mathematics

MDPI

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; w317153532@163.com (S.-W.W.); fyinsuse@163.com (F.Y.)
[2] College of Computer Science and Cyber Security, Chengdu University of Technology, Chengdu 610059, China
* Correspondence: huangx@cdut.edu.cn

Abstract: Ill-posed problems arise in many areas of science and engineering. Tikhonov is a usual regularization which replaces the original problem by a minimization problem with a fidelity term and a regularization term. In this paper, a tensor t-production structure preserved Conjugate-Gradient (tCG) method is presented to solve the regularization minimization problem. We provide a truncated version of regularization parameters for the tCG method and a preprocessed version of the tCG method. The discrepancy principle is used to automatically determine the regularization parameter. Several examples on image and video recover are given to show the effectiveness of the proposed methods by comparing them with some previous algorithms.

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

MSC: 65F10; 65F22

Citation: Wang, S.-W.; Huang, G.-X.; Yin, F. Tensor Conjugate Gradient Methods with Automatically Determination of Regularization Parameters for Ill-Posed Problems with t-Product. *Mathematics* **2024**, *12*, 159. https://doi.org/10.3390/math12010159

Received: 24 October 2023
Revised: 18 November 2023
Accepted: 27 December 2023
Published: 3 January 2024

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 [1–8]. A new t-product based on third-order tensors was proposed by Kilmer et al. [9,10]. When using high-dimensional data, t-product shows a greater potential value than matricization; see [1,2,10–16]. Compared to other products, the t-product preserves the inherent natural order and higher correlation embedded in the data, avoiding the loss of intrinsic information during the flattening process of the tensor; see [10]. The t-product has been found to have special value in many application fields, including image deblurring problems [1,2,9,11], image and video compression [8], facial recognition problems [10], etc.

The t-product is widely used in image and video restoration problems; see, e.g., [1,2,9]. In this paper, we consider the solution of large minimization problems of the form

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

Tensor $\mathcal{A}$ has a tubal rank that is difficult to determine, and many of its singular tubes are non-zero and have tiny Frobenius norms of different orders of magnitude. As the exponent increases, the Frobenius norms of these singular tubes rapidly decay to zero. Then, Problems (1) are called the tensor discrete linear ill-posed problems.

We assume that $\vec{\mathcal{B}}\in\mathbb{R}^{m\times 1\times n}$ is derived from an unknown and unavailable tensor $\vec{\mathcal{B}}_{true}$ polluted by noise $\vec{\mathcal{E}}\in\mathbb{R}^{m\times 1\times n}$,

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

We have $\vec{\mathcal{X}}_{true}$ represent a clear solution to Problem (1) to be found and obtain $\vec{\mathcal{B}}_{true}$ through $\mathcal{A} * \vec{\mathcal{X}}_{true} = \vec{\mathcal{B}}_{true}$. We assume that the upper bound of the Frobenius norm of $\vec{\mathcal{E}}$ is known,

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

A straightforward solution of (1) is usually meanless to obtain an approximation of $\vec{\mathcal{B}}_{true}$ because of the illposeness of $\mathcal{A} = [a]_{i,j,k=1}^{l,m,n}$ and because error $\vec{\mathcal{E}}$ is amplified severely. The Tikhonov regularization method is a mathematical approach proposed by Tikhonov [17] to address ill-posed problems. This method introduces a regularization term into the objective function, modeling the properties of the solution based on prior information. This serves to constrain the solution space, enhancing the stability of the problem. Therefore, we consider the application of the Tikhonov regularization method to address Problems (1) and subsequently proceed to solve the problems formulated in the manner of

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

where μ is a regularization parameter. We refer to (4) as the tensor penalty least-squares problems. 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 (4) is represented by

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

and under the condition given in (5), it admits a unique solution

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

There are many techniques to determine regularization parameter μ, such as the L-curve criterion, generalized cross-validation (GCV), and the discrepancy principle. We refer to [18–22] 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 small enough, and δ approaches 0, it results in $\vec{\mathcal{X}}_\mu \to \vec{\mathcal{X}}_{true}$. For more details regarding the discrepancy principle, please refer to [23].

In this paper, we additionally explore the extension of the minimization problems represented by (1), where the formulation takes 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 $\mathcal{B} \in \mathbb{R}^{m \times p \times n}$, $p > 1$.

In recent literature addressing discrete ill-posed Problems (1), the prevailing methodologies predominantly feature the application of Tikhonov regularization and truncated singular value methods. Reichel et al. confronted Problems (4) through the application of a subspace construction technique, thereby mitigating the inherent challenges of large-scale regularization problems by effecting a transformation into a more tractable smaller-scale formulation. As a result, they introduced the tensor Arnoldi–Tikhonov (tAT), GMRES-type methods (tGMRES) [2] and the tensor Golub–Kahan–Tikhonov method (tGKT) [1]. The truncated tensor singular value decomposition (T-tSVD) was introduced by Kilmer et al. and colleagues in [9]. Zhang et al. introduced the method of randomized tensor singular value decomposition (rt-SVD) in their work [24]. This method exhibits notable advantages

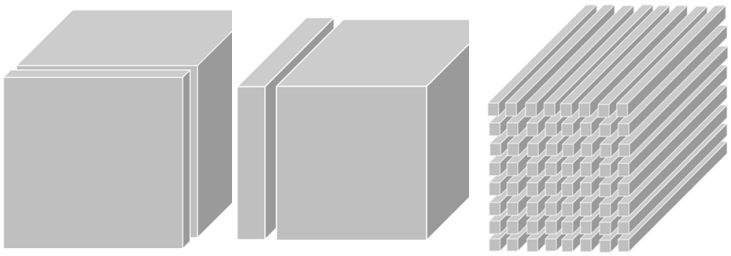
in handling large-scale datasets and holds significant potential for applications in image data compression and analysis. Ugwu and Reichel [25] proposed a new random tensor singular value decomposition (R-tSVD), which improves T-tSVD.

The conjugate gradient method (CG), initially proposed in [26], is well-suited for addressing large-scale problems, particularly in scenarios requiring multiple iterations. Compared to alternative methods, it demonstrates a relatively faster convergence rate. The method's favorable characteristic of low memory requirements renders it suitable for efficiently handling extensive datasets or high-dimensional problems. Detailed discussions on this approach can be found in the literature, as referenced in [26–30]. Song et al. [31] proposed a tensor conjugate gradient method for automatic parameterization in the Fourier domain (A-tCG-FFT). The A-tCG-FFT method projects Problems (1) into the Fourier domain and uses the CG method that preserves the matrix structure for computation. The solution obtained by the A-tCG-FFT method is of higher quality than the solution obtained by directly matrix or vectorizing the data. The tensor Conjugate Gradient (t-CG) method proposed by Kilmer et al. [10] is employed to address Problem (4), where the regularization parameter is user specified. In this article, we extend our work on the tCG method from [10] and utilize the discrepancy principle for automatic parameter estimation. 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. We remark that the auto-tCG, auto-ttCG, and auto-ttpCG methods are quite different from the methods in [31] because the former do not need to project the problems (1) into the Fourier domain and could maintain the t-product structure of tensors during the iteration process.

The remainder of this manuscript is structured as follows: Section 2 provides an introduction to relevant symbols and foundational concepts essential for the ensuing discussion. In Section 3, we expound upon the auto-tCG, auto-ttCG, and auto-ttpCG methodologies designed to address minimization Problems (4) and (9). Subsequently, Section 4 illustrates various examples pertaining to image and video restoration, while Section 5 encapsulates concluding remarks.

2. Preliminaries

This section provides notations and definitions, offering a concise overview of relevant results that are subsequently applied in the ensuing discourse. Figure 1 illustrates the frontal slices ($\mathcal{A}_{(:,:,k)}$), lateral slices ($\mathcal{A}_{(:,j,:)}$), and tube fibers ($\mathcal{A}_{(i,j,:)}$).



(**a**) (**b**) (**c**)

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

In this manuscript, we employ two operators, **unfold** and **fold**. The operator **unfold** unfolds the tensor into a matrix of dimensions $ln \times m$, while **fold** serves as the inverse of **unfold** folding the matrix back into its original three-order tensor form. For clarity of exposition, we denote matrix A_k specifically as the kth frontal slice of third-order tensor $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, i.e., $A_k = \mathcal{A}_{(:,:,k)}$. We have

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

The forthcoming definitions and remarks, introduced herein, are utilized in the subsequent theoretical proofs.

Definition 1. *Assuming $\mathcal{A}$ is a third-order tensor, the block-circulant matrix* $\mathbf{bcirc}(\mathcal{A})$ *is defined as follows:*

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

Definition 2 ([9])**.** *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 $\mathcal{C} \in \mathbb{R}^{l\times p\times n}$.

Remark 1 ([32])**.** *For any two tensors $\mathcal{A}$ and $\mathcal{B}$ for which the t-product is defined, they satisfy*

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

We define tensor $\hat{\mathcal{A}}$ obtained by applying the Fast Fourier Transform (FFT) along each tube of $\mathcal{A}$, i.e.,

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

where $\otimes$ is the Kronecker product, and matrix F_n^* represents the conjugate transpose of the n-by-n unitary discrete Fourier transform matrix F_n. The structure of F_n is defined as follows:

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

where $\omega = e^{\frac{-2\pi i}{n}}$. Thus the t-product in (10) can be represented as

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

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., ref. [11], 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
    $\vec{\mathcal{D}}_j \leftarrow$ **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 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 the t-product;

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

is an upper triangular tensor with the singular tubes s_j satisfying

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

Algorithm 2 introduces the tensor Conjugate Gradient (t-CG) method, presented in [11], for solving the least squares solution of the tensor linear systems (1).

Algorithm 2 The tCG method for sloving (4).

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \vec{\mathcal{B}} \in \mathbb{R}^{m\times 1\times n}$, μ.
Output: Approximate solution $\vec{\mathcal{X}}^*$ of Problem (4).
$\vec{\mathcal{X}}_0 = 0, k = 0$.
$[\vec{\mathcal{R}}_0, a] \leftarrow$ Normalize$(\mathcal{A}^T * \vec{\mathcal{B}} - (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{X}}_0)$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
for $i = 1, 2, \ldots$ until $\sigma < tol$ **do**
  $c = \left(\vec{\mathcal{P}}_{i-1}^T * (\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \vec{\mathcal{P}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)$.
  $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * c$.
  $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \left(\vec{\mathcal{P}}_{i+1} * c\right)$.
  $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
  $d = \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i\right)$.
  $\vec{\mathcal{P}}_i = \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * d$.
end for
$\vec{\mathcal{X}}^* = \vec{\mathcal{X}}_i * a$

The operators **squeeze** and **twist** [11] are expressed by

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

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

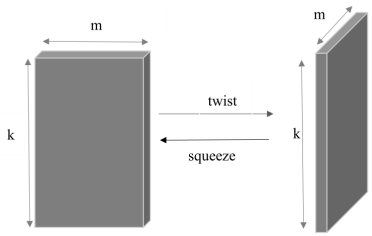


Figure 2. Twist squeeze.

Table 1. Notation Explanation.

Notation	Interpretation
A	matrix
I	identity matrix
$\mathcal{A}^T$	transpose of tensor
$\mathcal{A}^{-1}$	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
$\mathcal{I}$	identity tensor
$\|\mathcal{A}\|_F$	the Frobenius norm , 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 $\mathcal{A}$
$\mathcal{A}_{(:,:,k)}$	the kth 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 initiates the discussion on the automated determination of an appropriate regularization parameter for the tensor Conjugate Gradient (tCG) method. 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 regularization parameter in the t-CG method was not discussed and was user-specified. This subsection enhances the t-CG method by utilizing the discrepancy principle for the determination of an appropriate regularization parameter, operating 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 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 tensor $\mathcal{B}$ and is polluted by noise $\vec{\mathcal{E}}_j$. $\vec{\mathcal{B}}_{j,true}$ represents unknown error-free tensor. We assume 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.,

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

Algorithm 3 encapsulates the auto-tCG method designed for solving Equation (9). The initial tensor of Algorithm 3 is set as a zero tensor. The iteration concludes 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 ith iterative solution $\vec{\mathcal{X}}_{j,\mu_k}^i$ of the normal equation with μ_k of the jth independent subproblem. We 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 3 The auto-tCG method for sloving (9).

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}, \delta_j, j = 1, \ldots, p, \mu_0, \eta > 1$.
Output: Approximate solution $\mathcal{X}^*$ of Problem (9).
for $j = 1, 2, \ldots p$ **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., $\mu_k = \mu_0 q^k$
    $[\vec{\mathcal{R}}_0, a] \leftarrow \text{Normalize}(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_{int})$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
    $i = 0, \sigma > tol$.
    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 μ_k of the jth 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 [33] proposed a good condition that can roughly judge some inappropriate value of μ. We introduce this condition to improve Algorithm 3 by excluding some unsuitable value of μ, and present a truncated tensor Conjugate Gradient method for solving (9). We first provide the following results:

Theorem 1. *Given $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, we 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 $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$. We 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 $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$, we set $\vec{\mathcal{Z}} = \vec{\mathcal{X}}_\mu^* - \vec{\mathcal{X}}$. We let the singular value decomposition of $\mathcal{A}$ be $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$; then,

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

We 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,

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

We 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. We let the singular value decomposition of $\mathbf{S}$ be $\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r), r \leq \min\{nl, nm\}$, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r]$ 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 equation $s^2 = (s + \mu s^{-1})^{-2}(s^2 + \mu)^2$ with estimate

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

we have

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

We note that

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

It results from (19) and (20) that

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

We note that $\|\mathcal{A}*\vec{\mathcal{Z}}\|_F^2=\|\mathbf{S}\mathbf{d}\|_2^2$ 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

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

Thus,

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

We note that

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

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 apply Theorem 1 to predict in advance whether the exact solution $\vec{\mathcal{X}}_{\mu_k}^*$ satisfies the discrepancy principle in Algorithm 3. We add 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 3. If the ith 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 we 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 4 summarizes the auto-ttCG method.

3.3. A Preconditioned Truncated Tensor Conjugate Gradient Method

In this section, we explore the acceleration of Algorithm 4 through preconditioning. When tensor $\mathcal{M}$ is symmetric positive definite under the t-product structure, we can obtain its tensor approximate Cholesky decomposition (tChol) by Algorithm 5.

In Algorithm 4, coefficient tensor $\mathcal{A}^T*\mathcal{A}+\mu_k\mathcal{I}$ of kth 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 5 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 normal Equation (25) is preconditioned by $\mathcal{M}$, we solve preconditioned normal equations

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

instead of Equation (25) in Algorithm 4, where $\tilde{\mathcal{A}} = \mathcal{H}^{-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \mathcal{H}^{-T}$, $\tilde{\vec{\mathcal{X}}} = \mathcal{H}^T * \vec{\mathcal{X}}$, $\tilde{\vec{\mathcal{B}}} = \mathcal{H}^{-1} * \mathcal{A}^T * \vec{\mathcal{B}}$.

Algorithm 4 The auto-ttCG method for sloving (9)

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}$, $\vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}$, δ_j, $j = 1, \ldots, p$, μ_0, $\eta > 1$, tol.
Output: Approximate solution $\mathcal{X}^*$ of Problem (9).
for $j = 1, 2, \ldots p$ **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**
    $k = k + 1$, $(\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_j = \mathcal{A}^T * \vec{\mathcal{B}}_j$, e.g., $\mu_k = \mu_0 q^k$.
    $[\vec{\mathcal{R}}_0, a] \leftarrow \text{Normalize}(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{X}}_{int})$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
    $i = 0$, $\sigma = 10$ tol, $\vec{\mathcal{X}}^0_{j,\mu_k} = \vec{\mathcal{X}}_{int}$.
    while $\sigma > tol$ and $\|\mathcal{A} * \vec{\mathcal{X}}^i_{j,\mu_k} - \vec{\mathcal{B}}\|_F^2 - \frac{1}{4\mu_k}\|\vec{\mathcal{R}}_i * a\|_F^2 < \eta^2 \delta^2$ **do**
      $i = i + 1$.
      $c = \left(\vec{\mathcal{P}}_{i-1}^T * (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \vec{\mathcal{P}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)$.
      $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * c$, $\vec{\mathcal{X}}^i_{j,\mu_k} = \vec{\mathcal{X}}_i * a$.
      $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - (\mathcal{A}^T * \mathcal{A} + \mu_k \mathcal{I}) * \left(\vec{\mathcal{P}}_{i+1} * c\right)$
      $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
      $d = \left(\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1}\right)^{-1} * \left(\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i\right)$.
      $\vec{\mathcal{P}}_i = \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * d$.
    end while
    $\vec{\mathcal{X}}_{int} = \vec{\mathcal{X}}^i_{\mu_k}$.
  end while
  $\mathcal{X}^*_{(:,j,:)} = \vec{\mathcal{X}}^i_{j,\mu_k}$.
end for

Algorithm 5 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** $j = 1, 2, \ldots, n$ **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).

We apply Algorithm 4 to solve (26) instead of (25). We let $\vec{\mathcal{X}}_i$ and $\tilde{\vec{\mathcal{X}}}_i$ denote the solution of (25) and (26), respectively. Then, we have

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

We 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

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

and

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

In addition, we have iteration

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

and

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

together with

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

Implementing Preprocessing procedure (27)–(35) into Algorithm 4, 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 abbreviated as auto-ttpCG. Algorithm 6 summarizes the auto-ttpCG method. Numerical experiments in Section show Algorighm 6 converges faster than Algorithm 4.

Algorithm 6 The auto-ttpCG method for sloving (9)

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \vec{\mathcal{B}}_j \in \mathbb{R}^{m\times 1\times n}$, δ_j, $j = 1,\ldots,p$, μ_0, $\eta > 1$, tol.
Output: Approximate solution $\mathcal{X}^*$ of Problem (9).
for $j = 1, 2, \ldots p$ **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**
    $k = k+1$, $\mu_k = \mu_0 q^k$.
    $\mathcal{H} = tChol(\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I})$.
    $[\vec{\mathcal{R}}_0, a] \leftarrow \text{Normalize}(\mathcal{A}^T * \vec{\mathcal{B}}_j - (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{X}}_{int})$.
    $\vec{\mathcal{W}}_0 = \mathcal{H}^{-1} * \vec{\mathcal{R}}_0$, $\vec{\mathcal{P}}_0 = \mathcal{H}^{-T} * \vec{\mathcal{W}}_0$.
    $i = 0$, $\sigma = 10$ tol, $\vec{\mathcal{X}}^0_{j,\mu_k} = \vec{\mathcal{X}}_{int}$.
    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**
      $i = i+1$.
      $\tilde{c} = (\vec{\mathcal{P}}^T_{i-1} * (\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}) * \vec{\mathcal{P}}_{i-1})^{-1} * \vec{\mathcal{W}}^T_{i-1} * \vec{\mathcal{W}}_{i-1}$.
      $\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * \tilde{c}$, $\vec{\mathcal{X}}^i_{j,\mu_k} = \vec{\mathcal{X}}_i * a$.
      $\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - \left(\mathcal{A}^T * \mathcal{A} + \mu_k\mathcal{I}\right) * \vec{\mathcal{P}}_{i+1} * \tilde{c}$, $\vec{\mathcal{W}}_i = \mathcal{H}^{-1} * \vec{\mathcal{R}}_i$
      $\sigma = |\|\vec{\mathcal{R}}_i\|_F - \|\vec{\mathcal{R}}_{i-1}\|_F|$.
      $\tilde{d} = (\vec{\mathcal{W}}^T_{i-1} * \vec{\mathcal{W}}_{i-1})^{-1} * (\vec{\mathcal{W}}^T_i * \vec{\mathcal{W}}_i)$.
      $\vec{\mathcal{P}}_i = \mathcal{H}^{-T} * \vec{\mathcal{W}}_i + \vec{\mathcal{P}}_{i-1} * \tilde{d}$.
    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 illustrative examples showcasing the application of Algorithms 3, 4 and 6 in the context of image and video restoration. All computations are executed using MATLAB R2018a on computing platforms equipped with Intel Core i7 processors and 16 GB of RAM.

We suppose $\mathcal{X}_k$ is the kth approximate solution to 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 $\mathcal{X}_{true}$ represents 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. We let $\vec{\mathcal{E}}_j$ be the jth transverse slice of $\mathcal{E}$, whose entries are scaled and normally distributed with a mean of zero, i.e.,

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

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

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

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

with $N = 256$*,* $\sigma = 4$ *and* $band = 12$*. The condition numbers of* $\mathcal{A}_{(i)}$ *are* $cond(\mathcal{A}_{(:,:,1)}) = cond(\mathcal{A}_{(:,:,246)}) = \cdots = cond(\mathcal{A}_{(:,:,256)}) = 11.1559$*, while he condition numbers of the remaining slices are infinite. We 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 images characterized by blurring and noise are generated through the mathematical expression* $\vec{\mathcal{B}} = \mathcal{A} * \vec{\mathcal{X}}_{true} + \vec{\mathcal{E}}$*.*

The auto-tCG, auto-ttCG and auto-ttpCG methods are used to solve tensor discrete linear ill-posed Problems (1). The discrepancy principle is utilized to ascertain an appropriate regularization parameter and set $\mu_k = \mu_0 q^k$, $\mu_0 = \|\mathcal{A}\|_F$, $q = \frac{1}{2}$. 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 tCG, auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$ corresponding in Table 2. The iteration process is terminated when the discrepancy principle is satisfied. From Figure 3a, 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 Condition (24). Figure 3b shows that the auto-ttpCG method converges fastest among three methods.

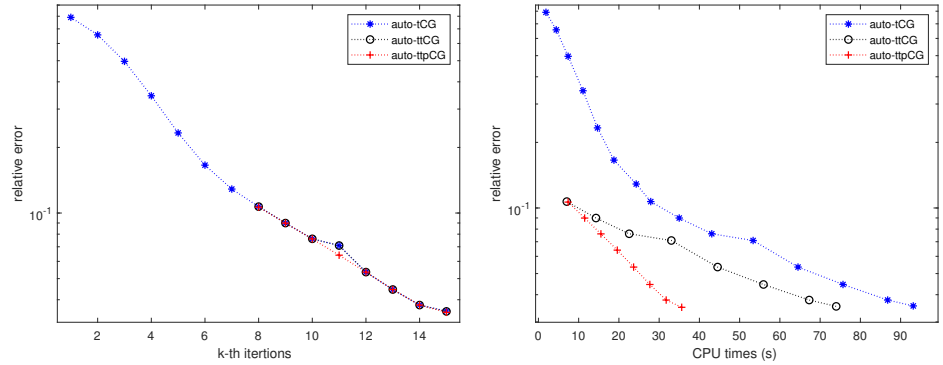


Figure 3. Example 1: Comparison of convergence between (**a**) relative errors verus the iteration number k and (**b**) relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$.

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 tCG, A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT, auto-tCG, auto-ttCG and auto-ttpCG methods with different noise levels $\nu = 10^{-i}, i = 2, 3$. The determination of the regularization parameter for the tCG method involved conducting several experiments to obtain a more appropriate value. The CPU time represents only the usage in a single CG process.

Table 2. Example 1: Comparison of relative error, SNR, and CPU time between the tCG with $\mu = 1 \times 10^{-3}$, the A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT, 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 (s)
10^{-3}	tCG	-	1×10^{-3}	9.14×10^{-2}	12.21	9.87
	A-tCG-FFT	-	-	5.44×10^{-2}	18.63	88.02
	A-tCGLS-FFT	-	-	5.44×10^{-2}	18.63	82.23
	A-tPCG-FFT	-	-	5.42×10^{-2}	18.67	76.09
	auto-tCG	15	1.96×10^{-5}	3.54×10^{-2}	22.36	109.87
	auto-ttCG	15	1.96×10^{-5}	3.52×10^{-2}	22.41	80.93
	auto-ttpCG	15	1.96×10^{-5}	3.49×10^{-2}	22.48	33.98
10^{-2}	tCG	-	1×10^{-3}	1.17×10^{-1}	11.97	9.64
	A-tCG-FFT	-	-	1.04×10^{-2}	12.75	79.33
	A-tCGLS-FFT	-	-	1.04×10^{-2}	12.75	72.29
	A-tPCG-FFT	-	-	9.81×10^{-2}	12.90	61.75
	auto-tCG	11	3.14×10^{-4}	8.74×10^{-2}	14.51	81.94
	auto-ttCG	11	3.14×10^{-4}	8.64×10^{-2}	14.61	26.42
	auto-ttpCG	11	3.14×10^{-4}	8.54×10^{-2}	14.72	18.50

The image restoration experiment of Example 1, the A-tCG-FFT, A-CGLS-FFT, and A-tpCG-FFT methods proposed by Song et al. [31] project the t-product into the Fourier domain and solve 256 ill-posed problems in matrix form, respectively. In Song et al.'s setting, when the frontal slice is small, the time required is very small. In the setting of our article, the number of frontal slices is related to the size of the image. As the number of frontal slices increases, the time cost increases. The calculation process of auto-tCG, auto-ttCG and auto-ttpCG methods always maintains the tensor t-product structure, resulting in higher quality image restoration in the end. The quality of the regularization solution obtained by the auto-tCG surpasses that of the solution obtained by the tCG method. 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.

Figure 4 shows the reconstructed images obtained by using the tCG, A-tCG-FFT, A-CGLS-FFT, 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 with the least CPU time. The image restoration performance of the tCG method is inferior to three conjugate gradient methods with automatically determined parameters.

Example 2 (Color image)**.** *This example illustrates the restoration of a blurred* **Lena** *color image using Algorithms 3, 4 and 6. 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 obtain* $\mathcal{A} \in \mathbb{R}^{256\times256\times256}$ *by*

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

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

Then, $cond(\mathcal{A}_{(:,:,1)}) = \cdots = cond(\mathcal{A}_{(:,:,12)}) = 4.68e + 07$, *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 5b.*

Figure 4. Example 1: (**a**) The original image and (**b**) the blurred and noised image, reconstructed images by (**c**) the tCG method (SNR = 12.21, CPU = 9.87), (**d**) the A-tCG-FFT method (SNR = 18.63, CPU = 88.02), (**e**) the A-CGLS-FFT method (SNR = 18.63, CPU = 82.23), (**f**) the auto-tCG method (SNR = 22.36, CPU = 109.87), (**g**) the auto-ttCG method (SNR = 22.41, CPU = 80.93) and (**h**) the auto-ttpCG method (SNR = 22.48, CPU = 33.98) according to the noise level $\nu = 10^{-3}$ in Table 2.

Figure 5. Example 2: (**a**) The original image *Lena*, (**b**) the blurred and noised image and reconstructed images by (**c**) the tCG method, (**d**) the A-tCG-FFT method, (**e**) the A-CGLS-FFT method, (**f**) the auto-tCG method, (**g**) the auto-ttCG and (**h**) the auto-ttpCG method according to the noise level $\nu = 10^{-3}$ in Table 3.

We set color image $\mathcal{B}$ to be divided into multiple lateral slices and independently process each slice through (1) by using the tCG, auto-tCG, auto-ttCG and auto-ttpCG methods. Figure 6 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 1 from Figure 6. We can see that the auto-ttCG and auto-ttpCG methods need less iterations than the auto-tCG method from Figure 6a and the auto-ttpCG method converges fastest among all methods from Figure 6b.

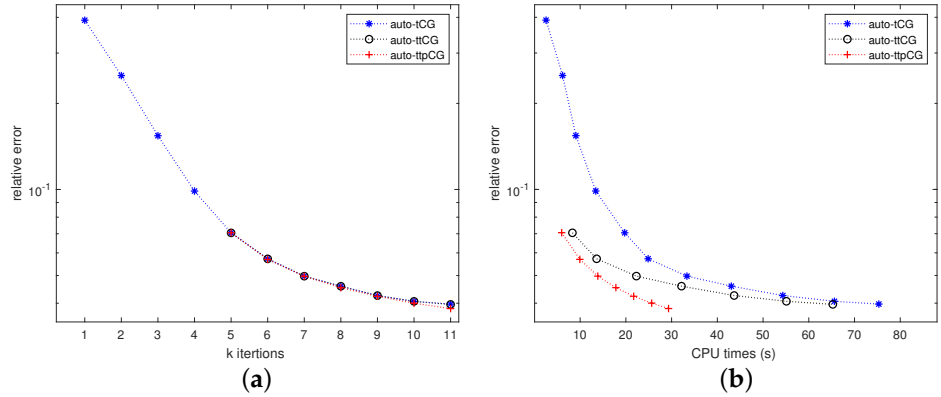


Figure 6. Example 2: Comparison of convergence between (**a**) relative errors verus the iteration number k and (**b**) relative errors verus the CPU time for the auto-tCG, auto-ttCG and auto-ttpCG methods with the noise level $\nu = 10^{-3}$.

Table 3 lists the relative error, SNR and the CPU time of the optimal solution obtained by using the tCG, A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT, 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 levels. In the application of the tCG method, we define distinct regularization parameters, specifically setting $\mu = 0.01$ when $\nu = 10^{-2}$, and $\mu = 0.005$ when $\nu = 10^{-3}$. The regularization parameter set for the tCG method has been determined through multiple iterations, yielding a reasonably suitable value. However, when applying tCG to solve Problem (9) corresponding to other regularization parameters attempted during this process, divergence or excessively large relative errors are commonly observed. When the condition number for frontal slicing is larger, the condition number for the matrix projected into the Fourier domain also increases, which leads to increased ill-posedness and results in more CPU time for the A-tCG-FFT, A-CGLS-FFT and A-tpCG-FFT methods to obtain regularization parameters. The quality of the solutions obtained by the tCG method is inferior to that of the other three versions with automatic parameter tuning. However, the CPU time used by the tCG method is shorter than the other three methods. This is because it only represents the time spent solving a regularized equation, whereas the process of manually selecting regularization parameters would consume more time. Table 3 also reflects the advantages of both truncation parameters and preprocessing operations.

Figure 5 shows the recovered images by the tCG, A-tCG-FFT, A-CGLS-FFT, 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 5.

Example 3 (Video)**.** *In this example, we employ three distinct reconstruction methods on MATLAB to recover the initial 10 consecutive frames of the blurred and noisy* **Rhinos** *video, with each frame containing* 240×240 *pixels. We store ten frames devoid of pollution and noise from the original video in the tensor* $\mathcal{X}_{true} \in \mathbb{R}^{240\times 10\times 240}$*. We let* z *be defined by* (37) *with* $N = 240$, $\sigma = 2$ *and* $band = 12$*. The coefficient tensor* $\mathcal{A}$ *is defined as follows:*

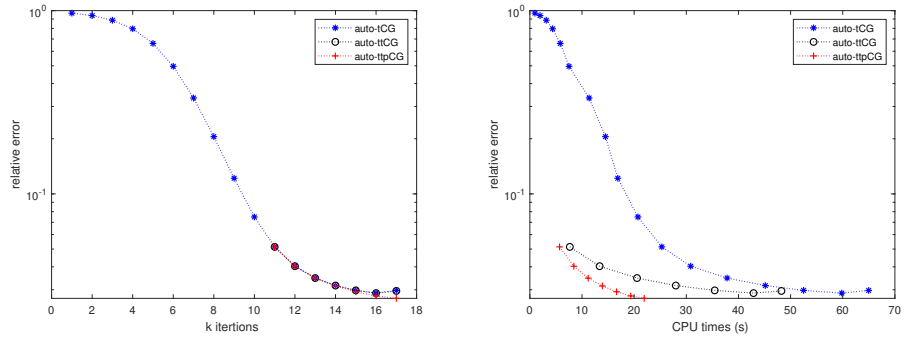
$$A = \frac{1}{\sqrt{2\pi\sigma}} toeplitz(z), \mathcal{A}_{(:,:,i)} = A(i,1)A, i = 1, \ldots, 240.$$

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* $\mathcal{E} \in \mathbb{R}^{120\times 30\times 120}$ *being defined by* (36)*.*

Table 3. Example 2: Comparison of relative error, SNR, and CPU time between the tCG ($\nu = 10^{-2}$: $\mu = 0.01$; $\nu = 10^{-3}$: $\mu = 0.005$), the A-tCG-FFT, A-CGLS-FFT and A-tpCG-FFT, auto-tCG, auto-ttCG and auto-ttpCG methods with different noise levels $\nu = 10^{-i}, i = 2, 3$.

Noise Level	Method	Relative Error	SNR	Time (s)
10^{-3}	tCG	8.90×10^{-2}	11.15	34.73
	A-tCG-FFT	7.34×10^{-2}	13.39	6939.36
	A-tCGLS-FFT	7.34×10^{-2}	13.39	5634.69
	A-tPCG-FFT	7.34×10^{-2}	13.39	1638.91
	auto-tCG	5.90×10^{-2}	14.62	314.73
	auto-ttCG	5.90×10^{-2}	14.62	262.81
	auto-ttpCG	5.43×10^{-2}	15.37	103.41
10^{-2}	tCG	9.64×10^{-2}	10.23	31.63
	A-tCG-FFT	8.98×10^{-2}	11.01	5236.55
	A-tCGLS-FFT	8.98×10^{-2}	11.01	4895.52
	A-tPCG-FFT	8.98×10^{-2}	11.01	1236.21
	auto-tCG	7.64×10^{-2}	12.37	117.48
	auto-ttCG	7.48×10^{-2}	12.55	62.01
	auto-ttpCG	7.01×10^{-2}	13.13	54.85

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 1.



(**a**) (**b**)

Figure 7. Example 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 tCG, the A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT, auto-tCG, auto-ttCG and auto-ttpCG methods for the second frame with different noise levels $\nu = 10^{-i}$, $i = 2, 3$. In video restoration experiments with continuous frames, using the A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT methods to perform matrix calculations in the Fourier domain may cause a certain degree of damage to the spatial structure that may exist between consecutive frames, resulting in a decrease in restoration quality. When employing the tCG method, we configured distinct regularization parameters, specifically, when $\nu = 10^{-2}$, $\mu = 0.05$; and when $\nu = 10^{-3}$, $\mu = 0.001$. With the increase in data volume, the auto-tCG method demonstrates better solution quality compared to the tCG method. Additionally, the truncation parameter operation and preprocessing strategy exhibit superior solution quality and time advantages over auto-tCG. 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 3: Comparison of relative error, SNR, and CPU time between the tCG ($\nu = 10^{-2}$: $\mu = 0.05$; $\nu = 10^{-3}$: $\mu = 0.001$), the A-tCG-FFT, A-CGLS-FFT, A-tpCG-FFT, 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 (s)
10^{-3}	tCG	3.94×10^{-2}	21.43	96.33
	A-tCG-FFT	3.67×10^{-2}	21.95	9396.36
	A-tCGLS-FFT	3.67×10^{-2}	21.95	7423.69
	A-tPCG-FFT	3.67×10^{-2}	21.95	3798.81
	auto-tCG	2.94×10^{-2}	23.17	697.78
	auto-ttCG	2.92×10^{-2}	23.23	487.35
	auto-ttpCG	2.66×10^{-2}	24.05	214.16
10^{-2}	tCG	8.31×10^{-2}	14.14	80.61
	A-tCG-FFT	7.89×10^{-2}	14.89	8972.69
	A-tCGLS-FFT	7.89×10^{-2}	14.89	7263.02
	A-tPCG-FFT	7.89×10^{-2}	14.89	3269.36
	auto-tCG	5.24×10^{-2}	18.15	480.75
	auto-ttCG	5.10×10^{-2}	18.38	281.54
	auto-ttpCG	4.74×10^{-2}	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 tCG, A-tCG-FFT, A-CGLS-FFT, A-tCG-FFT, A-CGLS-FFT, 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 3: (**a**) The second frame image of the original video, (**b**) the blurred and noisy image and recovered images by (**c**) the tCG method, (**d**) the A-tCG-FFT method, (**e**) the A-CGLS-FFT method, (**f**) the auto-tCG method, (**g**) the auto-ttCG and (**h**) the auto-ttpCG method according to the noise level $\nu = 10^{-3}$ in Table 4.

5. Conclusions

This paper introduces three tensor Conjugate Gradient methods designed for the resolution of large-scale linear discrete ill-posed problems formulated in tensor representation. Initially, we introduce an automated strategy for determining an appropriate regularization parameter for the tensor Conjugate Gradient (tCG) method. Furthermore, we develop a truncated version and a preprocessed version of the tCG method. The introduced methodologies are employed in diverse instances of image and video restoration. The efficacy of the proposed methodologies in image and video restoration applications is demonstrated through illustrative examples. These approaches circumvent the need for

problem matrixization or vectorization. Notably, these methods exhibit significant potential in terms of both speed and quality of computed restoration, as assessed by relative errors and SNR values, providing a comprehensive evaluation of algorithmic performance in image restoration.

Author Contributions: Software, S.-W.W.; Formal analysis, F.Y.; Investigation, F.Y.; Writing—original draft, S.-W.W.; Writing—review & editing, G.-X.H.; Supervision, G.-X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Sichuan Science and Technology Program (Grant No. 2022ZYD0008).

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the referees for their helpful and constructive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. 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. [CrossRef]
2. 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.
3. 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. [CrossRef]
4. 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. [CrossRef]
5. Kilmer, M.E.; Horesh, L.; Avron, H.; Newman, E. Tensor-tensor algebra for optimal representation and compression of multiway data. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2015851118. [CrossRef] [PubMed]
6. 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.* **2020**, *151*, 425–447. [CrossRef]
7. Bentbib, A.H.; Khouia, A.; Sadok, H. The LSQR method for solving tensor least-squares problems. *Electron. Trans. Numer. Anal.* **2022**, *55*, 92–111. [CrossRef]
8. 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. [CrossRef]
9. Kilmer, M.E.; Martin, C.D. Factorization strategies for third order tensors. *Linear Alg. Appl.* **2011**, *435*, 641–658. [CrossRef]
10. Hao, N.; Kilmer, M.E.; Braman, K.; Hoover, R.C. Facial recognition using tensor-tensor decompositions. *SIAM J. Imaging Sci.* **2013**, *6*, 437–463. [CrossRef]
11. 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. [CrossRef]
12. 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. [CrossRef]
13. 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. [CrossRef]
14. Zeng, C.; Ng, M.K. Decompositions of third-order tensors: HOSVD, T-SVD, and Beyond. *Numer. Linear Algebr. Appl.* **2020**, *27*, e2290. [CrossRef]
15. 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. [CrossRef]
16. Yu, Q.; Zhang, X. T-product factorization based method for matrix and tensor completion problems. *Comput. Optim. Appl.* **2023**, *84*, 761–788. [CrossRef]
17. Tikhonov, A.N.; Arsenin, V.Y. *Solutions of Ill-Posed Problems*; Transl. from Russian; V.H. Winston & Sons: Washington, DC, USA, 1977.
18. Fenu, C.; Reichel, L.; Rodriguez, G. GCV for tikhonov regularization via global Golub–Kahan decomposition. *Numer. Linear Algebr. Appl.* **2016**, *25*, 467–484. [CrossRef]
19. Hansen, P.C. Rank-deficient and Discrete Ill-posed Problems: Numerical Aspects of Linear Inversion. *SIAM J. Sci. Comput.* **1998**, *20*, 684–696.
20. 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.

21. Kindermann, S.; Raik, K. A simplified L-curve method as error estimator. *Electron. Trans. Numer. Anal.* **2020**, *53*, 217–238. [CrossRef]
22. Reichel, L.; Rodriguez, G. Old and new parameter choice rules for discrete ill-posed problems. *Numer. Algorithms* **2013**, *63*, 65–87. [CrossRef]
23. Engl, H.W.; Hanke, M.; Neubauer, A. Neubauer. In *Regularization of Inverse Problems*; Kluwer: Dordrecht, The Netherlands, 1996.
24. 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. [CrossRef]
25. 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.
26. Hestenes, M.R.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **1952**, *49*, 409–436. [CrossRef]
27. Polyak, B.T. The conjugate gradient method in extremal problems. *U.S.S.R. Comput. Math. Math. Phys.* **1969**, *9*, 94–112. [CrossRef]
28. Gilbert, J.C.; Nocedal, J. Global convergence properties of conjugate gradient methods for optimization. *SIAM J. Optim.* **1992**, *2*, 21–42. [CrossRef]
29. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: New York, NY, USA, 1999.
30. Saad, Y. *Iterative Methods for Sparse Linear Systems*; SIAM: Philadelphia, PA, USA, 2003.
31. Song, H.M.; Wang, S.W.; Huang, G.X. Tensor Conjugate-Gradient methods for tensor linear discrete ill-posed problems. *AIMS Math.* **2023**, *8*, 26782–26800. [CrossRef]
32. Lund, K. The tensor t-function: A definition for functions of third-order tensors. *Numer. Linear Algebr. Appl.* **2020**, *27*, e2288. [CrossRef]
33. Frommer, A.; Maass, P. Fast CG-based methods for Tikhonov–Phillips regularization. *SIAM J. Sci. Comput.* **1999**, *20*, 1831–1850. [CrossRef]