AIMS Mathematics, 6(8): 8477–8496.
DOI:10.3934/math.2021492
Received: 01 April 2021
Accepted: 27 May 2021
Published: 03 June 2021

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

Research article

Convergence analysis of a gradient iterative algorithm with optimal convergence factor for a generalized Sylvester-transpose matrix equation

Nunthakarn Boonruangkan and Pattrawut Chansangiam*

Department of Mathematics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

* **Correspondence:** Email: pattrawut.ch@kmitl.ac.th; Tel: +66935266600; Fax: +6602329840011 ext. 284.

Abstract: Consider a generalized Sylvester-transpose matrix equation with rectangular coefficient matrices. Based on gradients and hierarchical identification principle, we derive an iterative algorithm to produce a sequence of approximated solutions with a reasonable stopping rule concerning a relative norm-error. A convergence analysis via Banach fixed-point theorem reveals the sequence converges to a unique solution of the matrix equation for any given initial matrix if and only if the convergence factor is chosen appropriately in a certain range. The performance of algorithm is theoretically analysed through the convergence rate and error estimations. The optimal convergence factor is chosen to attain the fastest asymptotic behaviour. Finally, numerical experiments are provided to illustrate the capability and efficiency of the proposed algorithm, compared to recent gradient-based iterative algorithms.

Keywords: generalized Sylvester-transpose matrix equation; gradient; Kronecker product; matrix norm; Banach fixed-point theorem
Mathematics Subject Classification: 15A12, 15A60, 46A22, 65F45

1. Introduction

It is well known that the fundamental of differential equations deals with the algebraic linear system

$$x'(t) = Ax(t), \tag{1.1}$$

where $x(t)$ is an unknown vector-valued function and A is a given square matrix. To analyse the stability of an equilibrium point of the system (1.1), it suffices to find a positive definite matrix L such that $A^TL + LA$ is negative definite; see e.g., [1]. So, we need to solve the so-called Lyapunov equation

$$AX + XA^T = R \tag{1.2}$$

for some negative definite matrix R. To discuss the Eq (1.2), we investigate a more general form, namely, the generalized Sylvester-transpose matrix equation

$$\sum_{i=1}^{p} A_i X B_i + \sum_{j=1}^{q} C_j X^T D_j = F, \tag{1.3}$$

where A_i, B_i, C_j, D_j and F are given matrices of conforming dimensions and X is an unknown matrix to be determined. The Eq (1.3) includes important practical problems that are written as the matrix equations

$$AX + XB = C, \tag{1.4}$$
$$AX + X^T B = C, \tag{1.5}$$
$$AXB + CXD = E, \tag{1.6}$$
$$AXB + X = C, \tag{1.7}$$

known as Sylvester, Sylvester-transpose, generalized Sylvester, Kalman-Yakubovich matrix equations, respectively. The Eqs (1.2)–(1.7) have many essential applications in control theory; see e.g., [2–6]. In traditional method, a vectorization and the Kronecker product are used to find the unique solution. However, the large size of the Kronecker multiplication leads to computationally difficulty in that excessive computer. For this reason, iterative algorithms are received more attention.

Many researchers attempted to meliorate such iterative algorithms for finding the approximated solutions of matrix equations (1.2)–(1.7) using many ideas, e.g., matrix sign function [7, 8], recursive blocked algorithms [9, 10] and Hermitian and skew-Hermitian splitting algorithms [11–13]. In 2005, gradient-based iterative algorithms were firstly introduced by F. Ding and T. Chen for solving (1.4), (1.6) and (1.7); see [14, 15]. In a few year later, many iterative algorithms that relied on gradients and hierarchical identification principle for solving (1.2)–(1.7) are established, e.g., RGI [16], JGI [17, 18], MGI [19] and AGBI [20]. See more information in [21–25]. The Frobenious norm $\|\cdot\|_F$ and the spectral norm $\|\cdot\|_2$ for matrices are used to analyse the convergence property of such algorithms. There are defined respectively for any real matrix A by

$$\|A\|_F = (\mathrm{tr}\, A^T A)^{\frac{1}{2}} \quad \text{and} \quad \|A\|_2 = (\lambda_{\max}(A^T A))^{\frac{1}{2}}.$$

A gradient-based iterative algorithm for solving (1.3) was presented as follows:

Theorem 1.1 ([25]). *Suppose that the matrix equation* (1.3) *has a unique solution X. For each* $s = 1, 2, \ldots, p$ *and* $t = p+1, \ldots, q$, *construct*

$$\begin{aligned}
X_s(k) &= X(k-1) + \tau A_s^T [F - \sum_{i=1}^{p} A_i X(k-1) B_i - \sum_{j=1}^{q} C_j X^T(k-1) D_j] B_s^T, \\
X_t(k) &= X(k-1) + \tau D_t [F - \sum_{i=1}^{p} A_i X(k-1) B_i - \sum_{j=1}^{q} C_j X^T(k-1) D_j]^T C_t, \\
X(k) &= \frac{1}{p+q} \left(\sum_{s=1}^{p} X_s(k) + \sum_{t=p+1}^{q} X_t(k) \right).
\end{aligned}$$

A necessary and sufficient condition for which the sequence $\{X_k\}$ *converges to* X *for any given initial matrices* $X_1(0), X_2(0), \ldots, X_{p+q}(0)$ *is that*

$$0 < \tau < 2\left(\sum_{i=1}^{p} \|A_s\|_2^2\|B_s\|_2^2 + \sum_{t=p+1}^{q} \|C_t\|_2^2\|D_t\|_2^2\right).$$

Meanwhile, a least-squares based iterative algorithm for solving (1.3) was presented as follows:

Theorem 1.2 ([25]). *For each* $s = 1, 2, \ldots, p$ *and* $t = p+1, \ldots, q$, *construct*

$$\begin{aligned} R(k) &= E - \sum_{i=1}^{p} A_i X(k-1) B_i - \sum_{j=1}^{q} C_j X^T(k-1) D_j, \\ X_s(k) &= X(k-1) + \mu(A_s^T A_s)^{-1} A_s^T R(k) B_s^T (B_s B_s^T)^{-1}, \\ X_t(k) &= X(k-1) + \mu(D_t D_t^T)^{-1} D_t R(k) C_t (C_t^T C_t)^{-1}, \\ X(k) &= \frac{1}{p+q}\left(\sum_{s=1}^{p} X_s(k) + \sum_{t=p+1}^{q} X_t(k)\right). \end{aligned}$$

The sequence $\{X_k\}$ *converges to a unique solution* X *for any given initial matrices* $X_1(0), X_2(0), \ldots, X_{p+q}(0)$ *if and only if* $0 < \tau < 2(p+q)$.

In this work, we propose an effective iterative algorithm based on gradient and hierarchical identification principle, namely, a gradient iterative algorithm with optimal convergence factor. The proposed algorithm is applicable for the generalized Sylvester-transpose matrix equation (1.3); see Section 2. Convergence analysis (see Section 3) via Banach fixed-point theorem reveals that the iterative solutions converges to the unique solution for any initial value if and only if the convergence factor is chosen appropriately belong to an open interval. Then we study the performance of the algorithm from the convergence rate and error estimates. Moreover, we determine the fastest asymptotic convergence rate to minimize the spectral radius of the iteration matrix. Furthermore, we apply the algorithm to the Sylvester-transpose matrix equation; see Section 4. To show the applicability and the performance of the algorithm, we give numerical experiments in Section 5. In Section 6, we summarize the whole work. For benefits of reader, we include MATLAB-code for numerical experiment in Appendix.

2. Introducing a gradient iterative algorithm

Let us denote by $\mathbb{R}^{r\times s}$ the set of $r \times s$ real matrices. Consider the matrix equation (1.3) where $A_i \in \mathbb{R}^{m\times n}$, $B_i \in \mathbb{R}^{p\times q}$, $C_j \in \mathbb{R}^{m\times p}$, $D_j \in \mathbb{R}^{n\times q}$, $F \in \mathbb{R}^{m\times q}$ are given coefficient matrices and $X \in \mathbb{R}^{n\times p}$ is an unknown matrix to be determined. The dimension matching of matrices is assumed to be $mq = np$.

2.1. A traditional method for the generalize Sylvester-transpose matrix equation

Recall that the commutation matrix K_{mn} is defined by

$$K_{mn} = \left[I_m \otimes e_1^{n^T} I_m \otimes e_2^{n^T} \cdots I_m \otimes e_n^{n^T}\right] \in \mathbb{R}^{mn\times mn}$$

where e_i^n is the ith column of the $n \times n$ identity matrix I_n. Essential properties of the commutation matrices are given in the following lemma:

Lemma 2.1. *(see e.g., [26, Ch. 4]) For any $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, we have*

$$\text{vec}(A^T) = K_{mn}\,\text{vec}(A), \tag{2.1}$$

$$K_{mn}^{-1} = K_{mn}^T = K_{nm}, \tag{2.2}$$

$$(B \otimes A) = K_{pm}(A \otimes B)K_{nq}. \tag{2.3}$$

A traditional method to solve the matrix equation (1.3) is to take the vector operator and utilize Lemma 2.1. Indeed, we arrive at an equivalent linear system $Qx = b$ where

$$Q = \sum_{i=1}^{p}(B_i^T \otimes A_i) + \sum_{j=1}^{q}(D_j^T \otimes C_j)K_{np} \in \mathbb{R}^{np\times np},$$

$x = \text{vec}[X]$ and $b = \text{vec}[F]$. From now on, assume that Q is invertible. So, the matrix equation (1.3) has a unique solution

$$x = Q^{-1}b. \tag{2.4}$$

However, if the dimensions of A_i, B_i, C_j, D_j are not small, e.g., $10^2 \times 10^2$, then the dimension of Q is $10^4 \times 10^4$. Such a dimension problem leads to computational difficulty for computation and inversion of large matrices. Hence, this approach is only applicable for small dimensional matrices.

2.2. Gradient iterative algorithm for the matrix equation

We now propose an effective iterative algorithm to solve (1.3). If $p \geqslant q$ then we set $C_j = 0$ and $D_j = 0$ for any $j > q$. If $q > p$ then we set $A_i = 0$ and $B_i = 0$ for any $i > p$. For each $i = 1, 2, \ldots, p$, we assume that

$$M_i := F - \left(\sum_{s\neq i}(A_sXB_s + C_sX^TD_s)\right). \tag{2.5}$$

From the main system (1.3), we would like to solve p subsystems

$$A_iXB_i + C_iX^TD_i = M_i \qquad i = 1, 2, \ldots, p, \tag{2.6}$$

so that the following errors are minimized:

$$L_i(X) := \|A_iXB_i + C_iX^TD_i - M_i\|_F^2, \qquad i = 1, 2, \ldots, p. \tag{2.7}$$

We can derive the gradient of each L_i as follows:

$$\begin{aligned}\frac{\partial}{\partial X}L_i(X) &= \frac{\partial}{\partial X}\,\text{tr}[(A_iXB_i + C_iX^TD_i - M_i)^T(A_iXB_i + C_iX^TD_i - M_i)]\\ &= \frac{\partial}{\partial X}\,\text{tr}(B_i^TX^TA_i^TA_iXB_i) + \frac{\partial}{\partial X}\,\text{tr}(D_i^TXC_i^TA_iXB_i) - \frac{\partial}{\partial X}\,\text{tr}(M_i^TA_iXB_i)\\ &\quad + \frac{\partial}{\partial X}\,\text{tr}(B_i^TX^TA_i^TC_iX^TD_i) + \frac{\partial}{\partial X}\,\text{tr}(D_i^TXC_i^TC_iX^TD_i) - \frac{\partial}{\partial X}\,\text{tr}(M_i^TC_iX^TD_i)\end{aligned}$$

$$
\begin{aligned}
&-\frac{\partial}{\partial X}\operatorname{tr}(B_i^T X^T A_i^T M_i) - \frac{\partial}{\partial X}\operatorname{tr}(D_i^T X C_i^T M_i) + \frac{\partial}{\partial X}\operatorname{tr}(M_i^T M_i) \\
= \;& 2[A_i^T(A_i X B_i + C_i X^T D_i - M_i)B_i^T + D_i(A_i X B_i + C_i X^T D_i - M_i)^T C_i]. \qquad (2.8)
\end{aligned}
$$

Let $X_i(k)$ be the approximated solution of the system (2.6) at iteration k. The recursive formula of $X_i(k)$ come from the gradient formula of $L_i(X)$ as follows:

$$X_i(k) \;=\; X_i(k-1) + \mu\frac{\partial}{\partial X}L_i(X),$$

where μ is a step size parameter. To avoid duplicated computation, we introduce a matrix

$$E = F - \sum_{i=1}^{p} A_i X B_i - \sum_{j=1}^{q} C_j X^T D_j.$$

Taking the arithmetic mean of $X_1(k), \ldots, X_p(k)$ to get $X(k)$:

$$X(k) \;=\; \frac{1}{p}\sum_{i=1}^{p} X_i(k) \;=\; X(k-1) + \tau\left(\sum_{i=1}^{p} A_i^T E B_i^T + \sum_{j=1}^{q} D_j E^T C_j\right), \qquad (2.9)$$

where $\tau := -2\mu/p$ is called a convergence factor. The hierarchical identification principle suggests to replace the unknown solution X in (2.9) by its previous estimate $X(k-1)$. Thus we obtain the following procedure:

Algorithm 1: GIO for generalized Sylvester-transpose equation

$A_i \in \mathbb{R}^{m\times n}$, $B_i \in \mathbb{R}^{p\times q}$, $C_j \in \mathbb{R}^{m\times p}$, $D_j \in \mathbb{R}^{n\times q}$ for $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, q$ and $F \in \mathbb{R}^{m\times q}$.
initialization;
if $p \geq q$ **then**
  $C_j = 0$ and $D_j = 0$ for any $j > q$;
else
  $A_i = 0$ and $B_i = 0$ for any $i > p$.
end
Set $A_i' = A_i^T$ and $B_i' = B_i^T$. Choose $\tau \in \mathbb{R}$. Set $k := 0$. Choose initial matrix $X(0)$.
while $k = 0, 1, 2, \ldots, n$ **do**
  $E(k) \;=\; F - \sum_{i=1}^{p} A_i X(k) B_i - \sum_{j=1}^{q} C_j X^T(k) D_j$.
  if $\|E(k)\|_F / \|F\|_F < \epsilon$ **then**
    break;
  else
    $X(k+1) \;=\; X(k) + \tau\left(\sum_{i=1}^{p} A_i' E(k) B_i' + \sum_{j=1}^{q} D_j E^T(k) C_j\right)$,
    update k.
  end
end

For convenience, we write $X^T(k)$ and $E^T(k)$ for $X(k)^T$ and $E(k)^T$, respectively. The matrices $E(k), A_i', B_i'$ were introduced to avoid duplicate manipulations.

Remark 2.2. *To break the procedure, if $\|F\|_F$ is close to zero, then we should consider the error $\|E(k)\|_F$ or $\|X(k) - X(k-1)\|_F$ instead of the relative error $\|E(k)\|_F / \|F\|_F$.*

The convergent property of the algorithm relies on the convergence factor τ, which will be discussed in the next section.

3. Convergence analysis of GIO algorithm

In this section, we discuss a convergence criteria for the applicability of Algorithm 1. Then, we analyse the performance of the algorithm through its convergence rate and error estimates. The last job is to determine the convergence factor for which the algorithm fits with the fastest asymptotic behaviour. The main idea for the analysis starts with transforming a recursive equation of the error of approximated solutions into a fixed-point iteration $x(k) = S\,x(k-1)$, where $S : \mathbb{R}^{np} \to \mathbb{R}^{np}$ is a contraction mapping. Then, we apply the following Banach fixed-point theorem to analysis the algorithm.

Theorem 3.1. *(see e.g., [27]) Let (X, d) be a non-empty complete metric space with a contraction mapping $T : X \to X$. Then*

(i) The map T admits a unique fixed-point x^* in X i.e. $T(x^*) = x^*$.
(ii) The fixed-point x^* can be found as follows: start with an arbitrary element x_0 in X and define a sequence $\{x_n\}$ by $x_n = T(x_{n-1})$ for $n \geq 1$. Then $x_n \to x^*$.
(iii) The following inequalities hold and describe the speed of convergence:

$$d(x^*, x_n) \leq \frac{z^n}{1-z} d(x_1, x_0) \tag{3.1}$$

$$d(x^*, x_{n+1}) \leq \frac{z}{1-z} d(x_{n+1}, x_n) \tag{3.2}$$

$$d(x^*, x_{n+1}) \leq z d(x^*, x_n). \tag{3.3}$$

3.1. Convergence criteria

From Algorithm 1, at each k-th iteration, we start with considering the error matrix $\hat{X}(k) = X(k) - X$. Indeed, we have

$$\begin{aligned} \hat{X}(k) &= X(k-1) + \tau \left(\sum_{i=1}^{p} A_i^T E(k-1) B_i^T + \sum_{j=1}^{q} D_j E^T(k-1) C_j \right) - X \\ &= \hat{X}(k-1) + \tau \left(\sum_{i=1}^{p} A_i^T E(k-1) B_i^T + \sum_{j=1}^{q} D_j E^T(k-1) C_j \right), \end{aligned} \tag{3.4}$$

and

$$E(k-1) = -\left(\sum_{i=1}^{p} A_i \hat{X}(k-1) B_i + \sum_{j=1}^{q} C_j \hat{X}^T(k-1) D_j \right).$$

Thus, Lemma 2.1 implies that

$$\begin{aligned} \operatorname{vec} E(k-1) &= -\sum_{i=1}^{p} (B_i^T \otimes A_i) \operatorname{vec} \hat{X}(k-1) - \sum_{j=1}^{q} (D_j^T \otimes C_j) \operatorname{vec} \hat{X}^T(k-1) \\ &= -\sum_{i=1}^{p} (B_i^T \otimes A_i) \operatorname{vec} \hat{X}(k-1) - \sum_{j=1}^{q} (D_j^T \otimes C_j) K_{np} \operatorname{vec} \hat{X}(k-1) \end{aligned}$$

$$= -Q \operatorname{vec} \hat{X}(k-1). \tag{3.5}$$

Taking the vector operator to the Eq (3.4) and utilizing (3.5), we get

$$\begin{aligned}
\operatorname{vec} \hat{X}(k) &= \operatorname{vec} \hat{X}(k-1) + \tau \left[\sum_{i=1}^{p} \operatorname{vec} A_i^T E(k-1) B_i^T + \sum_{j=1}^{q} \operatorname{vec} D_j E^T(k-1) C_j \right] \\
&= \operatorname{vec} \hat{X}(k-1) + \tau \left[\sum_{i=1}^{p} (B_i \otimes A_i^T) \operatorname{vec} E(k-1) + \sum_{j=1}^{q} (C_j^T \otimes D_j) K_{mq} \operatorname{vec} E(k-1) \right] \\
&= \operatorname{vec} \hat{X}(k-1) - \tau \left[\sum_{i=1}^{p} (B_i \otimes A_i^T) Q \operatorname{vec} \hat{X}(k-1) + \sum_{j=1}^{q} K_{pn} (D_j \otimes C_j^T) Q \operatorname{vec} \hat{X}(k-1) \right] \\
&= (I_{np} - \tau Q^T Q) \operatorname{vec} \hat{X}(k-1).
\end{aligned}$$

Letting $S = I_{np} - \tau Q^T Q$ and $x(k) = \operatorname{vec} \hat{X}(k-1)$, we get a linear iteration

$$x(k) = S\, x(k-1). \tag{3.6}$$

Using Theorem 3.1 and [28, Theorem 1], we deduce that the following are equivalent:

(i) the sequence $\{X(k)\}$ converges to X for any initial value $X(0)$;
(ii) S is a contraction mapping (here, we view $S : \mathbb{R}^{np} \to \mathbb{R}^{np}$ as a mapping);
(iii) the spectral norm of S is less than 1.

Since S is symmetric, all its eigenvalue are real. Note that the eigenvalues of S are of the form $1 - \tau\lambda$ where λ is any eigenvalue of $Q^T Q$. We can compute the spectral radius of S as follows:

$$\rho[S] = \max\{|1 - \tau\lambda_{\max}(Q^T Q)|, |1 - \tau\lambda_{\min}(Q^T Q)|\}. \tag{3.7}$$

Thus, $\rho[S] < 1$ if and only if

$$0 < \tau\lambda_{\max}(Q^T Q) < 2. \tag{3.8}$$

Since Q is invertible, the matrix $Q^T Q$ is positive definite and hence, $\|Q\|_2^2 = \lambda_{\max}(Q^T Q) > 0$. The condition (3.8) now becomes

$$0 < \tau < \frac{2}{\|Q\|_2^2}. \tag{3.9}$$

We summarize convergence criteria for Algorithm 1 as follows:

Theorem 3.2. *Consider the matrix equation* (1.3) *under the assumption that the matrix Q is invertible (i.e., Eq* (1.3) *has a unique solution). Let* $\tau \in \mathbb{R}$. *Then a necessary and sufficient condition for which Algorithm 1 is applicable for any initial matrix X(0) is the condition* (3.9).

3.2. Performance of the algorithm

We now discuss the performance of Algorithm 1 through its convergence rate and error estimates. Considering Eq (3.6), we get

$$\|X(k) - X\|_F = \|\hat{X}(k)\|_F = \|\operatorname{vec} \hat{X}(k)\|_F$$

$$= \|S \operatorname{vec} \hat{X}(k-1)\|_F \leq \|S\|_2 \|\operatorname{vec} \hat{X}(k-1)\|_F.$$

Since S is symmetric, we have $\|S\|_2 = \rho[S]$. Thus the Eq (3.3) implies that

$$\|X(k) - X\|_F \leq \rho[S] \, \|X(k-1) - X\|_F, \tag{3.10}$$

By induction, we obtain

$$\|X(k) - X\|_F \leq \rho^k[S] \, \|X(0) - X\|_F. \tag{3.11}$$

According to the estimate (3.11), the asymptotic convergence rate of the algorithm relies on $\rho[S]$. Moreover, given an error $\epsilon > 0$, we would like to find the iteration number k for which

$$\rho^k(S)\|X(0) - X\|_F < \epsilon. \tag{3.12}$$

By taking the 10th-base logarithms, we obtain the following equivalent requirement:

$$k > \frac{\log \epsilon - \log \|X(0) - X\|_F}{\log \rho(S)}. \tag{3.13}$$

Moreover, by Theorem 3.1(iii), we have

$$\|X(k) - X\|_F \leq \frac{\rho^k[S]}{1 - \rho[S]} \, \|X(1) - X(0)\|_F, \tag{3.14}$$

$$\|X(k+1) - X\|_F \leq \frac{\rho[S]}{1 - \rho[S]} \, \|X(k+1) - X(k)\|_F. \tag{3.15}$$

The following results are discussed to concluding the convergence rate and several estimates of the proposed algorithm.

Theorem 3.3. *Suppose the convergence factor τ is chosen so that Algorithm 1 is applicable for any initial matrix X(0).*

(i) The spectral radius $\rho[S]$ in (3.7) *governs the asymptotic convergence rate of the algorithm.*

(ii) Equations (3.10) *and* (3.11) *show the error estimates $\|X(k) - X\|_F$ compared to the previous step and the first step, respectively. Meanwhile, the error at each iteration reduces from the previous one.*

(iii) The prior and posterior estimates are presented in (3.14) *and* (3.15)*, respectively.*

(iv) For each given error $\varepsilon > 0$, we have $\|X(k) - X\|_F < \epsilon$ after the k-th iteration for any $k \in \mathbb{N}$ that satisfies (3.13).

We can see that A_i, B_i, C_j, D_j affect the convergence rate of Algorithm 1 but E is not. However, the matrix E is required for stopping process. Furthermore, If we take $\epsilon = 0.5 \times 10^{-n}$ in (3.13) and k satisfies

$$k > \frac{\log 0.5 - \log \|X(0) - X\|_F - n}{\log \rho(S)},$$

then the approximated $X(k)$ has an accuracy of n decimal digit.

3.3. Optimal convergence factor

The fastest convergence factor of Algorithm 1 is discussed intensively. Recall that the condition number of a matrix A (relative to the spectral norm) is defined by

$$\kappa(A) = \left(\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}\right)^{\frac{1}{2}}.$$

Assume that Eq (3.9) holds. The convergence rate of Algorithm 1 is the same as that of the linear iteration (3.6), and thus, it is given by the spectral radius (3.7) of the iteration matrix S. The fastest convergence rate is equivalent to the smallest of $\rho[S]$. Thus, we would like to minimize this spectral radius subject to the condition (3.9). Now, we apply the following lemma:

Lemma 3.4. *([19]) For any real number a, b with $b > a > 0$, we have*

$$\min_{0<x<\frac{2}{b}} \{\max\{|\,1-ax\,|, |\,1-bx\,|\}\} = \frac{b-a}{b+a}.$$

The minimality is reached at $x_{opt} = \dfrac{2}{a+b}$.

Then the minimum value of $\rho[S]$ is

$$\tau_{opt} = \frac{2}{\lambda_{\max}(Q^TQ) + \lambda_{\min}(Q^TQ)}. \tag{3.16}$$

Meanwhile, we can notice that spectral radius of the iteration matrix is

$$\rho[S] = \frac{\lambda_{\max}(Q^TQ) - \lambda_{\min}(Q^TQ)}{\lambda_{\max}(Q^TQ) + \lambda_{\min}(Q^TQ)} = \frac{\kappa^2(Q)-1}{\kappa^2(Q)+1}. \tag{3.17}$$

Thus, we obtain the following theorem:

Theorem 3.5. *Among the convergence factors τ that meet the criteria of Algorithm 1, the one in Eq* (3.16) *attains the fastest asymptotic convergence rate of the algorithm, which is governed by the spectral radius* (3.17)*.*

The above theorem tells us that if $\lambda_{\max}(Q^TQ)$ is neighbouring to $\lambda_{\min}(Q^TQ)$, or equivalently, the condition number is close to one then Algorithm 1 has a fast convergence.

4. The proposed algorithm for Sylvester-transpose matrix equation

In this section, we consider an important special case of the matrix equation (1.3), namely, the Sylvester-transpose matrix equation. We apply GIO algorithm for this equation, and investigate the convergence property of the algorithm.

Let $m, n \in \mathbb{N}$. Consider the Sylvester-transpose matrix equation (1.5) where $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$, $F \in \mathbb{R}^{m\times m}$ are given constant matrices and $X \in \mathbb{R}^{n\times m}$ is an unknown matrix to be solved. Suppose that (1.5) has a unique solution, i.e., that following matrix is invertible:

$$P := (I \otimes A) + (B^T \otimes I)K_{nm} \in \mathbb{R}^{mn\times mn}.$$

Algorithm 2: GIO for Sylvester-transpose equation

initialization;
Set $A_i' = A_i^T$ and $B_i' = B_i^T$. Choose $\tau \in \mathbb{R}$. Set $k := 0$. Choose initial matrix $X(0)$.
while $k = 0, 1, 2, \ldots, n$ **do**
  $E(k) = F - AX(k) - X^T(k)B$.
  if $\|E(k)\|_F / \|F\|_F < \epsilon$ **then**
    break;
  else
    $X(k+1) = X(k) + \tau[A'E(k) + BE^T(k)]$,
    update k.
  end
end

Corollary 4.1. *Consider the matrix equation* (1.5) *when the matrix P is invertible. Let $\tau \in \mathbb{R}$. Then the given statements hold:*

(i) A equivalent condition for which Algorithm 2 is applicable for any initial matrix $X(0)$ is

$$0 < \tau < \frac{2}{\|P\|_2^2}. \tag{4.1}$$

Meanwhile, the spectral radius of the iteration matrix $T = I_{np} - \tau P^T P$ is given by

$$\rho[T] = \max\{|1 - \tau\lambda_{\max}(P^T P)|, |1 - \tau\lambda_{\min}(P^T P)|\}. \tag{4.2}$$

(ii) The spectral radius $\rho[T]$ in (4.2) *represents the asymptotic convergence rate of Algorithm 2.*
(iii) The fastest asymptotic convergence factor is determined by

$$\tau_{opt} = \frac{2}{\lambda_{\max}(P^T P) + \lambda_{\min}(P^T P)}. \tag{4.3}$$

5. Numerical simulations with discussion

In this section, we report some numerical results to illustrate the applicability the effectiveness of Algorithm 1. All simulations have been carried out by MATLAB R2018a, AMD Ryzen7 3700U with Redeon Vega Mobile Gfx @ 2.30 GHz, RAM 12.00 GB PC environment. In Example 5.1, we show that our algorithm is applicable for small-square-matrices in the case $p > q$. We also show that our algorithm is applicable and efficient for large-square-matrices and consider the effect of changing the convergence factor τ in Example 5.2. In Example 5.3, we illustrate the efficiency of Algorithm 1 when coefficients are non-square matrices of different moderate sizes. In Examples 5.4 and 5.5, we do experiments in the cases $p > q$ and $q > p$ with rectangular coefficient matrices. In Example 5.6, we test Algorithm 2 for the Sylvester-transpose equation with square coefficient matrices. In all examples, we compare the proposed algorithm to both the traditional method (Eq (2.4)) and recent iterative algorithms. The computational time is measured in seconds by MATLAB functions **tic** and **toc**.

Example 5.1. *Consider the matrix equation*

$$A_1XB_1 + C_1X^TD_1 + A_2XB_2 = F,$$

where

$$A_1 = \begin{bmatrix} -0.123 & 0.002 & 0.780 & -0.563 & 0.009 \\ -0.123 & -0.008 & 0.005 & 0.097 & 0.002 \\ 0.398 & 0.007 & -0.023 & 0.094 & 0.001 \\ -0.009 & 0.478 & -0.994 & 0.001 & 0.005 \\ 0.013 & -0.003 & 0.028 & 0.004 & -0.456 \end{bmatrix}, A_2 = \begin{bmatrix} 0.112 & -0.302 & -0.785 & 0.312 & -0.049 \\ 0.709 & -0.996 & -0.733 & 0.219 & -0.005 \\ 0.261 & -0.005 & -0.003 & 0.114 & -0.111 \\ 0.219 & 0.005 & -0.123 & -0.125 & 0.009 \\ 0.001 & 0.000 & 0.018 & -0.994 & 0.956 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0.667 & -0.209 & 0.346 & -0.675 & -0.099 \\ 0.099 & -0.218 & 0.278 & -0.219 & 0.004 \\ -0.002 & 0.005 & 0.109 & 0.678 & -0.234 \\ 0.056 & -0.005 & -0.006 & 0.195 & 0.009 \\ 0.004 & 0.065 & -0.187 & -0.984 & 0.000 \end{bmatrix}, B_2 = \begin{bmatrix} -0.004 & 0.056 & -0.005 & 0.004 & 0.049 \\ 0.579 & 0.096 & 0.114 & -0.008 & 0.112 \\ -0.113 & -0.119 & 0.284 & -0.003 & 0.014 \\ 0.089 & 0.027 & -0.009 & -0.145 & 0.036 \\ -0.001 & -0.079 & 0.456 & -0.458 & 1.000 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} -0.163 & 0.021 & 0.007 & -0.152 & 0.193 \\ -0.474 & -0.098 & 0.001 & 0.384 & 0.193 \\ -0.085 & 0.109 & 0.093 & -0.017 & 0.173 \\ 0.812 & -0.742 & -0.841 & 0.941 & 0.485 \\ 0.197 & 0.934 & 0.012 & 0.845 & -0.917 \end{bmatrix}, D_1 = \begin{bmatrix} -0.002 & 0.074 & 0.004 & -0.072 & 0.284 \\ 0.056 & 0.037 & 0.485 & 0.188 & 0.485 \\ 0.863 & -0.072 & 0.475 & 0.945 & -0.594 \\ 0.016 & -0.034 & 0.004 & 0.001 & 0.855 \\ 0.854 & 0.003 & 0.927 & -0.923 & 0.567 \end{bmatrix}.$$

In fact, the unique solution is given by

$$X = \begin{bmatrix} 1.000 & 0.010 & -0.224 & -0.111 & 0.908 \\ 0.980 & 0.765 & -0.365 & 0.482 & 0.528 \\ -0.649 & 0.309 & 0.849 & -0.030 & 0.612 \\ -0.495 & 0.008 & 0.862 & -0.001 & -0.004 \\ 0.239 & 0.937 & 0.251 & 0.364 & 0.062 \end{bmatrix}.$$

Let us apply Algorithm 1 to compute the sequence $\{X(k)\}$ *of approximated solutions. Take an initial point* $X(0) = 0$. *The optimal convergence factor can be computed according to Theorem 3.2 as follows:*

$$\tau_{opt} = \frac{2}{\lambda_{\min}(Q^TQ) + \lambda_{\max}(Q^TQ)} \approx \frac{2}{8.3389 \times 10^{-6} + 14.5024} \approx 0.1379.$$

Table 1 shows that the direct method consumes 15 ms to get the exact solution, while GIO algorithm takes 6 ms to perform 10 *iterations in order to get an approximated solution with a small relative error. Figure 1 and Table 1 illustrate that the computational time of GIO algorithm is less than that for GI algorithm with preferable relative error. Figure 1 and Table 1 also show that GIO algorithm is outperform than LS algorithm.*

Table 1. Numerical results for Example 5.1.

Algorithm	IT	CT	relative error
Direct	-	0.0150	-
GIO	10	0.0060	0.5088
GI with $\tau = 0.127$	10	0.0078	0.7510
GI with $\tau = 0.009$	10	0.0069	0.9755
LS	10	0.0223	1.0000

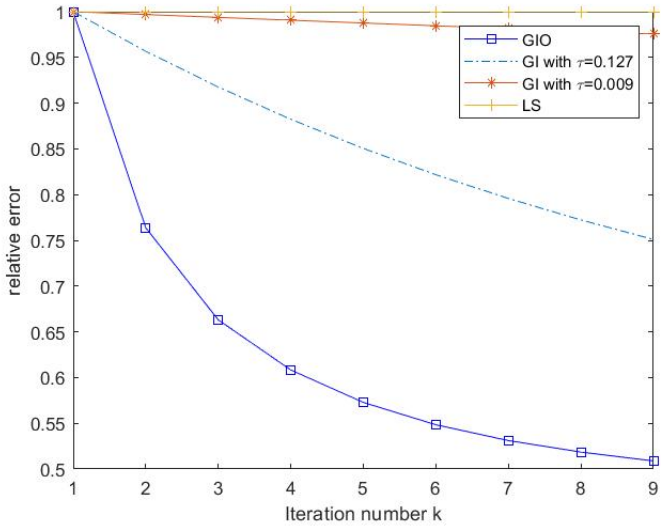

Figure 1. Relative error for Example 5.1.

Example 5.2. *Consider the matrix equation*

$$\sum_{i=1}^{2} A_i X B_i + \sum_{j=1}^{2} C_j X^T D_j = F$$

where all coefficients are 100×100 *tridiagonal matrices given by*

$$A_1 = \text{tridiag}(3, 1, -1),\ A_2 = \text{tridiag}(1, 0, 4),\ B_1 = \text{tridiag}(-1, 3, 2),\ B_2 = \text{tridiag}(-1, -2, -1),$$
$$C_1 = \text{tridiag}(1, 0, -2),\ C_2 = \text{tridiag}(1, -2, 3),\ D_1 = \text{tridiag}(0, 2, -4),\ D_2 = \text{tridiag}(1, -1, 1).$$

In fact, the unique solution is given by $X = \text{tridiag}(0, 1, -1)$. *Let us apply Algorithm 1 to compute the sequence* $\{X(k)\}$ *of approximated solutions. Take an initial point*

$$X(0) = 10^{-6} \times \text{tridiag}(-1, -1, -1).$$

The optimal convergence factor can be computed according to Theorem 3.2 as follows:

$$\tau_{opt} = \frac{2}{\lambda_{\min}(Q^T Q) + \lambda_{\max}(Q^T Q)} \approx \frac{2}{4.15 \times 10^{-13} + 7.15 \times 10^4} \approx 0.000028.$$

The computing results of changing τ are listed in Figure 2 and Table 2. Figure 2 shows that, for the given initial point, as k increases, for $\tau = \tau_{opt}, \tau = 0.00002, \tau = 0.000015$, and $\tau = 0.00001$, we see that the terms $\|E(k)\|_F/\|F\|_F$ are becoming smaller and approaching to zero. Moreover, the relative errors $\|E(k)\|_F/\|F\|_F$ for τ_{opt} go faster to 0 than those for another convergence factors. Furthermore, when $\tau = 0.00004$ and $\tau = -0.00001$ which do not satisfy the criteria (3.9), the approximated solutions diverge. Table 2 confirms that the computational time of the proposed algorithm is significantly less than the time of the traditional method. Thus, in this case, Algorithm 1 is applicable and effective.

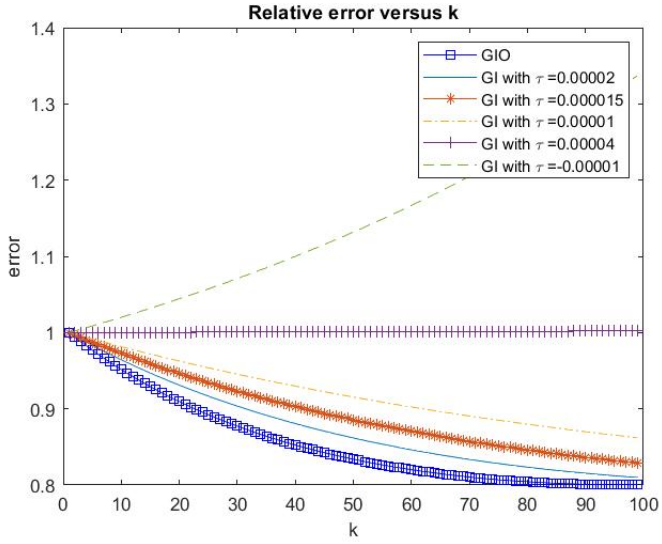

Figure 2. Relative errors for Example 5.2.

Table 2. Numerical results for Example 5.2.

Algorithm	IT	CT	relative error
Direct	-	15.2058	-
GIO	100	0.5467	0.8005
GI with $\tau = 0.00002$	100	1.6245	0.8099
GI with $\tau = 0.000015$	100	1.7007	0.8512
GI with $\tau = 0.00001$	100	2.8129	0.8619
GI with $\tau = 0.00004$	100	2.9998	1.0055
GI with $\tau = -0.00001$	100	4.0097	1.3314

Example 5.3. *Consider the matrix equation*

$$\sum_{i=1}^{3} A_i X B_i + \sum_{j=1}^{3} C_j X^T D_j = F$$

when $A_i \in \mathbb{R}^{40\times60}$, $B_i \in \mathbb{R}^{20\times30}$, $C_j \in \mathbb{R}^{40\times20}$, $D_j \in \mathbb{R}^{60\times30}$ and $F \in \mathbb{R}^{40\times30}$ are tridiagonal matrices given

by

$$A_1 = \text{tridiag}(1,-1,1), \quad A_2 = \text{tridiag}(2,0,-3), \quad A_3 = \text{tridiag}(-2,-1,-2),$$
$$B_1 = \text{tridiag}(1,-3,0), \quad B_2 = \text{tridiag}(-1,-2,-1), \quad B_3 = \text{tridiag}(0,1,-3),$$
$$C_1 = \text{tridiag}(-3,0,-2), \quad C_2 = \text{tridiag}(-1,-2,3), \quad C_3 = \text{tridiag}(2,-1,2),$$
$$D_1 = \text{tridiag}(0,2,-1), \quad D_2 = \text{tridiag}(1,2,-1), \quad D_3 = \text{tridiag}(0,1,-1).$$

Then the unique solution is $X = \text{tridiag}(0,1,-1) \in \mathbb{R}^{60\times 20}$. *We take the initial matrix*

$$X(0) = 10^{-6} \times \text{tridiag}(1,1,1).$$

The computing results of changing τ *are listed in Figure 3. As k large enough, the terms* $\|E(k)\|_F/\|F\|_F$ *for* $\tau_{opt} \approx 0.000085$ *are becoming smaller and go faster to zero than another convergence factors. Moreover, for* $\tau = 0.0003$ *and* $\tau = -0.000001$ *which do not satisfy the condition* (3.9)*, we see that the term* $\|E(k)\|_F/\|F\|_F$ *do not approach zero, so the approximated solutions diverge. From Table 3, it is clear that the computational time of GIO algorithm is significantly less than the time of the traditional method and another GI with different convergence factors.*

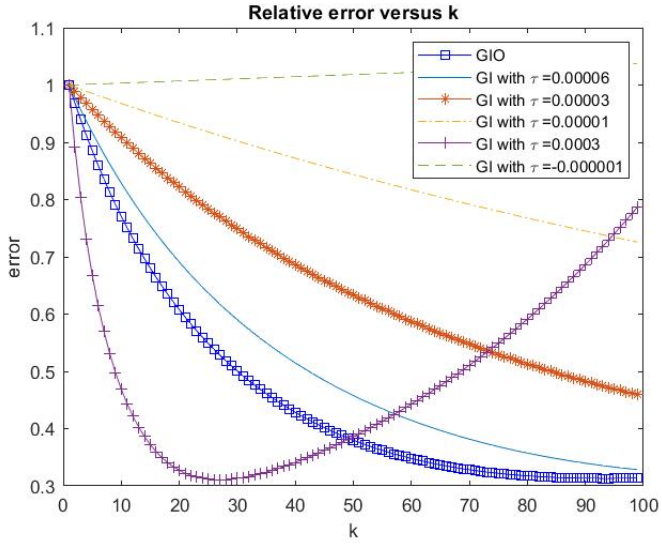

Figure 3. Relative error for Example 5.3.

Table 3. Numerical results for Example 5.3.

Algorithm	IT	CT	relative error
Direct	-	4.7478	-
GIO	100	0.2350	0.3012
GI with $\tau = 0.00006$	100	1.6954	0.3465
GI with $\tau = 0.00003$	100	2.7760	0.4802
GI with $\tau = 0.00001$	100	3.9008	0.7219
GI with $\tau = 0.0003$	100	4.4435	0.7906
GI with $\tau = -0.00001$	100	4.0978	1.0244

Example 5.4. *Consider the matrix equation*

$$\sum_{i=1}^{3} A_i X B_i + \sum_{j=1}^{2} C_j X^T D_j = F$$

when $A_i \in \mathbb{R}^{40\times 60}$, $B_i \in \mathbb{R}^{20\times 30}$, $C_j \in \mathbb{R}^{40\times 20}$, $D_j \in \mathbb{R}^{60\times 30}$ *and* $F \in \mathbb{R}^{40\times 30}$ *are tridiagonal matrices given by*

$$\begin{aligned}
&A_1 = \text{tridiag}(-2, 2, 2), \quad A_2 = \text{tridiag}(3, 3, -4), \quad A_3 = \text{tridiag}(3, -1, 2),\\
&B_1 = \text{tridiag}(2, 3, 5), \quad B_2 = \text{tridiag}(-2, -3, 1), \quad B_3 = \text{tridiag}(-1, 0, 3),\\
&C_1 = \text{tridiag}(-3, 4, -2), \quad C_2 = \text{tridiag}(2, 2, -3),\\
&D_1 = \text{tridiag}(5, 3, -1), \quad D_2 = \text{tridiag}(1, 2, 3).
\end{aligned}$$

Then the unique solution is $X = \text{tridiag}(1, 2, -1) \in \mathbb{R}^{60\times 20}$. *We take the initial matrix*

$$X(0) \;=\; 10^{-6} \times \text{tridiag}(-1, -1, -1) \in \mathbb{R}^{60\times 20}.$$

The numerical results in Figure 4 show that the direct method consumes around 11 *seconds, while GIO algorithm spends* 0.047 *seconds (*50 *iterations) with satisfactory error. Figure 4 and Table 4 show that the computational time of GIO algorithm is slightly more than that of GI algorithm, as the relative error of GIO is significantly less than that of GI algorithm. Figure 4 and Table 4 also show that GIO algorithm is outperform than LS algorithm in both computational time and relative error.*

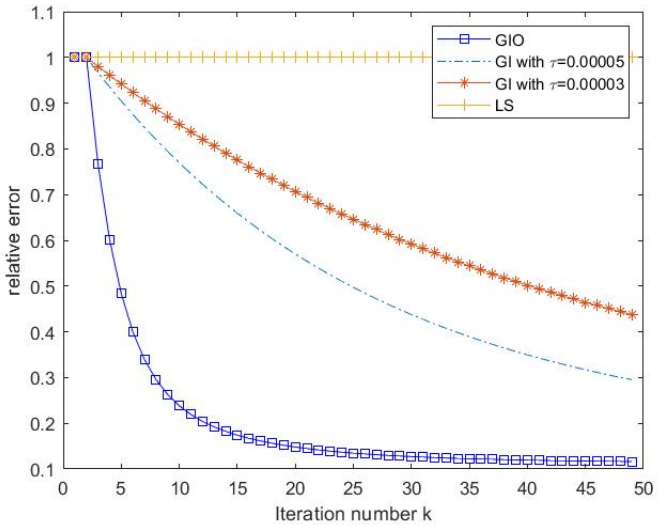

Figure 4. Relative error for Example 5.4.

Table 4. Numerical results for Example 5.4.

Algorithm	IT	CT	relative error
Direct	-	10.8753	-
GIO	50	0.0470	0.1163
GI with $\tau = 0.00005$	50	0.0411	0.2952
GI with $\tau = 0.00003$	50	0.0367	0.4376
LS	50	0.2664	1.0000

Example 5.5. *Consider the matrix equation*

$$\sum_{i=1}^{2} A_i X B_i + \sum_{j=1}^{3} C_j X^T D_j = F$$

where all coefficients are 10×10 *tridiagonal matrices given by*

$$A_1 = \text{tridiag}(-1, 2, 1),\ A_2 = \text{tridiag}(2, -4, -3),\ B_1 = \text{tridiag}(1, 3, 2),\ B_2 = \text{tridiag}(-2, -3, -1),$$
$$C_1 = \text{tridiag}(2, 3, 1),\ C_2 = \text{tridiag}(1, -3, -1),\ C_3 = \text{tridiag}(5, 3, 4),$$
$$D_1 = \text{tridiag}(-1, 2, -1),\ D_2 = \text{tridiag}(4, 2, 1),\ D_3 = \text{tridiag}(2, 3, 1).$$

In fact, the unique solution is given by $X = \text{tridiag}(1, 1, 1)$. *Let us apply Algorithm 1 to compute the sequence* $\{X(k)\}$ *of approximated solutions using an initial point*

$$X(0) = 10^{-6} \times \text{tridiag}(-1, -1, -1).$$

*Table 5 shows that for small size matrices, the difference between the computational times for the direct method and GIO algorithm (*50 *iterations) is not much as that for the moderate sizes in Example 5.4. Figure 5 and Table 5 illustrate that the computational time of GIO algorithm is less than that for GI algorithm with preferable relative error. Figure 5 and Table 5 also show that GIO algorithm is outperform than LS algorithm.*

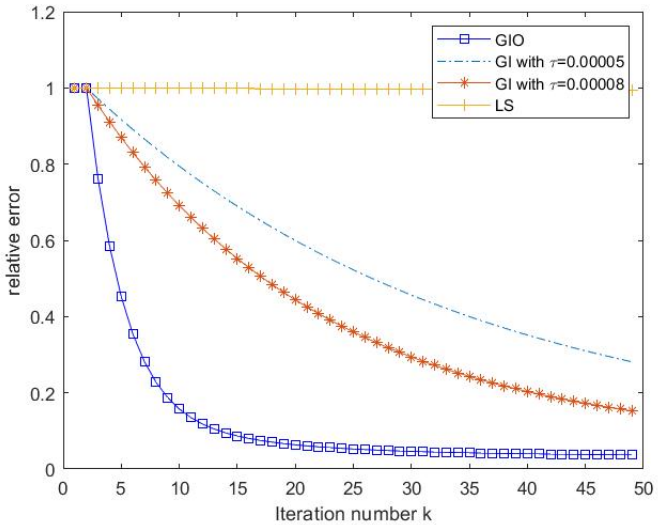

Figure 5. Relative error for Example 5.4.

Table 5. Numerical results for Example 5.5.

Algorithm	IT	CT	relative error
Direct	-	0.0266	-
GIO	50	0.0112	0.0370
GI with $\tau = 0.00005$	50	0.0146	0.2813
GI with $\tau = 0.00008$	50	0.0134	0.1526
LS	50	0.0621	0.9942

Example 5.6. *The Sylvester-transpose equation* $AX + X^T B = F$, *where* $A, B, F, X \in \mathbb{R}^{10\times 10}$ *are given by*

$$A = \text{tridiag}(1, -3, 1),\ \ B = \text{tridiag}(2, 2, 4),\ \ X = \text{tridiag}(4, 1, 4)$$

is considered by using the initial matrix $X(0) = \text{tridiag}(10^{-6}, 10^{-6}, 10^{-6})$. *The objective is to compare the capability of GIO algorithm to GI [25], LS [25], and AGBI [20] algorithms. We fix the iteration number to be* 50 *and investigate the relative error* $\|E(k)\|_F/\|F\|_F$. *Figure 6 and Table 6 indicate that Algorithm 2 is more efficient than the traditional method and another iterative algorithms in spite of a little more computational time.*

Table 6. Numerical results for Example 5.6.

Algorithm	IT	CT	relative error
Direct	-	0.0303	-
GIO	50	0.0056	0.8621
GI	50	0.0070	0.8770
LS	50	0.0121	0.9884
AGBI	50	0.0105	0.8838

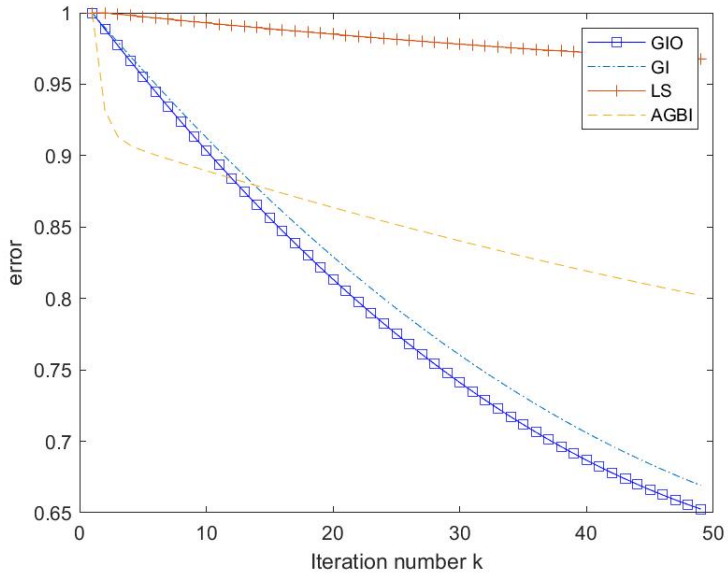

Figure 6. Relative error for Example 5.6.

6. Conclusions

The proposed algorithm, a gradient iterative algorithm with an optimal convergence factor, is applicable for a generalized Sylvester-transpose matrix equation (1.3) under the assumption that the matrix equation has a unique solution. The analysis tells us that a necessary and sufficient condition for which Algorithms 1 and 2 are applicable for any initial matrix $X(0)$ is the convergence factor is chosen appropriately belong to an open interval. The convergence rate and several estimations (e.g., error, prior, posterior) depend on the spectral radius of the associated iteration matrix. Moreover, we can determine the fastest asymptotic convergence rate. The numerical experiments illustrate that the algorithm can be applied for any matrix problems with conformable coefficient matrices of small/moderate/large sizes and square/non-square sizes. Moreover, the algorithm has more efficient than the traditional method and another recent gradient-based iterative algorithms.

Acknowledgements

The first author would like to thank Ministry of Education, Thailand for a financial support during her PhD study.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. G. Dullerud, F. Paganini, *A course in robust control theory – a convex approach*, New York: Springer-Verlag, 1994.
2. C. C. Tsui, On robust observer compensator design, *Automatica*, **24** (1988), 687–692.
3. P. V. Dooren, Reduce order observer: A new algorithm and proof, *Syst. Control Lett.*, **4** (1984), 243–251.
4. H. K. Wimmer, Consistency of a pair of generalized Sylvester equations, *IEEE T. Automat. Contr.*, **39** (1994), 1014–1016.
5. A. Wu, G. Duan, Y. Xue, Kronecker maps and Sylvester-polynomial matrix equations, *IEEE T. Automat. Contr.*, **52** (2007), 905–910.
6. A. Wu, G. Duan, B. Zhou, Solution to generalized Sylvester matrix equations, *IEEE T. Automat. Contr.*, **53** (2008), 811–815.
7. R. Bartels, G. Stewart, Solution of the matrix equation $AX + XB = C$, *Commun. ACM*, **15** (1972), 820–826.
8. P. Benner, S. Quintana, Solving stable generalized Lyapunov matrix equations with the matrix sign function, *Numer. Algorithms*, **20** (1999), 75–100.
9. I. Jonsson, B. Kagstrom, Recursive blocked algorithms for solving triangular systems-Part I: One-sided and couple Sylvester-type matrix equations, *ACM T. Math. Software*, **28** (2002), 392–415.

10. I. Jonsson, B. Kagstrom, Recursive blocked algorithms for solving triangular systems-Part II: Two-sided and generalized Sylvester and Lyapunov matrix equations, *ACM T. Math. Software*, **28** (2002), 416–435.

11. X. Wang, Y. Li, L. Dai, On the Hermitian and skew-Hermitian splitting iteration methods for the linear matrix equation $AXB = C$, *Comput. Math. Appl.*, **65** (2013), 657–664.

12. H. M. Zhang, F. Ding, A property of the eigenvalues of the symmetric positive definite matrix and the iterative algorithm for couple Sylvester matrix equations, *J. Franklin I.*, **351** (2014), 340–357.

13. Z. Z. Bai, On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equation, *J. Comput. Math.*, **29** (2011), 185–198.

14. F. Ding, T. Chen, Gradient based iterative algorithms for solving a class of matrix equations, *IEEE T. Automat. Contr.*, **50** (2005), 1216–1221.

15. F. Ding, T. Chen, Iterative least squares solutions of coupled Sylvester matrix equations, *Syst. Control Lett.*, **54** (2005), 95–107.

16. Q. Niu, X. Wang, L. Z. Lu, A relaxed gradient based algorithm for solving Sylvester equation, *Asian J. Control*, **13** (2011), 461–464.

17. W. Fan, C. Gu, Z. Tian, Jacobi-gradient iterative algorithms for Sylvester matrix equations, In: *Linear Algebra Society Topics*, Shanghai University, Shanghai, China, 2007, 16–20.

18. S. K. Li, T. Z. Huang, A shift-splitting Jacobi-gradient algorithm for Lyapunov matrix equation arising form control theory, *J. Comput. Anal. Appl.*, **13** (2011), 1246–1257.

19. 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.

20. X. Wang, L. Dai, D. Liao, A modified gradient based algorithm for solving Sylvester equations, *Appl. Math. Comput.*, **218** (2012), 5620–5628.

21. F. Ding, P. X. Liu, T. Chen, Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle, *Appl. Math. Comput.*, **197** (2008), 41–50.

22. A. Kittisopaporn, P. Chansangiam, Gradient-descent iterative algorithm for solving a class of linear matrix equations with applications to heat and Poisson equations, *Adv. Differ. Equ.*, **2020** (2020), 324.

23. A. Kittisopaporn, P. Chansangiam, W. Lewkeeratiyukul, Convergence analysis of gradient-based iterative algorithms for a class of rectangular Sylvester matrix equation based on Banach contraction principle, *Adv. Differ. Equ.*, **2021** (2021), 17.

24. N. Sasaki, P. Chansangiam, Modified Jacobi-gradient iterative method for generalized Sylvester matrix equation, *Symmetry*, **12** (2020), 1831.

25. Y. J. Xie, C. F. Ma, Gradient based and least square based iterative algorithms for matrix equation $AXB + CX^TB = F$, *Appl. Math. Comput.*, **217** (2010), 2191–2199.

26. R. A. Horn, C. R. Johnson, *Topics in matrix analysis*, New York: Cambridge University Press, 1991.

27. E. Kreyszig, *Introductory functional analysis with applications*, New York: John Wiley & Sons, 1978.

28. L. Teck, Nonexpansive matrices with applications to solutions of linear systems by fixed point iterations, *Fixed Point Theory Appl.*, **2010** (2009), 821928.

Appendix

This appendix contains MATLAB-code of Example 5.1

```
tic;
τ = 0.1379;
Xt = X;
XtT = transpose(Xt);
Et = F - (A1*Xt*B1 + A2*Xt*B2 + A3*Xt*B3 + C1*XtT*D1 + C2*XtT*D2 + C3*XT*D3);
EtT = transpose(Et);
et(1) = e(1)/norm(F,'fro');
let(1) = le(1);
kt = 2;
while (kt < 50)
Et = F - (A1*Xt*B1 + A2*Xt*B2 + A3*Xt*B3 + C1*XtT*D1 + C2*XtT*D2 + C3*XT*D3);
EtT = transpose(Et); Xt = Xt + τ*(A1T*Et*B1T+ A2T*Et*B2T + A3T*Et*B3T + D1*EtT*C1 +
D2*EtT*C2 + D3*EtT*C3);
et(kt) = norm(Et,'fro')/norm(F,'fro');
let(kt) = log(norm(Et,'fro'));
kt = kt+1;
end
tt = toc;
```