

RESEARCH ARTICLE

# Krylov subspace methods to solve a class of tensor equations via the Einstein product

Baohua Huang<sup>1</sup>  | Yajun Xie<sup>2</sup> | Changfeng Ma<sup>1</sup> 

<sup>1</sup>College of Mathematics and Informatics & FJKLMAA, Fujian Normal University, Fuzhou, China

<sup>2</sup>School of Mathematics and Physics, Fujian Jiangxia University, Fuzhou, China

## Correspondence

Changfeng Ma, College of Mathematics and Informatics & FJKLMAA, Fujian Normal University, Fuzhou 350117, China.  
 Email: macf@fjnu.edu.cn

Yajun Xie, School of Mathematics and Physics, Fujian Jiangxia University, Fuzhou, China.  
 Email: xiejayun0525@163.com

## Funding information

National Key Research and Development Program of China, Grant/Award Number: 2018YFC1054200 and 2018YFC0603500

## Summary

This paper is concerned with some of the well-known iterative methods in their tensor forms to solve a class of tensor equations via the Einstein product. More precisely, the tensor forms of the Arnoldi and Lanczos processes are derived and the tensor form of the global GMRES method is presented. Meanwhile, the tensor forms of the MINIRES and SYMMLQ methods are also established. The proposed methods use tensor computations with no matricizations involved. Numerical examples are provided to illustrate the efficiency of the proposed methods and testify the conclusions suggested in this paper.

## KEYWORDS

Einstein product, global GMRES method, MINIRES method, SYMMLQ method, tensor equations

## 1 | INTRODUCTION

As the extension of matrix computation, tensor computation is the latest research hotspot in the last twenty years.<sup>1–17</sup> In particular, various kinds of tensor equations arise from mechanics, physics, Markov process, control theory, partial differential equations, and engineering problems.<sup>18–22</sup> The radiation transfer equation, the high-dimensional Poisson equation, the Einstein gravitational field equation, and the piezoelectric effect equation are all tensor equations. As described in Sun et al.,<sup>22</sup> the multilinear system

$$\mathcal{A} *_2 X = C, \quad (1)$$

where  $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$ ,  $X \in \mathbb{R}^{I_1 \times I_2}$ , and  $C \in \mathbb{R}^{I_1 \times I_2}$ , has many applications in continuum physics, engineering, isotropic and anisotropic elastic models.<sup>19,23</sup> In physics, for noncentrosymmetric materials, the linear piezoelectric equation is expressed as<sup>24</sup>

$$\tilde{\mathcal{A}} *_2 T = p, \quad (2)$$

where  $\tilde{\mathcal{A}} \in \mathbb{R}^{3 \times 3 \times 3}$  is a piezoelectric tensor,  $T \in \mathbb{R}^{3 \times 3}$  is a stress tensor, and  $p \in \mathbb{R}^3$  is the electric charge density displacement. Consider the two-dimensional (2D) Poisson problem

$$\begin{cases} -\nabla^2 v = f, & \text{in } \Omega = \{(x, y), 0 < x, y < 1\}, \\ v = 0, & \text{on } \partial\Omega, \end{cases} \quad (3)$$

where  $f$  is a given function, and

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

By the standard central difference approximations, Poisson's equation (3) in two dimensions has the following higher order tensor representation<sup>18</sup>:

$$\mathcal{A} *_2 V = F, \quad (4)$$

where  $\mathcal{A} \in \mathbb{R}^{n \times n \times n \times n}$ , and matrices  $V \in \mathbb{R}^{n \times n}$  and  $F \in \mathbb{R}^{n \times n}$  are the discretized functions  $v$  and  $f$  on the unit square mesh. Similarly, the higher order tensor representation of the three-dimensional (3D) discretized Poisson problem is<sup>18</sup>

$$\bar{\mathcal{A}} *_3 \mathcal{V} = \mathcal{F}, \quad (5)$$

where  $\bar{\mathcal{A}} \in \mathbb{R}^{n \times n \times n \times n \times n \times n}$  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 general form of the tensor Equations (1), (2), (4), and (5) is

$$\mathcal{A} *_N \mathcal{X} = \mathcal{C}, \quad (6)$$

where  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$ ,  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ , and  $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_N \times K_1 \times \dots \times K_M}$ . The operation  $\mathcal{A} *_N \mathcal{X}$  is called the Einstein product of tensors  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$  and  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ . The Einstein product of tensors  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$  and  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$  is of the size  $I_1 \times \dots \times I_N \times K_1 \times \dots \times K_M$  and its elements are defined by<sup>25</sup>

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

We also comment here that the Einstein product of tensors  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$  and  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$  is a kind of compressed product of tensors  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$  and  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ , which compresses the last  $N$  indices of tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$  and the first  $N$  indices of tensor  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ . For more details, one can refer to Lim.<sup>26</sup>

Recently, some scholars investigated the generalized inverses of tensors via the Einstein product and gave the general solutions of the tensor Equation (6) by using the generalized inverses of tensors.<sup>22,27</sup> Wang et al.<sup>28</sup> considered the iterative algorithms for solving the tensor Equation (6). Huang et al.<sup>29</sup> developed the tensor form of the conjugate gradient least squares method to solve the following tensor equation via the Einstein product:

$$\mathcal{A} *_N \mathcal{X} *_M \mathcal{B} + \mathcal{C} *_N \mathcal{X} *_M \mathcal{D} = \mathcal{F}, \quad (7)$$

where  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$ ,  $\mathcal{B} \in \mathbb{R}^{K_1 \times \dots \times K_M \times L_1 \times \dots \times L_M}$ ,  $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$ ,  $\mathcal{D} \in \mathbb{R}^{K_1 \times \dots \times K_M \times L_1 \times \dots \times L_M}$ , and  $\mathcal{F} \in \mathbb{R}^{I_1 \times \dots \times I_N \times L_1 \times \dots \times L_M}$  are known tensors, and  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$  is an unknown tensor to be determined. Wang et al.<sup>28</sup> and Huang et al.<sup>29</sup> avoided standard numerical methods, which usually fail due to the so-called curse of dimensionality (see the work of Bellman<sup>30</sup>). They presented the tensor numerical methods, which were recognized as the basic tool to render numerical simulations in higher dimensions tractable. As Khoromskij<sup>31</sup> pointed out, the guiding principle of the tensor numerical methods is the reduction of the computational process onto a low parametric rank-structured manifold by using an approximation of multivariate functions and operators that relies on a certain separation of variables. An introductory description of traditional tensor formats with the focus on tensor-structured numerical methods for the calculation of multidimensional functions and operators is presented in Khoromskij<sup>32</sup> and Khoromskaia.<sup>33</sup> More recently, there are some monographies on the tensor numerical method in scientific computing. For more details, one can refer to the work of Khoromskij<sup>34</sup> and the references therein.

The tensor-structured numerical methods has opened new perspectives for solving the basic equations of mathematical physics in  $\mathbb{R}^d$ ,  $d \geq 3$ , in particular, the many-particle Schrödinger equation,<sup>35,36</sup> Hartree-Fock equation,<sup>37</sup> and multidimensional elliptic spectral problems.<sup>38</sup> Meanwhile, the tensor-product structured approximations of certain multidimensional integral operators have been introduced and analyzed.<sup>39</sup> Hackbusch et al.<sup>40</sup> have solved multidimensional boundary and eigenvalue problems using a reduced low-dimensional tensor-product space through separated representation and hierarchical Kronecker tensor from the underlying high spatial dimensions. Possible applications of tensor numerical methods include high-dimensional problems arising in biosciences, computational quantum chemistry, stochastic modeling and uncertainty quantification, dynamical systems, machine learning, financial mathematics, etc. See the survey papers<sup>32,41,42</sup> and the references therein for more applications and tensor-based methods.

Motivated by the research studies on tensor numerical methods,<sup>18,28,29,32,35-42</sup> in this paper, we continue to examine the iterative methods in their tensor forms to find the approximate solution of (6). That is, we will extend the global generalized minimal residual (GMRES), MINRES, and SYMMLQ methods to solve the tensor Equation (6), where the coefficient

tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  is a symmetric tensor,  $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ , and  $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ . To this end, we first construct the tensor forms of the Arnoldi and Lanczos processes to form an orthonormal tensor basis of the tensor Krylov subspace. Then, we establish the tensor forms of the global GMRES, MINIRES, and SYMMLQ methods to solve the tensor equation and give the implementation details. Numerical examples are provided to illustrate that the proposed algorithms are efficient for solving (6) and they are superior to the BiCG-BTF method in the work of Brazell et al.<sup>18</sup> and Algorithm 3.1 in the work of Wang et al.<sup>28</sup> We also apply our proposed iterative methods to solve the tensor equation in three-dimensional Poisson problem (5). The numerical results confirm that solving the tensor equation with the tensor-based iterative methods can preserve low bandwidth of the discrete tensor in high-dimension PDEs, thereby keeping the computational cost and memory requirement low. They also show that the one-to-one correspondence between the solution and the computational grid facilitate the integration of complicated boundary conditions. Furthermore, we compare the tensor-based iterative approaches and the matrix-based iterative approaches. We find that solving multilinear systems with the tensor-based iterative methods can keep the computational cost low and the tensor-based iterative methods are better than the matrix-based iterative methods. Unfortunately, we cannot compare the complexities of the algorithms in theory in order to illustrate the superiority of the tensor-based methods.

The contributions of this paper are twofold. First, we establish the tensor forms of the Arnoldi and Lanczos processes, which provide the framework for solving tensor equations via the Einstein product by the tensor Krylov subspace methods. It has been shown that the tensor Krylov subspace methods are efficient for solving the tensor equation via the Einstein product, especially for the discrete equation of the high-dimension PDEs. Second, we describe the way of the construction of the tensor-based iterative methods including the tensor forms of the global GMRES, MINIRES, and SYMMLQ methods.

The remainder of this paper is organized as follows. Section 2 presents some notations and definitions and properties, which are essential for deriving our main results. In Section 3, how the well-known Arnoldi and Lanczos processes can be used based on the tensor format is described. Section 4 derives the tensor form of the global GMRES method (denoted by the GI-GMRES-BTF method) for finding the solution of the tensor equation. The tensor forms of the MINIRES (denoted by MINIRES-BTF method) and SYMMLQ methods (denoted by SYMMLQ-BTF method) are also given in Section 4. Numerical examples of the proposed methods are shown and analyzed in Section 5. This paper ends up with some conclusions in Section 6.

## 2 | PRELIMINARIES

For convenience, we use the following notations throughout this paper. For a positive integer  $N$ , an order  $N$  tensor  $\mathcal{A} = (a_{i_1 \dots i_N}) (1 \leq i_j \leq I_j, j = 1, \dots, N)$  is a multidimensional array with  $I_1 I_2 \dots I_N$  entries. Let  $\mathbb{C}^{I_1 \times \dots \times I_N}$  and  $\mathbb{R}^{I_1 \times \dots \times I_N}$  be the sets of the order  $N$  dimension  $I_1 \times \dots \times I_N$  tensors over the complex field  $\mathbb{C}$  and the real field  $\mathbb{R}$ , respectively. For a tensor  $\mathcal{A} = (a_{i_1 \dots i_N j_1 \dots j_M}) \in \mathbb{C}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ , let  $\mathcal{B} = (b_{i_1 \dots i_N j_1 \dots j_M}) \in \mathbb{C}^{J_1 \times \dots \times J_M \times I_1 \times \dots \times I_N}$  be the conjugate transpose of  $\mathcal{A}$ , where  $b_{i_1 \dots i_N j_1 \dots j_M} = \bar{a}_{j_1 \dots j_M i_1 \dots i_N}$ . The tensor  $\mathcal{B}$  is denoted by  $\mathcal{A}^*$ . When  $b_{i_1 \dots i_N j_1 \dots j_M} = a_{j_1 \dots j_M i_1 \dots i_N}$ , the tensor  $\mathcal{B}$  is called the transpose of  $\mathcal{A}$ , denoted by  $\mathcal{A}^T$ . We say a tensor  $\mathcal{D} = (d_{i_1 \dots i_N j_1 \dots j_M}) \in \mathbb{C}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  is a diagonal tensor if  $d_{i_1 \dots i_N j_1 \dots j_M} = 0$  in the case that the indices  $i_1 \dots i_N$  are different from  $j_1 \dots j_N$ . Let  $\mathcal{I}$  be the unit tensor with all the entries that are zero except for the diagonal entries  $d_{i_1 \dots i_N i_1 \dots i_N} = 1$ . If all the entries of tensor are zero, we say this tensor is zero tensor, denoted by  $\mathcal{O}$ . The trace of a tensor  $\mathcal{A} \in \mathbb{C}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  is  $\text{tr}(\mathcal{A}) = \sum_{i_1 \dots i_N} a_{i_1 \dots i_N i_1 \dots i_N}$ .<sup>18</sup> Denote  $I^{(m)}$  as the identity matrix of order  $m$ . Let  $I$  be the product of  $I_1, I_2, \dots, I_N$ , that is,  $I = I_1 I_2 \dots I_N$ . Similarly, we denote  $J = J_1 J_2 \dots J_M$ .

**Definition 1** (See the work of Brazell et al.<sup>18</sup>). Let  $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ . The inner product of two tensors  $\mathcal{A}$  and  $\mathcal{B}$  is defined by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \text{tr}(\mathcal{B}^T *_N \mathcal{A}). \quad (8)$$

Then, for tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$ , the tensor norm induced by this inner product is Frobenius norm  $\|\mathcal{A}\| = (\sum_{i_1 \dots i_N j_1 \dots j_N} |a_{i_1 \dots i_N j_1 \dots j_N}|^2)^{\frac{1}{2}}$ . When  $\langle \mathcal{A}, \mathcal{B} \rangle = 0$ , we say that  $\mathcal{A}$  and  $\mathcal{B}$  are orthogonal. Wang et al.<sup>28</sup> derived the symmetric property of this inner product. That is, they established the following results.

**Lemma 1** (See the work of Wang et al.<sup>28</sup>). Let  $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ . Then,

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

In addition, it follows from the definition of inner product and the properties of tensor trace 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 all scalar  $\alpha \in \mathbb{R}$ , we have the following.

(1) Linearity in the first argument:

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

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

In addition, for two tensors  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ , by simple calculations, we obtain

$$\|\mathcal{X} - \mathcal{Y}\|^2 = \|\mathcal{X}\|^2 - 2\langle \mathcal{X}, \mathcal{Y} \rangle + \|\mathcal{Y}\|^2.$$

**Definition 2** (See the work of Kolda et al.<sup>42</sup>). The  $n$ -mode product of a tensor  $\mathcal{X} = (x_{i_1 i_2 \cdots i_N}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  with a matrix  $A = (u_{j_{i_n}}) \in \mathbb{R}^{J \times I_n}$  is denoted by  $\mathcal{X} \times_n A$  and is of the size  $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ , and its elements are defined by

$$(\mathcal{X} \times_n A)_{i_1 \cdots i_{n-1} j_{i_{n+1}} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j_{i_n} i_n}.$$

The  $n$ -mode product satisfies the following properties.

**Lemma 2** (See the work of De Lathauwer et al.<sup>43</sup>). Given tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ .

(1) Given matrices  $A \in \mathbb{R}^{J_m \times I_m}$ ,  $B \in \mathbb{R}^{J_n \times I_n}$  ( $m \neq n$ ), one has

$$\mathcal{X} \times_n A \times_m B = \mathcal{X} \times_m B \times_n A.$$

(2) Given matrices  $A \in \mathbb{R}^{J \times I_n}$ ,  $B \in \mathbb{R}^{I_n \times J}$ , one has

$$\mathcal{X} \times_n A \times_n B = \mathcal{X} \times_n (BA).$$

**Definition 3** (See the work of Kolda et al.<sup>42</sup>). The  $n$ -mode product of a tensor  $\mathcal{X} = (x_{i_1 i_2 \cdots i_N}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  with a vector  $v = (v_{i_n}) \in \mathbb{R}^{I_n}$  is denoted by  $\mathcal{X} \tilde{\times}_n v$ , which is an  $(N-1)$ -mode tensor and

$$(\mathcal{X} \tilde{\times}_n v)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}.$$

**Lemma 3** (See the work of Beik et al.<sup>44</sup>). If  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ ,  $A \in \mathbb{R}^{J_n \times I_n}$ ,  $y \in \mathbb{R}^{J_n}$ , then

$$\mathcal{X} \times_n A \tilde{\times}_n y = \mathcal{X} \tilde{\times}_n (A^T y). \quad (9)$$

For a given tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ , the notation

$$\mathcal{X}_{:\cdots:k}, \quad k = 1, 2, \dots, I_N$$

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 the work of Kolda et al.<sup>42</sup>

**Lemma 4** (See the work of Beik et al.<sup>44</sup>). If  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  and  $y = e_j$  such that  $e_j$  is the  $j$ th column of  $I_N \times I_N$  identity matrix, then

$$\mathcal{X} \times_n y = \mathcal{X}_j, \quad j = 1, 2, \dots, I_N, \quad (10)$$

where  $\mathcal{X}_j$  is the  $j$ th frontal slice of  $\mathcal{X}$ .

### 3 | THE TENSOR FORMS OF THE GLOBAL ARNOLDI AND LANCZOS PROCESSES

Given coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  and tensor  $\mathcal{V} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_M}$ , the  $m$ th tensor Krylov subspace associated to the pair  $(\mathcal{A}, \mathcal{V})$  is defined by

$$\mathcal{K}_m(\mathcal{A}, \mathcal{V}) = \text{span}\{\mathcal{V}, \mathcal{A} *_N \mathcal{V}, \dots, \mathcal{A}^{m-1} *_N \mathcal{V}\}, \quad (11)$$

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

We start with the Arnoldi-BTF process to produce an orthogonal basis for  $\mathcal{K}_m(\mathcal{A}, \mathcal{V})$ , which is described in Algorithm 1.

---

**Algorithm 1** Global Arnoldi-BTF process

---

- 1: **Input:** coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$ , tensor  $\mathcal{V} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_M}$ , and the dimension of Krylov subspace  $m$ .
  - 2: Set  $\beta = \|\mathcal{V}\|$ ,  $\mathcal{V}_1 = \mathcal{V}/\beta$ ;
  - 3: **for**  $j = 1, \dots, m$  **do**
  - 4:    $\mathcal{W} = \mathcal{A} *_N \mathcal{V}_j$ ;
  - 5:   **for**  $i = 1, \dots, j$  **do**
  - 6:      $h_{ij} = \langle \mathcal{V}_i, \mathcal{W} \rangle$ ;
  - 7:      $\mathcal{W} = \mathcal{W} - h_{ij} \mathcal{V}_i$ ;
  - 8:   **end for**
  - 9:    $h_{j+1,j} = \|\mathcal{W}\|$ , **if**  $h_{j+1,j} = 0$ , then **stop**;
  - 10:    $\mathcal{V}_{j+1} = \mathcal{W}/h_{j+1,j}$ ;
  - 11: **end for**
- 

Let  $\hat{H}_m$  be the  $(m+1) \times m$  upper Hessenberg matrix whose nonzero entries  $h_{ij}$  are computed by Algorithm 1. Suppose that  $H_m$  is the  $m \times m$  matrix obtained from  $\hat{H}_m$  by deleting its last row. It is not difficult to verify that the global Arnoldi-BTF process produces the orthonormal basis  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  for the tensor Krylov subspace  $\mathcal{K}_m(\mathcal{A}, \mathcal{V})$ . That is, we have the following result.

**Proposition 1.** Suppose that  $m$  steps of Algorithm 1 have been taken. Then, the tensors  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  form an orthonormal basis of the tensor Krylov space  $\mathcal{K}_m(\mathcal{A}, \mathcal{V})$ .

*Proof.* This will be shown by induction on  $m$ . For  $m = 1$ , by Line 2 of Algorithm 1, we immediately have  $\|\mathcal{V}_1\| = 1$ . Assume now that the result is true for some  $m$ . Then, for  $m+1$ , by Algorithm 1 and the induction principle, one has

$$\langle \mathcal{V}_j, \mathcal{V}_{m+1} \rangle = \left\langle \mathcal{V}_j, \frac{\mathcal{A} *_N \mathcal{V}_m - \sum_{i=1}^m h_{im} \mathcal{V}_i}{h_{m+1,m}} \right\rangle = \frac{\langle \mathcal{V}_j, \mathcal{A} *_N \mathcal{V}_m \rangle - \sum_{i=1}^m h_{im} \langle \mathcal{V}_j, \mathcal{V}_i \rangle}{h_{m+1,m}} = \frac{h_{jm} - h_{jm}}{h_{m+1,m}} = 0$$

for all  $j = 1, \dots, m$ . In addition, by Lines 9 and 10 of Algorithm 1, we immediately have  $\|\mathcal{V}_{m+1}\| = 1$ . Therefore, the result is true for  $m+1$ . By the induction principle, the proof is completed.  $\square$

Let  $\mathbb{V}_m$  be the  $(M+N+1)$ -mode tensor with frontal slices  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  and  $\mathcal{A} *_N \mathbb{V}_m$  be the  $(M+N+1)$ -mode tensor with frontal slices  $\mathcal{A} *_N \mathcal{V}_1, \mathcal{A} *_N \mathcal{V}_2, \dots, \mathcal{A} *_N \mathcal{V}_m$ , respectively. That is,

$$\mathbb{V}_m := [\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m], \quad (12)$$

$$\mathcal{A} *_N \mathbb{V}_m := [\mathcal{A} *_N \mathcal{V}_1, \mathcal{A} *_N \mathcal{V}_2, \dots, \mathcal{A} *_N \mathcal{V}_m]. \quad (13)$$

Then, by Algorithm 1, the following result is obvious.

**Proposition 2.** Suppose that the  $m$  steps of Algorithm 1 have been taken. Then, the following statements hold:

$$\mathcal{A} *_N \mathbb{V}_m = \mathbb{V}_{m+1} \times_{(M+N+1)} \hat{H}_m^T, \quad (14)$$

$$\mathcal{A} *_N \mathbb{V}_m = \mathbb{V}_m \times_{(M+N+1)} H_m^T + h_{m+1,m} \mathcal{Z} \times_{(M+N+1)} E_m, \quad (15)$$

where  $\mathbb{V}_m$  and  $\mathcal{A} *_N \mathbb{V}_m$  are defined as in (12) and (13), respectively;  $\mathcal{Z}$  is an  $(M+N+1)$ -mode tensor with  $(M+N)$ -mode column tensors  $\mathcal{O}, \dots, \mathcal{O}, \mathcal{V}_{m+1}$ ; and  $E_m$  is an  $m \times m$  matrix of the form  $E_m = [0, \dots, 0, e_m]$  with  $e_m$  being the  $m$ th column of the identity matrix of order  $m$ .

*Proof.* From the definition of the  $(M+N+1)$ -mode product of a tensor with a matrix, we have

$$\begin{aligned} & \left( \mathbb{V}_{m+1} \times_{(M+N+1)} \hat{H}_m^T \right)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} \\ &= \sum_{i=1}^{m+1} (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M i} (\hat{H}_m)_{ij} \\ &= (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M 1} (\hat{H}_m)_{1j} + \dots + (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} (\hat{H}_m)_{jj} \\ &\quad + (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M, j+1} (\hat{H}_m)_{j+1, j} \\ &= h_{1j} (\mathcal{V}_1)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \dots + h_{jj} (\mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + h_{j+1, j} (\mathcal{V}_{j+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M}. \end{aligned} \quad (16)$$

By Lines 4–10 of Algorithm 1, one has

$$\mathcal{A} *_N \mathcal{V}_j = \sum_{i=1}^{j+1} h_{ij} \mathcal{V}_i.$$

It then follows from the relation (13) and the definition of frontal slice tensor that

$$\begin{aligned} & (\mathcal{A} *_N \mathbb{V}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} = (\mathcal{A} *_N \mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} \\ &= h_{1j} (\mathcal{V}_1)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \dots + h_{jj} (\mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + h_{j+1, j} (\mathcal{V}_{j+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M}. \end{aligned} \quad (17)$$

Combining this with (3) yields (14). Similarly, (15) can be proved.  $\square$

If the coefficient tensor  $\mathcal{A}$  is symmetric, then

$$\begin{aligned} h_{ij} &= \langle \mathcal{V}_i, \mathcal{A} *_N \mathcal{V}_j \rangle = \text{tr} (\mathcal{V}_j^T *_N \mathcal{A}^T *_N \mathcal{V}_i) = \text{tr} (\mathcal{V}_j^T *_N \mathcal{A} *_N \mathcal{V}_i) \\ &= \langle \mathcal{A} *_N \mathcal{V}_i, \mathcal{V}_j \rangle = \langle \mathcal{V}_j, \mathcal{A} *_N \mathcal{V}_i \rangle = h_{ji}. \end{aligned} \quad (18)$$

Hence, the global Arnoldi-BTF process reduces to the global Lanczos-BTF process, which is presented in Algorithm 2.

**Algorithm 2** Global Lanczos-BTF process

---

```

1: Input: symmetric tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$ , tensor  $\mathcal{V} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ , and the dimension of Krylov subspace  $m$ .
2: Set  $\beta = \|\mathcal{V}\|$ ,  $\mathcal{V}_1 = \mathcal{V}/\beta$ ;
3: Set  $\beta_0 = 0$ ,  $\mathcal{V}_0 = \mathcal{O} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ ;
4: for  $j = 1, \dots, m$  do
5:    $\mathcal{W} = \mathcal{A} *_{\mathcal{N}} \mathcal{V}_j$ ;
6:    $\alpha_j = \langle \mathcal{V}_j, \mathcal{W} \rangle$ ;
7:    $\mathcal{W} = \mathcal{W} - \alpha_j \mathcal{V}_j - \beta_{j-1} \mathcal{V}_{j-1}$ ;
8:    $\beta_j = \|\mathcal{W}\|$ ;
9:   if  $\beta_j = 0$ , then stop;
10:   $\mathcal{V}_{j+1} = \mathcal{W}/\beta_j$ ;
11: end for

```

---

Denote

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \beta_{m-2} & \alpha_{m-1} & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_m & \end{bmatrix}, \hat{T}_m = \begin{bmatrix} T_m \\ \beta_m \mathbf{e}_m^T \end{bmatrix}. \quad (19)$$

By Algorithm 2, it is easy to see that

$$\mathcal{A} *_{\mathcal{N}} \mathcal{V}_1 = \alpha_1 \mathcal{V}_1 + \beta_1 \mathcal{V}_2, \mathcal{A} *_{\mathcal{N}} \mathcal{V}_j = \beta_{j-1} \mathcal{V}_{j-1} + \alpha_j \mathcal{V}_j + \beta_j \mathcal{V}_{j+1}, \quad j = 2, \dots, m. \quad (20)$$

Moreover, the tensors  $\mathcal{V}_1, \dots, \mathcal{V}_m$  generated by Algorithm 2 form an orthonormal basis for the tensor Krylov subspace  $\mathcal{K}_m(\mathcal{A}, \mathcal{V})$ .

**Proposition 3.** Suppose that  $m$  steps of Algorithm 2 have been taken. Then, the tensors  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  form an orthonormal basis of the tensor Krylov subspace  $\mathcal{K}_m(\mathcal{A}, \mathcal{V})$ .

*Proof.* This will be shown by induction on  $m$ . For  $m = 1$ , by Line 2 of Algorithm 2, we immediately have  $\|\mathcal{V}_1\| = 1$ . Assume now that the result is true for some  $m$ . Then, for  $m + 1$ , by Algorithm 2 and the symmetry of tensor  $\mathcal{A}$ , one has

$$\begin{aligned} \langle \mathcal{V}_j, \mathcal{V}_{m+1} \rangle &= \left\langle \mathcal{V}_j, \frac{\mathcal{A} *_{\mathcal{N}} \mathcal{V}_m - \beta_{m-1} \mathcal{V}_{m-1} - \alpha_m \mathcal{V}_m}{\beta_m} \right\rangle \\ &= \frac{\langle \mathcal{A} *_{\mathcal{N}} \mathcal{V}_j, \mathcal{V}_m \rangle - \beta_{m-1} \langle \mathcal{V}_j, \mathcal{V}_{m-1} \rangle - \alpha_m \langle \mathcal{V}_j, \mathcal{V}_m \rangle}{\beta_m} \\ &= \frac{\langle \beta_{j-1} \mathcal{V}_{j-1} + \alpha_j \mathcal{V}_j + \beta_j \mathcal{V}_{j+1}, \mathcal{V}_m \rangle - \beta_{m-1} \langle \mathcal{V}_j, \mathcal{V}_{m-1} \rangle - \alpha_m \langle \mathcal{V}_j, \mathcal{V}_m \rangle}{\beta_m}. \end{aligned} \quad (21)$$

If  $j = m$ , by the relation (3) and the induction principle, we obtain

$$\langle \mathcal{V}_m, \mathcal{V}_{m+1} \rangle = \frac{\alpha_m - \alpha_m}{\beta_m} = 0.$$

If  $j = m - 1$ , it follows from (3) and induction principle that

$$\langle \mathcal{V}_{m-1}, \mathcal{V}_{m+1} \rangle = \frac{\beta_{m-1} - \beta_{m-1}}{\beta_m} = 0.$$

Similarly, for  $j = 1, \dots, m - 2$ , we also have  $\langle \mathcal{V}_j, \mathcal{V}_{m+1} \rangle = 0$ . In addition, by Lines 8 and 10 of Algorithm 2, we immediately have  $\|\mathcal{V}_{m+1}\| = 1$ . Therefore, the result is true for  $m + 1$ . By the induction principle, the proof is completed.  $\square$

**Proposition 4.** Suppose that the  $m$  steps of Algorithm 2 have been taken. Then, the following statements hold:

$$\mathcal{A} *_N \mathbb{V}_m = \mathbb{V}_{m+1} \times_{(M+N+1)} \hat{T}_m^T, \quad (22)$$

$$\mathcal{A} *_N \mathbb{V}_m = \mathbb{V}_m \times_{(M+N+1)} T_m^T + \beta_m \mathcal{Z} \times_{(M+N+1)} E_m, \quad (23)$$

where  $\mathbb{V}_m$  and  $\mathcal{A} *_N \mathbb{V}_m$  are defined as in (12) and (13), respectively;  $\mathcal{Z}$  is an  $(M+N+1)$ -mode tensor with  $(M+N)$ -mode column tensors  $\mathcal{O}, \dots, \mathcal{O}, \mathcal{V}_{m+1}$ ; and  $E_m$  is an  $m \times m$  matrix of the form  $E_m = [0, \dots, 0, e_m]$  with  $e_m$  being the  $m$ th column of the identity matrix of order  $m$ .

*Proof.* From the definition of the  $(M+N+1)$ -mode product of a tensor with a matrix, for  $j = 1, \dots, m$ , we have

$$\begin{aligned} & \left( \mathbb{V}_{m+1} \times_{(M+N+1)} \hat{T}_m^T \right)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} \\ &= \sum_{i=1}^{m+1} (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M i} (\hat{T}_m)_{ij} \\ &= (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M, j-1} (\hat{T}_m)_{j-1, j} + (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} (\hat{T}_m)_{jj} \\ &\quad + (\mathbb{V}_{m+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M, j+1} (\hat{T}_m)_{j+1, j} \\ &= \beta_{j-1} (\mathcal{V}_{j-1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \alpha_j (\mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \beta_j (\mathcal{V}_{j+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M}, \end{aligned} \quad (24)$$

where  $\beta_0 = 0, \mathcal{V}_0 = \mathcal{O}$ . According to (13) and (20), for  $j = 1, \dots, m$ , we obtain

$$\begin{aligned} (\mathcal{A} *_N \mathbb{V}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M j} &= (\mathcal{A} *_N \mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} \\ &= \beta_{j-1} (\mathcal{V}_{j-1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \alpha_j (\mathcal{V}_j)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \beta_j (\mathcal{V}_{j+1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M}, \end{aligned} \quad (25)$$

where  $\beta_0 = 0, \mathcal{V}_0 = \mathcal{O}$ . Hence, by (24) and (25), we conclude that (22) holds. Similarly, we can prove (23).  $\square$

## 4 | THE PROPOSED METHODS

### 4.1 | GI-GMRES-BTF method: the tensor form of the GI-GMRES method

In this subsection, we briefly describe how the well-known GI-GMRES method can be extended based on tensor form.

Given symmetric coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  and the right-hand side tensor  $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$ . Let  $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  be a given initial tensor and  $\mathcal{R}_0$  be the corresponding residual. At  $m$ th iterative step of the GI-GMRES-BTF method, we seek for an approximate solution  $\mathcal{X}_m$  such that

$$\mathcal{X}_m \in \mathcal{X}_0 + \mathcal{K}_m(\mathcal{A}, \mathcal{R}_0) \quad (26)$$

and

$$\|\mathcal{R}_m\| = \min\{\|\mathcal{C} - \mathcal{A} *_N \mathcal{X}\| : \mathcal{X} \in \mathcal{X}_0 + \mathcal{K}_m(\mathcal{A}, \mathcal{R}_0)\}, \quad (27)$$

where  $\mathcal{R}_m = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_m$ . By Proposition 3, the tensors  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  generated by Algorithm 2 form an orthonormal basis for  $\mathcal{K}_m(\mathcal{A}, \mathcal{R}_0)$ . Hence, there exists a vector  $y_m = (y_m^{(1)}, \dots, y_m^{(m)})^T \in \mathbb{R}^m$  such that

$$\mathcal{X}_m = \mathcal{X}_0 + \sum_{j=1}^m y_m^{(j)} \mathcal{V}_j = \mathcal{X}_0 + \mathbb{V}_m \bar{x}_{(M+N+1)} y_m. \quad (28)$$

Because  $\mathcal{R}_0 = \beta \mathcal{V}_1$  ( $\beta = \|\mathcal{R}_0\|$ ), by Algorithm 1, Proposition 4, Lemma 3, and Lemma 4, we know that the residual at the  $m$ th iterative step is

$$\begin{aligned}
 \mathcal{R}_m &= C - \mathcal{A} *_N \mathcal{X}_m = C - \mathcal{A} *_N (\mathcal{X}_0 + \mathbb{V}_m \bar{\mathcal{X}}_{(M+N+1)} y_m) \\
 &= \mathcal{R}_0 - (\mathcal{A} *_N \mathbb{V}_m) \bar{\mathcal{X}}_{(M+N+1)} y_m \\
 &= \beta \mathcal{V}_1 - \mathbb{V}_{m+1} \mathcal{X}_{(M+N+1)} \hat{T}_m^T \bar{\mathcal{X}}_{(M+N+1)} y_m \\
 &= \beta \mathbb{V}_{m+1} \bar{\mathcal{X}}_{(M+N+1)} e_1^{(m+1)} - \mathbb{V}_{m+1} \bar{\mathcal{X}}_{(M+N+1)} (\hat{T}_m y_m) \\
 &= \mathbb{V}_{m+1} \bar{\mathcal{X}}_{(M+N+1)} \left( \beta e_1^{(m+1)} - \hat{T}_m y_m \right), \tag{29}
 \end{aligned}$$

where  $e_1^{(m+1)}$  is the first column of the identity matrix of order  $m+1$ . Denote  $p_{m+1} := \beta e_1^{(m+1)} - \hat{T}_m y_m$  and  $p_{m+1} := (p_{m+1}^{(1)}, p_{m+1}^{(2)}, \dots, p_{m+1}^{(m)}, p_{m+1}^{(m+1)})^T$ . Hence, (29) can be reformulated as

$$\mathcal{R}_m = p_{m+1}^{(1)} \mathcal{V}_1 + p_{m+1}^{(2)} \mathcal{V}_2 + \dots + p_{m+1}^{(m)} \mathcal{V}_m + p_{m+1}^{(m+1)} \mathcal{V}_{m+1}. \tag{30}$$

According to Proposition 3,  $\{\mathcal{V}_j\}_{j=1}^{m+1}$  is an orthonormal sequence. By some calculations, we have

$$\|\mathcal{R}_m\| = \sqrt{\sum_{j=1}^{m+1} \left(p_{m+1}^{(j)}\right)^2} = \|p_{m+1}\| = \|\beta e_1^{(m+1)} - \hat{T}_m y_m\|. \tag{31}$$

Therefore, we conclude that the GI-GMRES-BTF method computes the  $m$ th approximate solution such that

$$\mathcal{X}_m = \mathcal{X}_0 + \mathbb{V}_m \bar{\mathcal{X}}_{(M+N+1)} y_m, \tag{32}$$

where  $y_m$  is the solution of

$$\min_{y \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \hat{T}_m y\|_2, \tag{33}$$

where  $e_1^{(m+1)}$  is the first column of identity matrix of order  $m+1$ .

Combining Algorithm 2 with the minimizing norm condition (27) defines the GI-GMRES-BTF method. The descriptions of the GI-GMRES-BTF method are summarized in Algorithm 3.

---

### Algorithm 3 GI-GMRES-BTF method

---

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

- 1: **Input:** the initial tensor  $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ , the maximum number of iterations  $\text{Iter}_{\max}$ , the tolerance error  $\varepsilon > 0$ , the maximal dimension of the Krylov subspace  $m$ .
  - 2: Compute  $\mathcal{R}_0 = C - \mathcal{A} *_N \mathcal{X}_0$ ,  $r = \|\mathcal{R}_0\|$ ;
  - 3: **for**  $k = 1, \dots, \text{Iter}_{\max}$  **do**
  - 4:    $\beta = \|\mathcal{R}_0\|$ ;
  - 5:   Compute  $\hat{T}_m$  and  $\mathbb{V}_m$  by applying Algorithm 3 to  $\mathcal{K}_m(\mathcal{A}, \mathcal{R}_0)$ ;
  - 6:   Solve the problem  $y_m = \arg \min_{y \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \hat{T}_m y\|_2$ ;
  - 7:   Compute the approximate solution  $\mathcal{X}_m = \mathcal{X}_0 + \mathbb{V}_m \bar{\mathcal{X}}_{(M+N+1)} y_m$ ;
  - 8:   Compute  $\mathcal{R}_m = C - \mathcal{A} *_N \mathcal{X}_m$ ;
  - 9:   **if**  $\|\mathcal{R}_m\|/r < \varepsilon$  **then**
  - 10:     output the approximate solution  $\mathcal{X}_m$ ;
  - 11:   **else**
  - 12:      $\mathcal{R}_0 = \mathcal{R}_m$ ;
  - 13:   **end if**
  - 14: **end for**
-

## 4.2 | MINIRES-BTF method: the tensor form of the MINIRES method

As described above, in the GI-GMRES-BTF method, we need to solve the least squares problem (33). Because  $\hat{T}_m$  is a tridiagonal matrix, there are a lot of very sophisticated numerical methods for finding the solution of the least squares problem (33). For example, the QR decomposition method, that is, we perform the  $m$ th Givens transformation  $G_{1,2}, G_{2,3}, \dots, G_{m,m+1}$ , such that

$$G_{m,m+1}G_{m-1,m} \cdots G_{2,3}G_{1,2}\hat{T}_m = \begin{bmatrix} R_m \\ 0 \end{bmatrix}, \quad (34)$$

where

$$G_{k,k+1} = \text{diag} \left( I^{(k-1)}, \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}, I^{(m-k)} \right) \in \mathbb{R}^{(m+1) \times (m+1)}, c_k^2 + s_k^2 = 1, k = 1, 2, \dots, m, \quad (35)$$

$$R_m = \begin{bmatrix} \gamma_1 & \delta_1 & \varepsilon_1 & & & \\ & \gamma_2 & \delta_2 & \ddots & & \\ & & \ddots & \ddots & \varepsilon_{m-2} & \\ & & & \gamma_{m-1} & \delta_{m-1} & \\ & & & & \gamma_m & \end{bmatrix}. \quad (36)$$

Denote

$$Q_m := G_{m,m+1}G_{m-1,m} \cdots G_{2,3}G_{1,2}, \quad Q_m \left( \beta e_1^{(m+1)} \right) := \begin{bmatrix} z_m \\ \rho_m \end{bmatrix}, \quad z_m = (\zeta_1, \zeta_2, \dots, \zeta_m)^T. \quad (37)$$

It is obvious that  $Q_m$  is an orthogonal matrix. In addition, simple calculations show that

$$\begin{cases} \zeta_1 = \beta c_1, \\ \zeta_k = (-1)^{k-1} \beta s_1 s_2 \cdots s_{k-1} c_k, & k = 2, 3, \dots, m, \\ \rho_m = (-1)^m \beta s_1 s_2 \cdots s_m. \end{cases} \quad (38)$$

A consequence of the  $m$ th Givens reflection above is that the subproblem (33) may equivalently be rewritten as

$$\min_{y \in \mathbb{R}^m} \left\| \beta e_1^{(m+1)} - \hat{T}_m y \right\|_2 = \min_{y \in \mathbb{R}^m} \left\| Q_m \left( \beta e_1^{(m+1)} - \hat{T}_m y \right) \right\|_2 = \min_{y \in \mathbb{R}^m} \left\| \begin{bmatrix} z_m \\ \rho_m \end{bmatrix} - \begin{bmatrix} R_m \\ 0 \end{bmatrix} y \right\|_2,$$

whose solution is readily identified, that is,  $y_m := R_m^{-1} z_m$ , whose residual is  $|\rho_m|$ , that is,  $\|\beta e_1^{(m+1)} - \hat{T}_m y_m\|_2 = |\rho_m|$ .

Because  $y_m$  is the solution of an upper triangular system, all of its components likely change at each iteration. In addition, it needs to store  $\mathbb{V}_m = [\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m]$  in order to obtain the solution  $\mathcal{X}_m$ . Fortunately, it is possible to update  $\mathcal{X}_m$  directly without requiring  $y_m$ . Following the idea of the MINIRES method developed by Paige et al.,<sup>45</sup> let  $\mathbb{F}_m := \mathbb{V}_m \times_{(M+N+1)} R_m^{-T}$  and let  $\mathbb{F}_m$  be the  $(M+N+1)$ -mode tensor with frontal slices  $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_m$  (i.e.,  $\mathbb{F}_m = [\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_m]$ ). It then follows from Lemma 2 and Lemma 4 that

$$\begin{aligned} \mathbb{V}_m &= [\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m] = \left[ \mathbb{V}_m \bar{\times}_{(M+N+1)} e_1^{(m)}, \mathbb{V}_m \bar{\times}_{(M+N+1)} e_2^{(m)}, \dots, \mathbb{V}_m \bar{\times}_{(M+N+1)} e_m^{(m)} \right] \\ &= \mathbb{V}_m \times_{(M+N+1)} I^{(m)} = \mathbb{V}_m \times_{(M+N+1)} (R_m^{-T} R_m^T) \\ &= \mathbb{V}_m \times_{(M+N+1)} R_m^{-T} \times_{(M+N+1)} R_m^T \\ &= \mathbb{F}_m \times_{(M+N+1)} R_m^T. \end{aligned} \quad (39)$$

Meanwhile, from the definition of  $(M + N + 1)$ -mode product of a tensor with a matrix, for  $k = 1, \dots, m$ , we have

$$\begin{aligned}
 & (\mathbb{F}_m \times_{(M+N+1)} R_m^T)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M k} \\
 &= \sum_{i=1}^m (\mathbb{F}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M k} (R_m)_{ik} \\
 &= (\mathbb{F}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M, k-2} (R_m)_{k-2, k} + (\mathbb{F}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M, k-1} (R_m)_{k-1, k} \\
 &\quad + (\mathbb{F}_m)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M k} (R_m)_{kk} \\
 &= \varepsilon_{k-2} (\mathcal{F}_{k-2})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \delta_{k-1} (\mathcal{F}_{k-1})_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M} + \gamma_k (\mathcal{F}_k)_{i_1 i_2 \dots i_N j_1 j_2 \dots j_M},
 \end{aligned} \tag{40}$$

where  $\delta_0 = 0$ ,  $\varepsilon_{-1} = \varepsilon_0 = 0$ . Together, (39) with (40) yields

$$\begin{cases} \gamma_1 \mathcal{F}_1 = \mathcal{V}_1, \\ \delta_1 \mathcal{F}_1 + \gamma_2 \mathcal{F}_2 = \mathcal{V}_2, \\ \varepsilon_{k-2} \mathcal{F}_{k-2} + \delta_{k-1} \mathcal{F}_{k-1} + \gamma_k \mathcal{F}_k = \mathcal{V}_k, k = 3, 4, \dots, m. \end{cases}$$

Hence,

$$\begin{cases} \mathcal{F}_1 = \mathcal{V}_1 / \gamma_1, \\ \mathcal{F}_2 = (\mathcal{V}_2 - \delta_1 \mathcal{F}_1) / \gamma_2, \\ \mathcal{F}_k = (\mathcal{V}_k - \varepsilon_{k-2} \mathcal{F}_{k-2} - \delta_{k-1} \mathcal{F}_{k-1}) / \gamma_k, k = 3, 4, \dots, m. \end{cases} \tag{41}$$

With the above preparations, we may rewrite  $\mathcal{X}_m$  as

$$\begin{aligned}
 \mathcal{X}_m &= \mathbb{V}_m \bar{\times}_{(M+N+1)} y_m = \mathbb{V}_m \bar{\times}_{(M+N+1)} (R_m^{-1} z_m) \\
 &= \mathbb{V}_m \times_{(M+N+1)} R_m^{-T} \bar{\times}_{(M+N+1)} z_m = \mathbb{F}_m \bar{\times}_{(M+N+1)} z_m.
 \end{aligned} \tag{42}$$

In what follows, we illustrate the updates of  $\varepsilon_{m-1}$ ,  $\delta_m$ ,  $\gamma_{m+1}$ ,  $\zeta_{m+1}$ , and  $\rho_{m+1}$ . Notice that

$$\hat{T}_{m+1} = \begin{bmatrix} \hat{T}_m & \hat{t}_{m+1} \\ 0 & \beta_{m+1} \end{bmatrix}, \quad \hat{t}_{m+1} = (0, \dots, 0, \beta_m, \alpha_{m+1})^T.$$

Hence,  $R_{m+1}$  has the following block structure:

$$R_{m+1} = \begin{bmatrix} R_m & r_{m+1} \\ 0 & \gamma_{m+1} \end{bmatrix}, \quad r_{m+1} = (0, \dots, 0, \varepsilon_{m-1}, \delta_m)^T. \tag{43}$$

As described above,  $Q_{m+1} = G_{m+1, m+2} G_{m, m+1} G_{m-1, m} \dots G_{2,3} G_{1,2}$ ,  $G_{m-1, m}$  is the  $(m-1)$ th plane rotation in order to determine  $\varepsilon_{m-1}$ ,  $G_{m, m+1}$  is the  $m$ th plane rotation in order to determine  $\delta_m$ , and  $G_{m+1, m+2}$  is the  $(m+1)$ th plane rotation in order to determine  $\gamma_{m+1}$  and eliminate the subdiagonal element  $\beta_{m+1}$ . These give the simple recurrence relation

$$\begin{bmatrix} c_{m-1} & s_{m-1} \\ -s_{m-1} & c_{m-1} \end{bmatrix} \begin{bmatrix} 0 \\ \beta_m \end{bmatrix} = \begin{bmatrix} \varepsilon_{m-1} \\ \hat{\beta}_m \end{bmatrix}, \quad \begin{bmatrix} c_m & s_m \\ -s_m & c_m \end{bmatrix} \begin{bmatrix} \hat{\beta}_m \\ \alpha_{m+1} \end{bmatrix} = \begin{bmatrix} \delta_m \\ \hat{\alpha}_{m+1} \end{bmatrix},$$

and

$$\begin{bmatrix} c_{m+1} & s_{m+1} \\ -s_{m+1} & c_{m+1} \end{bmatrix} \begin{bmatrix} \hat{\alpha}_{m+1} \\ \beta_{m+1} \end{bmatrix} = \begin{bmatrix} \gamma_{m+1} \\ 0 \end{bmatrix},$$

which implies that

$$\varepsilon_{m-1} = s_{m-1} \beta_m, \quad \hat{\beta}_m = c_{m-1} \beta_m, \quad \delta_m = c_m \hat{\beta}_m + s_m \alpha_{m+1}, \quad \hat{\alpha}_{m+1} = -s_m \hat{\beta}_m + c_m \alpha_{m+1}. \tag{44}$$

At the same time, from (38), by some calculations, one can conclude that  $\mathbf{z}_{m+1} = (\mathbf{z}_m^T, \boldsymbol{\zeta}_{m+1}^T)^T$  with

$$\boldsymbol{\zeta}_{m+1} = (-1)^m \beta s_1 s_2 \cdots s_m c_{m+1} = \rho_m c_{m+1} \quad (45)$$

and

$$\rho_{m+1} = (-1)^{m+1} \beta s_1 s_2 \cdots s_m s_{m+1} = -\rho_m s_{m+1}. \quad (46)$$

Moreover, according to (42) and (45), we have the update formula

$$\begin{aligned} \mathcal{X}_{m+1} &= \mathbb{F}_{m+1} \bar{\mathbf{x}}_{(M+N+1)} \mathbf{z}_{m+1} = [\mathbb{F}_m, \mathbf{F}_{m+1}] \bar{\mathbf{x}}_{(M+N+1)} \begin{bmatrix} \mathbf{z}_m \\ \boldsymbol{\zeta}_{m+1} \end{bmatrix} \\ &= \mathbb{F}_m \bar{\mathbf{x}}_{(M+N+1)} \mathbf{z}_m + \boldsymbol{\zeta}_{m+1} \mathbf{F}_{m+1} = \mathcal{X}_m + \boldsymbol{\zeta}_{m+1} \mathbf{F}_{m+1}. \end{aligned} \quad (47)$$

The main steps of the MINIRES-BTF method are summarized as Algorithm 4.

---

**Algorithm 4** MINIRES-BTF method

---

Given symmetric 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  $\text{Iter}_{\max}$ , the tolerance error  $\varepsilon > 0$ .
  - 2: Compute  $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_{\mathcal{N}} \mathcal{X}_0$ ;
  - 3:  $\beta = \|\mathcal{R}_0\|$ ,  $\mathcal{V}_1 = \mathcal{R}_0 / \beta$ ;
  - 4:  $\alpha_1 = \langle \mathcal{V}_1, \mathcal{A} *_{\mathcal{N}} \mathcal{V}_1 \rangle$ ,  $\mathcal{W} = \mathcal{A} *_{\mathcal{N}} \mathcal{V}_1 - \alpha_1 \mathcal{V}_1$ ,  $\beta_1 = \|\mathcal{W}\|$ ;
  - 5: **if**  $\beta_1 = 0$
  - 6:    $\mathcal{X}_1 = \mathcal{X}_0 + \mathcal{R}_0 / \alpha_1$ ;
  - 7: **else**
  - 8:    $\mathcal{V}_2 = \mathcal{W} / \beta_1$ ;
  - 9: **end**
  - 10:  $c_0 = 1$ ,  $s_0 = 0$ ,  $\mathcal{F}_0 = \mathcal{O} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ ;
  - 11: Determine  $c_1$  and  $s_1$  such that  $\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ 0 \end{bmatrix}$ ;
  - 12:  $\mathcal{F}_1 = \mathcal{V}_1 / \gamma_1$ ,  $\rho_1 = -\beta s_1$ ,  $\tau_1 = \beta c_1$ ;
  - 13:  $\mathcal{X}_1 = \mathcal{X}_0 + \tau_1 \mathcal{F}_1$ ,  $m = 1$ ;
  - 14: **while** ( $|\rho_m| / \beta > \varepsilon$ )
  - 15:    $\alpha_{m+1} = \langle \mathcal{V}_{m+1}, \mathcal{A} *_{\mathcal{N}} \mathcal{V}_{m+1} \rangle$ ;
  - 16:    $\mathcal{W} = \mathcal{A} *_{\mathcal{N}} \mathcal{V}_{m+1} - \alpha_{m+1} \mathcal{V}_{m+1} - \beta_m \mathcal{V}_m$ ;
  - 17:    $\beta_{m+1} = \|\mathcal{W}\|$ ;
  - 18:   **if**  $\beta_{m+1} \neq 0$ ;
  - 19:      $\mathcal{V}_{m+2} = \mathcal{W} / \beta_{m+1}$ ;
  - 20:   **end**
  - 21:    $\varepsilon_{m-1} = s_{m-1} \beta_m$ ,  $\hat{\beta}_m = c_{m-1} \beta_m$ ;
  - 22:    $\delta_m = c_m \hat{\beta}_m + s_m \alpha_{m+1}$ ,  $\hat{\alpha}_{m+1} = -s_m \hat{\beta}_m + c_m \alpha_{m+1}$ ;
  - 23:   Determine  $c_{m+1}$  and  $s_{m+1}$  such that  $\begin{bmatrix} c_{m+1} & s_{m+1} \\ -s_{m+1} & c_{m+1} \end{bmatrix} \begin{bmatrix} \hat{\alpha}_{m+1} \\ \beta_{m+1} \end{bmatrix} = \begin{bmatrix} \gamma_{m+1} \\ 0 \end{bmatrix}$ ;
  - 24:    $\boldsymbol{\zeta}_{m+1} = \rho_m c_{m+1}$ ,  $\rho_{m+1} = -\rho_m s_{m+1}$ ;
  - 25:    $\mathcal{F}_{m+1} = (\mathcal{V}_{m+1} - \varepsilon_{m-1} \mathcal{F}_{m-1} - \delta_m \mathcal{F}_m) / \gamma_{m+1}$ ;
  - 26:    $\mathcal{X}_{m+1} = \mathcal{X}_m + \boldsymbol{\zeta}_{m+1} \mathcal{F}_{m+1}$ ;
  - 27:    $m = m + 1$ ;
  - 28: **end**
- 

### 4.3 | SYMMLQ-BTF method: the tensor form of the SYMMLQ method

In this subsection, we will derive the tensor form of the SYMMLQ method developed by Paige et al.<sup>45</sup> to solve the tensor Equation (6) with the coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_1 \times \cdots \times I_N}$  being symmetric.

Given symmetric coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$  and the right-hand side tensor  $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_M}$ . Let  $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_M}$  be a given initial tensor and  $\mathcal{R}_0$  be the corresponding residual. The  $m$ th approximate solution in the SYMMLQ-BTF method is derived such that

$$\mathcal{X}_m \in \mathcal{X}_0 + \mathcal{K}_m(\mathcal{A}, \mathcal{R}_0) \quad (48)$$

and

$$\mathcal{R}_m \perp \mathcal{K}_m(\mathcal{A}, \mathcal{R}_0), \quad (49)$$

where  $\mathcal{R}_m = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_m$ . By Proposition 3, the tensors  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$  generated by Algorithm 2 form an orthonormal basis for  $\mathcal{K}_m(\mathcal{A}, \mathcal{R}_0)$ . Hence, there exists a vector  $y_m \in \mathbb{R}^m$  such that

$$\mathcal{X}_m = \mathcal{X}_0 + \mathbb{V}_m \bar{\mathbf{x}}_{(M+N+1)} y_m, \quad (50)$$

where  $\mathbb{V}_m = [\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m]$ . Because  $\mathcal{R}_0 = \beta \mathcal{V}_1$  ( $\beta = \|\mathcal{R}_0\|$ ), by Algorithm 2, Proposition 4, Lemma 3, and Lemma 4, we know that the residual at the  $m$ th iterative step is

$$\begin{aligned} \mathcal{R}_m &= \mathcal{C} - \mathcal{A} *_N \mathcal{X}_m = \mathcal{C} - \mathcal{A} *_N (\mathcal{X}_0 + \mathbb{V}_m \bar{\mathbf{x}}_{(M+N+1)} y_m) \\ &= \mathcal{R}_0 - (\mathcal{A} *_N \mathbb{V}_m) \bar{\mathbf{x}}_{(M+N+1)} y_m \\ &= \beta \mathcal{V}_1 - (\mathbb{V}_m \mathbf{x}_{(M+N+1)} T_m^T + \beta_m \mathcal{Z} \mathbf{x}_{(M+N+1)} E_m) \bar{\mathbf{x}}_{(M+N+1)} y_m \\ &= \beta \mathbb{V}_m \bar{\mathbf{x}}_{(M+N+1)} e_1^{(m)} - \mathbb{V}_m \bar{\mathbf{x}}_{(M+N+1)} (T_m y_m) - \beta_m y_m^{(m)} \mathcal{V}_{m+1} \\ &= \mathbb{V}_m \bar{\mathbf{x}}_{(M+N+1)} (\beta e_1^{(m)} - T_m y_m) - \beta_m y_m^{(m)} \mathcal{V}_{m+1}, \end{aligned} \quad (51)$$

where  $e_1^{(m)}$  is the first column of the identity matrix of order  $m$ , and  $y_m^{(m)}$  denotes the last component of  $y_m$ . Denote  $t_m := \beta e_1^{(m)} - T_m y_m$  and  $t_m := (t_m^{(1)}, t_m^{(2)}, \dots, t_m^{(m)})^T$ . Then, we can relax (51) as

$$\mathcal{R}_m = t_m^{(1)} \mathcal{V}_1 + t_m^{(2)} \mathcal{V}_2 + \dots + t_m^{(m)} \mathcal{V}_m - \beta_m y_m^{(m)} \mathcal{V}_{m+1}. \quad (52)$$

By the orthogonal condition (49), we have

$$\langle \mathcal{R}_m, \mathcal{V}_1 \rangle = \langle \mathcal{R}_m, \mathcal{V}_2 \rangle = \dots = \langle \mathcal{R}_m, \mathcal{V}_m \rangle = 0.$$

That is,

$$\begin{cases} 0 = \langle \mathcal{R}_m, \mathcal{V}_1 \rangle = t_m^{(1)} \langle \mathcal{V}_1, \mathcal{V}_1 \rangle + t_m^{(2)} \langle \mathcal{V}_2, \mathcal{V}_1 \rangle + \dots + t_m^{(m)} \langle \mathcal{V}_m, \mathcal{V}_1 \rangle - \beta_m y_m^{(m)} \langle \mathcal{V}_{m+1}, \mathcal{V}_1 \rangle, \\ 0 = \langle \mathcal{R}_m, \mathcal{V}_2 \rangle = t_m^{(1)} \langle \mathcal{V}_1, \mathcal{V}_2 \rangle + t_m^{(2)} \langle \mathcal{V}_2, \mathcal{V}_2 \rangle + \dots + t_m^{(m)} \langle \mathcal{V}_m, \mathcal{V}_2 \rangle - \beta_m y_m^{(m)} \langle \mathcal{V}_{m+1}, \mathcal{V}_2 \rangle, \\ \dots \dots \\ 0 = \langle \mathcal{R}_m, \mathcal{V}_m \rangle = t_m^{(1)} \langle \mathcal{V}_1, \mathcal{V}_m \rangle + t_m^{(2)} \langle \mathcal{V}_2, \mathcal{V}_m \rangle + \dots + t_m^{(m)} \langle \mathcal{V}_m, \mathcal{V}_m \rangle - \beta_m y_m^{(m)} \langle \mathcal{V}_{m+1}, \mathcal{V}_m \rangle. \end{cases}$$

Because  $\{\mathcal{V}_j\}_{j=1}^{m+1}$  is an orthonormal tensor sequence, we get  $t_m^{(1)} = t_m^{(2)} = \dots = t_m^{(m)} = 0$ . Hence,

$$\beta e_1^{(m)} - T_m y_m = 0. \quad (53)$$

Substituting (53) into (51) yields

$$\mathcal{R}_m = -\beta_m y_m^{(m)} \mathcal{V}_{m+1}. \quad (54)$$

The efficient solution of Equation (53) is the heart of the SYMMLQ-BTF method, which is similar to the SYMMLQ method.<sup>45</sup> However, for completeness, we describe the detailed process.

As in the SYMMLQ method, we form the LQ decomposition

$$T_m = \hat{L}_m Q_m, \quad Q_m^T Q_m = I_m, \quad (55)$$

where  $\hat{L}_m$  is a lower triangular matrix. Because  $T_m$  is a symmetric tridiagonal matrix, the factorization (55) can be obtained by a series of Givens matrices  $G_{k,k+1}$  each of which differs from the unit matrix only in the elements  $g_{kk} = -g_{k+1,k+1} = c_k$ ,  $g_{k,k+1} = g_{k+1,k} = s_k$ , that is,

$$G_{k,k+1} = \text{diag} \left( I^{(k-1)}, \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix}, I^{(m-k)} \right) \in \mathbb{R}^{(m+1) \times (m+1)}, c_k^2 + s_k^2 = 1, k = 1, 2, \dots, m.$$

Hence,

$$T_m G_{1,2} G_{2,3} \cdots G_{m-1,m} \triangleq T_m Q_m^T = \hat{L}_m = \begin{bmatrix} v_1 & & & & \\ \delta_2 & v_2 & & & \\ \eta_3 & \delta_3 & v_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_m & \delta_m & \hat{v}_m \end{bmatrix} \quad (56)$$

with  $Q_m = G_{m-1,m} \cdots G_{2,3} G_{1,2}$ . Notice that

$$\begin{aligned} \hat{L}_{m+1} &= T_{m+1} Q_{m+1}^T = T_{m+1} \left[ \begin{array}{c|c} Q_m^T & \\ \hline & 1 \end{array} \right] G_{m,m+1} \\ &= \left[ \begin{array}{ccc|cc} v_1 & & & & \\ \delta_2 & v_2 & & & \\ \eta_3 & \delta_3 & v_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_m & \delta_m & \hat{v}_m \\ & & & \eta_{m+1} & \hat{\delta}_{m+1} \\ & & & & \alpha_{m+1} \end{array} \right] \left[ \begin{array}{c|cc} I^{(m-1)} & & \\ \hline & c_m & s_m \\ & s_m & -c_m \end{array} \right], \end{aligned}$$

where  $c_m$  and  $s_m$  satisfy

$$c_m \hat{v}_m + s_m \beta_m = v_m, \quad s_m \hat{v}_m - c_m \beta_m = 0.$$

Denote  $\hat{v}_1 = \alpha_1$ ,  $\hat{\delta}_2 = \beta_1$ . It is easy to see that

$$v_m = \sqrt{\hat{v}_m^2 + \beta_m^2}, \quad c_m = \frac{\hat{v}_m}{v_m}, \quad s_m = \frac{\beta_m}{v_m}, \quad (57)$$

$$\delta_{m+1} = c_m \hat{\delta}_{m+1} + s_m \alpha_{m+1}, \quad \hat{v}_{m+1} = s_m \hat{\delta}_{m+1} - c_m \alpha_{m+1}, \quad \hat{\delta}_{m+2} = -c_m \beta_{m+1}, \quad \eta_{m+2} = s_m \beta_{m+1}. \quad (58)$$

At this moment, we obtain the decomposition  $T_{m+1} = \hat{L}_{m+1} Q_{m+1}$  with  $Q_{m+1}^T Q_{m+1} = I^{(m+1)}$ . Of course, if we substitute this decomposition into Equation (53) to derive the solution  $y_{m+1}$ , then by (50),  $\mathcal{X}_{m+1} = \mathcal{X}_0 + \mathbb{V}_{m+1} \bar{\mathcal{X}}_{(M+N+1)} y_{m+1}$ . However, this approach is not practical, as it needs to store  $\mathbb{V}_{m+1}$  and increase the computation load. Similar to the idea in the MINIRES method, we can solve  $\mathcal{X}_{m+1}$  directly and bypassing  $y_{m+1}$ . As the special form of  $Q_{m+1}$ , we denote  $\hat{\mathbb{G}}_{m+1} := [\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_m, \hat{\mathcal{G}}_{m+1}]$  via  $\hat{\mathbb{G}}_{m+1} := \mathbb{V}_{m+1} \times_{(M+1)} Q_{m+1}^{-T}$ , where  $\mathcal{G}_k = (\hat{\mathbb{G}}_{m+1})_{:\dots:k}$  for  $k = 1, 2, \dots, m$  and  $\hat{\mathcal{G}}_{m+1} = (\hat{\mathbb{G}}_{m+1})_{:\dots:m+1}$  are the frontal slices of  $\hat{\mathbb{G}}_{m+1}$ . Then, substituting (55) into (53) and (50) yields

$$\begin{aligned} \hat{L}_{m+1} (Q_{m+1} y_{m+1}) &= \beta e_1^{(m+1)}, \\ \mathcal{X}_{m+1} &= \mathcal{X}_0 + (\mathbb{V}_{m+1} \times_{(M+N+1)} Q_{m+1}^{-T}) \bar{\mathcal{X}}_{(M+N+1)} (Q_{m+1} y_{m+1}) = \mathcal{X}_0 + \hat{\mathbb{G}}_{m+1} \bar{\mathcal{X}}_{(M+N+1)} (Q_{m+1} y_{m+1}). \end{aligned}$$

Let

$$\hat{\mathbf{z}}_{m+1} := Q_{m+1} \mathbf{y}_{m+1} = [\zeta_1, \zeta_2, \dots, \zeta_m, \hat{\zeta}_{m+1}]^T. \quad (59)$$

Then,

$$\hat{L}_{m+1} \hat{\mathbf{z}}_{m+1} = \beta e_1^{(m+1)}, \quad \mathcal{X}_{m+1} = \mathcal{X}_0 + \hat{\mathbb{G}}_{m+1} \bar{\mathbf{x}}_{(M+N+1)} \hat{\mathbf{z}}_{m+1}. \quad (60)$$

Unfortunately, the above formula cannot be computed recursively. To do this, we introduce tensor  $\mathbb{G}_m$ , vector  $\mathbf{z}_m$ , and matrix  $L_m$  as follows:

$$\mathbb{G}_m := [\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_m], \quad \mathbf{z}_m := [\zeta_1, \zeta_2, \dots, \zeta_m]^T, \quad L_m := \begin{bmatrix} v_1 & & & & & \\ \delta_2 & v_2 & & & & \\ \eta_3 & \delta_3 & v_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \eta_m & \delta_m & v_m & \end{bmatrix},$$

where

$$L_m \mathbf{z}_m = \beta e_1^{(m)}. \quad (61)$$

Hence, by (57), one has

$$\zeta_m = \frac{\hat{v}_m}{v_m} \hat{\zeta}_m = c_m \hat{\zeta}_m. \quad (62)$$

Meanwhile, by (61), it is easy to see that

$$\begin{cases} v_1 \zeta_1 = \beta \Rightarrow \zeta_1 = \frac{\beta}{v_1}, \\ \delta_2 \zeta_1 + v_2 \zeta_2 = 0 \Rightarrow \zeta_2 = -\frac{\delta_2 \zeta_1}{v_2}, \\ \eta_k \zeta_{k-2} + \delta_k \zeta_{k-1} + v_k \zeta_k = 0 \Rightarrow \zeta_k = -\frac{\eta_k \zeta_{k-2} + \delta_k \zeta_{k-1}}{v_k}, \quad 3 \leq k \leq m. \end{cases} \quad (63)$$

Moreover, rather than updating  $\mathcal{X}_m$  at each step, we update  $\hat{\mathcal{X}}_m$

$$\hat{\mathcal{X}}_m = \mathcal{X}_0 + \mathbb{G}_m \bar{\mathbf{x}}_{(M+N+1)} \mathbf{z}_m = \hat{\mathcal{X}}_{m-1} + \zeta_m \mathcal{G}_m, \quad (64)$$

which, together with (59) and (60), yields

$$\mathcal{X}_{m+1} = \mathcal{X}_0 + \hat{\mathbb{G}}_{m+1} \bar{\mathbf{x}}_{(M+N+1)} \hat{\mathbf{z}}_{m+1} = \hat{\mathcal{X}}_m + \hat{\zeta}_{m+1} \hat{\mathcal{G}}_{m+1}. \quad (65)$$

Finally, we give the recurrence formulas of  $\mathcal{G}_m$  and  $\hat{\mathcal{G}}_{m+1}$ . Because  $\hat{\mathbb{G}}_{m+1} = [\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_m, \hat{\mathcal{G}}_{m+1}]$ ,  $\hat{\mathbb{G}}_{m+1} = \mathbb{V}_{m+1} \bar{\mathbf{x}}_{(M+N+1)} Q_{m+1}^{-T}$  and  $Q_{m+1} = G_{m,m+1} G_{m-1,m} \cdots G_{2,3} G_{1,2}$ , by the special form of  $G_{m,m+1}$ , we immediately have

$$[\hat{\mathcal{G}}_m, \mathcal{V}_{m+1}] \bar{\mathbf{x}}_{(M+N+1)} \begin{bmatrix} c_m & s_m \\ s_m & -c_m \end{bmatrix} = [\mathcal{G}_m, \hat{\mathcal{G}}_{m+1}], \quad \hat{\mathcal{G}}_1 = \mathcal{V}_1.$$

That is,

$$\mathcal{G}_m = c_m \hat{\mathcal{G}}_m + s_m \mathcal{V}_{m+1}, \quad \hat{\mathcal{G}}_{m+1} = s_m \hat{\mathcal{G}}_m - c_m \mathcal{V}_{m+1}. \quad (66)$$

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

---

**Algorithm 5** SYMMLQ-BTF method
 

---

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

- 1: **Input:** the initial tensor  $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N}$ , the maximum number of iterations  $\text{Iter}_{\max}$ , the tolerance error  $\varepsilon > 0$ .
  - 2: Compute  $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_{\mathcal{N}} \mathcal{X}_0$ ;
  - 3:  $\beta = \|\mathcal{R}_0\|$ ,  $\mathcal{V}_1 = \mathcal{R}_0 / \beta$ ;
  - 4:  $\alpha_1 = \langle \mathcal{V}_1, \mathcal{A} *_{\mathcal{N}} \mathcal{V}_1 \rangle$ ,  $\mathcal{W} = \mathcal{A} *_{\mathcal{N}} \mathcal{V}_1 - \alpha_1 \mathcal{V}_1$ ,  $\beta_1 = \|\mathcal{W}\|$ ;
  - 5:  $\mathcal{V}_2 = \mathcal{W} / \beta_1$ ,  $\hat{\mathcal{G}}_1 = \mathcal{V}_1$ ;
  - 6:  $\hat{v}_1 = \alpha_1$ ,  $\hat{\delta}_2 = \beta_1$ ,  $\zeta_0 = 0$ ,  $\eta_2 = 0$ ;
  - 7:  $v_1 = \sqrt{\alpha_1^2 + \beta_1^2}$ ,  $c_1 = \hat{v}_1 / v_1$ ,  $s_1 = \beta_1 / v_1$ ;
  - 8:  $\mathcal{G}_1 = c_1 \hat{\mathcal{G}}_1 + s_1 \mathcal{V}_2$ ,  $\hat{\mathcal{G}}_2 = s_1 \hat{\mathcal{G}}_1 - c_1 \mathcal{V}_2$ ;
  - 9:  $\zeta_1 = \frac{\beta}{v_1}$ ,  $\hat{\mathcal{X}}_1 = \mathcal{X}_0 + \zeta_1 \mathcal{G}_1$ ;
  - 10:  $m = 1$ ;
  - 11: **while** ( $\|\mathcal{R}_m\| / \beta > \varepsilon$ )
  - 12:    $\alpha_{m+1} = \langle \mathcal{V}_{m+1}, \mathcal{A} *_{\mathcal{N}} \mathcal{V}_{m+1} \rangle$ ;
  - 13:    $\mathcal{W} = \mathcal{A} *_{\mathcal{N}} \mathcal{V}_{m+1} - \alpha_{m+1} \mathcal{V}_{m+1} - \beta_m \mathcal{V}_m$ ;
  - 14:    $\beta_{m+1} = \|\mathcal{W}\|$ ;
  - 15:   **if**  $\beta_{m+1} \neq 0$ ;
  - 16:      $\mathcal{V}_{m+2} = \mathcal{W} / \beta_{m+1}$ ;
  - 17:   **end**
  - 18:    $\hat{v}_{m+1} = s_m \hat{\delta}_{m+1} - c_m \alpha_{m+1}$ ,  $v_{m+1} = \sqrt{\hat{v}_{m+1}^2 + \beta_{m+1}^2}$ ;
  - 19:    $c_{m+1} = \frac{\hat{v}_{m+1}}{v_{m+1}}$ ,  $s_{m+1} = \frac{\beta_{m+1}}{v_{m+1}}$ ;
  - 20:    $\delta_{m+1} = c_m \hat{\delta}_{m+1} + s_m \alpha_{m+1}$ ;
  - 21:    $\hat{\delta}_{m+2} = -c_m \beta_{m+1}$ ,  $\eta_{m+2} = s_m \beta_{m+1}$ ;
  - 22:    $\zeta_{m+1} = \rho_m c_{m+1}$ ,  $\rho_{m+1} = -\rho_m s_{m+1}$ ;
  - 23:    $\zeta_{m+1} = -\frac{\eta_{m+1} \zeta_{m-1} + \delta_{m+1} \zeta_m}{v_{m+1}}$ ,  $\hat{\zeta}_{m+1} = \zeta_{m+1} / c_{m+1}$ ;
  - 24:    $\mathcal{G}_{m+1} = c_{m+1} \hat{\mathcal{G}}_{m+1} + s_{m+1} \mathcal{V}_{m+2}$ ,  $\hat{\mathcal{G}}_{m+2} = s_{m+1} \hat{\mathcal{G}}_{m+1} - c_{m+1} \mathcal{V}_{m+2}$ ;
  - 25:    $\mathcal{X}_{m+1} = \hat{\mathcal{X}}_m + \zeta_{m+1} \hat{\mathcal{G}}_{m+1}$ ;
  - 26:    $\hat{\mathcal{X}}_{m+1} = \hat{\mathcal{X}}_m + \zeta_{m+1} \mathcal{G}_{m+1}$ ;
  - 27:    $\mathcal{R}_{m+1} = \mathcal{C} - \mathcal{A} *_{\mathcal{N}} \mathcal{X}_{m+1}$ ;
  - 28:    $m = m + 1$ ;
  - 29: **end**
- 

## 5 | NUMERICAL EXPERIMENTS

In this section, we report some numerical results to support Algorithms 3–5. All of the tests were run on the Intel (R) Core (TM), where the CPU is 2.40 GHz and the memory is 8.0 GB. The programming language was MATLAB R2015a. We comment here that the tensor toolbox<sup>46</sup> is utilized for solving the succeeding discussed problems.

### 5.1 | Comparison with matrix-based approaches

In this subsection, we will compare the calculation time of our proposed tensor forms of the algorithms with the time for the corresponding matrix forms. First, we present the corresponding matrix form approaches. To this end, we outline the following definition and proposition.

**Definition 4** (See the work of Brazell et al.<sup>18</sup>). Define the transformation  $\Phi_{IJ} : \mathbb{R}^{I_1 \times \dots \times I_N \times I_1 \times \dots \times I_N} \rightarrow \mathbb{R}^{I \times J}$  with  $\Phi_{IJ}(\mathcal{A}) = A$  defined component-wise as

$$(\mathcal{A})_{i_1 \dots i_N j_1 \dots j_N} \rightarrow (A)_{st},$$

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

**Proposition 5** (See the work of Wang et al.<sup>28</sup>). Let  $\Phi_{IJ}$  be defined as Definition 4. Then,

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

where  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$ ,  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ , and  $C \in \mathbb{R}^{I_1 \times \dots \times I_N \times K_1 \times \dots \times K_M}$ .

Now, applying the well-known GMRES method to solve the equivalent matrix equation  $\Phi_{IJ}(\mathcal{A})\Phi_{IJ}(\mathcal{X}) = \Phi_{IJ}(C)$  of the tensor Equation (6), we have the matrix form of the global GMRES method (denoted by GI-GMRES-BMF).

Similarly, we can obtain the matrix form of the MINIRES method (denoted by MINIRES-BMF) and the matrix form of the SYMMLQ method (denoted by SYMMLQ-BMF) by applying the MINIRES method and the SYMMLQ method to solve the equivalent matrix equation  $\Phi_{IJ}(\mathcal{A})\Phi_{IJ}(\mathcal{X}) = \Phi_{IJ}(C)$ , respectively. We do not want to list them here.

Now, we give the following example to illustrate the advantage of avoiding the matricization for the case of full format tensors.

**Example 1.** Assume that  $N = 2$ ,  $M = 2$ ,  $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times J_1 \times J_2}$ ,  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times J_1 \times J_2}$ , and  $C \in \mathbb{R}^{I_1 \times I_2 \times J_1 \times J_2}$ . In particular, 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  $C = \mathcal{A} *_N \tilde{\mathcal{X}}$ , we generate the right-hand side tensor  $C$  of Equation(6).

---

#### Algorithm 6 GI-GMRES-BMF method

---

Given coefficient matrix  $\Phi_{IJ}(\mathcal{A}) \in \mathbb{R}^{I \times J}$  and the right-hand side matrix  $\Phi_{IJ}(C) \in \mathbb{R}^{I \times J}$ .

- 1: **Input:** the maximum number of iterations  $\text{Iter}_{\max}$ , the tolerance error  $\varepsilon > 0$ , the maximal dimension of the Krylov subspace  $m$ .
  - 2: Compute  $R_0 = \Phi_{IJ}(C) - \Phi_{IJ}(\mathcal{A})\Phi_{IJ}(\mathcal{X}_0)$ ,  $r = \|R_0\|$ ,
  - 3: **for**  $k = 1, \dots, \text{Iter}_{\max}$  **do**
  - 4:    $\beta = \|R_0\|$ ,  $V_1 = R_0/\beta$ ;
  - 5:   Define the  $(m+1) \times m$  matrix  $H_m = (h_{ij})$ , set  $H_m = O$ ;
  - 6:   **for**  $j = 1, 2, \dots, m$  **do**
  - 7:      $W_j = \Phi_{IJ}(\mathcal{A})V_j$ ;
  - 8:     **for**  $i = 1, 2, \dots, j$  **do**
  - 9:        $h_{ij} = \langle W_j, V_i \rangle$ ;
  - 10:        $W_j = W_j - h_{ij}V_i$
  - 11:     **end for**
  - 12:      $h_{j+1,j} = \|W_j\|$ , **if**  $h_{j+1,j} = 0$ , then **stop**;
  - 13:      $V_{j+1} = W_j/h_{j+1,j}$ ;
  - 14:   **end for**
  - 15:   Solve the problem  $y_m = \arg \min_{y \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - H_m y\|_2$ ;
  - 16:   Compute the approximate solution  $X_m = \Phi_{IJ}(\mathcal{X}_0) + \sum_{i=1}^m y_i V_i$ ;
  - 17:   Compute  $R_m = \Phi_{IJ}(C) - \Phi_{IJ}(\mathcal{A})X_m$ ;
  - 18:   **if**  $\|R_m\|/r < \varepsilon$  **then**
  - 19:     output the approximate solution  $X_m$ ;
  - 20:   **else**
  - 21:      $R_0 = R_m$ ;
  - 22:   **end if**
  - 23: **end for**
- 

Take the maximal dimension of Krylov subspace  $m = 10$  in the GI-GMRES-BTF method and the GI-GMRES-BMF method. We apply the GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods to compute the approximate solution  $\mathcal{X}_k$  with the initial tensor  $\mathcal{X}_1 = \mathcal{O} \in \mathbb{R}^{50 \times 50 \times 30 \times 20}$ . In addition, we also apply the GI-GMRES-BMF, MINIRES-BMF, and SYMMLQ-BMF methods to compute  $X_k$  with the initial matrix  $X_1 = \Phi_{IJ}(\mathcal{X}_1) = O \in \mathbb{R}^{1,500 \times 600}$ . We list the iterative steps (denoted by “Iter”), the elapsed CPU time in seconds (denoted by “Time”), and the relative residual (denoted by “Err(k)”),

**TABLE 1** Numerical results for Example 1

Method	Iter	Time	Err(k)
GI-GMRES-BTF	18	24.7661	9.8985e-07
GI-GMRES-BMF	18	78.5808	9.8985e-07
MINIRES-BTF	78	9.6736	9.5507e-07
MINIRES-BMF	76	23.6314	9.9258e-07
SYMMLQ-BTF	78	12.6918	9.9324e-07
SYMMLQ-BMF	76	20.2111	9.9258e-07

where  $\text{Err}(k)$  is defined as follows:

$$\text{Err}(k) = \frac{\|\mathcal{R}_k\|}{\|\mathcal{R}_0\|} \quad \text{or} \quad \text{Err}(k) = \frac{\|R_k\|}{\|R_0\|} \quad (67)$$

with

$$\mathcal{R}_k = C - \mathcal{A} *_N \mathcal{X}_k, R_k = \Phi_{II}(C) - \Phi_{II}(\mathcal{A})X_k.$$

The stopping criterion of the abovementioned algorithms is considered as  $\text{Err}(k) < 10^{-6}$ . The corresponding numerical results are listed in Table 1.

From Table 1, we find that the calculation time of the proposed tensor forms of the algorithms (i.e., GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods) is far less than the calculation time for the corresponding matrix forms (i.e., GI-GMRES-BMF, MINIRES-BMF, and SYMMLQ-BMF methods). This example confirms that solving multilinear systems with the tensor-based iterative methods can keep the computational cost low and the tensor-based iterative methods are better than the matrix-based iterative methods with respect to the elapse CPU time.

Although the corresponding steps in the tensor-based iterative methods and the matrix-based iterative methods seem to be algebraically equivalent, we point out that the advantage of the tensor-based iterative method may be due to accounting on the low-rank structure in the solution and the right-hand side, both represented as  $I_1 \times I_2 \times J_1 \times J_2$  data arrays.

We also randomly select  $I_1, I_2, I_3, J_1, J_2$ , and  $J_3$  in order to generate the coefficient tensor  $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_1 \times I_2 \times I_3}$  and the solution tensor  $\mathcal{X} = \text{tenrand}([I_1 \ I_2 \ I_3 \ J_1 \ J_2 \ J_3]) \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times J_1 \times J_2 \times J_3}$ . Then, by  $C = \mathcal{A} *_N \mathcal{X}$ , we generate the right-hand side tensor  $C$  of Equation (6). After several tests, we find that the calculation time of the tensor-based methods (i.e., GI-GMRES-BTF, MINIRES-BIF, and SYMMLQ-BTF methods) is less than those of the matrix-based methods (i.e., GI-GMRES-BMF, MINIRES-BMF, and SYMMLQ-BMF methods). This further indicates that the tensor implementation makes the algorithm faster.

## 5.2 | Tensor equations in the Poisson problem

Consider the three-dimensional (3D) Poisson problem

$$\begin{cases} -\nabla^2 v = f, & \text{in } \Omega = \{(x, y, z), 0 < x, y, z < 1\}, \\ v = 0, & \text{on } \partial\Omega, \end{cases} \quad (68)$$

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 compute an approximation of the unknown function  $v(x, y, z)$  in (68). Take the uniform mesh step size. That is, the step sizes, namely,  $\Delta x$  in the  $x$ -direction,  $\Delta y$  in the  $y$ -direction, and  $\Delta z$  in the  $z$ -direction, satisfy  $\Delta x = \Delta y = \Delta z = h = 1/(n+1)$ . By the standard central difference approximations, we obtain the difference formula

$$6v_{ijk} - v_{i-1jk} - v_{i+1jk} - v_{ij-1k} - v_{ij+1k} - v_{ijk-1} - v_{ijk+1} = h^3 f_{ijk}. \quad (69)$$

Hence, the higher order tensor representation of the 3D discretized Poisson problem (68) is<sup>18</sup>

$$\bar{\mathcal{A}} *_3 \mathcal{V} = \mathcal{F}, \quad (70)$$

Grid $n \times n \times n$	$10 \times 10 \times 10$ Iter/Time	$15 \times 15 \times 15$ Iter/Time	$20 \times 20 \times 20$ Iter/Time
GI-GMRES-BTF	5/0.1688	8/0.7215	13/5.1535
MINIRES-BTF	21/0.0652	33/0.2529	42/1.5959
SYMMLQ-BTF	21/0.0542	33/0.2688	42/1.5038
BiCG-BTF <sup>18</sup>	42/0.0824	101/0.7465	168/5.6629

**TABLE 2** The iteration number and the CPU time for the Poisson equation in three dimensions

where the Laplacian tensor  $\bar{\mathcal{A}} \in \mathbb{R}^{n \times n \times n \times n \times n \times n}$  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  $(\bar{\mathcal{A}})_{l,m,p}^{(2,4,6)}$  of  $\bar{\mathcal{A}}$  would follow a seven-point stencil, that is,

$$\begin{aligned}
 (\bar{\mathcal{A}})_{\alpha,\beta,\gamma}^{(2,4,6)} &= \frac{6}{h^3}, \\
 (\bar{\mathcal{A}})_{\alpha-1,\beta,\gamma}^{(2,4,6)} &= (\bar{\mathcal{A}})_{\alpha+1,\beta,\gamma}^{(2,4,6)} = -\frac{1}{h^3}, \\
 (\bar{\mathcal{A}})_{\alpha,\beta-1,\gamma}^{(2,4,6)} &= (\bar{\mathcal{A}})_{\alpha,\beta+1,\gamma}^{(2,4,6)} = -\frac{1}{h^3}, \\
 (\bar{\mathcal{A}})_{\alpha,\beta,\gamma-1}^{(2,4,6)} &= (\bar{\mathcal{A}})_{\alpha,\beta,\gamma+1}^{(2,4,6)} = -\frac{1}{h^3},
 \end{aligned}$$

where  $(\bar{\mathcal{A}})_{l,m,p}^{(2,4,6)} = \bar{\mathcal{A}}(:, l, :, m, :, p) \in \mathbb{R}^{n \times n \times n}$ .

We apply the GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods to solve the tensor Equation (70). We also compare our proposed methods with the BiCG-BTF method developed by Brazell et al.<sup>18</sup> The numerical results are reported in Table 2, where “Time” denotes the running time when an algorithm terminates, “Iter” denotes the iteration steps, and “Grid” denotes the grid size.

From Table 2, we find that it needs more calculation time and iterative steps to find the solution of the tensor Equation (70) as the increasing of grid size. As the increasing of grid size, the CPU time and iteration numbers of the GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods are fewer than those of the biconjugate gradient method (BiCG-BTF method) in the work of Brazell et al.<sup>18</sup> The numerical results illustrated in Table 2 also show the fast convergence rate and efficiency of the MINIRES-BTF and SYMMLQ-BTF methods because they need fewer flops than the BiCG-BTF method. This example shows that the GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods are quite effective to solve the Poisson problem in three dimensions.

From this example, we also find some advantages in computing in a tensor structured domain. First, without vectorization, the solution and the computational grid have a one-to-one correspondence so that sophisticated boundary conditions are easily integrated in the multilinear system. Second, the Laplacian tensor preserves the low bandwidth because the main nodal points sit on the tensor diagonal entries and the rest of the stencil points lie on the off-diagonal positions. Third, the Laplacian tensor  $\bar{\mathcal{A}}$  has a low bandwidth; hence, the tensor-based methods can reduce the number of operations and storage locations. The same things occur in the BiCG-BTF method and the Jacobi method of Brazell et al.<sup>18</sup>

### 5.3 | Comparison with other methods

We make the numerical comparison of the GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods with Algorithm 3.1 developed by Wang et al.<sup>28</sup>

**Example 2.** 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  $I_1 = I_2 = 20$ ,  $I_3 = 10$ ,  $J_1 = 20$ ,  $J_2 = 10$ , and  $J_3 = 5$ , and randomly construct the symmetric tensor  $\mathcal{A} \in \mathbb{R}^{20 \times 20 \times 10 \times 20 \times 20 \times 10}$ . The exact solution  $\tilde{\mathcal{X}}$  is a tensor with all elements equal to one, that is,  $\tilde{\mathcal{X}} = \text{tenones}([20 \ 20 \ 10 \ 20 \ 10 \ 5])$ . Then, by  $\mathcal{C} = \mathcal{A} *_N \tilde{\mathcal{X}}$ , we generate the right-hand side tensor  $\mathcal{C}$  of Equation (6).

Take the maximal dimension of Krylov subspace  $m = 10$  in GI-GMRES-BTF method. Take the initial tensor  $\mathcal{X}_0 = \mathcal{O} \in \mathbb{R}^{20 \times 20 \times 10 \times 20 \times 20 \times 10 \times 5}$ . In addition, the stopping criterion is considered as

$$\text{Err}(k) = \frac{\|\mathcal{R}_k\|}{\|\mathcal{R}_0\|} < 10^{-6}, \quad (71)$$

**TABLE 3** Numerical results for Example 2

Method	Iter	Time	Err(k)
GI-GMRES-BTF	6	68.5070	8.5180e-07
MINIRES-BTF	39	38.9021	9.5507e-07
SYMMLQ-BTF	40	40.1427	9.5507e-07
Algorithm 3.1 <sup>28</sup>	91	260.6761	9.9812e-07

**TABLE 4** Numerical results for Example 3

Method	Iter	Time	Err(k)
GMRES-BTF	9	58.7063	9.5573e-07
MINIRES-BTF	31	31.2020	9.8259e-07
SYMMLQ-BTF	33	33.5514	9.8259e-07
Algorithm 3.1 <sup>28</sup>	128	197.8398	9.4150e-07

where

$$\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_{\mathcal{N}} \mathcal{X}_k.$$

We demonstrate the efficiency of the abovementioned iterative methods from the aspects of iterative steps (denoted by “Iter”), elapsed CPU time in seconds (denoted by “Time”), and the norm of relative residual (denoted by “Err(k)”). The corresponding numerical results are listed in Table 3. Table 3 shows that the iteration number and the CPU time of the MINIRES-BTF, SYMMLQ-BTF, and GI-GMRES-BTF methods are less than those of Algorithm 3.1 in the work of Wang et al.<sup>28</sup> Table 3 also shows that the numerical results of the MINIRES-BTF method are almost the same as those of the SYMMLQ-BTF method. Although the GI-GMRES-BTF method takes the least amount of iterative steps, the CPU time is more than that of the MINIRES-BTF method and the SYMMLQ-BTF method. This example confirms that both the MINIRES-BTF method and the SYMMLQ-BTF method are efficient in solving tensor equation with symmetric coefficient tensor.

**Example 3.** Consider the tensor Equation (6) with  $\mathcal{A} \in \mathbb{R}^{60 \times 40 \times 60 \times 40}$  being symmetric tensor and  $\mathcal{C} \in \mathbb{R}^{60 \times 40 \times 50 \times 30}$ . The exact solution  $\tilde{\mathcal{X}}$  is a tensor with all elements equal to one, that is,  $\tilde{\mathcal{X}} = \text{tenones}([60 \ 40 \ 50 \ 30])$ . Then, by  $\mathcal{C} = \mathcal{A} *_{\mathcal{N}} \tilde{\mathcal{X}}$ , we generate the right-hand side tensor  $\mathcal{C}$  of Equation (6).

Take the initial tensor  $\mathcal{X}_0 = \mathcal{O} \in \mathbb{R}^{60 \times 40 \times 50 \times 30}$  for all these methods. The stopping criteria of all the abovementioned methods is  $\text{Err}(k) < 10^{-6}$ , where  $\text{Err}(k)$  is defined as in (71). The numerical results are reported in Table 4, where “Time” denotes the running time when an algorithm terminates, “Iter” denotes the iteration steps, and “Err(k)” denotes the norm of relative residual.

As seen in Table 4, we find that the iteration number and CPU time of the MINIRES-BTF and SYMMLQ-BTF methods are almost the same. Although the iteration number of the GI-GMRES-BTF method is less than that of other methods, the CPU time is more than that of the MINIRES-BTF and SYMMLQ-BTF methods. In a word, our proposed methods (i.e., GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods) outperform Algorithm 3.1 in the work of Wang et al.<sup>28</sup> in terms of the iteration number and the CPU time. This example confirms that the MINIRES-BTF method and the SYMMLQ-BTF method are effective for finding the solution of the tensor Equation (6) with symmetric coefficient tensor.

The test results of Examples 2 and 3 indicate that our proposed tensor-based methods (i.e., GI-GMRES-BTF, MINIRES-BTF, and SYMMLQ-BTF methods) are more efficient in the number of iterations than Algorithm 3.1 in the work of Wang et al.<sup>28</sup> Because no failure was detected for either of the methods in the tests, it is inconclusive which method would be more robust.

## 6 | CONCLUSIONS

Based on the GI-GMRES method, we have constructed and analyzed the GI-GMRES-BTF method (Algorithm 3) to solve a class of tensor equation. Meanwhile, the tensor forms of the MINIRES (Algorithm 4) and SYMMLQ (Algorithm 5) methods are derived to solve the tensor equation with symmetric coefficient tensor. Algorithms 3–5 solely use tensor computations, that is, no matricizations are involved. The applications of the proposed method have been compared with the tensor form of the classical conjugate gradient method (Algorithm 3.1 in the work of Wang et al.<sup>28</sup>). Our test examples have revealed that the proposed method can outperform Algorithm 3.1 in the work of Wang et al.<sup>28</sup> in terms of the required CPU time and iteration number for the convergence. We also compare the tensor-based iterative approaches and the

matrix-based iterative approaches, and we find that solving multilinear systems with the tensor-based iterative methods can keep the computational cost low, and the tensor-based iterative methods are better than the matrix-based iterative methods. Furthermore, we apply our proposed iterative methods to solve the tensor equation in the three-dimensional Poisson problem. We can conclude that solving the tensor equation with the tensor-based iterative methods has several advantages: (1) higher order tensor representation of PDEs preserve low bandwidth thereby keeping the computational cost and memory requirement low and (2) a one-to-one correspondence between the solution and the computational grid facilitates the integration of complicated boundary conditions, which has been illustrated by Brazell et al.<sup>18</sup>

## ACKNOWLEDGEMENTS

The authors would like to thank the area editor and the referees for their careful review and thoughtful critiques, which have improved the quality of this paper. This research is supported by National Key Research and Development Program of China (Grants 2018YFC1054200 and 2018YFC0603500).

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## ORCID

Baohua Huang  <https://orcid.org/0000-0002-4672-1191>

Changfeng Ma  <https://orcid.org/0000-0002-5936-789X>

## REFERENCES

1. Beik AFP, Ahmadi-Asl S. Residual norm steepest descent based iterative algorithms for Sylvester tensor equations. *J Math Model.* 2015;2:115–131.
2. Chen HB, Wang YJ. On computing minimal H-eigenvalue of sign-structured tensors. *Front Math China.* 2017;12:1289–1302.
3. Chen HB, Chen YN, Li GY, Qi L. A semidefinite program approach for computing the maximum eigenvalue of a class of structured tensors and its applications in hypergraphs and copositivity test. *Numer Linear Algebra Appl.* 2018;25:e2125.
4. Chen H, Qi L, Song Y. Column sufficient tensors and tensor complementarity problems. *Front Math China.* 2018;13:255–276.
5. Zhou G, Wang G, Qi L, Alqahtani M. A fast algorithm for the spectral radii of weakly reducible nonnegative tensors. *Numer Linear Algebra Appl.* 2017;25:e2134.
6. Wang X, Chen H, Wang Y. Solution structures of tensor complementarity problem. *Front Math China.* 2018;6:1–11.
7. Wang G, Zhou G, Caccetta L. Z-eigenvalue inclusion theorems for tensors. *Discrete Cont Dyn Syst B.* 2017;22:187–198.
8. Wang Y, Zhang K, Sun H. Criteria for strong H-tensors. *Front Math China.* 2016;11:577–592.
9. Zhang K, Wang Y. An H-tensor based iterative scheme for identifying the positive definiteness of multivariate homogeneous forms. *J Comput Appl Math.* 2016;305:1–10.
10. Wang Y, Qi L, Zhang X. A practical method for computing the largest M-eigenvalue of a fourth-order partially symmetric tensor. *Numer Linear Algebra Appl.* 2010;16:589–601.
11. Chang KC, Pearson K, Zhang T. Perron–Frobenius theorem for nonnegative tensors. *Commun Math Sci.* 2008;6:507–520.
12. Chen Z, Lu L. A projection method and Kronecker product preconditioner for solving Sylvester tensor equations. *Science China Math.* 2012;55:1281–1292.
13. Huang B, Ma C. Iterative criteria for identifying strong H-tensors. *J Comput Appl Math.* 2019;352:93–109.
14. Karimi S, Dehghan M. Global least squares method based on tensor form to solve linear systems in Kronecker format. *Trans Inst Meas Control.* 2018;40:2378–2386.
15. Qi L. Eigenvalues of a real supersymmetric tensor. *J Symbolic Comput.* 2005;40:1302–1324.
16. Shao J-Y. A general product of tensors with applications. *Linear Algebra Appl.* 2013;439:2350–2366.
17. Shi X, Wei Y, Ling S. Backward error and perturbation bounds for high order Sylvester tensor equation. *Linear Multilinear Algebra.* 2013;61:1436–1446.
18. Brazell M, Li N, Navasca C, Tamon C. Solving multilinear systems via tensor inversion. *SIAM J Matrix Anal Appl.* 2013;34:542–570.
19. Lai M, Krempel E, Rubin D. Introduction to continuum mechanics. Oxford, UK: Butterworth Heinemann; 2009.
20. Li BW, Tian S, Sun YS, Hu ZM. Schur-decomposition for in solving radiative discrete ordinates equations di method. *J Comput Phys.* 2010;229:1198–1212.
21. Li W, Ng MK. On the limiting probability distribution of a transition probability tensor. *Linear Multilinear Algebra.* 2014;62:362–385.
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. Li B-W, Sun Y-S, Zhang D-W. Chebyshev collocation spectral methods for coupled radiation and conduction in a concentric spherical participating medium. *ASME J Heat Transfer.* 2009;131:062701–062709.

24. Haussühl S. Physical properties of crystals. Weinheim, Germany: Wiley-VCH Verlag; 2007.
25. 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.
26. Lim LH. Tensors and hypermatrices. In: Handbook of linear algebra. Boca Raton, FL: CRC Press; 2013.
27. Behera R, Mishra D. Further results on generalized inverses of tensors via the Einstein product. *Linear Multilinear Algebra*. 2017;65:1662–1682.
28. Wang QW, Xu X. Iterative algorithms for solving some tensor equations. *Linear Multilinear Algebra*. 2018. <https://doi.org/10.1080/03081087.2018.1452889>
29. Huang B, Ma C. An iterative algorithm to solve the generalized Sylvester tensor equations. *Linear Multilinear Algebra*. 2018. <https://doi.org/10.1080/03081087.2018.1536732>
30. Bellman RE. Dynamic programming. Princeton, NJ: Princeton University Press; 1957.
31. Khoromskij BN. Tensor numerical methods for high-dimensional PDEs: basic theory and initial applications. *ESAIM: Proc Surv*. 2015;48:1–28.
32. Khoromskij BN. Tensors-structured numerical methods in scientific computing: survey on recent advances. *Chemom Intellingent Lab Syst*. 2012;110:1–19.
33. Khoromskaia V. Numerical solution of the Hartree-Fock equation by multilevel tensor-structured methods PhD [dissertation]. Berlin, Germany: Technische Universität Berlin; 2010. Available from: <http://opus.kobv.de/tuberlin/volltexte/2011/2948/>
34. Khoromskij BN. Tensor numerical methods in scientific computing. Berlin, Germany: Walter de Gruyter; 2018.
35. Griebel M, Hamaekers J. Sparse grids for the Schrödinger equation. *ESAIM: Math Model Numer Anal*. 2007;41:215–247.
36. Beylkin G, Mohlenkamp MJ. Algorithms for numerical analysis in high dimension. *SIAM J Sci Comput*. 2005;26:2133–2159.
37. Khoromskij BN, Khoromskaia V, Flad H-J. Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format. *SIAM J Sci Comput*. 2011;33:45–65.
38. Khoromskij BN. Tensor-structured preconditioners and approximate inverse of elliptic operators in  $\mathbb{R}^d$ . *Constructive Approximation*. 2009;30:599–620.
39. Hackbusch W, Khoromskij BN. Tensor-product approximation to operators and functions in high dimension. *J Complex*. 2007;23:697–714.
40. Hackbusch W, Khoromskij BN, Tyrtshnikov EE. Hierarchical Kronecker tensor-product approximations. *J Numer Math*. 2005;13:119–156.
41. De Lathauwer L. A survey of tensor methods. Paper presented at: 2009 IEEE International Symposium on Circuits and Systems; 2009; Taipei, Taiwan. Piscataway, NJ: IEEE; 2009.
42. Kolda TG, Bader BW. Tensor decompositions and applications. *SIAM Review*. 2009;3:455–500.
43. De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. *SIAM J Matrix Anal Appl*. 2000;21:1253–1278.
44. Beik FPA, Saberi Movahed F, 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.
45. Paige CC, Saunders MA. Solution of sparse indefinite systems of linear equations. *SIAM J Numer Anal*. 1975;12:617–629.
46. Bader BW, Kolda TG. MATLAB Tensor Toolbox Version 2.5. 2012. <http://www.sandia.gov/tgkolda/TensorToolbox/>

**How to cite this article:** Huang B, Xie Y, Ma C. Krylov subspace methods to solve a class of tensor equations via the Einstein product. *Numer Linear Algebra Appl*. 2019;26:e2254. <https://doi.org/10.1002/nla.2254>