Advances in Difference Equations
a SpringerOpen Journal

RESEARCH Open Access

Approximated least-squares solutions of a generalized Sylvester-transpose matrix equation via gradient-descent iterative algorithm

Adisorn Kittisopaporn[1] and Pattrawut Chansangiam[1*]

*Correspondence:
pattrawut.ch@kmitl.ac.th
[1]Department of Mathematics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, 10520 Bangkok, Thailand

Abstract

This paper proposes an effective gradient-descent iterative algorithm for solving a generalized Sylvester-transpose equation with rectangular matrix coefficients. The algorithm is applicable for the equation and its interesting special cases when the associated matrix has full column-rank. The main idea of the algorithm is to have a minimum error at each iteration. The algorithm produces a sequence of approximated solutions converging to either the unique solution, or the unique least-squares solution when the problem has no solution. The convergence analysis points out that the algorithm converges fast for a small condition number of the associated matrix. Numerical examples demonstrate the efficiency and effectiveness of the algorithm compared to renowned and recent iterative methods.

MSC: 15A60; 15A69; 26B25; 65F45

Keywords: Generalized Sylvester-transpose matrix equation; Gradient descent; Iterative method; Least-squares solution

1 Introduction

In differential equations and control engineering, there has been much attention for the following linear matrix equations:

$$AXB = C, \tag{1}$$

$$AX + XA^T = B: \quad \text{Lyapunov equation}, \tag{2}$$

$$AX + XB = C: \quad \text{Sylvester equation}, \tag{3}$$

$$AXB + CXD = E: \quad \text{a generalized Sylvester equation}, \tag{4}$$

$$AXB + CX^TD = E: \quad \text{a generalized Sylvester-transpose equation}, \tag{5}$$

$$X + AXB = C: \quad \text{Stein equation}, \tag{6}$$

$$X + AX^TB = C: \quad \text{Stein-transpose equation}. \tag{7}$$

Springer

These equations are special cases of a generalized Sylvester-transpose matrix equation:

$$\sum_{t=1}^{p} A_t X B_t + \sum_{s=1}^{q} C_s X^T D_s = E, \tag{8}$$

where, for each $t = 1, \dots, p$, $A_t \in \mathbb{R}^{l\times m}$, $B_t \in \mathbb{R}^{n\times r}$, for each $s = 1, \dots, q$, $C_s \in \mathbb{R}^{l\times n}$, $D_s \in \mathbb{R}^{m\times r}$, $E \in \mathbb{R}^{l\times r}$ are known matrices whereas $X \in \mathbb{R}^{m\times n}$ is the matrix to be determined. These equations play important roles in control and system theory, robust simulation, neural network, and statistics; see e.g. [1–4].

A traditional method of finding their exact solutions is to use the Kronecker product of a matrix and the vectorization to reduce the matrix equation to a linear system; see e.g. [5, Ch. 4]. However, the dimension of the linear system can be very large due to the Kronecker multiplication, so that the step of finding the inversion of the associated matrix will result in excessive computer storage memory. For that reason, iterative approaches have received much attention. The conjugate gradient (CG) is an interesting idea to formulate finite-step iterative procedures to obtain the exact solution at the final step. There are variants of CG method for solving linear matrix equations, namely, the generalized conjugate direction method (GCD) [6], the conjugated gradient least-squares method (CGLs) [7], generalized product-type methods based on a bi-conjugate gradient (GPBi) [8]. Another interesting idea to create an iterative method is to use Hermitian and skew-Hermitian splitting (HSS); see e.g. [9].

A group of methods, called gradient-based iterative methods, aim to construct a sequence of approximated solutions that converges to the exact solution for any given initial matrices. These methods are derived from the minimization of associated norm-error functions using gradients, and the hierarchical identification. Such techniques have stimulated and have played a role in many pieces of research in a few decades. In 2005, Ding and Chen [10] proposed a gradient-based iterative (GI) method for solving Eqs. (3), (4), and (6). Ding et al. [11] proposed the GI and the least-squares iterative (LSI) methods for solving $\sum_{j=1}^{p} A_j X B_j = F$ which includes Eqs. (1) and (4). Niu et al. [12] developed a relaxed gradient-based iterative (RGI) method for solving Eq. (3) by introducing a weighted factor. The MGI method, developed by Wang et al. [13], is a half-step-update modification of the GI method. Zhaolu et al. [14] presented two methods for solving Eq. (3). The first method is based on the GI method and called the Jacobi gradient iterative (JGI) method. Furthermore, they introduced relaxation factors to accelerate the speed of convergence and called the accelerated Jacobi gradient iterative (AJGI) method. Recently, Sun et al. (2019, [15]) proposed two modified least-squares iterative algorithms namely, LSIA1 [15, Theorem 2.3] and LSIA2 [15, Theorem 3.1] for the Lyapunov equation (2). See more algorithms in [16–24]. The developed iterative methods can be applied to state-space models [25], controlled autoregressive systems [26], and parameter estimation in signal processing [27].

Let us focus on gradient-based iterative methods for solving Eqs. (5) and (8). A recent gradient iterative method for Eq. (5) is AGBI method, developed in [28]. The following two methods were proposed to produce the sequence $X(k)$ of approximated solutions converging to the exact solution X^* of Eq. (8).

Method 1.1 ([29]) *Gradient iterative* (*GI*) *method.*

$$X(k) = \frac{1}{p+q}\left(\sum_{j=1}^{p} X_j(k) + \sum_{l=1}^{q} X_{p+l}(k)\right),$$

$$X_j(k) = X(k-1) + \mu A_j^T\left(E - \sum_{i=1}^{p} A_i X(k-1)B_i - \sum_{i=1}^{q} C_i X^T(k-1)D_i\right)B_j^T,$$

$$X_{p+l}(k) = X(k-1) + \mu D_l\left(E - \sum_{i=1}^{p} A_i X(k-1)B_i - \sum_{i=1}^{q} C_i X^T(k-1)D_i\right)^T C_l.$$

A conservative choice of the convergence factor μ is

$$0 < \mu < 2\left(\sum_{j=1}^{p} \lambda_{\max}\left(A_j A_j^T\right)\lambda_{\max}\left(B_j^T B_j\right) + \sum_{l=1}^{q} \lambda_{\max}\left(C_l C_l^T\right)\lambda_{\max}\left(D_l^T D_l\right)\right).$$

Method 1.2 ([29]) *Least-squares iterative* (*LSI*) *method.*

$$R(k) = E - \sum_{i=1}^{p} A_i X(k-1)B_i - \sum_{i=1}^{q} C_i X(k-1)D_i,$$

$$X_j(k) = X(k-1) + \mu\left(A_j^T A_j\right)^{-1} A_j^T R(k) B_j^T\left(B_j B_j^T\right)^{-1},$$

$$X_{p+l}(k) = X(k-1) + \mu\left(D_l D_l^T\right)^{-1} D_l R(k) C_l\left(C_l^T C_l\right)^{-1},$$

$$X(k) = \frac{1}{p+q}\left(\sum_{j=1}^{p} X_j(k) + \sum_{l=1}^{q} X_{p+l}(k)\right), \quad 0 < \mu < 2(p+q).$$

In this work, we introduce a new iterative algorithm based on gradient-descent for solving Eq. (8). The techniques of gradient and steepest descent let us obtain the search direction and the step sizes. Indeed, our varied step sizes are the optimal convergence factors that guarantee the algorithm to have a minimum error at each iteration. Our convergence analysis proves that, when Eq. (8) has a unique solution, the algorithm constructs a sequence of approximated solutions converging to the exact solution. On the other hand, when Eq. (8) has no solution, the generated sequence converges to the unique least-squares solution. We provide the convergence rate to show that the speed of convergence depends on the condition number of the associated certain matrix. In addition, we have an error analysis that gives an error estimation comparing the current iteration with the preceding and the initial iterations. Finally, we provide numerical simulations to guarantee the efficiency and effectiveness of our algorithm. The illustrative examples show that our algorithm is applicable to both Eq. (8) and its certain interesting special cases.

The organization of this paper is as follows. In Sect. 2, we recall the criterion for the matrix equation (8) to have a unique solution or a unique least-squares solution, via the Kronecker linearization. We propose the gradient-descent algorithm to solve Eq. (8) in Sect. 3. The proof of convergence criteria, convergence rates, and error estimation for the proposed algorithm are provided in Sect. 4. In Sect. 5, we present the comparison of the efficiency of our proposed algorithm to well-known and recent iterative algorithms.

In the remainder of this paper, all vectors and matrices are real. Denote the set of n columns vectors by $\mathbb{R}^n$ and the set of $m \times n$ matrices by $\mathbb{R}^{m\times n}$. The (i,j)th entry of a matrix

A is denoted by $A(i,j)$ or a_{ij}. To perform a convergence analysis, we use the Frobenius norm, the spectral norm, and the (spectral) condition number of $A \in \mathbb{R}^{m\times n}$, which are, respectively, defined by

$$\|A\|_F = \sqrt{\operatorname{tr}(A^TA)}, \qquad \|A\|_2 = \sqrt{\lambda_{\max}(A^TA)}, \qquad \kappa(A) = \left(\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}\right)^{1/2}.$$

2 Exact and least-squares solutions of the matrix equation by the Kronecker linearization

In this section, we explain how to solve the generalized Sylvester-transpose matrix equation (8) directly using the Kronecker linearization.

Recall that the Kronecker product of $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ is defined by $A \otimes B = [a_{ij}B] \in \mathbb{R}^{mp\times nq}$. The vector operator $\operatorname{Vec}(\cdot)$ turns each matrix $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ to the vector

$$\operatorname{Vec}(A) = \begin{bmatrix} a_{11}\dots a_{m1} & a_{12}\dots a_{m2} & \dots & a_{1n}\dots a_{mn} \end{bmatrix}^T \in \mathbb{R}^{mn}.$$

Lemma 2.1 (e.g. [5]) *For compatible matrices A, B, and C, we have the following properties of the Kronecker product and the vector operator.*

(i) $(A \otimes B)^T = A^T \otimes B^T$,

(ii) $\operatorname{Vec}(ABC) = (C^T \otimes A)\operatorname{Vec}(B)$.

Recall also that there is a permutation matrix $P(m,n) \in \mathbb{R}^{mn\times mn}$ such that

$$\operatorname{Vec}(X^T) = P(m,n)\operatorname{Vec}(X) \quad \text{for all } X \in \mathbb{R}^{m\times n}. \tag{9}$$

This matrix depends only on the dimensions m and n and is given by

$$P(m,n) = \sum_{i=1}^{m}\sum_{j=1}^{n} E_{ij} \otimes E_{ij}^T,$$

where E_{ij} has entry 1 in (i,j)th position and all other entries are 0.

Now, we can transform Eq. (8) to an equivalent linear system by applying the vector operator and utilizing Lemma 2.1(ii) and the property (9). Indeed, we get the linear system

$$Q\operatorname{Vec}(X) = \operatorname{Vec}(E), \tag{10}$$

where

$$Q = \sum_{t=1}^{p}\left(B_t^T \otimes A_t\right) + \sum_{s=1}^{q}\left(D_s^T \otimes C_s\right)P(m,n) \in \mathbb{R}^{rl\times mn}. \tag{11}$$

Thus Eq. (8) has a (unique) solution if and only if Eq. (10) does. We impose the assumption that Q is of full column-rank, or equivalently, Q^TQ is invertible.

If Eq. (8) has a solution, then we obtain the exact (vector) solution to be

$$\operatorname{Vec}(X^*) = \left(Q^TQ\right)^{-1}Q^T\operatorname{Vec}(E). \tag{12}$$

If Eq. (8) has no solution, then we can seek for a least-squares solution, i.e. a matrix X^* that minimizes the squared Frobenius norm $\|Q\operatorname{Vec}(X) - \operatorname{Vec}(E)\|_F^2$. The assumption on Q implies that the least-squares solution for Eq. (8) is uniquely determined by the solution of the associated normal equation, and it is also given by Eq. (12). In this case, the least-squares error is given by

$$\begin{aligned}\left\|Q\operatorname{Vec}\left(X^*\right) - \operatorname{Vec}(E)\right\|_F^2 &= \left\|\operatorname{Vec}(E)\right\|_F^2 - \operatorname{Vec}^T(E)Q\operatorname{Vec}\left(X^*\right)\\ &= \|E\|_F^2 - \operatorname{Vec}^T(E)Q\left(Q^TQ\right)^{-1}Q^T\operatorname{Vec}(E). \end{aligned} \tag{13}$$

We denote both the exact and the least-squares solutions of Eq. (8) by X^*.

3 Gradient-descent iterative solutions for the matrix equation

This section is intended to propose a new iterative algorithm for creating a sequence $\{X_k\}$ of well-approximated solutions of Eq. (8) that converges to the exact or least-squares solution X^*. This algorithm will be applicable if the matrix Q is of full column-rank, no matter Eq. (8) has a solution or not.

Our aim is to generate a sequence $\{x_k\}$, starting from an initial vector x_0, using the recurrence

$$x_{k+1} = x_k + \tau_{k+1}d_k, \quad k = 0, 1, \ldots,$$

where x_k is the kth approximation, $\tau_{k+1} > 0$ is the step size, and d_k is the search direction. To obtain the search direction, we consider the Frobenius-norm error $\|\sum_{t=1}^{p} A_tXB_t + \sum_{s=1}^{q} C_sX^TD_s - E\|_F$ which is then transformed into $\|Qx - \operatorname{Vec}(E)\|_F$ via Lemma 2.1(ii) and $x = \operatorname{Vec}(X)$. Let $f : \mathbb{R}^{mn} \to \mathbb{R}$ be the norm-error function defined by

$$f(x) := \frac{1}{2}\left\|Qx - \operatorname{Vec}(E)\right\|_F^2.$$

It is easily seen that f is convex. Hence, the gradient-descent iterative method can be shown as the following recursive equation:

$$x_{k+1} = x_k - \tau_{k+1}\nabla f(x_k).$$

To find the gradient of the function f, the following properties of the matrix trace will be used:

$$\begin{aligned}&\frac{d}{dX}\operatorname{tr}(AX) = A^T,\\ &\frac{d}{dX}\operatorname{tr}\left(XAX^TB\right) = BXA + B^TXA^T.\end{aligned}$$

By letting $\tilde{e} = \operatorname{Vec}(E)$, we compute the derivative of f as follows:

$$\begin{aligned}\nabla f(x) &= \frac{1}{2}\frac{d}{dx}\operatorname{tr}\left((Qx - \tilde{e})^T(Qx - \tilde{e})\right)\\ &= \frac{1}{2}\frac{d}{dx}\operatorname{tr}\left(Qxx^TQ^T - \tilde{e}x^TQ^T - Qx\tilde{e}^T + \tilde{e}\tilde{e}^T\right)\end{aligned}$$

$$= \frac{1}{2}\left(Q^T Qx + Q^T Qx - Q^T \tilde{e} - Q^T \tilde{e}\right)$$
$$= Q^T(Qx - \tilde{e}). \tag{14}$$

Thus, we have the new form of the iterative equation as follows:

$$x_{k+1} = x_k + \tau_{k+1} Q^T(\tilde{e} - Qx_k).$$

The above equation can be transformed into matrix form via Lemma 2.1(ii), i.e.,

$$X_{k+1} = X_k + \tau_{k+1}\left(\sum_{t=1}^{p} A_t^T R_k B_t^T + \sum_{s=1}^{q} D_s R_k^T C_s\right)$$

where $R_k = E - \sum_{t=1}^{p} A_t X_k B_t - \sum_{s=1}^{q} C_s X_k^T D_s$.

To choose a step size, we define $\phi_{k+1} : [0,\infty) \to \mathbb{R}$ by for each $k \in \mathbb{N} \cup \{0\}$,

$$\phi_{k+1}(\tau) = f(x_{k+1}) = \frac{1}{2}\|Qx_{k+1} - \tilde{e}\|_F^2$$
$$= \frac{1}{2}\left\|\tau QQ^T(\tilde{e} - Qx_k) + Qx_k - \tilde{e}\right\|_F^2.$$

We differentiate ϕ_{k+1} by using the properties of a matrix trace and obtain

$$\frac{d}{d\tau}\phi_{k+1}(\tau) = \tau_{k+1}\left\|QQ^T(\tilde{e} - Qx_k)\right\|_F^2 - \left\|Q^T(\tilde{e} - Qx_k)\right\|_F^2.$$

It is obvious that the second-order derivative of ϕ_{k+1} is $\|QQ^T(\tilde{e} - Qx_k)\|_F^2$ which is a positive constant. So when $\frac{d}{d\tau}\phi_{k+1}(\tau) = 0$, we get the minimizer of ϕ_{k+1}, i.e.

$$\tau_{k+1} = \frac{\|Q^T(\tilde{e} - Qx_k)\|_F^2}{\|QQ^T(\tilde{e} - Qx_k)\|_F^2}$$
$$= \frac{\|\operatorname{Vec}(W_k)\|_F^2}{\|\operatorname{Vec}(\sum_{t=1}^{p} A_t W_k B_t + \sum_{s=1}^{q} C_s W_k^T D_s)\|_F^2}$$
$$= \frac{\|W_k\|_F^2}{\|\sum_{t=1}^{p} A_t W_k B_t + \sum_{s=1}^{q} C_s W_k^T D_s\|_F^2}.$$

Here $W_k = \sum_{t=1}^{p} A_t^T R_k B_t^T + \sum_{s=1}^{q} C_s^T R_k D_s^T$.

An implementation of the gradient-descent iterative algorithm for solving Eq. (8) is given by the following algorithm where the search direction and the step size are taken into account. To terminate the algorithm, one can alternatively set the stopping rule to be $\|R_k\|_F - \delta < \epsilon$ where $\epsilon > 0$ is a small error and δ is the least-squares error described in Eq. (13).

4 Convergence analysis of the proposed algorithm

In this section, Algorithm 1 will be proved to converge to the exact solution or the unique least-squares solution. Recall the next lemma.

Algorithm 1: The gradient-descent iterative algorithm for Eq. (8)

A_t, B_t, C_s, D_s, E, X_0;
for $k = 1, \dots, n$ **do**
  $R_k = E - \sum_{t=1}^{p} A_t X_k B_t - \sum_{s=1}^{q} C_s X_k^T D_s$;
  $W_k = \sum_{t=1}^{p} A_t^T R_k B_t^T + \sum_{s=1}^{q} C_s^T R_k D_s^T$;
  $\tau_{k+1} = \| W_k \|_F^2 / \| \sum_{t=1}^{p} A_t W_k B_t + \sum_{s=1}^{q} C_s W_k^T D_s \|_F^2$;
  $X_{k+1} = X_k + \tau_{k+1} (\sum_{t=1}^{p} A_t^T R_k B_t^T + \sum_{s=1}^{q} D_s R_k^T C_s)$
end

Lemma 4.1 ([30]) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a strongly convex function, i.e. there exist two non-negative constants ψ, Ψ such that $\psi I \le \nabla^2 f(x) \le \Psi I$ for all $x \in \mathbb{R}^n$. Then, for any $x, y \in \mathbb{R}^n$,*

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{\psi}{2} \|y - x\|_F^2, \tag{15}$$

$$f(y) \le f(x) + \nabla f(x)^T (y - x) + \frac{\Psi}{2} \|y - x\|_F^2. \tag{16}$$

The following definition is an extension of the Frobenius norm and will be used in the convergence analysis.

Definition 4.2 Given a full-column-rank matrix $P \in \mathbb{R}^{k \times n}$, we define the P-weighted Frobenius norm of $A \in \mathbb{R}^{m \times n}$ by

$$\|A\|_P := \|PA\|_F = \sqrt{\operatorname{tr}\left(A^T P^T P A\right)}. \tag{17}$$

Theorem 4.3 *Consider Eq. (8). Assume that Q is of full column-rank.*

(i) *Suppose Eq. (8) has a solution (and thus, the solution is unique). Then, for any initial matrix X_0, the sequence X_k of approximated solutions generated by Algorithm 1 converges to the exact solution X^*.*

(ii) *Suppose Eq. (8) has no solution (and thus, it has the unique least-squares solution X^*). Then $\|X_k\|_Q \to \|X^*\|_Q$ for any initial matrix X_0. Here, $\|\cdot\|_Q$ is the Q-weighted Frobenius norm defined by Eq. (17).*

Proof Since $x^* = \mathrm{Vec}(X^*)$ is the optimal solution of $\min_{x \in \mathbb{R}^{mn}} f(x)$, we denote the minimum value, $\inf_{x \in \mathbb{R}^{mn}} f(x) = f(x^*)$ as δ. Note that δ is equal to the least-squares error determined by Eq. (13) and is zero if X^* is the unique exact solution. If there exists $k \in \mathbb{N}$ such that $\nabla f(x_k) = 0$, then $X_k = X^*$ and the result holds. To investigate the convergence of the algorithm, we assume that $\nabla f(x_k) \neq 0$ for all k. Considering the strong convexity of f, we have from Eq. (14) $\nabla^2 f(x_k) = Q^T Q$. Let $\lambda_{\min}$ ($\lambda_{\max}$) be the minimum (maximum) eigenvalue of $Q^T Q$, respectively. Since $Q^T Q$ is symmetric, we have

$$\lambda_{\min} I \le \nabla^2 f(x_k) \le \lambda_{\max} I.$$

Thus, f is strongly convex. From (15), substituting $y = x_{k+1}$ and $x = x_k$ yields

$$f(y) \ge f(x_k) - \tau \left\| \nabla f(x_k) \right\|_F^2 + \frac{\lambda_{\min} \tau^2}{2} \left\| \nabla f(x_k) \right\|_F^2.$$

We minimize the RHS by taking $\tau = 1/\lambda_{\min}$, so that

$$f(y) \geq f(x_k) - \frac{1}{2\lambda_{\min}} \left\| \nabla f(x_k) \right\|_F^2.$$

Since the above equation is true for all $y \in \mathbb{R}^{mn}$, we have

$$\delta \geq f(x_k) - \frac{1}{2\lambda_{\min}} \left\| \nabla f(x_k) \right\|_F^2. \tag{18}$$

Similarly, from (16), we have

$$f(x_{k+1}) \leq f(x_k) - \tau \left\| \nabla f(x_k) \right\|_F^2 + \frac{\lambda_{\max} \tau^2}{2} \left\| \nabla f(x_k) \right\|_F^2.$$

Minimizing the RHS by taking $\tau = 1/\lambda_{\max}$ yields

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2\lambda_{\max}} \left\| \nabla f(x_k) \right\|_F^2. \tag{19}$$

Subtracting each side of (19) by δ and combining with $\|\nabla f(x_k)\|_F^2 \geq 2\lambda_{\min}(f(x_k) - \delta)$ (from (18)), we get

$$\begin{aligned} f(x_{k+1}) - \delta &\leq f(x_k) - \delta - \frac{1}{2\lambda_{\max}} \left\| \nabla f(x_k) \right\|_F^2 \\ &\leq \big(f(x_k) - \delta\big) - \frac{2\lambda_{\min}}{2\lambda_{\max}} \big(f(x_k) - \delta\big) \\ &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right) \big(f(x_k) - \delta\big). \end{aligned}$$

Putting $\alpha := 1 - \lambda_{\min}/\lambda_{\max}$, we have

$$f(x_{k+1}) - \delta \leq \alpha \big(f(x_k) - \delta\big). \tag{20}$$

By induction, we obtain

$$f(x_k) - \delta \leq \alpha^k \big(f(x_0) - \delta\big). \tag{21}$$

Since $Q^T Q$ is assumed to be invertible, $Q^T Q > 0$, it follows that $\lambda_{\min} > 0$ and hence $0 < \alpha < 1$. Thus, $f(x_k) - \delta \to 0$, or equivalently, $f(x_k) \to \delta$ as $k \to \infty$.

Consider the case of X^* is the unique exact solution, i.e., $\delta = 0$. We have $f(x_k) \to 0$, or equivalently $Qx_k - \operatorname{Vec}(E) \to 0$ as $k \to \infty$. Now, the assumption that Q is of full column-rank implies that

$$x_k \to \big(Q^T Q\big)^{-1} Q^T \operatorname{Vec}(E).$$

Therefore, $X_k = \operatorname{Vec}^{-1}(x_k) \to X^*$ as $k \to \infty$.

The other case is that X^* is the unique least-squares solution, i.e., $\delta > 0$. We have $f(x_k) \to \delta$ or $\frac{1}{2}\|Qx_k - \operatorname{Vec}(E)\|_F^2 \to \|\operatorname{Vec}(E)\|_F^2 - \operatorname{Vec}(E)^T Qx^*$. Then

$$\frac{1}{2} \operatorname{tr}\Big(\big(Qx_k - \operatorname{Vec}(E)\big)^T \big(Qx_k - \operatorname{Vec}(E)\big)\Big) \to \operatorname{tr}\big(\operatorname{Vec}(E)^T \operatorname{Vec}(E)\big) - \operatorname{Vec}(E)^T Qx^*.$$

We omit some algebraic operations and hence immediately write

$$\|x_k\|_Q^2 = \operatorname{tr}\left(x_k^T Q^T Q x_k\right) \to \operatorname{tr}\left(\operatorname{Vec}(E)^T \operatorname{Vec}(E)\right) = \left\|x^*\right\|_Q^2.$$

Therefore, $\|X_k\|_Q \to \|X^*\|_Q$ as $k \to \infty$. □

We denote the condition number of Q by $\kappa = \kappa(Q)$. Observe that $\alpha = 1-\kappa^{-2}$. The relation between the quadratic norm-error $f(x_k)$ and the norm of residual error $\|R_k\|$ is given by

$$f(x_k) = \frac{1}{2}\|R_k\|_F^2.$$

Making use of Lemma 2.1(ii), the inequalities (20) and (21) become the following estimation:

$$\|R_k\|_F^2 \le \alpha\|R_{k-1}\|_F^2 + 2\delta\kappa^{-2}, \tag{22}$$

$$\|R_k\|_F^2 \le \alpha^k\|R_0\|_F^2 + 2\delta\left(1-\alpha^k\right). \tag{23}$$

In the case of Eq. (8) having a unique exact solution ($\delta = 0$), the error estimations (22) and (23) reduce to (24) and (25), respectively.

$$\|R_k\|_F \le \alpha^{\frac{1}{2}}\|R_{k-1}\|_F, \tag{24}$$

$$\|R_k\|_F \le \alpha^{\frac{k}{2}}\|R_0\|_F. \tag{25}$$

Since $0 < \alpha < 1$, it follows that, if $\|R_{k-1}\|_F$ are nonzero, then

$$\|R_k\|_F < \|R_{k-1}\|_F. \tag{26}$$

The above discussion is summarized in the following theorem.

Theorem 4.4 *Assume that Q is of full column-rank.*

(i) *Suppose Eq. (8) has a unique solution. The error estimation $\|R_k\|_F$ compared with $\|R_{k-1}\|_F$ (the preceding iteration) and $\|R_0\|_F$ (the initial iteration) are given by (24) and (25), respectively. Particularly, the relative error $\|R_k\|_F$ gets smaller than the preceding (nonzero) error, as in (26).*

(ii) *When Eq. (8) has a unique least-squares solution, the error estimation (22) and (23) hold.*

In both cases, the convergence rate of Algorithm 1 (regarding the error $\|R_k\|_F$) is governed by $\sqrt{1-\kappa^{-2}}$.

Remark 4.5 The relative errors (22) and (23) do not seem to decrease every step of iteration since the terms $2\delta\kappa^{-2}$ and $2\delta(1-\alpha^k)$ are positive. However, the inequality (19) implies that $\{\|R_k\|_F\}_{k=1}^{\infty}$ is a strictly decreasing sequence converging to δ.

We recall the following properties.

Lemma 4.6 (e.g. [5]) *For any compatible matrices A and B, we have*

(i) $\|A^TA\|_2 = \|A\|_2^2$,

(ii) $\|A^T\|_2 = \|A\|_2$,

(iii) $\|AB\|_F \le \|A\|_2\|B\|_F$.

Theorem 4.7 *Suppose that Q is of full column-rank and Eq.* (8) *has a unique exact solution. We have the error estimation* $\|X_k - X^*\|_F$ *compared with the preceding iteration and the initial iteration of Algorithm* 1 *are provided by*

$$\left\|X_k - X^*\right\|_F \le \kappa\sqrt{\kappa^2 - 1}\left\|X_{k-1} - X^*\right\|_F, \tag{27}$$

$$\left\|X_k - X^*\right\|_F \le \kappa^2\left(1 - \kappa^{-2}\right)^{\frac{k}{2}}\left\|X_0 - X^*\right\|_F. \tag{28}$$

Particularly, the convergence rate of the algorithm is governed by $\sqrt{1-\kappa^{-2}}$.

Proof Utilizing (25) and Lemma 4.6, we have

$$\begin{aligned}
\left\|X_k - X^*\right\|_F &= \left\|x_k - x^*\right\|_F \\
&= \left\|\left(Q^TQ\right)^{-1}\left(Q^TQ\right)x_k - \left(Q^TQ\right)^{-1}\left(Q^TQ\right)x^*\right\|_F \\
&\le \left\|\left(Q^TQ\right)^{-1}\right\|_2\left\|Q^T\right\|_2\left\|Qx_k - Qx^*\right\|_F \\
&\le \left(1-\kappa^{-2}\right)^{\frac{k}{2}}\left\|\left(Q^TQ\right)^{-1}\right\|_2\left\|Q^T\right\|_2\|Qx_0 - \tilde{e}\|_F \\
&\le \left(1-\kappa^{-2}\right)^{\frac{k}{2}}\left\|\left(Q^TQ\right)^{-1}\right\|_2\left\|Q^T\right\|_2\|Q\|_2\left\|X_0 - X^*\right\|_F \\
&= \left(1-\kappa^{-2}\right)^{\frac{k}{2}}\frac{\lambda_{\max}(Q^TQ)}{\lambda_{\min}(Q^TQ)}\left\|X_0 - X^*\right\|_F \\
&= \kappa^2\left(1-\kappa^{-2}\right)^{\frac{k}{2}}\left\|X_0 - X^*\right\|_F.
\end{aligned}$$

As the limiting behavior of $\|X_k - X^*\|_F$ depends on $(1-\kappa^{-2})^{\frac{k}{2}}$, the convergence rate for Algorithm 1 is governed by $\sqrt{1-\kappa^{-2}}$. Similarly, using (24), it follows that

$$\begin{aligned}
\left\|X_k - X^*\right\|_F &\le \left(1-\kappa^{-2}\right)^{\frac{1}{2}}\left\|\left(Q^TQ\right)^{-1}\right\|_2\left\|Q^T\right\|_2\|Qx_{k-1} - \tilde{e}\|_F \\
&\le \left(1-\kappa^{-2}\right)^{\frac{1}{2}}\left\|\left(Q^TQ\right)^{-1}\right\|_2\left\|Q^T\right\|_2\|Q\|_2\left\|X_{k-1} - X^*\right\|_F \\
&= \kappa^2\left(1-\kappa^{-2}\right)^{\frac{1}{2}}\left\|X_{k-1} - X^*\right\|_F,
\end{aligned}$$

and hence (28) is obtained. □

Theorem 4.8 *Suppose Q is of full column-rank and Eq.* (8) *has a unique least-squares solution. The error estimation* $\|X_k - X^*\|_F^2$ *compared to the preceding iteration and the initial iteration of Algorithm* 1 *are provided by*

$$\left\|X_k - X^*\right\|_F^2 \le \alpha\kappa^4\left\|X_{k-1} - X^*\right\|_F^2 + 2\delta\lambda_{\min}^{-1}, \tag{29}$$

$$\left\|X_k - X^*\right\|_F^2 \le \alpha^k\kappa^4\left\|X_0 - X^*\right\|_F^2 + 2\delta\kappa^2\left(1-\alpha^k\right)\lambda_{\min}^{-1}. \tag{30}$$

Proof The proof is similar to that of Theorem 4.7 and carried out by (22) and (23). We, therefore, omit the proof. □

Consequently, our convergence analysis indicates that the proposed algorithm always converges to the unique (exact or least-squares) solution for any initial matrices and small condition numbers. Moreover, the algorithm will converge fast when the condition number is close to 1.

5 Numerical experiments for the generalized Sylvester-transpose matrix equation and its special cases

In this section, we provide numerical results to show the efficiency and effectiveness of Algorithm 1. We perform the experiments in the following cases:

- a large-scaled square generalized Sylvester-transpose equation,
- a small-scaled rectangular generalized Sylvester-transpose equation,
- a small-scaled square Sylvester-transpose equation,
- a large-scaled square Sylvester equation,
- a moderate-scaled square Lyapunov equation.

Each example contains some comparisons of the proposed algorithm (denoted by *TauOpt*) with the mentioned existing algorithms as well as the direct method Eq. (12). CT stands for the computational time (in seconds) and is measured by the *tic toc* function in MATLAB. The relative error $\|R_k\|_F$ is used to measure error at the kth step of the iteration. All iterations have been evaluated by MATLAB R2020b, on a PC (2.60-GHz intel(R) Core(TM) i7 processor, 8 Gbyte RAM).

Example 5.1 Consider a generalized Sylvester-transpose matrix equation

$$\sum_{t=1}^{2} A_t X B_t + \sum_{s=1}^{3} C_s X^T D_s = E$$

with 100×100 coefficient matrices:

$$\begin{aligned}
&A_1 = \text{tridiag}(-0.242, 0.217, 0.109), && A_2 = \text{tridiag}(0.539, 0.253, -0.835),\\
&B_1 = \text{tridiag}(0.098, -0.793, 0.561), && B_2 = \text{tridiag}(0.001, 0.533, 0.212),\\
&C_1 = \text{tridiag}(0.586, 0.462, -0.688), && C_2 = \text{tridiag}(-0.245, -0.937, 0.687),\\
&C_3 = \text{tridiag}(-0.930, 0.471, -0.813), && D_1 = \text{tridiag}(0.440, -0.762, 0.008),\\
&D_2 = \text{tridiag}(0.995, 0.075, 0.169), && D_3 = \text{tridiag}(0.514, -0.779, 0.358),\\
&\text{and} \quad E = \text{septdiag}(-0.427, -0.158, -1.181, 1.182, -0.452, -0.014, -0.158).
\end{aligned}$$

We choose an initial matrix $X_0 = \text{zero}(100)$, where $\text{zero}(n)$ is the $n \times n$ zero matrix. In fact, this equation has the unique solution

$$X^* = \text{tridiag}(0.293, 0.152, 0.905).$$

Table 1 shows that the direct method consumes a big amount of time to get the exact solution, while Algorithm 1 produces a small-error solution in a small time (0.1726 sec-

Table 1 Relative errors and CTs for Ex. 5.1

Method	Iterations	Relative error	CT
TauOpt	100	0.0014	0.1726
GI	100	4.0260	0.3252
LSI	100	17.8265	2.0425
direct	–	0	5.4274e+03

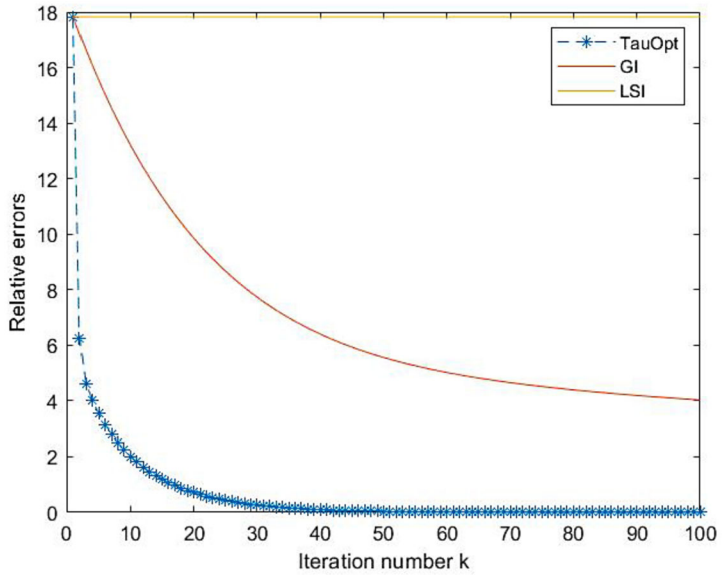

Figure 1 Relative errors for Ex. 5.1

onds after 100 iterations). We compare the efficiency of Algorithm 1 with another existing gradient-based iterative algorithms, namely, GI (Method 1.1) and LSI (Method 1.2). Figure 1 displays the error plot which supports the theoretical results i.e., the sequence of errors generated by Algorithm 1 is monotone decreasing. Table 1 indicates that our algorithm performs well in computational time.

Example 5.2 Consider the equation

$$\sum_{t=1}^{3} A_t X B_t + \sum_{s=1}^{2} C_s X^T D_s = E$$

with the rectangular coefficient matrices as follows:

$$A_1 = \begin{bmatrix} 0.491 & 0.064 \\ 0.071 & 0.436 \\ 0.887 & 0.826 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0.394 & 0.886 \\ 0.613 & 0.931 \\ 0.818 & 0.190 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0.258 & 0.503 \\ 0.897 & 0.612 \\ 0.593 & 0.819 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0.531 & 0.453 & 0.966 \\ 0.202 & 0.427 & 0.620 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.695 & 0.346 & 0.556 \\ 0.720 & 0.517 & 0.156 \end{bmatrix},$$

Table 2 Errors and CTs for Ex. 5.2

Method	Iterations	Error	CT
TauOpt	100	7.3178e–04	0.0015
GI	100	0.6164	0.0025
LSI	100	0.8453	0.0076
direct	–	0	0.0020

$$B_3 = \begin{bmatrix} 0.562 & 0.426 & 0.731 \\ 0.694 & 0.836 & 0.360 \end{bmatrix}, \qquad C_1 = \begin{bmatrix} 0.454 & 0.734 \\ 0.386 & 0.430 \\ 0.775 & 0.693 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} 0.945 & 0.109 \\ 0.784 & 0.389 \\ 0.705 & 0.590 \end{bmatrix}, \qquad D_1 = \begin{bmatrix} 0.459 & 0.228 & 0.015 \\ 0.050 & 0.834 & 0.863 \end{bmatrix},$$

$$D_2 = \begin{bmatrix} 0.078 & 0.500 & 0.571 \\ 0.669 & 0.218 & 0.122 \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} 0.671 & 0.056 & 0.435 \\ 0.599 & 0.152 & 0.832 \\ 0.056 & 0.019 & 0.617 \end{bmatrix}.$$

We find that $4 = \operatorname{rank} Q \neq \operatorname{rank}[Q \ \operatorname{Vec}(E)] = 5$, i.e., the matrix equation does not have an exact solution. However, the size of Q is 9×4, i.e., Q is of full-column rank. Hence, according to Theorem 4.3, Algorithm 1 will converge to the least-squares solution in which the least-squares error (13) is equal to 0.0231. We choose an initial matrix $X_0 = \operatorname{zero}(2)$. Algorithm 1 is compared with GI (Method 1.1), LSI (Method 1.2) and the direct method Eq. (12). In this case, we consider the error $\|X^* - X_k\|_F$ where X^* is the least-squares solution. Figure 2 displays the error plot, and Table 2 shows the errors and CTs for TauOpt, GI, LSI and the direct method. We see that the errors converge monotonically to zero, i.e., the approximate solutions X_k generated by Algorithm 1 converge to X^*. Moreover, Algorithm 1 consumes less computational time than other methods.

Next, we will consider the Sylvester-transpose equation (5) which is a special case of the generalized Sylvester-transpose equation (8). From Algorithm 1, the optimal step size τ is described by

$$\tau_{k+1} = \frac{\|W_k\|_F^2}{\|AW_kB + CW_k^TD\|_F^2},$$

where $W_k = A^TR_kB^T + C^TR_kD^T$ and $R_k = E - AX_kB - CX_k^TD$.

Example 5.3 Let us consider the Sylvester-transpose equation (5) with

$$A = \begin{bmatrix} 6 & -4 & -7 & -8 \\ 9 & -4 & 5 & 2 \\ -9 & 6 & -5 & 4 \\ 8 & -3 & 3 & 9 \end{bmatrix}, \qquad B = \begin{bmatrix} 6 & -5 & 4 & -2 \\ 9 & -7 & -5 & 6 \\ 6 & 2 & -8 & 2 \\ 7 & 3 & -1 & -1 \end{bmatrix},$$

$$C = \begin{bmatrix} -8 & -5 & -4 & 7 \\ 2 & 7 & -4 & 6 \\ 4 & 8 & -9 & -7 \\ 3 & 1 & 5 & 6 \end{bmatrix}, \qquad D = \begin{bmatrix} 3 & -5 & 1 & 2 \\ 6 & 6 & 3 & 1 \\ 4 & -8 & -5 & 4 \\ 3 & -5 & -1 & 9 \end{bmatrix} \quad \text{and}$$

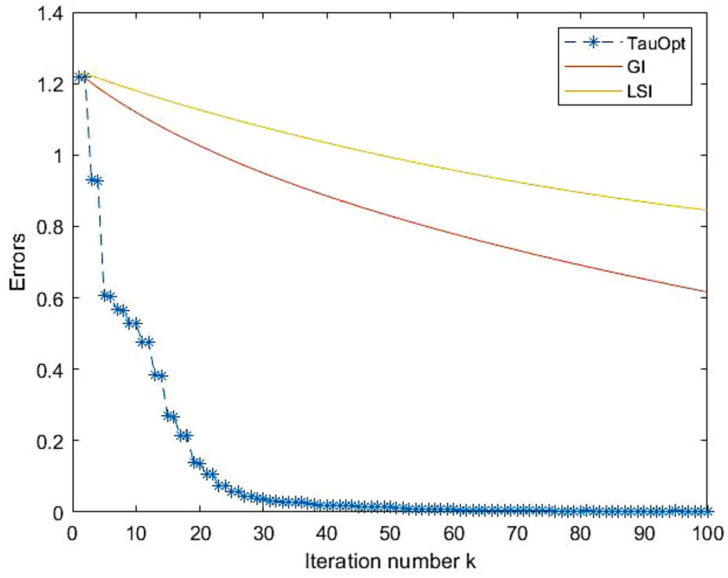

Figure 2 Errors for Ex. 5.2

Table 3 Relative errors and CTs for Ex. 5.3

Method	Iterations	Relative error	CT
TauOpt	100	0.3368	0.0011
GI	100	302.3879	0.0021
LSI	100	562.8838	0.0034
AGBI	100	80.7919	0.0015
direct	–	0	0.0051

$$E = \begin{bmatrix} -284 & 13 & 74 & -93 \\ 248 & -47 & -103 & 109 \\ -54 & 92 & 85 & -112 \\ 326 & -98 & -127 & 167 \end{bmatrix}.$$

Choosing $X_0 = \text{zero}(4)$, then the sequence of numerical solutions generated by Algorithm 1 converges to the exact solution,

$$X^* = \begin{bmatrix} 0.3342 & 0.3443 & 0.4843 & 0.7574 \\ 0.9568 & 0.7485 & 0.4250 & 0.2941 \\ 0.0177 & 0.8061 & 0.6380 & 0.6972 \\ 0.4516 & 0.1859 & 0.7069 & 0.6669 \end{bmatrix}.$$

We report the comparison of Algorithm 1 with GI (Method 1.1), LSI (Method 1.2), AGBI ([28]) and the direct method Eq. (12) by Fig. 3 and Table 3. Both of them imply that Algorithm 1 outperforms other algorithms.

Next, we will consider the Sylvester equation (3) which is also a special case of Eq. (8). For this equation, the optimal step size τ is described by

$$\tau_{k+1} = \frac{\|W_k\|_F^2}{\|AW_k + W_kB\|_F^2},$$

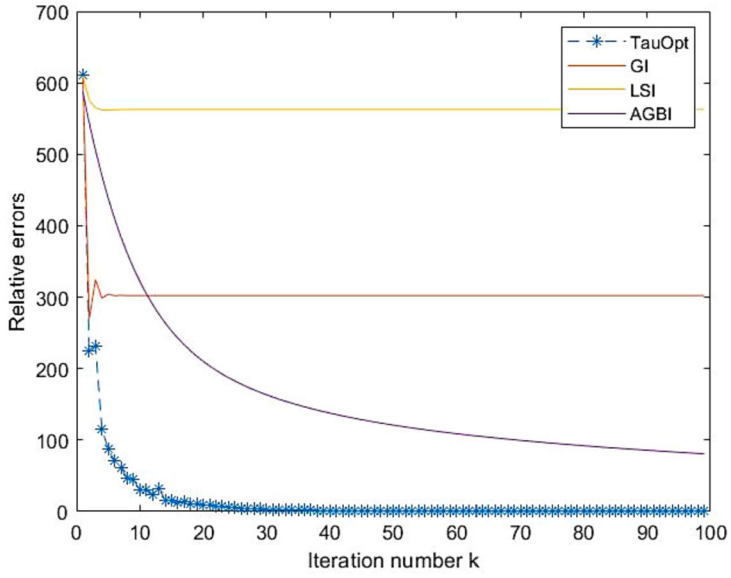

Figure 3 Relative errors for Ex. 5.3

Table 4 Relative errors and CTs for Ex. 5.4

Method	Iterations	Relative error	CT
TauOpt	100	0.1457	0.0681
GI	100	183.4122	0.0731
RGI	100	33.0116	0.0661
MGI	100	115.7206	0.0640
AGBI	100	57.8981	0.0839
JGI	100	515.9767	0.0385
AJGI	100	469.0704	0.0547
direct	–	0	5.4606e+03

where $W_k = A^T R_k + R_k B^T$ and $R_k = C - AX_k - X_k B$.

Example 5.4 Suppose that the Sylvester equation (3) has large-scaled tridiagonal coefficient matrices, i.e.,

$$A = \text{tridiag}(10, -2, 9), \qquad B = \text{tridiag}(-1, 2, -5), \quad \text{and} \quad C = \text{tridiag}(-45, 13, -20),$$

where $A, B, C \in \mathbb{R}^{100\times 100}$. We choose an initial matrix $X_0 = \text{zero}(100)$. Here, the symmetric exact solution is given by $X^* = \text{tridiag}(1, -5, 1)$, so that AGBI algorithm can be applicable. We compare Algorithm 1 with GI (Method 1.1), AGBI ([28]), RGI [12], MGI [13], JGI [14], and AJGI [14]. Although Table 4 tells us that our algorithm takes a slightly more time than some other algorithms, Fig. 4 illustrates that Algorithm 1 reaches the fastest convergence.

The last example presents another special case of Eq. (8) that is the Lyapunov equation (2). The optimal step size τ is described by

$$\tau_{k+1} = \frac{\|W_k\|_F^2}{\|AW_k + W_k A^T\|_F^2},$$

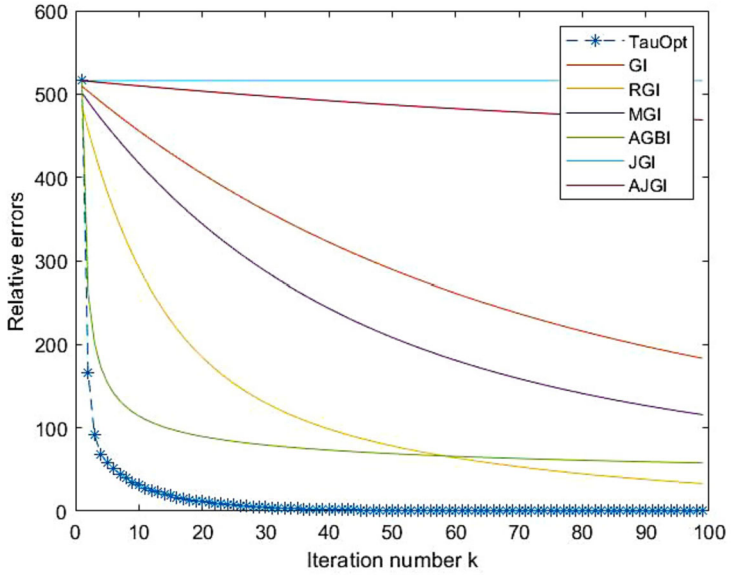

Figure 4 Relative errors for Ex. 5.4

Table 5 Relative errors and CTs for Ex. 5.5

Method	Iterations	Relative error	CT
TauOpt	50	2.4121e–06	0.0056
GI	50	8.9274	0.0056
RGI	50	6.8604	0.0063
MGI	50	2.5820	0.0061
AGBI	50	10.0789	0.0072
JGI	50	8.3219	0.0056
AJGI	50	0.5565	0.0058
LSIA 1	50	5.1063	0.0080
LSIA 2	50	5.1708	0.0041
direct	–	0	0.4636

where $W_k = A^T R_k + R_k A$ and $R_k = B - AX_k - X_k A^T$.

Example 5.5 We consider the Lyapunov equation (2) with medium-scale coefficient matrices

$$A = -\operatorname{triu}\big(\operatorname{rand}(n), 1\big) + \operatorname{diag}\big(8 - \operatorname{diag}\big(\operatorname{rand}(n)\big)\big), \qquad B = \operatorname{rand}(n).$$

We choose $n = 20$ and set $X_0 = \operatorname{zero}(20)$. Algorithm 1 is compared with GI, RGI, MGI, AGBI, JGI, AJGI, LSIA1, and LSIA2 methods. We report the results in Fig. 5 and Table 5. In conclusion, Algorithm 1 takes a slightly more computational time than some other algorithms but still outperforms distinctly in performance of convergence.

6 Concluding remarks

We properly establish a gradient-descent iterative algorithm for solving the generalized Sylvester-transpose matrix equation (8). We show that the proposed algorithm is useful and applicable for wide range of problems, even though the problem has no solution, as

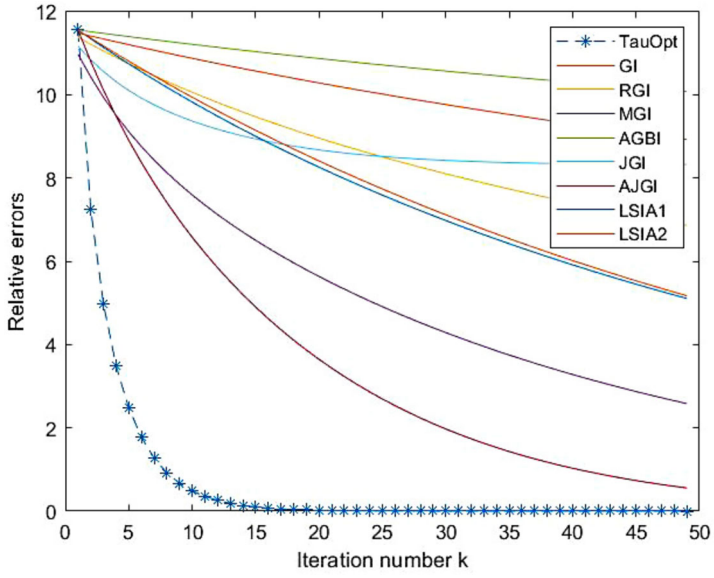

Figure 5 Relative errors for Ex. 5.5

long as the associated matrix Q, defined by Eq. (11), is of full column-rank. If the problem has the unique exact solution, then the approximate solutions converge to the exact solution. In the case of a no-solution problem, we have $\|X\|_Q \to \|X^*\|_Q$ where X^* is the unique least-squares solution. The convergence rate is described in terms of κ, the matrix condition number of Q, that is, $\sqrt{1-\kappa^{-2}}$. Moreover, the analysis shows that the sequence of errors generated by our algorithm is monotone decreasing. Numerical examples are provided to verify our theoretical findings.

Acknowledgements
The first author received financial support from KMITL Doctoral Scholarships, grant no. KDS 2019/022 during his Ph.D. study.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
Writing–original draft preparation, A.K.; writing–review and editing, P.C.; data curation, A.K.; supervision, P.C. All authors contributed equally and significantly in writing this article. All authors have read and approved the manuscript.

Publisher's Note

Received: 30 January 2021 Accepted: 17 May 2021 Published online: 21 May 2021

References

1. Geir, E.D., Fernando, P.: A Course in Robust Control Theory: A Convex Approach. Springer, New York (1999)
2. Varga, A.: Robust pole assignment via Sylvester equation based state feedback parametrization. In: Proceedings of the 2000 IEEE International Symposium on Computer-Aided Control System, pp. 13–18. Design, Alsaka (2000)
3. Magnus, J.R., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometries, 3rd edn. Wiley, Chichester (2007)

4. Nouri, K., Beik, S.P.A., et al.: An iterative algorithm for robust simulation of the Sylvester matrix differential equations. Adv. Differ. Equ. **2020**(1), Article ID 287 (2020). https://doi.org/10.1186/s13662-020-02757-z
5. Horn, R., Johnson, C.: Topics in Matrix Analysis. Cambridge University Press, New York (1991)
6. Hajarian, M.: Generalized conjugate direction algorithm for solving the general coupled matrix equations over symmetric matrices. Numer. Algorithms **73**(3), 591–609 (2016)
7. Hajarian, M.: Extending the CGLS algorithm for least squares solutions of the generalized Sylvester-transpose matrix equations. J. Franklin Inst. **353**(5), 1168–1185 (2016)
8. Dehghan, M., Mohammadi–Arani, R.: Generalized product-type methods based on Bi-conjugate gradient(GPBiCG) for solving shifted linear systems. Comput. Appl. Math. **36**(4), 1591–1606 (2017)
9. Bai, Z.: On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equation. J. Comput. Math. **29**(2), 185–198 (2011). https://doi.org/10.4208/jcm.1009-m3152
10. Ding, F., Chen, T.: Gradient based iterative algorithms for solving a class of matrix equations. IEEE Trans. Autom. Control **50**(8), 1216–1221 (2005). https://doi.org/10.1109/TAC.2005.852558
11. Ding, F., Liu, X.P., Ding, J.: Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle. Appl. Math. Comput. **197**(1), 41–50 (2008). https://doi.org/10.1016/j.amc.2007.07.040
12. Niu, Q., Wang, X., Lu, L.Z.: A relaxed gradient based algorithm for solving Sylvester equation. Asian J. Control **13**(3), 461–464 (2011). https://doi.org/10.1002/asjc.328
13. Wang, X., Dai, L., Liao, D.: A modified gradient based algorithm for solving Sylvester equation. Appl. Math. Comput. **218**(9), 5620–5628 (2012). https://doi.org/10.1016/j.amc.2011.11.055
14. Tian, Z., Tian, M., et al.: An accelerated Jacobi-gradient based iterative algorithm for solving Sylvester matrix equations. Filomat **31**(8), 2381–2390 (2017). https://doi.org/10.2298/FIL1708381T
15. Sun, M., Wang, Y., Liu, J.: Two modified least-squares iterative algorithms for the Lyapunov matrix equations. Adv. Differ. Equ. **2019**, 305 (2019). https://doi.org/10.1186/s13662-019-2253-7
16. Ding, F., Chen, T.: Hierarchical gradient-based identification of multivariable discrete-time systems. Automatica **41**(2), 315–325 (2005). https://doi.org/10.1016/j.automatica.2004.10.010
17. Ding, F., Chen, T.: Hierarchical least squares identification methods for multivariable systems. IEEE Trans. Autom. Control **50**(3), 397–402 (2005). https://doi.org/10.1109/TAC.2005.843856
18. Wu, A., Duan, G., Zhou, B.: Solution to generalized Sylvester matrix equations. IEEE Trans. Autom. Control **53**(3), 811–815 (2008). https://doi.org/10.1109/TAC.2008.919562
19. Xie, L., Liu, Y., Yang, H.: Gradient based and least squares based iterative algorithms for matrix equations $AXB + CX^TD = F$. Appl. Math. Comput. **217**(5), 2191–2199 (2010). https://doi.org/10.1016/j.amc.2010.07.019
20. Zhang, X., Sheng, X.: The relaxed gradient based iterative algorithm for the symmetric (skew symmetric) solution of the Sylvester equation $AX + XB = C$. Math. Probl. Eng. **2017**, 1–8 (2017). https://doi.org/10.1155/2017/1624969
21. Kittisopaporn, A., Chansangiam, P.: The steepest descent of gradient-based iterative method for solving rectangular linear systems with an application to Poisson's equation. Adv. Differ. Equ. **2020**(1), Article ID 259 (2020). https://doi.org/10.1186/s13662-020-02715-9
22. Boonruangkan, N., Chansangiam, P.: Gradient iterative method with optimal convergent factor for solving a generalized Sylvester matrix equation with applications to diffusion equations. Symmetry **12**(10), Article ID 1732 (2020). https://doi.org/10.3390/sym12101732
23. Sasaki, N., Chansangiam, P.: Modified Jacobi–gradient iterative method for generalized Sylvester matrix equation. Symmetry **12**(11), Article ID 1831 (2020). https://doi.org/10.3390/sym12111831
24. Kittisopaporn, A., Chansangiam, P., Lewkeeratiyutkul, W.: Convergence analysis of gradient–based iterative algorithms for a class of rectangular Sylvester matrix equations based on Banach contraction principle. Adv. Differ. Equ. **2021**(1), Article ID 17 (2021). https://doi.org/10.1186/s13662-020-03185-9
25. Ding, F., Zhang, X., Xu, L.: The innovation algorithms for multivariable state-space models. Int. J. Adapt. Control Signal Process. **33**, 1601–1618 (2019). https://doi.org/10.1002/acs.3053
26. Ding, F., Lv, L., Pan, J., et al.: Two-stage gradient-based iterative estimation methods for controlled autoregressive systems using the measurement data. Int. J. Control. Autom. Syst. **18**, 886–896 (2020). https://doi.org/10.1007/s12555-019-0140-3
27. Ding, F., Xu, L., Meng, D., et al.: Gradient estimation algorithms for the parameter identification of bilinear systems using the auxiliary model. J. Comput. Appl. Math. **369**, 112575 (2020). https://doi.org/10.1016/j.cam.2019.112575
28. Xie, Y.-J., Ma, C.-F.: The accelerated gradient based iterative algorithm for solving a class of generalized Sylvester-transpose matrix equation. Appl. Math. Comput. **273**, 1257–1269 (2016). https://doi.org/10.1016/j.amc.2015.07.022
29. Xie, L., Ding, J., Ding, F.: Gradient based iterative solutions for general linear matrix equations. Comput. Math. Appl. **58**(7), 1441–1448 (2009). https://doi.org/10.1016/j.camwa.2009.06.047
30. Stephen, P.B., Lieven, V.: Convex Optimization. Cambridge University Press, Cambridge (2004)