AIMS Mathematics, 9(12): 34734–34752.
DOI:10.3934/math.20241654
Received: 25 September 2024
Revised: 17 November 2024
Accepted: 29 November 2024
Published: 12 December 2024

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

Research article

Two accelerated gradient-based iteration methods for solving the Sylvester matrix equation AX + XB = C

Huiling Wang[1,*], **Nian-Ci Wu**[2] **and Yufeng Nie**[3]

[1] College of Applied Mathematics, Shanxi University of Finance and Economics, Taiyuan 030006, China

[2] School of Mathematics and Statistics, South-Central Minzu University, Wuhan 430074, China

[3] School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710072, China

* **Correspondence:** Email: wanghuiling@sxufe.edu.cn.

Abstract: In this paper, combining the precondition technique and momentum item with the gradient-based iteration algorithm, two accelerated iteration algorithms are presented for solving the Sylvester matrix equation $AX + XB = C$. Sufficient conditions to guarantee the convergence properties of the proposed algorithms are analyzed in detail. Varying the parameters of these algorithms in each iteration, the corresponding adaptive iteration algorithms are also provided, and the adaptive parameters can be explicitly obtained by the minimum residual technique. Several numerical examples are implemented to illustrate the effectiveness of the proposed algorithms.

Keywords: Sylvester matrix equation; gradient-based iteration; momentum term; precondition technique; minimum residual technique
Mathematics Subject Classification: 15A24, 65F30

1. Introduction

In this paper, we consider the iterative solution of the following Sylvester matrix equation:

$$AX + XB = C, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times n}$ are constant matrices and $X \in \mathbb{R}^{m\times n}$ is the unknown matrix to be obtained.

Due to the extensive applications of Eq (1.1) in control theory and stability analysis [1,10,14], it has garnered considerable attention, and many algorithms have been proposed over the past few decades. For example, the gradient-based iteration (GI) algorithm described in [6, 8, 9] has proven to be an

effective method for solving Eq (1.1). By incorporating a tunable parameter into the GI algorithm, a relaxed gradient-based iteration (RGI) algorithm [18] was introduced, which demonstrates improved performance over the GI algorithm when the relaxed parameter is appropriately adopted. To enhance the RGI algorithm's efficiency, an accelerated gradient-based iteration (AGBI) algorithm was proposed by leveraging the latest information from the preceding half-step in [25]. In order to achieve a lower computational cost, a Jacobi gradient iteration (JGI) method was outlined in [13] based on the Jacobi splitting of A and B. Drawing inspiration from the AGBI and JGI algorithms, Tian et al. [21] further developed an accelerated JGI (AJGI) algorithm. Additionally, various other iteration algorithms [7,20, 22] have been devised for solving Eq (1.1) and other related matrix equations [17, 23, 24, 29], because of its wide applications.

Preconditioning techniques aim to alter the spectral characteristics of matrices through linear transformations, which are often integrated with other iteration methods and lead to various new algorithms such as the preconditioned HSS method [2, 16], generalized preconditioned HSS methods [28], and preconditioned MHSS iteration methods [4], etc. The heavy-ball momentum method is widely applied to accelerate the convergence rate of the gradient method [5, 19]. In this paper, inspired by the references [2, 5, 19], we combine the precondition technique and the momentum item with the gradient-based iteration algorithm, and the specific work can be summarized as follows:

(a) Novel Methodology. We have developed the preconditioned gradient-based iteration (PGI) and gradient-based momentum iteration (GMI) algorithms for solving Eq (1.1), which are more efficient than existing methods in terms of computational complexity and accuracy.

(b) Theoretical Insights. Our work provides new theoretical insights into gradient-based iteration algorithms. The convergences of PGI and GMI algorithms are rigorously proved.

(c) Adaptive Parameter Selection. We have developed a new parameter selection strategy that minimizes the current residual norm, leading to improved performance of the proposed algorithms. This strategy is practical and can be easily implemented in various numerical algorithms, enhancing their efficiency and accuracy.

(d) Empirical Results. Through extensive numerical experiments, we have shown that our methods outperform current state-of-the-art techniques in the solving of Eq (1.1).

The remainder of this paper is organized as follows: In Section 2, we first review the GI algorithm, and present the PGI and GMI algorithms, whose convergence properties are analyzed in detail. In Section 3, we construct the adaptive PGI and GMI algorithms in which the parameters are updated by utilizing the iterative information. In Section 4, several numerical examples are employed to show the robustness and efficiencies of the proposed algorithms. Finally, some conclusions are drawn in the last section.

2. Two accelerated GI algorithms

In this section, we first review the GI algorithm. Subsequently, two accelerated GI algorithms are presented, and detailed analyses are conducted on their convergence properties. In the following, several lemmas are given, which will be used in the subsequent proofs.

Lemma 2.1. [12] Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, and

$$\mathcal{R}(A, B) := \{M \in \mathbb{R}^{n\times p} | \exists\, Z \in \mathbb{R}^{m\times q},\, s.t.\, M = A^T Z B^T\}.$$

For any matrix $M \in \mathcal{R}(A, B)$, it holds that

$$\|AMB\|_F^2 \geq \sigma_{min}^2(A)\sigma_{min}^2(B)\|M\|_F^2,$$

where $\sigma_{min}(A)$ and $\sigma_{min}(B)$ are the smallest singular values of the matrices A and B, respectively.

Lemma 2.2. [11, 15] Both roots of the real quadratic equation $x^2 - bx + c = 0$ are less than one in modulus if and only if $|c| < 1$ and $|b| < 1 + c$.

By utilizing the hierarchical identification principle, Eq (1.1) can be reformulated into two subsystems as follows:

$$AX = C - XB, \quad XB = C - AX. \tag{2.1}$$

The GI algorithm for solving (2.1) can be described as follows:

Algorithm 1 The GI algorithm [8].

Require: Given an initial approximate matrix $X^{(0)}$ and the parameter μ

Ensure: $X^{(k)}$

1: **For** $k = 1, 2, \cdots$, **until converges, do**
2: $X_1^{(k)} = X^{(k-1)} + \mu A^T[C - AX^{(k-1)} - X^{(k-1)}B]$,
3: $X_2^{(k)} = X^{(k-1)} + \mu[C - AX^{(k-1)} - X^{(k-1)}B]B^T$,
4: $X^{(k)} = [X_1^{(k)} + X_2^{(k)}]/2$.
5: **End**

It is shown that the GI algorithm [8] converges when

$$0 < \mu < \frac{2}{\lambda_{max}(AA^T) + \lambda_{max}(B^TB)},$$

where $\lambda_{max}(AA^T)$ and $\lambda_{max}(B^TB)$ are the largest eigenvalues of AA^T and B^TB, respectively.

2.1. The PGI algorithm

By introducing two preconditioners, P and Q, in Algorithm 1, a preconditioned gradient-based iterative (i.e., PGI) algorithm is constructed and summarized as follows:

Algorithm 2 The PGI algorithm.

Require: Given an initial matrix $X^{(0)}$, two preconditioners P and Q, and the parameter μ

Ensure: $X^{(k)}$

1: **For** $k = 1, 2, \cdots$, **until converges, do**
2: $X_1^{(k)} = X^{(k-1)} + \mu P^{-1}A^T[C - AX^{(k-1)} - X^{(k-1)}B]$,
3: $X_2^{(k)} = X^{(k-1)} + \mu[C - AX^{(k-1)} - X^{(k-1)}B]B^TQ^{-1}$,
4: $X^{(k)} = [X_1^{(k)} + X_2^{(k)}]/2$.
5: **End**

Remark 1. *If $P = I_m$ and $Q = I_n$ are adopted, the PGI iteration method is reduced to the original GI algorithm in [8], where I_s is an identity matrix with size s.*

Remark 2. *Two practical choices of the matrices P and Q are listed as follows:*

1) $P = \text{diag}(A), Q = \text{diag}(B)$, where $\text{diag}(A)$ and $\text{diag}(B)$ are the diagonal matrices of A and B, respectively.
2) $P = \text{tridiag}(A^TA), Q = \text{tridiag}(BB^T)$, where $\text{tridiag}(A^TA)$ and $\text{tridiag}(BB^T)$ are the tridiagonal matrices of A^TA and BB^T, respectively.

Theorem 2.1. Let X^* be the solution of Eq (1.1). The iterative solution $X^{(k)}$ generated by Algorithm 2 converges to X^* for any initial value if and only if the parameter μ satisfies the condition

$$\|2I_{mn} - \mu(I \otimes P^{-1}A^TA + B^T \otimes P^{-1}A^T + Q^{-T}B \otimes A + Q^{-T}BB^T \otimes I)\|_2 < 2, \tag{2.2}$$

where $\|\cdot\|_2$ is the 2-norm of the matrix.

Proof: For $k = 1, 2, \cdots$, define the kth error matrices $\widetilde{X}^{(k)} := X^{(k)} - X^*$, which satisfy the following recurrence:

$$\begin{aligned}\widetilde{X}^{(k)} =&\widetilde{X}^{(k-1)} - \frac{\mu}{2}P^{-1}A^TA\widetilde{X}^{(k-1)} - \frac{\mu}{2}P^{-1}A^T\widetilde{X}^{(k-1)}B\\ &- \frac{\mu}{2}A\widetilde{X}^{(k-1)}B^TQ^{-1} - \frac{\mu}{2}\widetilde{X}^{(k-1)}BB^TQ^{-1}.\end{aligned} \tag{2.3}$$

By using the Kronecker product [26], the above equation can be reformulated by

$$\begin{aligned}vec(\widetilde{X}^{(k)}) =&vec(\widetilde{X}^{(k-1)}) - \frac{\mu}{2}(I_n \otimes P^{-1}A^TA)vec(\widetilde{X}^{(k-1)}) - \frac{\mu}{2}(B^T \otimes P^{-1}A^T)vec(\widetilde{X}^{(k-1)})\\ &- \frac{\mu}{2}(Q^{-T}B \otimes A)vec(\widetilde{X}^{(k-1)}) - \frac{\mu}{2}(Q^{-T}BB^T \otimes I_m)vec(\widetilde{X}^{(k-1)}).\end{aligned}$$

Taking the 2-norm of $vec(\widetilde{X}^{(k)})$, it follows that

$$\|vec(\widetilde{X}^{(k)})\|_2 \le \eta\|vec(\widetilde{X}^{(k-1)})\|_2,$$

where

$$\eta = \frac{1}{2}\|2I_{mn} - \mu(I_n \otimes P^{-1}A^TA + B^T \otimes P^{-1}A^T + Q^{-T}B \otimes A + Q^{-T}BB^T \otimes I_m)\|_2.$$

Thus,

$$\|vec(\widetilde{X}^{(k)})\|_2 \le \eta\|vec(\widetilde{X}^{(k-1)})\|_2 \le \cdots \le \eta^k\|vec(\widetilde{X}^{(0)})\|_2.$$

If μ satisfies (2.2), we know that $\widetilde{X}^{(k)} \to 0$ as $k \to \infty$. The proof is completed.

2.2 The GMI algorithm

In order to make full use of the information from the previous iteration step, a momentum term will be added to the GI algorithm, and then the second accelerated GI (i.e., GMI) algorithm will be proposed and summarized as follows:

Algorithm 3 The GMI algorithm.

Require: Given two initial matrices $X^{(0)}$ and $X^{(1)}$, and two parameters μ and β
Ensure: $X^{(k)}$

1: **For** $k = 2, 3, 4, \cdots$, **until converges, do**
2: $X_1^{(k)} = X^{(k-1)} + \mu A^T[C - AX^{(k-1)} - X^{(k-1)}B]$,
3: $X_2^{(k)} = X^{(k-1)} + \mu[C - AX^{(k-1)} - X^{(k-1)}B]B^T$,
4: $X^{(k)} = [X_1^{(k)} + X_2^{(k)}]/2 + \beta(X^{(k-1)} - X^{(k-2)})$.
5: **End**

Remark 3. *If β is chosen to be 0, then the GMI algorithm is just the GI algorithm.*

Theorem 2.2. Assume that the matrices A and B are non-singular. If Eq (1.1) has a unique solution X^*, then the iterative solution $X^{(k)}$ obtained from Algorithm 3 converges to X^* for any initial values if and only if the parameters μ and β satisfy the following conditions:
Case 1: When $3q_2 - q_1^2 > 0$,

$$\begin{cases} \sqrt{\dfrac{q_1^2 - 2q_2}{4q_2}} \le \beta < a, \\ \dfrac{q_1 - \sqrt{c}}{q_2} < \mu < \dfrac{q_1 + \sqrt{c}}{q_2}, \end{cases} \tag{2.4}$$

or

$$\begin{cases} 0 < \beta < b, \\ \dfrac{q_1 - \sqrt{c}}{q_2} < \mu \le \dfrac{q_1 - \sqrt{d}}{q_2} \ or \ \dfrac{q_1 + \sqrt{d}}{q_2} \le \mu < \dfrac{q_1 + \sqrt{c}}{q_2}, \end{cases} \tag{2.5}$$

where $q_1 = \sigma_{min}^2(A) + \sigma_{min}^2(B) - 2\|A\|_2\|B\|_2$, $q_2 = (\|A\|_2^2 + \|B\|_2^2)(\|A\|_2 + \|B\|_2)^2$, $a = \min\left\{\frac{1}{2}, \sqrt{\frac{q_1^2 - q_2}{8q_2}}\right\}$, $b = \min\left\{\frac{1}{2}, \sqrt{\frac{q_1^2 - 2q_2}{4q_2}}\right\}$, $c = q_1^2 - q_2(8\beta^2 + 1)$ and $d = q_1^2 - q_2(4\beta^2 + 2)$.
Case 2: When $3q_2 - q_1^2 \le 0$,

$$\begin{cases} 0 < \beta < a, \\ \dfrac{q_1 - \sqrt{c}}{q_2} < \mu \le \dfrac{q_1 - \sqrt{d}}{q_2} \ or \ \dfrac{q_1 + \sqrt{d}}{q_2} \le \mu < \dfrac{q_1 + \sqrt{c}}{q_2}. \end{cases} \tag{2.6}$$

Proof: Define the error matrices

$$\widetilde{X}_1^{(k)} := X_1^{(k)} - X^*, \ \ \widetilde{X}_2^{(k)} := X_2^{(k)} - X^*, \ \ \widetilde{X}^{(k)} := X^{(k)} - X^*, \ \ k = 1, 2, \cdots.$$

From Algorithm 3, it follows that

$$\begin{cases} \widetilde{X}_1^{(k)} = \widetilde{X}^{(k-1)} + \mu A^T[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B], \\ \widetilde{X}_2^{(k)} = \widetilde{X}^{(k-1)} + \mu[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B]B^T. \end{cases} \tag{2.7}$$

Taking the F-norm on (2.7), it yields that

$$\begin{aligned} \left\|\widetilde{X}_1^{(k)}\right\|_F^2 &= \left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 2\mu tr\{(A\widetilde{X}^{(k-1)})^T[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B]\} \\ &\quad + \mu^2 \left\|A^T[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B]\right\|_F^2 \\ &\le \left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 2\mu tr\{(A\widetilde{X}^{(k-1)})^T[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B]\} \\ &\quad + \mu^2 \|A\|_2^2 \left\|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\right\|_F^2, \end{aligned} \tag{2.8}$$

and

$$\begin{aligned} \left\|\widetilde{X}_2^{(k)}\right\|_F^2 &\le \left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 2\mu tr\{[-A\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-1)}B](\widetilde{X}^{(k-1)}B)^T\} \\ &\quad + \mu^2 \|B\|_2^2 \left\|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\right\|_F^2. \end{aligned} \tag{2.9}$$

By the triangle inequality and the property of the F-norm, we have

$$\begin{aligned}&\|A\widetilde{X}^{(k-1)}\|_F - \|\widetilde{X}^{(k-1)}B\|_F \le \|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\|_F\\ &\le \|A\widetilde{X}^{(k-1)}\|_F + \|\widetilde{X}^{(k-1)}B\|_F \le (\|A\|_2 + \|B\|_2)\|\tilde{X}^{(k-1)}\|_F,\end{aligned}$$

or

$$\|\widetilde{X}^{(k-1)}B\|_F - \|A\widetilde{X}^{(k-1)}\|_F \le \|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\|_F \le (\|A\|_2 + \|B\|_2)\|\tilde{X}^{(k-1)}\|_F.$$

Squaring both sides, we obtain:

$$\begin{aligned}\|A\widetilde{X}^{(k-1)}\|_F^2 + \|\widetilde{X}^{(k-1)}B\|_F^2 - 2\|A\widetilde{X}^{(k-1)}\|_F\|\widetilde{X}^{(k-1)}B\|_F &\le \|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\|_F^2\\ &\le (\|A\|_2 + \|B\|_2)^2\|\tilde{X}^{(k-1)}\|_F^2.\end{aligned}$$

According to Lemma 2.1, we have

$$\begin{aligned}(\sigma_{min}^2(A) + \sigma_{min}^2(B) - 2\|A\|_2\|B\|_2)\left\|\widetilde{X}^{(k-1)}\right\|_F^2 &\le \left\|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\right\|_F^2\\ &\le (\|A\|_2 + \|B\|_2)^2\left\|\widetilde{X}^{(k-1)}\right\|_F^2.\end{aligned} \tag{2.10}$$

Combining (2.8)–(2.10), it yields that

$$\begin{aligned}\|\widetilde{X}^{(k)}\|_F^2 =& \left\|\frac{\widetilde{X}_1^{(k)} + \widetilde{X}_2^{(k)}}{2} + \beta(\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-2)})\right\|_F^2\\ \le& 2\left\|\frac{\widetilde{X}_1^{(k)} + \widetilde{X}_2^{(k)}}{2}\right\|_F^2 + 2\beta^2\left\|\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-2)}\right\|_F^2\\ \le& \left\|\widetilde{X}_1^{(k)}\right\|_F^2 + \left\|\widetilde{X}_2^{(k)}\right\|_F^2 + 2\beta^2\left\|\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-2)}\right\|_F^2\\ \le& 2\left\|\widetilde{X}^{(k-1)}\right\|_F^2 - 2\mu\left\|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\right\|_F^2 + \mu^2(\|A\|_2^2 + \|B\|_2^2)\\ &\left\|A\widetilde{X}^{(k-1)} + \widetilde{X}^{(k-1)}B\right\|_F^2 + 2\beta^2\left\|\widetilde{X}^{(k-1)} - \widetilde{X}^{(k-2)}\right\|_F^2\\ \le& 2\left\|\widetilde{X}^{(k-1)}\right\|_F^2 - 2\mu\left(\sigma_{min}^2(A) + \sigma_{min}^2(B) - 2\|A\|_2\|B\|_2\right)\left\|\widetilde{X}^{(k-1)}\right\|_F^2\\ &+ \mu^2\left(\|A\|_2^2 + \|B\|_2^2\right)(\|A\|_2 + \|B\|_2)^2\left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 4\beta^2\left(\left\|\widetilde{X}^{(k-1)}\right\|_F^2 + \left\|\widetilde{X}^{(k-2)}\right\|_F^2\right)\\ =& [2 - 2\mu(\sigma_{min}^2(A) + \sigma_{min}^2(B) - 2\|A\|_2\|B\|_2) + \mu^2(\|A\|_2^2 + \|B\|_2^2)\\ &\cdot(\|A\|_2 + \|B\|_2)^2]\left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 4\beta^2\left\|\widetilde{X}^{(k-1)}\right\|_F^2 + 4\beta^2\left\|\widetilde{X}^{(k-2)}\right\|_F^2.\end{aligned} \tag{2.11}$$

By (2.11), we have

$$\begin{bmatrix}\|\widetilde{X}^{(k)}\|_F^2\\ \|\widetilde{X}^{(k-1)}\|_F^2\end{bmatrix} \le H\begin{bmatrix}\|\widetilde{X}^{(k-1)}\|_F^2\\ \|\widetilde{X}^{(k-2)}\|_F^2\end{bmatrix} \le H^{k-1}\begin{bmatrix}\|\widetilde{X}^{(1)}\|_F^2\\ \|\widetilde{X}^{(0)}\|_F^2\end{bmatrix},$$

where

$$H = \begin{bmatrix}q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2 & 4\beta^2\\ 1 & 0\end{bmatrix}.$$

Let λ be the eigenvalue of the matrix H. We know that

$$\lambda^2 - \lambda(q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2) - 4\beta^2 = 0.$$

It then follows from Lemma 2.2 that $|\lambda| < 1$ if and only if

$$\begin{cases} 4\beta^2 < 1, \\ \left|q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2\right| < 1 - 4\beta^2, \end{cases}$$

which implies that

$$\begin{cases} 0 < \beta < \dfrac{1}{2}, \\ -1 + 4\beta^2 < q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2 < 1 - 4\beta^2. \end{cases} \tag{2.12}$$

In addition, $H \geq 0$ ($H \geq 0$, if $h_{ij} \geq 0$ holds for all $1 < i < 2$, $1 < j < 2$.) if and only if

$$q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2 \geq 0. \tag{2.13}$$

Together with (2.12) and (2.13), we have

$$\begin{cases} 0 < \beta < \dfrac{1}{2}, \\ 0 \leq q_2\mu^2 - 2q_1\mu + 2 + 4\beta^2 < 1 - 4\beta^2. \end{cases} \tag{2.14}$$

In the following, we mainly solve the inequalities (2.14) to obtain the range of μ and β. The second inequality of (2.14) is equivalent to

$$\begin{cases} q_2\mu^2 - 2q_1\mu + 8\beta^2 + 1 < 0, \\ q_2\mu^2 - 2q_1\mu + 4\beta^2 + 2 \geq 0. \end{cases} \tag{2.15}$$

Consider (2.15) as a system of quadratic inequalities in terms of μ, and determine the range of μ to make the inequalities hold. Let's first solve the first inequality of (2.15). When $\Delta_1 = 4q_1^2 - 4q_2(8\beta^2+1) > 0$, i.e., $0 < \beta < \sqrt{\frac{q_1^2 - q_2}{8q_2}}$, there are two solutions $\frac{q_1 - \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2}$ and $\frac{q_1 + \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2}$ for the quadratic equation $q_2\mu^2 - 2q_1\mu + 8\beta^2 + 1 = 0$. So the solution of the first inequality in (2.15) is

$$\frac{q_1 - \sqrt{q_1^2 - q_2(8\beta^2 + 1)}}{q_2} < \mu < \frac{q_1 + \sqrt{q_1^2 - q_2(8\beta^2 + 1)}}{q_2}.$$

When $\Delta_1 \leq 0$, $q_2\mu^2 - 2q_1\mu + 8\beta^2 + 1$ is always greater than or equal to 0. So the inequality of $q_2\mu^2 - 2q_1\mu + 8\beta^2 + 1 < 0$ has no solution.

Solving the the second inequality of (2.15) by the same method in the following. When $\Delta_2 = 4q_1^2 - 4q_2(4\beta^2 + 2) \leq 0$, i.e., $\beta \geq \sqrt{\frac{q_1^2 - 2q_2}{4q_2}}$, $q_2\mu^2 - 2q_1\mu + 4\beta^2 + 2$ is always greater than or equal to 0. So the solution of the second inequality of (2.15) is $\mu \in R$; when $\Delta_2 > 0$, i.e., $0 < \beta < \sqrt{\frac{q_1^2 - 2q_2}{4q_2}}$, the solution of the second inequality of (2.15) is

$$\mu \leq \frac{q_1 - \sqrt{q_1^2 - q_2(4\beta^2 + 2)}}{q_2} \text{ or } \mu \geq \frac{q_1 + \sqrt{q_1^2 - q_2(4\beta^2 + 2)}}{q_2}.$$

In order to find the solution for (2.15), we need to consider the following cases:
Case 1: If $3q_2 - q_1^2 > 0$, then

$$\begin{cases} \sqrt{\dfrac{q_1^2 - 2q_2}{4q_2}} \le \beta < \sqrt{\dfrac{q_1^2 - q_2}{8q_2}}, \\ \dfrac{q_1 - \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2} < \mu < \dfrac{q_1 + \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2}, \end{cases} \tag{2.16}$$

or

$$\begin{cases} 0 < \beta < \sqrt{\dfrac{q_1^2 - 2q_2}{4q_2}}, \\ \dfrac{q_1 - \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2} < \mu \le \dfrac{q_1 - \sqrt{q_1^2 - q_2(4\beta^2+2)}}{q_2} \ or \\ \dfrac{q_1 + \sqrt{q_1^2 - q_2(4\beta^2+2)}}{q_2} \le \mu < \dfrac{q_1 + \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2}. \end{cases} \tag{2.17}$$

Case 2: If $3q_2 - q_1^2 \le 0$, then

$$\begin{cases} 0 < \beta < \sqrt{\dfrac{q_1^2 - q_2}{8q_2}}, \\ \dfrac{q_1 - \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2} < \mu \le \dfrac{q_1 - \sqrt{q_1^2 - q_2(4\beta^2+2)}}{q_2} \ or \\ \dfrac{q_1 + \sqrt{q_1^2 - q_2(4\beta^2+2)}}{q_2} \le \mu < \dfrac{q_1 + \sqrt{q_1^2 - q_2(8\beta^2+1)}}{q_2}. \end{cases} \tag{2.18}$$

Together with the first inequality of (2.14) and (2.16)–(2.18), (2.4)–(2.6) are obtained. Thus, the proof is completed. □

Remark 4. *When the error iteration matrix of the proposed algorithm is of size 2×2, the idea of using Lemma 2.2 to determine the range of the coefficients in quadratic equations, thereby ensuring the convergence of the algorithm, has been widely applied in many literatures. For example, the SOR-like methods for solving the absolute value equations [11, 15].*

3. The APGI and AGMI algorithms

In this section, explicitly giving the varied parameters in the proposed algorithms by minimizing the residual in every iteration, the PGI and GMI algorithms with adaptive parameters are constructed.

3.1. The adaptive PGI algorithm

We first present the calculation rule for the parameter used in Algorithm 2 by minimizing the current residual norm. The details are described below:

Suppose the parameter μ_k is taken in Algorithm 2 and for $k = 1, 2, 3, \cdots$, the previous $k-1$ residuals are defined by

$$R^{(k-1)} = C - AX^{(k-1)} - X^{(k-1)}B, \tag{3.1}$$

then the PGI algorithm can be simply rewritten as

$$X^{(k)} = X^{(k-1)} + \frac{\mu_k}{2}P^{-1}A^T R^{(k-1)} + \frac{\mu_k}{2}R^{(k-1)}B^T Q^{-1}. \tag{3.2}$$

According to (3.1) and (3.2), the kth residual is given by

$$R^{(k)} = R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)} \tag{3.3}$$

with $M^{(k-1)} = P^{-1}A^T R^{(k-1)}B + R^{(k-1)}B^T Q^{-1}B + AP^{-1}A^T R^{(k-1)} + AR^{(k-1)}B^T Q^{-1}$.
Taking the F-norm on both sides of (3.3), it holds that

$$\begin{aligned} \|R^{(k)}\|_F^2 =& tr[(R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)})^T(R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)})] \\ =& \|R^{(k-1)}\|_F^2 - \mu_k tr((M^{(k-1)})^T R^{(k-1)}) + \frac{\mu_k^2}{4}\|M^{(k-1)}\|_F^2. \end{aligned}$$

Let $\phi(\mu_k) = \|R^{(k)}\|_F^2$. The first-order derivative of $\phi(\mu_k)$ yields

$$\frac{\partial \phi}{\partial \mu_k} = \frac{\mu_k}{2}\|M^{(k-1)}\|_F^2 - tr((M^{(k-1)})^T R^{(k-1)}).$$

It is easy to see that the unique stationary point of the function $\phi(\mu_k)$ is

$$\mu_k = \frac{2tr((M^{(k-1)})^T R^{(k-1)})}{\|M^{(k-1)}\|_F^2}. \tag{3.4}$$

It is obvious that the sencond-order derivative of $\phi(\mu_k)$ i.e., $\frac{\partial^2 \phi}{\partial \mu_k^2} = \frac{1}{2}\|M^{(k-1)}\|_F^2 > 0$, which implies that the stationary point mentioned in (3.4) is the unique minimum point of the function $\phi(\mu_k)$.

Through the above arrangement, we formally outline the APGI method in Algorithm 4.

Algorithm 4 The APGI algorithm.

Require: Given two preconditioners P and Q, an initial matrix $X^{(0)}$ and the parameter μ_1
Ensure: $X^{(k)}$

1: **For** $k = 1, 2, \cdots$, **until converges, do**
2: $X_1^{(k)} = X^{(k-1)} + \mu_k P^{-1}A^T[C - AX^{(k-1)} - X^{(k-1)}B]$,
3: $X_2^{(k)} = X^{(k-1)} + \mu_k[C - AX^{(k-1)} - X^{(k-1)}B]B^T Q^{-1}$,
4: $X^{(k)} = [X_1^{(k)} + X_2^{(k)}]/2$,
5: according to (3.4), compute μ_{k+1}.
6: **End**

Remark 5. *If $P = I_m$ and $Q = I_n$ are adopted, the APGI algorithm is reduced to AGI algorithm.*

3.2. The adaptive GMI algorithm

The kth iteration of the GMI algorithm can be simply rewritten as:

$$X^{(k)} = X^{(k-1)} + \frac{\mu_k}{2}A^T R^{(k-1)} + \frac{\mu_k}{2}R^{(k-1)}B^T + \beta_k(X^{(k-1)} - X^{(k-2)}).$$

The residual of the *kth* iteration is

$$R^{(k)} = R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)} + \beta_k N^{(k-1)} \tag{3.5}$$

with $M^{(k-1)} = A^T R^{(k-1)}B + R^{(k-1)}B^T B + AA^T R^{(k-1)} + AR^{(k-1)}B^T$, and $N^{(k-1)} = R^{(k-1)} - R^{(k-2)}$. Taking the F-norm on both sides of (3.5), it follows that

$$\begin{aligned}
\|R^{(k)}\|_F^2 &= tr[(R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)} + \beta_k N^{(k-1)})^T(R^{(k-1)} - \frac{\mu_k}{2}M^{(k-1)} + \beta_k N^{(k-1)})] \\
&= \|R^{(k-1)}\|_F^2 - \mu_k tr((M^{(k-1)})^T R^{(k-1)}) + 2\beta_k tr((N^{(k-1)})^T R^{(k-1)}) \\
&- \beta_k\mu_k tr((M^{(k-1)})^T N^{(k-1)}) + \frac{\mu_k^2}{4}\|M^{(k-1)}\|_F^2 + \beta_k^2\|N^{(k-1)}\|_F^2.
\end{aligned}$$

Let $\psi(\mu_k, \beta_k) = \|R^{(k)}\|_F^2$. The first-order derivative of $\psi(\mu_k, \beta_k)$ yields

$$\begin{cases}
\dfrac{\partial\psi}{\partial\mu_k} = \dfrac{\mu_k}{2}\|M^{(k-1)}\|_F^2 - tr((M^{(k-1)})^T R^{(k-1)}) - \beta_k tr((M^{(k-1)})^T N^{(k-1)}), \\
\dfrac{\partial\psi}{\partial\beta_k} = 2\beta_k\|N^{(k-1)}\|_F^2 + 2tr((N^{(k-1)})^T R^{(k-1)}) - \mu_k tr((M^{(k-1)})^T N^{(k-1)}).
\end{cases}$$

It is easy to see that the unique stationary point of the function $\psi(\mu_k, \beta_k)$ is

$$\begin{cases}
\mu_k = \dfrac{2a_{k-1}e_{k-1} - 2b_{k-1}c_{k-1}}{d_{k-1}e_{k-1} - b_{k-1}^2}, \\
\beta_k = \dfrac{b_{k-1}a_{k-1} - c_{k-1}d_{k-1}}{d_{k-1}e_{k-1} - b_{k-1}^2},
\end{cases} \tag{3.6}$$

where $a_{k-1} = tr((M^{(k-1)})^T R^{(k-1)})$, $b_{k-1} = tr((M^{(k-1)})^T N^{(k-1)})$, $c_{k-1} = tr((N^{(k-1)})^T R^{(k-1)})$, $d_{k-1} = \|M^{(k-1)}\|_F^2$, $e_{k-1} = \|N^{(k-1)}\|_F^2$. We also know that the sencond-order derivative of the function $\psi(\mu_k, \beta_k)$ is

$$\frac{\partial^2\psi}{\partial\mu_k^2} = \frac{1}{2}\|M^{(k-1)}\|_F^2, \quad \frac{\partial^2\psi}{\partial\mu_k\partial\beta_k} = \frac{\partial^2\psi}{\partial\beta_k\partial\mu_k} = -tr((M^{(k-1)})^T N^{(k-1)}), \quad \frac{\partial^2\psi}{\partial\beta_k^2} = 2\|N^{(k-1)}\|_F^2,$$

whose Hessian matrix of the function $\psi(\mu_k, \beta_k)$ at the stationary point is

$$\begin{pmatrix}
\frac{1}{2}\|M^{(k-1)}\|_F^2 & -tr((M^{(k-1)})^T N^{(k-1)}) \\
-tr((M^{(k-1)})^T N^{(k-1)}) & 2\|N^{(k-1)}\|_F^2
\end{pmatrix}.$$

It is obvious that the Hessian matrix is symmetric positive definite, which implies that the stationary point mentioned in (3.6) is the unique minimum point of the function $\psi(\mu_k, \beta_k)$.

According to the above explanation, the AGMI algorithm is formulated as described in Algorithm 5.

Algorithm 5 The AGMI algorithm.

Require: Given two initial approximate matrices $X^{(0)}$ and $X^{(1)}$, and two parameters μ_2 and β_2
Ensure: $X^{(k)}$

1: **For** $k = 2, 3, 4 \cdots$, **until converges, do**
2: $X_1^{(k)} = X^{(k-1)} + \mu_k A^T[C - AX^{(k-1)} - X^{(k-1)}B]$,
3: $X_2^{(k)} = X^{(k-1)} + \mu_k[C - AX^{(k-1)} - X^{(k-1)}B]B^T$,
4: $X^{(k)} = [X_1^{(k)} + X_2^{(k)}]/2 + \beta_k(X^{(k-1)} - X^{(k-2)})$,
5: according to (3.6), compute μ_{k+1} and β_{k+1}.
6: **End**

4. Numerical results

In this section, some examples are illustrated to verify the efficiencies of the proposed PGI, GMI, APGI, and AGMI algorithms compared with the GI [8], RGI [18], AGBI [25], AJGI [21], HSS [3], and NPHSS [16] algorithms. All examples are performed under MATLAB on a personal computer with a 1.61 GHz central processing unit (Intel(R) Core(TM) i7-10710) and 16GB memory.

The number of the iteration steps (denoted by IT), the computing time in seconds (denoted by CPU), and the relative residual norm (denoted by RRN) are listed in the tables below. All the initial matrices are set to be zero matrices, and the iterations are stopped if the RRN in the current step satisfies

$$RRN := \frac{\|C - AX^{(k)} - X^{(k)}B\|}{\|C - AX^{(0)} - X^{(0)}B\|} \leq 10^{-6},$$

or the numbers of iteration steps exceeds 10000.

Example 1. Consider Eq (1.1) with the matrices A and B defined by

$$\begin{cases} A = \text{diag}(1, 2, \cdots, n) + rL^T, \\ B = 2^{-t}I_n + \text{diag}(1, 2, \cdots, n) + rL^T + 2^{-t}L \end{cases}$$

with L is the strictly lower triangular matrix having ones in the lower triangle part, $r = 2$ and $t = \frac{1}{2}$. The right-hand side $C = AX + XB$ with $X = ones(n)$, where $ones$ is a MATLAB built-in function.

The numerical results of the tested algorithms for Example 1 are listed in Table 2, and the corresponding error convergence curves are shown in Figure 1. The matrices P and Q in the PGI and APGI algorithms are the identity matrices, so the PGI and APGI algorithms are just the GI and AGI algorithms. The parameters in AGBI (Algorithm 2.4 in [25]), GI ((13)–(15) in [8]), RGI (Algorithm 1 in [18]), and GMI algorithms are experimentally optimal, which are denoted by μ_{exp} and β_{exp} in Table 1. Like [18] and [25], the relaxation parameters in RGI and AJBI algorithms are both 0.5. Figure 1 shows the RRN of the AGBI, GI, AGI, GMI, and AGMI algorithms with $n = 200$. From the figure, it can be observed that the AGMI algorithm performs best among all these algorithms.

Table 1. The experimental optimal parameters for Example 1.

Algorithms		100	200	300	400
GI	μ_{exp}	9.713E-06	2.424E-06	1.077E-06	6.057E-07
RGI	μ_{exp}	2.356E-05	5.879E-06	2.612E-06	2.120E-06
AGBI	μ_{exp}	0.390E-04	0.901E-05	3.790E-06	8.500E-06
GMI	μ_{exp}	2.428E-05	6.062E-06	2.692E-06	1.514E-06
	β_{exp}	6.000E-01	6.000E-01	6.000E-01	6.000E-01

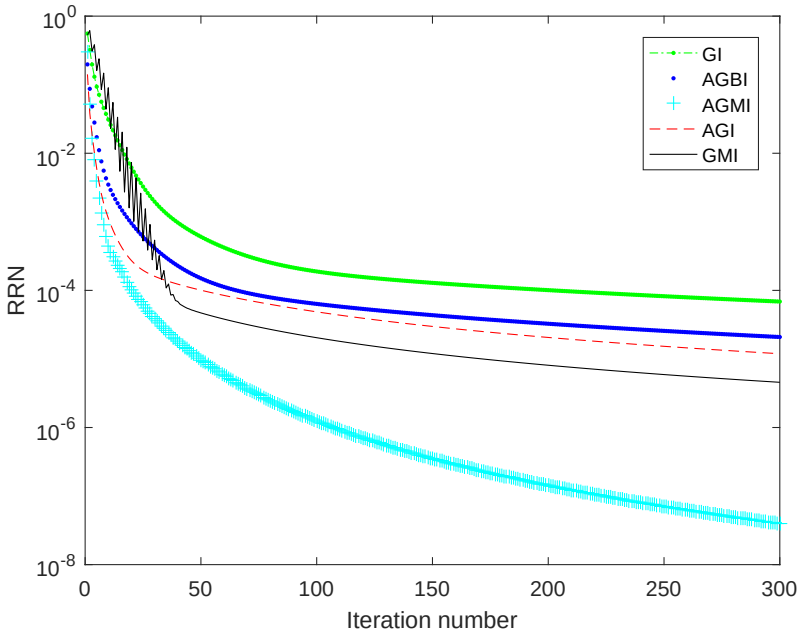

Figure 1. Convergence curves of different algorithms for Example 1.

From Table 2, compared with the GI, RGI, AGBI, and AGI algorithms, the GMI and AGMI algorithms have more effectiveness in terms of IT and CPU time. In addition, since the parameters μ_k and β_k in AGI and AGMI algorithms are varied and adaptive in each iteration, the two algorithms are more efficient than the GI and GMI algorithms, respectively.

Table 2. Numerical results of different algorithms for Example 1.

Algorithms		100	200	300	400
GI	IT	5413	5235	5174	5142
	CPU	3.349	14.091	39.421	116.627
	RRN	9.997E-07	9.999E-07	9.997E-07	9.999E-07
RGI	IT	4464	4318	4267	4241
	CPU	2.905	11.317	39.024	105.797
	RRN	9.997E-07	9.998E-07	9.998E-07	9.999E-07
AGBI	IT	2772	2879	2992	2985
	CPU	2.218	10.852	34.857	88.261
	RRN	9.996E-07	9.998E-07	9.995E-07	9.997E-07
AGI	IT	1681	1627	1608	1598
	CPU	2.212	9.114	31.939	86.779
	RRN	9.996E-07	9.992E-07	9.992E-07	9.995E-07
GMI	IT	864	836	826	821
	CPU	0.371	1.504	5.739	13.950
	RRN	9.993E-07	9.986E-07	9.988E-07	9.987E-07
AGMI	IT	94	93	92	91
	CPU	0.162	0.622	2.135	5.891
	RRN	9.785E-07	9.744E-07	9.753E-07	9.948E-07

Example 2. The matrices A and B in Eq (1.1) are given as

$$A = \begin{pmatrix} 10 & 1 & 1 & \cdots & 1 & 1 \\ 2 & 10 & 1 & \cdots & 1 & 1 \\ 1 & 2 & 10 & \cdots & 1 & 1 \\ \vdots & \vdots & & \ddots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 2 & 10 \end{pmatrix}, \quad B = \begin{pmatrix} 8 & 1 & 1 & \cdots & 1 & 1 \\ 3 & 8 & 1 & \cdots & 1 & 1 \\ 1 & 3 & 8 & \cdots & 1 & 1 \\ \vdots & \vdots & & \ddots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 3 & 8 \end{pmatrix}.$$

The right-hand side $C = AX + XB$ with $X = ones(n)$.

The numerical results of the tested algorithms for Example 2 are listed in Table 3, and the corresponding error convergence curves are shown in Figure 2. The matrices P and Q in the PGI and APGI algorithms are the diagonal parts of the matrices A and B, respectively. The experimentally optimal parameters contained in GI (see (13)–(15) in [8]), HSS ("The HSS Iteration Method" in [3]), NPHSS (Algorithm 1 in [16]), PGI, and GMI algorithms are given in Table 4. From Table 3, it is clear that the seven algorithms are convergent for all cases. Furthermore, the APGI and AGMI algorithms need fewer iteration numbers and CPU time than other algorithms, and the AGMI algorithm performs best among the seven algorithms. Figure 2 illustrates the convergence curves of the HSS, NPHSS, PGI, GMI, APGI, and AGMI algorithms for the case $n = 256$. It is clear that the RRN of the AGMI and NPHSS algorithms rapidly decreases below 10^{-13}, but NPHSS needs much more CPU time. So, AGMI is the most efficient one.

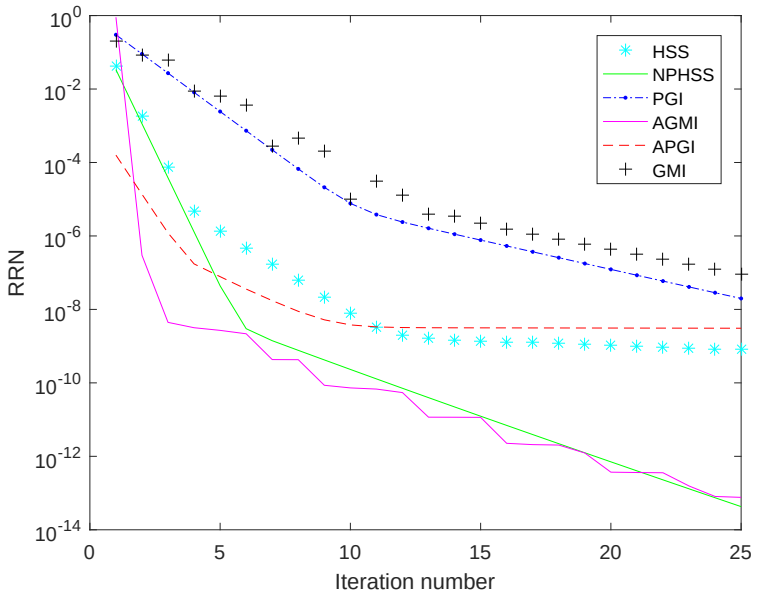

Figure 2. Convergence curves of different algorithms for Example 2.

Table 3. Numerical results of different algorithms for Example 2.

Algorithms		128	256	512	1024
GI	IT	43	38	35	31
	CPU	0.047	0.213	1.216	7.249
	RRN	9.362E-07	9.739E-07	9.743E-07	9.345E-07
HSS	IT	6	6	5	4
	CPU	0.344	1.817	8.361	46.524
	RRN	1.917E-07	4.749E-07	6.788E-07	6.678E-07
NPHSS	IT	5	5	4	3
	CPU	0.054	0.321	2.154	14.542
	RRN	6.137E-07	4.381E-08	3.839E-08	4.305E-07
PGI	IT	17	15	13	12
	CPU	0.026	0.106	0.588	3.919
	RRN	6.848E-07	7.719E-07	6.914E-07	6.468E-07
GMI	IT	22	18	19	18
	CPU	0.021	0.069	0.474	2.995
	RRN	8.216E-07	8.211E-07	7.853E-07	8.068E-07
APGI	IT	4	4	3	3
	CPU	0.023	0.084	0.337	2.365
	RRN	9.739E-07	1.733E-07	4.279E-07	1.525E-07
AGMI	IT	3	3	3	3
	CPU	0.029	0.064	0.247	1.961
	RRN	1.228E-07	1.204E-08	4.862E-09	1.592E-09

Table 4. The experimental optimal parameters for Example 2.

Algorithms		128	256	512	1024
GI	μ_{exp}	1.323E-05	3.547E-06	8.273E-07	1.872E-07
HSS	α_1	1.329E+02	3.229E+02	6.462E+02	1.091E+03
	α_2	1.046E+02	2.548E+02	5.104E+02	8.624E+02
NPHSS	α_1	2.248E+00	2.499E+00	1.999E+00	2.125E+00
	α_2	1.439E+01	1.599E+01	1.279E+01	1.359E+01
PGI	μ_{exp}	3.059E-04	8.201E-05	2.125E-05	5.409E-06
GMI	μ_{exp}	1.984E-05	5.675E-06	1.195E-06	2.575E-07
	β_{exp}	1.490E-01	1.550E-01	1.750E-01	1.850E-01

Example 3. The matrices in Eq (1.1) are described as

$$A = B = M + 2N + \frac{100}{(n+1)^2} I_n,$$

where $M = \text{tridiag}(-1, 2.6, -1)$ and $N = \text{tridiag}(0.5, 0, -0.5)$. The right-hand side $C = AX + XB$ with $X = ones(n)$.

The numerical results of the tested algorithms for Example 3 are listed in Table 6, and the corresponding error convergence curves are shown in Figure 3. Let P and Q be the tridiagonal parts of the matrices A^TA and BB^T, respectively. The experimentally optimal parameters contained in GI ((13)–(15) in [8]), AJGI (Algorithm 5 in [21]), PGI, and GMI algorithms are given in Table 5. Like [21], the relaxation parameters ω_1 and ω_2 in the AJGI algorithm are 0.5 and 3, respectively. From Table 6 it follows that all the algorithms are effective for this example. In addition, the APGI algorithm is the most efficient one among the six algorithms. Figure 3 shows their convergence performances for the case $n = 256$, and the APGI algorithm has the remarkably best convergence result.

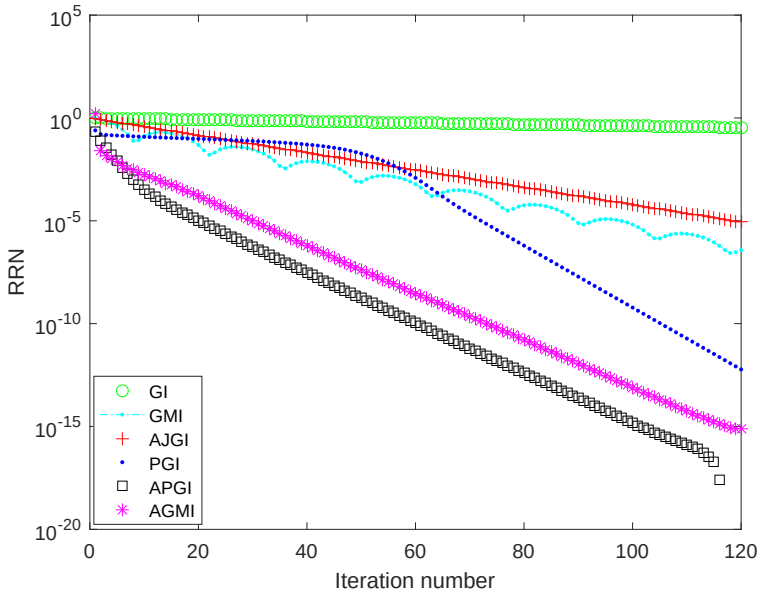

Figure 3. Convergence curves of different algorithms for Example 3.

Table 5. The experimental optimal parameters for Example 3.

Algorithms		128	256	512	1024
GI	μ_{exp}	4.714E-02	4.723E-02	4.725E-02	4.726E-02
AJGI	μ_{exp}	2.400E-02	2.400E-02	2.300E-02	2.300E-02
GMI	μ_{exp}	8.800E-02	8.300E-02	8.700E-02	8.800E-02
	β_{exp}	8.700E-01	8.700E-01	8.700E-01	8.700E-01
PGI	μ_{exp}	4.400E-01	4.200E-01	3.900E-01	3.900E-01

Table 6. Numerical results of different algorithms for Example 3.

Algorithms		128	256	512	1024
GI	IT	398	397	398	399
	CPU	0.429	2.085	14.791	117.676
	RRN	9.954E-07	9.703E-07	9.822E-07	9.751E-07
AJGI	IT	180	183	185	185
	CPU	0.166	0.941	5.978	51.097
	RRN	9.129E-07	9.161E-07	8.603E-07	8.928E-07
GMI	IT	190	186	182	181
	CPU	0.211	0.799	6.713	42.489
	RRN	8.229E-07	7.694E-07	7.376E-07	6.119E-07
PGI	IT	96	95	95	109
	CPU	0.209	0.768	5.23	46.233
	RRN	8.448E-07	7.775E-07	9.507E-07	9.342E-07
AGMI	IT	51	50	48	47
	CPU	0.138	0.642	4.217	38.757
	RRN	7.780E-07	8.138E-07	9.664E-07	8.567E-07
APGI	IT	30	28	26	24
	CPU	0.093	0.437	2.802	24.721
	RRN	9.078E-07	9.643E-07	9.197E-07	8.501E-07

5. Conclusions

In this paper, we provide two accelerated GI algorithms for solving Eq (1.1), which are the preconditioned gradient-based iteration (PGI) algorithm and the gradient-based momentum iteration (GMI) algorithm, respectively. Convergence analyses show that the proposed algorithms converge to the exact solution for any initial value with some assumptions. Moreover, the adaptive PGI and GMI algorithms are also established, and the adaptive parameters can be computed by minimizing the residual norms in the corresponding algorithms. Numerical experiments illustrate the excellent performances of our proposed algorithms. In addition, how to use the APGI and AGMI algorithms for solving other matrix equations will be investigated in our future work.

Use of Generative-AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Author contributions

Huiling Wang: gave the algorithms proposed in the manuscript, provided the numerical results and wrote the original draft of the manuscript; Nian-Ci Wu and Yufeng Nie: gave the clear guidance on the proof of the theorem and polished the language of the entire manuscript. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (12201651), the Fundamental Research Funds for the Central Universities, South Central Minzu University (CZQ23004), and the Research Project Supported by Shanxi Scholarship Council of China(2023-117).

Conflict of interest

The authors declare no conflicts of interest.

References

1. A. L. Andrew, Eigenvectors of certain matrices, *Linear Algebra Appl.*, **7** (1973), 151–162. http://dx.doi.org/10.1016/0024-3795(73)90049-9
2. Z. Z. Bai, G. H. Golub, J. Y. Pan, Preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems, *Numer. Math.*, **98** (2004), 1–32. http://dx.doi.org/10.1007/s00211-004-0521-1
3. Z. Z. Bai, On hermitian and skew-hermitian splitting iteration methods for continuous sylvester equations, *J. Comput. Math.*, **29** (2011), 185–198. http://dx.doi.org/10.4208/jcm.1009-m3152
4. Z. Z. Bai, M. Benzi, F. Chen, Z. Q. Wang, Preconditioned MHSS iteration methods for a class of block two-by-two linear systems with applications to distributed control problems, *IMA J. Numer. Anal.*, **33** (2013), 343–369. http://dx.doi.org/10.1093/imanum/drs001
5. A. Bhaya, E. Kaszkurewicz, Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method, *Neural Networks*, **17** (2004), 65–71. http://dx.doi.org/10.1016/S0893-6080(03)00170-9
6. Z. B. Chen, X. S. Chen, Conjugate gradient-based iterative algorithm for solving generalized periodic coupled Sylvester matrix equation, *J. Franklin I.*, **359** (2022), 9925–9951. http://dx.doi.org/10.1016/j.jfranklin.2022.09.049
7. M. Dehghan, A. Shirilord, The double-step scale splitting method for solving complex Sylvester matrix equation, *Comp. Appl. Math.*, **38** (2019), 146. http://dx.doi.org/10.1007/s40314-019-0921-6

8. F. Ding, T. W. Chen, Gradient based iterative algorithms for solving a class of matrix equations, *IEEE T. Automat. Contr.*, **50** (2005), 1216–1221. http://dx.doi.org/10.1109/TAC.2005.852558

9. F. Ding, P. X. Liu, J. Ding, Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle, *Appl. Math. Comput.*, **197** (2008), 41–50. http://dx.doi.org/10.1016/j.amc.2007.07.040

10. F. Ding, X. H. Wang, Q. J. Chen, Y. S. Xiao, Recursive least squares parameter estimation for a class of output nonlinear systems based on the model decompositions, *Circuits Syst. Signal Process.*, **35** (2016), 3323–3338. http://dx.doi.org/10.1007/s00034-015-0190-6

11. X. Dong, X. H. Shao, H. L. Shen, A new SOR-like method for solving absolute value equations, *Appl. Numer. Math.*, **156** (2020), 410–421. http://dx.doi.org/10.1016/j.apnum.2020.05.013

12. K. Du, C. C. Ruan, X. H. Sun, On the convergence of a randomized block coordinate descent algorithm for a matrix least squares problem, *Appl. Math. Lett.*, **124** (2022), 107689. http://dx.doi.org/10.1016/j.aml.2021.107689

13. W. Fan, C. Gu, Z. Tian, Jacobi-gradient iterative algorithms for Sylvester matrix equations, *14th Conference of the International Linear Algebra Society*, Shanghai, China, 2007, 16–20.

14. C. Q. Gu, H. Y. Xue, A shift-splitting hierarchical identification method for solving Lyapunov matrix equations, *Linear Algebra Appl.*, **430** (2009), 1517–1530. http://dx.doi.org/10.1016/j.laa.2008.01.010

15. B. H. Huang, W. Li, A modified SOR-like method for absolute value equations associated with second order cones, *J. Comput. Appl. Math.*, **400** (2022), 113745. http://dx.doi.org/10.1016/j.cam.2021.113745

16. X. Li, H. F. Huo, A. L. Yang, Preconditioned HSS iteration method and its non-alternating variant for continuous Sylvester equations, *Comput. Math. Appl.*, **75** (2018), 1095–1106. http://dx.doi.org/10.1016/j.camwa.2017.10.028

17. M. S. Mehany, Q. W. Wang, Three symmetrical systems of coupled Sylvester-like quaternion matrix equations, *Symmetry*, **14** (2022), 550. http://dx.doi.org/10.3390/sym14030550

18. Q. Niu, X. Wang, L. Z. Lu, A relaxed gradient based algorithm for solving Sylvester equations, *Asian J. Control*, **13** (2011), 461–464. http://dx.doi.org/10.1002/asjc.328

19. B. T. Polyak, Some methods of speeding up the convergence of iteration methods, *Comp. Math. Math. Phys.*, **4** (1964), 1–17. http://dx.doi.org/10.1016/0041-5553(64)90137-5

20. S. G. Shafiei, M. Hajarian, An iterative method based on ADMM for solving generalized Sylvester matrix equations, *J. Franklin I.*, **359** (2022), 8155–8170. http://dx.doi.org/10.1016/j.jfranklin.2022.07.049

21. Z. L. Tian, M. Y. Tian, C. Q. Gu, X. N. Hao, An accelerated Jacobi-gradient based iterative algorithm for solving Sylvester matrix equations, *Filomat*, **31** (2017), 2381–2390. http://dx.doi.org/10.2298/FIL1708381T

22. Z. L. Tian, Y. D. Wang, Y. H. Dong, S. Y. Wang, New results of the IO iteration algorithm for solving Sylvester matrix equation, *J. Franklin I.*, **359** (2022), 8201–8217. http://dx.doi.org/10.1016/j.jfranklin.2022.08.018

23. Q. W. Wang, R. Y. Lv, Y. Zhang, The least-squares solution with the least norm to a system of tensor equations over the quaternion algebra, *Linear Multilinear A.*, **70** (2022), 1942–1962. http://dx.doi.org/10.1080/03081087.2020.1779172

24. Q. W. Wang, X. Wang, A system of coupled two-sided Sylvester-type tensor equations over the quaternion algebra, *Taiwanese J. Math.*, **24** (2020), 1399–1416. http://dx.doi.org/10.11650/tjm/200504

25. Y. J. Xie, 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. http://dx.doi.org/10.1016/j.amc.2015.07.022

26. A. L. Yang, Y. Cao, Y. J. Wu, Minimum residual Hermitian and skew-Hermitian splitting iteration method for non Hermitian positive definite linear systems, *BIT Numer. Math.*, **59** (2019), 299–319. http://dx.doi.org/10.1007/s10543-018-0729-6

27. A. L. Yang, On the convergence of the minimum residual HSS iteration method, *Appl. Math. Lett.*, **94** (2019), 210–216. http://dx.doi.org/10.1016/j.aml.2019.02.031

28. J. F. Yin, Q. Y. Dou, Generalized preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive-definite linear systems, *J. Comput. Math.*, **30** (2012), 404–417. http://dx.doi.org/10.4208/jcm.1201-m3209

29. X. F. Zhang, Q. W. Wang, Developing iterative algorithms to solve Sylvester tensor equations, *Appl. Math. Comput.*, **409** (2021), 126403. http://dx.doi.org/10.1016/j.amc.2021.126403

AIMS Press