial tensor $\mathscr{X}_0 = 0$ and $\beta_1\mathscr{U}_1 = \mathscr{C}$. Suppose $||\mathscr{R}||_F^2 = \min_{\mathscr{X}} ||\mathscr{M}(\mathscr{X}) - \mathscr{C}||_F^2$ and $\hat{\mathscr{X}}$ is the solution of Problem 1.1. Then Problem 1.1 is equivalent to

$$\mathscr{R} + \mathscr{M}(\mathscr{X}) = \mathscr{C} = \beta_1\mathscr{U}_1,$$

and its optimality condition is

$$\mathscr{M}^*(\mathscr{M}(\hat{\mathscr{X}}) - \mathscr{C}) = \mathscr{O},$$

i.e.,

$$\mathscr{M}^*(\mathscr{R}) = \mathscr{O}.$$

By constructing the approximate solution $\mathscr{X}_k$ of Problem 1.1 as

$$\mathscr{X}_k = \mathbb{V}_k \circledast y_k,$$

we have

$$\mathscr{R}_k + \mathscr{M}(\mathbb{V}_k \circledast y_k) = \beta_1 \mathscr{U}_1,$$

where $y_k = (\eta_1, \eta_2, \cdots, \eta_k)^T$. Combining (3.3), we can deduce that

$$\mathscr{R}_k + (\mathbb{U}_{k+1} \circledast \tilde{L}_k) \circledast y_k = \beta_1 \mathscr{U}_1. \tag{3.6}$$

Simplifying (3.6) by (3.4), we obtain that

$$\tilde{L}_k y_k = \beta_1 e_1 - \mathbb{U}_{k+1}^T \lozenge \mathscr{R}_k. \tag{3.7}$$

Clearly, there are two cases of Problem 1.1. In the first case, the equations $\mathscr{M}(\mathscr{X}) = \mathscr{C}$ are consistent, i.e., $\mathscr{R} = \mathscr{O}$. In the second case, $\mathscr{R} \neq \mathscr{O}$, the equations are inconsistent.

We first consider the Case 1 with $\mathscr{R} = \mathscr{O}$. According to (3.7), we have

$$\tilde{L}_k y_k = \beta_1 e_1.$$

Thus y can be obtained by

$$\eta_1 = \frac{\beta_1}{\alpha_1}, \ \eta_{k+1} = -\frac{\beta_{k+1}}{\alpha_{k+1}} \eta_k.$$

The case with $\mathscr{R} \neq \mathscr{O}$, since $\mathscr{R}$ is unknown a priori, the ith element η_i of y obviously cannot be found at the same time as $\mathscr{U}_i$ and $\mathscr{V}_i$ are produced. We suppose that

$$\gamma \equiv \langle \mathscr{U}_1, \mathscr{R} \rangle, \ t \equiv (\tau_1, \tau_2, \cdots, \tau_{k-1})^T, \ \begin{pmatrix} 1 \\ t \\ \tau_k \end{pmatrix} \equiv \frac{\mathbb{U}_{k+1}^T \lozenge \mathscr{R}}{\gamma}.$$

Together with (3.3) and (3.4), we have

$$\tilde{L}_k (\mathbb{U}_{k+1}^T \lozenge \mathscr{R}) = (\mathbb{U}_{k+1} \tilde{L}_k)^T \lozenge \mathscr{R} = \mathscr{M}(\mathbb{V}_k)^T \lozenge \mathscr{R} = {\mathbb{V}_k}^T \lozenge (\mathscr{M}^*(\mathscr{R})) = 0.$$

Set

$$\bar{L}_k = \begin{bmatrix} \beta_2 & & & \\ \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \\ & & \alpha_k & \beta_{k+1} \end{bmatrix},$$

it follows on dividing by γ that

$$\bar{L}_k \begin{pmatrix} t \\ \tau_k \end{pmatrix} = -\alpha_1 e_1.$$

Therefore, t can be solved by

$$\tau_k = -\frac{\alpha_k}{\beta_{k+1}} \tau_{k-1}, \ k = 1, 2, \cdots$$

where $\tau_0 = 1$. Then assume that $z_k = (\xi_1, \xi_2, \cdots, \xi_k)^T$ and $w_k = (\omega_1, \omega_2, \cdots, \omega_k)^T$ such that

$$L_k z_k = \beta_1 e_1, \ L_k w_k = \begin{pmatrix} 1 \\ t \end{pmatrix}.$$

By $y_k = z_k - \gamma w_k$, we have

$$\beta_{k+1}\eta_k = \beta_{k+1}(\xi_k - \gamma\omega_k) = -\gamma\tau_k,$$

i.e.,

$$\gamma = \beta_{k+1} \frac{\xi_k}{\beta_{k+1}\omega - \tau_k}.$$

Then, $\mathscr{X}_k$ in Algorithm 3 can be redescribed as

$$\mathscr{X}_k = \mathbb{V}_k \circledast y_k = \mathbb{V}_k \circledast z_k - \gamma \mathbb{V}_k \circledast w_k.$$

It is easy to find that the residual after the kth step is

$$\mathscr{R}_k = \beta_{k+1}\gamma\omega_k \mathscr{U}_{k+1} - \beta_{k+1}\xi_k + \gamma \sum_{i=1}^{k} \tau_{i-1} \mathscr{U}_i.$$

In addition to judging that $\mathscr{X}_k$ tends to be stable, combining Algorithm 1 and Algorithm 3, we can easily find that α_i, β_i, ξ_i, and ω_i become negligible. And γ will also tend to be stable, so we can also use $||\gamma_k - \gamma_{k-1}|| \leq \varepsilon$ or $|\xi_i| \leq \varepsilon$ as the stopping criterion. It also shows that Algorithm 3 will converge to the least squares solution.

Now we will give the convergence theorem for Algorithm 3.

Theorem 3.3 *The sequence $\{\mathscr{X}_k\}$ generated by Algorithm 3 converges to the minimum Frobenius norm solution $\mathscr{X}^*$ of Problem 1.1 in a finite number of steps.*

Proof Suppose that the sequence $\{\mathscr{X}_k\}$ generated by Algorithm 3 converges to $\mathscr{X}^*$, which satisfies

$$\mathscr{X}^* = \mathbb{V}_k \circledast y_k \in \mathscr{K}_k(\mathscr{M}^*\mathscr{M}, \mathscr{V}_1), \ \mathscr{M}^*(\mathscr{R}^*) = \mathscr{O},$$

Algorithm 3: Paige1-TTP method for solving Problem 1.1

Input: The linear operator $\mathscr{M} : \mathbb{R}^{l\times p\times n} \to \mathbb{R}^{l\times q\times n}$, tensor $\mathscr{C} \in \mathbb{R}^{l\times q\times n}$ and tolerable error ε.
Output: The minimum Frobenius norm least squares solution $\mathscr{X}_k$ of Problem 1.1

Set $\tau_0 = 1;\ \xi_0 = -1;\ \omega_0 = 0;\ \mathscr{Z}_0 = \mathscr{O};\ \mathscr{W}_0 = \mathscr{O};\ \mathscr{X}_0 = \mathscr{O};$
Calculate $\beta_1\mathscr{U}_1 = \mathscr{C}$ and $\alpha_1\mathscr{V}_1 = \mathscr{M}^*(\mathscr{U}_1)$, let $k = 1;$
for $k = 1, 2, \cdots, m$ **do**
  $\xi_k = -\xi_{k-1}\beta_k/\alpha_i;$
  $\mathscr{Z}_k = \mathscr{Z}_{k-1} + \xi_k\mathscr{V}_k;$
  $\omega_k = (\tau_{k-1} - \beta_k\omega_{k-1})/\alpha_k;$
  $\mathscr{W}_k = \mathscr{W}_{k-1} + \omega_k\mathscr{V}_k;$
  $\beta_{k+1}\mathscr{U}_{k+1} = \mathscr{M}(\mathscr{V}_k) - \alpha_k\mathscr{U}_k;$
  $\tau_k = -\tau_{k-1}\alpha_k/\beta_{k+1};$
  $\alpha_{k+1}\mathscr{V}_{k+1} = \mathscr{M}^*(\mathscr{U}_{k+1}) - \beta_{k+1}\mathscr{V}_k;$
  $\gamma_k = \beta_{k+1}\xi_k/(\beta_{k+1}\omega_k - \tau_k);$
  $\mathscr{X}_k = \mathscr{Z}_k - \gamma_k\mathscr{W}_k;$
  if $||\mathscr{X}_k - \mathscr{X}_{k-1}|| \le \varepsilon$ **then**
    Stop
  end
end

where $\mathscr{R}^* = \mathscr{M}(\mathscr{X}^*) - \mathscr{C}$. By introducing an auxiliary tensor $\mathscr{D} \in \mathbb{R}^{l\times q\times n}$ such that $\mathscr{X}^* = \mathscr{M}^*(\mathscr{D})$. Let $\tilde{\mathscr{X}}$ be any least squares solution of Problem 1.1 and $\mathscr{Y} = \tilde{\mathscr{X}} - \mathscr{X}^*$. We can obtain that

$$\begin{aligned}||\mathscr{R}^*||_F^2 &= ||\tilde{\mathscr{R}}||_F^2 = ||\mathscr{C} - \mathscr{M}(\tilde{\mathscr{X}})||_F^2 = ||\mathscr{C} - \mathscr{M}(\mathscr{X}^* + \mathscr{Y})||_F^2\\ &= ||\mathscr{R}^* - \mathscr{M}(\mathscr{Y})||_F^2 = ||\mathscr{R}^*||_F^2 - 2\left\langle\mathscr{R}^*, \mathscr{M}(\mathscr{Y})\right\rangle + ||\mathscr{M}(\mathscr{Y})||_F^2\\ &= ||\mathscr{R}^*||_F^2 - 2\left\langle\mathscr{M}^*(\mathscr{R}^*), \mathscr{Y}\right\rangle + ||\mathscr{M}(\mathscr{Y})||_F^2\\ &= ||\mathscr{R}^*||_F^2 + ||\mathscr{M}(\mathscr{Y})||_F^2,\end{aligned}$$

which implies that $\mathscr{M}(\mathscr{Y}) = \mathscr{O}$. Then we have

$$\begin{aligned}||\tilde{\mathscr{X}}||_F^2 &= ||\mathscr{X}^* + \mathscr{Y}||_F^2 = ||\mathscr{X}^*||_F^2 - 2\left\langle\mathscr{X}^*, \mathscr{Y}\right\rangle + ||\mathscr{Y}||_F^2\\ &= ||\mathscr{X}^*||_F^2 - 2\left\langle\mathscr{M}^*(\mathscr{D}), \mathscr{Y}\right\rangle + ||\mathscr{Y}||_F^2\\ &= ||\mathscr{X}^*||_F^2 - 2\left\langle\mathscr{D}, \mathscr{M}(\mathscr{Y})\right\rangle + ||\mathscr{Y}||_F^2\\ &= ||\mathscr{X}^*||_F^2 + ||\mathscr{Y}||_F^2 \ge ||\mathscr{X}^*||_F^2.\end{aligned}$$

The equality holds if and only if $\mathscr{Y} = \mathscr{O}$, which implies that $\mathscr{X}^*$ is the minimum Frobenius norm solution. □

Next, we will give the Paige2-TTP method. The upper operator-bidiagonal procedure of the operator $\mathscr{M}$ is equivalent to the lower operator-bidiagonal procedure of the operator $\mathscr{M}^*$ as mentioned above. Therefore, we have

$$\mathscr{M}(\mathbb{V}_k) = \mathbb{P}_k \circledast L_k^T,\ \mathscr{M}^*(\mathbb{P}_k) = \mathbb{V}_{k+1} \circledast \tilde{L}_k,\ \mathbb{P}_k^T \lozenge \mathbb{P}_k = \mathbb{V}_k^T \lozenge \mathbb{V}_k = I_k.$$

As can be seen in Algorithm 3, we construct the approximate solution as

$$\mathscr{X}_k = \mathbb{V}_k \circledast y_k.$$

For convenience, we choose $\theta_1 \mathscr{V}_1 = \mathscr{M}^*(\mathscr{C})$. Paige [32] points out that when $\theta_1 \mathscr{V}_1 = \mathscr{M}^*(\mathscr{C})$, the diagonalization process will stop when $\theta_{k+1}\mathscr{V}_{k+1} = \mathscr{O}$, i.e.

$$\mathscr{M}(\mathbb{V}_k) = \mathbb{P}_k \circledast L_k^T, \ \mathscr{M}^*(\mathbb{P}_k) = \mathbb{V}_k \circledast L_k.$$

Let $y_k = (\eta_1, \eta_2, \cdots, \eta_3)^T$. We are more interested in $\mathbb{V}_k \circledast y_k$ than in the end of the bidiagonalization, which completely solves for y_k and then determines X_k. We rederive the iterative relation

$$\mathscr{M}(\mathscr{X}) = \mathscr{M}(\mathbb{V}_k \circledast y_k) = (\mathbb{P}_k \circledast L_k^T) \circledast y_k = \mathbb{P}_k \circledast (L_k^T y_k) = \mathscr{C} - \mathscr{R},$$

which implies

$$L_k^T y_k = \mathbb{P}_k \circledast (\mathscr{C} - \mathscr{R}).$$

Notice that

$$\begin{aligned} L_k({\mathbb{P}_k}^T \lozenge(\mathscr{C} - \mathscr{R})) &= (\mathbb{P}_k \circledast L_k^T)^T \lozenge(\mathscr{C} - \mathscr{R}) = (\mathscr{M}(\mathbb{V}_k))^T \lozenge(\mathscr{C} - \mathscr{R}) \\ &= \mathbb{V}_k^T \lozenge(\mathscr{M}^*(\mathscr{C} - \mathscr{R})) = \theta_1 e_1. \end{aligned}$$

By assuming that $z = {\mathbb{P}_k}^T \lozenge(\mathscr{C} - \mathscr{R})$, we have

$$L_k^T y_k = z, \ L_k z = \theta_1 e_1. \tag{3.8}$$

Combining (3.8) we obtain that

$$\mathscr{X}_k = \mathbb{V}_k \circledast y_k = (\mathbb{V}_k \circledast L_k^{-T}) \circledast (L_k^T y_k) = \mathbb{W}_k \circledast z,$$

where $\mathbb{W}_k = \mathbb{V}_k \circledast L_k^{-T}$ and $\mathbb{W}_k = [\mathscr{W}_1, \mathscr{W}_2, \cdots, \mathscr{W}]$. We can determine $\mathscr{W}_i$ sequentially by

$$\mathbb{W}_k \circledast L_k^T = \mathbb{V}_k.$$

We give the Paige2-TTP Algorithm for solving Problem 1.1 as follows.

We also give the convergence theorem for Algorithm 4.

Theorem 3.4 *The sequence $\{\mathscr{X}_k\}$ generated by Algorithm 4 converges to the minimum Frobenius norm solution $\mathscr{X}^*$ of Problem 1.1 in a finite number of steps.*

Proof It is similar with Theorem 3.3 and is omitted here. □

Algorithm 4: Paige2-TTP method for solving Problem 1.1

Input: The linear operator $\mathscr{M} : \mathbb{R}^{l\times p\times n} \to \mathbb{R}^{l\times q\times n}$, tensor $\mathscr{C} \in \mathbb{R}^{l\times q\times n}$ and tolerable error ε.
Output: The minimum Frobenius norm least squares solution $\mathscr{X}_k$ of Problem 1.1

Calculate $\theta_1\mathscr{V}_1 = \mathscr{M}^*(\mathscr{C})$; $\rho_1\mathscr{P}_1 = \mathscr{M}(\mathscr{V}_1)$, let $k = 1$;
Calculate $\mathscr{W}_1 = \mathscr{V}_1/\rho_1$; $\eta_1 = \theta_1/\rho_1$; $\mathscr{X}_1 = \eta_1\mathscr{W}_1$;
for $k = 1, 2, \cdots, m$ **do**
  $\theta_{k+1}\mathscr{V}_{k+1} = \mathscr{M}^*(\mathscr{P}_k) - \rho_k\mathscr{V}_k$;
  $\rho_{k+1}\mathscr{P}_{k+1} = \mathscr{M}(\mathscr{V}_{k+1}) - \theta_{k+1}\mathscr{P}_k$;
  $\mathscr{W}_{k+1} = (\mathscr{V}_{k+1} - \theta_{k+1}\mathscr{W}_k)/\rho_{k+1}$;
  $\eta_{k+1} = -\eta_k\theta_{k+1}/\rho_{k+1}$;
  $\mathscr{X}_{k+1} = \mathscr{X}_k - \eta_{k+1}\mathscr{W}_{k+1}$;
  if $||\mathscr{X}_k - \mathscr{X}_{k-1}|| \le \varepsilon$ **then**
    Stop
  end
end

3.3 Computational complexity analysis

In this subsection, we provide the computational complexity of Algorithms 3 and 4. We note that the computational cost of the multiplication operations of $L(\mathscr{A}) = \mathscr{A} \times_3 M$ and $\mathscr{A} *_L \mathscr{B}$ are $O(mln^2)$ and $O(mln^2 + lpn^2 + mlpn)$, respectively. Thus the cost of computing $\mathscr{U}_k$ and $\mathscr{V}_k$ in Algorithm 1 is $O(mln^2 + lpn^2 + pqn^2 + \min\{(l + q)mpn, (m + p)lqn\})$. Similarly, the cost of computing $\mathscr{V}_k$ and $\mathscr{P}_k$ in Algorithm 2 is $O(mln^2 + lpn^2 + pqn^2 + \min\{(l + q)mpn, (m + p)lqn\})$.

The computational cost of Algorithm 3 involves mainly tensor multiplication and addition. The computational cost of each iteration of Algorithm 3 is $O(lpn)$, except that the cost of calculating $\mathscr{U}_k$ and $\mathscr{V}_k$ is the same as in Algorithm 1, which means that the computational complexity of Algorithm 3 is $O(mln^2 + lpn^2 + pqn^2 + \min\{(l + q)mpn, (m + p)lqn\})$. Analogously, the computational cost of Algorithm 4 is mainly posed by steps 4 and 5. Therefore, the computational cost for Algorithm 4 is $O(mln^2 + lpn^2 + pqn^2 + \min\{(l + q)mpn, (m + p)lqn\})$.

4 Numerical experiments

In this section we give numerical examples and simulation experiments of image restoration to illustrate the performance of Algorithms 3 and 4. All of the tests are implemented in MATLAB R2018b with the machine precision 10^{-16} on PC (Intel(R) Core(TM) i5-1155G7), where the CPU is 2.50 GHz and the memory is 16.0 GB. The implementations of the algorithms are based on the functions from the MATLAB Tensor Toolbox developed by Bader and Kolda [1].

Example 4.1 Consider Problem 1.1 with $m = 5, l = 4, n = 3, p = 4$, and $q = 5$, where the tensors $\mathscr{A}$ and $\mathscr{B}$ are constructed by using the MATLAB command $rand(I_1, I_2, I_3)$

as follow

$$\mathscr{A}(:,:,1) = \begin{pmatrix} 0.7663 & 0.9361 & 0.3733 & 0.1862 \\ 0.1746 & 0.8927 & 0.1163 & 0.2087 \\ 0.5031 & 0.4435 & 0.5251 & 0.5304 \\ 0.3827 & 0.7515 & 0.3411 & 0.0843 \\ 0.9772 & 0.9094 & 0.8494 & 0.2408 \end{pmatrix},$$

$$\mathscr{A}(:,:,2) = \begin{pmatrix} 0.6346 & 0.3135 & 0.4382 & 0.7165 \\ 0.1111 & 0.7621 & 0.1864 & 0.6203 \\ 0.6073 & 0.9545 & 0.4431 & 0.0898 \\ 0.8898 & 0.6459 & 0.4881 & 0.1118 \\ 0.4907 & 0.4853 & 0.8611 & 0.5756 \end{pmatrix},$$

$$\mathscr{A}(:,:,3) = \begin{pmatrix} 0.4617 & 0.4819 & 0.5076 & 0.1959 \\ 0.5454 & 0.1605 & 0.2380 & 0.9748 \\ 0.0380 & 0.7544 & 0.1927 & 0.5751 \\ 0.5793 & 0.7935 & 0.8087 & 0.5832 \\ 0.6290 & 0.2914 & 0.0667 & 0.9500 \end{pmatrix};$$

$$\mathscr{B}(:,:,1) = \begin{pmatrix} 0.4521 & 0.0431 & 0.0431 & 0.8928 & 0.6670 \\ 0.0268 & 0.5561 & 0.7263 & 0.7794 & 0.9781 \\ 0.4253 & 0.6113 & 0.7133 & 0.4949 & 0.1171 \\ 0.1058 & 0.7388 & 0.5038 & 0.7615 & 0.8693 \end{pmatrix},$$

$$\mathscr{B}(:,:,2) = \begin{pmatrix} 0.5679 & 0.0623 & 0.9191 & 0.4355 & 0.1427 \\ 0.1202 & 0.2781 & 0.4152 & 0.6354 & 0.2006 \\ 0.9703 & 0.4096 & 0.7115 & 0.9016 & 0.6209 \\ 0.8400 & 0.6268 & 0.9709 & 0.8716 & 0.7750 \end{pmatrix},$$

$$\mathscr{B}(:,:,3) = \begin{pmatrix} 0.2468 & 0.1708 & 0.0821 & 0.6053 & 0.6116 \\ 0.3695 & 0.7456 & 0.7815 & 0.3545 & 0.8432 \\ 0.3411 & 0.0980 & 0.8487 & 0.7591 & 0.4570 \\ 0.4564 & 0.8903 & 0.4509 & 0.5894 & 0.8828 \end{pmatrix}.$$

The tensor $\mathscr{C}$ of Problem 1.1 is generated by $\mathscr{C} = \mathscr{A} *_L \mathscr{X}^* *_L \mathscr{B}$, where $\mathscr{X}^*$ is the exact solution.

$$\mathscr{C}(:,:,1) = \begin{pmatrix} 2.0709 & 2.9842 & 3.7128 & 4.6192 & 4.2295 \\ 1.2905 & 2.2195 & 2.7104 & 3.5677 & 3.2243 \\ 2.0320 & 2.6432 & 3.3329 & 4.2888 & 3.7896 \\ 1.3801 & 2.6006 & 2.7593 & 3.9167 & 3.8628 \\ 2.7502 & 3.6893 & 4.8344 & 5.7619 & 5.1022 \end{pmatrix},$$

$$\mathscr{C}(:,:,2) = \begin{pmatrix} 1.9866 & 1.1023 & 2.5292 & 2.4022 & 1.4136 \\ 1.3878 & 0.6701 & 1.9098 & 1.7834 & 0.9858 \\ 1.6750 & 0.8739 & 2.2706 & 2.1246 & 1.1841 \\ 1.9704 & 0.7190 & 2.2637 & 1.9022 & 1.0239 \\ 2.2009 & 1.1848 & 2.8530 & 2.7460 & 1.5583 \end{pmatrix},$$

$$\mathscr{C}(:,:,3) = \begin{pmatrix} 1.5129 & 2.6267 & 2.5285 & 3.1925 & 3.8561 \\ 1.6347 & 2.4602 & 2.6092 & 3.0806 & 3.3725 \\ 1.2401 & 2.0938 & 1.9912 & 2.8409 & 3.2008 \\ 2.4940 & 3.4742 & 3.7211 & 4.8108 & 5.2201 \\ 1.6014 & 3.1166 & 3.0049 & 3.6039 & 4.2954 \end{pmatrix}.$$

Given a random invertible linear transformation matrix

$$M = \begin{pmatrix} -0.5560 & 0.1429 & -0.8188 \\ -0.6669 & -0.6646 & 0.3368 \\ -0.4961 & 0.7334 & 0.4648 \end{pmatrix},$$

we solve Problem 1.1 by using Algorithms 3 and 4 with $\varepsilon = 10^{-6}$. After 129 iterations, we get the solution

$$\mathscr{X}^* = \mathscr{X}^{129}_{Paige1} = \mathscr{X}^{129}_{Paige2},$$

where

$$\mathscr{X}^{129}_{Paige1}(:,:,1) = \mathscr{X}^{129}_{Paige2}(:,:,1) = \begin{pmatrix} 0.1000 & 0.1670 & 0.3653 & 0.5469 \\ 0.8787 & 0.6461 & 0.4024 & 0.9860 \\ 0.8915 & 0.1803 & 0.0898 & 0.2127 \\ 0.2140 & 0.1165 & 0.6838 & 0.9190 \end{pmatrix},$$

$$\mathscr{X}^{129}_{Paige1}(:,:,2) = \mathscr{X}^{129}_{Paige2}(:,:,2) = \begin{pmatrix} 0.6652 & 0.1063 & 0.7282 & 0.7956 \\ 0.7435 & 0.8838 & 0.2856 & 0.0735 \\ 0.3785 & 0.4451 & 0.5937 & 0.9927 \\ 0.4748 & 0.6682 & 0.3705 & 0.7760 \end{pmatrix},$$

$$\mathscr{X}^{129}_{Paige1}(:,:,3) = \mathscr{X}^{129}_{Paige2}(:,:,3) = \begin{pmatrix} 0.0074 & 0.8850 & 0.9972 & 0.6549 \\ 0.8889 & 0.4656 & 0.8698 & 0.4118 \\ 0.8641 & 0.3716 & 0.0001 & 0.4764 \\ 0.6976 & 0.0830 & 0.6800 & 0.3883 \end{pmatrix}.$$

The residual errors are

$$\|\mathscr{A} *_L \mathscr{X}^{129}_{Paige1} *_L \mathscr{B} - \mathscr{C}\|_F = 2.5535 \times 10^{-7}, \ \|\mathscr{A} *_L \mathscr{X}^{129}_{Paige2} *_L \mathscr{B} - \mathscr{C}\|_F = 2.0378 \times 10^{-7}.$$

The relative errors are

$$\frac{\|\mathscr{X}^{129}_{Paige1} - \mathscr{X}^*\|_F}{\|\mathscr{X}^*\|_F} = 1.6625 \times 10^{-7}, \ \frac{\|\mathscr{X}^{129}_{Paige2} - \mathscr{X}^*\|_F}{\|\mathscr{X}^*\|_F} = 1.3598 \times 10^{-7}.$$

We also give the convergence curves of Algorithms 3 and 4 in the figure 1.

Example 1 demonstrates the feasibility of Algorithms 3 and 4 in solving Problem 1.1. To further illustrate their effectiveness, we present two simulation experiments for color image restoration as follows.

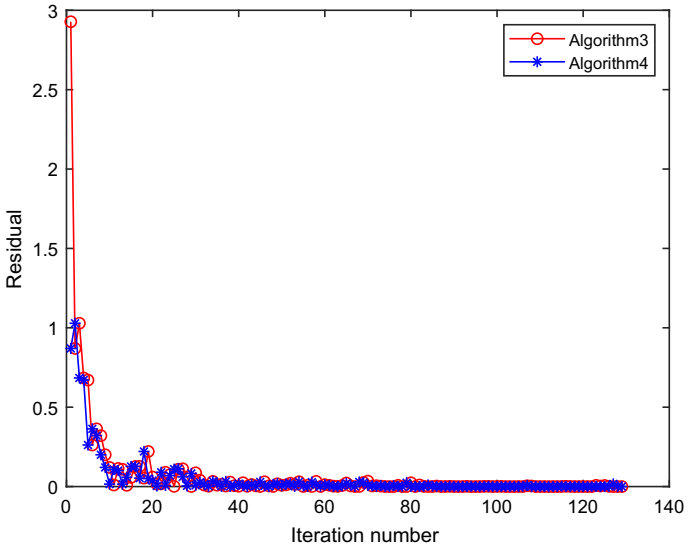


Fig. 1 Convergence curves of Algorithms 3 and 4

Colored 3D models for the image restoration can be modeled as tensor expressions [16]

$$\mathscr{B} = \mathscr{A} *_L \mathscr{X} + \mathscr{E},$$

where $\mathscr{B}$, $\mathscr{A}$ and $\mathscr{E} \in \mathbb{R}^{n\times n\times 3}$ are the observation image, the blurring operator and the noise, respectively. And $\mathscr{X}$ is the restored image to be sought. Each frontal slice of $\mathscr{A} \in \mathbb{R}^{n\times n\times 3}$ is a Toeplitz matrix obtained from a matrix $S \in \mathbb{R}^{n\times n}$, where S is a two-dimensional Gaussian function

$$S(i,j) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(i-j)^2}{2\sigma^2}), |i-j| \le r, \\ 0, \qquad otherwise, \end{cases} \tag{4.1}$$

and $\mathscr{A}$ is obtained from

$$\mathscr{A}^{(1)} = \alpha S,\ \mathscr{A}^{(2)} = \beta S,\ \mathscr{A}^{(3)} = \gamma S. \tag{4.2}$$

Here α, β and γ are the entries of the circular matrix

$$S_{color} = \begin{pmatrix} \alpha & \gamma & \beta \\ \beta & \alpha & \gamma \\ \gamma & \beta & \alpha \end{pmatrix},$$

obtained from [20], which satisfy $\alpha + \beta + \gamma = 1$. This matrix gives rise to cyclic mixing between different channels (RGB channels). We set the noise level

$$v := \frac{\|\mathscr{E}\|_F}{\|\mathscr{A} *_L \mathscr{X}\|_F}.$$

The noise $\mathscr{E}$ obeys a Gaussian distribution, which has a zero mean and v variance.

We use Algorithm 3, Algorithm 4, PGD algorithm [25], CG algorithm [33], NSPG algorithm [4] and GMRES algorithm [16] to solve the above model with $*_c$ product and t-product. The stopping criterion for all algorithms is either

$$\frac{\|\mathscr{X}_k - \mathscr{X}_{k-1}\|}{\|\mathscr{X}_{k-1}\|} \leq 10^{-4},$$

or the iteration step k reached the upper limit 1000. All color images in the experiments are from the USC-SIPI image database. In the experiments, we use CPU time, relative error (RE) and peak signal-to-noise ratio (PSNR) as the measurement for the restoration effect of the color image. The relative error (RE) and peak signal-to-noise ratio (PSNR) are defined as

$$RE = \frac{\|\mathscr{X} - \mathscr{Q}\|_F}{\|\mathscr{Q}\|_F}, \quad PSNR = 10\log_{10}\left(\frac{\mathscr{Q}_{\max}^2 I_1 I_2 I_3}{\|\mathscr{X} - \mathscr{Q}\|_F^2}\right),$$

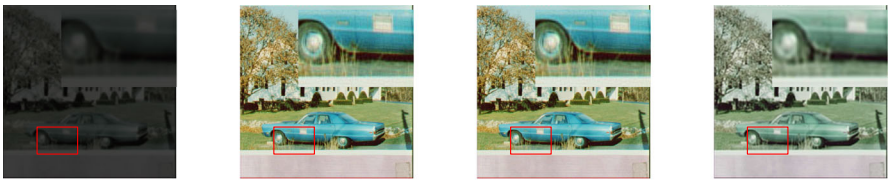
where $\mathscr{X}$ and $\mathscr{Q}$ are the restored image and the real image, respectively. The higher PSNR and the lower RE, the better restoration performance.

Example 4.2 We consider Problem 1.1 with $\mathscr{A}$ obtained from (4.1) and (4.2) by setting $\sigma = 2$, $r = 3$, $\alpha = 0.5$, $\beta = 0.1$ and $\gamma = 0.4$ as inside channel and cross-channel blur, respectively. Take the $House \in \mathbb{R}^{512\times512\times3}$ and $Airplane \in \mathbb{R}^{512\times512\times3}$ to be the true images. And we get a series of blurred images $\mathscr{C} = \mathscr{A} *_L \mathscr{X} *_L \mathscr{B} + \mathscr{E}$ by adding noise $\mathscr{E}$ with noise level $v = 10^{-4}$, where $\mathscr{B}$ is a third-order tensor with $\mathscr{B}^{(1)} = S^T$ and $\mathscr{B}^{(2)} = \mathscr{B}^{(3)} = 0$.

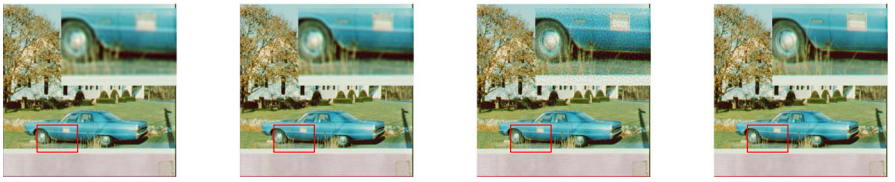
In the implementation of the algorithms for this example, we simultaneously considered the cases of t-product and $*_c$ product. We present visual representations of restored images obtained through various algorithms. Additionally, we magnify the local details in these images to compare the effectiveness of different restoration techniques on blurred images. We also provide numerical results for the CPU time, relative error (RE), and peak signal-to-noise ratio (PSNR) for each algorithm to compare their performance. All experimental results are listed in Figs. 2–3 and Tables 1–2.

Figures 2 and 3 show the visual effects of the restored images (House and Airplane). We can observe that the two images restored by Algorithms 3 and 4 have clearer texture details and features. This indicates that the results recovered by Algorithms 3 and 4 are of higher quality and more similar to the original images.

On the other hand, from Tables 1 and 2, we can see that our methods (Algorithms 3 and 4) obtain the highest PSNR value and the least RE value and take the least CPU time. The higher PSNR value and the lower RE value, the better recovery performance.

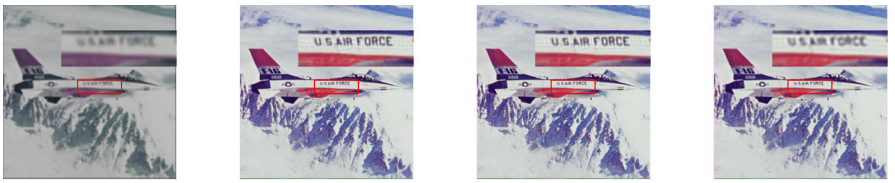


(a) Blurred image (b) Algorithm 3 (c) Algorithm 4 (d) PGD



(e) CG (f) NSPG (g) GMRES (h) Original image

Fig. 2 The visualization of the restoration image (House) for each algorithm. Experiments were performed under $*_c$ product



(a) Blurred image (b) Algorithm 3 (c) Algorithm 4 (d) PGD

(e) CG (f) NSPG (g) GMRES (h) Original image

Fig. 3 The visualization of the restoration image (Airplane) for each algorithm. Experiments were performed under t-product

So our methods outperform PGD, CG, NSPG and GMRES methods in PSNR and RE value.

The image restoration model is a typical ill-posed problem. When we use an algorithm to solve this problem, we often encounter the semi-convergence[1] phenomenon. Especially, when the residual of Problem 1.1 is less than $\|\mathscr{E}\|_F$, the tensor least squares problem will treat some of the noise as part of the image to be restored. The happening

[1] When we use an algorithm to solve the ill-posed problem, the algorithm exhibits a gradual convergence behavior for the first k iterations. However, after the kth step, the error will increase and the convergence will disappear. This is called "semi-convergence".

Table 1 The computational results for Algorithms 3, 4, PGD, CG, NSPG and GMRES when the stopping criterion is reached under $*_c$ product

Algorithms	House			Airplane		
	RE	$PSNR$	$Time/s$	RE	$PSNR$	$Time/s$
Algorithm 3	**0.0464**	**30.4615**	**16.3611**	**0.0302**	**32.3227**	19.1900
Algorithm 4	**0.0464**	**30.4615**	16.8903	**0.0302**	**32.3227**	**19.0411**
PGD	0.1166	22.4669	35.6320	0.0761	24.3182	27.9519
CG	0.0679	27.1624	42.1370	0.0444	29.0019	42.1587
NSPG	0.0573	28.6272	28.7995	0.0367	30.6470	39.0540
GMRES	0.0825	25.4646	165.5080	0.0726	24.7288	196.5575

The best RE, PSNR and CPU time results are highlighted in bold

Table 2 The computational results for Algorithms 3, 4, PGD, CG, NSPG and GMRES when the stopping criterion is reached under t-product

Algorithms	House			Airplane		
	RE	$PSNR$	$Time/s$	RE	$PSNR$	$Time/s$
Algorithm 3	**0.0465**	**30.4576**	**17.1224**	**0.0271**	**33.2824**	25.8698
Algorithm 4	**0.0465**	**30.4576**	17.4108	**0.0271**	**33.2824**	**25.6273**
PGD	0.0691	27.0057	24.7539	0.0449	28.8918	30.0570
CG	0.0487	30.0489	60.2831	0.0396	29.9860	45.6450
NSPG	0.0567	28.7228	38.0364	0.0367	30.6433	26.1994
GMRES	0.0528	29.3461	180.1900	0.0320	31.8283	156.7062

The best RE, PSNR and CPU time results are highlighted in bold

of overfitting leads to the semi-convergence phenomenon, which makes the restored image significantly different from the real image. This phenomenon is obvious in solving the image restoration model with a high noise level[2]. In this case, we add the Tikhonov regularization into Problem 1.1 to overcome this difficulty, i.e.,

$$\min_{\mathscr{X}\in\mathbb{R}^{n\times n\times 3}} \|\mathscr{A}*_L\mathscr{X}-\mathscr{B}\|_F^2+\mu\|\mathscr{X}\|_F^2. \tag{4.3}$$

Set the linear operator $\mathscr{M}:\mathbb{R}^{n\times n\times 3}\to\mathbb{R}^{2n\times n\times 3}$ be

$$\mathscr{M}(\mathscr{X})=\begin{bmatrix}\mathscr{A}\\ \mu^{1/2}\mathscr{J}\end{bmatrix}\mathscr{X},$$

and the tensor $\mathscr{C}\in\mathbb{R}^{2n\times n\times 3}$ be

$$\mathscr{C}=\begin{bmatrix}\mathscr{B}\\ \mathscr{O}\end{bmatrix}.$$

[2] Set $\mathscr{X}_k$ be the kth iterative value of an algorithm. If the noise $\mathscr{E}$ satisfies $\|\mathscr{E}\|_F\le\|\mathscr{A}*_L\mathscr{X}_k-\mathscr{B}\|_F$, it is called as a low noise level. Conversely, if $\mathscr{E}$ satisfies $\|\mathscr{E}\|_F>\|\mathscr{A}*_L\mathscr{X}_k-\mathscr{B}\|_F$, it is called as a high noise level.

(a) Blurred image (b) Algorithm 3 (c) Algorithm 4 (d) Original image

(a) Blurred image (b) Algorithm 3 (c) Algorithm 4 (d) Original image

Fig. 4 The visualization of the restoration images (Peppers and Sailboat). The first row of images is obtained under the t-product, and the second row of images is obtained under the $*_c$ product

Our proposed algorithms can also be used to solve Problem (4.3). In the upcoming experiments, we will address this problem by utilizing Algorithms 3 and 4. Additionally, we will compare our results not only with PGD, CG, NSPG, and GMRES algorithms but also with the recently proposed GG-tGKT [34, 35] and GG-LtAT [35] methods.

Example 4.3 This example studies the image restoration model with a higher noise level. We consider Problem (4.3) with $\mathscr{A}$ obtained from (4.1) and (4.2) by setting $\sigma = 3$, $r = 3$, $\alpha = 0.8$, $\beta = 0.1$ and $\gamma = 0.1$ as inside channel and cross-channel blur, respectively. Take the $Peppers \in \mathbb{R}^{512\times512\times3}$ and $Sailboat \in \mathbb{R}^{512\times512\times3}$ to be the true images. And we get a series of blurred images $\mathscr{B} = \mathscr{A} *_L \mathscr{X} + \mathscr{E}$ by adding noise $\mathscr{E}$ with noise level $v = 10^{-2}$. The obtained images are displayed on the left of Fig. 4. The optimal regularization parameter $\mu_{opt} = 5.1 \times 10^{-3}$ ($\mu_{opt} = 2.9 \times 10^{-3}$) under $*_c$ product (t-product) was obtained by using the generalized GCV method [15].

The visual results of two color images (Peppers and Sailboat) restored by Algorithms 3 and 4 are shown in Fig. 4. We can observe that the images restored by our algorithm are very close to the original images, with clear contour and texture features.

Further, the results of comparing Algorithms 3 and 4 with other algorithms are presented in Tables 3 and 4. These tables show that Algorithms 3 and 4 achieve the same PSNR and RE values as the PGD, CG, NSPG, GMRES, GG-tGKT and GG-LtAT algorithms. This suggests that all algorithms restore images to a comparable quality. However, our methods require the least CPU time compared to the other four algorithms. Therefore, the convergence rate of Algorithm 3 and 4 are faster than PGD, CG, NSPG, GMRES, GG-tGKT and GG-LtAT algorithms.

Table 3 The computational results for Algorithms 3, 4, PGD, CG, NSPG, GMRES, GG-tGKT and GG-LtAT when the stopping criterion is reached under $*_c$ product

	Peppers			Sailboat		
Algorithms	RE	$PSNR$	$Time/s$	RE	$PSNR$	$Time/s$
Algorithm 3	0.0858	26.6157	**2.8114**	0.0934	25.5349	2.8102
Algorithm 4	0.0858	26.6157	2.9385	0.0934	25.5349	**2.7089**
PGD	0.0847	26.7355	53.9421	0.0931	25.5632	51.7664
CG	0.0860	26.5994	19.7067	0.0940	25.4821	18.4134
NSPG	0.0861	26.5926	10.3421	0.0940	25.4789	13.5531
GMRES	0.0861	26.5905	8.5961	0.0941	25.4758	9.4450
GG-tGKT	0.0860	26.6001	7.3658	0.0939	25.4948	8.1918
GG-LtAT	0.0861	26.5837	9.0724	0.0940	25.4808	11.3202

The best CPU time result is highlighted in bold

Table 4 The computational results for Algorithms 3, 4, PGD, CG, NSPG, GMRES, GG-tGKT and GG-LtAT when the stopping criterion is reached under t-product

	Peppers			Sailboat		
Algorithms	RE	$PSNR$	$Time/s$	RE	$PSNR$	$Time/s$
Algorithm 3	0.0807	27.1563	2.5629	0.0922	25.6488	**2.0364**
Algorithm 4	0.0807	27.1563	**2.4029**	0.0922	25.6488	2.9412
PGD	0.0807	27.1526	26.1720	0.0923	25.6403	25.6363
CG	0.0807	27.1523	10.1876	0.0922	25.6471	9.9940
NSPG	0.0807	27.1516	10.2315	0.0922	25.6465	10.4668
GMRES	0.0807	27.1513	6.9074	0.0922	25.6468	5.3284
GG-tGKT	0.0926	25.9576	7.4188	0.0942	25.4608	5.8344
GG-LtAT	0.0843	26.7747	4.3153	0.0941	25.4720	4.7237

The best CPU time result is highlighted in bold

5 Conclusion

In this paper, we consider a class of tensor least squares problems under the tensor-tensor product with an invertible linear transforms, which arises in image restoration. Two Paige's Algorithms are proposed to solve this problem. The convergence theorems are also derived. Numerical experiments show that the new algorithms are feasible and effective for Problem 1.1. Two simulation experiments for the image restoration show the good performance of the new algorithms.

Acknowledgements The authors thank the editor and the reviewers for the constructive and helpful comments on the revision of this article.

Data Availability The data used to support the findings of this study are included within the article.

Declarations

Conflicts of interest This study does not have any conflicts to disclose.

References

1. Bader, B.W., Kolda, T.G., et al.: Tensor Toolbox for MATLAB, Version 3.2.1, https://www.tensortoolbox.org
2. Behera, R., Mishra, D.: Further results on generalized inverses of tensors via the Einstein product. Linear Multilinear Algebra **65**, 1662–1682 (2017)
3. Behera, R., Sahoo, J.K., Mohapatra, R.N., et al.: Computation of generalized inverses of tensors viat-product. Numer. Linear Algebra Appl. **29**, 1–23 (2022)
4. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM J. Optim. **10**, 1196–1211 (2000)
5. Bouhamidi, A., Jbilou, K., Raydan, M.: Convex constrained optimization for large-scale generalized Sylvester equations. Comput. Optim. Appl. **48**, 233–253 (2011)
6. Brazell, M., Li, N., Navasca, C., et al.: Solving multilinear systems via tensor inversion. SIAM J. Matrix Anal. Appl. **34**, 542–570 (2013)
7. Bykov, V.I., Kytmanov, A.M., Lazman, M.Z., et al.: Elimination Methods in Polynomial Computer Algebra. Kluwer Academic Publishers, The Netherlands (1998)
8. Cao, Z., Xie, P.: Perturbation analysis for t-product based tensor inverse, Moore-Penrose inverse and tensor system, arXiv preprint arXiv:2107.09544 (2021)
9. Chen, Z., Lu, L.Z.: A projection method and Kronecker product preconditioner for solving Sylvester tensor equations. Sci. China Math. **55**, 1281–1292 (2012)
10. Cui, L.B., Chen, C., Li, W., et al.: An eigenvalue problem for even order tensors with its applications. Linear Multilinear Algebra **64**, 602–621 (2016)
11. Dehdezi, E.K., Karimi, S.: A fast and efficient Newton-Shultz-type iterative method for computing inverse and Moore-Penrose inverse of tensors. J. Math. Model. **9**, 645–664 (2021)
12. Ding, W.Y., Wei, Y.M.: Solving multi-linear systems with M-tensors. J. Sci. Comput. **68**, 689–715 (2016)
13. Ely, G., Aeron, S., Hao, N., et al.: 5D and 4D pre-stack seismic data completion using tensor nuclear norm. TNN, SEG International Exposition and Eighty-Third Annual Meeting at Houston, TX (2013)
14. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. J. Soc. Industr. Appl. Math. **2**, 205–224 (1965)
15. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**, 215–223 (1979)
16. Guide, M.E., Ichi, A.E., Jbilou, K.: Discrete cosine transform LSQR and GMRES methods for multi-dimensional ill-posed problems, arXiv preprint arXiv:2103.11847 (2021)
17. Guide, M.E., Ichi, A.E., Jbilou, K.: Discrete cosine transform LSQR methods for multidimensional ill-posed problems. J. Math. Model. **10**, 21–37 (2022)
18. Guide, M.E., Ichi, A.E., Jbilou, K., et al.: Tensor Krylov subspace methods via the T-product for color image processing, arXiv preprint arXiv:2006.07133 (2020)
19. Guide, M.E., Ichi, A.E., Beik, F.P., et al.: Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing, arXiv preprint arXiv:2005.07458 (2020)
20. Hansen, P.C., Nagy, J.G., O'leary, D.P.: Deblurring Images: Matrices, Spectra, and Filtering. SIAM, Philadelphia (2006)
21. Hao, N., Kilmer, M.E., Braman, K., et al.: Facial recognition using tensor-tensor decompositions. SIAM J. Imaging Sci. **6**, 437–463 (2013)
22. Huang, B.: Numerical study on Moore-Penrose inverse of tensors via Einstein product. Numer. Algorithms **87**, 1767–1797 (2021)
23. Huang, B., Ma, C.: An iterative algorithm to solve the generalized Sylvester tensor equations. Linear Multilinear Algebra **68**, 1175–1200 (2020)

24. Huang, B., Ma, C.: Global least squares methods based on tensor form to solve a class of generalized Sylvester tensor equations. Appl. Math. Comput. **369**, 1–16 (2020)
25. Iusem, A.N.: On the convergence properties of the projected gradient method for convex optimization. Comput. Appl. Math. **22**, 37–52 (2003)
26. Karimi, S., Dehghan, M.: Global least squares method based on tensor form to solve linear systems in Kronecker format. T. I. Meas. Control **40**, 2378–2386 (2018)
27. Kernfeld, E., Kilmer, M., Aeron, S.: Tensor-tensor products with invertible linear transforms. Linear Algebra Appl. **485**, 545–570 (2015)
28. Kilmer, M.E., Martin, C.D.: Factorization strategies for third-order tensors. Linear Algebra Appl. **435**, 641–658 (2011)
29. Kilmer, M.E., Horesh, L., Avron, H., et al.: Tensor-tensor algebra for optimal representation and compression of multiway data. Proc. Natl. Acad. Sci. **118**, e2015851118 (2021)
30. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
31. Liu, X., Wang, L., Wang, J., et al.: A three-dimensional point spread function for phase retrieval and deconvolution. Opt. Express **20**, 15392–15405 (2012)
32. Paige, C.C.: Bidiagonalization of matrices and solution of linear equations. SIAM J. Numer. Anal. **11**, 197–209 (1974)
33. Polyak, B.T.: The conjugate gradient method in extremal problems. USSR Comp. Math. Math. Phys. **9**, 94–112 (1969)
34. Reichel, L., Ugwu, U.O.: The tensor Golub-Kahan-Tikhonov method applied to the solution of ill-posed problems with a t-product structure. Numer. Linear Algebra Appl. **29**, e2412 (2022)
35. Reichel, L., Ugwu, U.O.: Tensor Krylov subspace methods with an invertible linear transform product applied to image processing. Appl. Numer. Math. **166**, 186–207 (2021)
36. Savas, B., Eldén, L.: Krylov-type methods for tensor computations. Linear Algebra Appl. **438**, 891–918 (2013)
37. Sun, L., Zheng, B., Bu, C., et al.: Moore-Penrose inverse of tensors via Einstein product. Linear Multilinear Algebra **64**, 686–698 (2016)
38. Sun, L., Zheng, B., Wei, Y., et al.: Generalized inverses of tensors via a general product of tensors. Front. Math. China **13**, 893–911 (2018)
39. Wang, Q.W., Xu, X., Duan, X.: Least squares solution of the quaternion Sylvester tensor equation. Linear Multilinear Algebra **69**, 104–130 (2021)
40. Xie, Z.J., Jin, X.Q., Sin, V.K.: An optimal preconditioner for tensor equations involving Einstein product. Linear Multilinear Algebra **68**, 886–902 (2020)
41. Zhang, Z., Ely, G., Aeron, S., et al.: Novel methods for multilinear data completion and de-noising based on tensor-SVD. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3842–3849 (2014)