Filomat 39:11 (2025), 3789–3803
https://doi.org/10.2298/FIL2511789W

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

AJGI iterative algorithm for solving coupled Sylvester matrix equations with one side

Wenli Wang[a], Caiqin Song[a,*]

[a] *School of Mathematical Sciences, University of Jinan, Jinan 250022, PR China*

Abstract. This paper considers the coupled Sylvester matrix equations with one side, which have many important applications in control theory and system theory. Based on the Jacobi iterative method and the hierarchical identification principle, the Jacobi-gradient based iterative algorithm and the accelerated Jacobi-gradient based iterative algorithm are constructed. It is theoretically proved that the presented algorithms are convergent for any initial matrix under appropriate conditions, and numerical examples are given to show that the presented iterative algorithms are faster than some existing iterative algorithms. Moreover, the application of the accelerated Jacobi-gradient based iterative algorithm in dynamical systems is presented.

1. Introduction

Solving linear or nonlinear matrix equations involves many fields, such as system theory [7, 15, 26], control theory [32], signal processing [8], image restoration [38] and so on. For instance, the eigenstructure assignment problem for the second-order linear system [7]

$$\ddot{q} - A\dot{q} - Cq = Bu, \tag{1.1}$$

via controller

$$\begin{cases} u = -K_0 q - K_1 \dot{q} = -Kx, \\ K = [K_0 \ K_1], \end{cases}$$

can be solved by finding matrices V and K satisfying matrix equation

$$(A^{'} - B^{'}K)V = V\Lambda. \tag{1.2}$$

2020 *Mathematics Subject Classification*. Primary 15A24; Secondary 65F10, 15A21.
Keywords. Coupled Sylvester matrix equations; Jacobi-gradient based iterative algorithm; Accelerated Jacobi-gradient based iterative algorithm; Convergence performance.
Received: 05 August 2022; Revised: 16 September 2023; Accepted: 26 January 2025
Communicated by Marko Petković
The work is supported by the Fundamental Research Funds for the Central Universities (No. YA24YJS00040).
* Corresponding author: Caiqin Song
Email addresses: wenliwang_cba@163.com (Wenli Wang), songcaiqin1983@163.com (Caiqin Song)
ORCID iDs: https://orcid.org/0000-0002-7108-6388 (Wenli Wang), https://orcid.org/0000-0002-2632-5351 (Caiqin Song)

Therefore, solving various matrix equations has become an important research hotpot in recent years [3, 10, 13, 16, 17, 19, 34]. For example, the conjugate direction (CD) method was developed to solve the generalized nonhomogeneous Yakubovich-transpose matrix equation [15]. The Combined Real part and Imaginary part (CRI) method was presented to solve the generalized Lyapunov matrix equation [23]. The Lanczos version of the biconjugate residual algorithm was studied to solve the reflexive or anti-reflexive solutions of a class of generalized coupled Sylvester matrix equations [36].

For the matrix equation with smaller coefficient matrix, the exact solution can be obtained quickly and effectively. However, if the dimension of coefficient matrix is large, the direct method cannot solve the problem well. In order to overcome this difficulty, numerous iterative methods are presented to solve the matrix equations [2, 4, 11, 12, 14, 24, 25, 28–30, 37]. For example, Wu et al. [33] presented a gradient based iterative (GI) algorithm for solving a class of complex conjugate and transpose matrix equation

$$\sum_{l=1}^{s_1} A_l X B_l + \sum_{l=1}^{s_2} C_l \overline{X} D_l + \sum_{l=1}^{s_3} G_l X^T H_l + \sum_{l=1}^{s_4} M_l X^H N_l = F, \tag{1.3}$$

and left a conjecture that the sufficient condition is also the necessary condition for the convergence. To verify this conjecture, Zhang et al. [39] constructed the GI algorithm for solving the complex matrix equation

$$AXB + C\overline{X}D + GX^T H + MX^H N = F, \tag{1.4}$$

which is a special case of (1.3), and obtained the necessary and sufficient conditions for convergence of the GI algorithm. Based on the above work, Wang et al. [31] investigated a relaxed gradient based iterative (RGI) algorithm for solving (1.4) and verified that the RGI algorithm is more efficient than the GI algorithm. As an important and effective numerical algorithm, the RGI algorithm is one of the variants of the GI algorithm originally proposed by Ding et al. [5, 6] and has been studied for solving other significant matrix equations [18, 20–22]. Besides, other efficient variations of the GI algorithm include the accelerated GI algorithm [1, 35], the Jacobi GI algorithm [9], the accelerated Jacobi GI algorithm [27] and so on. While the methods proposed in [9, 27] successfully obtain sufficient condition for the convergence of the algorithm, it is a hard task to compute the appropriate parameter value μ to satisfy the sufficient condition and they do not consider the sufficient and necessary conditions for the convergence of the algorithm. Therefore, this fact further encourages us to explore the appropriate parameter value μ that can be easily obtained to satisfy the sufficient and necessary conditions for the convergence of the algorithm.

In this paper, we consider the numerical solutions of the coupled Sylvester matrix equations

$$\begin{cases} AX + XB = C, \\ DX + XE = F, \end{cases} \tag{1.5}$$

where $A, D \in \mathbb{R}^{m\times m}$, $B, E \in \mathbb{R}^{n\times n}$, $C, F \in \mathbb{R}^{m\times n}$ are given constant matrices, and $X \in \mathbb{R}^{m\times n}$ is the unknown matrix to be solved. Inspired by [1, 5, 6, 9, 18, 20–22, 27, 28, 30, 31, 33, 35, 39], the Jacobi-gradient based iterative (JGI) algorithm and the accelerated Jacobi-gradient based iterative (AJGI) algorithm for solving (1.5) are constructed by applying the Jacobi iterative method and the hierarchical identification principle, which have not been studied to solve the coupled Sylvester matrix equations (1.5) to our knowledge. This provides a variety of options for the numerical algorithms to solve (1.5). In the process of constructing the algorithms, we extract the diagonal parts of coefficient matrices and use them to construct iterative formula. In this case, the studied algorithms can greatly reduce the storage space and thus improve the operational efficiency.

Throughout this paper, we use the following notations. Let $\mathbb{R}^{m\times n}$ denotes the set of $m\times n$ real matrix. For $A \in \mathbb{R}^{m\times n}$, we use A^T and $\rho(A)$ to represent the transpose and the spectral radius of the matrix A, respectively. For any $A = (a_{ij}) \in \mathbb{R}^{m\times n}$, $B = (b_{ij}) \in \mathbb{R}^{s\times t}$, $A \otimes B = (a_{ij}B) \in \mathbb{R}^{ms\times nt}$ denotes the Kronecker product of two matrices. For a matrix $X = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^{m\times n}$, the vector stretching function $vec(\cdot) : X \to vec(X)$ is defined as $vec(X) = (x_1^T, x_2^T, \cdots, x_n^T)^T \in \mathbb{R}^{mn}$. The symbol I_n represents the identity matrix of size $n \times n$. The

spectral norm of the matrix A is denoted by $\|A\|_2 = \sqrt{\lambda_{max}(A^TA)} = \sigma_{max}(A)$ and the Frobenious norm of the matrix A is denoted by $\|A\| = \sqrt{tr(A^TA)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$.

The rest of this paper is arranged as follows. In Section 2, we investigate the JGI algorithm and the AJGI algorithm to solve (1.5). In Section 3, the convergence performance of the studied algorithms is discussed. In Section 4, some numerical examples are provided to show that the studied algorithms are superior to some existing algorithms. In Section 5, the application of the AJGI algorithm in dynamical systems is provided. Finally, a brief conclusion is arranged in Section 6.

2. Finite iterative algorithms

First, let us start with a lemma that we will use in the subsequent derivation.

Lemma 2.1. *([6]) Consider the following matrix equation*

$$AXB = F,$$

where $A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{s\times n}$ and $F \in \mathbb{R}^{m\times n}$ are known matrices, and $X \in \mathbb{R}^{r\times s}$ is the matrix to be determined. For this matrix equation, an iterative algorithm is constructed as

$$X(k+1) = X(k) + \mu A^T(F - AX(k)B)B^T,$$

with

$$0 < \mu < \frac{2}{\|A\|_2^2\|B\|_2^2}.$$

If A is a full column-rank matrix and B is a full row-rank matrix, then the iterative solution $X(k)$ converges to the unique solution X^*, that is $\lim_{k\to\infty} X(k) = X^*$.

In order to derive an iterative algorithm to (1.5), we need to define several intermediate matrices:

$$b_1 := C - XB, \quad b_2 := C - AX, \quad b_3 := F - XE, \quad b_4 := F - DX. \tag{2.1}$$

Now, by using the basic idea of Jacobi iterative method, the coefficient matrices A, B, D and E are decomposed into the following forms:

$$A = D_1 + F_1, \quad B = D_2 + F_2, \quad D = D_3 + F_3, \quad E = D_4 + F_4, \tag{2.2}$$

where D_1, D_2, D_3, D_4 are the diagonal parts of A, B, D, E, respectively, i.e.

$$\begin{aligned} D_1 &= diag[a_{11}, a_{22}, \ldots, a_{mm}] \in \mathbb{R}^{m\times m}, \quad D_2 = diag[b_{11}, b_{22}, \ldots, b_{nn}] \in \mathbb{R}^{n\times n}, \\ D_3 &= diag[d_{11}, d_{22}, \ldots, d_{mm}] \in \mathbb{R}^{m\times m}, \quad D_4 = diag[e_{11}, e_{22}, \ldots, e_{nn}] \in \mathbb{R}^{n\times n}. \end{aligned}$$

With the preceding definitions (2.1) and (2.2), equation (1.5) can be decomposed into four fictitious subsystems

$$(D_1 + F_1)X = b_1, \quad X(D_2 + F_2) = b_2, \quad (D_3 + F_3)X = b_3, \quad X(D_4 + F_4) = b_4. \tag{2.3}$$

Then, we get

$$\begin{aligned} D_1X &= C - XB - F_1X, \quad XD_2 = C - AX - XF_2, \\ D_3X &= F - XE - F_3X, \quad XD_4 = F - DX - XF_4. \end{aligned} \tag{2.4}$$

We define

$$\mathcal{A}_1 = D_1, \quad \mathcal{A}_2 = D_2, \quad \mathcal{A}_3 = D_3, \quad \mathcal{A}_4 = D_4, \tag{2.5}$$

and

$$\begin{aligned} &\widehat{b_1} = C - XB - F_1X, \quad \widehat{b_2} = C - AX - XF_2, \\ &\widehat{b_3} = F - XE - F_3X, \quad \widehat{b_4} = F - DX - XF_4. \end{aligned} \tag{2.6}$$

Thus, (2.4) can be simply written as

$$\mathcal{A}_1X = \widehat{b_1}, \quad X\mathcal{A}_2 = \widehat{b_2}, \quad \mathcal{A}_3X = \widehat{b_3}, \quad X\mathcal{A}_4 = \widehat{b_4}. \tag{2.7}$$

After the above treatment, equation (1.5) is decomposed into several simple matrix equations. Let $X(k)$ be the estimates of X at iteration k, associated with the subsystems in (2.7), respectively. From Lemma 2.1, it leads to the following recursive iterative forms:

$$X_1(k) = X(k-1) + \mu\mathcal{A}_1^T[\widehat{b_1} - \mathcal{A}_1X(k-1)], \tag{2.8}$$

$$X_2(k) = X(k-1) + \mu[\widehat{b_2} - X(k-1)\mathcal{A}_2]\mathcal{A}_2^T, \tag{2.9}$$

$$X_3(k) = X(k-1) + \mu\mathcal{A}_3^T[\widehat{b_3} - \mathcal{A}_3X(k-1)], \tag{2.10}$$

$$X_4(k) = X(k-1) + \mu[\widehat{b_4} - X(k-1)\mathcal{A}_4]\mathcal{A}_4^T. \tag{2.11}$$

Substituting (2.5)-(2.6) into (2.8)-(2.11) gives

$$X_1(k) = X(k-1) + \mu D_1[C - XB - F_1X - D_1X(k-1)], \tag{2.12}$$

$$X_2(k) = X(k-1) + \mu[C - AX - XF_2 - X(k-1)D_2]D_2, \tag{2.13}$$

$$X_3(k) = X(k-1) + \mu D_3[F - XE - F_3X - D_3X(k-1)], \tag{2.14}$$

$$X_4(k) = X(k-1) + \mu[F - DX - XF_4 - X(k-1)D_4]D_4. \tag{2.15}$$

The right hand side of the equations above contains the matrix X to be solved, so the above iterative algorithms are impossible to implement. In this case, we replace X in (2.12)–(2.15) with its estimate at iteration $(k-1)$, thus, we get

$$X_1(k) = X(k-1) + \mu D_1[C - AX(k-1) - X(k-1)B], \tag{2.16}$$

$$X_2(k) = X(k-1) + \mu[C - AX(k-1) - X(k-1)B]D_2, \tag{2.17}$$

$$X_3(k) = X(k-1) + \mu D_3[F - DX(k-1) - X(k-1)E], \tag{2.18}$$

$$X_4(k) = X(k-1) + \mu[F - DX(k-1) - X(k-1)E]D_4. \tag{2.19}$$

Since only one iterative solution $X(k)$ is needed, we take the average of $X_1(k), X_2(k), X_3(k)$ and $X_4(k)$. Then we can obtain

$$\begin{aligned} &X(k) \\ &= \frac{X_1(k) + X_2(k) + X_3(k) + X_4(k)}{4} \\ &= X(k-1) + \frac{\mu}{4}D_1[C - AX(k-1) - X(k-1)B] + \frac{\mu}{4}[C - AX(k-1) - X(k-1)B]D_2 \\ &\quad + \frac{\mu}{4}D_3[F - DX(k-1) - X(k-1)E] + \frac{\mu}{4}[F - DX(k-1) - X(k-1)E]D_4. \end{aligned} \tag{2.20}$$

In summary of the above analysis, we present the JGI algorithm for solving (1.5).

Algorithm 2.1 (The Jacobi-gradient iterative (JGI) algorithm)

Step 1. Input matrices $A, B, C, D, E, F \in \mathbb{R}^{m\times m}$. Given any small positive number ε and an appropriate convergence factor μ. Given any two initial matrices $X_1(0), X_2(0), X_3(0), X_4(0)$, and then $X(0) = [X_1(0) + X_2(0) + X_3(0) + X_4(0)]/4$. Set $k := 1$;

Step 2. If $\delta_k = \sqrt{\frac{\|X(k-1)-X^*\|^2}{\|X^*\|^2}} < \varepsilon$, stop; otherwise, go to Step 3;

Step 3. Update the sequences

$$
\begin{aligned}
X_1(k) &= X(k-1) + \mu D_1[C - AX(k-1) - X(k-1)B],\\
X_2(k) &= X(k-1) + \mu[C - AX(k-1) - X(k-1)B]D_2,\\
X_3(k) &= X(k-1) + \mu D_3[F - DX(k-1) - X(k-1)E],\\
X_4(k) &= X(k-1) + \mu[F - DX(k-1) - X(k-1)E]D_4.
\end{aligned}
$$

Compute

$$X(k) = \frac{X_1(k) + X_2(k) + X_3(k) + X_4(k)}{4};$$

Step 4. Set $k := k + 1$, return to Step 2.

In order to improve the convergence performance of Algorithm 2.1, we introduce a relaxation factor ω, then we get the following algorithm.

Algorithm 2.2 (The accelerated Jacobi-gradient based iterative (AJGI) algorithm)

Step 1. Input matrices $A, B, C, D, E, F \in \mathbb{R}^{m\times m}$. Given any small positive number ε, an appropriate convergence factor μ and an appropriate relaxation factor ω such that $0 < \omega < 1$. Given any two initial matrices $X_1(0), X_2(0), X_3(0), X_4(0)$, and then $X(0) = [X_1(0) + X_2(0) + X_3(0) + X_4(0)]/4$. Set $k := 1$;

Step 2. If $\delta_k = \sqrt{\frac{\|X(k-1)-X^*\|^2}{\|X^*\|^2}} < \varepsilon$, stop; otherwise, go to Step 3;

Step 3. Update the sequences

$$
\begin{aligned}
X_1(k) &= X(k-1) + \frac{1}{2}\omega\mu D_1\Big[C - AX(k-1) - X(k-1)B\Big],\\
X^{(1)}(k-1) &= \frac{1}{2}(1-\omega)X_1(k) + \frac{1}{2}\omega X_2(k-1) + \frac{1}{2}(1-\omega)X_3(k-1) + \frac{1}{2}\omega X_4(k-1),\\
X_2(k) &= X^{(1)}(k-1) + \frac{1}{2}(1-\omega)\mu\Big[C - AX^{(1)}(k-1) - X^{(1)}(k-1)B\Big]D_2,\\
X^{(2)}(k-1) &= \frac{1}{2}(1-\omega)X_1(k) + \frac{1}{2}\omega X_2(k) + \frac{1}{2}(1-\omega)X_3(k-1) + \frac{1}{2}\omega X_4(k-1),\\
X_3(k) &= X^{(2)}(k-1) + \frac{1}{2}\omega\mu D_3\Big[F - DX^{(2)}(k-1) - X^{(2)}(k-1)E\Big],\\
X^{(3)}(k-1) &= \frac{1}{2}(1-\omega)X_1(k) + \frac{1}{2}\omega X_2(k) + \frac{1}{2}(1-\omega)X_3(k) + \frac{1}{2}\omega X_4(k-1)\\
X_4(k) &= X^{(3)}(k-1) + \frac{1}{2}(1-\omega)\mu\Big[F - DX^{(3)}(k-1) - X^{(3)}(k-1)E\Big]D_4,\\
X(k) &= \frac{1}{2}(1-\omega)X_1(k) + \frac{1}{2}\omega X_2(k) + \frac{1}{2}(1-\omega)X_3(k) + \frac{1}{2}\omega X_4(k);
\end{aligned}
$$

Step 4. Set $k := k + 1$, return to Step 2.

Remark 2.2. *Different matrices are employed when updating $X_2(k)$, $X_3(k)$ and $X_4(k)$ at Step 3 of the JGI algorithm and AJGI algorithm, which is their primary distinction. To be specific, the matrix $X(k-1)$ is employed to update $X_2(k)$, $X_3(k)$ and $X_4(k)$ in the JGI algorithm. However, the matrices $X^{(1)}(k-1)$, $X^{(2)}(k-1)$ and $X^{(3)}(k-1)$, which are different from $X(k-1)$, are used to update $X_2(k)$, $X_3(k)$ and $X_4(k)$ in the AJGI algorithm.*

Remark 2.3. *Parallel computing is a subdivision of high performance computing. Its main idea is to decompose a complex problem into several parts and give each part to an independent processor for computation to improve efficiency. Although equations (1.5) were decomposed into four virtual subsystems in the construction of JGI algorithm and AJGI algorithm, these four virtual subsystems were not handed over to separate processors for calculation. Therefore, both the JGI algorithm and the AJGI algorithm are not parallelized. The design of parallel algorithm will be further studied in the future.*

3. Convergence analysis

In this section, we study the convergence of the JGI algorithm and AJGI algorithm, which are listed in the following theorems.

Theorem 3.1. *If the coupled Sylvester matrix equations (1.5) are consistent and have a unique solution X^*, then the iterative solution $X(k)$ generated by Algorithm 2.1 converges to X^*, i.e.,* $\lim_{k\to\infty} X(k) = X^*$*; or the error $X(k) - X^*$ converges to zero for any initial value $X(0)$ if and only if*

$$\rho(\mathcal{H}) < 1, \tag{3.1}$$

where

$$\begin{aligned}\mathcal{H} =& I_{mn} - \frac{1}{4}\mu\Big(I_n \otimes D_1A + B^T \otimes D_1 + D_2 \otimes A + D_2B^T \otimes I_m \\ &+ I_n \otimes D_3D + E^T \otimes D_3 + D_4 \otimes D + D_4E^T \otimes I_m\Big).\end{aligned} \tag{3.2}$$

Proof. Define the error matrix

$$\widetilde{X}(k) = X(k) - X^*. \tag{3.3}$$

According to Algorithm 2.1, it can be obtained

$$\widetilde{X}_1(k) = \widetilde{X}(k-1) - \mu D_1(A\widetilde{X}(k-1) + \widetilde{X}(k-1)B), \tag{3.4}$$
$$\widetilde{X}_2(k) = \widetilde{X}(k-1) - \mu(A\widetilde{X}(k-1) + \widetilde{X}(k-1)B)D_2, \tag{3.5}$$
$$\widetilde{X}_3(k) = \widetilde{X}(k-1) - \mu D_3(D\widetilde{X}(k-1) + \widetilde{X}(k-1)E), \tag{3.6}$$
$$\widetilde{X}_4(k) = \widetilde{X}(k-1) - \mu(D\widetilde{X}(k-1) + \widetilde{X}(k-1)E)D_4, \tag{3.7}$$

and

$$\begin{aligned}\widetilde{X}(k) =& \widetilde{X}(k-1) - \frac{1}{4}\Big(\mu D_1A\widetilde{X}(k-1) + \mu D_1\widetilde{X}(k-1)B + \mu A\widetilde{X}(k-1)D_2 + \mu\widetilde{X}(k-1)BD_2 \\ &+ \mu D_3D\widetilde{X}(k-1) + \mu D_3\widetilde{X}(k-1)E + \mu D\widetilde{X}(k-1)D_4 + \mu\widetilde{X}(k-1)ED_4\Big).\end{aligned} \tag{3.8}$$

Taking the vec-operator of the both sides of (3.8) gives

$$\begin{aligned}vec\Big(\widetilde{X}(k)\Big) &= vec\Big(\widetilde{X}(k-1)\Big) - \frac{1}{4}\mu\Big(I_n \otimes D_1A + B^T \otimes D_1 + D_2 \otimes A + D_2B^T \otimes I_m \\ &\quad + I_n \otimes D_3D + E^T \otimes D_3 + D_4 \otimes D + D_4E^T \otimes I_m\Big)vec\Big(\widetilde{X}(k-1)\Big) \\ &= \mathcal{H}vec\Big(\widetilde{X}(k-1)\Big).\end{aligned} \tag{3.9}$$

Therefore, Algorithm 2.1 is convergent if and only if inequality (3.1) holds. □

To facilitate the description of the convergence of the AJGI algorithm, define the following symbols,

$$\mathcal{M} = I - \frac{1}{2}\omega\mu(I \otimes D_1A) - \frac{1}{2}\omega\mu(B^T \otimes D_1), \tag{3.10}$$

$$\mathcal{N} = I - \frac{1}{2}(1-\omega)\mu(D_2 \otimes A) - \frac{1}{2}(1-\omega)\mu(D_2B^T \otimes I), \tag{3.11}$$

$$\mathcal{P} = I - \frac{1}{2}\omega\mu(I \otimes D_3D) - \frac{1}{2}\omega\mu(E^T \otimes D_3), \tag{3.12}$$

$$\mathcal{Q} = I - \frac{1}{2}(1-\omega)\mu(D_4 \otimes D) - \frac{1}{2}(1-\omega)\mu(D_4E^T \otimes I), \tag{3.13}$$

$$\mathcal{M}_1 = \frac{1}{2}(1-\omega)\mathcal{M}, \quad \mathcal{M}_2 = \frac{1}{2}\omega\mathcal{M}, \tag{3.14}$$

$$\mathcal{N}_1 = \frac{1}{2}(1-\omega)\mathcal{N}\mathcal{M}_1, \quad \mathcal{N}_2 = \frac{1}{2}(1-\omega)\mathcal{N}\mathcal{M}_2 + \frac{1}{2}\omega\mathcal{N}, \tag{3.15}$$

$$\mathcal{N}_3 = \frac{1}{2}(1-\omega)\mathcal{N}\mathcal{M}_1 + \frac{1}{2}(1-\omega)\mathcal{N}, \tag{3.16}$$

$$\mathcal{P}_1 = \frac{1}{2}(1-\omega)\mathcal{P}\mathcal{M}_1 + \frac{1}{2}\omega\mathcal{P}\mathcal{N}_1, \quad \mathcal{P}_2 = \frac{1}{2}(1-\omega)\mathcal{P}\mathcal{M}_2 + \frac{1}{2}\omega\mathcal{P}\mathcal{N}_2, \tag{3.17}$$

$$\mathcal{P}_3 = \frac{1}{2}(1-\omega)\mathcal{P}\mathcal{M}_1 + \frac{1}{2}\omega\mathcal{P}\mathcal{N}_3 + \frac{1}{2}(1-\omega)\mathcal{P}, \ \mathcal{P}_4 = \frac{1}{2}(1-\omega)\mathcal{P}\mathcal{M}_2 + \frac{1}{2}\omega\mathcal{P}\mathcal{N}_2 + \frac{1}{2}\omega\mathcal{P}, \tag{3.18}$$

$$\mathcal{Q}_1 = \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{M}_1 + \frac{1}{2}\omega\mathcal{Q}\mathcal{N}_1 + \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{P}_1, \tag{3.19}$$

$$\mathcal{Q}_2 = \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{M}_2 + \frac{1}{2}\omega\mathcal{Q}\mathcal{N}_2 + \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{P}_2, \tag{3.20}$$

$$\mathcal{Q}_3 = \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{M}_1 + \frac{1}{2}\omega\mathcal{Q}\mathcal{N}_3 + \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{P}_3, \tag{3.21}$$

$$\mathcal{Q}_4 = \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{M}_2 + \frac{1}{2}\omega\mathcal{Q}\mathcal{N}_2 + \frac{1}{2}(1-\omega)\mathcal{Q}\mathcal{P}_4 + \frac{1}{2}\omega\mathcal{Q}. \tag{3.22}$$

Now we are in a position to analyze the convergence of Algorithm 2.2.

Theorem 3.2. *If the coupled Sylvester matrix equations (1.5) are consistent and have a unique solution X^*, then the iterative solution $X(k)$ generated by Algorithm 2.2 converges to X^*, i.e., $\lim_{k\to\infty} X(k) = X^*$; or the error $X(k) - X^*$ converges to zero for any initial value $X(0)$ if and only if μ satisfies*

$$\rho(\mathcal{A}) < 1, \tag{3.23}$$

where

$$\mathcal{A} = \begin{bmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \mathcal{M}_1 & \mathcal{M}_2 \\ \mathcal{N}_1 & \mathcal{N}_2 & \mathcal{N}_3 & \mathcal{N}_2 \\ \mathcal{P}_1 & \mathcal{P}_2 & \mathcal{P}_3 & \mathcal{P}_4 \\ \mathcal{Q}_1 & \mathcal{Q}_2 & \mathcal{Q}_3 & \mathcal{Q}_4 \end{bmatrix}. \tag{3.24}$$

Proof. Define the error matrices

$$\widetilde{X}(k) = X(k) - X^*, \quad \widetilde{X}_i(k) = X_i(k) - X^*, \quad i \in I[1,4], \tag{3.25}$$

$$\widetilde{X}^{(j)}(k) = X^{(j)}(k) - X^*, \quad j \in I[1,3]. \tag{3.26}$$

From Algorithm 2.2, it gets

$$\widetilde{X}_1(k) = \widetilde{X}(k-1) - \frac{1}{2}\omega\mu D_1\Big(A\widetilde{X}(k-1) + \widetilde{X}(k-1)B\Big), \tag{3.27}$$

$$\widetilde{X}_2(k) = \widetilde{X}^{(1)}(k-1) - \frac{1}{2}(1-\omega)\mu\Big(A\widetilde{X}^{(1)}(k-1) + \widetilde{X}^{(1)}(k-1)B\Big)D_2, \tag{3.28}$$

$$\widetilde{X}_3(k) = \widetilde{X}^{(2)}(k-1) - \frac{1}{2}\omega\mu D_3\Big(D\widetilde{X}^{(2)}(k-1) + \widetilde{X}^{(2)}(k-1)E\Big), \tag{3.29}$$

$$\widetilde{X}_4(k) = \widetilde{X}^{(3)}(k-1) - \frac{1}{2}(1-\omega)\mu\Big(D\widetilde{X}^{(3)}(k-1) + \widetilde{X}^{(3)}(k-1)E\Big)D_4. \tag{3.30}$$

By combining vector operator with Kronecker product, it has $vec(AXB) = (B^T \otimes A)vec(X)$ and

$$\begin{aligned}
&vec(\widetilde{X}_1(k)) \\
&= vec(\widetilde{X}(k-1)) - \frac{1}{2}\omega\mu\Big[(I \otimes D_1A) + (B^T \otimes D_1)\Big]vec(\widetilde{X}(k-1)), \\
&= \Big[I - \frac{1}{2}\omega\mu(I \otimes D_1A) - \frac{1}{2}\omega\mu(B^T \otimes D_1)\Big]vec(\widetilde{X}(k-1)) \\
&= \mathcal{M}vec(\widetilde{X}(k-1)) \\
&= \mathcal{M}\Big[\frac{1}{2}(1-\omega)vec(\widetilde{X}_1(k-1)) + \frac{1}{2}\omega vec(\widetilde{X}_2(k-1)) \\
&\quad + \frac{1}{2}(1-\omega)vec(\widetilde{X}_3(k-1)) + \frac{1}{2}\omega vec(\widetilde{X}_4(k-1))\Big] \\
&= \mathcal{M}_1vec(\widetilde{X}_1(k-1)) + \mathcal{M}_2vec(\widetilde{X}_2(k-1)) \\
&\quad + \mathcal{M}_1vec(\widetilde{X}_3(k-1)) + \mathcal{M}_2vec(\widetilde{X}_4(k-1)),
\end{aligned} \tag{3.31}$$

$$\begin{aligned}
&vec(\widetilde{X}_2(k)) \\
&= vec(\widetilde{X}^{(1)}(k-1)) - \frac{1}{2}(1-\omega)\mu\Big[(D_2 \otimes A) + (D_2B^T \otimes I)\Big]vec(\widetilde{X}^{(1)}(k-1)), \\
&= \Big[I - \frac{1}{2}(1-\omega)\mu(D_2 \otimes A) - \frac{1}{2}(1-\omega)\mu(D_2B^T \otimes I)\Big]vec(\widetilde{X}^{(1)}(k-1)) \\
&= \mathcal{N}vec(\widetilde{X}^{(1)}(k-1)) \\
&= \mathcal{N}\Big[\frac{1}{2}(1-\omega)vec(\widetilde{X}_1(k)) + \frac{1}{2}\omega vec(\widetilde{X}_2(k-1)) \\
&\quad + \frac{1}{2}(1-\omega)vec(\widetilde{X}_3(k-1)) + \frac{1}{2}\omega vec(\widetilde{X}_4(k-1))\Big] \\
&= \mathcal{N}_1vec(\widetilde{X}_1(k-1)) + \mathcal{N}_2vec(\widetilde{X}_2(k-1)) \\
&\quad + \mathcal{N}_3vec(\widetilde{X}_3(k-1)) + \mathcal{N}_2vec(\widetilde{X}_4(k-1)),
\end{aligned} \tag{3.32}$$

$$\begin{aligned}
&vec(\widetilde{X}_3(k)) \\
&= vec(\widetilde{X}^{(2)}(k-1)) - \frac{1}{2}\omega\mu\Big[(I \otimes D_3D) + (E^T \otimes D_3)\Big]vec(\widetilde{X}^{(2)}(k-1)), \\
&= \Big[I - \frac{1}{2}\omega\mu(I \otimes D_3D) - \frac{1}{2}\omega\mu(E^T \otimes D_3)\Big]vec(\widetilde{X}^{(2)}(k-1)) \\
&= \mathcal{P}vec(\widetilde{X}^{(2)}(k-1)) \\
&= \mathcal{P}\Big[\frac{1}{2}(1-\omega)vec(\widetilde{X}_1(k)) + \frac{1}{2}\omega vec(\widetilde{X}_2(k)) \\
&\quad + \frac{1}{2}(1-\omega)vec(\widetilde{X}_3(k-1)) + \frac{1}{2}\omega vec(\widetilde{X}_4(k-1))\Big] \\
&= \mathcal{P}_1vec(\widetilde{X}_1(k-1)) + \mathcal{P}_2vec(\widetilde{X}_2(k-1))
\end{aligned}$$

$$+ \mathcal{P}_3 vec(\widetilde{X}_3(k-1)) + \mathcal{P}_4 vec(\widetilde{X}_4(k-1)), \tag{3.33}$$

$$\begin{aligned}
&vec(\widetilde{X}_4(k)) \\
&= vec(\widetilde{X}^{(3)}(k-1)) - \frac{1}{2}(1-\omega)\mu\Big[(D_4 \otimes D) + (D_4E^T \otimes I)\Big]vec(\widetilde{X}^{(3)}(k-1)), \\
&= \Big[I - \frac{1}{2}(1-\omega)\mu(D_4 \otimes D) - \frac{1}{2}(1-\omega)\mu(D_4E^T \otimes I)\Big]vec(\widetilde{X}^{(3)}(k-1)) \\
&= \mathcal{Q}vec(\widetilde{X}^{(3)}(k-1)) \\
&= \mathcal{Q}\Big[\frac{1}{2}(1-\omega)vec(\widetilde{X}_1(k)) + \frac{1}{2}\omega vec(\widetilde{X}_2(k)) + \frac{1}{2}(1-\omega)vec(\widetilde{X}_3(k)) + \frac{1}{2}\omega vec(\widetilde{X}_4(k-1))\Big] \\
&= \mathcal{Q}_1 vec(\widetilde{X}_1(k-1)) + \mathcal{Q}_2 vec(\widetilde{X}_2(k-1)) \\
&\quad + \mathcal{Q}_3 vec(\widetilde{X}_3(k-1)) + \mathcal{Q}_4 vec(\widetilde{X}_4(k-1)).
\end{aligned} \tag{3.34}$$

From Eqs(3.31)-(3.34), it can be obtain

$$\begin{bmatrix} vec(\widetilde{X}_1(k)) \\ vec(\widetilde{X}_2(k)) \\ vec(\widetilde{X}_3(k)) \\ vec(\widetilde{X}_4(k)) \end{bmatrix} = \mathcal{A} \begin{bmatrix} vec(\widetilde{X}_1(k-1)) \\ vec(\widetilde{X}_2(k-1)) \\ vec(\widetilde{X}_3(k-1)) \\ vec(\widetilde{X}_4(k-1)) \end{bmatrix}. \tag{3.35}$$

Thus, Algorithm 2.2 converges if and only if inequality (3.23) holds. This completes the proof. □

Apparently, calculating the appropriate parameter μ to satisfy relations (3.1) and (3.23) is a hard task. In addition, it is also difficult to obtain the optimal convergence parameter directly from Theorems 3.1 and 3.2. However, these problems can be addressed by programming with MATLAB software. For more details, please refer to Section 4, where we further explain these issues. Meanwhile, we will continue to study how to directly obtain appropriate parameter range and optimal convergence parameter through theoretical derivation in the future.

4. Numerical examples

In this section, we give an example to illustrate the performance of the proposed algorithms. All the computations are performed on Intel(R) Core(TM) i5-12500H CPU @ 2.50GHz, 16GB RAM by using MATLAB R2021b.

In the following examples, the initial iteration value is taken as $X(0) = 10^{-6}1_{m\times m}$, where $1_{m\times m}$ is an m-order square matrix with all elements of 1, and the relative error is defined as:

$$\delta_k = \sqrt{\frac{||X(k) - X^*||^2}{||X^*||^2}},$$

where $X(k)$ is the *kth* iterative solution.

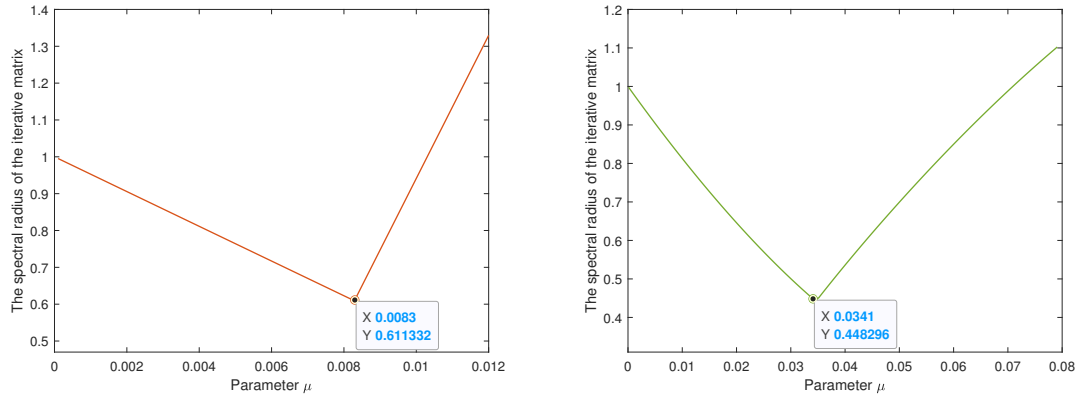
Example 4.1 Consider the coupled Sylvester matrix equation (1.5), where A, B, D, E, X are generated in MATLAB as follows:

$A = triu(rand(m), 1) + diag(\alpha + diag(rand(m)))$;
$B = tril(rand(m), 1) + eye(m)$;
$D = rand(m) + diag(\alpha + diag(rand(m)))$;
$E = rand(m) + diag(\alpha + diag(rand(m)))$;

$$X = rand(m) + eye(m) \times \beta; \quad X = X + X^T.$$

Let $C = AX + XB$, $F = DX + XE$, then equations (1.5) have a unique solution X^*. In this example, we choose $m = 30$, $\alpha = 6$ and $\beta = 2$.

In Fig. 1, we calculate the spectral radius of the iterative matrices of the JGI algorithm and the AJGI algorithm respectively according to Theorems 3.1 and 3.2. When $0 < \mu < 0.0104$ is met, then $\rho(\mathcal{H}) < 1$ holds and the JGI algorithm is convergent. Analogously, when $0 < \mu < 0.072$ is satisfied, then $\rho(\mathcal{A}) < 1$ holds and the AJGI algorithm is convergent. It can also be obtained that the optimal convergence parameters of JGI algorithm and AJGI algorithm are 0.0083 and 0.0341 respectively, and the spectral radius of their iterative matrices reaches the minimum value.



(a) $\rho(\mathcal{H})$ (b) $\rho(\mathcal{A})$

Fig. 1: The spectral radius of the iteration matrices of the JGI and AJGI algorithms.

The relative error δ_k of the algorithms is displayed in Fig. 2, and the values of convergence parameter μ and relaxation parameter ω for each algorithm in Fig. 2 are shown in Table 1. These algorithms are effective because their relative error δ_k decreases and tends to zero with the increase of iteration steps k. From Fig. 2, we can find the convergence performance of the JGI algorithm and the AJGI algorithm are better than that of the GI algorithm [5], RGI algorithm [22] and AGI algorithm [35].

In addition, we compare the iteration step (denoted as IT), the computational time in seconds (denoted as CPU) and the relative error (denoted as δ_k) of the GI algorithm [5], RGI algorithm [22] and AGI algorithm [35], the JGI algorithm and the AJGI algorithm, and the results are listed in Table 1. As it can be seen from Table 1, the JGI algorithm and the AJGI algorithm perform better in terms of efficiency and accuracy.

Table 1: The parameter values (μ, ω), the iterative steps (IT), the relative error (δ_k) and the computational time (CPU) of the algorithms

Method	μ	ω	IT	CPU	δ_k
GI algorithm [5]	0.0017	\	241	0.0356	9.9346e-05
RGI algorithm [22]	0.0136	$\frac{1}{3}$	135	0.0299	9.7989e-05
AGI algorithm [35]	0.0130	$\frac{1}{2}$	78	0.0284	9.1441e-05
JGI algorithm in this paper	0.0083	\	44	0.0154	8.9304e-05
AJGI algorithm in this paper	0.0341	$\frac{1}{2}$	22	0.0234	9.2566e-05

Example 4.2 Consider the coupled Sylvester matrix equation (1.5), where A, B, D, E, X are taken from Example 4.1. In this example, we take $m = 100$, $\alpha = 8$ and $\beta = 1$.

Fig. 3 shows the spectral radius of the iterative matrices of the JGI algorithm and the AJGI algorithm respectively. From Fig. 3, it can be concluded that the JGI algorithm is convergent when $0 < \mu < 0.0030$ and

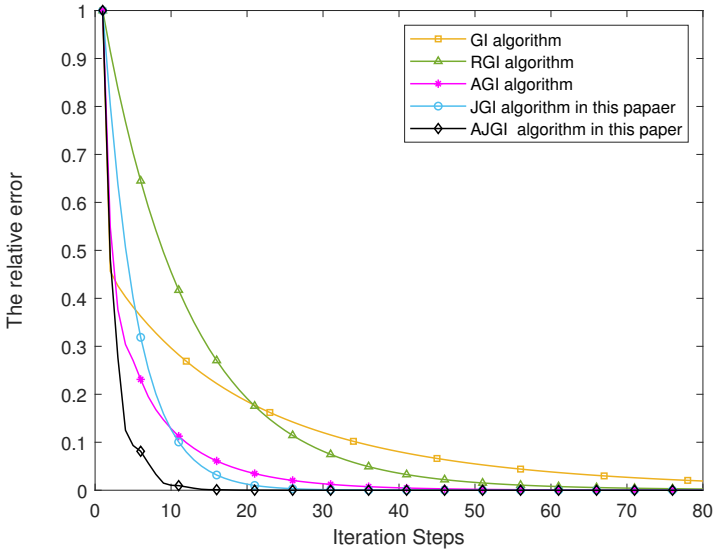
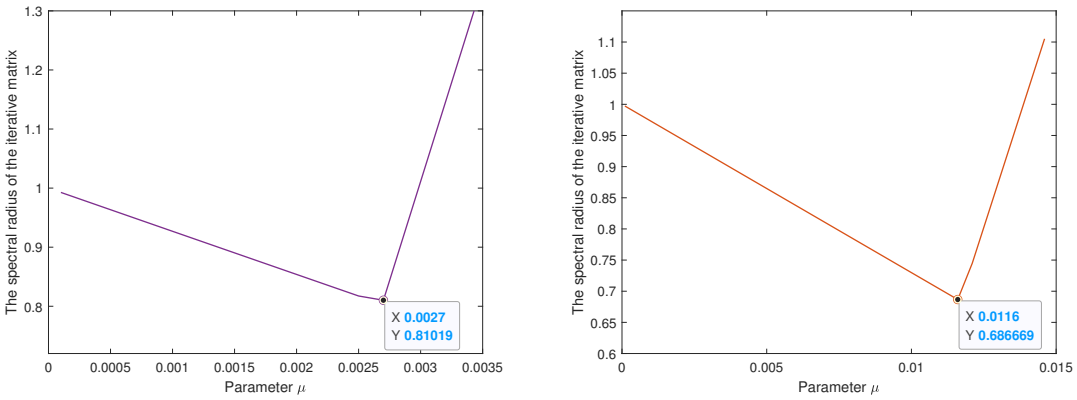


Fig. 2: Comparison of convergence curves

the AJGI algorithm is convergent when $0 < \mu < 0.0139$. Moreover, the optimal convergence parameters of JGI algorithm and AJGI algorithm are 0.0027 and 0.0116 respectively.

We compare the relative errors of the GI [5], RGI [22], AGI [35], JGI and AJGI algorithms, and the convergence curve is shown in Fig. 4. The values of convergence parameters (μ, ω) of these algorithms are described in Table 2. In addition, we compare the iteration step (denoted as IT), the computational time in seconds (denoted as CPU) and the relative error (denoted as δ_k) of these algorithms in Table 2. It can be concluded that the JGI algorithm and the AJGI algorithm have better convergence performance.



(a) $\rho(\mathcal{H})$ (b) $\rho(\mathcal{A})$

Fig. 3: The spectral radius of the iteration matrices of the JGI and AJGI algorithms.

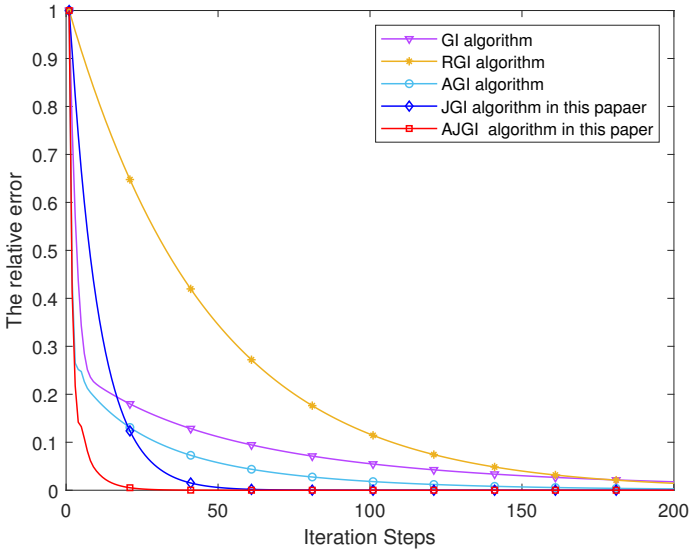


Fig. 4: Comparison of convergence curves

Table 2: The parameter values (μ, ω), the iterative steps (IT), the relative error (δ_k) and the computational time (CPU) of the algorithms.

Method	μ	ω	IT	CPU	δ_k
GI algorithm [5]	3.4947×10^{-4}	\	838	1.6783	6.3579e-05
RGI algorithm [22]	2.2300×10^{-3}	$\frac{1}{4}$	667	1.3165	6.3338e-05
AGI algorithm [35]	2.0000×10^{-3}	$\frac{1}{3}$	424	0.9833	6.3279e-05
JGI algorithm in this paper	2.7000×10^{-3}	\	92	0.1737	6.3315e-05
AJGI algorithm in this paper	1.1600×10^{-2}	$\frac{1}{3}$	46	0.1475	6.3707e-05

5. Application in dynamical systems

Example 5.1. Consider the following dynamical systems

$$\begin{cases} \dot{x} = Ax + Bu \\ \dot{w} = Fw + Pu \\ e = Cx + Qw, \end{cases} \tag{5.1}$$

where $A \in \mathbb{R}^{n\times n}$,$B \in \mathbb{R}^{n\times r}$, $F \in \mathbb{R}^{p\times p}$, $P \in \mathbb{R}^{p\times r}$, $C \in \mathbb{R}^{m\times n}$ and $Q \in \mathbb{R}^{m\times p}$ are constant matrices, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^r$ and $e \in \mathbb{R}^m$ are the state, the control input and the measurable error output, respectively. The symbol $w \in \mathbb{R}^p$ is the exogenous input which includes "reference signals to be tracked" and/or "disturbances to be rejected". If we assume that (A, B) is controllable, then F is critical stable. If the full information feedback $u = -Kx + Lw$ is applied on the system, then the closed-loop system can result in:

$$\begin{cases} \dot{x} = (A - BK)x + BLw \\ \dot{w} = (F + PL)w - PKx \\ e = Cx + Qw, \end{cases} \tag{5.2}$$

The aim of the output regulation problem is to find two matrices K and L such that the matrix $A - BK$ is stable and

$$\lim_{t\to\infty} e(t) = \lim_{t\to\infty}(Cx(t) + Qw(t)) = 0, \tag{5.3}$$

which $(x(0), w(0))$ are arbitrary and $(x(0), w(0)) \in \mathbb{R}^n \times \mathbb{R}^p$. It has been shown that such a problem is solvable if and only if there exist a matrix X such that

$$\begin{cases} A_1X + XB_1 = C_1 \\ A_2X + XB_2 = C_2. \end{cases} \tag{5.4}$$

If the parametric matrices are chosen as

$$\begin{aligned} A_1 &= triu(rand(m), 1) + diag(\alpha + diag(rand(m))); \\ B_1 &= tril(rand(m), 1) + eye(m); \\ A_2 &= rand(m) + diag(\alpha + diag(rand(m))); \\ B_2 &= rand(m) + diag(\alpha + diag(rand(m))); \end{aligned}$$

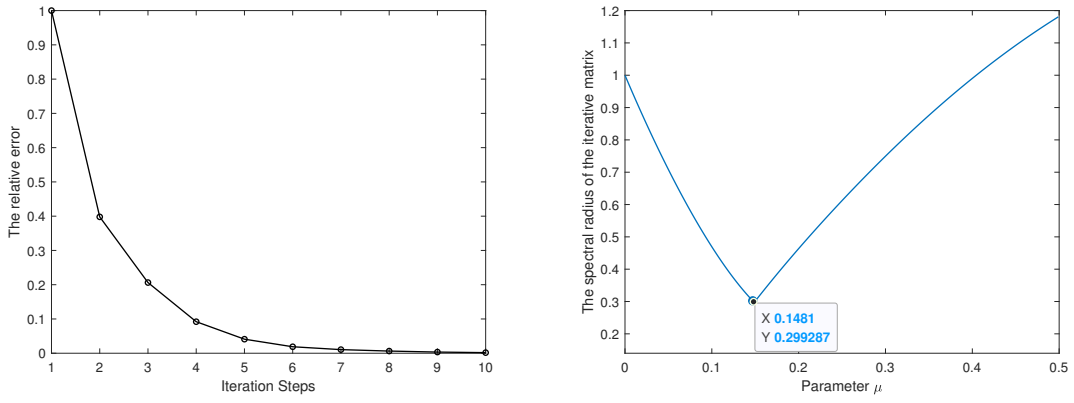
$$C_1 = \begin{pmatrix} 24.1669 & 7.4974 & 6.1789 & 6.7332 & 8.7541 & 8.3442 \\ 3.2026 & 23.0693 & 8.0316 & 8.6003 & 5.2917 & 4.3749 \\ 4.5302 & 6.1486 & 17.4244 & 7.3854 & 6.6593 & 6.8793 \\ 4.1416 & 9.2624 & 4.9138 & 18.7287 & 3.8105 & 3.5077 \\ 4.6083 & 7.6737 & 4.6326 & 7.5453 & 20.1149 & 5.9955 \\ 3.4422 & 7.7432 & 6.5496 & 9.1927 & 4.1675 & 22.4214 \end{pmatrix}.$$

and

$$C_2 = \begin{pmatrix} 37.8630 & 9.3800 & 7.4641 & 7.6002 & 13.4315 & 8.9297 \\ 5.4919 & 35.7256 & 6.1952 & 10.6085 & 10.2580 & 8.8828 \\ 11.4565 & 6.9082 & 28.2476 & 9.9142 & 10.9923 & 10.9328 \\ 9.5131 & 13.5593 & 11.4434 & 36.3935 & 8.7026 & 11.1344 \\ 10.1583 & 14.8293 & 6.4088 & 14.5341 & 32.7100 & 10.8894 \\ 7.5627 & 14.6808 & 11.7805 & 13.3619 & 14.8203 & 39.8488 \end{pmatrix}.$$

in which $m = 6, \alpha = 3$. In this case, we apply the AJGI algorithm to solve the coupled Sylvester matrix equations (5.4) with one side and the simulation results are provided in Fig. 5.

In Fig. 5(a), the parameter values (μ, ω) are set to $(0.1481, \frac{1}{2})$. From Fig. 5(b), the AJGI is convergent when $0 < \mu < 0.4058$ and the spectral radius of the iterative matrix of the AJGI algorithm reaches the minimum value when $\mu = 0.1481$.



(a) (b)

Fig. 5: (a) Convergence curves of the AJGI algorithm; (b) The spectral radius of the iteration matrices of the AJGI algorithm.

When the AJGI algorithm is iterated 10 times, the relative error is 7.9325×10^{-4} and the numerical solution is

$$X = \begin{pmatrix} 4.3266 & 0.3674 & 0.0987 & 0.1068 & 0.8909 & 0.5000 \\ 0.2316 & 4.1559 & 0.2619 & 0.6538 & 0.3342 & 0.4799 \\ 0.4889 & 0.0377 & 3.3778 & 0.4942 & 0.6987 & 0.9047 \\ 0.6241 & 0.8852 & 0.6797 & 3.9289 & 0.1978 & 0.6099 \\ 0.6791 & 0.9133 & 0.1366 & 0.7150 & 3.4629 & 0.6177 \\ 0.3955 & 0.7962 & 0.7212 & 0.9037 & 0.7441 & 4.6628 \end{pmatrix}.$$

6. Conclusion remarks and future works

In this paper, the Jacobi-gradient iterative algorithm and the accelerated Jacobi-gradient based iterative algorithm are established for solving coupled Sylvester matrix equations with one side. The convergence of the proposed algorithms is analysed by using the spectral norm and the Kronecker product. Some numerical examples are given to show that the iterative solution generated by the studied iterative algorithms can converge to the exact solution under certain conditions. Moreover, we give the application in dynamical systems. As an outlook for the future, we believe the algorithms presented in this paper may be applied to study the other vital matrix equations, such as the generalized nonhomogeneous Yakubovich-transpose matrix equation [15], the generalized Lyapunov matrix equation [4, 23] and so on.

References

[1] A. Bayoumi, M. Ramadan, An accelerated gradient-based iterative algorithm for solving extended Sylvester-conjugate matrix equations, *Trans. Inst. Meas. Control* 40(1)(2018) 341-347.

[2] M. Dehghan, M. Hajarian, Convergence of the descent Dai-Yuan conjugate gradient method for unconstrained optimization, *J. Vib. Control* 18 (9) (2012) 1249-1253.

[3] M. Dehghan, A. Shirilord, Matrix multisplitting Picard-iterative method for solving generalized absolute value matrix equation, *Appl. Numer. Math.* 158 (2020) 425-438.

[4] M. Dehghan, A. Shirilord, A new approximation algorithm for solving generalized Lyapunov matrix equations, *J. Comput. Appl. Math.* 404 (2022) 113898.

[5] F. Ding, T. W. Chen, On iterative solutions of general coupled matrix equations, *SIAM J. Control Optim.* 44 (6) (2006) 2269-2284.

[6] F. Ding, P. X. Liu, J. Ding, Iterative solutions of the generalized Sylvester matrix equation by using the hierarchical identification principle, *Appl. Math. Comput.* 197 (2008) 41-50.

[7] G. R. Duan, G. P. Liu, Complete parametric approach for eigenstrutture assignment in a class of secondorder linear systems, *Automatica* 38 (4) (2002) 725-729.

[8] H. L. Fan, Estimates for the lower bounds on the inverse elements of strictly diagonally dominant tridiagonal period matrices in signal processing, *Adv. Mat. Res.* 159 (2011) 459-463.

[9] W. W. Fan, C. Q. Gu, L. Z. Tian, Jacobi-gradient iterative algorithms for Sylvester matrix equations, in: *Proceedings of 14th Conference of the International Linear Algebra Society*, Shanghai University, Shanghai, China, (2007) 16-20.

[10] M. Hajarian, Matrix iterative methods for solving the Sylvester-transpose and periodic Sylvester matrix equations, *J. Franklin Inst.* 350 (2013) 3328-3341.

[11] M. Hajarian, Extending the CGLS algorithm for least squares solutions of the generalized Sylvester-transpose matrix equations, *J. Franklin Inst.* 353 (2016) 1168-1185.

[12] M. Hajarian, Generalized conjugate direction algorithm for solving the general coupled matrix equations over symmetric matrices, *Numerical Algorithms* 73 (2016) 591-609.

[13] M. Hajarian, Least squares solution of the linear operator equation, *J. Optim. Theory Appl.* 170 (2016) 205-219.

[14] M. Hajarian, Solving the general Sylvester discrete-time periodic matrix equations via the gradient based iterative method, *Appl. Math. Lett.* 52 (2016) 87-95.

[15] M. Hajarian, New finite algorithm for solving the generalized nonhomogeneous Yakubovich-transpose matrix equation, *Asian J. Control* 19 (2017) 164-172.

[16] Z. H. He, Pure PSVD approach to Sylvester-type quaternion matrix equations, *Electron. J. Linear Algebra*, 35 (2019) 266-284.

[17] Z. H. He, The general solution to a system of coupled Sylvester-type quaternion tensor equations involving η-Hermicity, *Bull. Iran. Math. Soc.*, 45 (2019) 1407–1430.

[18] B. H. Huang, C. F. Ma, The relaxed gradient-based iterative algorithms for a class of generalized coupled Sylvester-conjugate matrix equations, *J. Franklin Inst.* 355 (2018) 3168-3195.

[19] C. F. Ma, T. X. Yan, A finite iterative algorithm for the general discrete-time periodic Sylvester matrix equations, *J. Franklin Inst.* 359 (9) (2021) 4410-4432.

[20] M. Ramadan, T. El-Danaf, A. Bayoumi, A relaxed gradient based algorithm for solving extended Sylvester-conjugate matrix equations, *Asian J. Control* 16(5)(2014) 1-8.
[21] X. P. Sheng, A relaxed gradient based algorithm for solving generalized coupled Sylvester matrix equations, *J. Franklin Inst.* 355 (2018) 4282-4297.
[22] X. P. Sheng, W. W. Sun, The relaxed gradient based iterative algorithm for solving matrix equations $A_iXB_i = F_i$, *Comput. Math. with Appl.* 74 (2017) 597-604.
[23] A. Shirilord, M. Dehghan, Combined real and imaginary parts method for solving generalized Lyapunov matrix equation, *Appl. Numer. Math.* 181 (2022) 94-109.
[24] A. Shirilord, M. Dehghan, Double parameter splitting (DPS) iteration for solving complex symmetric linear systems, *Appl. Numer. Math.* 171 (2022) 176-192.
[25] C. Q. Song, G. L. Chen, Q. B. Liu. Explicit solutions to the quaternion matrix equations $X - AXF = C$ and $X - A\widetilde{X}F = C$, *Int. J. Comput. Math.* 89 (2012) 890-900.
[26] C. Q. Song, H. X. Rui, X. D. Wang, J. L. Zhao, Closed-form solutions to the nonhomogeneous Yakubovich-transpose matrix equation, *J. Comput. Appl. Math.* 267 (2014) 72-81.
[27] 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 (8) (2017) 2381-2390.
[28] W. L. Wang, G. R. Qu, C. Q. Song, Cyclic gradient based iterative algorithm for a class of generalized coupled Sylvester-conjugate matrix equations, *J. Franklin Inst.* 360 (11) (2023) 7206-7229.
[29] W. L. Wang, G. R. Qu, C. Q. Song, D. Liu. Block-row and block-column iterative algorithms for solving linear matrix equation, *Comput. Appl. Math.* (2023) 42: 174.
[30] W. L. Wang, C. Q. Song, Iterative algorithms for discrete-time periodic Sylvester matrix equations and its application in antilinear periodic system, *Appl. Numer. Math.* 168 (2021) 251-273.
[31] W. L. Wang, C. Q. Song, S. P. Ji, Iterative solution to a class of complex matrix equations and its application in time-varying linear system, *J. Appl. Math. Comput.*, 67 (2021) 317-341.
[32] A. G. Wu, Y. M. Fu, G. R. Duan, On solutions of matrix equations $V - AVF = BW$ and $V - A\overline{V}F = BW$, *Math. Comput. Modell.* 47 (2008) 1181-1197.
[33] A. G. Wu, L. L. Lv, G. R. Duan, Iterative algorithms for solving a class of complex conjugate and transpose matrix equations, *Appl. Math. Comput.* 217 (2011) 8343-8353.
[34] L. Xiao, R. B. Lu, A fully complex-valued gradient neural network for rapidly computing complex-valued linear matrix equations, *Chin. J. Electron.* 26 (6) (2017) 1194-1197.
[35] 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.
[36] T. X. Yan, C. F. Ma, The BCR algorithms for solving the reflexive or anti-reflexive solutions of generalized coupled Sylvester matrix equations, *J. Franklin Inst.* 357 (17) (2020) 12787-12807.
[37] T. X. Yan, C. F. Ma, An iterative algorithm for generalized Hamiltonian solution of a class of generalized coupled Sylvester-conjugate matrix equations, *Appl. Math. Comput.* 411 (2021) 126491.
[38] H. Q. Yang, B. Zhou, A multilevel iteration method for solving a coupled integral equation model in image restoration, *Mathematics* 8 (2020) 346.
[39] H. M. Zhang, H. C. Yin, New proof of the gradient-based iterative algorithm for a complex conjugate and transpose matrix equation, *J. Franklin Inst.* 354 (2017) 7585-7603.