Preconditioned TBiCOR and TCORS Algorithms for Solving the Sylvester Tensor Equation

Guang-Xin Huang[1]*, Qi-Xing Chen[1], Feng Yin[2]

1. *College of Mathematics and Physics, Chengdu University of Technology, P.R.China*

2. *College of Mathematics and statistics, Sichuan University of Science and Engineering, P.R.China*

Abstract. In this paper, the preconditioned TBiCOR and TCORS methods are presented for solving the Sylvester tensor equation. A tensor Lanczos $\mathcal{L}$-Biorthogonalization algorithm (TLB) is derived for solving the Sylvester tensor equation. Two improved TLB methods are presented. One is the biconjugate $\mathcal{L}$-orthogonal residual algorithm in tensor form (TBiCOR), which implements the LU decomposition for the triangular coefficient matrix derived by the TLB method. The other is the conjugate $\mathcal{L}$-orthogonal residual squared algorithm in tensor form (TCORS), which introduces a square operator to the residual of the TBiCOR algorithm. A preconditioner based on the nearest Kronecker product is used to accelerate the TBiCOR and TCORS algorithms, and we obtain the preconditioned TBiCOR algorithm (PTBiCOR) and preconditioned TCORS algorithm (PTCORS). The proposed algorithms are proved to be convergent within finite steps of iteration without roundoff errors. Several examples illustrate that the preconditioned TBiCOR and TCORS algorithms present excellent convergence.

Keywords. TLB, TBiCOR, TCORS, Sylvester tensor equation, Preconditioner

1 Introduction

This paper is concerned of the computation of the Sylvester tensor equation of the form

$$\mathcal{X} \times_1 \mathbf{A}_1 + \mathcal{X} \times_2 \mathbf{A}_2 + ... + \mathcal{X} \times_N \mathbf{A}_N = \mathcal{D}, \tag{1.1}$$

where matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times I_n} (n = 1, 2, ..., N)$ and the tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ are given, and the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ is unknown.

The Sylvester tensor equation (1.1) plays vital roles in many fields such as image processing [8], blind source separation [22] and the situation when we

*Emails: huangx@cdut.edu.cn (G.X. Huang), qixinggenius@163.com (Q.X. Chen), fyin@suse.edu.cn (F. Yin)

describes a chain of spin particles [2]. If $N = 2$, then (1.1) can be reduced to the Sylvester matrix equation

$$\mathbf{AX} + \mathbf{XB}^T = \mathbf{D}, \tag{1.2}$$

which has many applications in system and control theory [12–14]. When $N = 3$, (1.1) becomes

$$\mathcal{X} \times_1 \mathbf{A}_1 + \mathcal{X} \times_2 \mathbf{A}_2 + \mathcal{X} \times_3 \mathbf{A}_3 = \mathcal{D}, \tag{1.3}$$

which often arises from the finite element [20], finite difference [3] and spectral methods [21].

Many approaches are constructed to solve the Sylvester tensor equation (1.1) in recent years. Chen and Lu [9] proposed the GMRES method based on a tensor format for solving (1.1) and presented the gradient based iterative algorithms [10] for solving (1.3). Beik et al. [24] also presented some global iterative schemes based on Hessenberg process to solve the Sylvester tensor equation (1.3). Beik et al. [5] solved the Sylvester tensor equation (1.1) with severely ill-conditioned coefficient matrices and considered its application in color image restoration. Heyouni et al. [15] proposed a general framework by using tensor Krylov projection techniques to solve high order the Sylvester tensor equation (1.1). Beik et al. [1] proposed the Arnoldi process and full orthogonalization method in tensor form, and the conjugate gradient and nested conjugate gradient algorithms in tensor form to solve the Sylvester tensor equation (1.1). When $\mathcal{D}$ in (1.1) is a tensor with low rank, Bentbib et al. [6] proposed Arnoldi-based block and global methods. Kressner and Tobler [19] developed Krylov subspace methods based on extended Arnoldi process for solving the system of equation (1.1). The perturbation bounds and backward error are presented in [27] for solving (1.1). For more methods on other linear systems in tensor form we refer to [4, 16, 18, 23]. Using the nearest Kronecker product (NKP) in [28], Chen and Lu [9] presented an efficient preconditioner for solving Eq.(1.1) based on GMRES in tensor form. Very recently Zhang and Wang in [30] gave a preconditioned BiCG (PBiCG) and a preconditioned BiCR (PBiCR) based on the nearest Kronecker product (NKP) in [28].

Inspired by the Lanczos biorthogonalization (LB) algorithm in [26], BICOR and CORS methods in [11] for non-symmetric linear equation, in this paper, we present two improved Lanczos $\mathcal{L}$-orthogonal algorithms in tensor form for solving the Sylvester tensor equation (1.1). We further present preconditioned TBiCOR (PTLB) and preconditioned TCORS (TCORS) algorithms by using the NKP preconditioner in [9] for solving Eq (1.1). The preconditioned LB in tensor form (PTLB) is also considered.

The rest of this paper is organized as follows. Section 2 reviews some related symbols, concepts and lemmas that will be used in the contexture. Section 3 presents a tensor Lanczos $\mathcal{L}$-biorthogonalization algorithm (TLB) and two improved TLB methods are shown in section 4. The tensor biconjugate $\mathcal{L}$-orthogonal residual(TBiCOR) and tensor conjugate $\mathcal{L}$-orthogonal residual squared(TCORS) algorithms for solving the tensor equation (1.1) are presented in subsections 4.1 and 4.2, respectively. The convergence of the TBiCOR and

TCORS methods are proved. Section 5 presents the preconditioned TLB, TBiCOR and TCORS algorithms and the convergence of the preconditioned TBiCOR and TCORS algorithms. Section 6 presents several examples and some conclusions are drawn in section 7.

2 Preliminaries

The notations and definitions as follows are needed. For a positive integer N, an N-way or Nth-order tensor $\mathcal{X} = (x_{i_1 \dots i_N})$ is a multidimensional array with $I_1 I_2 ... I_N$ entries, where $1 \leq i_j \leq I_j, j = 1, ..., N$. $\mathbb{R}^{I_1 \times \dots \times I_N}$ denotes the set of the Nth-order $I_1 \times ... \times I_N$ dimension tensors over the real field $\mathbb{R}$, while $\mathbb{C}^{I_1 \times \dots \times I_N}$ defines the set of the Nth-order $I_1 \times ... \times I_N$ dimension tensors over the complex field $\mathbb{C}$.

Let $\mathcal{X} \times_n \mathbf{A}$ define the n-mode (matrix) product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$, i.e.,

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \dots i_{n-1} j i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \dots i_N} a_{j i_n}.$$

The n-mode (vector) product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and a vector $\mathbf{v} \in \mathbb{R}^{I_n}$ is denoted by $\mathcal{X} \overline{\times}_n \mathbf{v}$, i.e.,

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

The inner product of $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is defined by

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \dots i_N} y_{i_1 i_2 \dots i_N},$$

and the norm of $\mathcal{X}$ is denoted by

$$\|\mathcal{X}\| = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}.$$

Furthermore, it derives from [9] that

$$\langle \mathcal{X}, \mathcal{Y} \times_n \mathbf{A} \rangle = \langle \mathcal{X} \times_n \mathbf{A}^T, \mathcal{Y} \rangle. \tag{2.1}$$

We refer more notions and definitions in [17].

The following results from [1] will be used later.

Lemma 1. *Suppose* $\boldsymbol{A} \in \mathbb{R}^{J_n \times I_n}$, $\boldsymbol{y} \in \mathbb{R}^{J_n}$ *and* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, *we have*

$$\mathcal{X} \times_n \boldsymbol{A} \overline{\times}_n \boldsymbol{e} = \mathcal{X} \overline{\times}_n (\boldsymbol{A}^T \boldsymbol{y}). \tag{2.2}$$

Lemma 2. *If* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$*, then*

$$\mathcal{X} \overline{\times}_N \boldsymbol{e}_j = \mathcal{X}_j, j = 1, 2, ..., I_N, \tag{2.3}$$

where $\boldsymbol{e}_j$ *is the* j*-th column of the* I_N*-order identity matrix* $\boldsymbol{E}_{I_N}$*, and* $\mathcal{X}_j$ *denotes the* j*-th frontal slice of* $\mathcal{X}$*.*

For $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{N-1} \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{N-1} \times I_{\widehat{N}}}$, let $\mathcal{X} \boxtimes^{(N)} \mathcal{Y} \in \mathbb{R}^{I_N \times I_{\widehat{N}}}$ define the $\boxtimes^{(N)}$-product of $\mathcal{X}$ and $\mathcal{Y}$, i.e.,

$$[\mathcal{X} \boxtimes^{(N)} \mathcal{Y}]_{i,j} = trace(\mathcal{X}_{:,:,\ldots,:,i} \boxtimes^{(N-1)} \mathcal{Y}_{:,:,\ldots,:,j}), N = 2, 3, ...$$

In particular, $\mathcal{X} \boxtimes^1 \mathcal{Y} = \mathcal{X}^T \mathcal{Y}$ for $\mathcal{X} \in \mathbb{R}^{I_1}$ and $\mathcal{Y} \in \mathbb{R}^{I_1}$. For any $\mathcal{X}$, $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, a straightforward computation results in

$$\langle \mathcal{X}, \mathcal{Y} \rangle = trace(\mathcal{X} \boxtimes^{(N)} \mathcal{Y}), N = 1, 2, ..., \tag{2.4}$$

and (2.4) can be represented as

$$\mathcal{X} \boxtimes^{(N+1)} \mathcal{Y} = trace(\mathcal{X} \boxtimes^{(N)} \mathcal{Y}). \tag{2.5}$$

Therefore we have

$$\|\mathcal{X}\|^2 = \langle \mathcal{X}, \mathcal{X} \rangle = \mathcal{X} \boxtimes^{(N+1)} \mathcal{X}. \tag{2.6}$$

We also need the following results.

Lemma 3. *([1]) Let* $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times m}$ *be an* $(N+1)$*-order tensor with column tensors* $\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ *and vector* $\boldsymbol{z} \in \mathbb{R}^m$*. For any* $(N+1)$*-order tensor* $\mathcal{X}$ *with* N*-order column tensors* $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_m$*, it holds that*

$$\mathcal{X} \boxtimes^{(N+1)} (\mathcal{Y} \overline{\times}_{(N+1)} \boldsymbol{z}) = (\mathcal{X} \boxtimes^{(N+1)} \mathcal{Y}) \boldsymbol{z}. \tag{2.7}$$

Lemma 4. *([15]) Let* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{N-1} \times I_N}$ *and* $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{N-1} \times I_{\widehat{N}}}$ *be* N*-order tensors with column tensors* $\mathcal{X}_i (i = 1, ..., I_N)$ *and* $\mathcal{Y}_j (j = 1, ..., I_{\widehat{N}})$*. For* $\boldsymbol{A} \in \mathbb{R}^{I_N \times I_N}$ *and* $\boldsymbol{B} \in \mathbb{R}^{I_{\widehat{N}} \times I_{\widehat{N}}}$*, we have*

$$(\mathcal{X} \times_N \boldsymbol{A}^T) \boxtimes^{(N)} (\mathcal{Y} \times_N \boldsymbol{B}^T) = \boldsymbol{A}^T (\mathcal{X} \boxtimes^{(N)} \mathcal{Y}) \boldsymbol{B}. \tag{2.8}$$

3 A Tensor Lanczos $\mathcal{L}$-Biorthogonalization Algorithm

Define the linear operator of the form

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} &\to \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, \\ \mathcal{X} \mapsto \mathcal{L}(\mathcal{X}) &:= \mathcal{X} \times_1 \mathbf{A}_1 + \mathcal{X} \times_2 \mathbf{A}_2 + ... + \mathcal{X} \times_N \mathbf{A}_N, \end{aligned} \tag{3.1}$$

then the Sylvester tensor equation (1.1) can be represented as

$$\mathcal{L}(\mathcal{X}) = \mathcal{D}. \tag{3.2}$$

Let $\mathcal{L}^T$ define the dual linear operator of $\mathcal{L}$, i.e.,

$$\begin{aligned} \mathcal{L}^T : \mathbb{R}^{I_1\times I_2\times\ldots\times I_N} &\to \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}, \\ \mathcal{X} \mapsto \mathcal{L}^T(\mathcal{X}) := \mathcal{X}\times_1 \mathbf{A}_1^T &+ \mathcal{X}\times_2 \mathbf{A}_2^T + \ldots + \mathcal{X}\times_N \mathbf{A}_N^T, \end{aligned} \tag{3.3}$$

then it holds that $\langle \mathcal{L}(\mathcal{X}), \mathcal{Y}\rangle = \langle \mathcal{X}, \mathcal{L}^T(\mathcal{Y})\rangle$ for any $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$. Define the Krylov subspaces in tensor form as follows:

$$\mathcal{K}_m(\mathcal{L}, \mathcal{V}_1) = span\{\mathcal{V}_1, \mathcal{L}(\mathcal{V}_1), ..., \mathcal{L}^{m-1}(\mathcal{V}_1)\}, \tag{3.4}$$

where $\mathcal{L}^i(\mathcal{V}_1) = \mathcal{L}(\mathcal{L}^{i-1}(\mathcal{V}_1))$, $\mathcal{L}^0(\mathcal{V}_1) = \mathcal{V}_1$, then we have

$$\mathcal{K}_m(\mathcal{L}^T, \mathcal{W}_1) = span\{\mathcal{W}_1, \mathcal{L}^T(\mathcal{W}_1), ..., (\mathcal{L}^T)^{m-1}(\mathcal{W}_1)\}. \tag{3.5}$$

Algorithm 1 lists the Lanczos $\mathcal{L}$-Biorthogonalization procedure in tensor form that will be used to produce two series of biorthogonalization tensors.

Algorithm 1 A Lanczos $\mathcal{L}$-biorthogonalization procedure in tensor form.

Initial: Let $\mathcal{V}_0 = \mathcal{W}_0 = \mathcal{O} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$. Select $\mathcal{V}_1$ and $\mathcal{W}_1$ subject to $\langle \mathcal{W}_1, \mathcal{L}(\mathcal{V}_1)\rangle = 1$. Set $\delta_1 = \beta_1 = 0$.
Output: biorthogonalization tensor series $\mathcal{V}_j$, $\mathcal{W}_j$, $j = 1, 2, ...$
for $j = 1, 2, ...$ **do**
  $\alpha_j = \langle \mathcal{L}^2(\mathcal{V}_j), \mathcal{W}_j\rangle$
  $\overline{\mathcal{V}}_{j+1} = \mathcal{L}(\mathcal{V}_j) - \alpha_j\mathcal{V}_j - \beta_j\mathcal{V}_{j-1}$
  $\overline{\mathcal{W}}_{j+1} = \mathcal{L}^T(\mathcal{W}_j) - \alpha_j\mathcal{W}_j - \delta_j\mathcal{W}_{j-1}$
  $\delta_{j+1} = |\langle \overline{\mathcal{W}}_{j+1}, \mathcal{L}(\overline{\mathcal{V}}_{j+1})\rangle|^{\frac{1}{2}}$
  $\beta_{j+1} = \frac{\langle \overline{\mathcal{W}}_{j+1}, \mathcal{L}(\overline{\mathcal{V}}_{j+1})\rangle}{\delta_{j+1}}$
  $\mathcal{V}_{j+1} = \frac{\overline{\mathcal{V}}_{j+1}}{\delta_{j+1}}$
  $\mathcal{W}_{j+1} = \frac{\overline{\mathcal{W}}_{j+1}}{\beta_{j+1}}$
end for

We have the following results for Algorithm 1. The proofs of these results are similar to the proof of Proposition 1 in [11] by using the definitions of the inner product (2.4), (2.5) and linear operator $\mathcal{L}$ in (3.1) and are omitted.

Proposition 1. *If Alogrithm 1 stops at the m-th step, then the tensors $\mathcal{V}_j$ and $\mathcal{W}_i (i, j = 1, 2, ..., m)$ produced Algorithm 1 are $\mathcal{L}$-biorthogonal, i.e.,*

$$\langle \mathcal{W}_i, \mathcal{L}(\mathcal{V}_j)\rangle = \delta_{i,j}, 1 \le i, j \le m, \tag{3.6}$$

where

$$\delta_{i,j} = \begin{cases} 1, & i = j, \\ 0, & otherwise. \end{cases} \tag{3.7}$$

Proposition 2. *Suppose that $\widetilde{\mathcal{V}}_m$ is the $(N+1)$-order tensor with columns $\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_m$, and $\widetilde{\mathcal{W}}_m$ is the $(N+1)$-order tensor with the columns $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_m$,*

$\widetilde{\mathcal{H}}_m$ and $\widetilde{\mathcal{G}}_m$ are the $(N+1)$-order tensors with the columns $\mathcal{H}_j := \mathcal{L}(\mathcal{V}_j)$ and $\mathcal{G}_j := \mathcal{L}^T(\mathcal{W}_j)$ $(j = 1, 2, ..., m)$, respectively. Then we have

$$\widetilde{\mathcal{H}}_m = \widetilde{\mathcal{V}}_{m+1} \times_{(N+1)} \underline{\boldsymbol{T}_m^T} \tag{3.8}$$

and

$$\widetilde{\mathcal{G}}_m = \widetilde{\mathcal{W}}_{m+1} \times_{(N+1)} \underline{\boldsymbol{T}_m}, \tag{3.9}$$

where

$$\underline{\boldsymbol{T}_m} = \begin{pmatrix} \boldsymbol{T}_m \\ \delta_{m+1}\boldsymbol{e}_m^T \end{pmatrix} \tag{3.10}$$

with

$$\boldsymbol{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \tag{3.11}$$

being a triangular matrix with its elements generated by Algorithm 1. Moreover, it holds that

$$\widetilde{\mathcal{W}}_m \boxtimes^{(N+1)} \widetilde{\mathcal{H}}_m = \boldsymbol{E}_m \tag{3.12}$$

and

$$\widetilde{\mathcal{W}}_m \boxtimes^{(N+1)} \mathcal{L}(\widetilde{\mathcal{H}}_m) = \boldsymbol{T}_m, \tag{3.13}$$

where $\boldsymbol{E}_m$ denotes the identity matrix with m order.

We remark that (3.8) and (3.9) can be represented as

$$\widetilde{\mathcal{H}}_m = \widetilde{\mathcal{V}}_m \times_{(N+1)} T_m^T + \delta_{m+1}\mathcal{Z}_1 \times_{(N+1)} \mathbf{K}_m \tag{3.14}$$

and

$$\widetilde{\mathcal{G}}_m = \widetilde{\mathcal{W}}_m \times_{(N+1)} T_m + \beta_{m+1}\mathcal{Z}_2 \times_{(N+1)} \mathbf{K}_m, \tag{3.15}$$

where $\mathcal{Z}_1$ is an $(N+1)$-order tensor with m column tensors $\mathcal{O}, ..., \mathcal{O}, \mathcal{V}_{m+1}$, and $\mathcal{Z}_2$ is an $(N+1)$-order tensor with m column tensors $\mathcal{O}, ..., \mathcal{O}, \mathcal{W}_{m+1}$, and $\mathbf{K}_m$ is an $m \times m$ matrix of the form $\mathbf{K}_m = [0, ..., 0, \mathbf{e}_m]$ with $\mathbf{e}_m$ being the m-th column of $\mathbf{E}_m$.

With the results above we can present the Lanczos $\mathcal{L}$-Biorthogonalization algorithm in tensor form for solving (1.1). For any initial tensor $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$, let $\mathcal{R}_0 = \mathcal{D} - \mathcal{L}(\mathcal{X}_0)$ denote its residual. Let $\mathcal{V}_1 = \mathcal{R}_0/\|\mathcal{R}_0\|$ and

$$\mathcal{X}_m \in \mathcal{X}_0 + \mathcal{K}_m(\mathcal{L}, \mathcal{V}_1), \tag{3.16}$$

then

$$\mathcal{R}_m = (\mathcal{D} - \mathcal{L}(\mathcal{X}_m)) \perp \mathcal{L}^T(\mathcal{K}_m(\mathcal{L}^T, \mathcal{W}_1)). \tag{3.17}$$

It is easy to verify that a series of tensors $\{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_m\}$ produced via Algorithm 1 form a basis of $\mathcal{K}_m(\mathcal{L}, \mathcal{V}_1)$. Thus we have

$$\mathcal{X}_m = \mathcal{X}_0 + \widetilde{\mathcal{V}}_m \overline{\times}_{(N+1)} \mathbf{y}_m, \tag{3.18}$$

where $\mathbf{y}_m \in \mathbb{R}^m$. By Eq (3.17) and (3.18), we have

$$\langle \mathcal{D} - \mathcal{L}(\mathcal{X}_0 + \widetilde{\mathcal{V}}_m \overline{\times}_{(N+1)} \mathbf{y}_m), \mathcal{L}^T(\widetilde{\mathcal{W}}_m) \rangle = 0. \tag{3.19}$$

A further computation results in

$$\langle \mathcal{L}(\widetilde{\mathcal{H}}_m) \overline{\times}_{(N+1)} \mathbf{y}_m, \widetilde{\mathcal{W}}_m \rangle = \langle \mathcal{L}(\mathcal{R}_0), \widetilde{\mathcal{W}}_m \rangle. \tag{3.20}$$

Through a simple inner product operation and according to Eq.(2.6) and Lemma 3 we have

$$(\widetilde{\mathcal{W}}_m \boxtimes^{(N+1)} \mathcal{L}(\widetilde{\mathcal{H}}_m))\mathbf{y}_m = \widetilde{\mathcal{W}}_m \boxtimes^{(N+1)} \mathcal{L}(\mathcal{R}_0). \tag{3.21}$$

Submitting Eq.(3.13) into Eq.(3.21) results in the tridiagonal system on y_m:

$$\mathbf{T}_m \mathbf{y}_m = \|\mathcal{R}_0\| \mathbf{e}_1. \tag{3.22}$$

Once we compute y_m by (3.22), we get the solution $\mathcal{X}_m$ of (1.1) by (3.18). We summarize this method in Algorithm 2, which is called Tensor Lanczos $\mathcal{L}$-Biorthogonalization algorithm (TLB).

Algorithm 2 TLB: A tensor Lanczos $\mathcal{L}$-biorthogonalization Algorithm for solving (1.1)

Choose an initial tensor $\mathcal{X}_0$ and compute $\mathcal{R}_0 = \mathcal{D} - \mathcal{L}(\mathcal{X}_0)$.
Set $\mathcal{V}_1 = \frac{\mathcal{R}_0}{\|\mathcal{R}_0\|}$, choose a tensor $\mathcal{W}_1$ such that $\langle \mathcal{L}(\mathcal{V}_1), \mathcal{W}_1 \rangle = 1$.
for $m = 1, 2, ...$ until convergence **do**
  Compute Lanczos $\mathcal{L}$-Biorthogonalization tensors $\mathcal{V}_1, ..., \mathcal{V}_m$, $\mathcal{W}_1, ..., \mathcal{W}_m$ and $\mathbf{T}_m$ by Algorithm 1.
  Compute $\mathbf{y}_m$ by (3.22).
end for
Compute the solution $\mathcal{X}_m$ of (1.1) by (3.18).

We remark Algorithm 2 have to compute the inverse of $\mathbf{T}_m$. When $\mathbf{T}_m$ is of large size, it needs much computation. We present two improved algorithms for Algorithm 2 in the next section.

4 The TBiCOR and TCORS Algorithms

4.1 The TBiCOR Algorithm

In this subsection, we develop an improved algorithm by introducing the LU decomposition to $\mathbf{T}_m$ in Algorithm 2.

Let the LU decomposition of $\mathbf{T}_m$ be

$$\mathbf{T}_m = \mathbf{L}_m \mathbf{U}_m, \tag{4.1}$$

then, according to Lemma 1, substituting (3.22) and (4.1) into (3.18) results in

$$\begin{aligned}\mathcal{X}_m &= \mathcal{X}_0 + \widetilde{\mathcal{V}}_m \overline{\times}_{(N+1)} \mathbf{y}_m \\ &= \mathcal{X}_0 + \widetilde{\mathcal{V}}_m \overline{\times}_{(N+1)} (\mathbf{U}_m^{-1} \mathbf{L}_m^{-1} (\|\mathcal{R}_0\| \mathbf{e}_1)) \\ &= \mathcal{X}_0 + \widetilde{\mathcal{P}}_m \overline{\times}_{(N+1)} \mathbf{z}_m, \end{aligned} \tag{4.2}$$

where $\mathbf{z}_m = \mathbf{L}_m^{-1}(\|\mathcal{R}_0\|\mathbf{e}_1)$ and $\widetilde{\mathcal{P}}_m = \widetilde{\mathcal{V}}_m \times_{(N+1)} (\mathbf{U}_m^{-1})^T$.

We consider the solution of the system $\mathcal{L}^T(\mathcal{X}^*) = \mathcal{D}^*$. The dual approximation $\mathcal{X}_m^*$ is the subspace $\mathcal{X}_0^* + \mathcal{K}_m(\mathcal{L}^T, \mathcal{W}_1)$ that satisfies

$$(\mathcal{D}^* - \mathcal{L}^T(\mathcal{X}_m^*)) \perp \mathcal{L}(\mathcal{K}_m(\mathcal{L}, \mathcal{V}_1)).$$

Set $\mathcal{R}_0^* = \mathcal{D}^* - \mathcal{L}^T(\mathcal{X}_0^*)$ and $\mathcal{W}_1 = \mathcal{R}_0^*/\|\mathcal{R}_0^*\|$. If we choose $\mathcal{V}_1$ such that $\langle \mathcal{V}_1, \mathcal{L}(\mathcal{W}_1)\rangle = 1$, then similar to (3.18)-(3.22), the solution of the dual system $\mathcal{L}^T(\mathcal{X}^*) = \mathcal{D}^*$ can be represented as

$$\mathcal{X}_m^* = \mathcal{X}_0^* + \widetilde{\mathcal{W}}_m \overline{\times}_{(N+1)} \mathbf{y}_m^*, \tag{4.3}$$

where $\mathbf{y}_m^*$ is derived from

$$\mathbf{T}_m^T \mathbf{y}_m^* = \|\mathcal{R}_0^*\| \mathbf{e}_1. \tag{4.4}$$

Similar to (4.2), according to Lemma 1, (4.3) together with (4.1) and (4.4) results in

$$\mathcal{X}_m^* = \mathcal{X}_0^* + \widetilde{\mathcal{P}}_m^* \overline{\times}_{(N+1)} \mathbf{z}_m^*,$$

where $\widetilde{\mathcal{P}}_m^* = \widetilde{\mathcal{W}}_m \times_{(N+1)} \mathbf{L}_m^{-1}$, and $\mathbf{z}_m^* = (\mathbf{U}_m^T)^{-1}(\|\mathcal{R}_0^*\|\mathbf{e}_1)$.

Proposition 3. *Let $\mathcal{R}_i = \mathcal{D} - \mathcal{L}(\mathcal{X}_i)$ and $\mathcal{R}_i^* = \mathcal{D}^* - \mathcal{L}^T(\mathcal{X}_i^*)$ are the i-th residual tensor and the i-th dual residual tensor, respectively, then it holds that*

$$\langle \mathcal{L}(\mathcal{R}_i), \mathcal{R}_j^* \rangle = 0, (0 \le i \ne j \le k). \tag{4.5}$$

Proof. According to Lemma 1, (3.14), (3.15), (3.18) and (4.3), we have

$$\begin{aligned}\mathcal{R}_i &= \mathcal{D} - \mathcal{L}(\mathcal{X}_0 + \widetilde{\mathcal{V}}_i \overline{\times}_{(N+1)} \mathbf{y}_i) \\ &= \mathcal{R}_0 - \mathcal{L}(\widetilde{\mathcal{V}}_i) \overline{\times}_{(N+1)} \mathbf{y}_i \\ &= \mathcal{R}_0 - \widetilde{\mathcal{V}}_i \times_{(N+1)} \mathbf{T}_i^T \overline{\times}_{(N+1)} \mathbf{y}_i - \delta_{i+1} \mathcal{Z}_1 \times_{(N+1)} \mathbf{K}_i \overline{\times}_{(N+1)} \mathbf{y}_i \\ &= \mathcal{R}_0 - \widetilde{\mathcal{V}}_i \overline{\times}_{(N+1)} (\mathbf{T}_i \mathbf{y}_i) - \delta_{i+1} \mathcal{Z}_1 \overline{\times}_{(N+1)} (\mathbf{K}_i^T \mathbf{y}_i) \\ &= -\delta_{i+1} \mathbf{e}_i^T \mathbf{y}_i \mathcal{V}_{i+1}. \end{aligned} \tag{4.6}$$

Similarly, we can prove that

$$\mathcal{R}_j^* = -\beta_{j+1} \mathbf{e}_j^T \mathbf{y}_j^* \mathcal{W}_{j+1}. \tag{4.7}$$

(4.6) and (4.7) together with Proposition 1 result in (4.5). □

Proposition 4. *Let $\mathcal{P}_i$ and $\mathcal{P}_i^*(i=1,...,k)$ are the i-th column tensor of $\widetilde{\mathcal{P}}_k$ and $\widetilde{\mathcal{P}}_k^*$, respectively. It holds that*

$$\langle \mathcal{L}^2(\mathcal{P}_i), \mathcal{P}_j^* \rangle = 0 (i,j=1,...,k, i \neq j). \tag{4.8}$$

Proof. According to Lemma 4 and (3.13), we have

$$\begin{aligned}
(\widetilde{\mathcal{P}}_k^* \boxtimes^{(N+1)} \mathcal{L}^2(\widetilde{\mathcal{P}}_k))_{ij} &= ((\widetilde{\mathcal{W}}_k \times_{(N+1)} \mathbf{L}_k^{-1}) \boxtimes^{(N+1)} \mathcal{L}^2(\widetilde{\mathcal{V}}_k \times_{(N+1)} (\mathbf{U}_k^{-1})^T))_{ij} \\
&= (\mathbf{L}_k^{-1}(\widetilde{\mathcal{W}}_k \boxtimes^{(N+1)} \mathcal{L}(\widetilde{\mathcal{H}}_k))\mathbf{U}_k^{-1})_{ij} \\
&= (\mathbf{L}_k^{-1}\mathbf{T}_k\mathbf{U}_k^{-1})_{ij} \\
&= (\mathbf{E}_k)_{ij},
\end{aligned} \tag{4.9}$$

which implies that (4.8) holds. □

For a given initial guess $\mathcal{X}_0$, let $\mathcal{R}_0 = \mathcal{D} - \mathcal{L}(\mathcal{X}_0)$ and $\mathcal{P}_0 = \mathcal{R}_0$. Set

$$\mathcal{X}_{j+1} = \mathcal{X}_j + \alpha_j \mathcal{P}_j, \tag{4.10}$$

$$\mathcal{R}_{j+1} = \mathcal{R}_j - \alpha_j \mathcal{L}(\mathcal{P}_j), \tag{4.11}$$

$$\mathcal{P}_{j+1} = \mathcal{R}_{j+1} + \beta_j \mathcal{P}_j, \quad j = 0, 1, ... \tag{4.12}$$

Similarly, for the dual linear system $\mathcal{L}^T(\mathcal{X}^*) = \mathcal{D}^*$, we set

$$\mathcal{R}_{j+1}^* = \mathcal{R}_j^* - \alpha_j \mathcal{L}^T(\mathcal{P}_j^*), \mathcal{R}_0^* = \mathcal{L}(\mathcal{R}_0), \tag{4.13}$$

$$\mathcal{P}_{j+1}^* = \mathcal{R}_{j+1}^* + \beta_j \mathcal{P}_j^* \quad for \quad j = 0, 1, ... \tag{4.14}$$

Now we determine α_j and β_j in (4.11)-(4.14). According to (4.11) we have that

$$\langle \mathcal{L}(\mathcal{R}_{j+1}), \mathcal{R}_j^* \rangle = \langle \mathcal{L}(\mathcal{R}_j) - \alpha_j \mathcal{L}^2(\mathcal{P}_j), \mathcal{R}_j^* \rangle = 0, \tag{4.15}$$

then by Propositions 3, 4 and (4.11)-(4.14) it holds that

$$\alpha_j = \frac{\langle \mathcal{L}(\mathcal{R}_j), \mathcal{R}_j^* \rangle}{\langle \mathcal{L}^2(\mathcal{P}_j), \mathcal{R}_j^* \rangle} = \frac{\langle \mathcal{L}(\mathcal{R}_j), \mathcal{R}_j^* \rangle}{\langle \mathcal{L}(\mathcal{P}_j), \mathcal{L}^T(\mathcal{P}_j^*) \rangle}. \tag{4.16}$$

Similarly according to

$$\langle \mathcal{L}^2(\mathcal{P}_{j+1}), \mathcal{P}_j^* \rangle = \langle \mathcal{L}(\mathcal{P}_{j+1}), \mathcal{L}^T(\mathcal{P}_j^*) \rangle = 0, \tag{4.17}$$

we have

$$\beta_j = -\frac{\langle \mathcal{L}(\mathcal{R}_{j+1}), \mathcal{L}^T(\mathcal{P}_j^*) \rangle}{\langle \mathcal{L}(\mathcal{P}_j), \mathcal{L}^T(\mathcal{P}_j^*) \rangle} = \frac{\langle \mathcal{L}(\mathcal{R}_{j+1}), \mathcal{R}_{j+1}^* \rangle}{\langle \mathcal{L}(\mathcal{R}_j), \mathcal{R}_j^* \rangle}. \tag{4.18}$$

Algorithm 3 summarizes the biconjugate $\mathcal{L}$-orthogonal residual algorithm in Tensor form for solving (1.1), which is abbreviated as TBiCOR.

We have the following convergence properties on Algorithm 3.

Algorithm 3 TBiCOR: A tensor biconjugate $\mathcal{L}$-orthogonal residual algorithm for solving (1.1)

Compute $\mathcal{R}_0 = \mathcal{D} - \mathcal{L}(\mathcal{X}_0)$ ($\mathcal{X}_0$ is an initial guess)
Set $\mathcal{R}_0^* = \mathcal{L}(\mathcal{R}_0)$
Set $\mathcal{P}_{-1}^* = \mathcal{P}_{-1} = 0$, $\beta_{-1} = 0$
for n=0,1,..., until convergence **do**
  $\mathcal{P}_n = \mathcal{R}_n + \beta_{n-1}\mathcal{P}_{n-1}$
  $\mathcal{P}_n^* = \mathcal{R}_n^* + \beta_{n-1}\mathcal{P}_{n-1}^*$
  $\mathcal{S}_n = \mathcal{L}(\mathcal{P}_n)$
  $\mathcal{S}_n^* = \mathcal{L}^T(\mathcal{P}_n^*)$
  $\mathcal{T}_n = \mathcal{L}(\mathcal{R}_n)$
  $\alpha_n = \frac{\langle \mathcal{R}_n^*, \mathcal{T}_n \rangle}{\langle \mathcal{S}_n^*, \mathcal{S}_n \rangle}$
  $\mathcal{X}_{n+1} = \mathcal{X}_n + \alpha_n \mathcal{P}_n$
  $\mathcal{R}_{n+1} = \mathcal{R}_n - \alpha_n \mathcal{S}_n$
  $\mathcal{R}_{n+1}^* = \mathcal{R}_n^* - \alpha_n \mathcal{S}_n^*$
  $\mathcal{T}_{n+1} = \mathcal{L}(\mathcal{R}_{n+1})$
  $\beta_n = \frac{\langle \mathcal{R}_{n+1}^*, \mathcal{T}_{n+1} \rangle}{\langle \mathcal{R}_n^*, \mathcal{T}_n \rangle}$
end for

Theorem 1. *Assume that the Sylvester tensor equation* (1.1) *is consistent. For any initial tensor* $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, *Algorithm 3 converges to an exact solution of* (1.1) *at most* $M = I_1 \times I_2 \times \ldots \times I_N$ *iteration steps in the absence of roundoff errors.*

Proof. Suppose that $\mathcal{R}_k \neq \mathcal{O}(k = 0, 1, ..., M)$ and

$$\sum_{k=0}^{M} \lambda_k \mathcal{R}_k = \mathcal{O}.$$

According to Proposition 3, we have

$$\begin{aligned} 0 = \langle \mathcal{R}_i^*, \sum_{k=0}^{M} \lambda_k \mathcal{L}(\mathcal{R}_k) \rangle &= \sum_{k=0}^{M} \lambda_k \langle \mathcal{R}_i^*, \mathcal{L}(\mathcal{R}_k) \rangle \\ &= \lambda_i \langle \mathcal{R}_i^*, \mathcal{L}(\mathcal{R}_i) \rangle, i = 0, 1, ..., M. \end{aligned}$$

When Algorithm 3 does not break down, $\langle \mathcal{R}_i^*, \mathcal{L}(\mathcal{R}_i) \rangle \neq 0 (i = 0, 1, ..., M)$, which leads to $\lambda_i = 0 (i = 0, 1, ..., M)$. This means that $\mathcal{R}_0$,$\mathcal{R}_1$,...,$\mathcal{R}_M$ are linearly independent, while the dimension of tensor space $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is M. This is a contradiction. Thus Algorithm 3 converges to an exact solution within M steps. □

4.2 The TCORS Algorithm

This subsection presents an improved method on Algorithm 3 by introducing a squared operator of the residual of $\mathcal{X}_n$ produced by Algorithm 3. The proposed method is called the conjugate $\mathcal{L}$-orthogonal residual squared algorithm in tensor form, which is abbreviated as TCORS.

Algorithm 4 TCORS: A tensor conjugate $\mathcal{L}$-orthogonal residual squared algorithm for solving (1.1)

Compute $\mathcal{R}_0 = \mathcal{D} - \mathcal{L}(\mathcal{X}_0)$; ($\mathcal{X}_0$ is an initial guess)
Set $\mathcal{R}_0^* = \mathcal{L}(\mathcal{R}_0)$
for n=1,2,..., until convergence **do**
  $\mathcal{U}_0 = \mathcal{R}_0$, $\widehat{\mathcal{Z}} = \mathcal{L}(\mathcal{U}_{n-1})$; $\rho_{n-1} = \langle \mathcal{R}_0^*, \widehat{\mathcal{Z}} \rangle$; $\mathcal{Z}_{n-1} = \mathcal{U}_{n-1}$
  if $\rho_{n-1} = 0$, stop and reset the initial tensor $\mathcal{X}_0$.
  if $n = 1$
  $\mathcal{T}_0 = \mathcal{U}_0$; $\mathcal{D}_0 = \mathcal{T}_0$; $\mathcal{C}_0 = \widehat{\mathcal{Z}}$; $\mathcal{Q}_0 = \widehat{\mathcal{Z}}$
  else
  $\beta_{n-2} = \rho_{n-1}/\rho_{n-2}$; $\mathcal{T}_{n-1} = \mathcal{U}_{n-1} + \beta_{n-2}\mathcal{H}_{n-2}$
  $\mathcal{D}_{n-1} = \mathcal{Z}_{n-1} + \beta_{n-2}\mathcal{V}_{n-2}$; $\mathcal{C}_{n-1} = \widehat{\mathcal{Z}} + \beta_{n-2}\mathcal{F}_{n-2}$
  $\mathcal{Q}_{n-1} = \mathcal{C}_{n-1} + \beta_{n-2}(\mathcal{F}_{n-2} + \beta_{n-2}\mathcal{Q}_{n-2})$
  end if
  $\widehat{\mathcal{Q}} = \mathcal{L}(\mathcal{Q}_{n-1})$
  $\alpha_{n-1} = \rho_{n-1}/\langle \mathcal{R}_0^*, \widehat{\mathcal{Q}} \rangle$; $\mathcal{H}_{n-1} = \mathcal{T}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1}$
  $\mathcal{V}_{n-1} = \mathcal{D}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1}$; $\mathcal{F}_{n-1} = \mathcal{C}_{n-1} - \alpha_{n-1}\widehat{\mathcal{Q}}$
  $\mathcal{X}_n = \mathcal{X}_{n-1} + \alpha_{n-1}(2\mathcal{D}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1})$
  $\mathcal{U}_n = \mathcal{U}_{n-1} - \alpha_{n-1}(2\mathcal{C}_{n-1} - \alpha_{n-1}\widehat{\mathcal{Q}})$
end for

The residual tensor of $\mathcal{X}_n$ produced by Algorithm 3 can be represented as

$$\begin{aligned} \mathcal{R}_n &= a_0\mathcal{R}_0 + a_1\mathcal{L}(\mathcal{R}_0) + a_2\mathcal{L}^2(\mathcal{R}_0) + ... + a_n\mathcal{L}^n(\mathcal{R}_0) \\ &= (a_0\mathcal{L}^0 + a_1\mathcal{L} + a_2\mathcal{L}^2 + ... + a_n\mathcal{L}^n)\mathcal{R}_0, \end{aligned} \tag{4.19}$$

where a_i is determined by Algorithm 3. Denote $\varphi_n(\mathcal{L}) = a_0\mathcal{L}^0 + a_1\mathcal{L} + a_2\mathcal{L}^2 + ... + a_n\mathcal{L}^n$, then (4.19) can be represented as

$$\mathcal{R}_n = \varphi_n(\mathcal{L})\mathcal{R}_0. \tag{4.20}$$

Similarly, we have

$$\mathcal{P}_n = \phi_n(\mathcal{L})\mathcal{R}_0, \tag{4.21}$$

where $\phi_n(\mathcal{L}) = b_0\mathcal{L}^0 + b_1\mathcal{L} + b_2\mathcal{L}^2 + ... + b_n\mathcal{L}^n$, and b_i can be derived by Algorithm 3. For the directions $\mathcal{R}_n^*$ and $\mathcal{P}_n^*$ in Algorithm (3), replacing $\mathcal{L}$ in (4.20) and (4.21) with $\mathcal{L}^T$ results in

$$\mathcal{R}_n^* = \varphi_n(\mathcal{L}^T)\mathcal{R}_0^*, \mathcal{P}_n^* = \phi_n(\mathcal{L}^T)\mathcal{R}_0^*.$$

Thus α_n in (4.16) and β_n in (4.18) can be represented as

$$\alpha_n = \frac{\langle \mathcal{L}(\varphi_n(\mathcal{L})\mathcal{R}_0), \varphi_n(\mathcal{L}^T)\mathcal{R}_0^* \rangle}{\langle \mathcal{L}(\phi_n(\mathcal{L})\mathcal{R}_0), \mathcal{L}^T(\phi_n(\mathcal{L}^T)\mathcal{R}_0^*) \rangle} = \frac{\langle \mathcal{L}(\varphi_n^2(\mathcal{L})\mathcal{R}_0), \mathcal{R}_0^* \rangle}{\langle \mathcal{L}^2(\phi_n^2(\mathcal{L})\mathcal{R}_0), \mathcal{R}_0^* \rangle}, \tag{4.22}$$

$$\beta_n = \frac{\langle \varphi_{n+1}(\mathcal{L}^T)\mathcal{R}_0^*, \mathcal{L}(\varphi_{n+1}(\mathcal{L})\mathcal{R}_0)\rangle}{\langle \varphi_n(\mathcal{L}^T)\mathcal{R}_0^*, \mathcal{L}(\varphi_n(\mathcal{L})\mathcal{R}_0)\rangle} = \frac{\langle \mathcal{L}(\varphi_{n+1}^2(\mathcal{L})\mathcal{R}_0), \mathcal{R}_0^*\rangle}{\langle \mathcal{L}(\varphi_n^2(\mathcal{L})\mathcal{R}_0), \mathcal{R}_0^*\rangle}. \tag{4.23}$$

According to (4.11)-(4.12), φ_j and ϕ_j can be expressed as

$$\varphi_{j+1}(\mathcal{L}) = \varphi_j(\mathcal{L}) - \alpha_j\mathcal{L}(\phi_j(\mathcal{L})), \tag{4.24}$$

$$\phi_{j+1}(\mathcal{L}) = \varphi_{j+1}(\mathcal{L}) + \beta_j\phi_j(\mathcal{L}), \tag{4.25}$$

respectively. Squaring on both sides of (4.24) and (4.25) results in

$$\varphi_{j+1}^2(\mathcal{L}) = \varphi_j^2(\mathcal{L}) - 2\alpha_j\mathcal{L}(\phi_j(\mathcal{L})\varphi_j(\mathcal{L})) + \alpha_j^2\mathcal{L}^2(\phi_j^2(\mathcal{L})), \tag{4.26}$$

$$\phi_{j+1}^2(\mathcal{L}) = \varphi_{j+1}^2(\mathcal{L}) + 2\beta_j\varphi_{j+1}(\mathcal{L})\phi_j(\mathcal{L}) + \beta_j^2\phi_j^2(\mathcal{L}). \tag{4.27}$$

Furthermore, we have

$$\varphi_j(\mathcal{L})\phi_j(\mathcal{L}) = \varphi_j^2(\mathcal{L}) + \beta_{j-1}\varphi_j(\mathcal{L})\phi_{j-1}(\mathcal{L}), \tag{4.28}$$

$$\varphi_{j+1}(\mathcal{L})\phi_j(\mathcal{L}) = \varphi_j^2(\mathcal{L}) + \beta_{j-1}\varphi_j(\mathcal{L})\phi_{j-1}(\mathcal{L}) - \alpha_j\mathcal{L}(\phi_j^2(\mathcal{L})). \tag{4.29}$$

Taking (4.28) into (4.26) results in

$$\varphi_{j+1}^2(\mathcal{L}) = \varphi_j^2(\mathcal{L}) - \alpha_j\mathcal{L}(2\varphi_j^2(\mathcal{L}) + 2\beta_{j-1}\varphi_j(\mathcal{L})\phi_{j-1}(\mathcal{L}) - \alpha_j\mathcal{L}(\phi_j^2(\mathcal{L}))). \tag{4.30}$$

Denote

$$\mathcal{U}_j = \varphi_j^2(\mathcal{L})\mathcal{R}_0, \tag{4.31}$$

$$\mathcal{Q}_j = \mathcal{L}(\phi_j^2(\mathcal{L}))\mathcal{R}_0, \tag{4.32}$$

$$\mathcal{F}_j = \mathcal{L}(\varphi_{j+1}(\mathcal{L})\phi_j(\mathcal{L}))\mathcal{R}_0, \tag{4.33}$$

then

$$\mathcal{U}_{j+1} = \mathcal{U}_j - \alpha_j(2\mathcal{L}(\mathcal{U}_j) + 2\beta_{j-1}\mathcal{F}_{j-1} - \alpha_j\mathcal{L}(\mathcal{Q}_j)), \tag{4.34}$$

$$\mathcal{Q}_{j+1} = \mathcal{L}(\mathcal{U}_{j+1}) + 2\beta_j\mathcal{F}_j + \beta_j^2\mathcal{Q}_j, \tag{4.35}$$

$$\mathcal{F}_j = \mathcal{L}(\mathcal{U}_j) + \beta_{j-1}\mathcal{F}_{j-1} - \alpha_j\mathcal{L}(\mathcal{Q}_j). \tag{4.36}$$

Denote

$$\mathcal{L}(\mathcal{U}_j) + \beta_{j-1}\mathcal{F}_{j-1} = \mathcal{C}_j, \tag{4.37}$$

then (4.34)-(4.36) can be represented as

$$\mathcal{U}_{j+1} = \mathcal{U}_j - \alpha_j(2\mathcal{C}_j - \alpha_j\mathcal{L}(\mathcal{Q}_j)), \tag{4.38}$$

$$\mathcal{Q}_{j+1} = \mathcal{C}_{j+1} + \beta_j\mathcal{F}_j + \beta_j^2\mathcal{Q}_j, \tag{4.39}$$

$$\mathcal{F}_j = \mathcal{C}_j - \alpha_j\mathcal{L}(\mathcal{Q}_j). \tag{4.40}$$

Algorithm 4 summarizes the TCORS algorithm for solving (1.1). The following results list the convergence of Algorithm 4.

Theorem 2. *Assume the Sylvester tensor equation* (1.1) *is consistent. For any initial tensor* $\mathcal{X}_0 \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$*, the iteration solution* $\{\mathcal{X}_n\}$ *produced by Algorithm 4 converge to an exact solution of* (1.1) *at most* $M = I_1 \times I_2 \times \ldots \times I_N$ *iteration steps without roundoff errors.*

Proof. The proof of Theorem 2 is similar to that of Theorem 1 by replacing $\mathcal{R}_k$ with $\mathcal{U}_k$, thus is omitted. □

5 Preconditioned BiCOR and TCORSs Algorithms

This section presents two preconditioned methods based on Algorithms 3-4 for solving Eq.(5.1).

Using the definition of the Kronecker product in [4], one can transform Eq.(1.1) to its equivalent linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.1}$$

where $\mathbf{A} = \mathbf{E}_{I_N} \otimes \cdots \otimes \mathbf{E}_{I_2} \otimes \mathbf{A}_1 + \cdots + \mathbf{A}_N \otimes \mathbf{E}_{I_{N-1}} \otimes \cdots \otimes \mathbf{E}_{I_1}$, '$\otimes$' denotes the Kronecker product, $\mathbf{x} = vec(\mathcal{X})$, $\mathbf{b} = vec(\mathcal{D})$. We refer to [17] for more details.

Algorithm 5 PTBiCOR: A preconditioned tensor biconjugate $\tilde{\mathcal{L}}$-orthogonal residual algorithm for solving (1.1)

Compute matrices $\mathbf{Q}_i (i = 1 \dots N)$ and $\tilde{\mathcal{D}} = \mathcal{D} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N \mathbf{Q}_1^{-1}$.
Replace $\mathcal{L}$, $\mathcal{L}^T$ in algorithm 3 with $\tilde{\mathcal{L}}$, $\tilde{\mathcal{L}}^T$, $\tilde{\mathcal{L}}(\mathcal{X}) = \mathcal{X} \times_1 (\mathbf{Q}_N^{-1}\mathbf{A}_1) \times_2 \cdots \times_N \mathbf{Q}_1^{-1} + \cdots + \mathcal{X} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N (\mathbf{Q}_1^{-1}\mathbf{A}_N)$ and $\tilde{\mathcal{L}}^T(\mathcal{X}) = \mathcal{X} \times_1 (\mathbf{Q}_N^{-1}\mathbf{A}_1)^T \times_2 \cdots \times_N (\mathbf{Q}_1^{-1})^T + \cdots + \mathcal{X} \times_1 (\mathbf{Q}_N^{-1})^T \times_2 \cdots \times_N (\mathbf{Q}_1^{-1}\mathbf{A}_N)^T$.
Compute $\mathcal{R}_0 = \tilde{\mathcal{D}} - \tilde{\mathcal{L}}(\mathcal{X}_0)$ ($\mathcal{X}_0$ is an initial guess)
Set $\mathcal{R}_0^* = \tilde{\mathcal{L}}(\mathcal{R}_0)$
Set $\mathcal{P}_{-1}^* = \mathcal{P}_{-1} = 0$, $\beta_{-1} = 0$
for n=0,1,..., until convergence **do**
  $\mathcal{P}_n = \mathcal{R}_n + \beta_{n-1}\mathcal{P}_{n-1}$
  $\mathcal{P}_n^* = \mathcal{R}_n^* + \beta_{n-1}\mathcal{P}_{n-1}^*$
  $\mathcal{S}_n = \tilde{\mathcal{L}}(\mathcal{P}_n)$
  $\mathcal{S}_n^* = \tilde{\mathcal{L}}^T(\mathcal{P}_n^*)$
  $\mathcal{T}_n = \tilde{\mathcal{L}}(\mathcal{R}_n)$
  $\alpha_n = \frac{\langle \mathcal{R}_n^*, \mathcal{T}_n \rangle}{\langle \mathcal{S}_n^*, \mathcal{S}_n \rangle}$
  $\mathcal{X}_{n+1} = \mathcal{X}_n + \alpha_n \mathcal{P}_n$
  $\mathcal{R}_{n+1} = \mathcal{R}_n - \alpha_n \mathcal{S}_n$
  $\mathcal{R}_{n+1}^* = \mathcal{R}_n^* - \alpha_n \mathcal{S}_n^*$
  $\mathcal{T}_{n+1} = \tilde{\mathcal{L}}(\mathcal{R}_{n+1})$
  $\beta_n = \frac{\langle \mathcal{R}_{n+1}^*, \mathcal{T}_{n+1} \rangle}{\langle \mathcal{R}_n^*, \mathcal{T}_n \rangle}$
end for

We are interested in constructing a preconditioner $\mathbf{M}$ that transforms Eq.(1.1) to a new system

$$\mathbf{M}\mathbf{A}\mathbf{x} = \mathbf{M}\mathbf{b}, \tag{5.2}$$

which has the same solution with Eq.(5.1) and has better spectral properties than Eq.(5.1) does. In particular, if $\mathbf{M}$ is a good approximation of $\mathbf{A}^{-1}$, then Eq.(5.2) can be solved more effectively than Eq.(5.1). Using the nearest Kronecker product (NKP) in [28], Chen and Lu [9] presented an efficient preconditioner for solving Eq.(5.1) based on GMRES in tensor form, which is abbreviated as preconditioned GMRES (PGMRES) later. Zhang and Wang in [30] gave a preconditioned BiCG (PBiCG) and a preconditioned BiCR (PBiCR) based on NKP in [28]. The preconditioner based on NKP approximates $\mathbf{A}^{-1}$ by $\mathbf{Q}_1^{-1} \otimes \mathbf{Q}_2^{-1} \otimes \cdots \otimes \mathbf{Q}_N^{-1}$ with

$$\begin{cases} \mathbf{Q}_1 \approx a_{11}\mathbf{A}_N + a_{12}\mathbf{E}_{I_N}, \\ \mathbf{Q}_2 \approx a_{21}\mathbf{A}_{N-1} + a_{22}\mathbf{E}_{I_{N-1}}, \\ \vdots \\ \mathbf{Q}_N \approx a_{N1}\mathbf{A}_1 + a_{N2}\mathbf{E}_1, \end{cases} \tag{5.3}$$

where the optimal parameters a_{ij} in (5.3) can be computed by using the nonlinear optimization software, such as *fminsearch* in MATLAB.

Introducing the preconditioner based on NKP to Algorithms 3 and 4, we get our preconditioned TBiCOR (PTLB) algorithm and preconditioned TCORS (TCORS) algorithm for solving Eq (1.1), which are summarized in Algorithms 5 and 6, respectively.

Algorithm 6 PTCORS: A preconditioned tensor conjugate $\tilde{\mathcal{L}}$-orthogonal residual squared algorithm for solving (1.1)

Compute matrices $\mathbf{Q}_i(i = 1 \dots N)$ and $\tilde{\mathcal{D}} = \mathcal{D} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N \mathbf{Q}_1^{-1}$.
Replace $\mathcal{L}$ in algorithm 4 with $\tilde{\mathcal{L}}$, $\tilde{\mathcal{L}}(\mathcal{X}) = \mathcal{X} \times_1 (\mathbf{Q}_N^{-1}\mathbf{A}_1) \times_2 \cdots \times_N \mathbf{Q}_1^{-1} + \cdots + \mathcal{X} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N (\mathbf{Q}_1^{-1}\mathbf{A}_N)$.
Compute $\mathcal{R}_0 = \tilde{\mathcal{D}} - \tilde{\mathcal{L}}(\mathcal{X}_0)$; ($\mathcal{X}_0$ is an initial guess)
Set $\mathcal{R}_0^* = \tilde{\mathcal{L}}(\mathcal{R}_0)$
for n=1,2,..., until convergence **do**
  $\mathcal{U}_0 = \mathcal{R}_0$, $\widehat{\mathcal{Z}} = \tilde{\mathcal{L}}(\mathcal{U}_{n-1})$; $\rho_{n-1} = \langle \mathcal{R}_0^*, \widehat{\mathcal{Z}} \rangle$; $\mathcal{Z}_{n-1} = \mathcal{U}_{n-1}$
  if $\rho_{n-1} = 0$, stop and reset the initial tensor $\mathcal{X}_0$.
  if $n = 1$
  $\mathcal{T}_0 = \mathcal{U}_0$; $\mathcal{D}_0 = \mathcal{T}_0$; $\mathcal{C}_0 = \widehat{\mathcal{Z}}$; $\mathcal{Q}_0 = \widehat{\mathcal{Z}}$
  else
  $\beta_{n-2} = \rho_{n-1}/\rho_{n-2}$; $\mathcal{T}_{n-1} = \mathcal{U}_{n-1} + \beta_{n-2}\mathcal{H}_{n-2}$
  $\mathcal{D}_{n-1} = \mathcal{Z}_{n-1} + \beta_{n-2}\mathcal{V}_{n-2}$; $\mathcal{C}_{n-1} = \widehat{\mathcal{Z}} + \beta_{n-2}\mathcal{F}_{n-2}$
  $\mathcal{Q}_{n-1} = \mathcal{C}_{n-1} + \beta_{n-2}(\mathcal{F}_{n-2} + \beta_{n-2}\mathcal{Q}_{n-2})$
  end if
  $\widehat{\mathcal{Q}} = \tilde{\mathcal{L}}(\mathcal{Q}_{n-1})$
  $\alpha_{n-1} = \rho_{n-1}/\langle \mathcal{R}_0^*, \widehat{\mathcal{Q}} \rangle$; $\mathcal{H}_{n-1} = \mathcal{T}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1}$
  $\mathcal{V}_{n-1} = \mathcal{D}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1}$; $\mathcal{F}_{n-1} = \mathcal{C}_{n-1} - \alpha_{n-1}\widehat{\mathcal{Q}}$
  $\mathcal{X}_n = \mathcal{X}_{n-1} + \alpha_{n-1}(2\mathcal{D}_{n-1} - \alpha_{n-1}\mathcal{Q}_{n-1})$
  $\mathcal{U}_n = \mathcal{U}_{n-1} - \alpha_{n-1}(2\mathcal{C}_{n-1} - \alpha_{n-1}\widehat{\mathcal{Q}})$
end for

We only give the convergence of Algorithm 5. Similarly we can obtain the convergence of Algorithm 6, thus omit it.

Theorem 3. *Let* $\{\mathcal{R}_i^*\}$, $\{\mathcal{R}_i^*\}$, $\{\mathcal{P}_i\}$ *and* $\{\mathcal{P}_i^*\}(i = 0, 1, ..., k)$ *be the iterative sequences given by Algorithm 5, then we have*

$$\langle \tilde{\mathcal{L}}(\mathcal{R}_i), \mathcal{R}_j^* \rangle = 0 \tag{5.4}$$

and

$$\langle \tilde{\mathcal{L}}(\mathcal{P}_i), \tilde{\mathcal{L}}^T(\mathcal{P}_j^*) \rangle = 0 (i, j = 0, 1, ..., k, i \neq j). \tag{5.5}$$

Proof. The proof is very similar to those of Propositions (3) and (4) with the operator $\mathcal{L}$ being replaced by $\tilde{\mathcal{L}}$ in Algorithm 5, and is omitted. □

Theorem 4. *Assume that the Sylvester tensor equation* (1.1) *is consistent. For any initial tensor* $\mathcal{X}_0 \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$, *Algorithm 5 converges to an exact solution of* (1.1) *at most* $M = I_1 \times I_2 \times ... \times I_N$ *iteration steps in the absence of roundoff errors.*

Proof. The proof is similar to that of Theorem 1 by replacing $\mathcal{L}$ with $\tilde{\mathcal{L}}$ and is omitted. □

We can also obtain a preconditioned TLB (PTLB) by introducing the NKP preconditioner in [9] to Algorithm 2, which is listed in Algorithm 7.

Algorithm 7 PTLB: A preconditioned tensor Lanczos $\tilde{\mathcal{L}}$-biorthogonalization Algorithm for solving (1.1)

Compute matrices $\mathbf{Q}_i (i = 1 \dots N)$ and $\tilde{\mathcal{D}} = \mathcal{D} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N \mathbf{Q}_1^{-1}$.
Replace $\mathcal{L}$, $\mathcal{L}^T$ in algorithm 1, 2 with $\tilde{\mathcal{L}}$, $\tilde{\mathcal{L}}^T$, $\tilde{\mathcal{L}}(\mathcal{X}) = \mathcal{X} \times_1 (\mathbf{Q}_N^{-1}\mathbf{A}_1) \times_2 \cdots \times_N \mathbf{Q}_1^{-1} + \cdots + \mathcal{X} \times_1 \mathbf{Q}_N^{-1} \times_2 \cdots \times_N (\mathbf{Q}_1^{-1}\mathbf{A}_N)$ and $\tilde{\mathcal{L}}^T(\mathcal{X}) = \mathcal{X} \times_1 (\mathbf{Q}_N^{-1}\mathbf{A}_1)^T \times_2 \cdots \times_N (\mathbf{Q}_1^{-1})^T + \cdots + \mathcal{X} \times_1 (\mathbf{Q}_N^{-1})^T \times_2 \cdots \times_N (\mathbf{Q}_1^{-1}\mathbf{A}_N)^T$.
Choose an initial tensor $\mathcal{X}_0$ and compute $\mathcal{R}_0 = \tilde{\mathcal{D}} - \tilde{\mathcal{L}}(\mathcal{X}_0)$.
Set $\mathcal{V}_1 = \frac{\mathcal{R}_0}{\|\mathcal{R}_0\|}$, choose a tensor $\mathcal{W}_1$ such that $\langle \tilde{\mathcal{L}}(\mathcal{V}_1), \mathcal{W}_1 \rangle = 1$.
for $m = 1, 2, ...$ until convergence **do**
  Compute Lanczos $\tilde{\mathcal{L}}$-Biorthogonalization tensors $\mathcal{V}_1, ..., \mathcal{V}_m, \mathcal{W}_1, ..., \mathcal{W}_m$ and $\mathbf{T}_m$ by Algorithm 1.
  Compute $\mathbf{y}_m$ by (3.22).
end for
Compute the solution $\mathcal{X}_m$ of (1.1) by (3.18).

---

6 Numerical Experiments

In this section, we show several numerical examples to illustrate Algorithms 2–7 and compare them with CGLS in [16], MCG in [23], preconditioned GMRES (PGMRES) in [9], preconditioned BiCG (PBiCG) and preconditioned BiCR (PBiCR) in [30]. All experiments are implemented on a computer with macOS Big Sur 11.1 and 8G memory. The MATLAB R2018a (9.4.0) is used to run all examples. All algorithms are stopped when the relative error $r_k = \|\mathcal{X}_k - \mathcal{X}^*\| / \|\mathcal{X}^*\| < 10^{-10}$, where $\mathcal{X}^*$ is assumed to be an exact solution of (1.1).

Example 6.1. *In this example, we consider the Poisson equation in d-dimensional space [4]*

$$\begin{cases} -\triangle u = f, & in \quad \Omega = (0,1)^d, \\ u = 0, & on \quad \partial\Omega. \end{cases}$$

A finite difference discretization leads to the Sylvester tensor equation (1.1), *where* $\boldsymbol{A}_i \in \mathbb{R}^{10\times 10}(i = 1, 2, ..., d)$ *are*

$$\boldsymbol{A}_i = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}_{10\times 10} \tag{6.1}$$

with the mesh-width $h = \frac{1}{11}$.

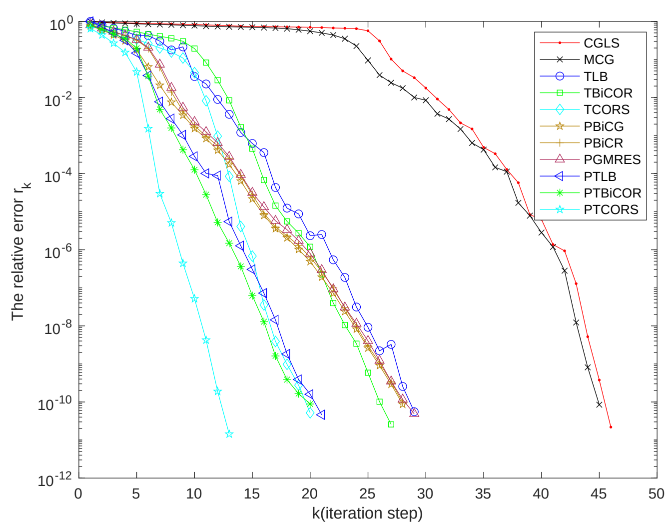

Figure 1: Plot of r_k for Example 6.1.

We set $d = 3$ and let the initial tensor $\mathcal{X}_0 = \mathcal{O}$. The right-hand side $\mathcal{D}$ of (1.1) is constructed by (1.1) with the exact solution $\mathcal{X}^*$ of (1.3) derived by the MATLAB command *tenones*$(10, 10, 10)$ in [7]. Algorithms 2-6 are used to solve (1.1) with the matrices A_i in (6.1). These methods are compared with CGLS in [16] and MCG in [23], respectively.

Figure 1 shows the convergence of the relative error r_k versus the number of iterations for all methods. From Figure 1, we can see that our preconditioned Algorithms 5-7 present better convergence than Algorithms 2-4 without preconditioning, PGMRES in [9], PBiCG and PBiCR in [30]. While Algorithms 2-4 can compare with PGMRES, PBiCG and PBiCR, and are better than CGLS [16] and MCG [23]. Algorithm 6 converges fastest among all algorithms. Algorithm 5 converges the second fastest among all algorithms.

Example 6.2. *Consider the convection-diffusion equation in [4, 29]*

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

A standard finite difference discretization on equidistant nodes combined with the second order convergent scheme [18, 20] for the convection term leads to the linear system (1.1) *with*

$$\boldsymbol{A}_n = \frac{v}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} + \frac{c_n}{4h}\begin{pmatrix} 3 & -5 & 1 & & & \\ 1 & 3 & -5 & 1 & & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{pmatrix}_{p\times p}, \tag{6.2}$$

where $n = 1, 2, ..., N$ *and the mesh-size* $h = \frac{1}{p+1}$.

We consider the case when $N = 3$ and $p = 10$. The right-hand side $\mathcal{D}$ is constructed by (1.1) with the exact solution $\mathcal{X}^*$ of (1.3) produced by the MATLAB commend *tenones*(10, 10, 10) in [7].

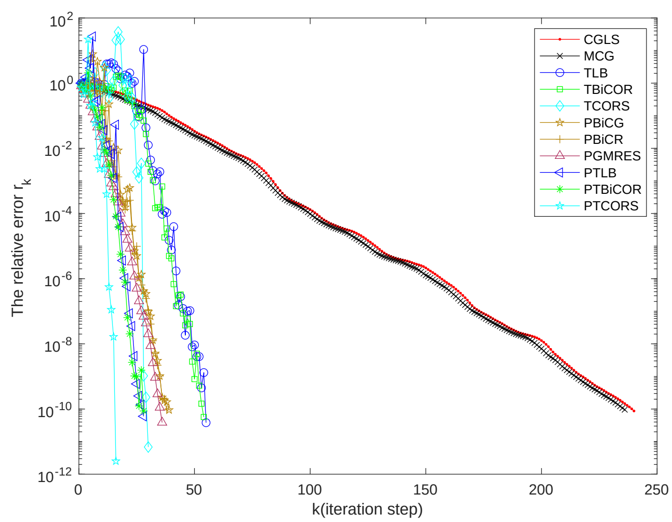

Figure 2: Plot of r_k for Example 6.2 when $v = 0.01, c_1 = 1, c_2 = 2, c_3 = 3$.

Let the initial solution $\mathcal{X}_0$ be a tensor with each element being zero. Algorithms 2-7 are used to solve (1.1) with A_i given in (6.2). These methods

Table 1: Comparison of the running time, total iteration number and the corresponding relative error for different method with different parameters when the criterion is satisfied for Example 6.2

	Methods	time(s)	TIN	r_{TIN}	Methods	time(s)	TIN	r_{TIN}
$v = 1$	CGLS	0.422117	131	9.7291e-11	PBiCG	0.407984	27	5.7115e-11
$c_1 = 1$	MCG	0.314574	130	9.1914e-11	PBiCR	0.333801	27	6.2653e-11
$c_2 = 1$					PGMRES	1.195048	26	8.4492e-11
$c_3 = 1$	TLB	0.439216	48	8.0006e-11	PTLB	0.413785	25	4.4981e-11
	TBiCOR	0.379316	48	1.2006e-11	PTBiCOR	0.283238	24	4.5384e-11
	TCORS	0.235870	32	7.9107e-11	PTCORS	0.202668	15	7.9490e-12
$v = 0.1$	CGLS	0.420019	142	9.7422e-11	PBiCG	0.327606	39	7.7596e-11
$c_1 = 1$	MCG	0.349199	141	9.1934e-11	PBiCR	0.365583	39	7.7732e-11
$c_2 = 1$					PGMRES	2.643044	37	7.6420e-11
$c_3 = 1$	TLB	0.547317	57	2.2617e-11	PTLB	0.452652	24	6.9294e-11
	TBiCOR	0.275463	51	6.4086e-11	PTBiCOR	0.240380	22	9.2513e-11
	TCORS	0.199584	30	3.6561e-11	PTCORS	0.167069	13	1.5901e-11
$v = 0.01$	CGLS	0.400485	137	7.8655e-11	PBiCG	0.345495	23	5.9009e-12
$c_1 = 1$	MCG	0.314983	136	7.9150e-11	PBiCR	0.420298	23	5.1618e-12
$c_2 = 1$					PGMRES	0.716276	20	2.9726e-11
$c_3 = 1$	TLB	0.490665	53	4.5843e-11	PTLB	0.412092	24	1.3665e-12
	TBiCOR	0.431553	49	2.6997e-11	PTBiCOR	0.271265	22	6.3548e-11
	TCORS	0.289261	29	4.6591e-11	PTCORS	0.182132	14	1.4634e-11
$v = 1$	CGLS	0.619692	231	9.8161e-11	PBiCG	0.302540	24	9.6344e-11
$c_1 = 1$	MCG	0.479290	228	9.5604e-11	PBiCR	0.332252	25	1.8973e-11
$c_2 = 2$					PGMRES	0.969060	23	8.3597e-11
$c_3 = 3$	TLB	0.574761	60	6.1992e-11	PTLB	0.463983	25	6.7805e-11
	TBiCOR	0.359788	59	6.2755e-11	PTBiCOR	0.294740	25	3.1182e-11
	TCORS	0.259970	33	9.4662e-11	PTCORS	0.204611	15	1.2034e-11
$v = 0.1$	CGLS	0.661334	234	8.5041e-11	PBiCG	0.321293	26	7.8704e-12
$c_1 = 1$	MCG	0.508388	231	9.1198e-11	PBiCR	0.435445	24	7.5118e-11
$c_2 = 2$					PGMRES	0.885821	22	7.6295e-11
$c_3 = 3$	TLB	0.500843	53	7.0480e-12	PTLB	0.360964	22	1.1481e-12
	TBiCOR	0.275404	48	7.3762e-11	PTBiCOR	0.244606	20	5.4227e-11
	TCORS	0.229283	28	1.0059e-12	PTCORS	0.146302	12	1.2266e-11
$v = 0.01$	CGLS	0.658699	240	8.7261e-11	PBiCG	0.321684	39	9.3669e-11
$c_1 = 1$	MCG	0.517902	236	9.5783e-11	PBiCR	0.356874	38	9.4120e-11
$c_2 = 2$					PGMRES	2.472089	36	3.8573e-11
$c_3 = 3$	TLB	0.531822	55	6.9508e-11	PTLB	0.491639	29	8.5332e-12
	TBiCOR	0.316190	54	5.6945e-11	PTBiCOR	0.268896	28	8.7378e-11
	TCORS	0.203383	30	6.8170e-12	PTCORS	0.164989	16	2.5240e-12

are compared with CGLS [16], MCG [23], PGMRES in [9], PBiCG and PBiCR in [30].

Table 1 displays the running time, total iteration number (TIN) and relative error of different method with different parameters $v = 1, 0.1, 0.001$ and c_i. Figure 2 shows the convergence of the relative error r_k for each method with the parameters $v = 0.01$, $c_1 = 1$, $c_2 = 2$ and $c_3 = 3$.

Table 1 shows that, when the stop criterion is satisfied, preconditioned Algorithms 5-6 require less CPU time and iterations than Algorithms 2-4, PGMRES, PBiCG and PBiCR. In most cases Algorithms 2-4 requires much less CPU time but more iterations than PGMRES, PBiCG and PBiCR, and are better than CGLS [16] and MCG [23] both in CPU time and the number of iterations. Algorithm 6 requires the minimal CPU time and iterations among all methods. Figure 2 shows similar results to that in Figure 1.

Example 6.3. *We consider the Sylvester tensor equation* (1.1) *with the coefficient matrices* $\boldsymbol{A}_i, i = 1, 2, 3$*, which comes from the discretization of the operator*

$$Lu := \triangle u - e^{xy}\frac{\partial u}{\partial x} + sin(xy)\frac{\partial u}{\partial y} + y^2 - x^2 \tag{6.3}$$

on the unit square $[0, 1] \times [0, 1]$ *with homogeneous Dirichlet boundary conditions. We use the MATLAB command fdm_2d_matrix in the Lyapack package [25] to generate matrices* A_i*:*

$$\boldsymbol{A}_i = fdm\_2d\_matrix(I_i, e^{xy}, sin(xy), y^2 - x^2), , \tag{6.4}$$

where $I_i = 1 + i, i = 1, 2, 3$. *We construct* $\mathcal{D}$ *by* (1.1) *with the exact solution* $\mathcal{X}^*$ *of* (1.3) *produced by the MATLAB commend tenones*$(4, 9, 16)$ *in [7].*

The initial solution $\mathcal{X}_0$ is selected as zero tensor. Algorithms 2-6 are used to solve (1.1) with $\mathbf{A}_i$ given in (6.4). These methods are compared with CGLS [16], MCG [23], PGMRES in [9], PBiCG and PBiCR in [30]. Figure 3 shows that Algorithm 6 converges fastest among all methods and Algorithm 6 converges the second fastest among all methods, which are very similar to those in Figures 1 and 2.

7 Conclusion

This paper first presents a tensor Lanczos $\mathcal{L}$-Biorthogonalization (TLB) algorithm for solving the Sylvester tensor equation (1.1) based on the Lanczos $\mathcal{L}$-Biorthogonalization procedure. Then two improved methods based on the TLB algorithm are developed. The one is the biconjugate $\mathcal{L}$-orthogonal residual algorithm in tensor form (TBiCOR). The other is the conjugate $\mathcal{L}$-orthogonal residual squared algorithm in tensor form (TCORS). The preconditioner based on the nearest Kronecker product (NKP) are used to accelerate the TBiCOR and TCORS algorithms, thus we present preconditioned a preconditioned TBiCOR method and a preconditioned TCORS method. The convergence of these proposed algorithms are proved. Numerical examples show the advantage of the preconditioned TBiCOR and TCORS methods.

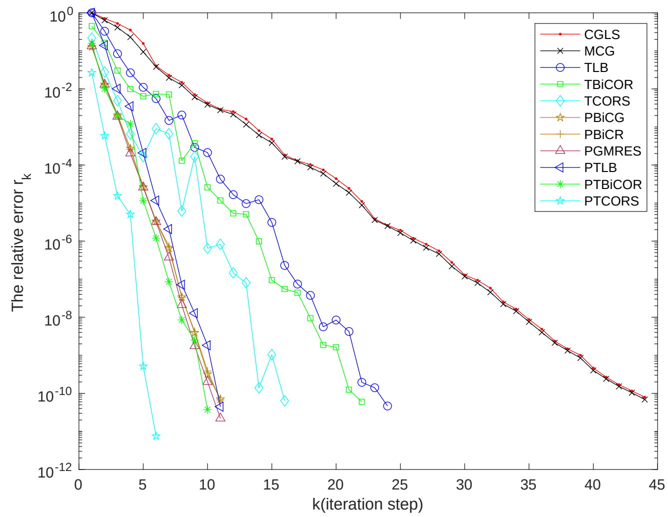

Figure 3: Plot of the relative error r_k for Example 6.3.

8 Acknowledgments

The authors would like to thank the referees for their helpful comments which form the present version of this paper. The preconditioned methods are added according to one comment. Research by G.H. was supported in part by Application Fundamentals Foundation of STD of Sichuan (2020YJ0366) and Key Laboratory of bridge nondestructive testing and engineering calculation Open fund projects (2020QZJ03), and research by F.Y. was partially supported by NNSF (11501392) and SUSE (2019RC09).

References

[1] F.A. Beik, F. Movahed, S. Ahmadi-Asl, On the Krylov subspace methods based on tensor format for positive definite Sylvester tensor equations, Numer. Linear Algebr. 23 (2016) 444-466.

[2] M. August, M.C. Banuls, T. Huckle, On the approximation of functionals of very large hermitian matrices represented as matrix product operators, Electron. T. Numer. Ana. 46 (2017) 215-232.

[3] Z.Z. Bai, G. Golub, M. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, SIAM J. Matrix Anal. Appl. 24 (2002) 603-626.

[4] J. Ballani, L. Grasedyck, A projection method to solve linear systems in tensor format, Numer. Linear Algebr. 20 (2013) 27-43.

[5] F.A. Beik, M. Najafi-Kalyani, L. Reiche, Iterative Tikhonov regularization of tensor equations based on the Arnoldi process and some of its generalizations, Appl. Numer. Math. 151 (2020) 425-447.

[6] A.H. Bentbib, S. El-Halouy, E.M. Sadek, Krylov subspace projection method for Sylvester tensor equation with low rank right-hand side, Numer. Alg. 84 (2020) 1411-1430.

[7] B.W. Bader, T.G. Kolda, Matlab tensor toolbox, Version 2.5, Available online at http://www.sandia.gov/tgkolda/TensorToolbox/, 2012.

[8] D. Calvetti, L. Reichel, Application of ADI iterative methods to the restoration of noisy images, SIAM J. Matrix Anal. Appl. 17 (1) (1996) 165-186.

[9] Z. Chen, L. Lu, A projection method and Kronecker product preconditioner for solving Sylvester tensor equations, SCI. China Ser. A. Math. 55 (2012) 1281-1292.

[10] Z. Chen, L. Lu, A Gradient Based Iterative Solutions for Sylvester Tensor Equations, Math. Probl. Eng. (2013) 1-7.

[11] B. Carpentieri, Y.F. Jing, T.Z. Huang, The BiCOR and CORS iterative algorithms for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 33 (2011) 3020-3036.

[12] F. Ding, T. Chen, Gradient based iterative algorithms for solving a class of matrix equations, IEEE T. Automat. Contr. 50 (2005) 1216-1221.

[13] F. Ding, T. Chen, Iterative least-squares solutions of coupled Sylvester matrix equations, Syst. Contr. Lett. 54 (2005) 95-107.

[14] G. Golub, S. Nash, C. Van Loan, A Hessenberg-Schur method for the problem $AX + XB = C$, IEEE T. Automat. Contr. 24 (1979) 909-913.

[15] M. Heyouni, F. Saberi-Movahed, A. Tajaddini, A tensor format for the generalized Hessenberg method for solving Sylvester tensor equations, J. Comput. Appl. Math. 377 (2020) 112878.

[16] B. Huang, C. Ma, An iterative algorithm to solve the generalized Sylvester tensor equations, Linear Multilinear A. 68 (2018) 1175-1200.

[17] T.G. Kolda, B.W. Bader, Tensor Decompositions and Applications, SIAM Rev. 51 (2009) 455-500.

[18] D. Kressner, C. Tobler, Krylov subspace methods for linear systems with tensor product structure, SIAM J. Matrix Anal. Appl. 31 (2010) 1688-1714.

[19] D. Kressner, C. Tobler, Low-rank tensor Krylov subspace methods for parametrized linear systems, SIAM J. Matrix Anal. Appl. 32 (2011) 1288-1316.

[20] L. Grasedyck, Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure, Computing 72 (2004) 247-265.

[21] B.W. Li, Y.S. Sun, D.W. Zhang, Chebyshev collocation spectral methods for coupled radiation and conduction in a concentric spherical participating medium, J. Heat Trans. 131 (2009) 1-9.

[22] N. Li, C. Navasca, C. Glemn, Iterative methods for symmetric outer product tensor decomposition, Electron. T. Numer. Ana. 44 (2015) 124-139.

[23] C. Lv, C. Ma, A modified CG algorithm for solving generalized coupled Sylvester tensor equations, Appl. Math. Comput. 365 (2020) 124699.

[24] M. Najafi-Kalyani, F.A. Beik, K. Jbilou, On global iterative schemes based on Hessenberg process for (ill-posed) Sylvester tensor equations, J. Comput. Appl. Math. 373 (2020) 112216.

[25] T. Penzl, Lyapack, A MATLAB toolbox for large Lyapunov and Riccati equations, model reduction problems,and linear-quadratic optimal control problems, Available online at https://www.tu-chemnitz.de/sfb393/lyapack/, 2000.

[26] Y. Saad, Iterative methods for sparse linear systems, Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[27] X.H. Shi, Y.M. Wei, S.Y. Ling, Backward error and perturbation bounds for high order Sylvester tensor equation, Linear Multilinear A. 61 (2013) 1436-1446.

[28] C.F. Van Loan, N. Pitsianis, Approximation with Kronecker products, In Proc.: Linear Algebra for Large Scale and Real-Time Applications, Kluwer Publications 232 (1993) 293-314.

[29] H. Xiang, L. Grigori, Kronecker product approximation preconditioners for convection-diffusion model problems, Numer. Linear Algebr. 17 (2010) 691-712.

[30] X.F. Zhang, Q.W. Wang, Developing iterative algorithms to solve Sylvester tensor equations, Appl. Math. Comput. 409 (2021) 126403.