TAIWANESE JOURNAL OF MATHEMATICS
Vol. 26, No. 3, pp. 501–519, June 2022
DOI: 10.11650/tjm/220103

On RGI Algorithms for Solving Sylvester Tensor Equations

Xin-Fang Zhang and Qing-Wen Wang*

Abstract. This paper is concerned with studying the relaxed gradient-based iterative method based on tensor format to solve the Sylvester tensor equation. From the information given by the previous steps, we further develop a modified relaxed gradient-based iterative method which converges faster than the method above. Under some suitable conditions, we prove that the introduced methods are convergent to the unique solution for any initial tensor. At last, we provide some numerical examples to show that our methods perform much better than the GI algorithm proposed by Chen and Lu (Math. Probl. Eng. 2013) both in the number of iteration steps and the elapsed CPU time.

1. Introduction

In this paper, we investigate the iterative solutions to the following Sylvester tensor equation

$$\mathscr{X} \times_1 A_1 + \mathscr{X} \times_2 A_2 + \mathscr{X} \times_3 A_3 = \mathscr{B}, \tag{1.1}$$

where $A_1 \in \mathbb{R}^{N_1 \times N_1}$, $A_2 \in \mathbb{R}^{N_2 \times N_2}$, $A_3 \in \mathbb{R}^{N_3 \times N_3}$, $\mathscr{B} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ are given, and the unknown tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is required to be determined. The details of the operators $\times_i$ $(i = 1, 2, 3)$ will be described in Section 2. If $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2}$, that is a matrix X, (1.1) can reduce to the following Sylvester matrix equation

$$A_1 X + X A_2^T = B. \tag{1.2}$$

It was not merely applied to control theory [12, 17, 18, 34, 49], also extensively penetrated into model reduction [4], image processing [8], quantum information [43], disturbance decoupling problem [11] and system identification [15, 33, 44]. Existing methods for solving (1.2) are classified into two categories: direct methods and iterative methods. From the Kronecker product or the Hessenberg–Schur form, some direct methods [3, 20–24, 41] have

Received February 2, 2021; Accepted January 16, 2022.
Communicated by Eric Chung.
2020 *Mathematics Subject Classification.* 15A69, 65F10.
Key words and phrases. tensor format, relaxed gradient-based iterative method, modified relaxed gradient-based iterative method, Sylvester tensor equation.
This research was supported by National Natural Science Foundation of China [grant number 11971294].
*Corresponding author.

been presented to solve a large linear system. However, the above methods are difficult to achieve in actual implementations as the dimension of matrices increases. Inspired by these issues, some researchers are desire to utilize the iterative methods to solve the matrix equation (1.2). For example, Bai [2] derived a Hermitian and skew-Hermitian splitting (HSS) iteration method to solve Sylvester matrix equation with non-Hermitian and positive definite/semi-definite matrices. Zhou et al. [50] introduced a modified version to improve the performance of HSS iteration. In [28, 29, 35], some Krylov subspace methods for obtaining an approximate solution of (1.2) have been proposed. From the hierarchical identification principle [14,16], some efficient gradient-based iterative methods for solving Sylvester matrix equation were proposed in [13,19]. Benner et al. [5] presented a generalization ADI method based on the Cholesky factor to consider the matrix equation (1.2). Moreover, Niu et al. [38] developed a relaxed gradient-based iterative (RGI) method to investigate the numerical solutions of (1.2). By using the information provided by the previous steps, an accelerated gradient-based iterative method [46] was proposed for searching the iterative solutions of (1.2), which is a promising method.

In recent years, some algorithms and theoretical results involved with (1.1) have been well developed. Chen and Lu [10] extended the gradient-based iterative (GI) algorithm and its modification version to find an approximate solution of (1.1). In [9], the GMRES method in its tensor format was introduced to solve the Sylvester tensor equation. Moreover, a nearest Kronecker preconditioner was also proposed for accelerating the convergence of the method mentioned. After that, a residual norm steepest descent method [6] was proposed for solving the Sylvester tensor equation. Beik et al. [7] derived the conjugate gradient and nested conjugate gradient methods in their tensor forms for searching the solutions of (1.1). By using the bidiagonal process, Karimi and Dehghan [30] developed a global least squares method based on tensor form to approximate the solution of Sylvester tensor equation. The convergence results of this method were also established. In [47], Xu and Wang extended the bi-conjugate gradient and bi-conjugate residual methods for solving nonsymmetric linear system to solve Stein tensor equation. Wang and Xu [42] developed a class of iterative algorithms for solving some tensor equation under the Einstein product. Zhang and Wang [48] presented the tensor forms of bi-conjugate gradient and bi-conjugate residual methods for solving Sylvester tensor equation. Xie and Wang [45] investigated the existence of the reducible solution to a quaternion tensor equation. Lv and Ma [37] proposed a modified conjugate gradient algorithm for solving the generalized coupled Sylvester tensor equations. Also they [36] established a Levenberg–Marquardt method for solving semi-symmetric tensor equations. Huang and Ma [25–27] developed some Kryolv subspace methods for solving the generalized Sylvester tensor equations.

The purpose of this paper is to extend the relaxed gradient-based iterative method proposed by Niu et al. [38] to solve the Sylvester tensor equation. By introducing two relaxation parameters, we develop a relaxed gradient-based iterative algorithm to consider the solution of (1.1). By taking fully advantage of the information given by the previous steps, we further develop a tensor form of modified relaxed gradient-based iterative algorithm which can greatly improve the performance of the above method. The detailed analysis of the convergence for the introduced algorithms is also established.

We organize this paper as follows. In Section 2, we initially recall some preliminary definitions and conclusions related to tensors. In Section 3, we propose a relaxed gradient-based iterative algorithm and its modification version in their tensor forms for solving the Sylvester tensor equation, respectively. Moreover, we also present the convergence analysis of the introduced methods. In Section 4, we provide several examples to demonstrate that our algorithms outperform the GI algorithm both in the elapsed time and the number of iteration steps. Finally, we give some concluding remarks in Section 5.

2. Preliminaries

In this part, we will present several useful definitions and results which are used in the sequel. Matrices are written as capital letters, e.g., A, tensors are written as Euler script letters, e.g., $\mathscr{A}$. The symbol I^n denotes the $n \times n$ identity matrix. $\mathbb{R}$ denotes the real number field. For any given matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, operator $V_c(A)$ stacks the columns of A to a vector that could be described as

$$V_c(A) = [a_{11}\ a_{21}\ \cdots\ a_{m1}\ a_{12}\ a_{22}\ \cdots\ a_{m2}\ \cdots\ a_{1n}\ a_{2n}\ \cdots\ a_{mn}].$$

An order m dimension $N_1 \times N_2 \times \cdots \times N_m$ tensor $\mathscr{X}$ over $\mathbb{R}$ is a multidimensional array with its entries given by

$$\mathscr{X} = (x_{i_1 i_2 \cdots i_m}), \quad x_{i_1 i_2 \cdots i_m} \in \mathbb{R},\ 1 \leq i_j \leq N_j,\ 1 \leq j \leq m.$$

Let $\mathbb{R}^{N_1 \times N_2 \times \cdots \times N_m}$ be the set of all these tensors over $\mathbb{R}$.

Definition 2.1. [31] If $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_m}$ and $A \in \mathbb{R}^{J \times N_n}$, then their n-mode product defined as

$$(\mathscr{X} \times_n A)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_m} = \sum_{i_n=1}^{N_n} x_{i_1 i_2 \cdots i_m} a_{j i_n}$$

is an $N_1 \times \cdots \times N_{n-1} \times J \times N_{n+1} \times \cdots \times N_m$ tensor.

Definition 2.2. [40] If $\mathscr{X} \in \mathbb{R}^{N_1 \times \cdots \times N_m}$, then its mode-$n$ matricization $\mathscr{X}_{(n)}$ defined as

$$(\mathscr{X}_{(n)})_{i_n j} = x_{i_1 i_2 \cdots i_m}, \quad j = 1 + \sum_{k=1, k \neq n}^{m} (i_k - 1) J_k, \quad J_k = \prod_{p=1, p \neq n}^{k-1} N_p$$

is an $N_n \times N_1 \cdots N_{n-1} N_{n+1} \cdots N_m$ matrix.

For any given tensor $\mathscr{X} \in \mathbb{R}^{N_1\times\cdots\times N_m}$, operator $V_c(\mathscr{X})$ is the column stacking form of the corresponding matrix $\mathscr{X}_{(1)}$. Then the inner product of $\mathscr{X}$ and $\mathscr{Y}$ over $\mathbb{R}^{N_1\times\cdots\times N_m}$ can be defined as

$$\langle \mathscr{X}, \mathscr{Y}\rangle = V_c(\mathscr{X})^T V_c(\mathscr{Y}) = \sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\cdots\sum_{i_m=1}^{N_m} x_{i_1i_2\cdots i_m} y_{i_1i_2\cdots i_m}.$$

Particularly, the induced Frobenious norm of $\mathscr{X}$ is

$$\|\mathscr{X}\|^2 = \sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\cdots\sum_{i_m=1}^{N_m} x^2_{i_1i_2\cdots i_m}. \tag{2.1}$$

From the above definitions, we have the following results.

Proposition 2.3. [32,39] *Let $\mathscr{X}$ and $\mathscr{Y}$ be the tensors over $\mathbb{R}^{N_1\times\cdots N_n\times\cdots\times N_m}$.*

(1) *If $\mathscr{Y} = \mathscr{X} \times_1 A_1 \times_2 A_2 \times_3 \cdots \times_m A_m$ holds, then*

$$\mathscr{Y}_{(n)} = A_n \mathscr{X}_{(n)} (A_m \otimes \cdots \otimes A_{n+1} \otimes A_{n-1} \otimes \cdots \otimes A_1)^T,$$

where the matrices $A_i \in \mathbb{R}^{N_i\times N_i}$, $1 \le i \le m$.

(2) *For any given matrices A_i and A_j, we have*

$$\mathscr{X} \times_i A_i \times_j A_j = \begin{cases} \mathscr{X} \times_i (A_jA_i) & \text{if } i = j, \\ \mathscr{X} \times_j A_j \times_i A_i & \text{if } i \neq j. \end{cases}$$

(3) *For any given matrix A_i, we have*

$$\langle \mathscr{X}, \mathscr{Y} \times_i A_i\rangle = \langle \mathscr{X} \times_i A_i^T, \mathscr{Y}\rangle$$

for $1 \le i \le m$.

(4) $2\langle \mathscr{X}, \mathscr{Y}\rangle \le \|\mathscr{X}\|^2 + \|\mathscr{Y}\|^2$, $\langle \mathscr{X}, \mathscr{Y}\rangle \le \|\mathscr{X}\|\|\mathscr{Y}\|$;

(5) $\|\mathscr{X} \times_i A_i\| \le \|\mathscr{X}\|\|A_i\|_2$, *where $\|A_i\|_2$ denotes the spectral norm of matrix A_i.*

By using Proposition 2.3, we can easily obtain that (1.1) is equivalent to the following linear system of equations

$$(I^{N_3} \otimes I^{N_2} \otimes A_1 + I^{N_3} \otimes A_2 \otimes I^{N_l} + A_3 \otimes I^{N_2} \otimes I^{N_1})V_c(\mathscr{X}) = V_c(\mathscr{B}). \tag{2.2}$$

Then Chen [10] gave the following theorem to show the uniqueness of the solution of the tensor equation (1.1).

Lemma 2.4. [10] *Let $\lambda_p(A_1)$, $\lambda_q(A_2)$ and $\lambda_r(A_3)$ be the eigenvalues of A_1, A_2 and A_3, respectively. Then $\mathscr{X}^*$ is a unique solution of* (1.1) *if and only if $\lambda_p(A_1) + \lambda_q(A_2) + \lambda_r(A_3) \neq 0$ for any p, q, and r.*

From the hierarchical identification principle, Chen and Lu [10] proposed a gradient-based iterative (GI) algorithm for searching the solutions of (1.1).

Algorithm 2.1 Gradient-based iterative algorithm for solving (1.1).

Input: Given an initial guess $\mathscr{X}^0$.

Output: $\mathscr{X}$.

For $k = 0, 1, 2, \ldots$ until converges

1: $\mathscr{R}^k = \mathscr{B} - \mathscr{X}^k \times_1 A_1 - \mathscr{X}^k \times_2 A_2 - \mathscr{X}^k \times_3 A_3$,

2: $\mathscr{X}_1^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_1 A_1^T$,

3: $\mathscr{X}_2^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_2 A_2^T$,

4: $\mathscr{X}_3^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_3 A_3^T$,

5: $\mathscr{X}^{k+1} = \frac{(\mathscr{X}_1^{k+1} + \mathscr{X}_2^{k+1} + \mathscr{X}_3^{k+1})}{3}$.

The convergence analysis of GI algorithm was investigated in [10].

Lemma 2.5. [10] *Suppose that* (1.1) *has a unique solution $\mathscr{X}^*$. Then the iterative sequence $\{\mathscr{X}^k\}$ derived by GI algorithm is convergent to $\mathscr{X}^*$ if and only if*

$$0 < \gamma < \frac{2}{\|A_1\|_2^2 + \|A_2\|_2^2 + \|A_3\|_2^2}.$$

Algorithm 2.2 Modified gradient-based iterative algorithm for solving (1.1).

Input: Given three initial.tensors $\mathscr{X}_1^0$, $\mathscr{X}_2^0$, $\mathscr{X}_3^0$.

Output: $\mathscr{X}$.

For $k = 0, 1, 2, \ldots$ until converges

1: $\mathscr{R}^k = \mathscr{B} - \mathscr{X}^k \times_1 A_1 - \mathscr{X}^k \times_2 A_2 - \mathscr{X}^k \times_3 A_3$,

2: $\mathscr{X}_1^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_1 A_1^T$,

3: $\mathscr{X}^k = \frac{\mathscr{X}_1^{k+1} + \mathscr{X}_2^k + \mathscr{X}_3^k}{3}$,

4: $\mathscr{X}_2^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_2 A_2^T$,

5: $\mathscr{X}^k = \frac{\mathscr{X}_1^{k+1} + \mathscr{X}_2^{k+1} + \mathscr{X}_3^k}{3}$,

6: $\mathscr{X}_3^{k+1} = \mathscr{X}^k + \gamma \mathscr{R}^k \times_3 A_3^T$,

7: $\mathscr{X}^{k+1} = \frac{\mathscr{X}_1^{k+1} + \mathscr{X}_2^{k+1} + \mathscr{X}_3^{k+1}}{3}$.

According to the information provided in the previous steps, Chen and Lu [10] further derived a modified gradient-based iterative (MGI) algorithm to accelerate the convergence rate of GI algorithm. The convergence results of MGI algorithm were also established in [10].

Lemma 2.6. [10] *Suppose that* (1.1) *has a unique solution* $\mathscr{X}^*$. *Then the iterative sequence* $\{\mathscr{X}^k\}$ *derived by MGI algorithm is convergent to* $\mathscr{X}^*$ *if and only if*

$$0 < \gamma < \min\left\{\frac{1}{\|A_1\|_2^2}, \frac{1}{\|A_2\|_2^2}, \frac{1}{\|A_3\|_2^2}\right\}.$$

Compared with GI algorithm, it is easy to see that the RGI algorithm has better performance in [38]. Therefore, we propose a relaxed gradient-based iterative algorithm based on tenor form for studying the solutions of (1.1). Unless otherwise specified, we always suppose that (1.1) has a unique solution $\mathscr{X}^*$ in the rest of this paper.

3. Tensor forms of relaxed gradient-based algorithm

In this part, we introduce two parameters to propose a relaxed gradient-based iterative algorithm based on tensor format (RGI_BTF) for solving the tensor equation (1.1). By using the efficient information given by the previous steps, a modified relaxed gradient-based iterative algorithm in its tensor form (MRGI_BTF) is also proposed. Now we present the details of RGI_BTF algorithm as follows.

Define three residual tensors:

$$\begin{aligned}
\mathscr{W}_1 &= \mathscr{B} - \mathscr{X} \times_3 A_3 - \mathscr{X} \times_2 A_2,\\
\mathscr{W}_2 &= \mathscr{B} - \mathscr{X} \times_3 A_3 - \mathscr{X} \times_1 A_1,\\
\mathscr{W}_3 &= \mathscr{B} - \mathscr{X} \times_2 A_2 - \mathscr{X} \times_1 A_1.
\end{aligned} \tag{3.1}$$

It follows from (1.1) that we can attain three fictitious subsystems, respectively,

$$\mathscr{X} \times_1 A_1 = \mathscr{W}_1, \quad \mathscr{X} \times_2 A_2 = \mathscr{W}_2, \quad \mathscr{X} \times_3 A_3 = \mathscr{W}_3.$$

If $\mathscr{X}^k$ is the k-th iterative solution of (1.1), then the relaxed recursive equations are given by

$$\begin{aligned}
\mathscr{X}_1^{k+1} &= \mathscr{X}_1^k + (\alpha-\beta)\beta\gamma(\mathscr{W}_1 - \mathscr{X}_1^k \times_1 A_1) \times_1 A_1^T,\\
\mathscr{X}_2^{k+1} &= \mathscr{X}_2^k + (1-\alpha)\beta\gamma(\mathscr{W}_2 - \mathscr{X}_2^k \times_2 A_2) \times_2 A_2^T,\\
\mathscr{X}_3^{k+1} &= \mathscr{X}_3^k + (1-\alpha)(\alpha-\beta)\gamma(\mathscr{W}_3 - \mathscr{X}_3^k \times_3 A_3) \times_3 A_3^T,
\end{aligned} \tag{3.2}$$

where γ is the step length, α and β are the relaxation parameters satisfying $0 < \beta < \alpha < 1$. We substitute (3.1) into (3.2) such that

$$\begin{aligned}
\mathscr{X}_1^{k+1} &= \mathscr{X}_1^k + (\alpha-\beta)\beta\gamma(\mathscr{B} - \mathscr{X} \times_3 A_3 - \mathscr{X} \times_2 A_2 - \mathscr{X}_1^k \times_1 A_1) \times_1 A_1^T,\\
\mathscr{X}_2^{k+1} &= \mathscr{X}_2^k + (1-\alpha)\beta\gamma(\mathscr{B} - \mathscr{X} \times_3 A_3 - \mathscr{X} \times_1 A_1 - \mathscr{X}_2^k \times_2 A_2) \times_2 A_2^T,\\
\mathscr{X}_3^{k+1} &= \mathscr{X}_3^k + (1-\alpha)(\alpha-\beta)\gamma(\mathscr{B} - \mathscr{X} \times_2 A_2 - \mathscr{X} \times_1 A_1 - \mathscr{X}_3^k \times_3 A_3) \times_3 A_3^T.
\end{aligned}$$

By replacing the unknown tensor $\mathscr{X}$ with $\mathscr{X}^k$, we have

$$
\begin{aligned}
\mathscr{X}_1^{k+1} &= \mathscr{X}_1^k + (\alpha-\beta)\beta\gamma\mathscr{R}_1^k \times_1 A_1^T,\\
\mathscr{X}_2^{k+1} &= \mathscr{X}_2^k + (1-\alpha)\beta\gamma\mathscr{R}_2^k \times_2 A_2^T,\\
\mathscr{X}_3^{k+1} &= \mathscr{X}_3^k + (1-\alpha)(\alpha-\beta)\gamma\mathscr{R}_3^k \times_3 A_3^T,
\end{aligned} \tag{3.3}
$$

where $\mathscr{R}_i^k = \mathscr{B} - \mathscr{X}_i^k \times_1 A_1 - \mathscr{X}_i^k \times_2 A_2 - \mathscr{X}_i^k \times_3 A_3$, $i = 1, 2, 3$. Since the two relaxation parameters have been introduced, we update $\mathscr{X}^{k+1}$ as follows:

$$
\mathscr{X}^{k+1} = (1-\alpha)\mathscr{X}_1^{k+1} + (\alpha-\beta)\mathscr{X}_2^{k+1} + \beta\mathscr{X}_3^{k+1}. \tag{3.4}
$$

In fact, the above formula is based on the idea of MAOR iteration for solving linear complementarity problem [1]. It is easy to see that the performance of MAOR algorithm depends on the chosen parameters. However, the optimal relaxation parameters of RGI_BTF algorithm are very difficult to determine. We will study how to choose these parameters in the future work.

In what follows, we present the details of the RGI_BTF algorithm for solving (1.1).

Algorithm 3.1 RGI algorithm based on tensor format for solving (1.1).

Input: Given an initial guess $\mathscr{X}^0$.

Output: $\mathscr{X}$.

For $k = 0, 1, 2, \dots$ until converges

1: $\mathscr{R}^k = \mathscr{B} - \mathscr{X}^k \times_1 A_1 - \mathscr{X}^k \times_2 A_2 - \mathscr{X}^k \times_3 A_3$,

2: $\mathscr{X}_1^{k+1} = \mathscr{X}^k + (\alpha-\beta)\beta\gamma\mathscr{R}^k \times_1 A_1^T$,

3: $\mathscr{X}_2^{k+1} = \mathscr{X}^k + (1-\alpha)\beta\gamma\mathscr{R}^k \times_2 A_2^T$,

4: $\mathscr{X}_3^{k+1} = \mathscr{X}^k + (1-\alpha)(\alpha-\beta)\gamma\mathscr{R}^k \times_3 A_3^T$,

5: $\mathscr{X}^{k+1} = (1-\alpha)\mathscr{X}_1^{k+1} + (\alpha-\beta)\mathscr{X}_2^{k+1} + \beta\mathscr{X}_3^{k+1}$.

Remark 3.1.

(1) If $\alpha = 2/3$ and $\beta = 1/3$, then (3.4) is equivalent to the one proposed in [10].

(2) If $\mathscr{X} \in \mathbb{R}^{N_1\times N_2}$ and $\alpha = \beta$ or $\beta = 0$, then the proposed algorithm is equivalent to Algorithm 2.2 derived in [38].

(3) Algorithm 3.1 cannot reduce to Algorithm 2.1 because the formulas (3.3) are different from that of [10].

Now we discuss the convergence conditions of the algorithm mentioned. The following theorem present a sufficient condition for the convergence of RGI_BTF algorithm.

Theorem 3.2. *Let $\{\mathscr{X}^k\}$ be the iterative sequence given by Algorithm* 3.1*. For any initial tensor, if the step length γ satisfies*

$$0<\gamma<\frac{2}{(\alpha-\beta)\beta\|A_1\|_2^2+(1-\alpha)\beta\|A_2\|_2^2+(1-\alpha)(\alpha-\beta)\|A_3\|_2^2}, \quad 0<\beta<\alpha<1,$$

then the iterative sequence $\{\mathscr{X}^k\}$ is convergent to $\mathscr{X}^*$.

Proof. Firstly, we define the error tensors as:

$$\overrightarrow{\mathscr{X}}_i^k=\mathscr{X}_i^k-\mathscr{X}^*, \quad i=1,2,3$$

and

$$\mathscr{P}^k=\overrightarrow{\mathscr{X}}^k\times_1 A_1, \quad \mathscr{Q}^k=\overrightarrow{\mathscr{X}}^k\times_2 A_2, \quad \mathscr{U}^k=\overrightarrow{\mathscr{X}}^k\times_3 A_3.$$

It follows from Algorithm 3.1 that we have

$$\begin{aligned}
\overrightarrow{\mathscr{X}}_1^{k+1}&=\overrightarrow{\mathscr{X}}^k-(\alpha-\beta)\beta\gamma\mathscr{V}^k\times_1 A_1^T,\\
\overrightarrow{\mathscr{X}}_2^{k+1}&=\overrightarrow{\mathscr{X}}^k-(1-\alpha)\beta\gamma\mathscr{V}^k\times_2 A_2^T,\\
\overrightarrow{\mathscr{X}}_3^{k+1}&=\overrightarrow{\mathscr{X}}^k-(1-\alpha)(\alpha-\beta)\gamma\mathscr{V}^k\times_3 A_3^T,
\end{aligned}$$

where $\mathscr{V}^k=\mathscr{P}^k+\mathscr{Q}^k+\mathscr{U}^k$. Let

$$\overrightarrow{\mathscr{X}}^{k+1}=\mathscr{X}^{k+1}-\mathscr{X}^*. \tag{3.5}$$

By using (2.1) and (3.5), we have

$$\begin{aligned}
&\|\overrightarrow{\mathscr{X}}^{k+1}\|^2\\
&=\|(1-\alpha)\mathscr{X}_1^{k+1}+(\alpha-\beta)\mathscr{X}_2^{k+1}+\beta\mathscr{X}_3^{k+1}-\mathscr{X}^*\|^2\\
&=\|(1-\alpha)\overrightarrow{\mathscr{X}}_1^{k+1}+(\alpha-\beta)\overrightarrow{\mathscr{X}}_2^{k+1}+\beta\overrightarrow{\mathscr{X}}_3^{k+1}\|^2\\
&=(1-\alpha)^2\|\overrightarrow{\mathscr{X}}_1^{k+1}\|^2+(\alpha-\beta)^2\|\overrightarrow{\mathscr{X}}_2^{k+1}\|^2+\beta^2\|\overrightarrow{\mathscr{X}}_3^{k+1}\|^2\\
&\quad+2(1-\alpha)(\alpha-\beta)\langle\overrightarrow{\mathscr{X}}_1^{k+1},\overrightarrow{\mathscr{X}}_2^{k+1}\rangle+2(1-\alpha)\beta\langle\overrightarrow{\mathscr{X}}_1^{k+1},\overrightarrow{\mathscr{X}}_3^{k+1}\rangle\\
&\quad+2(\alpha-\beta)\beta\langle\overrightarrow{\mathscr{X}}_2^{k+1},\overrightarrow{\mathscr{X}}_3^{k+1}\rangle\\
&\le(1-\alpha)^2\|\overrightarrow{\mathscr{X}}_1^{k+1}\|^2+(\alpha-\beta)^2\|\overrightarrow{\mathscr{X}}_2^{k+1}\|^2+\beta^2\|\overrightarrow{\mathscr{X}}_3^{k+1}\|^2\\
&\quad+(1-\alpha)(\alpha-\beta)(\|\overrightarrow{\mathscr{X}}_1^{k+1}\|^2+\|\overrightarrow{\mathscr{X}}_2^{k+1}\|^2)+(1-\alpha)\beta(\|\overrightarrow{\mathscr{X}}_1^{k+1}\|^2+\|\overrightarrow{\mathscr{X}}_3^{k+1}\|^2)\\
&\quad+(\alpha-\beta)\beta(\|\overrightarrow{\mathscr{X}}_2^{k+1}\|^2+\|\overrightarrow{\mathscr{X}}_3^{k+1}\|^2)\\
&=(1-\alpha)\|\overrightarrow{\mathscr{X}}_1^{k+1}\|^2+(\alpha-\beta)\|\overrightarrow{\mathscr{X}}_2^{k+1}\|^2+\beta\|\overrightarrow{\mathscr{X}}_3^{k+1}\|^2\\
&\le\|\overrightarrow{\mathscr{X}}^k\|^2-2(1-\alpha)(\alpha-\beta)\beta\gamma[\langle\mathscr{V}^k,\overrightarrow{\mathscr{X}}^k\times_1 A_1\rangle+\langle\mathscr{V}^k,\overrightarrow{\mathscr{X}}^k\times_2 A_2\rangle+\langle\mathscr{V}^k,\overrightarrow{\mathscr{X}}^k\times_3 A_3\rangle]\\
&\quad+(1-\alpha)(\alpha-\beta)\beta\gamma((\alpha-\beta)\beta\gamma\|A_1\|_2^2+(1-\alpha)\beta\gamma\|A_2\|_2^2+(1-\alpha)(\alpha-\beta)\gamma\|A_3\|_2^2)\|\mathscr{V}^k\|^2
\end{aligned}$$

$$
\begin{aligned}
&= \|\overrightarrow{\mathscr{X}}^k\|^2 - (1-\alpha)(\alpha-\beta)\beta\gamma\big[2 - (\alpha-\beta)\beta\gamma\|A_1\|_2^2 - (1-\alpha)\beta\gamma\|A_2\|_2^2 \\
&\qquad\qquad - (1-\alpha)(\alpha-\beta)\gamma\|A_3\|_2^2\big]\|\mathscr{V}^k\|^2 \\
&\le \|\overrightarrow{\mathscr{X}}^0\|^2 - (1-\alpha)(\alpha-\beta)\beta\gamma\big[2 - (\alpha-\beta)\beta\gamma\|A_1\|_2^2 - (1-\alpha)\beta\gamma\|A_2\|_2^2 \\
&\qquad\qquad - (1-\alpha)(\alpha-\beta)\gamma\|A_3\|_2^2\big]\sum_{j=0}^{k}\|\mathscr{V}^j\|^2.
\end{aligned}
$$

If $0<\gamma<\frac{2}{(\alpha-\beta)\beta\|A_1\|_2^2+(1-\alpha)\beta\|A_2\|_2^2+(1-\alpha)(\alpha-\beta)\|A_3\|_2^2}$ and $0<\beta<\alpha<1$, then

$$
\sum_{j=0}^{k}\|\mathscr{V}^j\|^2<\infty,
$$

which shows $\mathscr{V}^k\to 0$ as $k\to\infty$, i.e., $\overrightarrow{\mathscr{X}}^k\times_1 A_1+\overrightarrow{\mathscr{X}}^k\times_2 A_2+\overrightarrow{\mathscr{X}}^k\times_3 A_3\to 0$. By Lemma 2.4, $\mathscr{X}^k\to\mathscr{X}^*$ holds. □

To improve the performance of Algorithm 3.1, we take the information given by the previous steps to develop a tensor form of modified relaxed gradient-based iterative algorithm. It is worth noting that the last iterative solutions $\mathscr{X}_1^{k+1}$ and $\mathscr{X}_2^{k+1}$ have been calculated in the process of updating $\mathscr{X}_2^{k+1}$ and $\mathscr{X}_3^{k+1}$, respectively. By taking $\mathscr{X}_1^{k+1}$ and $\mathscr{X}_2^{k+1}$ to update $\mathscr{X}^k$, we propose a modified RGI algorithm based on tensor format (MRGI_BTF) for solving (1.1) as follows.

Algorithm 3.2 MRGI algorithm based on tensor format for solving (1.1).

Input: Given three initial tensors $\mathscr{X}_1^0$, $\mathscr{X}_2^0$, $\mathscr{X}_3^0$.
Output: $\mathscr{X}$.

For $k=0,1,2,\ldots$ until converges

1: $\mathscr{R}_i^k=\mathscr{B}-\mathscr{X}_i^k\times_1 A_1-\mathscr{X}_i^k\times_2 A_2-\mathscr{X}_i^k\times_3 A_3$,
2: $\mathscr{X}_1^{k+1}=\mathscr{X}^k+(\alpha-\beta)\beta\gamma\mathscr{R}_1^k\times_1 A_1^T$,
3: $\mathscr{X}^k=(1-\alpha)\mathscr{X}_1^{k+1}+(\alpha-\beta)\mathscr{X}_2^k+\beta\mathscr{X}_3^k$,
4: $\mathscr{X}_2^{k+1}=\mathscr{X}^k+(1-\alpha)\beta\gamma\mathscr{R}_2^k\times_2 A_2^T$,
5: $\mathscr{X}^k=(1-\alpha)\mathscr{X}_1^{k+1}+(\alpha-\beta)\mathscr{X}_2^{k+1}+\beta\mathscr{X}_3^k$,
6: $\mathscr{X}_3^{k+1}=\mathscr{X}^k+(1-\alpha)(\alpha-\beta)\gamma\mathscr{R}_3^k\times_3 A_3^T$,
7: $\mathscr{X}^{k+1}=(1-\alpha)\mathscr{X}_1^{k+1}+(\alpha-\beta)\mathscr{X}_2^{k+1}+\beta\mathscr{X}_3^{k+1}$.

Similar to Algorithm 3.1, we present a sufficient condition for guaranteeing the convergence of the above method as follows.

Theorem 3.3. *Let $\{\mathscr{X}^k\}$ be the iterative sequence given by Algorithm 3.2. For any initial tensor, if the step length γ satisfies*

$$
0<\gamma<\min\left\{\frac{2}{(\alpha-\beta)\beta\|A_1\|_2^2},\frac{2}{(1-\alpha)\beta\|A_2\|_2^2},\frac{2}{(1-\alpha)(\alpha-\beta)\|A_3\|_2^2}\right\},\quad 0<\beta<\alpha<1,
$$

then the iterative sequence $\{\mathscr{X}^k\}$ *is convergent to* $\mathscr{X}^*$.

Proof. For the sake of convenience, we replace $\mathscr{X}^k$ with $\overline{\mathscr{X}}^k$ and $\overline{\overline{\mathscr{X}}}^k$ in the third and fifth iterations of Algorithm 3.2, respectively. Then we define the error tensors as:

$$\widehat{\mathscr{X}}^k = \mathscr{X}^k - \mathscr{X}^*, \quad \widehat{\overline{\mathscr{X}}}^k = \overline{\mathscr{X}}^k - \mathscr{X}^*, \quad \widehat{\overline{\overline{\mathscr{X}}}}^k = \overline{\overline{\mathscr{X}}}^k - \mathscr{X}^*, \quad \widehat{\mathscr{X}}_i^k = \mathscr{X}_i^k - \mathscr{X}^*, \quad i = 1, 2, 3.$$

Let

$$\begin{aligned}
\mathscr{P}^k &= \widehat{\mathscr{X}}^k \times_1 A_1, & \overline{\mathscr{P}}^k &= \widehat{\overline{\mathscr{X}}}^k \times_1 A_1, & \overline{\overline{\mathscr{P}}}^k &= \widehat{\overline{\overline{\mathscr{X}}}}^k \times_1 A_1,\\
\mathscr{Q}^k &= \widehat{\mathscr{X}}^k \times_2 A_2, & \overline{\mathscr{Q}}^k &= \widehat{\overline{\mathscr{X}}}^k \times_2 A_2, & \overline{\overline{\mathscr{Q}}}^k &= \widehat{\overline{\overline{\mathscr{X}}}}^k \times_2 A_2,\\
\mathscr{U}^k &= \widehat{\mathscr{X}}^k \times_3 A_3, & \overline{\mathscr{U}}^k &= \widehat{\overline{\mathscr{X}}}^k \times_3 A_3, & \overline{\overline{\mathscr{U}}}^k &= \widehat{\overline{\overline{\mathscr{X}}}}^k \times_3 A_3.
\end{aligned}$$

Then we can easily obtain that

$$\begin{aligned}
\widehat{\mathscr{X}}_1^{k+1} &= \widehat{\mathscr{X}}^k - (\alpha-\beta)\beta\gamma\mathscr{V}^k \times_1 A_1^T,\\
\widehat{\mathscr{X}}_2^{k+1} &= \widehat{\overline{\mathscr{X}}}^k - (1-\alpha)\beta\gamma\overline{\mathscr{V}}^k \times_2 A_2^T,\\
\widehat{\mathscr{X}}_3^{k+1} &= \widehat{\overline{\overline{\mathscr{X}}}}^k - (1-\alpha)(\alpha-\beta)\gamma\overline{\overline{\mathscr{V}}}^k \times_3 A_3^T,
\end{aligned} \tag{3.6}$$

where

$$\mathscr{V}^k = \mathscr{P}^k + \mathscr{Q}^k + \mathscr{U}^k, \quad \overline{\mathscr{V}}^k = \overline{\mathscr{P}}^k + \overline{\mathscr{Q}}^k + \overline{\mathscr{U}}^k, \quad \overline{\overline{\mathscr{V}}}^k = \overline{\overline{\mathscr{P}}}^k + \overline{\overline{\mathscr{Q}}}^k + \overline{\overline{\mathscr{U}}}^k.$$

It follows from (2.1) and (3.6) that we have

$$\begin{aligned}
\|\widehat{\mathscr{X}}_1^{k+1}\|^2 &= \|\widehat{\mathscr{X}}^k - (\alpha-\beta)\beta\gamma\mathscr{V}^k \times_1 A_1^T\|^2\\
&= \langle \widehat{\mathscr{X}}^k, \widehat{\mathscr{X}}^k\rangle - 2(\alpha-\beta)\beta\gamma\langle \widehat{\mathscr{X}}^k \times_1 A_1, \mathscr{V}^k\rangle + (\alpha-\beta)^2\beta^2\gamma^2\|\mathscr{V}^k \times_1 A_1^T\|^2\\
&\le \|\widehat{\mathscr{X}}^k\|^2 - 2(\alpha-\beta)\beta\gamma\langle \mathscr{P}^k, \mathscr{V}^k\rangle + (\alpha-\beta)^2\beta^2\gamma^2\|\mathscr{V}^k\|^2\|A_1\|_2^2,\\
\|\widehat{\mathscr{X}}_2^{k+1}\|^2 &= \|\widehat{\overline{\mathscr{X}}}^k - (1-\alpha)\beta\gamma\overline{\mathscr{V}}^k \times_2 A_2^T\|^2\\
&= \langle \widehat{\overline{\mathscr{X}}}^k, \widehat{\overline{\mathscr{X}}}^k\rangle - 2(1-\alpha)\beta\gamma\langle \widehat{\overline{\mathscr{X}}}^k \times_2 A_2^T, \mathscr{V}^k\rangle + (1-\alpha)^2\beta^2\gamma^2\|\overline{\mathscr{V}}^k \times_2 A_2^T\|^2\\
&\le \|\widehat{\overline{\mathscr{X}}}^k\|^2 - 2(1-\alpha)\beta\gamma\langle \overline{\mathscr{Q}}^k, \overline{\mathscr{V}}^k\rangle + (1-\alpha)^2\beta^2\gamma^2\|\overline{\mathscr{V}}^k\|^2\|A_2\|_2^2,\\
\|\widehat{\mathscr{X}}_3^{k+1}\|^2 &= \|\widehat{\overline{\overline{\mathscr{X}}}}^k - (1-\alpha)(\alpha-\beta)\gamma\overline{\overline{\mathscr{V}}}^k \times_3 A_3^T\|^2\\
&= \|\widehat{\overline{\overline{\mathscr{X}}}}^k\|^2 - 2(1-\alpha)(\alpha-\beta)\gamma\langle \overline{\overline{\mathscr{U}}}^k, \overline{\overline{\mathscr{V}}}^k\rangle + (1-\alpha)^2(\alpha-\beta)^2\gamma^2\|\overline{\overline{\mathscr{V}}}^k \times_3 A_3^T\|^2\\
&\le \|\widehat{\overline{\overline{\mathscr{X}}}}^k\|^2 - 2(1-\alpha)(\alpha-\beta)\gamma\langle \overline{\overline{\mathscr{U}}}^k, \overline{\overline{\mathscr{V}}}^k\rangle + (1-\alpha)^2(\alpha-\beta)^2\gamma^2\|\overline{\overline{\mathscr{V}}}^k\|^2\|A_3\|_2^2.
\end{aligned}$$

Therefore, we have

$$
\begin{aligned}
\|\widehat{\mathscr{X}}^{k+1}\|^2 &= \|(1-\alpha)\widehat{\mathscr{X}}_1^{k+1} + (\alpha-\beta)\widehat{\mathscr{X}}_2^{k+1} + \beta\widehat{\mathscr{X}}_3^{k+1}\|^2 \\
&= (1-\alpha)^2\|\widehat{\mathscr{X}}_1^{k+1}\|^2 + (\alpha-\beta)^2\|\widehat{\mathscr{X}}_2^{k+1}\|^2 + \beta^2\|\widehat{\mathscr{X}}_3^{k+1}\|^2 \\
&\quad + 2(1-\alpha)(\alpha-\beta)\langle \widehat{\mathscr{X}}_1^{k+1}, \widehat{\mathscr{X}}_2^{k+1}\rangle + 2(\alpha-\beta)\beta\langle \widehat{\mathscr{X}}_2^{k+1}, \widehat{\mathscr{X}}_3^{k+1}\rangle \\
&\quad + 2(1-\alpha)\beta\langle \widehat{\mathscr{X}}_1^{k+1}, \widehat{\mathscr{X}}_3^{k+1}\rangle \\
&\leq (1-\alpha)^2\|\widehat{\mathscr{X}}_1^{k+1}\|^2 + (\alpha-\beta)^2\|\widehat{\mathscr{X}}_2^{k+1}\|^2 + \beta^2\|\widehat{\mathscr{X}}_3^{k+1}\|^2 \\
&\quad + (1-\alpha)(\alpha-\beta)(\|\widehat{\mathscr{X}}_1^{k+1}\|^2 + \|\widehat{\mathscr{X}}_2^{k+1}\|^2) + (\alpha-\beta)\beta(\|\widehat{\mathscr{X}}_2^{k+1}\|^2 + \|\widehat{\mathscr{X}}_3^{k+1}\|^2) \\
&\quad + (1-\alpha)\beta(\|\widehat{\mathscr{X}}_1^{k+1}\|^2 + \|\widehat{\mathscr{X}}_3^{k+1}\|^2) \\
&= (1-\alpha)\|\widehat{\mathscr{X}}_1^{k+1}\|^2 + (\alpha-\beta)\|\widehat{\mathscr{X}}_2^{k+1}\|^2 + \beta\|\widehat{\mathscr{X}}_3^{k+1}\|^2 \\
&\leq (1-\alpha)[\|\widehat{\mathscr{X}}^{k}\|^2 - 2(\alpha-\beta)\beta\gamma\langle \mathscr{P}^k, \mathscr{V}^k\rangle + (\alpha-\beta)^2\beta^2\gamma^2\|\mathscr{V}^k\|^2\|A_1\|_2^2] \\
&\quad + (\alpha-\beta)[\|\widehat{\overline{\mathscr{X}}}^{k}\|^2 - 2(1-\alpha)\beta\gamma\langle \overline{\mathscr{Q}}^k, \overline{\mathscr{V}}^k\rangle + (1-\alpha)^2\beta^2\gamma^2\|\overline{\mathscr{V}}^k\|^2\|A_2\|_2^2] \\
&\quad + \beta[\|\widehat{\overline{\overline{\mathscr{X}}}}^{k}\|^2 - 2(1-\alpha)(\alpha-\beta)\gamma\langle \overline{\overline{\mathscr{U}}}^k, \overline{\overline{\mathscr{V}}}^k\rangle + (1-\alpha)^2(\alpha-\beta)^2\gamma^2\|\overline{\overline{\mathscr{V}}}^k\|^2\|A_3\|_2^2] \\
&= (1-\alpha)\|\widehat{\mathscr{X}}^{k}\|^2 + (\alpha-\beta)\|\widehat{\overline{\mathscr{X}}}^{k}\|^2 + \beta\|\widehat{\overline{\overline{\mathscr{X}}}}^{k}\|^2 \\
&\quad - 2(1-\alpha)(\alpha-\beta)\beta\gamma[\langle \mathscr{P}^k, \mathscr{V}^k\rangle + \langle \overline{\mathscr{Q}}^k, \overline{\mathscr{V}}^k\rangle + \langle \overline{\overline{\mathscr{U}}}^k, \overline{\overline{\mathscr{V}}}^k\rangle] \\
&\quad + (1-\alpha)(\alpha-\beta)^2\beta^2\gamma^2(\|\mathscr{V}^k\|^2\|A_1\|_2^2) + (1-\alpha)^2(\alpha-\beta)\beta^2\gamma^2(\|\overline{\mathscr{V}}^k\|^2\|A_2\|_2^2) \\
&\quad + (1-\alpha)^2(\alpha-\beta)^2\beta\gamma^2(\|\overline{\overline{\mathscr{V}}}^k\|^2\|A_3\|_2^2) \\
&\leq (1-\alpha)\|\widehat{\mathscr{X}}^{k}\|^2 + (\alpha-\beta)\|\widehat{\overline{\mathscr{X}}}^{k}\|^2 + \beta\|\widehat{\overline{\overline{\mathscr{X}}}}^{k}\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (\alpha-\beta)\beta\gamma\|A_1\|_2^2]\|\mathscr{V}^k\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (1-\alpha)\beta\gamma\|A_2\|_2^2]\|\overline{\mathscr{V}}^k\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (1-\alpha)(\alpha-\beta)\gamma\|A_3\|_2^2]\|\overline{\overline{\mathscr{V}}}^k\|^2 \\
&\leq \|\widehat{\mathscr{X}}^{0}\|^2 + (\alpha-\beta)\sum_{j=0}^{k}\|\widehat{\overline{\mathscr{X}}}^{k}\|^2 + \beta\sum_{j=0}^{k}\|\widehat{\overline{\overline{\mathscr{X}}}}^{k}\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (\alpha-\beta)\beta\gamma\|A_1\|_2^2]\sum_{j=0}^{k}\|\mathscr{V}^k\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (1-\alpha)\beta\gamma\|A_2\|_2^2]\sum_{j=0}^{k}\|\overline{\mathscr{V}}^k\|^2 \\
&\quad - (1-\alpha)(\alpha-\beta)\beta\gamma[2 - (1-\alpha)(\alpha-\beta)\gamma\|A_3\|_2^2]\sum_{j=0}^{k}\|\overline{\overline{\mathscr{V}}}^k\|^2.
\end{aligned}
$$

From the above process, it is not difficult to verify that $\sum_{j=0}^{k}\|\widehat{\overline{\mathscr{X}}}^{j}\|^2 < \infty$ and $\sum_{j=0}^{k}\|\widehat{\overline{\overline{\mathscr{X}}}}^{j}\|^2$

$< \infty$. Then Algorithm 3.2 is convergent to $\mathscr{X}^*$ if

$$0 < \gamma < \min\left\{\frac{2}{(\alpha-\beta)\beta\|A_1\|_2^2}, \frac{2}{(1-\alpha)\beta\|A_2\|_2^2}, \frac{2}{(1-\alpha)(\alpha-\beta)\|A_3\|_2^2}\right\}.$$

Consequently,

$$\sum_{j=0}^{k}\|\mathscr{V}^j\|^2 < \infty, \quad \sum_{j=0}^{k}\|\overline{\mathscr{V}}^j\|^2 < \infty, \quad \sum_{j=0}^{k}\|\overline{\overline{\mathscr{V}}}^j\|^2 < \infty.$$

This fact shows that $\mathscr{V}^k \to 0$, $\overline{\mathscr{V}}^k \to 0$ and $\overline{\overline{\mathscr{V}}}^k \to 0$ as $k \to \infty$, i.e., $\widehat{\mathscr{X}}^k \to 0$, $\widehat{\overline{\mathscr{X}}}^k \to 0$ and $\widehat{\overline{\overline{\mathscr{X}}}}^k \to 0$ hold. □

4. Numerical experiments

In this part, we provide several examples to test the feasibility and validity of the algorithms proposed. The proposed methods were executed by Matlab R2016b on PC with Inter(R) Core(TM) i7-4720M@2.2 GHz and 8.00 G memory. Moreover, we implemented all the operations via the tensor toolbox (version 2.5) [31]. The symbols CPU and IT denote the elapsed time and the number of iteration steps, respectively. We take the relative residual error RES $= \|\mathscr{R}^k\|/\|\mathscr{R}^0\| < 10^{-10}$ as the stopping rule for the above four algorithms, where $\mathscr{R}^k$ is the residual tensor at k-th iteration. Besides, the step length μ involved in GI algorithm and RGI_BTF algorithm is chosen by $\frac{1}{\|A_1\|_2^2+\|A_2\|_2^2+\|A_3\|_2^2}$ and $\frac{1}{(\alpha-\beta)\beta\|A_1\|_2^2+(1-\alpha)\beta\|A_2\|_2^2+(1-\alpha)(\alpha-\beta)\|A_3\|_2^2}$, respectively.

Example 4.1. We reconsider the Example 1 in [10] such that

$$A_1 = \begin{pmatrix} 3 & 1 \\ -1 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 1 & 0 \\ 1 & -2 \end{pmatrix},$$
$$\mathscr{B}(:,:,1) = \begin{pmatrix} 10 & 13 \\ 15 & 11 \end{pmatrix}, \quad \mathscr{B}(:,:,2) = \begin{pmatrix} 14 & 3 \\ 3 & 0 \end{pmatrix}.$$

Starting with the chosen tensors $\mathscr{X}_1^0 = \mathscr{X}_2^0 = \mathscr{X}_3^0 = 10^{-6} \cdot \text{tenones}(2,2,2)$, we implemented the proposed algorithms and presented the test results in Table 4.1. As shown in Table 4.1, it is easy to see that RGI_BTF and MRGI_BTF algorithms converge faster than GI algorithm, where MRGI_BTF algorithm outperforms than other algorithms. The convergence curves of these algorithms were recorded in Figure 4.1.

Algorithms	IT	CPU	RES
GI	623	2.7601	9.8923e-11
MGI	136	1.1472	9.6950e-11
RGI_BTF	202	1.4510	9.2056e-11
MRGI_BTF	58	0.5297	7.2931e-11

Table 4.1: Test results for Example 4.1.

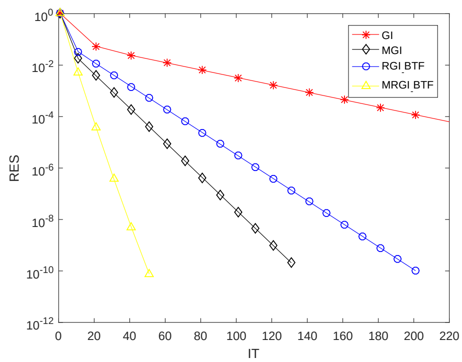

Figure 4.1: Comparison of convergence curves.

Example 4.2. In this example, we reconsider the Example 2 [10] with different parameter ρ such that

$$\begin{aligned}
&N_1 = N_2 = N_3 = 30;\\
&A_1 = \mathrm{triu}(\mathrm{rand}(N_1, N_2), 1) + \mathrm{diag}(\rho + \mathrm{diag}(\mathrm{rand}(N_1)));\\
&A_2 = \mathrm{triu}(\mathrm{rand}(N_2, N_2), 1) + \mathrm{diag}(\rho + \mathrm{diag}(\mathrm{rand}(N_2)));\\
&A_3 = \mathrm{triu}(\mathrm{rand}(N_3, N_3), 1) + \mathrm{diag}(\rho + \mathrm{diag}(\mathrm{rand}(N_3)));\\
&\mathscr{B} = \mathrm{tenrand}(N_1, N_2, N_3).
\end{aligned}$$

We set the initial tensors

$$\mathscr{X}_1^0 = \mathscr{X}_2^0 = \mathscr{X}_3^0 = 10^{-6} \cdot \mathrm{tenones}(N_1, N_2, N_3).$$

Then we tested the proposed algorithms and presented the corresponding results in Tables 4.2 and 4.3. It follows from these tables that the relative residual error, the elapsed CPU time and the iteration steps involved with RGI_BTF and MRGI_BTF algorithms are commonly less than GI algorithm. Besides, the CPU time and the iteration steps costed by MRGI_BTF algorithm are less than that of MGI algorithm. According to Figure 4.2, we can easily observe that MRGI_BTF algorithm converges much faster than MGI algorithm.

Algorithms	IT	CPU	RES
GI	697	8.7022	9.7943e-11
MGI	170	3.4464	9.2514e-11
RGI_BTF	328	4.0872	9.8076e-11
MRGI_BTF	121	1.9227	9.2225e-11

Table 4.2: Test results for Example 4.2 with $\rho = 3$.

Algorithms	IT	CPU	RES
GI	223	2.0230	9.6016e-11
MGI	53	0.8519	8.3845e-11
RGI_BTF	120	1.0771	9.8830e-11
MRGI_BTF	39	0.5985	8.2435e-11

Table 4.3: Test results for Example 4.2 with $\rho = 5$.

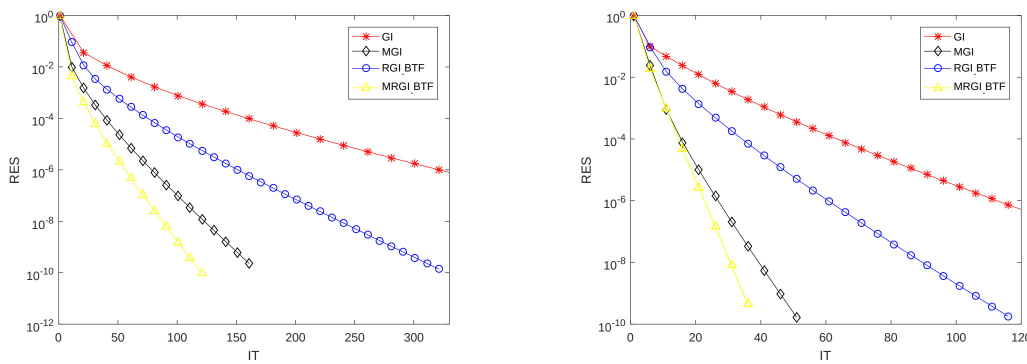

Figure 4.2: Comparison of convergence curves for Example 4.2 with $\rho = 3$ (left) and $\rho = 5$ (right).

Example 4.3. We consider the solution of the following convection-diffusion equation [7]

$$
\begin{aligned}
-x\Phi y + z^T \Psi y &= g \qquad \text{in } \Gamma = [0,1] \times [0,1] \times [0,1], \\
x &= 0 \qquad \text{on } \partial\Gamma.
\end{aligned}
$$

According to a standard finite difference discretization on equidistant nodes and a second order convergent scheme (Fromm's scheme), we solve the linear system (2.2) with

$$
\begin{aligned}
&A_1 = A_2 = A_3 \\
&= \begin{bmatrix}
2vh^{-2} + \frac{3}{4}ch^{-1} & -vh^{-2} - \frac{5}{4}ch^{-1} & \frac{1}{4}ch^{-1} & \\
-vh^{-2} + \frac{1}{4}ch^{-1} & 2vh^{-2} + \frac{3}{4}ch^{-1} & -vh^{-2} - \frac{5}{4}ch^{-1} & \frac{1}{4}ch^{-1} \\
\ddots & \ddots & \ddots & \ddots \\
 & \ddots & \ddots & -vh^{-2} - \frac{5}{4}ch^{-1} \\
0 & \cdots & -vh^{-2} + \frac{1}{4}ch^{-1} & 2vh^{-2} + \frac{3}{4}ch^{-1}
\end{bmatrix}_{n\times n}.
\end{aligned}
$$

If we take $h = 1/(n+1)$, then

$$A_1 = A_2 = A_3 = \frac{v}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} + \frac{c}{4h}\begin{bmatrix} 3 & -5 & 1 & & \\ 1 & 3 & -5 & & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{bmatrix},$$

$$\mathscr{B} = \mathrm{tenrand}(n,n,n).$$

Let $v = c = 1$ and $\mathscr{X}_1^0 = \mathscr{X}_2^0 = \mathscr{X}_3^0 = 10^{-6} \cdot \mathrm{tenones}(n,n,n)$. We used the proposed algorithms to obtain the iterative solutions of (1.1) with $n = 3$ and $n = 6$, respectively. We recorded the convergence curves of these algorithms in Figure 4.3. The recorded figure demonstrates that the introduced algorithms are feasible and effective, where the MRGI_BTF algorithm performs at their best.

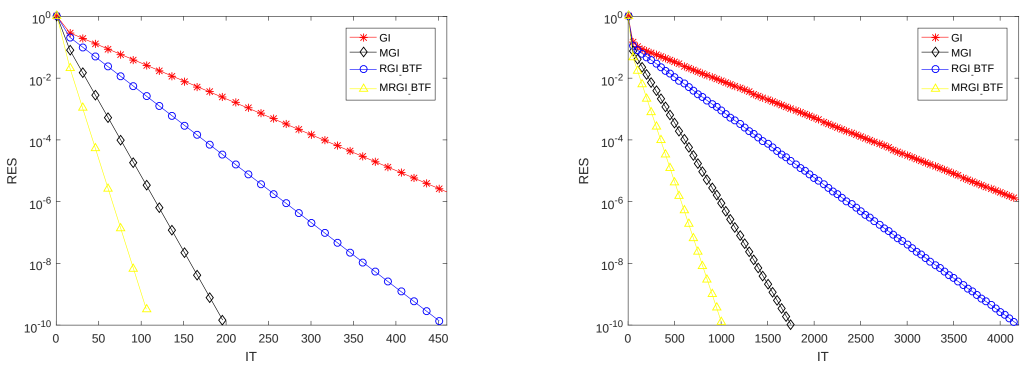

Figure 4.3: Comparison of convergence curves for Example 4.3 with $n = 3$ (left) and $n = 6$ (right).

5. Conclusion

In this paper, we proposed a relaxed gradient-based iterative algorithm based on tensor form for solving (1.1). By using the information provided in the previous steps, we further developed a modified version to improve the RGI_BTF algorithm. Under some appropriate conditions, the convergence analysis shows that the introduced algorithms converge to the unique solution for any initial value. Finally, the limited numerical results illustrate that the MRGI_BTF algorithm performs at their best.

References

[1] Z.-Z. Bai, *Modulus-based matrix splitting iteration methods for linear complementarity problems*, Numer. Linear Algebra Appl. **17** (2010), no. 6, 917–933.

[2] ———, *On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equations*, J. Comput. Math. **29** (2011), no. 2, 185–198.

[3] R. H. Bartels and G. W. Stewart, *Algorithm 432: Solution of the matrix equation* $AX + XB = C$, Comm. ACM **15** (1972), no. 9, 820–826.

[4] U. Baur and P. Benner, *Cross-Gramian based model reduction for data-sparse systems*, Electron. Trans. Numer. Anal. **31** (2008), 256–270.

[5] P. Benner, R.-C. Li and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math. **233** (2009), no. 4, 1035–1045.

[6] F. P. A. Beik and S. Ahmadi-Asl, *Residual norm steepest descent based iterative algorithms for Sylvester tensor equations*, J. Math. Model. **2** (2015), no. 2, 155–131.

[7] F. P. A. Beik, F. Saberi Movahed and S. Ahmadi-Asl, *On the Krylov subspace methods based on tensor format for positive definite Sylvester tensor equations*, Numer. Linear Algebra Appl. **23** (2016), no. 3, 444–466.

[8] D. Calvetti and L. Reichel, *Application of ADI iterative methods to the restoration of noisy images*, SIAM J. Matrix Anal. Appl. **17** (1996), no. 1, 165–186.

[9] Z. Chen and L. Lu, *A projection method and Kronecker product preconditioner for solving Sylvester tensor equations*, Sci. China Math. **55** (2012), no. 6, 1281–1292.

[10] ———, *A gradient based iterative solutions for Sylvester tensor equations*, Math. Probl. Eng. **2013**, Art. ID 819479, 7 pp.

[11] D. Chu and V. Mehrmann, *Disturbance decoupling for descriptor systems by state feedback*, SIAM J. Control Optim. **38** (2000), no. 6, 1830–1858.

[12] M. Dehghan and M. Hajarian, *An iterative method for solving the generalized coupled Sylvester matrix equations over generalized bisymmetric matrices*, Appl. Math. Model. **34** (2010), no. 3, 639–654.

[13] F. Ding and T. Chen, *Gradient based iterative algorithms for solving a class of matrix equations*, IEEE Trans. Automat. Control **50** (2005), no. 8, 1216–1221.

[14] ______, *Hierarchical gradient-based identification of multivariable discrete-time systems*, Automatica J. IFAC **41** (2005), no. 2, 315–325.

[15] ______, *Hierarchical identification of lifted state-space models for general dual-rate systems*, IEEE Trans. Circuits Syst. I. Regul. Pap. **52** (2005), no. 6, 1179–1187.

[16] ______, *Hierarchical least squares identification methods for multivariable systems*, IEEE Trans. Automat. Control **50** (2005), no. 3, 397–402.

[17] ______, *Iterative least-squares solutions of coupled Sylvester matrix equations*, Systems Control Lett. **54** (2005), no. 2, 95–107.

[18] ______, *On iterative solutions of general coupled matrix equations*, SIAM J. Control Optim. **44** (2006), no. 6, 2269–2284.

[19] F. Ding, P. X. Liu and J. Ding, *Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle*, Appl. Math. Comput. **197** (2008), no. 1, 41–50.

[20] G. H. Golub, S. Nash and C. Van Loan, *A Hessenberg–Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control **24** (1979), no. 6, 909–913.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Third edition, Johns Hopkins University Press, Baltimore, MD, 1996.

[22] Y. Guan and D. Chu, *Numerical computation for orthogonal low-rank approximation of tensors*, SIAM J. Matrix Anal. Appl. **40** (2019), no. 3, 1047–1065.

[23] Y. Guan, M. T. Chu and D. Chu, *Convergence analysis of an SVD-based algorithm for the best rank-1 tensor approximation*, Linear Algebra Appl. **555** (2018), 53–69.

[24] ______, *SVD-based algorithms for the best rank-1 approximation of a symmetric tensor*, SIAM J. Matrix Anal. Appl. **39** (2018), no. 3, 1095–1115.

[25] B. Huang and C. Ma, *An iterative algorithm to solve the generalized Sylvester tensor equations*, Linear Multilinear Algebra **68** (2020), no. 6, 1175–1200.

[26] ______, *Global least squares methods based on tensor form to solve a class of generalized Sylvester tensor equations*, Appl. Math. Comput. **369** (2020), 124892, 16 pp.

[27] B. Huang, Y. Xie and C. Ma, *Krylov subspace methods to solve a class of tensor equations via the Einstein product*, Numer. Linear Algebra Appl. **26** (2019), no. 4, e2254, 22 pp.

[28] A. Kaabi, A. Kerayechian and F. Toutounian, *A new version of successive approximations method for solving Sylvester matrix equations*, Appl. Math. Comput. **186** (2007), no. 1, 638–645.

[29] A. Kaabi, F. Toutounian and A. Kerayechian, *Preconditioned Galerkin and minimal residual methods for solving Sylvester equations*, Appl. Math. Comput. **181** (2006), no. 2, 1208–1214.

[30] S. Karimi and M. Dehghan, *Global least squares method based on tensor form to solve linear systems in Kronecker format*, Trans. Inst. Meas. Control **40** (2018), no. 7, 2378–2386.

[31] T. Kolda, *Multilinear operators for higher-order decompositions*, Technical report, Sandia National Laboratories, Albuquerque, NM, and Livermore, CA, 2006.

[32] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev. **51** (2009), no. 3, 455–500.

[33] J. Li, F. Ding and G. Yang, *Maximum likelihood least squares identification method for input nonlinear finite impulse response moving average systems*, Math. Comput. Modelling **55** (2012), no. 3-4, 442–450.

[34] T. Li, Q.-W. Wang and X.-F. Zhang, *Gradient based iterative methods for solving symmetric tensor equations*, Accepted in Numer. Linear Algebra Appl. (2021), e2414. `https://doi.org/10.1002/nla.2414`

[35] Y.-Q. Lin, *Implicitly restarted global FOM and GMRES for nonsymmetric matrix equations and Sylvester equations*, Appl. Math. Comput. **167** (2005), no. 2, 1004–1025.

[36] C.-Q. Lv and C.-F. Ma, *A Levenberg–Marquardt method for solving semi-symmetric tensor equations*, J. Comput. Appl. Math. **332** (2018), 13–25.

[37] ———, *A modified CG algorithm for solving generalized coupled Sylvester tensor equations*, Appl. Math. Comput. **365** (2020), 124699, 15 pp.

[38] Q. Niu, X. Wang and L.-Z. Lu, *A relaxed gradient based algorithm for solving Sylvester equations*, Asian J. Control **13** (2011), no. 3, 461–464.

[39] L. Qi, H. Chen and Y. Chen, *Tensor Eigenvalues and Their Applications*, Advances in Mechanics and Mathematics **39**, Springer, Singapore, 2018.

[40] L. Qi and Z. Luo, *Tensor Analysis: Spectral theory and special tensors*, SIAM, Philadelphia, 2017.

[41] D. C. Sorensen and Y. Zhou, *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math. **2003**, no. 6, 277–303.

[42] Q.-W. Wang and X. Xu, *Iterative algorithms for solving some tensor equations*, Linear Multilinear Algebra **67** (2019), no. 7, 1325–1349.

[43] Q.-W. Wang, X. Xu and X. Duan, *Least squares solution of the quaternion Sylvester tensor equation*, Linear Multilinear Algebra **69** (2021), no. 1, 104–130.

[44] W. Wang, F. Ding and J. Dai, *Maximum likelihood least squares identification for systems with autoregressive moving average noise*, Appl. Math. Model. **36** (2012), no. 5, 1842–1853.

[45] M. Xie and Q.-W. Wang, *Reducible solution to a quaternion tensor equation*, Front. Math. China **15** (2020), no. 5, 1047–1070.

[46] Y.-J. Xie and C.-F. Ma, *The accelerated gradient based iterative algorithm for solving a class of generalized Sylvester-transpose matrix equation*, Appl. Math. Comput. **273** (2016), 1257–1269.

[47] X. Xu and Q.-W. Wang, *Extending BiCG and BiCR methods to solve the Stein tensor equation*, Comput. Math. Appl. **77** (2019), no. 12, 3117–3127.

[48] X. F. Zhang and Q.-W. Wang, *Developing iterative algorithms to solve Sylvester tensor equations*, Appl. Math. Comput. **409** (2021), Paper No. 126403, 14 pp.

[49] B. Zhou, J. Lam and G.-R. Duan, *Gradient-based maximal convergence rate iterative method for solving linear matrix equations*, Int. J. Comput. Math. **87** (2010), no. 1-3, 515–527.

[50] D. Zhou, G. Chen and Q. Cai, *On modified HSS iteration methods for continuous Sylvester equations*, Appl. Math. Comput. **263** (2015), 84–93.

Xin-Fang Zhang
Department of Mathematics, Shanghai University, Shanghai 200444, China
E-mail address: `zhangxinfang0703@163.com`

Qing-Wen Wang
Department of Mathematics, Shanghai University, Shanghai 200444, China
and
Collaborative Innovation Center for the Marine Artificial Intelligence, Shanghai 200444, China
E-mail addresses: `wqw@t.shu.edu.cn`, `wqw369@yahoo.com`