Article

Tensor Conjugate-Gradient Methods With Automatically Determination of Regularization Parameters for Ill-Posed Problems With T-product

Shi-Wei Wang , Guang-Xin Huang * , Feng Yin

Posted Date: 26 October 2023

doi: 10.20944/preprints202310.1696.v1

Keywords: linear discrete ill-posed problems; tensor Conjugate-Gradient method; t-product; discrepancy principle; Tikhonov regularization

Article

Tensor Conjugate-Gradient Methods with Automatically Determination of Regularization Parameters for Ill-Posed Problems with t-Product

Shi-Wei Wang [1], Guang-Xin Huang [2,*] and Feng Yin [1]

[1] Geomathematics Key Laboratory of Sichuan, College of Mathematics and Physics, Chengdu University of Technology, Chengdu 610059, China
[2] College of Computer Science and Cyber Security, Chengdu University of Technology, Chengdu 610059, China
[*] Correspondence: huangx@cdut.edu.cn

Abstract: This paper presents three types of tensor Conjugate-Gradient methods for solving large-scale linear discrete ill-posed problems based on the t-product between third-order tensors. An automatical determination strategy of a suitable regularization parameter is proposed for the tensor conjugate gradient (tCG) method. A truncated version and a preprocessed verion of the tCG method are further presented. The discrepancy principle is employed to determine a suitable regularization parameter. Several numerical examples are given to show the effectiveness of the proposed tCG methods in image and video restoration.

Keywords: linear discrete ill-posed problems; tensor Conjugate-Gradient method; t-product; discrepancy principle; Tikhonov regularization

1. Introduction

Tensors are high-dimensional arrays that have many applications in science and engineering, including in image, video and signal processing, computer vision, and network analysis [11,12,16–20,26]. A new t-product based on third-order tensors proposed by Kilmer et al [1,2]. When using high-dimensional data, t-product shows a greater potential value than matricization, see [2,6,11,12,21,22,24,25,27]. The t-product has been found to have special value in many application fields, including image deblurring problems [1,6,11,12], image and video compression [26], facial recognition problems [2], etc.

In this paper, we consider the solution of large minimization problems of the form

$$\min_{\vec{\mathcal{X}}\in\mathbb{R}^{m\times1\times n}} \|\mathcal{A}*\vec{\mathcal{X}}-\vec{\mathcal{B}}\|_F, \mathcal{A}=[a]_{i,j,k=1}^{l,m,n}\in\mathbb{R}^{l\times m\times n}, \vec{\mathcal{B}}\in\mathbb{R}^{l\times1\times n}. \tag{1}$$

The Frobenius norm of singular tube of $\mathcal{A}$ rapidly attenuates to zero with the increase of the index number. In particular, $\mathcal{A}$ has ill-determined tubal rank. Many of its singular tubes are nonvanishing with tiny Frobenius norm of different orders of magnitude. Problems (1) with such a tensor is called the tensor discrete linear ill-posed problems. They arise from the restoration of color image and video, see e.g., [1,11,12]. Throughout this paper, the operation $*$ represents tensor t-product and $\|\cdot\|_F$ denotes the tensor Frobenius norm or the spectral matrix norm.

We assume that the observed tensor $\vec{\mathcal{B}}\in\mathbb{R}^{m\times1\times n}$ is polluted by an error tensor $\vec{\mathcal{E}}\in\mathbb{R}^{m\times1\times n}$, i.e.,

$$\vec{\mathcal{B}}=\vec{\mathcal{B}}_{true}+\vec{\mathcal{E}}, \tag{2}$$

where $\vec{\mathcal{B}}_{true}\in\mathbb{R}^{m\times1\times n}$ is an unknown and unavailable error-free tensor related to $\vec{\mathcal{B}}$. $\vec{\mathcal{B}}_{true}$ is determined by $\mathcal{A}*\vec{\mathcal{X}}=\vec{\mathcal{B}}_{true}$, where $\vec{\mathcal{X}}_{true}$ represents the explicit solution of problems (1) that is to be found. We assume that the upper bound of the Frobenius norm of $\vec{\mathcal{E}}$ is known, i.e,

$$\|\vec{\mathcal{E}}\|_F \leq \delta. \tag{3}$$

Straightforward solution of (1) is usually meanless to get an approximation of $\vec{\mathcal{B}}_{true}$ because of the illposeness of $\mathcal{A} = [a]_{i,j,k=1}^{l,m,n}$ and the error $\vec{\mathcal{E}}$ will be amplified severely. We use Tikhonov regularization to reduce this effect in this paper and replace (1) with penalty least-squares problems

$$\min_{\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}} \left\{ \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F^2 + \mu \|\vec{\mathcal{X}}\|_F^2 \right\}, \tag{4}$$

where μ is a regularization parameter. We assume that

$$\mathcal{N}(\mathcal{A}) \cap \mathcal{N}(\mathcal{I}) = \vec{\mathcal{O}}, \tag{5}$$

where $\mathcal{N}(\mathcal{A})$ denotes the null space of $\mathcal{A}$, $\mathcal{I}$ is the identity tensor and $\vec{\mathcal{O}} \in \mathbb{R}^{m\times 1\times n}$ is a lateral slice whose elements are all zero. The normal equation of minimization problem (4) is

$$(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}} = \mathcal{A}^T * \vec{\mathcal{B}}, \tag{6}$$

then

$$\vec{\mathcal{X}}_\mu = \left(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}\right)^{-1} * \mathcal{A}^T * \vec{\mathcal{B}} \tag{7}$$

is the unique solution of the Tikhonov minimization problem (4) under the assumption (5).

There are many techniques to determine the regularization parameter μ, such as the L-curve criterion, generalized cross validation (GCV), and the discrepancy principle. We refer to [4,5,8–10] for more details. In this paper, the discrepancy principle is extended to tensors based on t-product and is employed to determine a suitable μ in (4). The solution $\vec{\mathcal{X}}_\mu$ of (4) satisfies

$$\|\mathcal{A} * \vec{\mathcal{X}}_\mu - \vec{\mathcal{B}}\|_F \leq \eta\delta, \tag{8}$$

where $\eta > 1$ is usually a user-specified constant and is independent of δ in (3). When $\|\vec{\mathcal{E}}\|_F$ is smaller enough, and δ approaches 0, result in $\vec{\mathcal{X}}_\mu \to \vec{\mathcal{X}}_{true}$. For more details on the discrepancy principle, see e.g., [7].

In this paper, we also consider the expansion of minimization problem (1) of the form

$$\min_{\mathcal{X} \in \mathbb{R}^{m\times p\times n}} \left\{ \|\mathcal{A} * \mathcal{X} - \mathcal{B}\|_F^2 + \mu \|\mathcal{X}\|_F^2 \right\}, \tag{9}$$

where $\mathcal{B} \in \mathbb{R}^{m\times p\times n}$, $p > 1$.

There are many methods for solving large-scale discrete linear ill-posed problems (1). Recently, a tensor Golub- Kahan bidiagonalization method [11] and a GMRES method [12] were introduced for solving large-scale linear ill-posed problems (4). The randomized tensor singular value decomposition (rt-SVD) method in [3] was presented for computing super large data sets, and has prospects in image data compression and analysis. Ugwu and Reichel [23] proposed a new random tensor singular value decomposition (R-tSVD), which improves the truncated tensor singular value decomposition (T-tSVD) in [1]. Kilmer et al. [2] presented a tensor Conjugate-Gradient (t-CG) method for tensor linear systems $\mathcal{A} * \vec{\mathcal{X}} = \vec{\mathcal{B}}$ corresponding to the least-squares problems. The regularization parameter in the t-CG method is user-specified. In this paper, we further discuss the automatical determinization of suitable regularization parameters of the tCG method by the discrepancy principle. The proposed method is called the tCG method with automatical determination of regularization parameters (auto-tCG). We also present a truncated auto-tCG method (auto-ttCG) to improve the auto-tCG method by reducing the computation. At last, a preprocessed version of the auto-ttCG method is proposed, which is abbreviated as auto-ttpCG.

The rest of this paper is organized as follows. Section 2 introduces some symbols and preliminary knowledge that will be used in the context. Section 3 presents the auto-tCG, auto-ttCG and auto-ttpCG methods for solving the minimization problems (4) and (9). Section 4 gives several examples on image and video restoration and Section 5 draws some conclusions.

2. Preliminaries

This section gives some notations and definitions, and briefly summarizes some results that will be used later. For a third-order tensor $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, Figure 1 shows the frontal slices $\mathcal{A}_{(:,:,k)}$, lateral slices $\mathcal{A}_{(:,j,:)}$ and tube fibers $\mathcal{A}_{(i,j,:)}$. We abbreviate $A_k = \mathcal{A}_{(:,:,k)}$ for simplication. An $ln \times m$ matrix is obtained by the operator **unfold**$(\mathcal{A})$, whereas the operator **fold** folds this matrix back to the tensor $\mathcal{A}$, i.e.,

$$\mathbf{unfold}\,(\mathcal{A}) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}, \mathbf{fold}\,(\mathbf{unfold}\,(\mathcal{A})) = \mathcal{A}.$$

Definition 1. *Let $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, then a block-circulant matrix of $\mathcal{A}$ is denoted by* **bcirc**$(\mathcal{A})$*, i.e.,*

$$\mathbf{bcirc}\,(\mathcal{A}) = \begin{bmatrix} A_1 & A_n & \cdots & A_2 \\ A_2 & A_1 & \cdots & A_3 \\ \vdots & \vdots & \ddots & \vdots \\ A_n & A_{n-1} & \cdots & A_1 \end{bmatrix}.$$

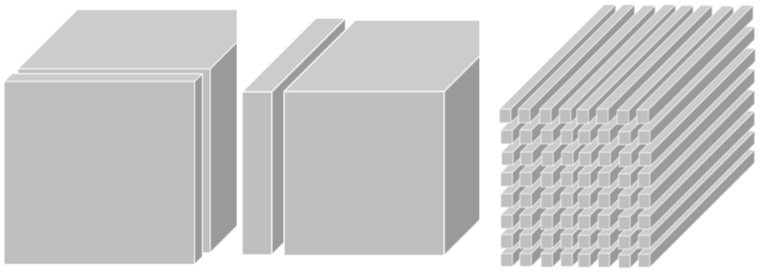

Figure 1. (a) frontal slices $\mathcal{A}_{(:,:,k)}$, (b) lateral slices $\mathcal{A}_{(:,j,:)}$ and (c) tube fibers $\mathcal{A}_{(i,j,:)}$

Definition 2. *([1]) Given two tensors $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$ and $\mathcal{B} \in \mathbb{R}^{m\times p\times n}$, the t-product $\mathcal{A} * \mathcal{B}$ is defined as*

$$\mathcal{A} * \mathcal{B} = \mathbf{fold}(\mathbf{bcirc}(\mathcal{A})\mathbf{unfold}(\mathcal{B})) = \mathcal{C}, \tag{10}$$

where $\mathcal{C} \in \mathbb{R}^{l\times p\times n}$.

The following remarks will be used in Section 3.

Remark 1. *([14]) For suitable tensors $\mathcal{A}$ and $\mathcal{B}$, it holds that*

(1). $\mathbf{bcirc}(\mathcal{A} * \mathcal{B}) = \mathbf{bcirc}(\mathcal{A}) * \mathbf{bcirc}(\mathcal{B})$.
(2). $\mathbf{bcirc}(\mathcal{A}^T) = \mathbf{bcirc}(\mathcal{A})^T$.
(3). $\mathbf{bcirc}(\mathcal{A} + \mathcal{B}) = \mathbf{bcirc}(\mathcal{A}) + \mathbf{bcirc}(\mathcal{B})$.

Let F_n be an n-by-n unitary discrete Fourier transform matrix, i.e,

$$F_n = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix},$$

where $\omega = e^{\frac{-2\pi i}{n}}$, then we get the tensor $\hat{\mathcal{A}}$ generated by using FFT along each tube of $\mathcal{A}$, i.e,

$$\mathbf{bdiag}\left(\hat{\mathcal{A}}\right) = \begin{bmatrix} \hat{A}_1 & & & \\ & \hat{A}_2 & & \\ & & \ddots & \\ & & & \hat{A}_n \end{bmatrix} = (F_n \otimes I_l)\, \mathbf{bcirc}\,(\mathcal{A})\, (F_n^* \otimes I_m)\,, \tag{11}$$

where $\otimes$ is the Kronecker product, F_n^* is the conjugate transposition of F_n and $\hat{A}_i$ denetes the frontal slices of $\hat{\mathcal{A}}$. Thus the t-product of $\mathcal{A}$ and $\mathcal{B}$ in (10) can be expressed by

$$\mathcal{A} * \mathcal{B} = \mathbf{fold}\left((F_n^* \otimes I_l)\left((F_n \otimes I_l)\,\mathbf{bcirc}\,(\mathcal{A})\,(F_n^* \otimes I_m)\right)(F_n \otimes I_m)\,\mathbf{unfold}\,(\mathcal{B})\right), \tag{12}$$

and (10) is reformulated as

$$\begin{pmatrix} \hat{A}_1 & & & \\ & \hat{A}_2 & & \\ & & \ddots & \\ & & & \hat{A}_n \end{pmatrix} \begin{pmatrix} \hat{B}_1 \\ \hat{B}_2 \\ \vdots \\ \hat{B}_n \end{pmatrix} = \begin{pmatrix} \hat{C}_1 \\ \hat{C}_2 \\ \vdots \\ \hat{C}_n \end{pmatrix}. \tag{13}$$

It is easy to calculate (12) in MATLAB.

For a non-zero tensor $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$, we can decompose it in the form

$$\vec{\mathcal{X}} = \vec{\mathcal{D}} * \boldsymbol{d}, \tag{14}$$

where $\vec{\mathcal{D}} \in \mathbb{R}^{m\times 1\times n}$ is a normalized tensor; see, e.g., [6] and $d \in \mathbb{R}^{1\times 1\times n}$ is a tube scalar. Algorithm 1 summarizes the decomposition in (14).

Algorithm 1 Normalization

```
Input:$\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$ is a nonzero tensor
Output:$\vec{\mathcal{D}}$, $\boldsymbol{d}$ with $\vec{\mathcal{X}} = \vec{\mathcal{D}} * \boldsymbol{d}$, $\|\vec{\mathcal{D}}\| = 1$
$\vec{\mathcal{D}} \leftarrow$ fft($\vec{\mathcal{X}}$,[ ],3)
for $j = 1, 2, \ldots, n$ do
    $\boldsymbol{d}_j \leftarrow \|\vec{\mathcal{D}}_j\|_2$ ($\vec{\mathcal{D}}_j$ is a vector)
    if $\boldsymbol{d}_j > tol$ then
        $\vec{\mathcal{D}}_j \leftarrow \frac{1}{\boldsymbol{d}_j}\vec{\mathcal{D}}_j$
    else
        $\vec{\mathcal{D}}_j \leftarrow \mathbf{randn}(m,1)$; $\boldsymbol{d}_j \leftarrow \|\vec{\mathcal{D}}_j\|_2$; $\vec{\mathcal{D}}_j \leftarrow \frac{1}{\boldsymbol{d}_j}\vec{\mathcal{D}}_j$; $\boldsymbol{d}_j \leftarrow 0$
    end if
end for
$\vec{\mathcal{D}} \leftarrow$ ifft($\vec{\mathcal{D}}$,[ ],3); $\boldsymbol{d} \leftarrow$ ifft($\boldsymbol{d}$,[ ],3)
```

Given a tensor $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, the singular value decomposition (tSVD) of $\mathcal{A}$ is expressed as

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T,$$

where $\mathcal{U} \in \mathbb{R}^{l\times l\times n}$ and $\mathcal{V} \in \mathbb{R}^{m\times m\times n}$ are orthogonal under t-product,

$$\mathcal{S} = \text{diag}[s_1, s_2, ..., s_{\min\{l,m\}}] \in \mathbb{R}^{m\times l\times n}$$

is an upper triangular tensor with the singular tubes s_j satisfying

$$\|s_1\|_F \geq \|s_2\|_F \geq \cdots \geq \|s_{\min\{l,m\}}\|_F.$$

The operators **squeeze** and **twist** [13] are expressed by

$$X = \textbf{squeeze}(\vec{\mathcal{X}}_j) \Longrightarrow X(i,j) = \vec{\mathcal{X}}_{(i,1,j)}, \textbf{twist}(\textbf{squeeze}(\vec{\mathcal{X}})) = \vec{\mathcal{X}}.$$

Figure 2 illustrates the transformation between a matrix and a tensor column by using **squeeze** and **twist**. Generally, the operators **multi_squeeze** and **multi_twist** are defined for a third-order tensor to make it squeezed or twisted. For a tensor $\mathcal{D} \in \mathbb{R}^{m\times p\times n}$ with $p > 1$, $\mathcal{C} = \textbf{multi\_squeeze}(\mathcal{D})$ means that all side slices of $\mathcal{D}$ are squeezed and stacked as front slices of $\mathcal{C}$, the operator **multi_twist** is the reverse operation of **multi_squeeze**. Thus $\textbf{multi\_twist}(\textbf{multi\_squeeze}(\mathcal{D})) = \mathcal{D}$. We refer to Table 1 for more notations and definitions.

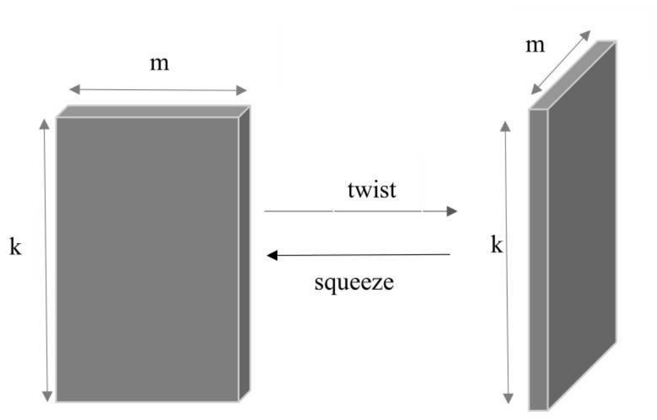

Figure 2. twist-squeeze

Table 1. Description of notations

Notation	Interpretation
$\mathcal{A}^T$	transpose of tensors
$\mathcal{A}^{-1}$	inverse of tensor,$\mathcal{A}^{-T} = (\mathcal{A}^{-1})^T = (\mathcal{A}^T)^{-1}$
$\hat{\mathcal{A}}$	FFT of $\mathcal{A}$ along the third mode
$\textbf{unfold}(\mathcal{A})$	the block column matrix of $\mathcal{A}$
$\textbf{bcirc}(\mathcal{A})$	the block-circulant matrix
$\mathcal{I}$	identity tensor
A	matrix
I	identity matrix
$\|\mathcal{A}\|_F$	the Frobenius norm of tensors $\mathcal{A}$, i.e, $\|\mathcal{A}\|_F = \sqrt{\sum_{i=1}^{l}\sum_{j=1}^{m}\sum_{k=1}^{n} a_{ijk}^2}$.
$*$	t-product
$\vec{\mathcal{A}}_j$, $\mathcal{A}_{(:,j,:)}$	the jth tensor column of $\mathcal{A}$, jth lateral slice of $\mathcal{A}$
$\mathcal{A}_{(:,:,j)}$	the jth frontal slice of tensor $\mathcal{A}$
$\boldsymbol{d}$	tube
$\langle\mathcal{A},\mathcal{B}\rangle$	$\langle\mathcal{A},\mathcal{B}\rangle = \sum_{ijk} a_{ijk}b_{ijk}$
$\left\langle\vec{\mathcal{A}},\vec{\mathcal{B}}\right\rangle$	$\left\langle\vec{\mathcal{A}},\vec{\mathcal{B}}\right\rangle = \sum_{ik} a_{i1k}b_{i1k}$

3. Tensor Conjugate-Gradient methods

This section first discusses the automatical determination of a suitable regularization parameter for the tensor conjugate gradient (tCG) method presented by Kilmer et al. in [13]. We abbreviate the improved method as auto-tCG. A truncated auto-tCG method is developed to improve the auto-tCG method and is abbreviated as auto-ttCG. A preprocessed version of the auto-ttCG method is presented, which is abbreviated as auto-ttpCG.

3.1. The auto-tCG Method

The tensor Conjugate-Gradient (t-CG) method is presented in [2] for the least-squares solution of the tensor linear systems $\mathcal{A} * \vec{\mathcal{X}} = \vec{\mathcal{B}}$. The regularization parameter in the t-CG method was not discussed and was user-specified. This subsection improves the t-CG method by employing the discrepancy principle to determine a suitable regularization parameter under the assumption (3) and uses it to solve the normal equation (6). We consider the polynomial function

$$\mu_k = \mu_0 q^k, k = 0, 1, \ldots, \tag{15}$$

where $q \in (0, 1)$. We set $\mu_0 = \|\mathcal{A}\|_F$, and obtain an optimal regularization parameter by continuously reducing the parameter. An effective method to deal with the general problems (9) is to regard it as p independent subproblems (4), i.e.,

$$\min_{\vec{\mathcal{X}}_j \in \mathbb{R}^{m\times 1\times n}} \left\{ \|\mathcal{A} * \vec{\mathcal{X}}_j - \vec{\mathcal{B}}_j\|_F^2 + \mu \|\vec{\mathcal{X}}_j\|_F^2 \right\}, j = 1, \ldots, p, \tag{16}$$

where $\vec{\mathcal{B}}_j$ is the tensor column of the tensor $\mathcal{B}$ and is polluted by the noise $\vec{\mathcal{E}}_j$. $\vec{\mathcal{B}}_{j,true}$ represents unknown error-free tensor. Assume the noise tensor

$$\vec{\mathcal{E}}_j = \vec{\mathcal{B}}_j - \vec{\mathcal{B}}_{j,true}$$

can be used or the norm of $\vec{\mathcal{E}}_j$ can be estimated, i.e.,

$$\|\vec{\mathcal{E}}_j\|_F \le \delta_j, j = 1, \ldots, p.$$

Algorithm 2 summarizes the auto-tCG method for solving (9). The initial tensor of Algorithm 2 is set as zero tensor. The iteration is stopped when the Frobenius norm of the residual tensor

$$\vec{\mathcal{R}}_{j,\mu_k}^i = \mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_{j,\mu_k}^i$$

is small enough, where $\vec{\mathcal{R}}_{j,\mu_k}^i$ denotes the residual generated by the i-th iterative solution $\vec{\mathcal{X}}_{j,\mu_k}^i$ of the normal equation with μ_k of the j-th independent subproblem. Let $\vec{\mathcal{X}}_{int} = \vec{\mathcal{X}}_{\mu_k}^*$ be the initial tensor of the normal equation of μ_{k+1}. When $\mu = \mu_k$ with m iterations for the CG-process, the affine space is $\vec{\mathcal{X}}_{\mu_k}^0 + \mathcal{K}_m \left(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}, r_{\mu_k}^0 \right)$, where $r_{\mu_k}^0 = \mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_{\mu_k}^0$.

Algorithm 2 The auto-tCG method for sloving (9).

```
**Input:** $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}, \delta_j, j = 1, ..., p, \mu_0, \eta > 1$.
**Output:** Approximate solution $\mathcal{X}^*$ of problem (9).
**for** $j = 1, 2, ...p$ **do**
    $\vec{\mathcal{X}}_{int} = 0, k = 0$.
    **while do** $\|\mathcal{A} * \vec{\mathcal{X}}^*_{j,\mu_k} - \vec{\mathcal{B}}_j\|_F^2 > \eta^2\delta_j^2$
        $k = k+1$, $(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_j = \mathcal{A}^T * \vec{\mathcal{B}}_j$, e.g., $\mu_k = \mu_0 q^k$
        $[\vec{\mathcal{R}}_0, a] \leftarrow$ Normalize$(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_{int})$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
        $i = 0, \sigma > tol$.
        **while** $\sigma > tol$ **do**
            $i = i+1$.
            $c = \left(\vec{\mathcal{P}}_{i-1}^T * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{P}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)$.
            $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * c$.
            $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \left(\vec{\mathcal{P}}_{i+1} * c\right)$.
            $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
            $d = \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i\right)$.
            $\vec{\mathcal{P}}_i = \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * d$.
        **end while**
        $\vec{\mathcal{X}}^*_{j,\mu_k} = \vec{\mathcal{X}}_i * a$ ($\vec{\mathcal{X}}^*_{j,\mu_k}$ is the solution of the normal equation about $\mu_k$ of the $j$-th independent subproblem (4)).
        $\vec{\mathcal{X}}_{int} = \vec{\mathcal{X}}^*_{j,\mu_k}$.
    **end while**
    $\mathcal{X}^*_{(:,j,:)} = \vec{\mathcal{X}}^*_{j,\mu_k}$.
**end for**
```

3.2. The truncated tensor Conjugate-Gradient method

Frommer and Maass in [15] proposed a good condition that can roughly judge some inappropriate value of μ. We introduce this condition to improve Algorithm 2 by excluding some unsuitable value of μ, and present a truncated tensor conjugate-gradient method for solving (9). We first give the following results.

Theorem 1. *Given $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, define a t-linear operator T: $\mathbb{R}^{m\times 1\times n} \to \mathbb{R}^{l\times 1\times n}$, i.e., $T(\vec{\mathcal{X}}) = \mathcal{A} * \vec{\mathcal{X}}$ with $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$. Let $\vec{\mathcal{X}}^*_\mu$ be the exact solution of the normal equations*

$$(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}} = \mathcal{A}^T * \vec{\mathcal{B}},$$

then for an arbitrary $\mathcal{X} \in \mathbb{R}^{m\times 1\times n}$, we have

$$\|\mathcal{A} * \vec{\mathcal{X}}^*_\mu - \vec{\mathcal{B}}\|_F^2 \geq \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu}\|\mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}}\|_F^2.$$

Proof. For an arbitrary $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$, set $\vec{\mathcal{Z}} = \vec{\mathcal{X}}^*_\mu - \vec{\mathcal{X}}$. Let the singular value decomposition of $\mathcal{A}$ be $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, then

$$\mathcal{A} * \vec{\mathcal{Z}} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T * \vec{\mathcal{Z}}.$$

Suppose $\mathcal{V}^T * \vec{\mathcal{Z}} = \vec{\mathcal{D}} \in \mathbb{R}^{m\times 1\times n}$, then

$$\|\mathcal{A} * \vec{\mathcal{Z}}\|_F^2 = \|\mathcal{U} * \mathcal{S} * \mathcal{V}^T * \vec{\mathcal{Z}}\|_F^2 = \|\mathcal{S} * \vec{\mathcal{D}}\|_F^2 = \|\mathbf{bcirc}(\mathcal{S})\mathbf{unfold}(\vec{\mathcal{D}})\|_2^2. \tag{17}$$

Thus

$$\begin{aligned}
&\|(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{Z}}\|_F^2 \\
=&\|\mathcal{V} * (\mathcal{S}^T * \mathcal{S} + \mu\mathcal{I}) * \mathcal{V}^T * \vec{\mathcal{Z}}\|_F^2 = \|\mathcal{V} * (\mathcal{S}^T * \mathcal{S} + \mu\mathcal{I}) * \vec{\mathcal{D}}\|_F^2 \\
=&\|\left(\mathcal{S}^T * \mathcal{S} + \mu\mathcal{I}\right) * \vec{\mathcal{D}}\|_F^2 = \|(\mathbf{bcirc}(\mathcal{S}^T * \mathcal{S}) + \mu\mathbf{bcirc}(\mathcal{I}))\mathbf{unfold}(\vec{\mathcal{D}})\|_2^2 \\
=&\|(\mathbf{bcirc}(\mathcal{S})^T\mathbf{bcirc}(\mathcal{S}) + \mu\mathbf{bcirc}(\mathcal{I}))\mathbf{unfold}(\vec{\mathcal{D}})\|_2^2.
\end{aligned} \tag{18}$$

Denote $\mathbf{bcirc}(\mathcal{S}) = \mathbf{S} \in \mathbb{R}^{nl\times nm}$, $\mathbf{bcirc}(\mathcal{I}) = \mathbf{I} \in \mathbb{R}^{nm\times nm}$ and $\mathbf{unfold}(\vec{\mathcal{D}}) = \mathbf{d} \in \mathbb{R}^{nm\times 1}$, then $\|\mathcal{A} * \vec{\mathcal{Z}}\|_F^2 = \|\mathbf{Sd}\|_2^2$ and $\|(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{Z}}\|_F^2 = \|(\mathbf{S}^T\mathbf{S} + \mu\mathbf{I})\mathbf{d}\|_2^2$. Thus we transform the tensor norm into the equivalent matrix norm. Let the singular value decomposition of $\mathbf{S}$ be $\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\Sigma = \text{diag}\left(\sigma_1, \sigma_2, ..., \sigma_r\right), r \le \min\{nl, nm\}$, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_r]$ are orthogonal matrices with orthogonal columns $\mathbf{u}_k \in \mathbb{R}^{nl\times 1}$ and $\mathbf{v}_k \in \mathbb{R}^{nm\times 1}$, respectively. Thus we have

$$\mathbf{Sd} = \sum_{\sigma_k>0} \sigma_k \langle \mathbf{d}, \mathbf{v}_k\rangle \mathbf{u}_k.$$

Using the equation $s^2 = (s + \mu s^{-1})^{-2}(s^2 + \mu)^2$ with the estimate

$$\frac{1}{s + \mu s^{-1}} \le \frac{1}{2\sqrt{\mu}}, (s, \mu > 0),$$

we have

$$\begin{aligned}
\|\mathbf{Sd}\|_2^2 &= \sum_{\sigma_k>0} \sigma_k^2 \left|\langle \mathbf{d}, v_k\rangle\right|^2 = \sum_{\sigma_k>0} \left(\sigma_k + \mu\sigma_k^{-1}\right)^{-2} \left(\sigma_k^2 + \mu\right)^2 |\langle \mathbf{d}, v_k\rangle|^2 \\
&\le \frac{1}{4\mu} \sum_{\sigma_k>0} \left(\sigma_k^2 + \mu\right)^2 |\langle \mathbf{d}, v_k\rangle|^2.
\end{aligned} \tag{19}$$

Note that

$$\|\left(\mathbf{S}^T\mathbf{S} + \mu\mathbf{I}\right)\mathbf{d}\|_2^2 = \sum_{\sigma_k>0} \left(\sigma_k^2 + \mu\right)^2 \left|\langle \mathbf{d}, v_k\rangle\right|^2, \tag{20}$$

It results from (19) and (20) that

$$\|\mathbf{Sd}\|_2^2 \le \frac{1}{4\mu}\|\left(\mathbf{S}^T\mathbf{S} + \mu\mathbf{I}\right)\mathbf{d}\|_2^2.$$

Note that $\|\mathcal{A} * \vec{\mathcal{Z}}\|_F^2 = \|\mathbf{Sd}\|_2^2$ and $\|(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{Z}}\|_F^2 = \|(\mathbf{S}^T\mathbf{S} + \mu\mathbf{I})\mathbf{d}\|_2^2$, we have

$$\|\mathcal{A} * \vec{\mathcal{Z}}\|_F^2 \le \frac{1}{4\mu}\|(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{Z}}\|_F^2. \tag{21}$$

Thus

$$\begin{aligned}
\|\mathcal{A} * \vec{\mathcal{X}}_\mu^* - \vec{\mathcal{B}}\|_F^2 &= \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}} + \mathcal{A} * \left(\vec{\mathcal{X}}_\mu^* - \vec{\mathcal{X}}\right)\|_F^2 \\
&\ge \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F^2 - \|\mathcal{A} * \vec{\mathcal{Z}}\|_F^2 \\
&\ge \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu}\|\left(\mathcal{A}^T * \mathcal{A} + \mu\vec{\mathcal{I}}\right)\vec{\mathcal{Z}}\|_F^2
\end{aligned} \tag{22}$$

Note that

$$(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{Z}} = (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * (\vec{\mathcal{X}}_\mu^* - \vec{\mathcal{X}}) = \mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}}, \tag{23}$$

doi:10.20944/preprints202310.1696.v1

then (23) and (22) result in

$$\|\mathcal{A} * \vec{\mathcal{X}}_{\mu}^{*} - \vec{\mathcal{B}}\|_F^2 \geq \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu}\|\mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}}\|_F^2.$$

□

We will apply Theorem 1 to predict in advance whether the exact solution $\vec{\mathcal{X}}_{\mu_k}^{*}$ satisfies the discrepancy principle in Algorithm 2. We add the condition

$$\|\mathcal{A} * \vec{\mathcal{X}}_{\mu_k}^{i} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu_k}\|\vec{\mathcal{R}}_{\mu_k}^{i}\|_F^2 > \eta^2\delta^2 \tag{24}$$

in steps 9-16 of Algorithm 2. If the i-th iteration solution of the normal equation with μ_k is $\vec{\mathcal{X}}_{\mu_k}^{i}$ and its residual $\vec{\mathcal{R}}_{\mu_k}^{i}$ satisfies (24), then $\|\mathcal{A} * \vec{\mathcal{X}}_{\mu_k}^{*} - \vec{\mathcal{B}}\|_F^2 > \eta^2\delta^2$. This indicates that the exact solution of the normal equation with μ_k does not satisfy the discrepancy principle, so continue to solve next normal equation with μ_{k+1}. Therefore, we obtain a truncated tensor conjugate-gradient method of automatical determination of a suitable regularization parameter, which is abbreviated as auto-ttCG. Algorithm 3 summarizes the auto-ttCG method.

Algorithm 3 The auto-ttCG method for sloving (9)

```
Input:$\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}, \delta_j, j = 1, ..., p, \mu_0, \eta > 1,$ tol.
Output: Approximate solution $\mathcal{X}^*$ of problem (9).
for $j = 1, 2, ...p$ do
    $\vec{\mathcal{X}}_{int} = 0, k = 0$
    while $\|\mathcal{A} * \vec{\mathcal{X}}_{j,\mu_k}^{i} - \vec{\mathcal{B}}_j\|_F^2 > \eta^2\delta_j^2$ do
        $k = k + 1$, $(\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{X}}_j = \mathcal{A}^T * \vec{\mathcal{B}}_j$, e.g.$\mu_k = \mu_0 q^k$.
        $[\vec{\mathcal{R}}_0, a] \leftarrow$ Normalize$(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{X}}_{int})$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
        $i = 0$, $\sigma$=10tol, $\vec{\mathcal{X}}_{j,\mu_k}^{0} = \vec{\mathcal{X}}_{int}$.
        while $\sigma > tol$ and $\|\mathcal{A} * \vec{\mathcal{X}}_{j,\mu_k}^{i} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu_k}\|\vec{\mathcal{R}}_i * a\|_F^2 < \eta^2\delta^2$ do
            $i = i + 1$.
            $c = \left(\vec{\mathcal{P}}_{i-1}^T * (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{P}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)$.
            $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * c$, $\vec{\mathcal{X}}_{j,\mu_k}^{i} = \vec{\mathcal{X}}_i * a$.
            $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \left(\vec{\mathcal{P}}_{i+1} * c\right)$
            $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
            $d = \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i\right)$.
            $\vec{\mathcal{P}}_i = \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * d$.
        end while
        $\vec{\mathcal{X}}_{int} = \vec{\mathcal{X}}_{\mu_k}^{i}$.
    end while
    $\mathcal{X}_{(:,j,:)}^{*} = \vec{\mathcal{X}}_{j,\mu_k}^{i}$.
end for
```

3.3. *A preconditioned truncated tensor Conjugate-Gradient method*

In this section, we consider the acceleration of Algorithm 3 by preconditioning. When the tensor $\mathcal{M}$ is symmetric positive definite under the t-product structure, we can get its tensor approximate Cholesky decomposition (tChol) by Algorithm 4.

Algorithm 4 Tensor Cholesky decomposition (tChol)

```
1: Input: $\mathcal{M} \in \mathbb{R}^{m\times m\times n} \neq \mathcal{O}$
2: Output: $\mathcal{H} \in \mathbb{R}^{m\times m\times n}$ and $\mathcal{M} = \mathcal{H} * \mathcal{H}^T$.
3: $\hat{\mathcal{M}} \leftarrow$ fft($\mathcal{M}$,[ ],3)
4: for $j = 1, 2, ..., n$ do
5:     $H \leftarrow chol(\hat{\mathcal{M}}_{(:,:,j)})$, $H$ is the lower triangular matrix, which is obtained by approximate Cholesky decomposition.
6:     $\hat{\mathcal{H}}_{(:,:,j)} \leftarrow H$.
7: end for
8: $\mathcal{H} \leftarrow$ ifft($\hat{\mathcal{H}}$,[ ],3).
```

In Algorithm 3, the coefficient tensor $\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}$ of the $k-$th normal equation

$$(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}} = \mathcal{A}^T * \vec{\mathcal{B}} \tag{25}$$

is symmetric and positive definite. We set $\mathcal{M} = \mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}$ and apply Algorithm 4 to obtain the decomposition of $\mathcal{M} = \mathcal{H} * \mathcal{H}^T$, where each frontal slice of $\mathcal{H}$ is a fully sparse lower triangular matrix. After the normal equation (25) is preconditioned by $\mathcal{M}$, we solve the preconditioned normal equations

$$\tilde{\mathcal{A}} * \tilde{\vec{\mathcal{X}}} = \tilde{\vec{\mathcal{B}}}, \tag{26}$$

instead of equations (25) in Algorithm 3, where $\tilde{\mathcal{A}} = \mathcal{H}^{-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \mathcal{H}^{-T}$, $\tilde{\vec{\mathcal{X}}} = \mathcal{H}^T * \vec{\mathcal{X}}$, $\tilde{\vec{\mathcal{B}}} = \mathcal{H}^{-1} * \mathcal{A}^T * \vec{\mathcal{B}}$.

Applying Algorithm 3 to solve (26) instead of (25). Let $\vec{\mathcal{X}}_i$ and $\tilde{\vec{\mathcal{X}}}_i$ denote the solution of (25) and (26), respectively. Then we have

$$\begin{aligned}
\tilde{\vec{\mathcal{R}}}_i &= \tilde{\vec{\mathcal{B}}} - \tilde{\mathcal{A}} * \tilde{\vec{\mathcal{X}}}_i & (27)\\
&= \mathcal{H}^{-1} * \mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{H}^{-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \mathcal{H}^{-T}) * \mathcal{H}^T * \vec{\mathcal{X}}_i \\
&= \mathcal{H}^{-1} * (\mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_i) & (28)\\
&= \mathcal{H}^{-1} * \vec{\mathcal{R}}_i, & (29)
\end{aligned}$$

Let $\vec{\mathcal{W}}_i = \mathcal{H}^{-1} * \vec{\mathcal{R}}_i$, $\tilde{\vec{\mathcal{P}}}_{i-1} = \mathcal{H}^T * \vec{\mathcal{P}}_{i-1}$, then we have

$$\begin{aligned}
\tilde{d} &= (\tilde{\vec{\mathcal{R}}}_{i-1}^T * \tilde{\vec{\mathcal{R}}}_{i-1})^{-1} * (\tilde{\vec{\mathcal{R}}}_i^T * \tilde{\vec{\mathcal{R}}}_i) & (30)\\
&= ((\mathcal{H}^{-1} * \vec{\mathcal{R}}_{i-1})^T * \mathcal{H}^{-1} * \vec{\mathcal{R}}_{i-1})^{-1} * ((\mathcal{H}^{-1} * \vec{\mathcal{R}}_i)^T * \mathcal{H}^{-1} * \vec{\mathcal{R}}_i) \\
&= (\vec{\mathcal{W}}_{i-1}^T * \vec{\mathcal{W}}_{i-1})^{-1} * (\vec{\mathcal{W}}_i^T * \vec{\mathcal{W}}_i), & (31)
\end{aligned}$$

and

$$\begin{aligned}
\tilde{c} &= (\tilde{\vec{\mathcal{P}}}_{i-1}^T * \tilde{\mathcal{A}} * \tilde{\vec{\mathcal{P}}}_{i-1})^{-1} * (\tilde{\vec{\mathcal{R}}}_{i-1}^T * \tilde{\vec{\mathcal{R}}}_{i-1}) \\
&= ((\mathcal{H}^T * \vec{\mathcal{P}}_{i-1})^T * \mathcal{H}^{-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \mathcal{H}^{-T} * (\mathcal{H}^T * \vec{\mathcal{P}}_{i-1}))^{-1} * ((\mathcal{H}^{-1} * \vec{\mathcal{R}}_{i-1})^T * \mathcal{H}^{-1} * \vec{\mathcal{R}}_{i-1}) \\
&= ((\mathcal{H}^T * \vec{\mathcal{P}}_{i-1})^T * \mathcal{H}^{-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{P}}_{i-1})^{-1} * \vec{\mathcal{W}}_{i-1}^T * \vec{\mathcal{W}}_{i-1} \\
&= (\vec{\mathcal{P}}_{i-1}^T * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{P}}_{i-1})^{-1} * \vec{\mathcal{W}}_{i-1}^T * \vec{\mathcal{W}}_{i-1}. & (32)
\end{aligned}$$

In addition, we have the iteration

$$\begin{aligned}
\tilde{\vec{\mathcal{X}}}_i &= \tilde{\vec{\mathcal{X}}}_{i-1} + \tilde{\vec{\mathcal{P}}}_{i-1} * \tilde{c} \\
\mathcal{H}^T * \vec{\mathcal{X}}_i &= \mathcal{H}^T * \vec{\mathcal{X}}_{i-1} + \mathcal{H}^T * \vec{\mathcal{P}}_{i-1} * \tilde{c} \\
\vec{\mathcal{X}}_i &= \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * \tilde{c},
\end{aligned} \tag{33}$$

and

$$\begin{aligned}
\tilde{\vec{\mathcal{R}}}_i &= \tilde{\vec{\mathcal{R}}}_{i-1} - \tilde{\mathcal{A}} * \tilde{\vec{\mathcal{P}}}_{i+1} * \tilde{c} \\
\mathcal{H}^{-1} * \vec{\mathcal{R}}_i &= \mathcal{H}^{-1} * \vec{\mathcal{R}}_{i-1} - \mathcal{H}^{-1} * \left(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}\right) * \mathcal{H}^{-T} * \mathcal{H}^T * \vec{\mathcal{P}}_{i+1} * \tilde{c} \\
\vec{\mathcal{R}}_i &= \vec{\mathcal{R}}_{i-1} - \left(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}\right) * \vec{\mathcal{P}}_{i+1} * \tilde{c},
\end{aligned} \tag{34}$$

together with

$$\begin{aligned}
\tilde{\vec{\mathcal{P}}}_i &= \tilde{\vec{\mathcal{R}}}_i + \tilde{\vec{\mathcal{P}}}_{i-1} * \tilde{d} \\
\mathcal{H}^T * \vec{\mathcal{P}}_i &= \mathcal{H}^{-1} * \vec{\mathcal{R}}_i + \mathcal{H}^T * \vec{\mathcal{P}}_{i-1} * \tilde{d} \\
\vec{\mathcal{P}}_i &= \mathcal{H}^{-T} * \mathcal{H}^{-1} * \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * \tilde{d} = \mathcal{H}^{-T} * \vec{\mathcal{W}}_i + \vec{\mathcal{P}}_{i-1} * \tilde{d}.
\end{aligned} \tag{35}$$

Taking the preprocessing procedure (27)-(35) into Algorithm 3, we obtain the improved auto-ttCG method, which is called the truncated tensor preconditioned conjugate-gradient method of automatical determination of a suitable regularization parameter, and is abbreviated as auto-ttpCG. Algorithm 5 summarizes the auto-ttpCG method. Numerical experiments in Section show Algorighm 5 converges faster than Algorithm 3.

Algorithm 5 The auto-ttpCG method for sloving (9)

```
Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}$, $\vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}$, $\delta_j$, $j = 1, ..., p$, $\mu_0$, $\eta > 1$, tol.
Output: Approximate solution $\mathcal{X}^*$ of problem (9).
for $j = 1, 2, ...p$ do
    $\vec{\mathcal{X}}_{int} = 0, k = 0$
    while $\|\mathcal{A} * \vec{\mathcal{X}}^i_{j,\mu_k} - \vec{\mathcal{B}}_j\|_F^2 > \eta^2\delta_j^2$ do
        $k = k + 1$, $\mu_k = \mu_0 q^k$.
        $\mathcal{H} = tChol(\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I})$.
        $[\vec{\mathcal{R}}_0, a] \leftarrow \text{Normalize}(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{X}}_{int})$.
        $\vec{\mathcal{W}}_0 = \mathcal{H}^{-1} * \vec{\mathcal{R}}_0$, $\vec{\mathcal{P}}_0 = \mathcal{H}^{-T} * \vec{\mathcal{W}}_0$.
        $i = 0$, $\sigma$=10tol, $\vec{\mathcal{X}}^0_{j,\mu_k} = \vec{\mathcal{X}}_{int}$.
        while $\sigma > tol$ and $\|\mathcal{A} * \vec{\mathcal{X}}^i_{j,\mu_k} - \vec{\mathcal{B}}_j\|_F^2 - \frac{1}{4\mu_k}\|\vec{\mathcal{R}}_i * a\|_F^2 < \eta^2\delta^2$ do
            $i = i + 1$.
            $\tilde{c} = (\vec{\mathcal{P}}^T_{i-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{P}}_{i-1})^{-1} * \vec{\mathcal{W}}^T_{i-1} * \vec{\mathcal{W}}_{i-1}$.
            $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * \tilde{c}$, $\vec{\mathcal{X}}^i_{j,\mu_k} = \vec{\mathcal{X}}_i * a$.
            $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{P}}_{i+1} * \tilde{c}$, $\vec{\mathcal{W}}_i = \mathcal{H}^{-1} * \vec{\mathcal{R}}_i$
            $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
            $\tilde{d} = (\vec{\mathcal{W}}^T_{i-1} * \vec{\mathcal{W}}_{i-1})^{-1} * (\vec{\mathcal{W}}^T_i * \vec{\mathcal{W}}_i)$.
            $\vec{\mathcal{P}}_i = \mathcal{H}^{-T} * \vec{\mathcal{W}}_i + \vec{\mathcal{P}}_{i-1} * \tilde{d}$.
        end while
        $\vec{\mathcal{X}}_{int} = \vec{\mathcal{X}}^i_{\mu_k}$.
    end while
    $\mathcal{X}^*_{(:,j,:)} = \vec{\mathcal{X}}^i_{j,\mu_k}$.
end for
```

4. Numerical Examples

This section presents three examples to show the application of Algorithms 2, 3 and 5 on the restoration of image and video. All calculations are performed in MATLAB R2018a on computers with intel core i7 and 16GB ram.

Suppose $\mathcal{X}_k$ is the k-th approximate solution to the minimization problem (9). The quality of the approximate solution $\mathcal{X}_k$ is defined by *the relative error*

$$\mathbf{Err_k} = \frac{\|\mathcal{X}_k - \mathcal{X}_{true}\|_F}{\|\mathcal{X}_{true}\|_F},$$

and *the signal-to-noise ratio (SNR)*

$$\mathbf{SNR}(\mathcal{X}_k) = 10\log_{10}\frac{\|\mathcal{X}_{true} - E(\mathcal{X}_{true})\|_F^2}{\|\mathcal{X}_k - \mathcal{X}_{true}\|_F^2},$$

where $\mathcal{X}_{true}$ denotes the uncontaminated data tensor and $E(\mathcal{X}_{true})$ is the average gray-level of $\mathcal{X}_{true}$. The observed data $\mathcal{B}$ in (9) is contaminated by a "noise" tensor $\mathcal{E}$, i.e., $\mathcal{B} = \mathcal{B}_{true} + \mathcal{E}$. $\mathcal{E}$ is determined as follows. Let $\vec{\mathcal{E}}_j$ be the $j-$th transverse slice of $\mathcal{E}$, whose entries are scaled and normally distributed with a mean of zero, i.e.,

$$\vec{\mathcal{E}}_j = \nu\frac{\vec{\mathcal{E}}_{r,j}}{\|\vec{\mathcal{E}}_{r,j}\|_F}\|\vec{\mathcal{B}}_{true,j}\|_F, j = 1, ..., p, \tag{36}$$

where the data of $\vec{\mathcal{E}}_{r,j}$ is generated according to N(0, 1).

Example 4.1 (**Gray image**) This example considers the restoration of the blurred and noised *cameraman* image with the size of $256 \times 1 \times 256$. For the operator $\mathcal{A}$, its front slices $\mathcal{A}_{(:,:,i)}, i = 1, ..., 256$, are generated by using the MATLAB function *blur*, i.e.,

$$\begin{aligned} &z = [exp(-([0:band-1].^2)/(2\sigma^2)), zeros(1, N-band)],\\ &A = \frac{1}{\sigma\sqrt{2\pi}}toeplitz\left(\left[z\left(1\right) fliplr\left(z\left(2:end\right)\right)\right], z\right),\ \ \mathcal{A}_{(:,:,i)} = A\left(i,1\right)A \end{aligned} \tag{37}$$

with $N = 256$, $\sigma = 4$ and $band = 12$. The condition numbers of $\mathcal{A}_{(i)}$ are $cond(\mathcal{A}_{(:,:,1)}) = cond(\mathcal{A}_{(:,:,246)}) = ... = cond(\mathcal{A}_{(:,:,256)}) = 11.1559$, while he condition numbers of the remaining slices are infinite. Let X_{true} denote the original undaminated *cameraman* image. The operator **twist** converts X_{true} into tensor column $\vec{\mathcal{X}}_{true} \in \mathbb{R}^{256\times1\times256}$ for storage. The noised tensor $\vec{\mathcal{E}}$ is generated by (36) with different noise level $\nu = 10^{-i}, i = 2, 3$. The blurred and noisy images are generated by $\vec{\mathcal{B}} = \mathcal{A} * \vec{\mathcal{X}}_{true} + \vec{\mathcal{E}}$.

The auto-tCG, auto-ttCG and auto-ttpCG methods are used to solve the tensor discrete linear ill-posed problems (1). The discrepancy principle is employed to determine a suitable regularization parameter by using $\mu_k = \mu_0 q^k$ with $\mu_0 = \|\mathcal{A}\|_F$ and $q = \frac{1}{2}$. We set $\eta = 1.05$ in (8).

Figure 3 shows the convergence of relative errors verus (a) the iteration number k and (b) the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$ corresponding in the Table 2. The iteration process is terminated when the discrepancy principle is satisfied. From Figure 3 (a), we can see that the auto-ttCG and auto-ttpCG methods do not need to solve the normal equation for all $\mu_k (k < 8)$. This shows that the auto-ttCG and auto-ttpCG methods improve the auto-tCG method by the condition (24). Figure 3 (b) shows that the auto-ttpCG method converges fastest among three methods.

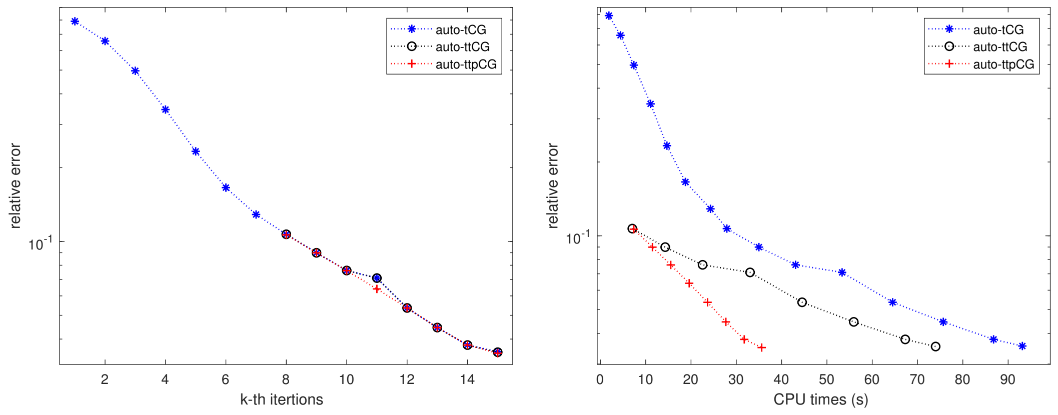

Figure 3. Example 4.1: Comparison of convergence between (a) relative errors verus the iteration number k and (b) relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$.

Table 2 lists the regularization parameter, the iteration number, the relative error, SNR and the CPU time of the optimal solution obtained by using the auto-tCG, auto-ttCG and auto-ttpCG methods with different noise levels $\nu = 10^{-i}, i = 2, 3$. It can be seen from Table 2 that the auto-ttpCG method has the lowest relative error, highest SNR and the least CPU time for different noise level.

Table 2. Example 4.1: Comparison of relative error, SNR, and CPU time between the auto-tCG, auto-ttCG and auto-ttpCG methods with different noise level $\nu = 10^{-i}, i = 2, 3$.

Noise level	Method	k	μ_k	Relative error	SNR	CPU (secs)
10^{-3}	auto-tCG	15	1.96e-05	3.54e-02	22.36	109.87
	auto-ttCG	15	1.96e-05	3.52e-02	22.41	80.93
	auto-ttpCG	15	1.96e-05	3.49e-02	22.48	33.98
10^{-2}	auto-tCG	11	3.14e-04	8.74e-02	14.51	81.94
	auto-ttCG	11	3.14e-04	8.64e-02	14.61	26.42
	auto-ttpCG	11	3.14e-04	8.54e-02	14.72	18.50

Figure 4 shows the reconstructed images obtained by using the auto-tCG, auto-ttCG and auto-ttpCG methods on the blurred and noised image with the noise level $\nu = 10^{-3}$ in Table 2. From Figure 4 we can see that the restored image by the auto-ttpCG method looks a bit better than others but the least CPU time.

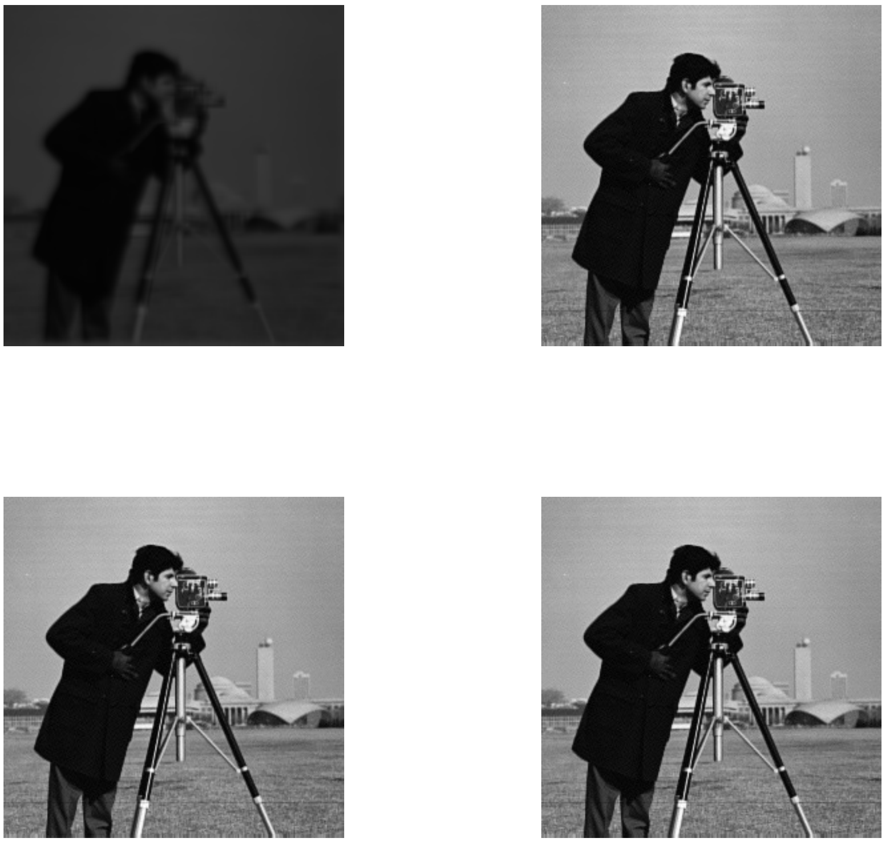

Figure 4. Example 4.1: (a) The blurred and noised image and reconstructed images by (b) the auto-tCG method (SNR=22.36, CPU=109.87), (c) the auto-ttCG method (SNR=22.41, CPU=80.93) and (d) the auto-ttpCG method (SNR=22.48, CPU=33.98) according to the noise level $\nu = 10^{-3}$ in Table 2.

Example 4.2 (**Color image**) This example shows the restoration of a blurred **Lena** color image by Algorithms 2, 3 and 5. The original **Lena** image $\mathcal{X}_{ori} \in \mathbb{R}^{256\times256\times3}$ is stored as a tensor $\mathcal{X}_{true} \in \mathbb{R}^{256\times3\times256}$ through the MATLAB function **multi_twist**. We set $N = 256, \sigma = 3$ and band=12, and get $\mathcal{A} \in \mathbb{R}^{256\times256\times256}$ by

$$z = \left[exp(-([0 : band - 1] .^{2})/(2\sigma^{2})), zeros(1, N - band)\right],$$

$$A = toeplitz(z), \mathcal{A}_{(:,:,i)} = \frac{1}{2\pi\sigma} A\,(i,1)\,A, i = 1, ..., 256.$$

Then $cond(\mathcal{A}_{(:,:,1)}) = ... = cond(\mathcal{A}_{(:,:,12)}) = 4.68e + 07$, and the condition number of other tensor slices of $\mathcal{A}$ is infinite. The noise tensor $\mathcal{E}$ is defined by (36). The blurred and noised tensor is derived by $\mathcal{B} = \mathcal{A} * \mathcal{X}_{true} + \mathcal{E}$, which is shown in Figure 6 (a).

We set the color image $\mathcal{B}$ to be divided into multiple lateral slices and independently process each slice through (1) by using the auto-tCG, auto-ttCG and auto-ttpCG methods. Figure 5 shows the convergence of relative errors verus (a) the iteration number k and (b) the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods when dealing with the first tensor lateral slice $\mathcal{B}_{(:,1,:)}$ of $\mathcal{B}$ with $\nu = 10^{-3}$. Similar results can be derived as that in Example 5.1 from Figure 5. We can see that the auto-ttCG and auto-ttpCG methods need less iterations than the auto-tCG method from Figure 5 (a) and the auto-ttpCG method converges fastest among all methods from Figure 5 (b).

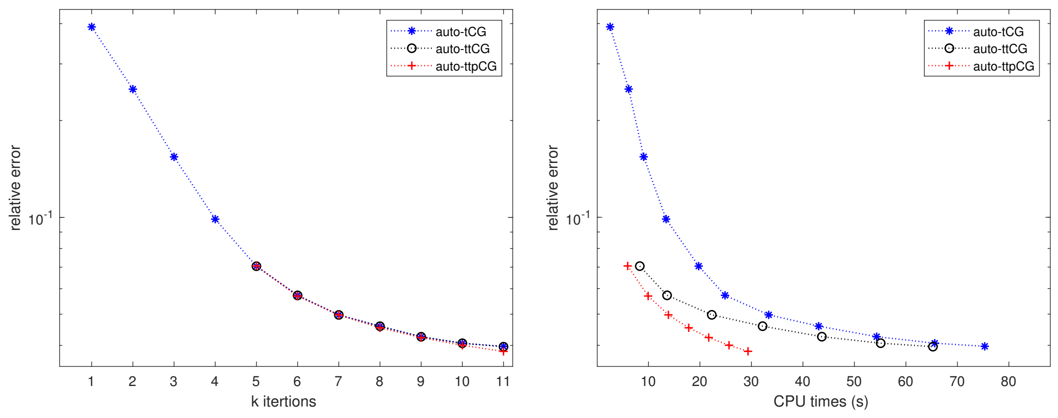

Figure 5. Example 4.2: Comparison of convergence between (a) relative errors verus the iteration number k and (b) relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$.

Table 3 lists the relative error, SNR and the CPU time of the optimal solution obtained by using the auto-tCG, auto-ttCG and auto-ttpCG methods with different noise levels $\nu = 10^{-i}, i = 2, 3$. The results are very similar to that in Table 2 for different noise level.

Table 3. Example 4.2: Comparison of relative error, SNR, and CPU time between the auto-tCG, auto-ttCG and auto-ttpCG methods with different noise level $\nu = 10^{-i}, i = 2, 3$.

Noise level	Method	Relative error	SNR	time (secs)
	auto-tCG	5.90e-02	14.62	314.73
10^{-3}	auto-ttCG	5.90e-02	14.62	262.81
	auto-ttpCG	5.43e-02	15.37	103.41
	auto-tCG	7.64e-02	12.37	117.48
10^{-2}	auto-ttCG	7.48e-02	12.55	62.01
	auto-ttpCG	7.01e-02	13.13	54.85

Figure 6 shows the recovered images by the auto-tCG, auto-ttCG and auto-ttpCG methods corresponding to the results with noise level $\nu = 10^{-3}$. The results are very similar to that in Figure 6.

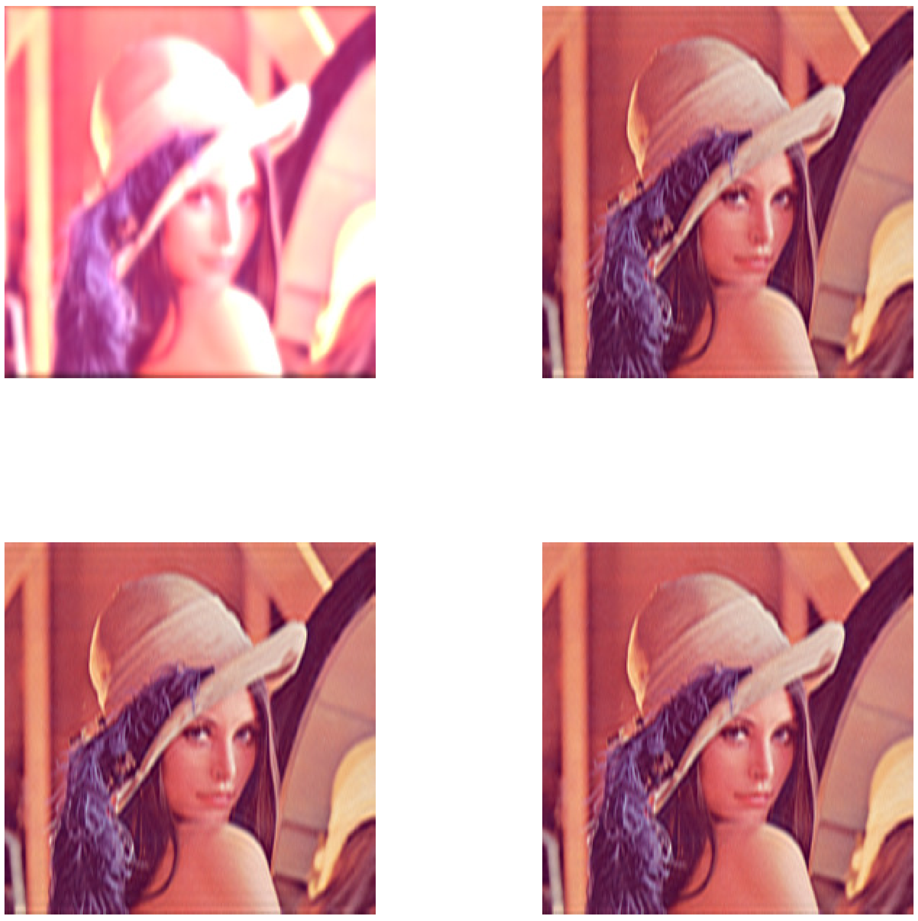

Figure 6. Example 4.2: (a) The blurred and noised *Lena* image and reconstructed images by (b) the auto-tCG method, (c) the auto-ttCG and (d) the auto-ttpCG method according to the noise level $\nu = 10^{-3}$ in Table 3.

Example 4.3 (**Video**) We recover the first 10 consecutive frames of blurred and noised *Rhinos* video from MATLAB. Each frame has 240 × 240 pixels. We store 10 pollution- and noise-free frames of the original video in the tensor $\mathcal{X}_{true} \in \mathbb{R}^{240\times10\times240}$. Let z be defined by (37) with $N = 240$, $\sigma = 2$ and $band = 12$. The coefficient tensor $\mathcal{A}$ is defined as follows:

$$A = \frac{1}{\sqrt{2\pi\sigma}} toeplitz\,(z)\,, \mathcal{A}_{(:,:,i)} = \frac{1}{2\pi\sigma^2} A\,(i,1)\,A, i = 1, ..., 240.$$

The condition number of the frontal slices of $\mathcal{A}$ is $cond(\mathcal{A}_{(:,:,i)}) = 7.4484e + 09(i \leq 12)$, and the condition number of the remaining frontal sections of $\mathcal{A}$ is infinite. The suitable regularization parameter is determined by using the discrepancy principle with $\eta = 1.1$. The blurred- and noised tensor $\mathcal{B}$ is generated by $\mathcal{B} = \mathcal{A} * \mathcal{X}_{true} + \mathcal{E}$ with $\mathcal{E} \in \mathbb{R}^{120\times30\times120}$ being defined by (36).

Figure 7 shows the convergence of relative errors verus the iteration number k and relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods when the second frame of the video with $\nu = 10^{-3}$ is restored. Very similar results can be derived from Figure 7 to that in Example 5.1.

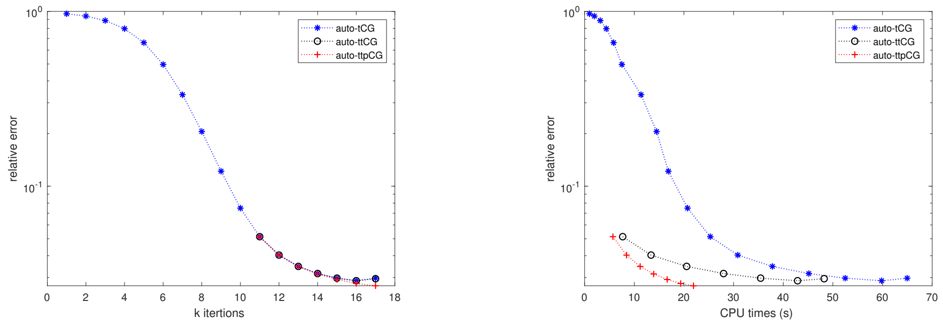

Figure 7. Example 4.3: Comparison of convergence between (a) relative errors verus the iteration number k and (b) relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$.

Table 4 displays the relative error, SNR and the CPU time of the optimal solution obtained by using the auto-tCG, auto-ttCG and auto-ttpCG methods for the second frame with different noise levels $\nu = 10^{-i}, i = 2, 3$. We can see that the auto-ttpCG method has the largest SNR and the lowest CPU time for different noise level $\nu = 10^{-i}, i = 2, 3$.

Table 4. Example 4.3: Comparison of relative error, SNR, and CPU time between the auto-tCG, auto-ttCG and auto-ttpCG methods with different noise level $\nu = 10^{-i}, i = 2, 3$.

Noise level	Method	Relative error	SNR	time (secs)
10^{-3}	auto-tCG	2.94e-02	23.17	697.78
	auto-ttCG	2.92e-02	23.23	487.35
	auto-ttpCG	2.66e-02	24.05	214.16
10^{-2}	auto-tCG	5.24e-02	18.15	480.75
	auto-ttCG	5.10e-02	18.38	281.54
	auto-ttpCG	4.74e-02	19.02	156.44

Figure 8 shows the original video, blurred and noised video, and the recovered video of the second frame of the video for the auto-tCG, auto-ttCG and the auto-ttpCG methods with noise level $\nu = 10^{-3}$ corresponding to the results in Table 4. The recovered frame by the auto-ttpCG method looks best among all recovered frames.

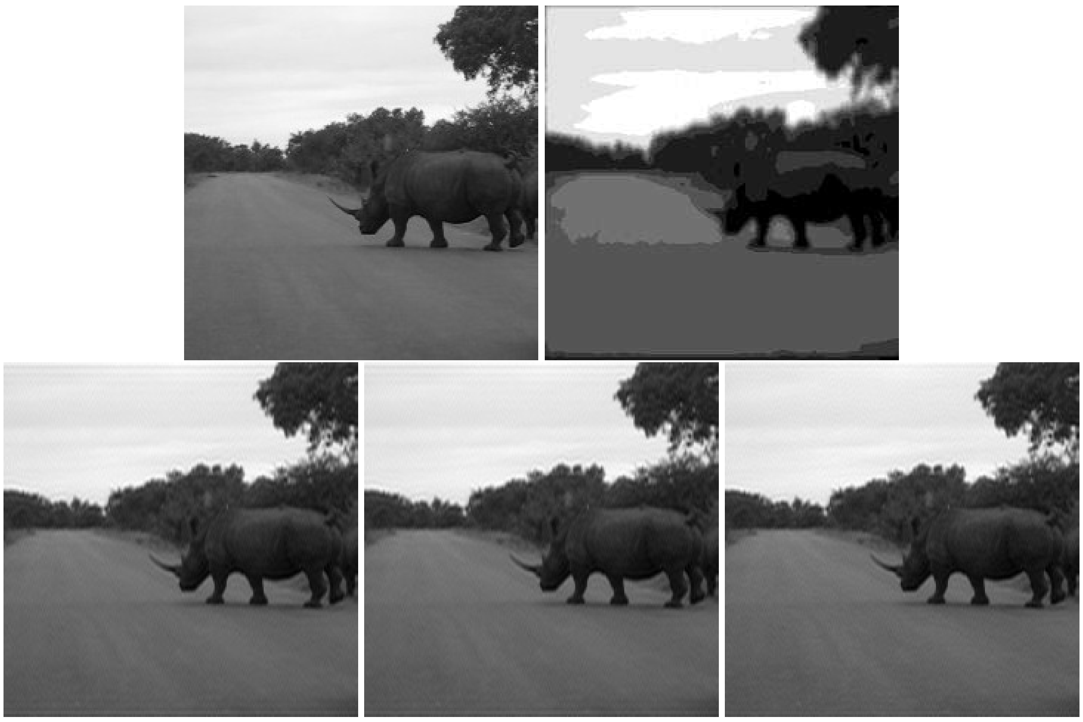

Figure 8. Example 4.3: (a) Original image, (b) the blurred and noisy image and recovered images by (c) the auto-tCG method, (d) the auto-ttCG and (e) the auto-ttpCG method according to the noise level $\nu = 10^{-3}$ in Table 4.

5. Conclusion

This paper presents three types of tensor Conjugate-Gradient methods for solving large-scale linear discrete ill-posed problems in tensor form. We first present an automatical determination strategy of a suitable regularization parameter for the tensor conjugate gradient (tCG) method. Furthermore, we develop a truncated version and a preprocessed verion of the tCG method. The proposed methods are used to different examples in image and video restoration.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgements

The authors would like to thank the referees for their helpful and constructive comments. This research was supported in part by the Sichuan Science and Technology Program (grant 2022ZYD0008).

Conflict of interest

The authors declare no conflict of interest.

References

1. Kilmer, M.E.; Martin, C.D. Factorization strategies for third order tensors. *Linear Alg. Appl.* **2011**, *435*, 641–658.
2. Hao, N.; Kilmer, M.E.; Braman, K.;Hoover, R.C. Facial recognition using tensor-tensor decompositions. *SIAM J. Imaging Sci.* **2013**, *6*, 437–463.

3. Zhang, J.; Saibaba, A. K.; Kilmer, M. E.; Aeron, S. A randomized tensor singular value decomposition based on the t-product. *Numer. Linear Algebr. Appl.* **2018**, *25*, e2179.
4. Fenu, C.; Reichel, L.; Rodriguez, G. GCV for tikhonov regularization via global Golub–Kahan decomposition. *Numer. Linear Algebr. Appl.* **2016**, *25*, 467–484.
5. Hansen, P. C. *Rank-Deficient and Discrete Ill-Posed Problems*. Publisher: SIAM, Philadelphia, 1998.
6. Kilmer, M. E.; Braman, K.; Hao, N.; Hoover, R. C. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 148–172.
7. Engl, H. W.; Hanke, M.; Neubauer, A. Neubauer. *Regularization of Inverse Problems.* Kluwer, Dordrecht, 1996.
8. Kindermann, S. Convergence analysis of minimization-based noise level-free parameter choice rules for linear ill-posed problems. *Electron. Trans. Numer. Anal.* **2011**, *38*, 233–257.
9. Kindermann, S.; Raik, K. A simplified L-curve method as error estimator. *Electron.Trans. Numer. Anal.* **2020**, *53*, 217–238.
10. Reichel, L.; Rodriguez, G. Old and new parameter choice rules for discrete ill-posed problems. *Numer. Algorithms* **2013**, *63*, 65-87.
11. Reichel, L.; Ugwu, U. O. The tensor Golub–Kahan–Tikhonov method applied to the solution of ill-posed problems with at-product structure. *Numer. Linear Algebr. Appl.* **2022**, *29*, e2412.
12. Ugwu, U. O.; Reichel, L. Tensor Arnoldi–Tikhonov and GMRES-Type Methods for Ill-Posed Problems with a t-Product Structure. *J. Sci. Comput.* **2022**, *90*, 1–39.
13. Kilmer, M. E.; Braman, K.; Hao, N.; Hoover, R. C. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM J. Matrix Anal. Appl. Appl.* **2013**, *34*, 148–172.
14. Lund, K. The tensor t-function: A definition for functions of third-order tensors. *Numer. Linear Algebr. Appl.* **2020**, *27*, e2288.
15. Frommer, A.; Maass, P. Fast CG-based methods for Tikhonov–Phillips regularization. *SIAM J. Sci. Comput.* **1999**, *20*, 1831-1850.
16. Cichocki, A.; Mandic, D.; De Lathauwer, L.; Zhou, G.; Zhao, Q.; Caiafa, C.; Phan, H. A. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Process. Mag.* **2015**, *32*, 145–163.
17. Signoretto, M.; Tran Dinh, Q.; De Lathauwer, L.; Suykens, J. A. Learning with tensors: a framework based on convex optimization and spectral regularization. Mach. Learn., 2014, 94, 303-351. *Mach. Learn.* **2014**, *94*, 303–351.
18. Kilmer, M. E.; Horesh, L.; Avron, H.; Newman, E. Tensor-tensor algebra for optimal representation and compression of multiway data. Proceedings of the National Academy of Sciences. *Proc. Natl. Acad. Sci. U. S. A.* **2021**, *118*, e2015851118.
19. Beik, F. P. A.; Najafi–Kalyani, M.; Reichel, L. Iterative Tikhonov regularization of tensor equations based on the Arnoldi process and some of its generalizations. *Appl. Numer. Math.* **151**, *118*, 425–447.
20. Bentbib, A. H.; Khouia, A.; Sadok, H. The LSQR method for solving tensor least-squares problems. *Electron. Trans. Numer. Anal.* **2022**, *55*, 92–111.
21. Bentbib, A. H.; El Hachimi, A.; Jbilou, K.; Ratnani, A. Fast multidimensional completion and principal component analysis methods via the cosine product. *Calcolo* **2022**, *59*, 26.
22. Khaleel, H. S.; Sagheer, S. V. M.; Baburaj, M.; George, S. N. Denoising of Rician corrupted 3D magnetic resonance images using tensor-SVD. *Biomed. Signal Process. Control* **2018**, *44*, 82-95.
23. Ugwu, U. O.; Reichel, L. Tensor regularization by truncated iteration: a comparison of some solution methods for large-scale linear discrete ill-posed problem with a t-product. *arXiv* **2021**, arXiv:2110.02485.
24. Zeng, C.; Ng, M. K. Decompositions of third-order tensors: HOSVD, T-SVD, and Beyond. *Numer. Linear Algebr. Appl.* **2020**, *27*, e2290.
25. El Hachimi, A.; Jbilou, K.; Ratnani, A.; Reichel, L. Spectral computation with third-order tensors using the t-product. *Appl. Numer. Math.* **2023**, *193*, 1-21.
26. Zheng, M. M.; Ni, G. Approximation strategy based on the T-product for third-order quaternion tensors with application to color video compression. *Appl. Math. Lett.* **2023**, *140*, 108587.
27. Yu, Q.; Zhang, X. (2023). T-product factorization based method for matrix and tensor completion problems.*Comput. Optim. Appl.* **2023**, *84*, 761–788.