AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 8(11): 26782–26800.
DOI: 10.3934/math.20231371
Received: 09 July 2023
Revised: 02 September 2023
Accepted: 05 September 2023
Published: 20 September 2023

Research article

Tensor Conjugate-Gradient methods for tensor linear discrete ill-posed problems

Hong-Mei Song[1,2], **Shi-Wei Wang**[1,2] **and Guang-Xin Huang**[2,3,*]

[1] College of Mathematical and physics, Chengdu University of Technology, Chengdu, China
[2] Sichuan Geomathematics Key Laboratory, Chengdu University of Technology, Chengdu, China
[3] College of Computer Science and Cyber Security, Chengdu University of Technology, Chengdu, China

* **Correspondence:** Email: huangx@cdut.edu.cn.

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

Keywords: linear discrete ill-posed problems; Conjugate-Gradient method; t-product; regularization parameter; discrepancy principle; Tikhonov regularization
Mathematics Subject Classification: 15A69, 65F05, 65J20

1. Introduction

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

$$\min_{\mathcal{X}\in\mathbb{R}^{m\times p\times n}} \|\mathcal{A} * \mathcal{X} - \mathcal{B}\|_F, \mathcal{A} = [a]_{i,j,k=1}^{l,m,n} \in \mathbb{R}^{l\times m\times n}, \mathcal{B} \in \mathbb{R}^{l\times p\times n}, \tag{1.1}$$

where 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. Problem (1.1) with such a tensor is called tensor linear discrete ill-posed problems. They arise from the restoration of color image and video, see e.g., [1–5]. Throughout this paper, the operation $*$ represents tensor t-product introduced in [1] and

$\|\cdot\|_F$ denotes the tensor Frobenius norm, represented by

$$\|\mathcal{A}\|_F = \sqrt{\sum_{i=1}^{l}\sum_{j=1}^{m}\sum_{k=1}^{n} a_{ijk}^2}. \tag{1.2}$$

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

$$\mathcal{B} = \mathcal{B}^* + \mathcal{E}, \tag{1.3}$$

where $\mathcal{B}^* \in \mathbb{R}^{m\times p\times n}$ is an unknown and unavailable error-free tensor related to $\mathcal{B}$. $\mathcal{B}^*$ is determined by $\mathcal{A} * \mathcal{X}^* = \mathcal{B}^*$, where $\mathcal{X}^*$ represents the explicit solution of problem (1.1) that is to be found. We assume that the upper bound of the Frobenius norm of $\mathcal{E}$ is known, i.e.,

$$\|\mathcal{E}\|_F \le \delta. \tag{1.4}$$

Straightforward solution of (1.1) generally is not meaningful due to propagation and severe amplification of the error $\mathcal{E}$ into the solution of (1.1) and the ill-posedness of $\mathcal{A} = [a]_{i,j,k=1}^{l,m,n}$. In this paper we use Tikhonov regularization to reduce this effect and this regularization replaces (1.1) with penalty least-squares problems of the form

$$\min_{\mathcal{X}\in\mathbb{R}^{m\times p\times n}} \left\{\|\mathcal{A} * \mathcal{X} - \mathcal{B}\|_F^2 + \mu\|\mathcal{X}\|_F^2\right\}, \tag{1.5}$$

where $\mu > 0$ is a regularization parameter. We assume that

$$\mathcal{N}(\mathcal{A}) \cap \mathcal{N}(\mathcal{I}) = \{\mathcal{O}\}, \tag{1.6}$$

where $\mathcal{N}(\mathcal{A})$ denotes the null space of the tensor $\mathcal{A}$ under $*$, is the set of all solutions $\mathcal{X}$ of the equation $\mathcal{A} * \mathcal{X} = \mathcal{O}$. $\mathcal{I}$ is the identity tensor and $\mathcal{O} \in \mathbb{R}^{m\times p\times n}$ is a tensor whose elements are all zero, respectively.

The normal equation of the minimization problem (1.5) is

$$(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}) * \mathcal{X} = \mathcal{A}^T * \mathcal{B}, \tag{1.7}$$

then

$$\mathcal{X}_\mu = \left(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I}\right)^{-1} * \mathcal{A}^T * \mathcal{B} \tag{1.8}$$

is the unique solution of the Tikhonov minimization problem (1.5) under the assumption (1.6).

There are many methods for solving large-scale tensor linear discrete ill-posed problems (1.1). Recently, a tensor Golub–Kahan bidiagonalization method [4] and a GMRES method [5] were introduced for solving large-scale linear ill-posed problems (1.1) by iteratively solving (1.5). The randomized tensor singular value decomposition (rt-SVD) method in [6] was presented for computing super large data sets, and has prospects in image data compression and analysis. Ugwu and Reichel [7] 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 method (tCG) for tensor linear systems $\mathcal{A} * \mathcal{X} = \mathcal{B}$ corresponding to the least-squares problems (1.1), where the regularization parameter in the tCG method is user-specified.

This paper mainly extends CG methods from matrix problems to tensor problems. Using matrix methods to solve the problem of image restoration and denoising is to flatten the three-dimensional

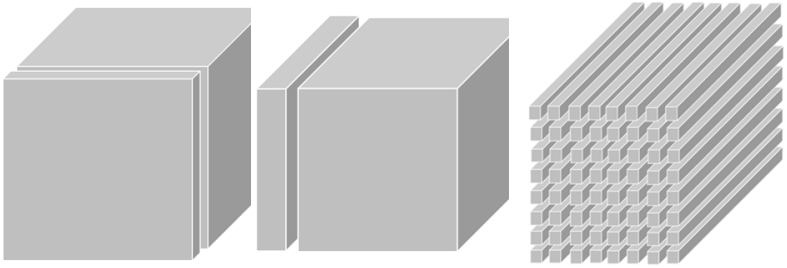
data of color images into matrix form in turn for processing. Based on t-product, we extend the matrix method to tensor, which can preserve the data correlation between three-dimensional tensor sections. The tensor problem is projected into the Fourier domain, which makes use of the particularity of t-product structure. We further discuss the tCG method for the approximate solution of (1.1) in the Fourier domain. The discrepancy principle is used to determine a suitable regularization parameter of the tCG method. The proposed automatic determination strategy is called the tCG method with automatic determination of regularization parameters (A-tCG-FFT). A least-squares method based on the tCG method is provided for (1.1), which is called A-tCGLS-FFT. A preconditioned version of the A-tCG-FFT method is presented, which is abbreviated as A-tPCG-FFT. The Conjugate Gradient method only needs to save the current and last gradient values, which takes up less memory resources. Moreover, only the previously calculated gradient information is needed in each iteration process, so parallel calculation can be carried out and the calculation efficiency can be improved. Moreover, the way of projecting tensor problem into Fourier domain also greatly avoids the time and space complexity required to smooth tensor problem into matrix problem.

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 A-tCG-FFT, A-tCGLS-FFT and A-tPCG-FFT methods for solving the minimization problem (1.5). 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 $\mathcal{A}(i,j,:)$, and we abbreviate $A_k = \mathcal{A}(:,:,k)$ for simplicity. 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.,

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



(a) (b) (c)

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

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.,*

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

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*

$$\mathcal{A} * \mathcal{B} = \mathbf{fold}(\mathbf{bcirc}(\mathcal{A})\mathbf{unfold}(\mathcal{B})) = C, \tag{2.1}$$

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

The following remarks will be used in Section 3.

Remark 2.1. *[8] 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.,

$$\mathbf{bdiag}\left(\hat{\mathcal{A}}\right) = \begin{bmatrix} \hat{A}_1 & & & \\ & \hat{A}_2 & & \\ & & \ddots & \\ & & & \hat{A}_n \end{bmatrix} = (F_n \otimes I_l)\,\mathbf{bcirc}\,(\mathcal{A})\left(F_n^H \otimes I_m\right), \tag{2.2}$$

where $\otimes$ is the Kronecker product, F_n^H is the conjugate transposition of F_n and $\hat{A}_i$ denotes the frontal slices of $\hat{\mathcal{A}}$.

We also need the following remark.

Remark 2.2. *[9] For appropriately sized tensors $\mathcal{A}$ and $\mathcal{B}$,*
(1) $\mathbf{bdiag}(\widehat{\mathcal{A} * \mathcal{B}}) = \mathbf{bdiag}(\hat{\mathcal{A}})\mathbf{bdiag}(\hat{\mathcal{B}})$.
(2) $\mathbf{bdiag}(\widehat{\mathcal{A} + \mathcal{B}}) = \mathbf{bdiag}(\hat{\mathcal{A}} + \hat{\mathcal{B}})$.
(3) $\mathbf{bdiag}(\hat{\mathcal{A}}^H) = \mathbf{bdiag}(\hat{\mathcal{A}})^H$. *Additionally, if* $\mathbf{bdiag}(\mathcal{A})$ *is symmetric,* $\mathbf{bdiag}(\hat{\mathcal{A}})$ *is also symmetric.*
(4) $\mathbf{bdiag}(\hat{\mathcal{A}^{-1}}) = \mathbf{bdiag}(\hat{\mathcal{A}})^{-1}$.

Then the t-product of $\mathcal{A}$ and $\mathcal{B}$ in (2.1) can be expressed by

$$\mathcal{A} * \mathcal{B} = \textbf{fold}\left(\left(F_n^* \otimes I_l\right)\left(\left(F_n \otimes I_l\right)\textbf{bcirc}\left(\mathcal{A}\right)\left(F_n^* \otimes I_m\right)\right)\left(F_n \otimes I_m\right)\textbf{unfold}\left(\mathcal{B}\right)\right), \tag{2.3}$$

and (2.1) is reformulated as

$$\begin{pmatrix} \hat{A}_1 & & & \\ & \hat{A}_2 & & \\ & & \ddots & \\ & & & \hat{A}_n \end{pmatrix}\begin{pmatrix} \hat{B}_1 \\ \hat{B}_2 \\ \vdots \\ \hat{B}_n \end{pmatrix} = \begin{pmatrix} \hat{C}_1 \\ \hat{C}_2 \\ \vdots \\ \hat{C}_n \end{pmatrix}. \tag{2.4}$$

It is easy to implement (2.1) in MATLAB. Using MATLAB notation, let $\hat{\mathcal{M}}$=fft($\mathcal{M}$, [ ], 3) be the tensor obtained by applying the FFT along the third dimension. Then (2.1) can be computed by taking the FFT along tube of $\mathcal{A}$ and $\mathcal{B}$ to obtain $\hat{\mathcal{A}}$=fft($\mathcal{A}$, [ ], 3) and $\hat{\mathcal{B}}$=fft($\mathcal{B}$, [ ], 3). Then for the matrix-matrix product of each pair of front slices of $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$, there is

$$\hat{C}(:,:,i) = \hat{A}(:,:,i)\hat{B}(:,:,i), \quad i = 1, 2, \ldots, n,$$

and then taking the inverse FFT along the third dimension to obtain C=ifft($\hat{C}$, [ ], 3). We refer to Table 1 for more notations.

Table 1. Description of notations.

Notation	Interpretation
$\mathcal{A}$	tensor
$\mathcal{A}^T$	transpose of tensors
$\mathcal{A}^{-1}$	inverse of tensor, and $\mathcal{A}^{-T} = (\mathcal{A}^{-1})^T = (\mathcal{A}^T)^{-1}$
$\mathcal{A}(:,:,k)$	the k-th frontal slice of tensor $\mathcal{A}$
$\mathcal{A}(i,j,:)$	the tube fibers of tensor $\mathcal{A}$
A_k	$\mathcal{A}(:,:,k)$
$\hat{\mathcal{A}}$	FFT of $\mathcal{A}$ along the third mode
$\hat{\mathcal{A}}^H$	transpose of complex tensor $\hat{\mathcal{A}}$
unfold($\mathcal{A}$)	the block column matrix of $\mathcal{A}$
bcirc($\mathcal{A}$)	the block-circulant matrix
$\mathcal{I}$	identity tensor
A	matrix
I	identity matrix
$\|\mathcal{A}\|_F$	the Frobenius-Norm of tensor $\mathcal{A}$
$\|A\|$	the 2-Norm of matrix A
$\langle A, B\rangle$	the matrix inner product $\langle A, B\rangle = tr(A^T B)$
$\mathbf{A}$	$\hat{A}_k^H \hat{A}_k + \mu_j I$
$\hat{A}_k$	$\hat{\mathcal{A}}(:,:,k)$
$\hat{A}_k^H$	transpose of complex matrix $\hat{A}_k$
$*$	t-product

3. Tensor Conjugate-Gradient methods

3.1. The A-tCG-FFT method

Based on the tensor conjugate gradient (tCG) method proposed by Kilmer et al. [10], this section presents a method to solve the regularization (1.5) by using the CG process in Fourier domain, where an suitable regularization parameter is automatically determined by the discrepancy principle. This method is abbreviated as A-tCG-FFT. The A-tCG-FFT improves the tCG method presented in [2] where the regularization parameter was user-specified. The following result shows the equivalent equation of (1.5) in the Fourier domain.

Theorem 3.1. *Let* $\hat{\mathcal{A}}$=*fft*($\mathcal{A}$,[ ],3), $\hat{\mathcal{B}}$=*fft*($\mathcal{B}$,[ ],3) *and* $\hat{\mathcal{X}}$=*fft*($\mathcal{X}$,[ ],3). *Then, solving the tensor regularization* (1.5) *is equivalent to solving a sequence of regularized least-squares problems of the matrix form*

$$\min_{\hat{X}_k} \left\{ \|\hat{A}_k\hat{X}_k - \hat{B}_k\|^2 + \mu\|\hat{X}_k\|^2 \right\}, k = 1, 2, \cdots, n, \tag{3.1}$$

where $\|\cdot\|$ *represents the 2-norm of the matrix, and* $\hat{X}_k = \hat{\mathcal{X}}(:,:,k)$, $\hat{A}_k = \hat{\mathcal{A}}(:,:,k)$ *and* $\hat{B}_k = \hat{\mathcal{B}}(:,:,k)$.

The normal equation of (3.1) *can be represented as*

$$(\hat{A}_k^H\hat{A}_k + \mu I)\hat{X}_k = \hat{A}_k^H\hat{B}_k, \tag{3.2}$$

Once getting $\hat{\mathcal{X}}$, *we have the approximation solution of* (1.1) *with the form* $\mathcal{X}$=*iff*($\hat{\mathcal{X}}$,[ ],3).

Proof. Following Remarks 2.1 and 2.2, (1.8) is represented as

$$\begin{aligned}
\mathbf{unfold}(\hat{\mathcal{X}}) &= (F_n \otimes I_l)\mathbf{unfold}(\mathcal{X}) \\
&= (F_n \otimes I_l)\mathbf{bcirc}(\mathcal{A}^T * \mathcal{A} + \mu\mathcal{I})^{-1}\mathbf{unfold}(\mathcal{A}^T * \mathcal{B}) \\
&= \mathbf{bdiag}(\mathcal{A}^T * \widehat{\mathcal{A}} + \mu\mathcal{I})^{-1}(F_n \otimes I_l)\mathbf{unfold}(\mathcal{A}^T * \mathcal{B}) \\
&= \mathbf{bdiag}(\mathcal{A}^T * \widehat{\mathcal{A}} + \mu\mathcal{I})^{-1}(F_n \otimes I_l)\mathbf{bcirc}(\mathcal{A}^T)\mathbf{unfold}(\mathcal{B}) \\
&= \mathbf{bdiag}(\mathcal{A}^T * \widehat{\mathcal{A}} + \mu\mathcal{I})^{-1}\mathbf{bdiag}(\hat{\mathcal{A}}^H)\mathbf{unfold}(\hat{\mathcal{B}}) \\
&= \mathbf{bdiag}(\hat{\mathcal{A}}^H * \hat{\mathcal{A}} + \mu\mathcal{I})^{-1}\mathbf{bdiag}(\hat{\mathcal{A}}^H)\mathbf{unfold}(\hat{\mathcal{B}}) \\
&= \mathbf{bdiag}((\hat{\mathcal{A}}^H * \hat{\mathcal{A}} + \mu\mathcal{I})^{-1}\hat{\mathcal{A}}^H)\mathbf{unfold}(\hat{\mathcal{B}}).
\end{aligned}$$

Thus, by using (2.4) we have

$$\hat{X}_k = [\hat{A}_k^H\hat{A}_k + \mu I]^{-1}\hat{A}_k^H\hat{B}_k,$$

which implies (3.2).

Now we discuss the determination of a suitable regularization parameter for (3.1). Let $\hat{\mathcal{B}}^*$=fft($\mathcal{B}^*$,[ ],3) and $\hat{\mathcal{B}}$=fft($\mathcal{B}$,[ ],3), and denote $\hat{B}_k^* = \hat{B}^*(:,:,k)$ and $\hat{B}_k = \hat{\mathcal{B}}(:,:,k)$. The assumption in (1.4) implies that

$$\|\hat{E}_k\| \le \delta_k, \tag{3.3}$$

where $\hat{E}_k = \hat{B}_k - \hat{B}_k^*$. The availability of the bound (3.3) allows us to determine μ by the discrepancy principle. Especially, the solution $\hat{X}_k$ of (3.1) satisfies

$$\|\hat{A}_k\hat{X}_k - \hat{B}_k\| \le \eta\delta_k, \tag{3.4}$$

where $\eta > 1$ is usually a user-specified constant and is independent of δ_k. For more details on the discrepancy principle, see e.g., [11].

In [12, 13], the method of selecting regularization parameter through an a-priori strategy and a-posteriori parameter choice rule is mentioned to verify the convergence estimation between exact solution and regularization solution. Here, we choose regularization parameter to be obtained by automatically updating the polynomial function

$$\mu_j = \mu_0 \rho^j, j = 0, 1, 2, \cdots, \tag{3.5}$$

where $0 < \rho < 1$ and $\mu_0 = \|\hat{\mathcal{A}}\|$. Then, we obtain a suitable regularization parameter by iteratively applying (3.5) until (3.4) is satisfied.

Algorithm 1 summarizes the A-tCG-FFT method for solving (1.5). Algorithm 1 contains two nested iterations. The outer iteration updates μ by applying (3.5) so that the discrepancy principle is satisfied. The inner iteration is to use the CG method (CG-iteration) to solve the k-th normal equation (3.2) with the value of μ_j determined by the outer iteration, and $\langle M, N\rangle$ represents the inner product between matrices M and N. The inner iteration is stopped when the norm of the residual

$$r_{k,\mu_j,i} = \hat{A}_k^H \hat{B}_k - \left(\hat{A}_k^H \hat{A}_k + \mu_j I\right) X_{k,\mu_j,i} \tag{3.6}$$

of the i-th iterate $X_{k,\mu_j,i}$ is less than a tolerance tol.

Algorithm 1: The A-tCG-FFT method for sloving (1.5)

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \mathcal{B} \in \mathbb{R}^{m\times m\times n}, \delta_1, ..., \delta_n, \mu_0, \rho, \eta > 1$.
Output: Least-square solution $\mathcal{X}$ of (1.5).
$\hat{\mathcal{A}}$=fft($\mathcal{A}$, [ ], 3).
$\hat{\mathcal{B}}$=fft($\mathcal{B}$, [ ], 3).
for $k = 1, ..., n$ **do**
  Let $j = 0, X_{k,\mu_0} = 0$.
  while $\|\hat{A}_k X_{k,\mu_j} - \hat{B}_k\| \geq \eta\delta_k$ **do**
    $j = j + 1$, $\mathbf{A} = \hat{A}_k^H \hat{A}_k + \mu_j I$, $\mu_j = \mu_0 \rho^j$.
    $X_0 = 0$ *or* $X_0 = X_{k,\mu_{j-1}}$; $R_0 = \hat{A}_k^H \hat{B}_k$-$\mathbf{A}X_0$;$P_0 = R_0$.
    for $i = 0, 1, ...,$until convergence **do**
      $\upsilon_i = \langle R_i, R_i\rangle / \langle \mathbf{A}P_i, P_i\rangle$.
      $X_{i+1} = X_i + \upsilon_i P_i$.
      $R_{i+1} = R_i - \upsilon_i \mathbf{A} P_i$.
      $\omega_i = \langle R_{i+1}, R_{i+1}\rangle \,/\, \langle R_i, R_i\rangle$.
      $P_{i+1} = R_{i+1} + \omega_i P_i$.
    end for
    $X_{k,u_j} = X_{i+1}$.
  end while
  $\hat{\mathcal{X}}(:, :, k) = X_{k,u_j}$.
end for
$\mathcal{X}$=ifft($\hat{\mathcal{X}}$, [ ], 3).

In the following Corollay 3.1, the rationality of the two methods for initial X_0 is expounded.

Corollary 3.1. *Let $X_0 = 0$ be the initial solution of the inner iteration of Algorithm 1, then we have the initial residual*

$$\|R_0\| = \left\|\hat{A}_k^H \hat{B}_k\right\| \leq \left\|\hat{A}_k^H\right\| \left\|\hat{B}_k\right\|.$$

Otherwise, if $X_0 = X_{k,\mu_{j-1}}$, then we have

$$\|R_0\| \leq \left\|(I - \frac{\mu_j}{\mu_{j-1}} I)\right\| \left\|\hat{A}_k^H \hat{B}_k\right\| + tol.$$

Proof. Denote

$$r_{k,\mu_j,i} = \hat{A}_k^H \hat{B}_k - \left(\hat{A}_k^H \hat{A}_k + \mu_j I\right) X_{k,\mu_j,i}.$$

It is easy to see that

$$\|R_0\| = \hat{A}_k^H \hat{B}_k - \left(\hat{A}_k^H \hat{A}_k + \mu_j I\right) X_0 = \left\|\hat{A}_k^H \hat{B}_k\right\| \leq \left\|\hat{A}_k^H\right\| \left\|\hat{B}_k\right\|$$

for $X_0 = 0$. Let $X_0 = X_{k,\mu_{j-1}}$, then we have

$$\begin{aligned}
\|R_0\| &= \left\|\hat{A}_k^H \hat{B}_k - (\hat{A}_k^H \hat{A}_k + \mu_j I) X_{k,\mu_{j-1}}\right\| \\
&= \left\|\hat{A}_k^H \hat{B}_k - (\hat{A}_k^H \hat{A}_k + \mu_j I)(\hat{A}_k^H \hat{A}_k + \mu_{j-1} I)^{-1} \hat{A}_k^H \hat{B}_k\right\| \\
&= \left\|\left[I - (\hat{A}_k^H \hat{A}_k + \mu_j I)(\hat{A}_k^H \hat{A}_k + \mu_{j-1} I)^{-1}\right] \hat{A}_k^H \hat{B}_k\right\| \\
&\leq \left\|(I - \frac{\mu_j}{\mu_{j-1}} I) \hat{A}_k^H \hat{B}_k\right\| + tol \\
&\leq \left\|(I - \frac{\mu_j}{\mu_{j-1}} I)\right\| \left\|\hat{A}_k^H \hat{B}_k\right\| + tol.
\end{aligned}$$

Theorem 3.2. *If $\hat{X}_k^*$ is the exact solution of the symmetric positive definite equations* (1.7)*, $X_{k,\mu_j,i}$ and $X_{k,\mu_j,i+1}$ are generated by the inner CG-iteration of Algorithm 1, then*

$$\left\|X_{k,\mu_j,i+1} - \hat{X}_k^*\right\| \leq \left(1 - \frac{1}{\kappa(\boldsymbol{A})}\right)^{\frac{1}{2}} \cdot \left\|X_{k,\mu_j,i} - \hat{X}_k^*\right\|, \tag{3.7}$$

where $\boldsymbol{A} = \hat{A}_k^H \hat{A}_k + \mu_j I$, $\kappa(\boldsymbol{A}) = \frac{\lambda_{max}(\boldsymbol{A})}{\lambda_{min}(\boldsymbol{A})}$. Here $\lambda_{max}(\boldsymbol{A})$ and $\lambda_{min}(\boldsymbol{A})$ are the maximum and minimum eigenvalues of $\boldsymbol{A}$ respectively.

Theorem 3.2 illustrates the convergence of Algorithm 1. Refer to [14] for the detailed proof process.

3.2. The A-tCGLS-FFT method

In the actual numerical calculation, if $\hat{A}_k^H \hat{A}_k$ is singular and μ is very small, then $\hat{A}_k^H \hat{A}_k + \mu_j I$ is ill-conditioned. Inspired by the idea in [15] for the numerical stability of a linear system in matrix form, in this section, we use the CGLS method instead of the CG method in Algorithm 1.

The main difference between Algorithms 2 and 1 is the updating of $z_i = \hat{B}_k - \hat{A}_k X_{k,\mu_j,i}$ in Algorithm 2 rather than the residuals (3.6) of Algorithm 1. We refer to [16] for more details for a corresponding

linear system in matrix form.

Algorithm 2: The A-tCGLS-FFT method for sloving (1.7)

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \mathcal{B} \in \mathbb{R}^{m\times m\times n}, \delta_1, ..., \delta_n, \mu_0, \rho, \eta > 1$
Output: Least-square solution $\mathcal{X}$ of (1.7)
$\hat{\mathcal{A}}$=fft($\mathcal{A}$, [ ], 3).
$\hat{\mathcal{B}}$=fft($\mathcal{B}$, [ ], 3).
for $k = 1, ..., n$ **do**
  $j = 0, X_{k,\mu_0} = 0$.
  while $\|\hat{A}_k X_{k,\mu_j} - \hat{B}_k\| \geq \eta\delta_k$ **do**
    $j = j + 1, \mu_j = \mu_0\rho^j$.
    $\mathbf{A}$=$\hat{A}_k^H\hat{A}_k + \mu_j I$,$\mathbf{B}$=$\hat{A}_k^H\hat{B}_k$.
    $X_0 = 0$ or $X_0 = X_{k,\mu_{j-1}}$;$z_0 = \hat{B}_k - \hat{A}_k X_0$;
    $R_0 = \hat{A}_k^H z_0 - \mu X_0$;$P_0 = R_0$.
    for $i = 0, 1, ...,$until convergence **do**
      $Q_i = \hat{A}_k P_i$.
      $\gamma_i = \left\langle R_j, R_j \right\rangle / \langle Q_i, Q_i\rangle + \mu_j \langle P_i, P_i\rangle$.
      $X_{i+1} = X_i + \gamma_i P_i$.
      $z_{i+1} = z_i - \gamma_i Q_i$.
      $R_{i+1} = \hat{A}_k^H z_{i+1} - \mu_j X_{i+1}$.
      $\omega_i = \langle R_{i+1}, R_{i+1}\rangle / \langle R_i, R_i\rangle$.
      $P_{i+1} = R_{i+1} + \omega_i P_i$.
    end for
    $X_{k,u_j} = X_{i+1}$.
  end while
  $\hat{\mathcal{X}}(:, :, k) = X_{k,u_j}$.
end for
$\mathcal{X}$=ifft($\hat{\mathcal{X}}$, [ ], 3).

3.3. A preconditioned tensor Conjugate-Gradient method

In this subsection, we consider the acceleration of Algorithm 1 by preconditioning. In Algorithm 1, the coefficient matrix $\hat{A}_k^H\hat{A}_k+\mu I$ of the k-th normal equation (3.2) is symmetric and positive definite. We set $M = \hat{A}_k^H\hat{A}_k + \mu I$ and apply approximate Cholesky decomposition to M. The process of approximate Cholesky decomposition can be directly realized by Matlab function *chol*, then $M_{chol} = chol(M)$ can be obtained. Let $M_L = M_{chol}^H$ and $M_R = M_{chol}$, then there is $M = M_L M_R$, where M_L is a lower triangular nonsingular sparse matrix and $M_R = M_L^H$. Then we solve the preconditioned normal equations

$$\tilde{A}_k\tilde{X}_k = \tilde{B}_k, \tag{3.8}$$

instead of (3.2) in Algorithm 1, where $\tilde{A} = M_L^{-1}\left(\hat{A}_k^H\hat{A}_k + \mu I\right)M_R^{-1}$, $\tilde{X} = M_R\hat{X}_k$ and $\tilde{B} = M_L^{-1}\hat{A}_k^H\hat{B}_k$.

Let $\hat{X}_{k,i}$ and $\tilde{X}_{k,i}$ represent the i-th iterate of (3.8) and (3.2) in the inner CG- iteration of Algorithm 1 under a certain regularization parameter μ, respectively. Then we have

$$\tilde{R}_{k,i} = \tilde{B}_k - \tilde{A}_k\tilde{X}_{k,i}$$

$$
\begin{aligned}
&= M_L^{-1}\hat{B}_k - M_L^{-1}\hat{A}_k M_R^{-1} M_R \hat{X}_{k,i} \\
&= M_L^{-1}(\hat{B}_k - \hat{A}_k \hat{X}_{k,i}) \\
&= M_L^{-1}\hat{R}_{k,i}.
\end{aligned} \tag{3.9}
$$

Denote $\tilde{P}_{k,i} = M_R\hat{P}_{k,i}$ and $\hat{t}_{k,i} = M_L^{-1}\hat{R}_{k,i}$, then we have

$$
\begin{aligned}
\tilde{\upsilon}_{k,i} &= \frac{\left\langle \tilde{R}_{k,i}, \tilde{R}_{k,i} \right\rangle}{\left\langle \tilde{A}_k \tilde{P}_{k,i}, \tilde{P}_{k,i} \right\rangle} \\
&= \frac{\left\langle M_L^{-1}\hat{R}_{k,i}, M_L^{-1}\hat{R}_{k,i} \right\rangle}{\left\langle M_L^{-1}\hat{A}_k M_R^{-1} M_R \hat{P}_{k,i}, M_R\hat{P}_{k,i} \right\rangle} \\
&= \frac{\left\langle M_L^{-1}\hat{R}_{k,i}, M_L^{-1}\hat{R}_{k,i} \right\rangle}{\left\langle M_L^{-1}\hat{A}_k \hat{P}_{k,i}, M_R\hat{P}_{k,i} \right\rangle}.
\end{aligned} \tag{3.10}
$$

According to the definition of matrix inner product $\langle A, B\rangle = tr(A^T B)$, where tr denotes the trace of a square matrix. Then, we have

$$
\begin{aligned}
\left\langle M_L^{-1}\hat{A}_k \hat{P}_{k,i}, M_R\hat{P}_{k,i} \right\rangle &= tr((M_L^{-1}\hat{A}_k \hat{P}_{k,i})^T M_R \hat{P}_{k,i}) \\
&= tr(\hat{P}_{k,i}^T \hat{A}_k^H M_L^{-T} M_R \hat{P}_{k,i}) \\
&= tr(\hat{P}_{k,i}^T \hat{A}_k^H \hat{P}_{k,i}) \\
&= \left\langle \hat{A}_k \hat{P}_{k,i}, \hat{P}_{k,i} \right\rangle.
\end{aligned} \tag{3.11}
$$

According to (3.10) and (3.11), we get

$$
\tilde{\upsilon}_{k,i} = \frac{\langle \hat{t}_{k,i}, \hat{t}_{k,i} \rangle}{\left\langle \hat{A}_k \hat{P}_{k,i}, \hat{P}_{k,i} \right\rangle}, \tag{3.12}
$$

and

$$
\begin{aligned}
\tilde{X}_{k,i+1} &= \tilde{X}_{k,i} + \tilde{\upsilon}_{k,i}\tilde{P}_{k.i}, \\
M_R\hat{X}_{k,i+1} &= M_R\hat{X}_{k,i} + \tilde{\upsilon}_{k,i} M_R \hat{P}_{k,i}, \\
\hat{X}_{k,i+1} &= \hat{X}_{k,i} + \tilde{\upsilon}_{k,i}\hat{P}_{k,i}.
\end{aligned} \tag{3.13}
$$

Then, we have

$$
\begin{aligned}
\tilde{R}_{k,i+1} &= \tilde{R}_{k,i} - \tilde{\upsilon}_{k,i}\tilde{A}_k \tilde{P}_{k,i}, \\
M_L^{-1}\hat{R}_{k,i+1} &= M_L^{-1}\hat{R}_{k,i} - \tilde{\upsilon}_{k,i} M_L^{-1}\hat{A}_k M_R^{-1} M_R \hat{P}_{k,i}, \\
\hat{R}_{k,i+1} &= \hat{R}_{k,i} - \tilde{\upsilon}_{k,i}\hat{A}_k \hat{P}_{k,i},
\end{aligned} \tag{3.14}
$$

and

$$
\tilde{\omega}_{k,i} = \frac{\left\langle \tilde{R}_{k,i+1}, \tilde{R}_{k,i+1} \right\rangle}{\left\langle \tilde{R}_{k,i}, \tilde{R}_{k,i} \right\rangle}
$$

$$= \frac{\left\langle M_L^{-1}\hat{R}_{k,i+1}, M_L^{-1}\hat{R}_{k,i+1}\right\rangle}{\left\langle M_L^{-1}\hat{R}_{k,i}, M_L^{-1}\hat{R}_{k,i}\right\rangle}$$
$$= \frac{\langle \hat{t}_{k,i+1}, \hat{t}_{k,i+1}\rangle}{\langle \hat{t}_{k,i}, \hat{t}_{k,i}\rangle}. \tag{3.15}$$

Finally, we have

$$\begin{aligned} \tilde{P}_{k,i+1} &= \tilde{R}_{k,i+1} + \tilde{\omega}_{k,i}\tilde{P}_{k,i}, \\ M_R\hat{P}_{k,i+1} &= M_L^{-1}\hat{R}_{k,i+1} + \tilde{\omega}_{k,i}M_R\hat{P}_{k,i}, \\ \hat{P}_{k,i+1} &= M_R^{-1}M_L^{-1}\hat{R}_{k,i+1} + \tilde{\omega}_{k,i}M_R^{-1}M_R\hat{P}_{k,i} = M_R^{-1}\hat{t}_{k,i+1} + \tilde{\omega}_{k,i}\hat{P}_{k,i}. \end{aligned} \tag{3.16}$$

Based on (3.10)–(3.16), we have the preconditioned process of Algorithm 1. Algorithm 3 lists the A-tPCG-FFT method.

Algorithm 3: The A-tPCG-FFT method for sloving (1.5)

Input: $\mathcal{A} \in \mathbb{R}^{m\times m\times n}, \mathcal{B} \in \mathbb{R}^{m\times m\times n}, \delta_1, ..., \delta_n, \mu_0, \rho, \eta > 1$
Output: Least-square solution $\mathcal{X}$ of (1.5)
$\hat{\mathcal{A}}$=fft($\mathcal{A}$, [ ], 3).
$\hat{\mathcal{B}}$=fft($\mathcal{B}$, [ ], 3).
for $k = 1, ..., n$ **do**
  $j = 0, X_{k,\mu_0} = 0$.
  while $\|\hat{A}_k X_{k,\mu_j} - \hat{B}_k\| \geq \eta\delta_k$ **do**
    $j = j + 1, \mu_j = \mu_0\rho^j$.
    $\mathbf{A}$=$\hat{A}_k^H\hat{A}_k + \mu_j I$,$\mathbf{B}$=$\hat{A}_k^H\hat{B}_k$.
    Decompose $\mathbf{A}$ to get M_R and M_L.
    $X_0 = 0$ or $X_0 = X_{k,\mu_{j-1}}$; $R_0 = M_L^{-1}(\mathbf{B} - \mathbf{A}X_0)$;$P_0 = R_0$.
    for $i = 0, 1, ...,$until convergence **do**
      $t_i = M_L^{-1}R_i$, $\tilde{\upsilon}_i = \langle t_i, t_i\rangle$ / $\langle \mathbf{A}P_i, P_i\rangle$.
      $X_{i+1} = X_i + \tilde{\upsilon}_i P_i$.
      $R_{i+1} = R_i - \tilde{\upsilon}_i \mathbf{A}P_i$.
      $\tilde{\omega}_i = \langle t_{i+1}, t_{i+1}\rangle$ / $\langle t_i, t_i\rangle$.
      $P_{i+1} = M_R^{-1}t_i + \tilde{\omega}_i P_i$.
    end for
    $X_{k,u_j} = X_{i+1}$.
  end while
  $\hat{\mathcal{X}}(:, :, k) = X_{k,u_j}$.
end for
$\mathcal{X}$=ifft($\hat{\mathcal{X}}$, [ ], 3).

The following result gives the convergence of Algorithm 3.

Theorem 3.3. *If $\hat{X}_k^*$ is the true solution of the symmetric positive definite equation* (1.7)*, $X_{k,\mu_j,0}$ is the*

initial solution in the internal CG iteration of Algorithm 3, and $X_{k,\mu_j,i}$ is the i-th iterate, then

$$\left\|X_{k,\mu_j,i} - \hat{X}_k^*\right\| \leq 2\left(\frac{\kappa(M_L^{-1}\boldsymbol{A}M_R^{-1}) - 1}{\kappa(M_L^{-1}\boldsymbol{A}M_R^{-1}) + 1}\right)^k \left\|X_{k,\mu_j,0} - \hat{X}_k^*\right\|, \tag{3.17}$$

where $\boldsymbol{A} = \hat{A}_k^H\hat{A}_k + \mu_j I$ and $\kappa(M)$ is the same as that in Theorem 3.2.

Proof. Let $\mathbf{A} = \hat{A}_k^H\hat{A}_k + \mu_j I$, Taking $\mathbf{A}$ as $M_L^{-1}\mathbf{A}M_R^{-1}$ in the similar result of Algorithm 1 in [17] results in (3.17).

We can expect from Theorem 3.3 that Algorithm 3 generally achieves better convergence than Algorithm 1 since

$$cond(M_L^{-1}\mathbf{A}M_R^{-1}) < cond(\mathbf{A}).$$

We will illustrate this in numerical experiments in Section 4.

4. Numerical examples

This section presents three examples to show the application of Algorithms 1–3 on the restoration of image and video. All calculations were performed in MATLAB R2018a on computers with intel core i7 and 16GB RAM.

Let $\mathcal{X}_{iter}$ denote the iterative solution to (1.5). The quality of the approximate solution $\mathcal{X}_{iter}$ is defined by *the relative error*

$$E_{iter} = \frac{\|\mathcal{X}_{iter} - \mathcal{X}_{true}\|_F}{\|\mathcal{X}_{true}\|_F},$$

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

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

where $\mathcal{X}_{true}$ denotes the uncontaminated data tensor and $E(\mathcal{X}_{true})$ is the average gray-level of $\mathcal{X}_{true}$.

In (1.5), we generate a noise tensor $\mathcal{E}$ and simulate the error in the data tensor $\mathcal{B} = \mathcal{B}_{true} + \mathcal{E}$. $\mathcal{E}$ is a noise tensor with a random term of normal distribution, the mean value is zero, and the variance selection corresponds to a specific noise level $\nu = \frac{\|\mathcal{E}\|_F}{\|\mathcal{B}_{true}\|_F}$.

We summarize the cross-channel blurring and within-channel blurring in the original blurring process in [3] that will be used in the following examples. The complete blurring model is presented in the form of

$$\left(A_{blur} \otimes A^{(1)} \otimes A^{(2)}\right) x_{true} = b_{true}, \tag{4.1}$$

where

$$A_{blur} = \begin{bmatrix} a_{rr} & a_{rg} & a_{rb} \\ a_{gr} & a_{gg} & a_{gb} \\ a_{br} & a_{bg} & a_{bb} \end{bmatrix},$$

and A_{blur} is a 3×3 matrix with the sum of each row equal to 1, which denotes cross-channel blurring as in [18].

$$b_{true} = \begin{bmatrix} \mathbf{vec}\,(B_{true,1}) \\ \mathbf{vec}\,(B_{true,2}) \\ \mathbf{vec}\,(B_{true,3}) \end{bmatrix}, x_{true} = \begin{bmatrix} \mathbf{vec}\,(X_{true,1}) \\ \mathbf{vec}\,(X_{true,2}) \\ \mathbf{vec}\,(X_{true,3}) \end{bmatrix},$$

and **vec**$(\cdot)$ is an operator that transforms a matrix into a vector by stacking columns of the matrix from left to right. $A^{(1)} \in \mathbb{R}^{n\times n}$ and $A^{(2)} \in \mathbb{R}^{n\times n}$ define within-channel blurring, which are the horizontal inner blurring matrix and the vertical inner blurring matrix, respectively, see [18] for more details. A special case to consider is $a_{rr} = a_{gg} = a_{bb} = \mu$, $a_{gr} = a_{rg} = a_{gb} = a_{bg} = \beta$ and $a_{br} = a_{rb} = \gamma$. If the formula (4.1) is calculated directly in MATLAB, the results are stored in the matrix and cannot be calculated effectively. Therefore, t-product structure is considered, and (4.1) is presented in the following tensor form

$$\mathcal{A}_v * \mathcal{X}_{true} * \mathcal{A}_\omega = \mathcal{B}_{true}, \tag{4.2}$$

where $\mathcal{A}_v \in \mathbb{R}^{n\times n\times 3}$ and $\mathcal{A}_\omega \in \mathbb{R}^{n\times n\times 3}$. Each blurring matrix $A^{(i)}$ is defined as follows:

$$a_{kl} = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(k-l)^2}{2\sigma^2}\right), |k-l| \le r \\ 0, \qquad otherwise \end{cases}, \tag{4.3}$$

where σ is a parameter that controls the amount of smoothing. Therefore, if σ is larger, the problem becomes more ill posed.

Example 4.1. (**Color image**) This example shows the restoration of a blurred penguin color image by Algorithms 1 – 3. This example compares the effects of A-tCG-FFT, A-tCGLS-FFT and A-tPCG-FFT in color image deblurring, and the image is contaminated by cross-channel blur and additive noise. The cross-channel bluring is determined by the matrix

$$A_{blur} = \begin{bmatrix} 0.7 & 0.15 & 0.15 \\ 0.15 & 0.7 & 0.15 \\ 0.15 & 0.15 & 0.7 \end{bmatrix}.$$

For the within-channel blurring, we set $\sigma = 4$ and $r = 7$. Let $\mathcal{A}_{v(:,:,1)} = \mu A^{(2)}, \mathcal{A}_{v(:,:,2)} = \beta A^{(2)}$, $\mathcal{A}_{v(:,:,3)} = \gamma A^{(2)}$, and $\mathcal{A}_\omega = \mathcal{I}$. The condition number of the obtained front slice of $\mathcal{A}$ is $cond(\mathcal{A}(:,:,k)) = 8.7257e+04 (k = 1, 2, 3)$. Let the two noise levels be $\nu = 10^{-2}$ and $\nu = 10^{-3}$, respectively. Formula (4.2) can be used to generate the blurred images, and $\mathcal{B} = \mathcal{B}_{true} + \mathcal{E}$ adds noise to the blurred image. Figure 2 shows the original image and the blurred and noisy image with $\nu = 10^{-3}$.

(a) (b)

Figure 2. (a) The original image of penguin, (b) the blurred and noisy image of penguin with $\nu = 10^{-3}$.

The essence of Algorithms 1–3 is to solve three inner and outer loops in the Fourier domain in turn. Each outer iteration uses the discrepancy principle to select the appropriate regularization parameter μ.

We specify $\eta = 1.05$ and $\rho = \frac{1}{2}$. We set the initial solution of inner iteration as $X_0 = X_{k,\mu_{j-1}}$, and the convergence criterion of the inner iteration is given by the residual norm being less than $tol = 10^{-6}$.

Table 2 lists CPU time, SNR and relative error of three algorithms for the restoration of penguin image. We can see from Table 2 that the A-tPCG-FFT algorithm needs the least CPU time among all the methods, while the A-tCG-FFT algorithm is the most time consuming. There is little difference between SNR and relative errors for all the methods.

Table 2. CPU time, SNR and relative error of different algorithms for the restoration of penguin image.

Noise level	Method	Relative error	SNR	CPU(secs)
10^{-3}	A-tCG-FFT	2.94×10^{-2}	22.26	27.98
	A-tCGLS-FFT	2.94×10^{-2}	22.26	16.68
	A-tPCG-FFT	2.93×10^{-2}	22.27	7.00
10^{-2}	A-tCG-FFT	5.01×10^{-2}	17.61	12.79
	A-tCGLS-FFT	5.01×10^{-2}	17.61	7.84
	A-tPCG-FFT	5.01×10^{-2}	17.61	3.46

Figure 3 shows the change of relative error with CPU time for the algorithms to process the third slice of penguin image with $\nu = 10^{-3}$ in Table 2. Figure 3 shows the iterative solution of the inner iteration and outer iteration of the three algorithms. It can be seen that when the regularization parameters are selected, the relative error norm values of the iterative solutions obtained by the inner iteration process of the three algorithms are gradually decreasing. Similarly, after the three algorithms automatically update the regularization parameters in the outer iteration, the iterative solutions corresponding to the updated regularization parameters are all smaller than the relative error norm of the iterative solutions before updating. We can still get from Figure 3 that the A-tPCG-FFT algorithm converges the fastest among all methods.

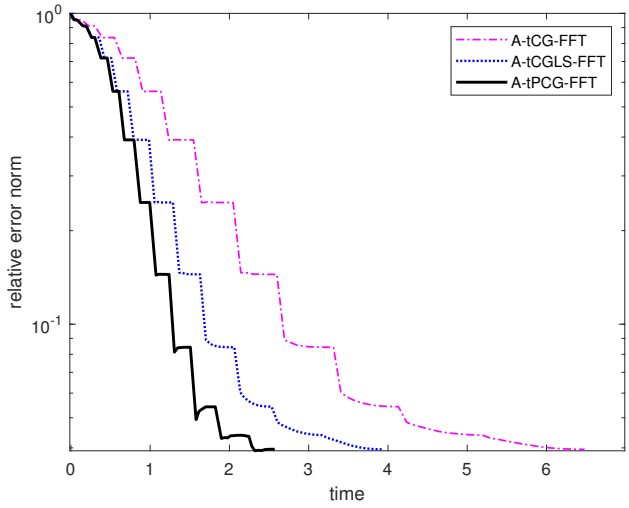


Figure 3. Comparison of relative errors versus CPU time for different methods.

Figure 4 displays the recovered penguin image for Algorithms 1–3 corresponding to the results with

$\nu = 10^{-3}$ in Table 2.

(a) (b) (c)

Figure 4. Recovered penguin images with different methods. (a) A-tCG-FFT, (b) A-tCGLS-FFT and (c) A-tPCG-FFT.

From Example 4.1 we can see that the A-tPCG-FFT method displays the best convergence among three methods.

Example 4.2 (**Color video**) We recover 10 consecutive frames (frames 41 to 50) of blurred and noised vipbarcode color video from MATLAB. Each frame has 240×240 pixels. We store 10 original video frames in the tensor $\mathcal{X}_{true} \in \mathbb{R}^{240\times30\times240}$, obtained by stacking the grayscale images that constitute the three channels of each blurred color frame.

These frames are blurred by (4.2), where $\mathcal{A}_\upsilon$ is a 3-way tensors such that $\mathcal{A}_{\upsilon(:,:,1)} = A^{(1)}$, $\mathcal{A}_{\upsilon(:,:,i)} = \mathcal{O}$ for $i = 2, \ldots 30$, and $\mathcal{A}_\omega = \mathcal{I}$. Using $\sigma = 3$ and $r = 5$ to build the blurring matrices with periodic boundary conditions. The condition number of the obtained first slice of $\mathcal{A}$ is $1.6191e + 03$, and the condition number of the remaining frontal sections of $\mathcal{A}$ is infinite. Use $\mathcal{B} = \mathcal{B}_{true} + \mathcal{E}$ adds noise to the blurred image, and the considered noise levels are $\nu = 10^{-2}$ and $\nu = 10^{-3}$. We set the initial solutions of their iterations as $X_0 = X_{k,\mu_{j-1}}$, and under the convergence condition that the residual norm is less than $tol = 10^{-6}$. Using the discrepancy principle to determine regularization parameters, specify $\eta = 1.1, \alpha_0 = \frac{1}{2}$ and $\rho = \frac{1}{2}$. Table 3 records the concrete results of recovering these 10 frames of video data, including relative error, SNR and CPU time. Because the same criteria for selecting parameters and stopping iteration are set, these three algorithms differ slightly in relative error and SNR. However, the A-tPCG-FFT algorithm has a strong advantage in stopping iteration CPU time.

Table 3. CPU time, SNR and relative error of different algorithms for the restoration of 10 frames of vipbarcode.

Noise level	Method	Relative error	SNR	time (secs)
10^{-3}	A-tCG-FFT	1.55×10^{-2}	25.91	115.08
	A-tCGLS-FFT	1.55×10^{-2}	25.91	68.25
	A-tPCG-FFT	1.55×10^{-2}	25.91	20.44
10^{-2}	A-tCG-FFT	4.66×10^{-2}	16.36	73.03
	A-tCGLS-FFT	4.66×10^{-2}	16.36	44.15
	A-tPCG-FFT	4.66×10^{-2}	16.36	12.53

Below we analyze the video recovery process of the 50th frame. Figure 5 shows the 50th original frame of this video file and the blurred and noisy video frame with $\nu = 10^{-3}$.

(a) (b)

Figure 5. (a) The original 50th frame of vipbarcode, (b) The blurred and noisy 50th frame of vipbarcode with $\nu = 10^{-3}$.

Figure 6 shows the variation of the relative errors of the third piece of the 50th video frame with CPU time when the three algorithms process $\nu = 10^{-3}$ in Table 3.

As can be seen from Figure 6, the convergence speed of algorithm A-tPCG-FFT is the fastest among the three algorithms. We can also get from Figure 6 that with the automatic updating of regularization parameters by the outer iteration and the progress of the inner iteration algorithm, the relative error norms of the iterative solutions obtained by the three algorithms are gradually decreasing.

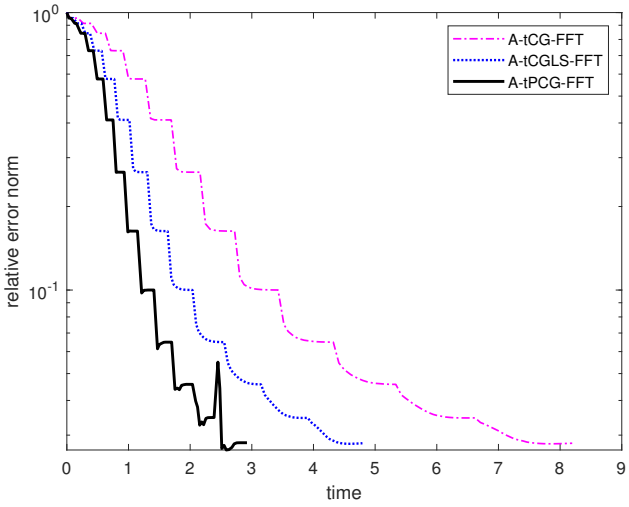


Figure 6. Comparison of relative errors versus CPU time for different methods.

Figure 7 is the result of the restoration of the blurred and noisy image of the 50th frame by A-tCG-FFT, A-tCGLS-FFT, and A-tPCG-FFT.

According to the results of Example 4.2, A-tCGLS-FFT and A-tPCG-FFT require less CPU time compared to A-tCG-FFT when recovering the data of each frame of video, so when processing multi-

frame video data, this lead time will be accumulated in each process. That is to say, the more video frames, the more obvious the time advantages of A-tCGLS-FFT and A-tPCG-FFT, especially A-tPCG-FFT.

(a) (b) (c)

Figure 7. Recovered the 50th frame with different methods. (a) A-tCG-FFT, (b) AtCGLS-FFT and (c) A-tPCG-FFT.

5. Conclusions

Based on the matrix conjugate gradient method, this paper presents three types of tensor Conjugate-Gradient methods for solving large-scale linear discrete ill-posed problems in tensor form. Firstly, we project the tensor equation to Fourier domain, and propose a strategy to automatically determine the regularization parameters of the tensor conjugate gradient method in the Fourier domain (A-tCG-FFT). In addition, we developed the A-tCGLS-FFT method and the preconditioned version of A-tCG-FFT. These proposed methods are used in different examples of color image and video restoration. Numerical experiments show that the tensor conjugate gradient methods are effective in solving ill-posed problems with t-product structure the Fourier domain.

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. TThis 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. M. E. Kilmer, C. D. Martin, Factorization strategies for third order tensors, *Linear Algebra Appl.*, **435** (2011), 641–658. https://doi.org/10.1016/j.laa.2010.09.020
2. N. Hao, M. E. Kilmer, K. Braman, R. C. Hoover, Facial recognition using tensor-tensor decompositions, *SIAM J. Imaging Sci.*, **6** (2013), 437–463. https://doi.org/10.1137/110842570
3. M. E. Guide, A. E. Ichi, K. Jbilou, R. Sadaka, On tensor GMRES and Golub-Kahan methods via the T-product for color image processing, *Electron. J. Linear Algebra*, **37** (2021), 524–543. https://doi.org/10.13001/ela.2021.5471
4. L. Reichel, U. O. Ugwu, The tensor Golub–Kahan–Tikhonov method applied to the solution of ill-posed problems with a t-product structure, *Numer. Linear Algebra Appl.*, **29** (2021), e2412. https://doi.org/10.1002/nla.2412
5. L. Reichel, U. O. Ugwu, Tensor Arnoldi–Tikhonov and GMRES-type methods for ill-posed problems with a t-product structure, *J. Sci. Comput.*, **90** (2022), 59. https://doi.org/10.1007/s10915-021-01719-1
6. J. Zhang, A. K. Saibaba, M. E. Kilmer, S. Aeron, A randomized tensor singular value decomposition based on the t-product, *Numer. Linear Algebra Appl.*, **25** (2018), e2179. https://doi.org/10.1002/nla.2179
7. U. Ugwu, L. Reichel, Tensor regularization by truncated iteration: A comparison of some solution methods for large-scale linear discrete ill-posed problem with a t-product, 2021, arXiv: 2110.02485. https://doi.org/10.48550/arXiv.2110.02485
8. K. Lund, The tensor t-function: A definition for functions of third-order tensors, *Numer. Linear Algebra Appl.*, **27** (2020), e2288. https://doi.org/10.1002/nla.2288
9. A. Ma, D. Molitor, Randomized Kaczmarz for tensor linear systems, *Bit Numer. Math.*, **62** (2022), 171–194. https://doi.org/10.1007/s10543-021-00877-w
10. M. E. Kilmer, K. Braman, N. Hao, R. C. Hoover, Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging, *SIAM J. Matrix Anal. Appl.*, **34** (2013), 148–172. https://doi.org/10.1137/110837711
11. H. W. Engl, M. Hanke, A. Neubauer, *Regularization of inverse problems*, Dordrecht: Springer, 2000.
12. S. Djennadi, N. Shawagfeh, O. A. Arqub, A fractional Tikhonov regularization method for an inverse backward and source problems in the time-space fractional diffusion equations, *Chaos Solition. Fract.*, **150** (2021), 111127. https://doi.org/10.1016/j.chaos.2021.111127
13. S. Djennadi, N. Shawagfeh, M. Inc, M. S. Osman, J. F. Gómez-Aguilar, O. A. Arqub, The Tikhonov regularization method for the inverse source problem of time fractional heat equation in the view of ABC-fractional technique, *Phys. Scr.*, **96** (2021), 094006. https://doi.org/10.1088/1402-4896/ac0867
14. G. H. Golub, C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, 1996.
15. J. Y. Yuan, Numerical methods for generalized least squares problems, *J. Comput. Appl. Math.*, **66** (1996), 571–584. https://doi.org/10.1016/0377-0427(95)00167-0

16. Å. Bjorck, T. Elfving, Z. Strakos, Stability of conjugate gradient and Lanczos methods for linear least squares problems, *SIAM J. Matrix Anal. Appl.*, **19** (1998), 720–736. https://doi.org/10.1137/S089547989631202X

17. L. N. Trefethen, D. Bau, *Numerical linear algebra*, SIAM, 1997. https://doi.org/10.1137/1.9780898719574

18. P. C. Hansen, J. G. Nagy, D. P. O'Leary, *Deblurring images: Matrices, spectra, and filtering*, SLAM, 2006.