Received: 21 February 2020 | Revised: 25 August 2020 | Accepted: 26 October 2020
DOI: 10.1002/nla.2351

RESEARCH ARTICLE

WILEY

Numerical subspace algorithms for solving the tensor equations involving Einstein product

Baohua Huang | **Wen Li**

School of Mathematical Sciences, South China Normal University, Guangzhou, P.R. China

Correspondence
Wen Li, School of Mathematical Sciences, South China Normal University, Guangzhou 510631, P.R. China.
Email: liwen@scnu.edu.cn

Funding information
China Postdoctoral Science Foundation, Grant/Award Number: 2019M660203; National Natural Science Foundation of China, Grant/Award Numbers: 11671158, 12001211, 12071159, U1811464; Guangdong Basic and Applied Basic Research Foundation, Grant/Award Numbers: 2020B1515310013

Abstract
In this article, we propose some subspace methods such as the conjugate residual, generalized conjugate residual, biconjugate gradient, conjugate gradient squared and biconjugate gradient stabilized methods based on the tensor forms for solving the tensor equation involving the Einstein product. These proposed algorithms keep the tensor structure. The convergence analysis shows that the proposed methods converge to the solution of the tensor equation for any initial value. Some numerical results confirm the feasibility and applicability of the proposed algorithms in practice.

KEYWORDS
conjugate residual algorithm, Einstein product, generalized conjugate residual algorithm, image restoration, projection method, tensor equation

1 | INTRODUCTION

For the sake of convenience, we first introduce some notations and definitions. For a positive integer N, let $I_1, \ldots, I_N$ be positive integers. An order N tensor $\mathcal{A} = (a_{i_1 \ldots i_N})(1 \leq i_j \leq I_j,\ j = 1, \ldots, N)$ is a multidimensional array with $I_1 I_2 \ldots I_N$ entries. Let $\mathbb{C}^{I_1\times\cdots\times I_N}$ and $\mathbb{R}^{I_1\times\cdots\times I_N}$ be the sets of the order N dimension $I_1\times\cdots\times I_N$ tensors over the complex field $\mathbb{C}$ and the real field $\mathbb{R}$, respectively. Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$ and $\mathcal{X} \in \mathbb{R}^{J_1\times\cdots\times J_N\times K_1\times\cdots\times K_M}$. The Einstein product $\mathcal{A}*_N\mathcal{X}$ of tensors $\mathcal{A}$ and $\mathcal{X}$ is a tensor in $\mathbb{R}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_M}$ whose $(i_1,\cdots,i_N,k_1,\cdots,k_M)$-entry is defined by[1]

$$(\mathcal{A}*_N\mathcal{X})_{i_1\cdots i_N k_1\cdots k_M} = \sum_{j_1\cdots j_N} a_{i_1\cdots i_N j_1\cdots j_N} b_{j_1\cdots j_N k_1\cdots k_M}.$$

It is noted that the Einstein product of tensors $\mathcal{A}$ and $\mathcal{X}$ is a kind of contracted product, which compresses the last N indices of tensor $\mathcal{A}$ and the first N indices of tensor $\mathcal{X}$.[2] The Einstein product reduces to the N-mode product $\mathcal{A}\overline{\times}_N b$ of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and a vector $b = (b_{i_N}) \in \mathbb{R}^{I_N}$, which is a tensor in $\mathbb{R}^{I_1\times\cdots\times I_{N-1}}$, and its $(i_1,\cdots,i_{N-1})$-entry is defined by

$$(\mathcal{A}\overline{\times}_N b)_{i_1\cdots i_{N-1}} = \sum_{i_N=1}^{I_N} a_{i_1 i_2\cdots i_{N-1} i_N} b_{i_N}.$$

The n-mode product (Tucker product) $\mathcal{X}\times_n A$ of a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and a matrix $A \in \mathbb{R}^{J\times I_n}$ is of the size $I_1\times\cdots\times I_{n-1}\times J\times I_{n+1}\times\cdots\times I_N$, and whose $(i_1,\cdots,i_{n-1},j,i_{n+1},\cdots,i_N)$-entry is defined by

$$(\mathcal{X}\times_n A)_{i_1\cdots i_{n-1} j i_{n+1}\cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1\cdots i_n\cdots i_N} a_{j i_n}.$$

Tensors are natural extensions of matrices. The tensor computing becomes a hot topic in numerical analysis, and have attracted the attentions of a lot of scholars.[3-8] Because various tensor equations are encountered in many applications of mechanics, physics, Markov process, control theory, partial differential equations, and engineering problems, the tensor equations play an important role in the tensor research. For example, the radiation transfer problem, the high-dimensional Poisson problem, and the Einstein gravitational field problem are modeled as some classic tensor equations. Particularly, by means of the spectrum allocation method, the discretization of the radiation transfer problem gives rise to the following Sylvester tensor equation via the Tucker product[9]

$$\mathcal{X}^m \times_1 A + \mathcal{X}^m \times_2 B + \mathcal{X}^m \times_3 C + K_a \mathcal{X}^m = \mathcal{F}, \tag{1}$$

where $\mathcal{X}^m$ is an unknown three order tensor, A, B, and C are square matrices associated with differential operators in each of three dimensions, and $\mathcal{F}$ is the known right-hand side tensor. Based on mixed-collocation and the finite difference method, the three-dimensional microscale dual phase lag problem[10,11]

$$\frac{\rho C_p}{\kappa}\left(\frac{\partial T}{\partial t} + \tau_p \frac{\partial^2 T}{\partial t^2}\right) = \nabla^2 T + \tau_t \frac{\nabla^2 T}{\partial t^2} + \frac{Q + \tau_p \partial T/\partial t}{\kappa}$$

leads to a tensor equation via the Tucker product

$$\mathcal{T}^{m+1} \times_1 B(l_x) + \mathcal{T}^{m+1} \times_2 B(l_y) + \mathcal{T}^{m+1} \times_3 B(l_z) = \mathcal{D}_m. \tag{2}$$

The discretization of the partial differential equation

$$\begin{cases} -\Delta u + c^T \nabla u = f, & \text{in } \Omega = [0,1]^N, \\ u = 0, & \text{on } \partial\Omega \end{cases}$$

leads to the following Sylvester tensor equation via the Tucker product[12,13]

$$\mathcal{X} \times_1 A^{(1)} + \mathcal{X} \times_2 A^{(2)} + \cdots + \mathcal{X} \times_N A^{(N)} = \mathcal{D}, \tag{3}$$

where the matrices $A^{(i)} \in \mathbb{R}^{I_i \times I_i} (i = 1, 2, \cdots, N)$ and the right-hand side tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ are known, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the unknown tensor to be determined.

Different from the tensor equations (1)–(3), another type of tensor equations is based on the Einstein product. The tensor equation via the Einstein product comes from engineering, the isotropic and anisotropic elastic models,[14] the brain MRI images restoration,[15,16] the biomedical science,[17] the finite element,[18] the finite difference or the spectral method[9,12] and plays very vital role in the discretization of some linear partial differential equations in high dimension. Such as, in physics, for noncentrosymmetric materials, the linear piezoelectric equation is expressed as[19]

$$\tilde{\mathcal{A}} *_2 T = p, \tag{4}$$

where $\tilde{\mathcal{A}} \in \mathbb{R}^{3\times3\times3}$ is a piezoelectric tensor, $T \in \mathbb{R}^{3\times3}$ is a stress tensor, and $p \in \mathbb{R}^3$ is the electric change density displacement. The three-dimensional model of image formation can be expressed as the convolution of an object and a point spread function associated with the noise[20]

$$G(x, y, z) = F(x, y, z) \circ S(x, y, z) + N(x, y, z), \tag{5}$$

where $G(x, y, z)$ is the image, $F(x, y, z)$ is the object, $N(x, y, z)$ is the noise, and $\circ$ denotes the three-dimensional convolution operator. Using the tensor representation, we can rewritten (5) as[15]

$$\mathcal{G} = \mathcal{T} *_3 \mathcal{F} + \mathcal{N}, \tag{6}$$

where $\mathcal{G}$, $\mathcal{F}$, and $\mathcal{N}$ are third-order tensor representations for the image, the object, and noise functions, respectively, and $\mathcal{T}$ is the sixth-order Toeplitz tensor (convolution operator) obtained from a third-order tensor $\mathcal{S}$, that is,

$$\mathcal{T}(i_1, i_2, i_3, j_1, j_2, j_3) = \mathcal{S}(j_1 - i_1, j_2 - i_2, j_3 - i_3), \quad 1 \le i_k, j_k \le n_k,\ 1 \le k \le 3. \tag{7}$$

Consider the high-dimensional Poisson problem

$$\begin{cases} -\nabla^2 v = f, & \text{in } \Omega = (0,1)^d, \\ v = 0, & \text{on } \partial\Omega, \end{cases} \tag{8}$$

where f is a given function and

$$\nabla^2 v = \frac{\partial^2 v}{\partial x_1^2} + \frac{\partial^2 v}{\partial x_2^2} + \cdots + \frac{\partial^2 v}{\partial x_d^2}.$$

For a positive integer n, set $h = 1/(n+1)$, $x_{i_1} = i_1 h, \cdots, x_{i_d} = i_d h$ and $f(x_{i_1}, \cdots, x_{i_d}) = f_{i_1,\cdots,i_d}$ for $i_1, \cdots, i_d = 1, 2, \cdots, n$. Set $v(x_{i_1}, \cdots, x_{i_d}) = v_{i_1,\cdots,i_d}$ for $i_1, \cdots, i_d = 0, 1, 2, \cdots, n, n+1$. By the standard central difference discretization on equidistant nodes of (8), the difference formula,

$$\begin{aligned} &\frac{v_{i_1-1,i_2,\cdots,i_d} - 2v_{i_1,i_2,\cdots,i_d} + v_{i_1+1,i_2,\cdots,i_d}}{h^2} + \frac{v_{i_1,i_2-1,\cdots,i_d} - 2v_{i_1,i_2,\cdots,i_d} + v_{i_1,i_2+1,\cdots,i_d}}{h^2} \\ &+ \cdots + \frac{v_{i_1,i_2,\cdots,i_d-1} - 2v_{i_1,i_2,\cdots,i_d} + v_{i_1+1,i_2,\cdots,i_d+1}}{h^2} = f_{i_1,i_2,\cdots,i_d}, \end{aligned} \tag{9}$$

is obtained. Since the Dirichlet boundary conditions are imposed so the values of v are zero at the boundary of Ω, that is, $v_{i_1,\cdots,0} = v_{i_1,\cdots,n+1} = \cdots = v_{0,\cdots,i_d} = v_{n+1,\cdots,i_d} = 0$ for $0 < i_1, \cdots, i_d < n+1$. Hence, by the Kronecker product, the discretized problem can be rewritten as

$$A_n \otimes \underbrace{I_n \otimes \cdots \otimes I_n}_{d-1} + I_n \otimes A_n \otimes \underbrace{I_n \otimes \cdots \otimes I_n}_{d-1} + \cdots + \underbrace{I_n \otimes \cdots \otimes I_n}_{d-1} \otimes A_n = \mathrm{vec}(\mathcal{V}) = h^2 \mathrm{vec}(\mathcal{F}), \tag{10}$$

where $\mathcal{V} = (v_{i_1,\cdots,i_d})$, $\mathcal{F} = (f_{i_1,\cdots,i_d}) \in \mathbb{R}^{\overbrace{n \times \cdots \times n}^{d}}$, and A_n and I_n denote the $n \times n$ Laplacian and identity matrices, respectively. By (9), we have

$$2dv_{i_1,i_2,\cdots,i_d} - v_{i_1-1,i_2,\cdots,i_d} - v_{i_1+1,i_2\cdots,i_d} - v_{i_1,i_2-1,\cdots,i_d} - v_{i_1,i_2+1,\cdots,i_d} - \cdots - v_{i_1,i_2,\cdots,i_d-1} - v_{i_1,i_2,\cdots,i_d+1} = h^2 f_{i_1\cdots i_d}. \tag{11}$$

Using the Einstein product, the higher order tensor representation of (11) (or say (10)) is

$$\overline{\mathcal{A}} *_d \mathcal{V} = \mathcal{F}, \tag{12}$$

where $\mathcal{V}, \mathcal{F} \in \mathbb{R}^{\overbrace{n \times \cdots \times n}^{d}}$, and the Laplacian tensor $\overline{\mathcal{A}} \in \mathbb{R}^{\overbrace{n \times \cdots \times n}^{2d}}$ would follow a $(2d+1)$-point stencil, that is,

$$\begin{aligned} &((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1,\alpha_2,\cdots,\alpha_d} = \frac{2d}{h^2}, \\ &((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1-1,\alpha_2,\cdots,\alpha_d} = ((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1+1,\alpha_2,\cdots,\alpha_d} = -\frac{1}{h^2}, \\ &((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1,\alpha_2-1,\cdots,\alpha_d} = ((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1,\alpha_2+1,\cdots,\alpha_d} = -\frac{1}{h^2}, \\ &\cdots \quad \cdots \\ &((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1,\alpha_2,\cdots,\alpha_d-1} = ((\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d})_{\alpha_1,\alpha_2,\cdots,\alpha_d+1} = -\frac{1}{h^2}, \end{aligned}$$

where $(\overline{\mathcal{A}})^{(2,4,\cdots,2d)}_{\alpha_1,\alpha_2,\cdots,\alpha_d} = \overline{\mathcal{A}}(:, \alpha_1, :, \alpha_2, \cdots, :, \alpha_d) \in \mathbb{R}^{\overbrace{n \times \cdots \times n}^{d}}$. When $d = 2$ and $d = 3$, the tensor equation (12) reduces to the discretization of 2D and 3D Poisson problems which have been discussed by Brazell et al.[21] The general form of the tensor equation (12) is

$$\mathcal{A} *_N \mathcal{X} = \mathcal{C}, \tag{13}$$

where $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_1 \times \cdots \times I_N}$, $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ and $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$. It is noted that the tensor equation (13) reduces to standard linear system if $M = 1$ and $N = 0$.

There are two different ways to solve tensor equations involving the Einstein product: one is the direct method, and another is the iterative method. The direct method is to employ the generalized inverses of tensors to give the analytical expression of the solution, see References 22-25. Compared with the direct method, the iterative method is more popular. For example, the conjugate gradient method, the preconditioned conjugate gradient method, and the Krylov subspace methods have been proposed for solving the tensor equation (13) by Wang and Xu,[26] Xie et al.,[16] and Huang et al.,[27] respectively. Hajarian[28] established the conjugate gradient-like methods for solving general tensor equation with the Einstein product. Huang and Ma[29] established the tensor form of conjugate gradient least squares method to find the solution of a class of tensor equation involving the Einstein product. The numerical methods in[16,27-29] are different from the standard numerical methods, which usually fail due to the so-called “curse of dimensionality” (see Bellman[30]). They can be classified as the tensor numerical methods, which were recognized as the basic tool to render numerical simulations in higher dimensions tractable. Khoromskij[31] presented an introductory description of traditional tensor formats with the focus on tensor-structured numerical methods for the calculation of multidimensional functions and operators. Moreover, there are some monographies and survey papers on the tensor numerical method in scientific computing, one can see References 31-38 and the references therein.

It is noted that the tensor equation (13) can be reformulated as a linear system, which can be solved by employing the standard numerical algorithms. Therefore, a natural question is that why do we explore the iteration methods for the tensor equation involving the Einstein product? We can answer this question by four aspects:

1. For the high-dimensional problems, the data have an inherent tensor structure. For example, the human brain is a three-dimensional structure, any point in the brain can be localized on the x, y, and z planes. The brain can be cut on any of these planes and are named the coronal plane, the horizontal plane, and the sagittal plane. If we want to reconstruct a part of telencephalon, only a slice of brain MRI image on horizontal is not enough, a lot of slices are needed. These slices have a relationship since they contain information of the part of telencephalon. If these MRI images are processed slice by slice, some information of the tensor structure of MRI images may be lost.
2. The Einstein product for tensor multiplication provides a natural setting for multilinear systems appeared in high-dimensional problems and data with inherent tensor structure. Without vectorization, the solution and the computational grid have a one-to-one correspondence so that the sophisticated boundary conditions are easily integrated in the multilinear system.
3. The tensor formulation via the Einstein product preserves the low-rank structure in the solution and the right-hand side. Such as, in high-dimensional Poisson problem, the solution and the right-hand side, both represented as $n \times \cdots \times n$ data arrays.
4. The matrix unfolding of a tensor may give rise to larger bandwidths than the original tensor which increases the number of operations and storage locations. For example, the Laplacian matrices in high dimensions have larger bandwidths than the Laplacian tensors.

Furthermore, due to the differences between tensors and matrices, the adoption of matrix-based methods requires a reduction of the spatial dimensions, which seriously destroys the intrinsic tensor structure of high-dimensional data and increases the computational cost of data analysis. In particular, by Tables 1 and 4 in Section 6, it shows that the tensor-based methods perform better than matrix-based ones for solving (13). Therefore, it is necessary to develop some tensor-based methods for solving multilinear systems arising from PDEs in high dimensions.

Although the multilinear systems arising from the Poisson problem in high dimensions are symmetric, it is meaningful for us to study the nonsymmetric tensor systems which arise from the convection diffusion problem. On the other hand, the Toeplitz/circular tensor linear systems can be applied to image processing. The efficient iterative methods for Toeplitz/circular tensor linear systems will be very important for image restoration.

By these motivations, in this article, we try to develop some efficient numerical algorithms in tensor forms for solving the tensor equation (13), and then apply our proposed tensor-based iterative methods to the three-dimensional Poisson problem, the convection diffusion problem and the brain MRI image restoration.

The main contributions of this article are given below.

- We propose the conjugate residual algorithm and the generalized conjugate residual algorithm in their tensor forms for solving the tensor equation (13) and give the corresponding convergence analysis via introducing the subspace spanned by tensor sequences when the coefficient tensor is symmetric positive definite and positive definite, respectively.
- We establish a unified framework of the projection method in the tensor format for solving the tensor equation (13) when the coefficient tensor is a general tensor, which has not been found in the existing research. From the established framework, BiCG, CGS, and BiCGSTAB methods and their convergence analysis in the tensor format can be derived.

Experimental results demonstrate that our proposed tensor-based methods, which preserve low bandwidth of the discrete tensor in high-dimension PDEs and keep the computational cost low, yields superior performance for the discrete tensor equation of the high-dimension PDEs, such as the three-dimensional Poisson problems and convection diffusion equation. The brain MRI image restoration shows that our proposed tensor-based iterative methods can use the information between different slices and recover a stack of images from the stack of blurred images at the same time, but not getting them slice by slice. By CPU time cost comparisons against the resolutions between our proposed tensor-based iterative methods and the matrix-based iterative methods, it demonstrates that our proposed tensor-based iterative methods have superior performance.

The remainder of this article is organized as follows. In Section 2, we describe some notations and common operations which will be used throughout this article. In Section 3, we derive the tensor form of the conjugate residual algorithm to solve the tensor equation, where the coefficient tensor is symmetric positive definite. Then, we prove that the solution can be obtained within a finite number of iterative steps in the absence of round-off errors. In Section 4, we present the tensor form of the generalized conjugate residual algorithm to solve the tensor equation, where the coefficient tensor is positive definite. Then, the finite termination of the proposed algorithm is proved. In Section 5, we describe how the well-known projection methods can be used based on tensor formats. Especially, the tensor forms of the biconjugate gradient, conjugate gradient squared, and biconjugate gradient stabilized methods are established. In Section 6, we give some numerical examples to illustrate our proposed iterative methods are feasible and effective. The article ends up with some concluding remarks in the final section.

2 | PRELIMINARIES

For convenience, we use the following notations throughout this article. Let $\mathcal{A} = (a_{i_1\cdots i_N j_1\cdots j_M}) \in \mathbb{C}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$. The tensor $\mathcal{A}^* \in \mathbb{C}^{J_1\times\cdots\times J_M\times I_1\times\cdots\times I_N}$ is called the conjugate transpose of $\mathcal{A}$, where its entries are given by $\overline{a}_{j_1\cdots j_N i_1\cdots i_M}$ for $1 \le i_s \le J_s$, $1 \le j_t \le I_t$ and $1 \le s \le M$, $1 \le t \le N$. If $\mathcal{A} = \mathcal{A}^*$, then $\mathcal{A}$ is called a Hermitian tensor. Similarly, the tensor $\mathcal{A}^T \in \mathbb{C}^{J_1\times\cdots\times J_M\times I_1\times\cdots\times I_N}$ is called the transpose of $\mathcal{A}$, where its entries are given by $a_{j_1\cdots j_N i_1\cdots i_M}$ for $1 \le i_s \le J_s$, $1 \le j_t \le I_t$ and $1 \le s \le M$, $1 \le t \le N$. If $\mathcal{A} = \mathcal{A}^T$, then $\mathcal{A}$ is called a real symmetric tensor. A tensor $\mathcal{D} = (d_{i_1\cdots i_N j_1\cdots j_M}) \in \mathbb{C}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ is called a diagonal tensor if all its entries are zero except for $d_{i_1\cdots i_N i_1\cdots i_N}$ for $1 \le i_s \le I_s$ and $1 \le s \le N$. In this case, if all the diagonal entries $d_{i_1\cdots i_N i_1\cdots i_N} = 1$, then $\mathcal{D}$ is the identity tensor, denoted by $\mathcal{I}$. Let $\mathcal{O}$ be the zero tensor in the case that all the entries are zero. We call $\mathrm{tr}(\mathcal{A}) = \sum_{i_1\cdots i_N} a_{i_1\cdots i_N i_1\cdots i_N}$ the trace of a tensor $\mathcal{A} \in \mathbb{C}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$.[21] Let I denote the product of $I_1, I_2, \cdots, I_N$, that is, $I = I_1 I_2 \cdots I_N$. Similarly, we write $J = J_1 J_2 \cdots J_N$, $K = K_1 K_2 \cdots K_M$ and $L = L_1 L_2 \cdots L_M$. For a given tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and any given $k \in \{1, 2, \cdots, I_N\}$, the notation $\mathcal{X}_{:,:,\ldots,:,k}$ denotes a tensor in $\mathbb{R}^{I_1\times I_2\times\cdots\times I_{N-1}}$, which is obtained by fixing the last index and is called the frontal slice. For more details, one can refer to Reference 34. For any given polynomial $f(x) = x^k + a_1 x^{k-1} + \cdots + a_{k-1}x + a_k$ and tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots I_N\times I_1\times\cdots\times I_N}$, we denote

$$f(\mathcal{A}) = \mathcal{A}^k + a_1\mathcal{A}^{k-1} + \cdots + a_{k-1}\mathcal{A} + a_k\mathcal{I},$$

where $\mathcal{A}^i = \mathcal{A} *_N \mathcal{A}^{i-1}$ and $\mathcal{I} \in \mathbb{R}^{I_1\times\cdots I_N\times I_1\times\cdots\times I_N}$ is an identity tensor.

Now, we recall the definition of inner product of two tensors.[21] Let $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$; the inner product of two tensors $\mathcal{A}, \mathcal{B}$ is defined by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \mathrm{tr}(\mathcal{B}^T *_N \mathcal{A}). \tag{14}$$

Then, for tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$, the tensor norm induced by this inner product is

$$\|\mathcal{A}\| = \sqrt{\sum_{i_1=1}^{I_1}\cdots\sum_{i_N=1}^{I_N}\sum_{j_1=1}^{I_1}\cdots\sum_{j_N=1}^{I_N}|a_{i_1\cdots i_N j_1\cdots j_N}|^2}.$$

When $\langle\mathcal{A},\mathcal{B}\rangle = 0$, we say $\mathcal{A}$ and $\mathcal{B}$ are orthogonal. Wang and Xu[26] established the symmetric property of this inner product.

Lemma 1 (See the work of Wang et al.[26]). *Let $\mathcal{A},\mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$. Then*

$$\langle\mathcal{A},\mathcal{B}\rangle = \mathrm{tr}(\mathcal{B}^T *_N \mathcal{A}) = \mathrm{tr}(\mathcal{A} *_M \mathcal{B}^T) = \mathrm{tr}(\mathcal{B} *_M \mathcal{A}^T) = \mathrm{tr}(\mathcal{A}^T *_N \mathcal{B}) = \langle\mathcal{B},\mathcal{A}\rangle.$$

Moreover, by the definition of inner product and the properties of tensor trace, it is easy to see that for any $\mathcal{A},\mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$ and any scalar $\alpha \in \mathbb{R}$,

(i) linearity in the first argument:

$$\langle\alpha\mathcal{A},\mathcal{B}\rangle = \alpha\langle\mathcal{A},\mathcal{B}\rangle,\ \langle\mathcal{A}+\mathcal{C},\mathcal{B}\rangle = \langle\mathcal{A},\mathcal{B}\rangle + \langle\mathcal{C},\mathcal{B}\rangle,$$

(ii) positive definiteness: $\langle\mathcal{A},\mathcal{A}\rangle > 0$ for all nonzero tensor $\mathcal{A}$.

Proposition 1 (See the work of Sun et al.[22]). *Let $\mathcal{A} \in \mathbb{C}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_N}$ and $\mathcal{B} \in \mathbb{R}^{K_1\times\cdots\times K_N\times J_1\times\cdots\times J_M}$. Then*

(i) $(\mathcal{A} *_N \mathcal{B})^* = \mathcal{B}^* *_N \mathcal{A}^*$;

(ii) $(\mathcal{I}_N *_N \mathcal{B}) = \mathcal{B}$ and $(\mathcal{B} *_M \mathcal{I}_M) = \mathcal{B}$, where the identity tensor $\mathcal{I}_N \in \mathbb{C}^{K_1\times\cdots\times K_N\times K_1\times\cdots\times K_N}$ and $\mathcal{I}_M \in \mathbb{C}^{J_1\times\cdots\times J_M\times J_1\times\cdots\times J_M}$.

Definition 1 (See the work of Brazell et al.[21]). Define the transformation $\Phi_{IJ} : \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N} \to \mathbb{R}^{I\times J}$ with $\Phi_{IJ}(\mathcal{A}) = A$ defined component-wise as

$$(\mathcal{A})_{i_1\cdots i_N j_1\cdots j_N} \to (A)_{st},$$

where $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$, $A \in \mathbb{R}^{I\times J}$, $s = i_N + \sum_{p=1}^{N-1}\left((i_p-1)\prod_{q=p+1}^{N-1} I_q\right)$, and $t = j_N + \sum_{p=1}^{N-1}\left((j_p-1)\prod_{q=p+1}^{N-1} J_q\right)$.

Definition 2. Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$. If $\Phi_{II}(\mathcal{A})$ is a symmetric positive definite (or symmetric positive semidefinite) matrix, then we say $\mathcal{A}$ is a symmetric positive definite (or symmetric positive semidefinite) tensor. If $\Phi_{II}(\mathcal{A})$ is a positive definite (or positive semidefinite) matrix, then we say $\mathcal{A}$ is a positive definite (or positive semidefinite) tensor.

Definition 3 (See the work of Wang et al.[26]). Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ be a symmetric positive definite tensor, and let $\mathcal{Q}_k \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$ and $\mathcal{P}_k \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M} (k = 1,2,\cdots)$ be two nonzero tensor sequences. If $\langle\mathcal{Q}_i,\mathcal{Q}_j\rangle = 0 (i \neq j,\ i,j = 1,2,\cdots)$ and $\langle\mathcal{P}_i,\mathcal{A} *_N \mathcal{P}_j\rangle = 0\ (i \neq j,\ i,j = 1,2,\cdots)$, then the tensor sequences $\{\mathcal{Q}_k\}$ and $\{\mathcal{P}_k\}$ are called orthogonal tensor sequence and $\mathcal{A}$-orthogonal tensor sequence, respectively.

Definition 4. For tensors $\mathcal{P}_1,\cdots,\mathcal{P}_k \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, the subspace spanned by the tensors $\mathcal{P}_1,\cdots,\mathcal{P}_k$ is

$$\mathrm{span}\{\mathcal{P}_1,\cdots,\mathcal{P}_k\} = \{\mathcal{X} : \mathcal{X} = \alpha_1\mathcal{P}_1 + \cdots + \alpha_k\mathcal{P}_k, \alpha_i \in \mathbb{R}, i = 1,\cdots,k\}.$$

We also need the $\boxtimes^{(N)}$ product between two N-mode tensors $\mathcal{X}$ and $\mathcal{Y}$.[7]

Definition 5 (See the work of Beik et al.[7]). The $\boxtimes^{(1)}$ and $\boxtimes^{(2)}$ products have the following form

$$\mathcal{X}\boxtimes^{(1)}\mathcal{Y} = \mathcal{X}^T\mathcal{Y}, \quad \mathcal{X},\mathcal{Y} \in \mathbb{R}^{I_1}$$

and

$$\mathcal{X}\boxtimes^{(2)}\mathcal{Y} = \mathcal{X}^T\mathcal{Y}, \quad \mathcal{X} \in \mathbb{R}^{I_1\times I_2}, \quad \mathcal{Y} \in \mathbb{R}^{I_1\times\tilde{I}_2}.$$

For $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3}$ and $\mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times\tilde{I}_3}$, the $\boxtimes^{(3)}$ product is an $I_3\times\tilde{I}_3$ matrix, which has the following form

$$\mathcal{X}\boxtimes^{(3)}\mathcal{Y} = \begin{pmatrix} \mathrm{tr}(\mathcal{X}_{::1}\boxtimes^{(2)}\mathcal{Y}_{::1}) & \mathrm{tr}(\mathcal{X}_{::1}\boxtimes^{(2)}\mathcal{Y}_{::2}) & \cdots & \mathrm{tr}(\mathcal{X}_{::1}\boxtimes^{(2)}\mathcal{Y}_{::\tilde{I}_3}) \\ \mathrm{tr}(\mathcal{X}_{::2}\boxtimes^{(2)}\mathcal{Y}_{::1}) & \mathrm{tr}(\mathcal{X}_{::2}\boxtimes^{(2)}\mathcal{Y}_{::2}) & \cdots & \mathrm{tr}(\mathcal{X}_{::2}\boxtimes^{(2)}\mathcal{Y}_{::\tilde{I}_3}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{tr}(\mathcal{X}_{::I_3}\boxtimes^{(2)}\mathcal{Y}_{::1}) & \mathrm{tr}(\mathcal{X}_{::I_3}\boxtimes^{(2)}\mathcal{Y}_{::2}) & \cdots & \mathrm{tr}(\mathcal{X}_{::I_3}\boxtimes^{(2)}\mathcal{Y}_{::\tilde{I}_3}) \end{pmatrix}.$$

Generally, for $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_{N-1}\times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times\cdots I_{N-1}\times\tilde{I}_N}$, the $\boxtimes^{(N)}$ product is an $I_N \times \tilde{I}_N$ matrix whose (i,j)th element is

$$\left[\mathcal{X}\boxtimes^{(N)}\mathcal{Y}\right]_{ij} = \mathrm{tr}(\mathcal{X}_{::\cdots:i}\boxtimes^{(N-1)}\mathcal{Y}_{::\cdots:j}), \quad N = 2, 3, \cdots.$$

The following proposition, which can be seen in Reference 26, serves as a “bridge” between the Einstein multiplication and the usual matrix multiplication.

Proposition 2 (See the work of Wang et al.[26]). *Let Φ_{IJ} be defined as in Definition 1. Then*

$$\mathcal{A}*_N\mathcal{X} = \mathcal{C} \Leftrightarrow \Phi_{IJ}(\mathcal{A}) \cdot \Phi_{JK}(\mathcal{X}) = \Phi_{IK}(\mathcal{C}),$$

where $\cdot$ refers to the usual matrix multiplication, $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$, $\mathcal{X} \in \mathbb{R}^{J_1\times\cdots\times J_N\times K_1\times\cdots\times K_M}$, and $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_M}$.

Lemma 2. *Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ be a symmetric positive definite tensor. Then, for any nonzero tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, we have $\langle \mathcal{X}, \mathcal{A}*_N\mathcal{X}\rangle > 0$.*

Proof. Assume that $\Phi_{II}(\mathcal{A}) = A$ and $\Phi_{IJ}(\mathcal{X}) = X$. Then, by Definition 1, we know $A \in \mathbb{R}^{II}$ and $X \in \mathbb{R}^{IJ}$. According to Definition 2, $A \in \mathbb{R}^{II}$ is a symmetric positive definite matrix.

Let

$$\mathcal{Y} = \mathcal{X}^T*_N\mathcal{A}*_N\mathcal{X}.$$

It then follows from Proposition 2 that $Y = X^TAX$, where $Y = \Phi_{JJ}(\mathcal{Y}) \in \mathbb{R}^{JJ}$. Let $X = (x_1, x_2\cdots, x_J)$ with $x_j \in \mathbb{R}^I$ for $j = 1, 2, \cdots, J$. Then

$$Y = X^TAX = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_J^T \end{pmatrix} A(x_1, x_2\cdots, x_J) = \begin{pmatrix} x_1^TAx_1 & x_1^TAx_2 & \cdots & x_1^TAx_J \\ x_2^TAx_1 & x_2^TAx_2 & \cdots & x_2^TAx_J \\ \vdots & \vdots & \ddots & \vdots \\ x_J^TAx_1 & x_J^TAx_2 & \cdots & x_J^TAx_J \end{pmatrix}.$$

Since $\mathcal{X}$ is a nonzero tensor, we know X is a nonzero matrix. So, at least one of $x_j \in \mathbb{R}^J$ is a nonzero vector. It then follows from the symmetric positive definiteness of the matrix A that $\mathrm{tr}(Y) = \sum_{j\in J}(Y)_{jj} = \sum_{j\in J}x_j^TAx_j > 0$. Hence, by Definition 1, Lemma 1, and the definitions of inner product and tensor trace, we conclude that

$$\langle \mathcal{X}, \mathcal{A}*_N\mathcal{X}\rangle = \mathrm{tr}(\mathcal{X}^T*_N\mathcal{A}*_N\mathcal{X}) = \mathrm{tr}(\mathcal{Y}) = \sum_{j_1j_2\cdots j_M}(\mathcal{Y})_{j_1j_2\cdots j_Mj_1j_2\cdots j_M} = \sum_{j\in J}(Y)_{jj} = \mathrm{tr}(Y) > 0,$$

which completes the proof. ∎

Corollary 1. *Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ be a positive definite tensor. Then, for any nonzero tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, we have $\langle \mathcal{X}, \mathcal{A}*_N\mathcal{X}\rangle > 0$.*

Proof. The proof is similar to that of Lemma 2, so we omit here. ∎

Lemma 3 (See the work of Huang et al.[29]). *Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_N}$. Then, for any $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$ and $\mathcal{Y} \in \mathbb{R}^{K_1\times\cdots\times K_N\times J_1\times\cdots\times J_M}$, we have $\langle \mathcal{X}, \mathcal{A}*_N\mathcal{Y}\rangle = \langle \mathcal{A}^T*_N\mathcal{X}, \mathcal{Y}\rangle$.*

Lemma 4. *Let $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ be a symmetric tensor. Then, for any $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, we have $\langle \mathcal{X}, \mathcal{A}*_N\mathcal{Y}\rangle = \langle \mathcal{A}*_N\mathcal{X}, \mathcal{Y}\rangle$.*

Proof. By the similar proof method of Lemma 3, we can derive this result. So, we omit here. ■

Lemma 5 (See the work of Beik et al.[7]). *Suppose that $\mathcal{B} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N\times m}$ is an $(N+1)$-mode tensor with the column tensors $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_m \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and $z = (z_1, z_2, \cdots, z_m)^T \in \mathbb{R}^m$. Then, for any $(N+1)$-mode tensor $\mathcal{A}$with N-mode column tensors $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, the following statement holds*

$$\mathcal{A}\boxtimes^{(N+1)}(\mathcal{B}\overline{\times}_{(N+1)}z) = (\mathcal{A}\boxtimes^{(N+1)}\mathcal{B})z.$$

3 | THE TENSOR FORM OF CONJUGATE RESIDUAL ALGORITHM AND CONVERGENCE ANALYSIS

In this section, we derive the tensor form of the conjugate residual algorithm (CR-BTF) for solving the tensor equation (13), where the coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ is symmetric positive definite.

Wang and Xu[26] established the conjugate gradient (CG) method to solve the tensor equation (13) and proved that the solution (the least Frobenius norm solution) of the related problems can be obtained within a finite number of iterative steps in the absence of round-off errors. The conjugate residual (CR) algorithm, which is proposed by Saad,[39] is another classical method for solving linear equations. Considering the robustness and effectivity of the CR method in solving linear equations, in this section, we propose the conjugate residual algorithm based on tensor format (CR-BTF) to solve the tensor equation (13), which is described as in Algorithm 1.

Algorithm 1. The CR-BTF method

Given the symmetric positive definite tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ and the right-hand side tensor $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$.

1: **Input**: the initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$,the maximum number of iterations $\mathrm{Iter}_{\max}$, the tolerance error $\varepsilon>0$.
2: Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$;
3: $\mathcal{P}_0 = \mathcal{R}_0$;
4: $\mathcal{Z}_0 = \mathcal{A} *_N \mathcal{R}_0$;
5: $k = 0$;
6: **while** ($k< \mathrm{Iter}_{\max}$)
7: $\quad \mathcal{U}_k = \mathcal{A} *_N \mathcal{P}_k$;
8: $\quad \alpha_k = \dfrac{\langle \mathcal{R}_k, \mathcal{U}_k\rangle}{\langle \mathcal{U}_k, \mathcal{U}_k\rangle}$;
9: $\quad \mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k\mathcal{P}_k$;
10: $\quad \mathcal{R}_{k+1} = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_{k+1} = \mathcal{R}_k - \alpha_k\mathcal{U}_k$;
11: $\quad$ **if** $\|\mathcal{R}_{k+1}\| < \varepsilon$, then **stop**
12: $\quad \mathcal{Z}_{k+1} = \mathcal{A} *_N \mathcal{R}_{k+1}$;
13: $\quad \beta_k = \dfrac{\langle \mathcal{Z}_{k+1}, \mathcal{R}_{k+1}\rangle}{\langle \mathcal{Z}_k, \mathcal{R}_k\rangle}$;
14: $\quad \mathcal{P}_{k+1} = \mathcal{R}_{k+1} + \beta_k\mathcal{P}_k$;
15: **end while**

Remark 1. In order to save computation and memory requirements, we calculate $\mathcal{U}_{k+1}$ by means of line 14 of Algorithm 1. Then, we have $\mathcal{U}_{k+1} = \mathcal{Z}_{k+1} + \beta_k\mathcal{U}_k$.

For the convenience of discussion, we adopt the following notation. Given a symmetric positive definite tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$, for any tensor $\mathcal{Y} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, we denote

$$\underbrace{\mathcal{A}*_N\cdots\mathcal{A}*_N}_{\mathcal{Y}} \triangleq \mathcal{A}^k *_N \mathcal{Y}. \tag{15}$$

Then, we have the following lemma.

Lemma 6. *Let the sequences* $\{\mathcal{P}_k\}$ *and* $\{\mathcal{R}_k\}$ *be generated by* Algorithm 1. Then

$$\mathrm{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_k\} = \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_k\} = \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^k*_N\mathcal{P}_0\}. \tag{16}$$

Proof. We prove this result by mathematical induction. For $k=1$, by lines 3 and 14 of Algorithm 1, we have

$$\mathcal{P}_1 = \mathcal{R}_1 + \beta_0\mathcal{P}_0 = \mathcal{R}_1 + \beta_0\mathcal{R}_0 \in \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1\}$$

and

$$\mathcal{R}_1 = \mathcal{P}_1 - \beta_0\mathcal{P}_0 \in \mathrm{span}\{\mathcal{P}_0, \mathcal{P}_1\}.$$

Hence, $\mathrm{span}\{\mathcal{P}_0, \mathcal{P}_1\} = \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1\}$. On the other hand, according to lines 7 and 10 of Algorithm 1, we obtain

$$\mathcal{R}_1 = \mathcal{R}_0 - \alpha_0\mathcal{U}_0 = \mathcal{R}_0 - \alpha_0\mathcal{A}*_N\mathcal{P}_0 = \mathcal{P}_0 - \alpha_0\mathcal{A}*_N\mathcal{P}_0 \in \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0\}$$

and

$$\mathcal{A}*_N\mathcal{P}_0 = \frac{1}{\alpha_0}(\mathcal{R}_0 - \mathcal{R}_1) \in \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1\}.$$

Hence, $\mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0\} = \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1\}$. Therefore, (16) holds for $k=1$. Assume that (16) holds for $k=l$. For $k=l+1$, by lines 7, 10, and 14 of Algorithm 1, we have

$$\mathcal{R}_{l+1} = \mathcal{R}_l - \alpha_l\mathcal{U}_l = \mathcal{R}_l - \alpha_l\mathcal{A}*_N\mathcal{P}_l,$$

which, together with the induction principle, yields that

$$\mathcal{R}_{l+1} \in \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\}$$

and

$$\mathcal{P}_{l+1} = \mathcal{R}_{l+1} + \beta_l\mathcal{P}_l \in \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\}.$$

Hence

$$\mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\} \subseteq \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\} \tag{17}$$

and

$$\mathrm{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_{l+1}\} \subseteq \mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\}. \tag{18}$$

On the other hand, according to line 14 of Algorithm 1 and the induction principle, we have

$$\begin{aligned}\mathcal{A}^{l+1}*_N\mathcal{P}_0 &= \mathcal{A}*_N(\mathcal{A}^l*_N\mathcal{P}_0) = \mathcal{A}*_N(\sum_{i=0}^{l}\xi_i\mathcal{P}_i) = \sum_{i=0}^{l}\xi_i(\mathcal{A}*_N\mathcal{P}_i)\\ &= \sum_{i=0}^{l}\xi_i\frac{1}{\alpha_i}(\mathcal{R}_i - \mathcal{R}_{i+1}) \in \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\}.\end{aligned}$$

That is,

$$\mathrm{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\} \subseteq \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\}. \tag{19}$$

Combining (17), (18), and (19) leads to

$$\operatorname{span}\{\mathcal{P}_0, \mathcal{A}*_N\mathcal{P}_0, \cdots, \mathcal{A}^{l+1}*_N\mathcal{P}_0\} = \operatorname{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\}$$

and

$$\operatorname{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_{l+1}\} \subseteq \operatorname{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\}. \tag{20}$$

Finally, by line 14 of Algorithm 1, we immediately have

$$\mathcal{R}_{l+1} = \mathcal{P}_{l+1} - \beta_l\mathcal{P}_l \in \operatorname{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_{l+1}\},$$

which, together with (20), yields that

$$\operatorname{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_{l+1}\} = \operatorname{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{l+1}\}.$$

Thus, the relation (16) holds for $k = l + 1$. By the induction principle, we obtain the result and the proof is completed. ■

Lemma 7. *Let the tensor sequences $\{\mathcal{U}_k\}$, $\{\mathcal{R}_k\}$, and $\{\mathcal{Z}_k\}$ be generated by Algorithm 1. Then*
(i) $\langle \mathcal{R}_k, \mathcal{U}_s\rangle = 0,\ k > s$;
(ii) $\langle \mathcal{R}_k, \mathcal{U}_k\rangle = \langle \mathcal{R}_k, \mathcal{Z}_k\rangle$;
(iii) $\langle \mathcal{U}_k, \mathcal{U}_s\rangle = 0,\ k \neq s$.

Proof. We apply mathematical induction. For $k = 0$, we only need consider (ii). Notice that $\mathcal{R}_0 = \mathcal{P}_0$, we immediately have

$$\langle \mathcal{R}_0, \mathcal{U}_0\rangle = \langle \mathcal{R}_0, \mathcal{A}*_N\mathcal{P}_0\rangle = \langle\ \mathcal{R}_0, \mathcal{A}*_N\mathcal{R}_0\rangle = \langle \mathcal{R}_0, \mathcal{Z}_0\rangle.$$

For $k = 1$, by line 10 of Algorithm 1 and the definition of α_0, we obtain

$$\langle \mathcal{R}_1, \mathcal{U}_0\rangle = \langle \mathcal{R}_0, \mathcal{U}_0\rangle - \alpha_0\langle \mathcal{U}_0, \mathcal{U}_0\rangle = \langle \mathcal{R}_0, \mathcal{U}_0\rangle - \frac{\langle \mathcal{R}_0, \mathcal{U}_0\rangle}{\langle \mathcal{U}_0, \mathcal{U}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle = 0. \tag{21}$$

According to lines 7, 12, and 14 of Algorithm 1 and the relation (21), we have

$$\langle \mathcal{R}_1, \mathcal{Z}_1\rangle = \langle \mathcal{R}_1, \mathcal{A}*_N\mathcal{R}_1\rangle = \langle \mathcal{R}_1, \mathcal{A}*_N(\mathcal{P}_1 - \beta_0\mathcal{P}_0)\rangle = \langle \mathcal{R}_1, \mathcal{U}_1\rangle - \beta_0\langle \mathcal{R}_1, \mathcal{U}_0\rangle = \langle \mathcal{R}_1, \mathcal{U}_1\rangle. \tag{22}$$

It then follows from Algorithm 1, Lemmas 1 and 4, Remark 1, and the definitions of α_0 and β_0 that

$$\begin{aligned}
\langle \mathcal{U}_1, \mathcal{U}_0\rangle &= \langle \mathcal{Z}_1 + \beta_0\mathcal{U}_0, \mathcal{U}_0\rangle = \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \beta_0\langle \mathcal{U}_0, \mathcal{U}_0\rangle \\
&= \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\langle \mathcal{Z}_0, \mathcal{R}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle = \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\langle \mathcal{A}*_N\mathcal{R}_0, \mathcal{R}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle \\
&= \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\langle \mathcal{A}*_N\mathcal{P}_0, \mathcal{R}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle = \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\langle \mathcal{U}_0, \mathcal{R}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle \\
&= \langle \mathcal{Z}_1, \mathcal{U}_0\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\alpha_0} = \langle \mathcal{Z}_1, \frac{\mathcal{R}_0 - \mathcal{R}_1}{\alpha_0}\rangle + \frac{\langle \mathcal{Z}_1, \mathcal{R}_1\rangle}{\alpha_0} \\
&= \frac{1}{\alpha_0}\langle \mathcal{Z}_1, \mathcal{R}_0\rangle = \frac{1}{\alpha_0}\langle \mathcal{A}*_N\mathcal{R}_1, \mathcal{P}_0\rangle \\
&= \frac{1}{\alpha_0}\langle \mathcal{R}_1, \mathcal{A}*_N\mathcal{P}_0\rangle = \frac{1}{\alpha_0}\langle \mathcal{R}_1, \mathcal{U}_0\rangle = 0.
\end{aligned} \tag{23}$$

From (21)–(23), we conclude that the results hold for $k = 1$.
Suppose that the results hold for $k = l$. Now we consider $k = l + 1$.

First, we prove that (i) holds for $k=l+1$. According to line 10 of Algorithm 1, we have

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_s\rangle = \langle \mathcal{R}_l - \alpha_l \mathcal{U}_l, \mathcal{U}_s\rangle = \langle \mathcal{R}_l, \mathcal{U}_s\rangle - \alpha_l\langle \mathcal{U}_l, \mathcal{U}_s\rangle.$$

When $s<l$, by the induction principle, we get

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_s\rangle = \langle \mathcal{R}_l, \mathcal{U}_s\rangle - \alpha_l\langle \mathcal{U}_l, \mathcal{U}_s\rangle = 0.$$

When $s=l$, by the definition of α_l, we immediately have

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_l\rangle = \langle \mathcal{R}_l, \mathcal{U}_l\rangle - \alpha_l\langle \mathcal{U}_l, \mathcal{U}_l\rangle = \langle \mathcal{R}_l, \mathcal{U}_l\rangle - \frac{\langle \mathcal{R}_l, \mathcal{U}_l\rangle}{\langle \mathcal{U}_l, \mathcal{U}_l\rangle}\langle \mathcal{U}_l, \mathcal{U}_l\rangle = 0.$$

Thus, (i) holds for $k=l+1$.

Second, we prove that (ii) holds for $k=l+1$. By lines 7, 12, and 14 of Algorithm 1 and the first claim of Lemma 7, one has

$$\langle \mathcal{R}_{l+1}, \mathcal{Z}_{l+1}\rangle = \langle \mathcal{R}_{l+1}, \mathcal{A}*_N\mathcal{R}_{l+1}\rangle = \langle \mathcal{R}_{l+1}, \mathcal{A}*_N(\mathcal{P}_{l+1} - \beta_l\mathcal{P}_l)\rangle = \langle \mathcal{R}_{l+1}, \mathcal{U}_{l+1}\rangle - \beta_l\langle \mathcal{R}_{l+1}, \mathcal{U}_l\rangle = \langle \mathcal{R}_{l+1}, \mathcal{U}_{l+1}\rangle.$$

Thus, (ii) holds for $k=l+1$.

Finally, we prove that (iii) holds for $k=l+1$. Based on Remark 1, we have

$$\langle \mathcal{U}_{l+1}, \mathcal{U}_s\rangle = \langle \mathcal{Z}_{l+1} + \beta_l\mathcal{U}_l, \mathcal{U}_s\rangle = \langle \mathcal{Z}_{l+1}, \mathcal{U}_s\rangle + \beta_l\langle \mathcal{U}_l, \mathcal{U}_s\rangle.$$

When $s=l$, using Lemmas 1 and 4, lines 12–14 of Algorithm 1 and the first and second claims of Lemma 7, we get

$$\begin{aligned}
\langle \mathcal{U}_{l+1}, \mathcal{U}_l\rangle &= \langle \mathcal{Z}_{l+1}, \mathcal{U}_l\rangle + \beta_l\langle \mathcal{U}_l, \mathcal{U}_l\rangle = \langle \mathcal{Z}_{l+1}, \mathcal{U}_l\rangle + \frac{\langle \mathcal{Z}_{l+1}, \mathcal{R}_{l+1}\rangle}{\langle \mathcal{Z}_l, \mathcal{R}_l\rangle}\langle \mathcal{U}_l, \mathcal{U}_l\rangle \\
&= \langle \mathcal{Z}_{l+1}, \mathcal{U}_l\rangle + \frac{\langle \mathcal{Z}_{l+1}, \mathcal{R}_{l+1}\rangle}{\langle \mathcal{R}_l, \mathcal{U}_l\rangle}\langle \mathcal{U}_l, \mathcal{U}_l\rangle = \langle \mathcal{Z}_{l+1}, \mathcal{U}_l\rangle + \frac{\langle \mathcal{Z}_{l+1}, \mathcal{R}_{l+1}\rangle}{\alpha_l} \\
&= \langle \mathcal{Z}_{l+1}, \frac{\mathcal{R}_l - \mathcal{R}_{l+1}}{\alpha_l}\rangle + \frac{1}{\alpha_l}\langle \mathcal{Z}_{l+1}, \mathcal{R}_{l+1}\rangle \\
&= \frac{1}{\alpha_l}\langle \mathcal{Z}_{l+1}, \mathcal{R}_l\rangle = \frac{1}{\alpha_l}\langle \mathcal{A}*_N\mathcal{R}_{l+1}, \mathcal{R}_l\rangle = \frac{1}{\alpha_l}\langle \mathcal{R}_{l+1}, \mathcal{A}*_N\mathcal{R}_l\rangle \\
&= \frac{1}{\alpha_l}\langle \mathcal{R}_{l+1}, \mathcal{A}*_N(\mathcal{P}_l - \beta_{l-1}\mathcal{P}_{l-1})\rangle \\
&= \frac{1}{\alpha_l}\langle \mathcal{R}_{l+1}, \mathcal{U}_l - \beta_{l-1}\mathcal{U}_{l-1}\rangle = 0.
\end{aligned} \tag{24}$$

When $s<l$, by lines 10, 12, and 14 of Algorithm 1, the first claim of Lemma 7 and the induction principle, we obtain

$$\begin{aligned}
\langle \mathcal{U}_{l+1}, \mathcal{U}_s\rangle &= \langle \mathcal{Z}_{l+1}, \mathcal{U}_s\rangle + \beta_l\langle \mathcal{U}_l, \mathcal{U}_s\rangle = \langle \mathcal{Z}_{l+1}, \mathcal{U}_s\rangle \\
&= \langle \mathcal{Z}_{l+1}, \frac{\mathcal{R}_s - \mathcal{R}_{s+1}}{\alpha_s}\rangle = \frac{1}{\alpha_s}[\langle \mathcal{Z}_{l+1}, \mathcal{R}_s\rangle - \langle \mathcal{Z}_{l+1}, \mathcal{R}_{s+1}\rangle] \\
&= \frac{1}{\alpha_s}[\langle \mathcal{A}*_N\mathcal{R}_{l+1}, \mathcal{R}_s\rangle - \langle \mathcal{A}*_N\mathcal{R}_{l+1}, \mathcal{R}_{s+1}\rangle] \\
&= \frac{1}{\alpha_s}[\langle \mathcal{R}_{l+1}, \mathcal{A}*_N\mathcal{R}_s)\rangle - \langle \mathcal{R}_{l+1}, \mathcal{A}*_N\mathcal{R}_{s+1})\rangle] \\
&= \frac{1}{\alpha_s}[\langle \mathcal{R}_{l+1}, \mathcal{A}*_N(\mathcal{P}_s - \beta_{s-1}P_{s-1})\rangle - \langle \mathcal{R}_{l+1}, \mathcal{A}*_N(\mathcal{P}_{s+1} - \beta_s P_s)\rangle] \\
&= \frac{1}{\alpha_s}[\langle \mathcal{R}_{l+1}, \mathcal{U}_s - \beta_{s-1}\mathcal{U}_{s-1}\rangle - \langle \mathcal{R}_{l+1}, \mathcal{U}_{s+1} - \beta_s\mathcal{U}_s\rangle] = 0.
\end{aligned} \tag{25}$$

According to (24) and (25), we know (iii) holds for $k=l+1$. By the induction principle, we obtain these results. The proof is completed. ■

Lemma 8. *Let the tensor sequences $\{\mathcal{U}_k\}$, $\{\mathcal{R}_k\}$, and $\{\mathcal{Z}_k\}$ be generated by Algorithm 1. Then*
(i) $\langle \mathcal{R}_k, \mathcal{U}_s\rangle = \langle \mathcal{R}_0, \mathcal{U}_s\rangle,\ k \le s$;
(ii) $\langle \mathcal{R}_k, \mathcal{Z}_s\rangle = 0,\quad k > s.$

Proof. (i) According to Algorithm 1 and the third claim of Lemma 7, we have

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_s\rangle = \langle \mathcal{R}_l - \alpha_l \mathcal{U}_l, U_s\rangle = \langle \mathcal{R}_l, \mathcal{U}_s\rangle - \alpha_l\langle \mathcal{U}_l, \mathcal{U}_s\rangle = \langle \mathcal{R}_l, \mathcal{U}_s\rangle = \cdots = \langle \mathcal{R}_0, \mathcal{U}_s\rangle.$$

(ii) By Algorithm 1 and the first claim of Lemma 7, one has

$$\langle \mathcal{R}_k, \mathcal{Z}_s\rangle = \langle \mathcal{R}_k, \mathcal{A}*_N\mathcal{R}_s\rangle = \langle \mathcal{R}_k, \mathcal{A}*_N(\mathcal{P}_s - \beta_{s-1}\mathcal{P}_{s-1})\rangle = \langle \mathcal{R}_k, \mathcal{U}_s\rangle - \beta_{s-1}\langle \mathcal{R}_k, \mathcal{U}_{s-1}\rangle = 0,$$

which completes the proof. ■

Lemma 9. *Let the tensor sequences $\{\mathcal{U}_k\}$, $\{\mathcal{R}_k\}$, and $\{\mathcal{P}_k\}$ be generated by Algorithm 1. If $\mathcal{R}_k \neq \mathcal{O}$, then $\mathcal{U}_k \neq \mathcal{O}$ and $\mathcal{P}_k \neq \mathcal{O}$.*

Proof. By the second claim of Lemma 7 and Algorithm 1, we obtain

$$\langle \mathcal{R}_k, \mathcal{U}_k\rangle = \langle \mathcal{R}_k, \mathcal{Z}_k\rangle = \langle \mathcal{R}_k, \mathcal{A}*_N\mathcal{R}_k\rangle. \tag{26}$$

Since $\mathcal{R}_k \neq \mathcal{O}$ and $\mathcal{A}$ is symmetric positive definite, we have from Lemma 4 that $\langle \mathcal{R}_k, \mathcal{A}*_N\mathcal{R}_k\rangle > 0$. It then follows from (26) that $\mathcal{U}_k \neq \mathcal{O}$.

By Algorithm 1, we know $\mathcal{U}_k = \mathcal{A}*_N\mathcal{P}_k$. It then follows from Proposition 2 that

$$\Phi_{IJ}(\mathcal{U}_k) = \Phi_{II}(\mathcal{A})\Phi_{IJ}(\mathcal{P}_k).$$

Since $\mathcal{A}$ is symmetric positive definite, we have from Definition 2 that $\Phi_{II}(\mathcal{A})$ is symmetric positive definite. As $\mathcal{U}_k \neq \mathcal{O}$, we get $\Phi_{IJ}(\mathcal{U}_k) \neq O$. Hence, $\Phi_{IJ}(\mathcal{P}_k) = [\Phi_{II}(\mathcal{A})]^{-1}\Phi_{IJ}(\mathcal{U}_k) \neq O$. Therefore, we conclude that $\mathcal{P}_k \neq \mathcal{O}$, which completes the proof. ■

Lemma 10. *For any initial tensor $\mathcal{X}_0$, we have*

$$\|\mathcal{R}_{k+1}\| = \|\mathcal{C} - \mathcal{A}*_N\mathcal{X}_{k+1}\| = \min_{\mathcal{X}\in\mathfrak{F}_k}\|\mathcal{C} - \mathcal{A}*_N\mathcal{X}\|, \tag{27}$$

where $\mathcal{X}_{k+1}$ is generated by Algorithm 1 at the $(k+1)$th iterative step and $\mathfrak{F}_k$ denotes an affine space which has the following form

$$\mathfrak{F}_k = \mathcal{X}_0 + \mathrm{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_k\} = \mathcal{X}_0 + \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_k\}, \tag{28}$$

where $\mathcal{P}_k$ and $\mathcal{R}_k$ are generated by Algorithm 1 at the kth iterative step.

Proof. According to the expression in Equation (28), for any $\mathcal{X} \in \mathfrak{F}_k$, there exist numbers $\gamma_0, \gamma_1, \cdots, \gamma_k$ such that

$$\mathcal{X} = \mathcal{X}_0 + \sum_{i=0}^{k}\gamma_i\mathcal{P}_i. \tag{29}$$

Now, we introduce a scalar function $f(\gamma_0, \gamma_1, \cdots, \gamma_k)$ by formula

$$f(\gamma_0, \gamma_1, \cdots, \gamma_k) = \|\mathcal{C} - \mathcal{A}*_N\mathcal{X}\|^2. \tag{30}$$

Substituting Equation (29) into Equation (30) yields

$$f(\gamma_0,\gamma_1,\cdots,\gamma_k)=\|\mathcal{C}-\mathcal{A}*_N(\mathcal{X}_0+\sum_{i=0}^{k}\gamma_i\mathcal{P}_i)\|^2=\|\mathcal{R}_0-\sum_{i=0}^{k}\gamma_i\mathcal{U}_i\|^2$$
$$=\langle\mathcal{R}_0-\sum_{i=0}^{k}\gamma_i\mathcal{U}_i,\mathcal{R}_0-\sum_{i=0}^{k}\gamma_i\mathcal{U}_i\rangle=\|\mathcal{R}_0\|^2-2\sum_{i=0}^{k}\gamma_i\langle\mathcal{R}_0,\mathcal{U}_i\rangle+\sum_{i=0}^{k}\gamma_i^2\|\mathcal{U}_i\|^2,$$

where the last equality follows from the third claim of Lemma 7.

Obviously, $f(\gamma_0,\gamma_1,\cdots,\gamma_k)$ is a continuous and differentiable function with respect to the variables $\gamma_0,\gamma_1,\cdots,\gamma_k$. Next, we minimize the function $f(\gamma_0,\gamma_1,\cdots,\gamma_k)$. Clearly, the minimum of this function occurs when

$$\frac{\partial f(\gamma_0,\gamma_1,\cdots,\gamma_k)}{\gamma_i}=2\gamma_i\|\mathcal{U}_i\|^2-2\langle\mathcal{R}_0,\mathcal{U}_i\rangle=0. \tag{31}$$

Solving Equation (31) gives

$$\gamma_i=\frac{\langle\mathcal{R}_0,\mathcal{U}_i\rangle}{\langle\mathcal{U}_i,\mathcal{U}_i\rangle},\quad i=0,1,2,\cdots,k. \tag{32}$$

According to the definition of α_i and the second claim of Lemma 8, we know

$$\alpha_i=\frac{\langle\mathcal{R}_i,\mathcal{U}_i\rangle}{\langle\mathcal{U}_i,\mathcal{U}_i\rangle}=\frac{\langle\mathcal{R}_0,\mathcal{U}_i\rangle}{\langle\mathcal{U}_i,\mathcal{U}_i\rangle}=\gamma_i,\quad i=0,1,2,\cdots,k.$$

This implies that $\mathcal{X}_{k+1}=\mathcal{X}_0+\sum_{i=0}^{k}\alpha_i\mathcal{P}_i$ minimizes the residual norm in the affine space $\mathfrak{F}_k$. The proof is completed. ■

Theorem 1. *If the coefficient tensor $\mathcal{A}\in\mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ is symmetric positive definite, then for any initial tensor $\mathcal{X}_0\in\mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, the solution of the tensor equation (13) can be derived in at most IJ iterative steps by Algorithm 1.*

Proof. Assume that $\mathcal{R}_k\neq\mathcal{O}$ for $k=0,1,2,\cdots,IJ-1$. It follows from Lemma 9 that $\mathcal{U}_k\neq\mathcal{O}$ and $\mathcal{P}_k\neq\mathcal{O}$ for all $k=0,1,2,\cdots,IJ-1$. Then $\mathcal{U}_{IJ}$ and $\mathcal{P}_{IJ}$ can be derived by Algorithm 1. According to Lemma 7, we know that $\langle\mathcal{U}_i,\mathcal{U}_j\rangle=0$ for all $i,j=0,1,2,\cdots,IJ-1$, and $i\neq j$. So the tensor sequence of $\mathcal{U}_0,\mathcal{U}_1,\cdots,\mathcal{U}_{IJ-1}$ is an orthogonal basis in $\mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$. This implies that there exist some scalars $\xi_0,\cdots,\xi_{IJ-1}$ such that

$$\mathcal{R}_0=\xi_0\mathcal{U}_0+\cdots+\xi_{IJ-1}\mathcal{U}_{IJ-1}=\sum_{i=0}^{IJ-1}\xi_i\mathcal{U}_i.$$

It then follows from Lemma 10 that

$$\|\mathcal{R}_{IJ}\|=\|\mathcal{C}-\mathcal{A}*_N\mathcal{X}_{IJ}\|=\min_{\mathcal{X}\in\mathfrak{F}_{IJ-1}}\|\mathcal{C}-\mathcal{A}*_N\mathcal{X}\|=\min_{\gamma_i}\|\mathcal{R}_0-\sum_{i=0}^{IJ-1}\gamma_i\mathcal{U}_i\|=\|\mathcal{R}_0-\sum_{i=0}^{IJ-1}\xi_i\mathcal{U}_i\|=0,$$

which implies that $\mathcal{R}_{IJ}=\mathcal{O}$. The proof is completed. ■

4 | THE TENSOR FORM OF GENERALIZED CONJUGATE RESIDUAL ALGORITHM AND CONVERGENCE ANALYSIS

In this section, motivated by Elman,[40] we propose the tensor form of the generalized conjugate residual algorithm to solve the tensor equation (13) (denoted by GCR-BTF), where the coefficient tensor $\mathcal{A}\in\mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ needs only to be positive definite but not symmetric. The details of the GCR-BTF method are listed as in Algorithm 2.

Algorithm 2. The GCR-BTF method

Given the positive definite coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ and the right-hand side tensor $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$.

1: **Input**: the initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$,the maximum number of iterations $\mathrm{Iter}_{\max}$, the tolerance error $\varepsilon>0$.
2: Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$;
3: $\mathcal{P}_0 = \mathcal{R}_0$;
4: $\mathcal{Z}_0 = \mathcal{A} *_N \mathcal{R}_0$;
5: $k = 0$;
6: **while** ($k< \mathrm{Iter}_{\max}$)
7: $\quad \mathcal{U}_k = \mathcal{A} *_N \mathcal{P}_k$;
8: $\quad \alpha_k = \dfrac{\langle \mathcal{R}_k, \mathcal{U}_k\rangle}{\langle \mathcal{U}_k, \mathcal{U}_k\rangle}$;
9: $\quad \mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k \mathcal{P}_k$;
10: $\quad \mathcal{R}_{k+1} = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_{k+1} = \mathcal{R}_k - \alpha_k \mathcal{U}_k$;
11: $\quad$ **if** $\|\mathcal{R}_{k+1}\| < \varepsilon$, then **stop**
12: $\quad \mathcal{Z}_{k+1} = \mathcal{A} *_N \mathcal{R}_{k+1}$;
13: $\quad \beta_s^{(k)} = -\dfrac{\langle \mathcal{Z}_{k+1}, \mathcal{U}_s\rangle}{\langle \mathcal{U}_s, \mathcal{U}_s\rangle}, s = 0, 1, 2, \ldots, k$;
14: $\quad \mathcal{P}_{k+1} = \mathcal{R}_{k+1} + \sum_{s=0}^{k} \beta_s^{(k)} \mathcal{P}_s$;
15: **end while**

Remark 2. In order to save memory requirements, we can calculate $\mathcal{U}_{k+1}$ by using line 14 of Algorithm 2. Obviously, we have

$$\mathcal{U}_{k+1} = \mathcal{Z}_{k+1} + \sum_{s=0}^{k} \beta_s^{(k)} \mathcal{U}_s. \tag{33}$$

Lemma 11. *Let the sequence* $\{\mathcal{U}_k\}$ *be generated by Algorithm 2. Then*

$$\langle \mathcal{U}_i, \mathcal{U}_j\rangle = 0, \quad i,j = 0, 1, 2, \cdots, k,\ i \neq j.$$

Proof. First, we prove

$$\langle \mathcal{U}_i, \mathcal{U}_j\rangle = 0, \quad 0 \le i < j \le k. \tag{34}$$

By mathematical induction, for $k=1$, by the update rules of $\mathcal{U}_1$ and $\mathcal{P}_1$, the definition of $\beta_0^{(0)}$ and Lemma 1, we obtain

$$\begin{aligned}\langle \mathcal{U}_0, \mathcal{U}_1\rangle &= \langle \mathcal{U}_0, \mathcal{A} *_N \mathcal{P}_1\rangle = \langle \mathcal{U}_0, \mathcal{A} *_N (\mathcal{R}_1 + \beta_0^{(0)} \mathcal{P}_0)\rangle \\ &= \langle \mathcal{U}_0, \mathcal{Z}_1\rangle + \beta_0^{(0)} \langle \mathcal{U}_0, \mathcal{U}_0\rangle = \langle \mathcal{U}_0, \mathcal{Z}_1\rangle - \frac{\langle \mathcal{Z}_1, \mathcal{U}_0\rangle}{\langle \mathcal{U}_0, \mathcal{U}_0\rangle}\langle \mathcal{U}_0, \mathcal{U}_0\rangle = 0.\end{aligned}$$

Suppose that (34) holds for $k=l$. For $k=l+1$, by the update rules of $\mathcal{U}_{l+1}$ and $\mathcal{P}_{l+1}$, the definition of $\beta_i^{(l)}$ and the induction principle, we have

$$\begin{aligned}\langle \mathcal{U}_i, \mathcal{U}_{l+1}\rangle &= \langle \mathcal{U}_i, \mathcal{A} *_N \mathcal{P}_{l+1}\rangle = \langle \mathcal{U}_i, \mathcal{A} *_N (\mathcal{R}_{l+1} + \sum_{s=0}^{l} \beta_s^{(l)} \mathcal{P}_s)\rangle \\ &= \langle \mathcal{U}_i, \mathcal{A} *_N \mathcal{R}_{l+1}\rangle + \sum_{s=0}^{l} \beta_s^{(l)} \langle \mathcal{U}_i, \mathcal{A} *_N \mathcal{P}_s\rangle = \langle \mathcal{U}_i, \mathcal{Z}_{l+1}\rangle + \sum_{s=0}^{l} \beta_s^{(l)} \langle \mathcal{U}_i, \mathcal{U}_s\rangle \\ &= \langle \mathcal{U}_i, \mathcal{Z}_{l+1}\rangle + \beta_i^{(l)} \langle \mathcal{U}_i, \mathcal{U}_i\rangle = \langle \mathcal{U}_i, \mathcal{Z}_{l+1}\rangle - \frac{\langle \mathcal{Z}_{l+1}, \mathcal{U}_i\rangle}{\langle \mathcal{U}_i, \mathcal{U}_i\rangle}\langle \mathcal{U}_i, \mathcal{U}_i\rangle = 0.\end{aligned}$$

So the relation (34) holds for $k = l + 1$. By the induction principle, the relation (34) holds for all $0 \le i < j \le k$. For $i > j$, by Lemma 1, we have $\langle \mathcal{U}_i, \mathcal{U}_j \rangle = \langle \mathcal{U}_j, \mathcal{U}_i \rangle = 0$, which completes the proof. ■

Lemma 12. *Let the sequences* $\{\mathcal{U}_k\}$, $\{\mathcal{R}_k\}$ *and* $\{\mathcal{Z}_k\}$ *be generated by Algorithm 2. Then*

(i) $\langle \mathcal{R}_i, \mathcal{U}_j \rangle = 0,\ i, j = 0, 1, 2, \cdots, k,\ i > j$.

(ii) $\langle \mathcal{R}_i, \mathcal{U}_j \rangle = \langle \mathcal{R}_0, \mathcal{U}_j \rangle,\ i, j = 0, 1, 2, \cdots, k,\ i \le j$.

(iii) $\langle \mathcal{R}_i, \mathcal{U}_i \rangle = \langle \mathcal{R}_i, \mathcal{Z}_i \rangle$.

(iv) $\langle \mathcal{R}_i, \mathcal{Z}_j \rangle = 0,\ i, j = 0, 1, 2, \cdots, k,\ i > j$.

Proof. (i) We apply mathematical induction. For $i = 1$, by the update rule of $\mathcal{R}_1$ and the definition of α_0, one has

$$\langle \mathcal{R}_1, \mathcal{U}_0 \rangle = \langle \mathcal{R}_0 - \alpha_0 \mathcal{U}_0, \mathcal{U}_0 \rangle = \langle \mathcal{R}_0, \mathcal{U}_0 \rangle - \alpha_0 \langle \mathcal{U}_0, \mathcal{U}_0 \rangle = \langle \mathcal{R}_0, \mathcal{U}_0 \rangle - \frac{\langle \mathcal{R}_0, \mathcal{U}_0 \rangle}{\langle \mathcal{U}_0, \mathcal{U}_0 \rangle} \langle \mathcal{U}_0, \mathcal{U}_0 \rangle = 0.$$

Assume that $\langle \mathcal{R}_l, \mathcal{U}_j \rangle = 0$ for all $l > j$. We shall show that $\langle \mathcal{R}_{l+1}, \mathcal{U}_j \rangle = 0$ for all $l + 1 > j$. According to the update rule of $\mathcal{R}_{l+1}$, we get

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_j \rangle = \langle \mathcal{R}_l - \alpha_l \mathcal{U}_l, \mathcal{U}_j \rangle = \langle \mathcal{R}_l, \mathcal{U}_j \rangle - \alpha_l \langle \mathcal{U}_l, \mathcal{U}_j \rangle.$$

When $j < l$, by Lemma 11 and the induction principle, we know $\langle \mathcal{R}_{l+1}, \mathcal{U}_j \rangle = 0$.

When $j = l$, it follows from the definition of α_l that

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_l \rangle = \langle \mathcal{R}_l, \mathcal{U}_l \rangle - \alpha_l \langle \mathcal{U}_l, \mathcal{U}_l \rangle = \langle \mathcal{R}_l, \mathcal{U}_l \rangle - \frac{\langle \mathcal{R}_l, \mathcal{U}_l \rangle}{\langle \mathcal{U}_l, \mathcal{U}_l \rangle} \langle \mathcal{U}_l, \mathcal{U}_l \rangle = 0.$$

Thus, we draw the conclusion by the induction principle.

(ii) For $i = 0$, the result is trivial. Assume now that $\langle \mathcal{R}_l, \mathcal{U}_j \rangle = \langle \mathcal{R}_0, \mathcal{U}_j \rangle$ for all $l < j$. According to Algorithm 2 and Lemma 11, we have

$$\langle \mathcal{R}_{l+1}, \mathcal{U}_j \rangle = \langle \mathcal{R}_l - \alpha_l \mathcal{U}_l, \mathcal{U}_j \rangle = \langle \mathcal{R}_l, \mathcal{U}_j \rangle - \alpha_l \langle \mathcal{U}_l, \mathcal{U}_j \rangle = \langle \mathcal{R}_l, \mathcal{U}_j \rangle = \cdots = \langle \mathcal{R}_0, \mathcal{U}_j \rangle.$$

By the induction principle, we obtain this result.

(iii) By Remark 2 and the first claim of Lemma 12, one has

$$\langle \mathcal{R}_i, \mathcal{U}_i \rangle = \langle \mathcal{R}_i, \mathcal{Z}_i + \sum_{s=0}^{i-1} \beta_s^{(i)} \mathcal{U}_s \rangle = \langle \mathcal{R}_i, \mathcal{Z}_i \rangle + \sum_{s=0}^{i-1} \beta_s^{(i)} \langle \mathcal{R}_i, \mathcal{U}_s \rangle = \langle \mathcal{R}_i, \mathcal{Z}_i \rangle.$$

(iv) By the update rules of $\mathcal{Z}_j$ and $\mathcal{P}_j$ and the first claim of Lemma 12, we have

$$\begin{aligned}
\langle \mathcal{R}_i, \mathcal{Z}_j \rangle &= \langle \mathcal{R}_i, \mathcal{A} *_N \mathcal{R}_j \rangle = \langle \mathcal{R}_i, \mathcal{A} *_N (\mathcal{P}_j - \sum_{s=0}^{j-1} \beta_s^{(j-1)} \mathcal{P}_s) \rangle \\
&= \langle \mathcal{R}_i, \mathcal{A} *_N \mathcal{P}_j \rangle - \langle \mathcal{R}_i, \sum_{s=0}^{j-1} \beta_s^{(j-1)} \mathcal{A} *_N \mathcal{P}_s \rangle \\
&= \langle \mathcal{R}_i, \mathcal{U}_j \rangle - \sum_{s=0}^{j-1} \beta_s^{(j-1)} \langle \mathcal{R}_i, \mathcal{U}_s \rangle = 0, \quad \forall i > j,
\end{aligned}$$

which completes the proof. ■

Similar to Lemmas 9 and 10 and Theorem 1, we can derive the following results by using Lemmas 11 and 12.

Lemma 13. *Let the tensor sequences* $\{\mathcal{U}_k\}$, $\{\mathcal{R}_k\}$, *and* $\{\mathcal{P}_k\}$ *be generated by Algorithm 2. If* $\mathcal{R}_k \neq \mathcal{O}$, *then* $\mathcal{U}_k \neq \mathcal{O}$ *and* $\mathcal{P}_k \neq \mathcal{O}$.

Proof. This result can be derived by using the third claim of Lemma 12, which is similar to Lemma 9. ∎

Lemma 14. *For any initial tensor $\mathcal{X}_0$, we have*

$$\|\mathcal{R}_{k+1}\| = \|\mathcal{C} - \mathcal{A}*_N\mathcal{X}_{k+1}\| = \min_{\mathcal{X}\in\mathfrak{F}_k}\|\mathcal{C} - \mathcal{A}*_N\mathcal{X}\|,$$

where $\mathcal{X}_{k+1}$ is generated by Algorithm 2 at the $(k+1)$th iterative step and $\mathfrak{F}_k$ denotes an affine space which has the following form

$$\mathfrak{F}_k = \mathcal{X}_0 + \text{span}\{\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_k\} = \mathcal{X}_0 + \text{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_k\},$$

where $\mathcal{P}_k$ and $\mathcal{R}_k$ are generated by Algorithm 2 at the kth iterative step.

Proof. By the second claim of Lemma 12, we can establish this result. ∎

Theorem 2. *If the coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ is positive definite, then for any initial tensor $\mathcal{X}_0$, the solution of the tensor equation (13) can be derived in at most IJ iterative steps by Algorithm 2.*

Proof. We can prove this result by using Lemmas 13 and 14. ∎

5 | PROJECTION METHODS

The tensor forms of conjugate residual and generalized conjugate residual algorithms are all based on the minimal residual norm condition. In this section, we will derive other efficient iterative algorithms to solve the tensor equation (13). Different from the CR-BTF and GCR-BTF methods, we no longer restrict the coefficient tensor $\mathcal{A}$ to be symmetric positive definite or positive definite.

For coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ and tensor $\mathcal{V} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$, the kth tensor Krylov subspace associated to the pair $(\mathcal{A}, \mathcal{V})$ is defined by

$$\mathcal{K}_k(\mathcal{A}, \mathcal{V}) = \text{span}\left\{\mathcal{V}, \mathcal{A}*_N\mathcal{V}, \cdots, \mathcal{A}^{k-1}*_N\mathcal{V}\right\}, \tag{35}$$

where $\mathcal{A}^i*_N\mathcal{V} = \mathcal{A}*_N(\mathcal{A}^{i-1}*_N\mathcal{V})$ and $\mathcal{A}^0*_N\mathcal{V} = \mathcal{V}$.

Let $\mathcal{X}_0$ be a given initial approximation tensor and $\mathcal{R}_0 = \mathcal{C} - \mathcal{A}*_N\mathcal{X}_0$ be the corresponding residual. At the kth step of the projection method, we find an approximate solution $\mathcal{X}_k \in \mathcal{X}_0 + \mathcal{K}_k(\mathcal{A}, \mathcal{R}_0)$ such that

$$\mathcal{R}_k \perp \mathfrak{K}_k, \tag{36}$$

where $\mathcal{R}_k = \mathcal{C} - \mathcal{A}*_N\mathcal{X}_k$ and $\mathfrak{K}_k$ is a properly selected k-dimensional tensor subspace. Assume that $\mathfrak{K}_k = \text{span}\{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_k\}$ and $\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0) = \text{span}\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k\}$. Let $\mathbb{V}_k$ be the $(M+N+1)$-mode tensor with frontal slices $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k$, $\mathbb{W}_k$ be the $(M+N+1)$-mode tensor with frontal slices $\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_k$, and $\mathbb{G}_k$ be the $(M+N+1)$-mode tensor with frontal slices $\mathcal{A}*_N\mathcal{V}_1, \mathcal{A}*_N\mathcal{V}_2, \cdots, \mathcal{A}*_N\mathcal{V}_m$, respectively. That is,

$$\mathbb{V}_k := [\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k], \tag{37}$$

$$\mathbb{W}_k := [\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_k], \tag{38}$$

$$\mathbb{G}_k := [\mathcal{A}*_N\mathcal{V}_1, \mathcal{A}*_N\mathcal{V}_2, \cdots, \mathcal{A}*_N\mathcal{V}_k]. \tag{39}$$

Since the tensor sequence of $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k$ forms an orthonormal basis for $\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0)$, we have

$$\mathcal{X}_k = \mathcal{X}_0 + \mathbb{V}_k\overline{\times}_{(M+N+1)}y_k, \tag{40}$$

where $y_k \in \mathbb{R}^k$. It then follows from (39) that

$$\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_k = \mathcal{C} - \mathcal{A} *_N (\mathcal{X}_0 + \mathbb{V}_k \overline{\times}_{(M+N+1)} y_k) = \mathcal{R}_0 - \mathbb{G}_k \overline{\times}_{(M+N+1)} y_k, \tag{41}$$

which, together with (36), yields

$$\begin{cases} 0 = \langle \mathcal{W}_1, \mathcal{R}_k \rangle = \langle \mathcal{W}_1, \mathcal{R}_0 \rangle - \langle \mathcal{W}_1, \mathbb{G}_k \overline{\times}_{(M+N+1)} y_k \rangle, \\ 0 = \langle \mathcal{W}_2, \mathcal{R}_k \rangle = \langle \mathcal{W}_2, \mathcal{R}_0 \rangle - \langle \mathcal{W}_2, \mathbb{G}_k \overline{\times}_{(M+N+1)} y_k \rangle, \\ \cdots \quad \cdots \\ 0 = \langle \mathcal{W}_k, \mathcal{R}_k \rangle = \langle \mathcal{W}_k, \mathcal{R}_0 \rangle - \langle \mathcal{W}_k, \mathbb{G}_k \overline{\times}_{(M+N+1)} y_k \rangle. \end{cases} \tag{42}$$

Hence, by Definition 5, Lemma 5, and the relation (38), we conclude that

$$\mathbb{W}_k \boxtimes^{(M+N+1)} \mathcal{R}_k = \mathbb{W}_k \boxtimes^{(M+N+1)} (\mathbb{G}_k \overline{\times}_{(M+N+1)} y_k) = (\mathbb{W}_k \boxtimes^{(M+N+1)} \mathbb{G}_k) \overline{\times}_{(M+N+1)} y_k. \tag{43}$$

From the geometric point of view, if $\mathbb{W}_k \boxtimes^{(M+N+1)} \mathbb{V}_k = I_k$, then $\mathbb{W}_k \boxtimes^{(M+N+1)} \mathbb{G}_k$ is the projection of $\mathcal{A}$ along $\mathfrak{K}_k^{\perp}$ onto the $\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0)$. So, we say (43) is the projection equation and this kind of iterative method is the projection method.

Take $\mathfrak{K}_k = \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0)$, where $\langle \mathcal{R}_0, \tilde{\mathcal{R}}_0 \rangle \neq 0$. Then, we can establish the following three projection methods: the tensor form of the biconjugate gradient method (denoted by BiCG-BTF), the tensor form of the conjugate gradient squared method (denoted by CGS-BTF), and the tensor form of the biconjugate gradient stabilized method (denoted by BiCGSTAB-BTF).

5.1 | The BiCG-BTF method

The BiCG-BTF method is a basic projection method, where $\mathfrak{K}_k = \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0)$ and $\langle \mathcal{R}_0, \tilde{\mathcal{R}}_0 \rangle \neq 0$. Of course, we can apply Lanczos method to generate bases of tensor Krylov subspaces $\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0)$ and $\mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0)$. However, the BiCG-BTF method is closely related to the tensor form of the conjugate gradient method.[26] So, we can derive the basic iteration format directly by using the idea of conjugate gradient method. The main steps of the BiCG-BTF method are described as in Algorithm 3.

Algorithm 3. The BiCG-BTF method

Given the coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_1 \times \cdots \times I_N}$ and the right-hand side tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$.

Input: the initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$, the tolerance error $\varepsilon > 0$.
Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$;
$\mathcal{Q}_{-1} = \widetilde{\mathcal{Q}}_{-1} = \mathcal{O}$, $\rho_{-1} = 1$;
Choose $\widetilde{\mathcal{R}}_0$(e.g., $\widetilde{\mathcal{R}}_0 = \mathcal{R}_0$) such that $\langle \mathcal{R}_0, \widetilde{\mathcal{R}}_0 \rangle \neq 0$;
$k = 0$;
while $(\|\mathcal{R}_k\| / \|\mathcal{R}_0\| > \varepsilon)$
  $\rho_k = \langle \widetilde{\mathcal{R}}_k, \mathcal{R}_k \rangle$;
  $\beta_k = \rho_k / \rho_{k-1}$;
  $\mathcal{Q}_k = \mathcal{R}_k + \beta_k \mathcal{Q}_{k-1}$;
  $\widetilde{\mathcal{Q}}_k = \widetilde{\mathcal{R}}_k + \beta_k \widetilde{\mathcal{Q}}_{k-1}$;
  $\sigma_k = \langle \widetilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle$;
  $\alpha_k = \rho_k / \sigma_k$;
  $\mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k \mathcal{Q}_k$;
  $\mathcal{R}_{k+1} = \mathcal{R}_k - \alpha_k \mathcal{A} *_N \mathcal{Q}_k$;
  $\widetilde{\mathcal{R}}_{k+1} = \widetilde{\mathcal{R}}_k - \alpha_k \mathcal{A}^T *_N \widetilde{\mathcal{Q}}_k$;
  $k = k + 1$;
end

Now, we give the convergence analysis of the BiCG-BTF method.

Theorem 3. *Let the sequences $\{\mathcal{Q}_k\}$, $\{\tilde{\mathcal{Q}}_k\}$, $\{\mathcal{R}_k\}$, and $\{\tilde{\mathcal{R}}_k\}$ be generated by Algorithm 3. Then*

$$\langle \mathcal{R}_k, \tilde{\mathcal{R}}_l \rangle = \langle \tilde{\mathcal{R}}_k, \mathcal{R}_l \rangle = \rho_k \delta_{kl}, \tag{44}$$

$$\langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_l \rangle = \langle \mathcal{Q}_k, \mathcal{A} *_N \tilde{\mathcal{Q}}_l \rangle = \sigma_k \delta_{kl}, \tag{45}$$

$$\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0) = \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{k-1}\} = \mathrm{span}\{\mathcal{Q}_0, \mathcal{Q}_1, \cdots, \mathcal{Q}_{k-1}\}, \tag{46}$$

$$\mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0) = \mathrm{span}\{\tilde{\mathcal{R}}_0, \tilde{\mathcal{R}}_1, \cdots, \tilde{\mathcal{R}}_{k-1}\} = \mathrm{span}\{\tilde{\mathcal{Q}}_0, \tilde{\mathcal{Q}}_1, \cdots, \tilde{\mathcal{Q}}_{k-1}\}. \tag{47}$$

Proof. We first prove (44) and (45) by mathematic induction. When $k=0$, the result is trivial. Assume now that the relations (44) and (45) are true for some k. Then for $k+1$, by lines 14 and 15 of Algorithm 3, we have

$$\langle \mathcal{R}_{k+1}, \tilde{\mathcal{R}}_l \rangle = \langle \mathcal{R}_k, \tilde{\mathcal{R}}_l \rangle - \alpha_k \langle \mathcal{A} *_N \mathcal{Q}_k, \tilde{\mathcal{R}}_l \rangle = \langle \mathcal{R}_k, \tilde{\mathcal{R}}_l \rangle - \alpha_k \langle \mathcal{A} *_N \mathcal{Q}_k, \tilde{\mathcal{Q}}_l - \beta_l \tilde{\mathcal{Q}}_{l-1} \rangle. \tag{48}$$

When $l=k$, by the definition of α_k, Lemma 1 and the induction principle, we know

$$\begin{aligned}
\langle \mathcal{R}_{k+1}, \tilde{\mathcal{R}}_k \rangle &= \langle \mathcal{R}_k, \tilde{\mathcal{R}}_k \rangle - \frac{\langle \tilde{\mathcal{R}}_k, \mathcal{R}_k \rangle}{\langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle} \langle \mathcal{A} *_N \mathcal{Q}_k, \tilde{\mathcal{Q}}_k - \beta_k \tilde{\mathcal{Q}}_{k-1} \rangle \\
&= \langle \mathcal{R}_k, \tilde{\mathcal{R}}_k \rangle - \langle \tilde{\mathcal{R}}_k, \mathcal{R}_k \rangle = 0.
\end{aligned}$$

When $l<k$, by the induction principle, we immediately have $\langle \mathcal{R}_{k+1}, \tilde{\mathcal{R}}_l \rangle = 0$. Similarly, we can prove that $\langle \tilde{\mathcal{R}}_{k+1}, \mathcal{R}_l \rangle = 0\,(l \le k)$. Hence, the relation (44) is true.

By lines 10 and 14 of Algorithm 3, we have

$$\begin{aligned}
\langle \tilde{\mathcal{Q}}_{k+1}, \mathcal{A} *_N \mathcal{Q}_l \rangle &= \langle \tilde{\mathcal{R}}_{k+1} + \beta_{k+1} \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_l \rangle = \langle \tilde{\mathcal{R}}_{k+1}, \mathcal{A} *_N \mathcal{Q}_l \rangle + \beta_{k+1} \langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_l \rangle \\
&= \langle \tilde{\mathcal{R}}_{k+1}, \frac{\mathcal{R}_l - \mathcal{R}_{l+1}}{\alpha_l} \rangle + \beta_{k+1} \langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_l \rangle.
\end{aligned} \tag{49}$$

When $l=k$, by the definitions of α_k and β_{k+1} and the induction principle, we know

$$\begin{aligned}
\langle \tilde{\mathcal{Q}}_{k+1}, \mathcal{A} *_N \mathcal{Q}_k \rangle &= \langle \tilde{\mathcal{R}}_{k+1}, \frac{\mathcal{R}_k - \mathcal{R}_{k+1}}{\alpha_k} \rangle + \beta_{k+1} \langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle \\
&= -\frac{1}{\alpha_k} \langle \tilde{\mathcal{R}}_{k+1}, \mathcal{R}_{k+1} \rangle + \beta_{k+1} \langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle \\
&= -\frac{\langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle}{\langle \tilde{\mathcal{R}}_k, \mathcal{R}_k \rangle} \langle \tilde{\mathcal{R}}_{k+1}, \mathcal{R}_{k+1} \rangle + \frac{\langle \tilde{\mathcal{R}}_{k+1}, \mathcal{R}_{k+1} \rangle}{\langle \tilde{\mathcal{R}}_k, \mathcal{R}_k \rangle} \langle \tilde{\mathcal{Q}}_k, \mathcal{A} *_N \mathcal{Q}_k \rangle \\
&= 0.
\end{aligned}$$

When $l<k$, by the relation (44) and the induction principle, we immediately have $\langle \tilde{\mathcal{Q}}_{k+1}, \mathcal{A} *_N \mathcal{Q}_l \rangle = 0$. Similarly, we can prove that $\langle \mathcal{Q}_{k+1}, \mathcal{A} *_N \tilde{\mathcal{Q}}_l \rangle = 0\,(l \le k)$. Hence, the relation (45) is true.

Similar to Lemma 6, it is easy to see that

$$\mathcal{K}_k(\mathcal{A}, \mathcal{R}_0) = \mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{k-1}\}, \quad \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0) = \mathrm{span}\{\tilde{\mathcal{R}}_0, \tilde{\mathcal{R}}_1, \cdots, \tilde{\mathcal{R}}_{k-1}\}. \tag{50}$$

Meanwhile, it follows from Algorithm 3 that

$$\mathrm{span}\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_{k-1}\} = \mathrm{span}\{\mathcal{Q}_0, \mathcal{Q}_1, \cdots, \mathcal{Q}_{k-1}\}, \quad \mathrm{span}\{\tilde{\mathcal{R}}_0, \tilde{\mathcal{R}}_1, \cdots, \tilde{\mathcal{R}}_{k-1}\} = \mathrm{span}\{\tilde{\mathcal{Q}}_0, \tilde{\mathcal{Q}}_1, \cdots, \tilde{\mathcal{Q}}_{k-1}\},$$

which, together with (50), implies that (46) and (47) are true. The proof is completed. ∎

Remark 3. According to the first and fourth claims of Theorem 3, we have $\mathcal{R}_k \perp \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0)$.

By Theorem 3, it is easy to see that there are some polynomials of order k such that

$$\mathcal{R}_k = \phi_k(\mathcal{A}) *_N \mathcal{R}_0, \quad \tilde{\mathcal{R}}_k = \phi_k(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0, \tag{51}$$

$$\mathcal{Q}_k = \psi_k(\mathcal{A}) *_N \mathcal{R}_0, \quad \tilde{\mathcal{Q}}_k = \psi_k(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0. \tag{52}$$

Denote $\phi_0(t) = 1$, $\psi_{-1}(t) = 0$ and $\vartheta(t) = t$. Let $\mathfrak{P}_k$ be the set of the k degree polynomials over the real field $\mathbb{R}$. Now, we define the following bilinear functional in linear spaces $\mathfrak{P}_k$:

$$[\phi, \psi] = \langle \phi(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0, \psi(\mathcal{A}) *_N \mathcal{R}_0 \rangle, \quad \forall \phi, \psi \in \mathfrak{P}_k.$$

Then, the BiCG-BTF method can be equivalently described as follows.

$$(1)\ \phi_0 = 1,\ \psi_{-1} = 0,\ \rho_{-1} = 1, \tag{53a}$$

$$(2)\ \rho_k = [\phi_k, \phi_k],\ \beta_k = \rho_k / \rho_{k-1}, \tag{53b}$$

$$\psi_k = \phi_k + \beta_k \psi_{k-1}, \tag{53c}$$

$$\sigma_k = [\psi_k, \vartheta\psi_k], \quad \alpha_k = \rho_k / \sigma_k; \tag{53d}$$

$$\phi_{k+1} = \phi_k - \alpha_k \vartheta \psi_k, \quad k = 0, 1, 2, \cdots. \tag{53e}$$

We must comment here that the relation (53) is important for deriving the other two projection methods, that is, the CGS-BTF and BiCGSTAB-BTF methods.

5.2 | The CGS-BTF method

According to Algorithm 3, we find that the BiCG-BTF method uses $\mathcal{A}^T$ and increases the computing costs. Meanwhile, it is obvious that $\tilde{\mathcal{R}}_k$ needs the knowledge of $\mathcal{A}^T$. And, the knowledge of $\tilde{\mathcal{R}}_k$ are only needed in the computations of ρ_k and σ_k. In fact, the computations of ρ_k and σ_k can avoid the use of $\mathcal{A}^T$. Let

$$\begin{aligned}
\rho_k &= [\phi_k, \phi_k] = \langle \phi_k(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0, \phi_k(\mathcal{A}) *_N \mathcal{R}_0 \rangle \\
&= \langle \tilde{\mathcal{R}}_0, \phi_k(\mathcal{A}) *_N \phi_k(\mathcal{A}) *_N \mathcal{R}_0 \rangle = [\phi_0, \phi_k^2], \\
\sigma_k &= [\psi_k, \vartheta\psi_k] = \langle \psi_k(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0, \vartheta(\mathcal{A}) *_N \psi_k(\mathcal{A}) *_N \mathcal{R}_0 \rangle \\
&= \langle \tilde{\mathcal{R}}_0, \psi_k(\mathcal{A}) *_N \vartheta(\mathcal{A}) *_N \psi_k(\mathcal{A}) *_N \mathcal{R}_0 \rangle \\
&= \langle \tilde{\mathcal{R}}_0, \vartheta(\mathcal{A}) *_N \psi_k^2(\mathcal{A}) *_N \mathcal{R}_0 \rangle \\
&= [\psi_0, \vartheta\psi_k^2].
\end{aligned} \tag{54}$$

If we let

$$\mathcal{R}_k = \phi_k^2(\mathcal{A}) *_N \mathcal{R}_0, \quad \mathcal{Q}_k = \psi_k^2(\mathcal{A}) *_N \mathcal{R}_0,$$

by the relation (54), we immediately have

$$\rho_k = \langle \tilde{\mathcal{R}}_0, \mathcal{R}_k \rangle, \quad \sigma_k = \langle \tilde{\mathcal{R}}_0, \mathcal{A} *_N \mathcal{Q}_k \rangle.$$

At this case, the key to calculate $\mathcal{R}_k$ and $\mathcal{Q}_k$ is to calculate ϕ_k^2 and ψ_k^2 efficiently. According to (53), we get

$$\psi_k^2 = (\phi_k + \beta_k \psi_{k-1})^2 = \phi_k^2 + 2\beta_k \phi_k \psi_{k-1} + \beta_k^2 \psi_{k-1}^2, \tag{55}$$

$$\phi_{k+1}^2 = (\phi_k - \alpha_k \vartheta \psi_k)^2 = \phi_k^2 - 2\alpha_k \vartheta \phi_k \psi_k + \alpha_k^2 \vartheta^2 \psi_k^2. \tag{56}$$

In the relations (55) and (56), $\phi_k\psi_{k-1}$ and $\phi_k\psi_k$ are still unknown. Using the relation (53) again, we have

$$\phi_{k+1}\psi_k = \phi_k\psi_k - \alpha_k\vartheta\psi_k^2, \quad \phi_k\psi_k = \phi_k^2 + \beta_k\phi_k\psi_{k-1}. \tag{57}$$

Denote

$$\mathcal{P}_k = \phi_k(\mathcal{A})*_N\psi_{k-1}(\mathcal{A})*_N\mathcal{R}_0, \quad \mathcal{U}_k = \psi_k(\mathcal{A})*_N\psi_k(\mathcal{A})*_N\mathcal{R}_0.$$

It then follows from (55)–(57) that

$$\begin{aligned} \mathcal{Q}_k &= \mathcal{R}_k + 2\beta_k\mathcal{P}_k + \beta_k^2\mathcal{Q}_{k-1}, \\ \mathcal{R}_{k+1} &= \mathcal{R}_k - 2\alpha_k\mathcal{A}*_N\mathcal{U}_k + \alpha_k^2\mathcal{A}^2*_N\mathcal{Q}_k, \\ \mathcal{P}_{k+1} &= \mathcal{U}_k - \alpha_k\mathcal{A}*_N\mathcal{Q}_k, \\ \mathcal{U}_k &= \mathcal{R}_k + \beta_k\mathcal{P}_k. \end{aligned} \tag{58}$$

Furthermore, by the first and fourth formulas of (58), we get

$$\mathcal{Q}_k = \mathcal{U}_k + \beta_k(\mathcal{P}_k + \beta_k\mathcal{Q}_{k-1}). \tag{59}$$

And, by the second and third formulas of (58), we have

$$\mathcal{R}_{k+1} = \mathcal{R}_k - \alpha_k\mathcal{A}*_N(\mathcal{U}_k + \mathcal{P}_{k+1}), \tag{60}$$

which implies that if

$$\mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k(\mathcal{U}_k + \mathcal{P}_{k+1}), \tag{61}$$

then $\mathcal{R}_{k+1} = \mathcal{C} - \mathcal{A}*_N\mathcal{X}_{k+1}$ is exactly the residual of the corresponding approximate solution.

Hence, the main steps of the CGS-BTF method are summarized as in Algorithm 4.

Algorithm 4. The CGS-BTF method

Given the coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$ and the right-hand side tensor $\mathcal{C} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$.

Input: the initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_M}$,the tolerance error $\varepsilon>0$.
Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$;
$\mathcal{Q}_{-1} = \mathcal{P}_0 = \mathcal{O}$, $\rho_{-1} = 1$;
Choose $\widetilde{\mathcal{R}}_0$(e.g., $\widetilde{\mathcal{R}}_0 = \mathcal{R}_0$) such that $\langle \mathcal{R}_0, \widetilde{\mathcal{R}}_0\rangle \neq 0$;
$k = 0$;
while $(\|\mathcal{R}_k\|/\|\mathcal{R}_0\| > \varepsilon)$
    $\rho_k = \langle \widetilde{\mathcal{R}}_0, \mathcal{R}_k\rangle$;
    $\beta_k = \rho_k/\rho_{k-1}$;
    $\mathcal{U}_k = \mathcal{R}_k + \beta_k\mathcal{P}_k$;
    $\mathcal{Q}_k = \mathcal{U}_k + \beta_k(\mathcal{P}_k + \beta_k\mathcal{Q}_{k-1})$;
    $\mathcal{Q}_k = \mathcal{A} *_N \mathcal{Q}_k$;
    $\sigma_k = \langle \widetilde{\mathcal{R}}_0, \mathcal{Q}_k\rangle$;
    $\alpha_k = \rho_k/\sigma_k$;
    $\mathcal{P}_{k+1} = \mathcal{U}_k - \alpha_k\mathcal{Q}_k$;
    $\mathcal{Z}_k = \alpha_k(\mathcal{U}_k + \mathcal{P}_{k+1})$;
    $\mathcal{X}_{k+1} = \mathcal{X}_k + \mathcal{Z}_k$;
    $\mathcal{R}_{k+1} = \mathcal{R}_k - \mathcal{A} *_N \mathcal{Z}_k$;
    $k = k + 1$;
end

Remark 4. From the derivation process of the CGS-BTF method, we know that the residual of the approximate solution generated by the CGS-BTF method is $\mathcal{R}_k^{\text{CGS-BTF}} = \phi_k^2(\mathcal{A}) *_N \mathcal{R}_0$. And, by Algorithm 3, we know that the residual of the approximate solution generated by the iCG-BTF method is $\mathcal{R}_k^{\text{BiCG-BTF}} = \phi_k(\mathcal{A}) *_N \mathcal{R}_0$. When $\|\mathcal{R}_k^{\text{BiCG-BTF}}\|$ approaches zero, $\phi_k(\mathcal{A})$ is just a contraction factor. In this sense, $\|\mathcal{R}_k^{\text{CGS-BTF}}\|$ should converge to zero twice as fast as $\|\mathcal{R}_k^{\text{BiCG-BTF}}\|$.

5.3 | The BiCGSTAB-BTF method

Although the CGS-BTF method converges to the solution of tensor equation almost twice as fast as the BiCG-BTF method. In the numerical tests, we find that the residual norm of the CGS-BTF method will shake violently. In order to overcome this shortcoming, we introduce the BiCGSTAB-BTF method. As we all know, the residual of the CGS-BTF method satisfies

$$\mathcal{R}_k^{\text{CGS-BTF}} = \phi_k(\mathcal{A}) *_N \mathcal{R}_k^{\text{BiCG-BTF}} = \phi_k^2(\mathcal{A}) *_N \mathcal{R}_0. \tag{62}$$

We can choose another k degree polynomial $\tilde{\phi}_k$ such that

$$\mathcal{R}_k = \tilde{\phi}_k(\mathcal{A}) *_N \mathcal{R}_k^{\text{BiCG-BTF}} = \tilde{\phi}_k(\mathcal{A}) *_N \phi_k(\mathcal{A}) *_N \mathcal{R}_0. \tag{63}$$

This selection is expected to improve the oscillation of the residual norm $\|\mathcal{R}_k^{\text{CGS-BTF}}\|$. One way of choosing $\tilde{\phi}_k$ is based on the following recursive formula

$$\tilde{\phi}_0(t) = 1, \quad \tilde{\phi}_{k+1}(t) = (1 - \omega_{k+1} t)\tilde{\phi}_k(t), \tag{64}$$

where ω_{k+1} is an undetermined parameter. By (53) and (64), we have

$$\tilde{\phi}_{k+1}\phi_{k+1} = (1 - \omega_{k+1}\vartheta)\tilde{\phi}_k(\phi_k - \alpha_k \vartheta \psi_k) = (1 - \omega_{k+1}\vartheta)(\tilde{\phi}_k \phi_k - \alpha_k \vartheta \tilde{\phi}_k \psi_k), \tag{65}$$

$$\tilde{\phi}_k \psi_k = \tilde{\phi}_k(\phi_k + \beta_k \psi_{k-1}) = \tilde{\phi}_k \phi_k + \beta_k (1 - \omega_k \vartheta)\tilde{\phi}_{k-1}\psi_{k-1}. \tag{66}$$

Let $\mathcal{R}_k = \tilde{\phi}_k(\mathcal{A}) *_N \phi_k(\mathcal{A}) *_N \mathcal{R}_0$ and $\mathcal{P}_k = \tilde{\phi}_k(\mathcal{A}) *_N \psi_k(\mathcal{A}) *_N \mathcal{R}_0$. Then, by the relations (65) and (66), we have

$$\mathcal{R}_{k+1} = (\mathcal{I} - \omega_{k+1}\mathcal{A}) *_N (\mathcal{R}_k - \alpha_k \mathcal{A} *_N \mathcal{P}_k), \tag{67}$$

$$\mathcal{P}_k = \mathcal{R}_k + \beta_k (\mathcal{I} - \omega_k \mathcal{A}) *_N \mathcal{P}_{k-1}. \tag{68}$$

In the sequel, we consider the computations of α_k and β_k. By (53), we know

$$\alpha_k = \frac{\rho_k}{\sigma_k}, \quad \beta_k = \frac{\rho_k}{\rho_{k-1}}, \tag{69}$$

where

$$\rho_k = [\phi_k, \phi_k], \quad \sigma_k = [\psi_k, \vartheta \psi_k].$$

According to Theorem 3, one get

$$\begin{aligned} &\phi_k(\mathcal{A}) *_N \mathcal{R}_0 = \mathcal{R}_k^{\text{BiCG-BTF}} \perp \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0), \\ &\psi_k(\mathcal{A}) *_N \mathcal{R}_0 = \mathcal{Q}_k^{\text{BiCG-BTF}} \quad \text{and} \quad \mathcal{A} *_N \mathcal{Q}_k^{\text{BiCG-BTF}} \perp \mathcal{K}_k(\mathcal{A}^T, \tilde{\mathcal{R}}_0). \end{aligned}$$

Hence, for arbitrary $\psi \in \mathfrak{P}_{k-1}$, we immediately have

$$[\psi, \phi_k] = \langle \psi(\mathcal{A}^T) *_N \tilde{\mathcal{R}}_0, \phi_k(\mathcal{A}) *_N \mathcal{R}_0 \rangle = 0, \tag{70}$$

$$[\psi, \vartheta\psi_k] = \langle \psi(\mathcal{A}^T)*_N\tilde{\mathcal{R}}_0, \mathcal{A}*_N\psi_k(\mathcal{A})*_N\mathcal{R}_0\rangle = 0. \tag{71}$$

Let ξ_k and η_k be the leading coefficient of the polynomial ϕ_k and $\tilde{\phi}_k$, respectively. Then, by the relations (53) and (64), we have

$$\xi_{k+1} = -\alpha_k\xi_k, \quad \eta_{k+1} = -\omega_{k+1}\eta_k, \tag{72}$$

where $\xi_0 = \eta_0 = 1$. As $\psi_k = \phi_k + \beta_k\psi_{k-1}$, we conclude that ϕ_k and ψ_k have the same leading coefficient. Then

$$\psi = \psi_k - \frac{\xi_k}{\eta_k}\tilde{\phi}_k \quad \text{and} \quad \phi = \phi_k - \frac{\xi_k}{\eta_k}\tilde{\phi}_k$$

are all $k-1$ degree polynomial. It then follows from the relations (70) and (71) that

$$\begin{aligned}
\rho_k &= [\phi_k, \phi_k] = [\phi + \frac{\xi_k}{\eta_k}\tilde{\phi}_k, \phi_k] = [\frac{\xi_k}{\eta_k}\tilde{\phi}_k, \phi_k] = \frac{\xi_k}{\eta_k}\langle \tilde{\phi}_k(\mathcal{A}^T)*_N\tilde{\mathcal{R}}_0, \phi_k(\mathcal{A})*_N\mathcal{R}_0\rangle \\
&= \frac{\xi_k}{\eta_k}\langle \tilde{\mathcal{R}}_0, \tilde{\phi}_k(\mathcal{A})*_N\phi_k(\mathcal{A})*_N\mathcal{R}_0\rangle = \frac{\xi_k}{\eta_k}\langle \tilde{\mathcal{R}}_0, \mathcal{R}_k\rangle, \\
\sigma_k &= [\psi_k, \vartheta\psi_k] = [\psi + \frac{\xi_k}{\eta_k}\tilde{\phi}_k, \vartheta\psi_k] = \frac{\xi_k}{\eta_k}[\tilde{\phi}_k, \vartheta\psi_k] \\
&= \frac{\xi_k}{\eta_k}\langle \tilde{\phi}_k(\mathcal{A}^T)*_N\tilde{\mathcal{R}}_0, \mathcal{A}*_N\psi_k(\mathcal{A})*_N\mathcal{R}_0\rangle \\
&= \frac{\xi_k}{\eta_k}\langle \tilde{\mathcal{R}}_0, \mathcal{A}*_N\tilde{\phi}_k(\mathcal{A})*_N\psi_k(\mathcal{A})*_N\mathcal{R}_0\rangle = \frac{\xi_k}{\eta_k}\langle \tilde{\mathcal{R}}_0, \mathcal{A}*_N\mathcal{P}_k\rangle.
\end{aligned}$$

Hence, by the relation (72), we have

$$\begin{aligned}
\alpha_k &= \frac{\rho_k}{\sigma_k} = \frac{\langle \tilde{\mathcal{R}}_0, \mathcal{R}_k\rangle}{\langle \tilde{\mathcal{R}}_0, \mathcal{A}*_N\mathcal{P}_k\rangle}, \\
\beta_k &= \frac{\rho_k}{\rho_{k-1}}\frac{\eta_{k-1}}{\eta_k}\frac{\langle \tilde{\mathcal{R}}_0, \mathcal{R}_k\rangle}{\langle \tilde{\mathcal{R}}_0, \mathcal{R}_{k-1}\rangle} = \frac{\alpha_{k-1}}{\omega_k}\frac{\langle \tilde{\mathcal{R}}_0, \mathcal{R}_k\rangle}{\langle \tilde{\mathcal{R}}_0, \mathcal{R}_{k-1}\rangle}.
\end{aligned}$$

Finally, we consider the choice of ω_{k+1}. Let $\mathcal{S}_k = \mathcal{R}_k - \alpha_k\mathcal{A}*_N\mathcal{P}_k$. Then, by (67), we get

$$\mathcal{R}_{k+1} = (\mathcal{I} - \omega_{k+1}\mathcal{A})*_N\mathcal{S}_k. \tag{73}$$

Hence, we can choose ω_{k+1} such that

$$\|\mathcal{R}_{k+1}\| = \min_{\omega}\|(\mathcal{I} - \omega\mathcal{A})*_N\mathcal{S}_k\|. \tag{74}$$

Solving the least squares problem (74) yields

$$\omega_{k+1} = \frac{\langle \mathcal{S}_k, \mathcal{A}*_N\mathcal{S}_k\rangle}{\langle \mathcal{A}*_N\mathcal{S}_k, \mathcal{A}*_N\mathcal{S}_k\rangle}.$$

Notice that $\mathcal{R}_{k+1}$ in (73) can be rewritten as

$$\mathcal{R}_{k+1} = \mathcal{S}_k - \omega_{k+1}\mathcal{A}*_N\mathcal{S}_k = \mathcal{R}_k - \alpha_k\mathcal{A}*_N\mathcal{P}_k - \omega_{k+1}\mathcal{A}*_N\mathcal{S}_k,$$

which implies that if

$$\mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k\mathcal{P}_k + \omega_{k+1}\mathcal{S}_k,$$

then $\mathcal{R}_{k+1} = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_{k+1}$ is exactly the residual of the corresponding approximate solution.

Now, the main steps of the BiCGSTAB-BTF method are summarized as in Algorithm 5.

Algorithm 5. The BiCGSTAB-BTF method

Given the coefficient tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_1 \times \cdots \times I_N}$ and the right-hand side tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$.

Input: the initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$, the tolerance error $\varepsilon > 0$.
Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$;
$\mathcal{P}_0 = \mathcal{R}_0$;
Choose $\widetilde{\mathcal{R}}_0$(e.g., $\widetilde{\mathcal{R}}_0 = \mathcal{R}_0$) such that $\langle \mathcal{R}_0, \widetilde{\mathcal{R}}_0 \rangle \neq 0$;
$k = 0$;
while $(\|\mathcal{R}_k\| / \|\mathcal{R}_0\| > \varepsilon)$
  $k = k + 1$;
  $\mathcal{U}_k = \mathcal{A} *_N \mathcal{P}_k$;
  $\rho_k = \langle \widetilde{\mathcal{R}}_0, \mathcal{U}_k \rangle$;
  $\alpha_k = \rho_k / \sigma_k$;
  $\mathcal{S}_k = \mathcal{R}_k - \alpha_k \mathcal{U}_k$;
  $\mathcal{Q}_k = \mathcal{A} *_N \mathcal{S}_k$;
  $\omega_{k+1} = \dfrac{\langle \mathcal{S}_k, \mathcal{Q}_k \rangle}{\langle \mathcal{Q}_k, \mathcal{Q}_k \rangle}$;
  $\mathcal{X}_{k+1} = \mathcal{X}_k + \alpha_k \mathcal{P}_k + \omega_{k+1} \mathcal{S}_k$;
  $\mathcal{R}_{k+1} = \mathcal{S}_k - \omega_{k+1} \mathcal{Q}_k$;
  $\rho_{k+1} = \langle \widetilde{\mathcal{R}}_0, \mathcal{R}_{k+1} \rangle$;
  $\beta_{k+1} = (\alpha_k \rho_{k+1}) / (\omega_{k+1} \rho_k)$;
  $\mathcal{P}_{k+1} = \mathcal{R}_{k+1} + \beta_{k+1}(\mathcal{P}_k - \omega_{k+1} \mathcal{U}_k)$;
end

6 | NUMERICAL EXPERIMENTS

In this section, we give some numerical examples to show the performance of our proposed tensor-based iterative methods. In our computations, all of the tests are implemented in MATLAB R2015a with the machine precision 10^{-16} on a personal computer (Intel (R) Core (TM)i7-5500U), where the CPU is 2.40 GHz and the memory is 8.0 GB. All the implementation is based on the functions from the MATLAB Tensor Toolbox developed by Bader and Kolda.[41]

Example 1. Assume that $N = 3$, $M = 3$, $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_1 \times I_2 \times I_3}$, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times J_1 \times J_2 \times J_3}$, and $\mathcal{C} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times J_1 \times J_2 \times J_3}$. Especially, we take $I_1 = I_2 = I_3 = 15$, $J_1 = J_2 = J_3 = 10$, and construct a tensor $\mathcal{A} \in \mathbb{R}^{15 \times 15 \times 15 \times 15 \times 15 \times 15}$, where the code is as follows:

```
m1=15;
m2=15;
m3=15;
A=100*tenrand([m1 m2 m3 m1 m2 m3])-50;
for i1=1:m1
      for i2=1:m2
            for i3=1:m3
                  A(i1,i2,i3,i1,i2,i3)=sum(abs(double(tenmat(A(i1,i2,i3,:,:,:),
                  1:3,'t'))))+25*rand(1);
            end
      end
end
```

The exact solution $\tilde{\mathcal{X}}$ is a tensor with all elements equal to one, that is, $\tilde{\mathcal{X}}$ = tenones([15 15 15 10 10 10]). Then, by $\mathcal{C} = \mathcal{A} *_N \tilde{\mathcal{X}}$, we generate the right-hand side tensor $\mathcal{C}$ of Equation (13).

	BTF			BVF		
	IT	CPU	ERR	IT	CPU	ERR
BiCG	11	7.0801	8.0745e−09	11	56.6351	8.0745e−09
CGS	3	4.2798	2.5210e−10	2	37.9136	2.5210e−10
BiCGSTAB	3	4.4376	1.3238e−10	3	39.3289	1.3238e−10

TABLE 1 Comparison BTF with BVF by Example 1

By Example 1, we test the numerical performance of the tensor-based iterative methods and the corresponding vector-based iterative methods. If we do not use the compact tensor format, we should first transform the tensor equation $\mathcal{A}*_3\mathcal{X} = \mathcal{C}$ into the matrix equation $AX = C$ by using the matrix unfolding, where $A = \Phi_{II}(\mathcal{A}) \in \mathbb{R}^{I\times I}$, $X = \Phi_{IJ}(\mathcal{X}) \in \mathbb{R}^{I\times J}$ and $C = \Phi_{IJ}(\mathcal{C}) \in \mathbb{R}^{I\times J}$. In order to use the classical iterative method for linear systems, we have the two ways:

- to reformulate the matrix equation $AX = C$ as a linear equation $(I^{(IJ)} \otimes A)\text{vec}(X) = \text{vec}(C)$ by using the Kronecker product and the vec operator, where $I^{(IJ)}$ is the identity matrix of order IJ.
- to reformulate the matrix equation $AX = C$ as some linear equations $AX(:,j) = C(:,j)$ for $j = 1, \cdots, J$.

In the first way, the size of matrix $(I^{(IJ)} \otimes A)$ is very large, and is equal to 3,375,000 × 3,375,000, which takes up 84,866.8GB of memory. Creation of this array may take a long time and cause MATLAB unresponsive. And, some properties of matrix A might be lost after taking the Kronecker product of A and identity matrix $I^{(IJ)}$. Moreover, the iteration number may be big as the size of linear equation $(I^{(IJ)} \otimes A)\text{vec}(X) = \text{vec}(C)$ is very large. Hence, in our test, we adopt the second way. In this case, after finding the solutions of all linear equations $AX(:,j) = C(:,j)$ for $j = 1, \cdots, J$, we obtain the solution of tensor equation $\mathcal{A}*_3\mathcal{X} = \mathcal{C}$. And, we call this method a vector-based method. For convenience, in this example, we only consider the tensor-based projection method (i.e., BiCG-BTF, CGS-BTF, BiCGSTAB-BTF methods) and the corresponding vector-based projection method (i.e., BiCG-BVF, CGS-BVF, BiCGSTAB-BVF methods). The initial values of all tested iterative methods are chosen to be zero tensors or zero vectors of the appropriate dimension. And, the stopping criterion is set to the relative residual norm ERR$:= \|\mathcal{R}_k\|/\|\mathcal{R}_0\| < 10^{-7}$. For fairness, the iteration number of the vector-based iterative method is the average iteration number for solving all reformulated linear equations. The comparison results are shown in Table 1. It can be seen from Table 1 that the elapse CPU time of the tensor-based iterative methods is much less than the vector-based iterative methods. This example confirms that the tensor-based iterative method keeps the computational cost low and the tensor-based iterative methods are better than the vector-based iterative methods.

Example 2. Consider the following three-dimensional Poisson problem

$$\begin{cases} -\nabla^2 v = f, & \text{in } \Omega = \{(x, y, z), 0 < x, y, z < 1\}, \\ v = 0, & \text{on } \partial\Omega, \end{cases} \tag{75}$$

where f is a given function, and

$$\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}.$$

We want to find an approximation of the unknown function $v(x, y, z)$ in (77). Take the uniform mesh step size. That is, the step sizes, Δx in the x-direction, Δy in the y-direction, and Δz in the z-direction, satisfy $\Delta y = \Delta z = \Delta x = 1/(n+1)$. The higher order tensor representation of the discretized Poisson problem (77) is (see Reference 21)

$$\overline{\mathcal{A}}*_3\mathcal{V} = \mathcal{F}, \tag{76}$$

where the Laplacian tensor $\overline{\mathcal{A}} \in \mathbb{R}^{n\times n\times n\times n\times n\times n}$ and $\mathcal{V}, \mathcal{F} \in \mathbb{R}^{n\times n\times n}$. Both $\mathcal{V}$ and $\mathcal{F}$ are discretized on the unit cube. The entries on the tensor block $(\overline{\mathcal{A}})^{(2,4,6)}_{l,m,p}$ of $\overline{\mathcal{A}}$ would follow a seven-point stencil, that is,

$$((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha,\beta,\gamma})_{\alpha,\beta,\gamma} = \frac{6}{\Delta x^3},$$

TABLE 2 The numerical results for Example 2 (RES < 10^{-6})

$n\times n\times n$	$10\times10\times10$ IT/CPU/RES	$15\times15\times15$ IT/CPU/RES	$20\times20\times20$ IT/CPU/RES	$25\times25\times25$ IT/CPU/RES
CR-BTF	28/0.0648/0.7773e−06	47/0.3730/0.9048e−06	63/2.1539/0.8825e−06	82/10.6875/0.7178e−06
GCR-BTF	28/0.3002/0.7773e−06	47/1.0185/0.9048e−06	63/3.3996/0.8825e−06	82/12.9731/0.7178e−06
BiCG-BTF	57/0.4099/0.7908e−06	95/1.7242/0.9800e−06	129/8.0511/0.4954e−06	165/34.2438/0.9013e−06
CGS-BTF	20/0.5087/0.2614e−06	36/2.2938/0.3608e−06	47/11.3258/0.8310e−06	64/51.2787/0.6126e−06
BiCGSTAB-BTF	21/0.5985/0.0808e−06	36/2.8270/0.4498e−06	50/14.7538/0.1685e−06	59/67.3584/0.3551e−06

$$((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha-1,\beta,\gamma})_{\alpha-1,\beta,\gamma} = ((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha+1,\beta,\gamma})_{\alpha+1,\beta,\gamma} = -\frac{1}{\Delta x^3},$$
$$((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha,\beta-1,\gamma})_{\alpha,\beta-1,\gamma} = ((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha,\beta+1,\gamma})_{\alpha,\beta+1,\gamma} = -\frac{1}{\Delta x^3},$$
$$((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha,\beta,\gamma-1})_{\alpha,\beta,\gamma-1} = ((\overline{\mathcal{A}})^{(2,4,6)}_{\alpha,\beta,\gamma+1})_{\alpha,\beta,\gamma+1} = -\frac{1}{\Delta x^3},$$

for $\alpha, \beta, \gamma = 2, \cdots, n-1$ since

$$6v_{ijk} - v_{i-1jk} - v_{i+1jk} - v_{ij-1k} - v_{ij+1k} - v_{ijk-1} - v_{ijk+1} = \Delta x^3 f_{ijk}. \tag{77}$$

Here, $(\overline{\mathcal{A}})^{(2,4,6)}_{l,m,p} = \overline{\mathcal{A}}(:, l, :, m, :, p) \in \mathbb{R}^{n\times n\times n}$ is the block tensor of $\overline{\mathcal{A}}$.

In this example, we find the solution of tensor equation (13) by the five iterative algorithms mentioned above. We illustrate the numerical efficiency of the above methods from the aspects of iterative steps (denoted by "IT"), elapsed CPU time in seconds (denoted by "CPU"), and the norm of residual (denoted by "RES"). Here, "RES" is defined by

$$\text{RES} = \|\mathcal{R}_k\|, \tag{78}$$

where $\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_k$. The stopping criterion of the above mentioned algorithms is considered as RES < 10^{-6}. The initial tensor of all iterative methods is chosen as the zero tensor. And, the corresponding numerical results are presented in Table 2.

From Table 2, we find that our proposed iterative methods are feasible and valid for the three-dimensional Poisson problem, in which one can see the elapsed CPU time of the CR-BTF method is less than others.

For this example, we draw Figure 1 based on the iterative step and the residual for different grid sizes. These figures show that the proposed methods are almost quadratic convergent, and BiCGSTAB-BTF performs the best than others. It is also showed that the proposed tensor-based iterative methods are quite effective to solve Poisson problem in three dimensions.

For Example 2, we also test the numerical performance of the abovementioned iterative methods by imposing another stopping criterion. That is,

$$\text{ERR} = \frac{\|\mathcal{R}_k\|}{\|\mathcal{R}_0\|} < 10^{-7}, \tag{79}$$

where $\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_k$. We report the iterative steps, the elapsed CPU time in seconds and the norm of relative residual in Table 3 and draw Figure 2 based on the iterative number and relative residual, in which one may see that the proposed methods have the similar numerical results to the RES case given in (78). The BiCG-BTF method requires more iterations comparing with CGS-BTF and BiCGSTAB-BTF methods. The numerical result is consistent with our theoretical analysis. According to Figure 2, we also find that the relative residual norm of CGS-BTF method $\|\mathcal{R}_k^{\text{CGS-BTF}}\|/\|\mathcal{R}_0^{\text{CGS-BTF}}\|$ will jitter violently similar to Figure 1 as the iteration number k changes. By choosing the appropriate parameter ω_{k+1}, the oscillation of relative residual norm of the BiCGSTAB-BTF method is improved when compared with CGS-BTF method. This confirms our theoretical analysis in Section 5.3.

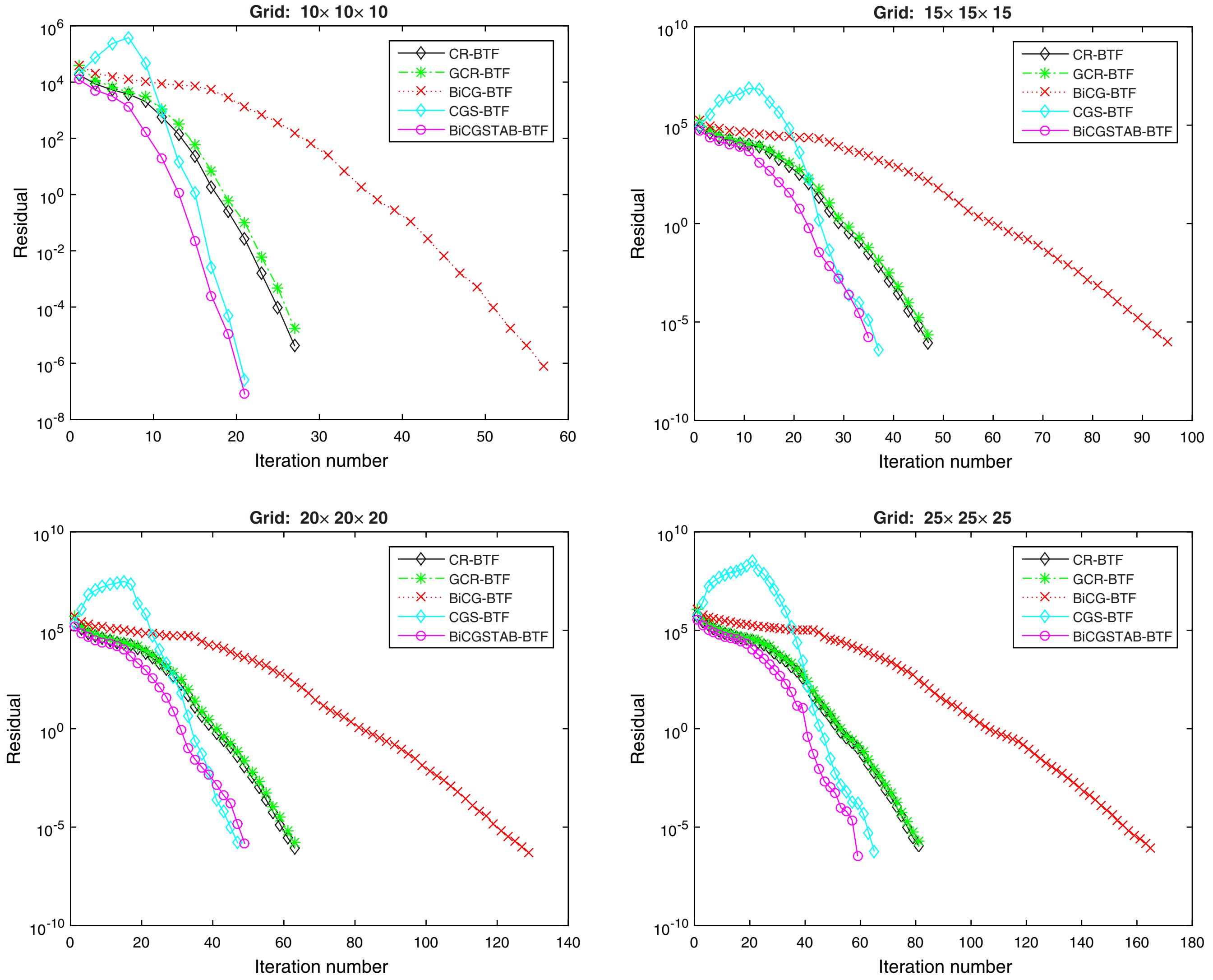

FIGURE 1 The numerical result for Example 2 (RES < 10^{-6})

Example 3. Assume that $N = 2$, $M = 2$, $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times J_1 \times J_2}$, and $\mathcal{C} \in \mathbb{R}^{I_1 \times I_2 \times J_1 \times J_2}$. Especially, we take $I_1 = 50$, $I_2 = 30$, $J_1 = 30$, and $J_2 = 20$, and randomly construct the symmetric tensor $\mathcal{A} \in \mathbb{R}^{50 \times 30 \times 50 \times 30}$. The exact solution $\tilde{\mathcal{X}}$ is a tensor with all elements equal to one, that is, $\tilde{\mathcal{X}} = \text{tenones}([50\ 30\ 30\ 20])$. Then, by $\mathcal{C} = \mathcal{A} *_N \tilde{\mathcal{X}}$, we generate the right-hand side tensor $\mathcal{C}$ of Equation (13).

In order to compare the matrix-based methods with their corresponding tensor-based methods, in Example 3, equivalently, we reformulate the tensor equation (13) to $\Phi_{II}(\mathcal{A})\Phi_{IJ}(\mathcal{X}) = \Phi_{IJ}(\mathcal{C})$, and then employ the tensor-based methods and the matrix-based method for CR, GCR, BiCG, CGS, and BiCGSTAB to solve the tensor equation (13) and the deduced matrix equation, respectively.

Next, we make the CPU time cost comparisons against the resolutions between our proposed tensor-based iterative methods and the corresponding matrix forms.

The initial tensor (matrix) of the tensor (matrix)-based iterative methods are all set to zero tensors (matrices) with the appropriate dimension. The relative residual norm ERR is defined by

$$\text{ERR} = \frac{\|\mathcal{R}_k\|}{\|\mathcal{R}_0\|} \quad \text{or} \quad \text{ERR} = \frac{\|R_k\|}{\|R_0\|} \tag{80}$$

with $\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_k$ and $R_k = \Phi_{IJ}(\mathcal{C}) - \Phi_{II}(\mathcal{A})X_k$. The stopping criterion of the above mentioned algorithms is considered as ERR < 10^{-6}.

TABLE 3 The numerical results for Example 2 (ERR < 10^{-7})

$n \times n \times n$	$10 \times 10 \times 10$ IT/CPU/RES	$15 \times 15 \times 15$ IT/CPU/RES	$20 \times 20 \times 20$ IT/CPU/RES	$25 \times 25 \times 25$ IT/CPU/RES
CR-BTF	23/0.0548/0.4077e−07	36/0.2697/0.8284e−07	47/1.5418/0.8526e−07	59/7.5835/0.8826e−07
GCR-BTF	23/0.1794/0.4077e−07	36/0.6834/0.8284e−07	47/2.2669/0.8526e−07	59/9.1169/0.8826e−07
BiCG-BTF	47/0.2700/0.4220e−07	73/1.2063/0.9453e−07	97/5.4746/0.5752e−07	121/24.3932/0.7569e−07
CGS-BTF	17/0.3425/0.6936e−07	28/1.6065/0.4753e−07	38/8.0266/0.4662e−07	48/36.4637/0.7886e−07
BiCGSTAB-BTF	16/0.4102/0.5439e−07	26/1.9907/0.8920e−07	34/10.2559/0.9754e−07	42/47.3173/0.5076e−07

TABLE 4 Comparison BTF with BMF by Example 3

	BTF			BMF		
	IT	CPU	ERR	IT	CPU	ERR
CR	76	6.6105	0.9622e−06	76	20.9571	0.9622e−06
GCR	67	38.4323	0.9603e−06	67	109.2688	0.9603e−06
BiCG	517	43.3713	0.9090e−06	531	89.0988	0.9827e−06
CGS	67	11.4128	0.9485e−06	65	24.0156	0.9165e−06
BiCGSTAB	63	10.8441	0.9631e−06	69	24.6760	0.9984e−06

The corresponding numerical results are reported in Table 4, which shows that the CPU time of the proposed tensor-based iterative methods (i.e., CR-BTF, GCR-BTF, BiCG-BTF, CGS-BTF, and BiCGSTAB-BTF methods) is far less than the corresponding one for the matrix-based method (i.e., CR-BMF, GCR-BMF, BiCG-BMF, CGS-BMF, and BiCGSTAB-BMF methods).

Example 4. Consider the following convective diffusion equation with Dirichlet boundary condition

$$-\triangle u+\frac{\partial u}{\partial x}+2\frac{\partial u}{\partial y}=f,\quad \Omega=\{(x,y),0<x,y<1\}. \tag{81}$$

For a positive integer n, set $h=1/(n+1), x_k=kh, y_l=lh$ and $u(x_k,y_l)=u_{kl}$ for $k,l=1,2,\cdots,n$, then the standard central difference discretization on equidistant nodes of (81) leads to the following tensor equation

$$\mathcal{A}*_2X=C, \tag{82}$$

where $X, C\in\mathbb{R}^{n\times n}$, and $\mathcal{A}\in\mathbb{R}^{n\times n\times n\times n}$ satisfies

$$\begin{aligned}
&((\mathcal{A})^{(1,2)}_{\alpha,\beta})_{\alpha,\beta}=1,\\
&((\mathcal{A})^{(1,2)}_{\alpha,\beta})_{\alpha-1,\beta}=-\frac{2+h}{8},\quad ((\mathcal{A})^{(1,2)}_{\alpha,\beta})_{\alpha+1,\beta}=-\frac{2-h}{8},\\
&((\mathcal{A})^{(1,2)}_{\alpha,\beta})_{\alpha,\beta-1}=-\frac{1+h}{4},\quad ((\mathcal{A})^{(1,2)}_{\alpha,\beta})_{\alpha,\beta+1}=-\frac{1-h}{4}.
\end{aligned}$$

Since the coefficient tensor $\mathcal{A}$ in Example 4 is not symmetric, we only test the numerical performances of projection methods, that is, BiCG-BTF, CGS-BTF, and BiCGSTAB-BTF methods. The initial tensors are all set to zero tensors with appropriate dimensions. The stopping criterion is considered as ERR<10^{-6}. The numerical results are listed in Table 5. It can be seen from Table 5 that the BiCG-BTF, CGS-BTF, and BiCGSTAB-BTF methods are all efficient for the nonsymmetric tensor equation arising from the discretization of convective diffusion problem.

Example 5. Consider the Toeptiz tensor equation $\mathcal{T}*_3\mathcal{X}=\mathcal{B}$, where $\mathcal{X}\in\mathbb{R}^{n_1\times n_2\times n_3}$ is an unknown tensor to be determined, $\mathcal{B}\in\mathbb{R}^{n_1\times n_2\times n_3}$ is a tensor whose elements are all one and $\mathcal{T}=(t_{i_1i_2i_3j_1j_2j_3})\in\mathbb{R}^{n_1\times n_2\times n_3\times n_1\times n_2\times n_3}$ is a Toeptiz tensor

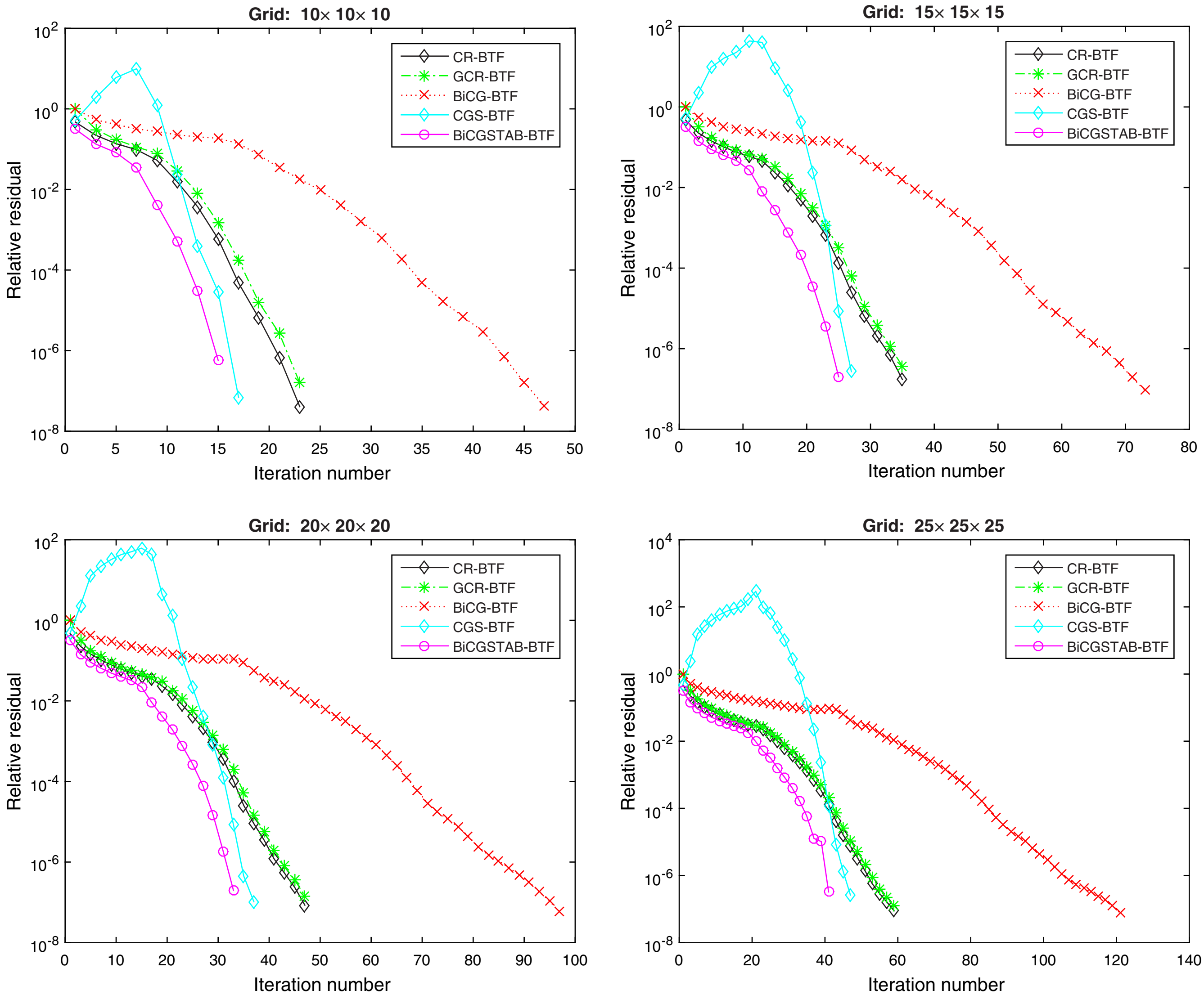

FIGURE 2 The numerical result for Example 2 (ERR $< 10^{-7}$)

TABLE 5 The numerical results for Example 4 (ERR $< 10^{-6}$)

$n \times n$	100 × 100 IT/CPU/ERR	120 × 120 IT/CPU/ERR	150 × 150 IT/CPU/ERR	200 × 200 IT/CPU/ERR
BiCG-BTF	395/18.2659/ 9.5805e−07	417/24.4972/9.3857e−07	513/52.9508/9.8619e−07	675/124.3921/9.8737e−07
CGS-BTF	136/11.5777/9.9774e−07	148/16.3097/9.8899e−07	199/42.2098/9.1737e−07	162/56.9811/9.5376e−07
BiCGSTAB-BTF	159/13.9125/3.3997e−07	151/18.3625/9.8746e−07	198/42.7870/8.5198e−07	183/74.8179/9.8120e−07

given by

$$t_{i_1 i_2 i_3 j_1 j_2 j_3} = g_{i_1 - j_1, i_2 - j_2, i_3 - j_3}, \quad 1 \le i_s, j_s \le n_s, \quad 1 \le s \le 3,$$

where $\mathcal{G} = (g_{i_1 i_2 i_3}) \in \mathcal{R}^{(2n_1-1)\times(2n_2-1)\times(2n_3-1)}$ is given by the following generating sequence

$$g_{i_1 i_2 i_3} = \frac{1}{(|i_1| + 0.5)(|i_2| + 0.5)(|i_3| + 0.5)},$$

TABLE 6 The numerical results for Example 5

	n	20	50	100	150	180
CR-BTF	IT	51	83	113	132	142
	CPU	0.3212	8.1150	90.7866	405.9779	1185.7073
	RES	7.0395e−09	9.4364e−09	8.5654e−09	9.4799e−09	9.5822e−09
GCR-BTF	IT	48	80	107	126	136
	CPU	1.4375	12.7703	118.7214	5172.4670	19626.3887
	RES	7.6302e−09	7.9135e−09	9.3238e−09	9.0523e−09	8.1632e−09
BiCG-BTF	IT	51	86	119	139	150
	CPU	0.5378	15.8511	200.5883	826.8328	1932.8881
	RES	9.4920e−09	9.1060e−09	9.2042e−09	9.9608e−09	8.1031e−09
CGS-BTF	IT	34	64	97	112	130
	CPU	0.3618	11.8513	169.8737	684.1736	1697.6377
	RES	6.4324e−09	8.7401e−09	3.0732e−09	4.7263e−09	9.6354e−09
BiCGSTAB-BTF	IT	37	58	77	99	102
	CPU	0.4370	10.6712	132.5104	604.1611	1309.5693
	RES	6.0296e−09	8.2537e−09	9.3611e−09	8.3826e−09	9.6353e−09

for $1-n_s \leq k_s \leq n_s-1$ and $1 \leq s \leq 3$.

For convenience, we set $n_1 = n_2 = n_3 = n$ and the initial tensor is chosen as the zero tensor. The stopping criterion is that the residual satisfies $\|\mathcal{R}_k\| < 10^{-8}$, where $\mathcal{R}_k = \mathcal{B} - \mathcal{T} *_3 \mathcal{X}_k$. The iterative steps (denoted by "IT"), the elapse CPU time (denoted by "CPU"), and the residual norm (denoted by "RES") are given in Table 6. It can be seen from Table 6 that the iteration number and elapse CPU time of CR-BTF method is the least. With the increasing of the iteration index k, the iteration number of all the mentioned iteration methods increases slowly and the elapse CPU time increases quickly. In a word, all the mentioned iteration methods can solve the Toeptiz tensor equation efficiently. Hence, it is promising for us to apply our proposed iterative algorithms for image restoration.

In the following example, we apply the proposed methods to image restoration.

Example 6 (See the works of Cui et al.[15] and Xie et al.[16]). As illustrated in Section 1, the tensor equation (83) models the three-dimensional model of image formation,[15,16,20] that is,

$$\mathcal{G} = \mathcal{T} *_3 \mathcal{F} + \mathcal{N}, \tag{83}$$

where $\mathcal{G}, \mathcal{F}, \mathcal{N} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ are third-order tensor representations for the image, the object and noise functions, respectively. The tensor $\mathcal{T}$ is a sixth-order Toeplitz tensor obtained from a third-order tensor $\mathcal{S}$ which satisfies (7). Suppose that $\mathcal{S}$ is a three-dimensional Gaussian function, that is,

$$\mathcal{S}(x,y,z) = \frac{1}{(\sqrt{2\pi}\sigma)^3} \exp\left(-\frac{x^2+y^2+z^2}{2\sigma^2}\right). \tag{84}$$

In Example 6, we use some MRI images in MATLAB. Figure 3 shows a stack of brain MRI images restoration. The size of each image is 128×128. The first row in Figure 3 are clean images. We get a stack of blurred images by using the Gaussian function (84) ($\sigma = 1$) and adding a random noise $\mathcal{N} = 0.001 \times \text{rand}(128,128,6)$. And the blurred images are located in the second row of Figure 3. Our purpose is to recover a stack of images from the stack of blurred images at the same time without getting them slice by slice. The CR-BTF method can be employed to solve the tensor equation (83) for getting solution $\mathcal{F}$ (the original MRI images). The initial tensor is set to the zero tensor, and the stopping criterion is that the residual satisfies $\|\mathcal{R}_k\| < 10^{-8}$. The numerical result are reported at the third row of Figure 3.

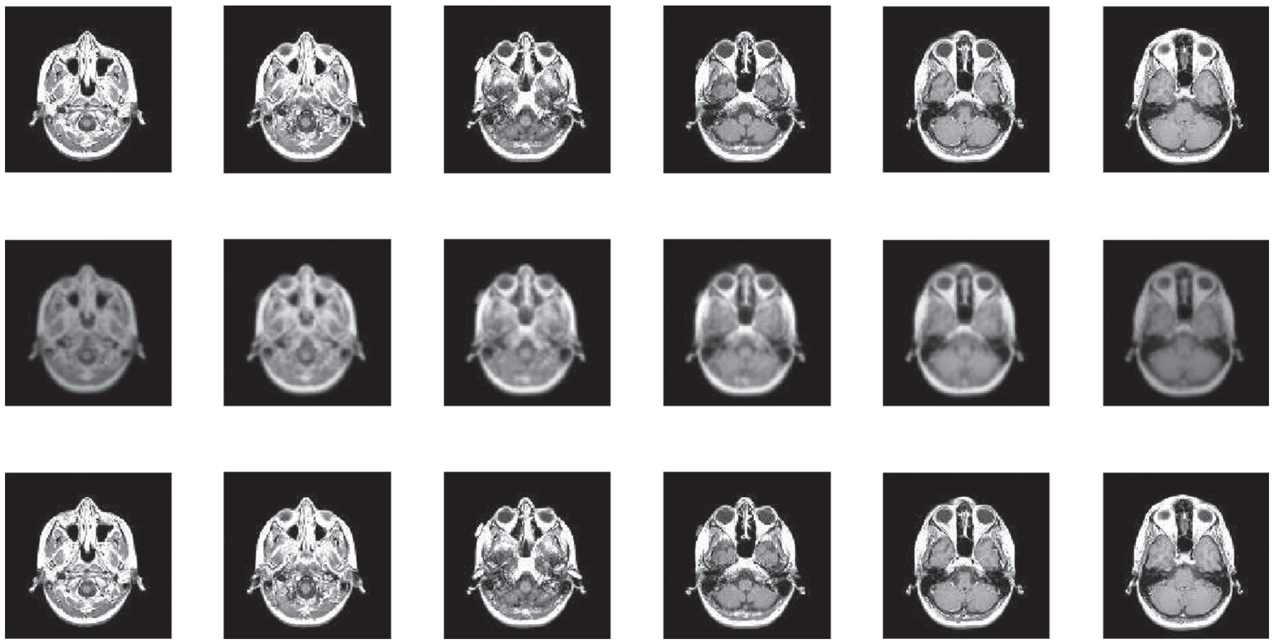

FIGURE 3 An example of a stack of MRI images restoration. The images in the first, second, and third row are the clean MRI images, blurred images with 3D Gaussian function ($\sigma = 1$) and random noise, and restoration images by CR-BTF method, respectively

Although we can restore each slice in the stack of images by image processing method, the CR-BTF method can recover all images in the stack at the same time. Our proposed tensor methods have the similar numerical performances. These show that the tensor method is efficient. And the tensor method can use the information between difference slices. For example, each slice is different but the image structure of adjacent slices is similar.

7 | CONCLUDING REMARKS

In the article, based on the CR and GCR methods for linear systems, we have developed the tensor forms of the corresponding ones (i.e., CR-BTF and GCR-BTF methods) for solving the tensor equation (13). Then, we proved that both CR-BTF and GCR-BTF methods converge to the exact solution within a finite number of iterative steps in the absence of round-off errors.

By using a product between two tensors, we established a unified framework of the projection method. Then, we extended the biconjugate gradient, conjugate gradient squares, and biconjugate gradient stabilized methods to the tensor case for solving the tensor equation (13). In order to show the necessary of the proposed algorithms for tensor versions, we compare them with the corresponding standard one by transforming tensor equations to linear (matrix) equations of systems. The numerical results show the high performance of those algorithms for the tensor version.

As numerical experiments, we apply the proposed iterative methods to solve the tensor equation arising from three-dimensional Poisson problem and convective diffusion problem, and we find that solving multilinear systems with the tensor-based iterative methods preserve low bandwidth of PDEs by means of the higher order tensor representation of PDEs, thereby keeping the computational cost and memory requirement low. Furthermore, we make the CPU time comparisons against the resolutions between our proposed tensor-based iterative methods and the matrix-based iterative methods, and we find that the tensor-based iterative methods are superior to the matrix-based iterative methods.

To improve the applicability of tensor-based methods in solving the tensor equation in scientific computing, we will develop tensor-based methods in the future research:

- For the high-dimensional problems, combining the idea of tensor partitioning in tensor principle component analysis (see Feng et al.[42]), a novel method which can operate on each sparse tensor blocks as opposed to the whole tensor structure may be discussed, which will cut down the memory and operational costs (also see in Brazell et al.[21]).

- For some mathematical models arising from big data et al., we may explore numerical algorithms to overcome or reduce the "curse of dimension" by the use of tensor structure.
- In order to accelerate the convergence, we may consider the Anderson acceleration of Walker and Ni,[43] preconditioner technique given by Li et al.[44] and relaxation technique given by Liu et al.[45] in designing the algorithm for solving the tensor equation.

ACKNOWLEDGMENTS

The authors would like to thank the editor and anonymous referees for their valuable comments and suggestions which have considerably improved this article. Baohua Huang is supported by China Postdoctoral Science Foundation (Grant No. 2019M660203) and National Natural Science Foundation of China (Grant Nos. 12001211, 12071159, 61976053). Wen Li is supported by National Natural Science Foundation of China (Grant Nos. 11671158, U1811464, 12071159) and Guangdong Basic and Applied Basic Research Foundation (2020B1515310013).

CONFLICT OF INTEREST

This work does not have any conflicts of interest.

ORCID

Baohua Huang https://orcid.org/0000-0002-5011-0801

REFERENCES

1. Einstein A. The foundation of the general theory of relativity. In: Kox AJ, Klein MJ, Schulmann R, editors. The collected papers of Albert Einstein. Vol. 6. Princeton, NJ: Princeton University Press, 2007; p. 146–200.
2. Lim LH. Tensors and hypermatrices, Chapter 15 in Handbook of linear algebra. Boca Raton, FL: CRC Press, 2013.
3. Qi L. Eigenvalues of a real supersymmetric tensor. J Symbolic Comput. 2005;40:1302–1324.
4. Shao J. A general product of tensors with applications. Linear Algebra Appl. 2013;439:2350–2366.
5. Chang KC, Pearson K, Zhang T. Perron Frobenius theorem for nonnegative tensors. Commun Math Sci. 2008;6:507–520.
6. Chen Z, Lu LZ. A projection method and Kronecker product preconditioner for solving Sylvester tensor equations. Sci Chin Math. 2012;55:1281–1292.
7. Beik AFP, Movahed FS, Ahmadi-Asl S. On the Krylov subspace methods based on tensor format for positive definite Sylvester tensor equations. Numer Linear Algebra Appl. 2016;23:444–466.
8. Kressner D, Tobler C. Krylov subspace methods for linear systems with tensor product structures. SIAM J Matrix Anal Appl. 2010;31:1688–1714.
9. Li BW, Tian S, Sun YS, Mao Z. Schur-decomposition for 3D matrix equations and its application in solving radiative discrete ordinates equations discretized by Chebyshev collocation spectral method. J Comput Phys. 2010;229:1198–1212.
10. Malek A, Bojdi ZK, Golbarg PNN. Solving fully three-dimensional microscale dual phase lag problem using mixed-collocation finite difference discretization. J Heat Transfer. 2012;134:0094504–094504–6.
11. Malek A, Momeni-Masuleh SH. Mixed collocation-finite difference method for 3D microscopic heat transport problems. J Comput Appl Math. 2008;217:137–147.
12. Li BW, Sun YS, Zhang DW. Chebyshev collocation spectral methods for coupled radiation and conduction in a concentric spherical participating medium. ASME J Heat Transfer. 2009;131:062701–062709.
13. Xiang H, Grigori L. Kronecker product approximation preconditioners for convection-diffusion model problems. Numer Linear Algebra Appl. 2010;17:723–752.
14. Lai WM, Rubin D, Krempl E. Introduction to continuum mechanics. Oxford, UK: Butterworth Heinemann, 2009.
15. Cui L, Chen C, Li W, Ng MK. An eigenvalue problem for even order tensors with its applications. Linear Multilinear Algebra. 2016;64:602–621.
16. Xie ZJ, Jin XQ, Sin VK. An optimal preconditioner for tensor equations involving Einstein product. Linear Multilinear Algebra. 2020;68:886–902.
17. Fry A, Navasca C. Tensor restricted isometry property for multilinear sparse system of genomic interactions. Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers. IEEE; 2014.
18. Grasedyck L. Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. Computing. 2004;72:247–265.
19. Haussühl S. Physical properties of crystals. Wiley-VCH Verlag: Weinheim, 2007.
20. Liu X, Wang L, Wang J, Meng H. A three-dimensional point spread function for phase retrieval and deconvolution. Opt Express. 2012;20:15392–15405.
21. Brazell M, Li N, Navasca C, Tamon C. Solving multilinear systems via tensor inversion. SIAM J Matrix Anal Appl. 2013;34:542–570.
22. Sun L, Zheng B, Bu C, Wei Y. Moore-Penrose inverse of tensors via Einstein product. Linear Multilinear Algebra. 2016;64:686–698.

23. Behera R, Mishra D. Further results on generalized inverses of tensors via the Einstein product. Linear Multilinear Algebra. 2017;65:1662–1682.
24. Ma HF, Li N, Stanimirović PS, Katsikis VN. Perturbation theory for Moore-Penrose inverse of tensor via Einstein product. Comput Appl Math. 2019;38:111. https://doi.org/10.1007/s40314-019-0893-6.
25. Ji J, Wei Y. The Drazin inverse of an even-order tensor and its application to singular tensor equations. Comput Math Appl. 2018;75:3402–3413.
26. Wang QW, Xu X. Iterative algorithms for solving some tensor equations. Linear Multilinear Algebra. 2019;67:1325–1349.
27. Huang BH, Xie YJ, Ma CF. Krylov subspace methods to solve a class of tensor equations via the Einstein product. Numer Linear Algebra Appl. 2019;26:e2254.
28. Hajarian M. Conjugate gradient-like methods for solving general tensor equation with Einstein product. J Franklin Inst. 2020;357:4272–4285.
29. Huang BH, Ma CF. An iterative algorithm to solve the generalized Sylvester tensor equations. Linear Multilinear Algebra. 2020;68:1175–1200.
30. Bellman RE. Dynamic programming. Princeton, NJ: Princeton University Press, 1957.
31. Khoromskij BN. Tensors-structured numerical methods in scientific computing: survey on recent advances. Chemom Intell Lab Syst. 2012;110:1–19.
32. Khoromskij BN. Tensor numerical methods in scientific computing. Berlin, Germany: Walter de Gruyter, 2018.
33. De Lathauwer L. A survey of tensor methods. Taipei: ISCAS, 2009.
34. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51:455–500.
35. Khoromskij BN, Khoromskaia V, Flad HJ. Numerical solution of the Hartree-Fock equation in multilevel tensor-structured format. SIAM J Sci Comput. 2011;33:45–65.
36. Beylkin G, Mohlenkamp MJ. Algorithms for numerical analysis in high dimension. SIAM J Sci Comput. 2005;26:2133–2159.
37. Griebel M, Hamaekers J. Sparse grids for the Schrodinger equation. M2AN. 2007;41:215–247.
38. Khoromskij BN. Tensor-structured preconditioners and approximate inverse of elliptic operators in Rd. J Construct Approx. 2009;30:599–620.
39. Saad Y. Iterative methods for sparse linear systems. 2nd ed. Philadelphia, PA: SIAM, 2003.
40. Elman HC. Iterative methods for large sparse nonsymmetric systems of linear equations [Ph.D. thesis]. New Haven, CT: Computer Science Department, Yale University; 1982.
41. Bader BW, Kolda TG. MATLAB Tensor Toolbox Version 2.5; 2012. http://www.sandia.gov/tgkolda/TensorToolbox/.
42. Feng L, Liu Y, Chen L, Zhang X, Zhu C. Robust block tensor principal component analysis. Signal Process. 2020;166:107271.
43. Walker HF, Ni P. Anderson acceleration for fixed-point iterations. SIAM J Numer Anal. 2011;49:1715–1735.
44. Li W, Liu DD, Vong SW. Comparison results for splitting iterations for solving multi-linear systems. Appl Numer Math. 2018;134:105–121.
45. Liu DD, Li W, Vong SW. Relaxation methods for solving the tensor equation arising from the higher-order Markov chains. Numer Linear Algebra Appl. 2019;26:e2260.

How to cite this article: Huang B, Li W. Numerical subspace algorithms for solving the tensor equations involving Einstein product. *Numer Linear Algebra Appl*. 2021;28:e2351. https://doi.org/10.1002/nla.2351