

RESEARCH

Open Access



Convergence analysis of gradient-based iterative algorithms for a class of rectangular Sylvester matrix equations based on Banach contraction principle

Adisorn Kittisopaporn¹, Patrawut Chansangiam^{1*}  and Wicharn Lewkeeratiyutkul²

*Correspondence:

patrawut.ch@kmitl.ac.th

¹Department of Mathematics,
Faculty of Science, King Mongkut's
Institute of Technology Ladkrabang,
10520, Bangkok, Thailand
Full list of author information is
available at the end of the article

Abstract

We derive an iterative procedure for solving a generalized Sylvester matrix equation $AXB + CXD = E$, where A, B, C, D, E are conforming rectangular matrices. Our algorithm is based on gradients and hierarchical identification principle. We convert the matrix iteration process to a first-order linear difference vector equation with matrix coefficient. The Banach contraction principle reveals that the sequence of approximated solutions converges to the exact solution for any initial matrix if and only if the convergence factor belongs to an open interval. The contraction principle also gives the convergence rate and the error analysis, governed by the spectral radius of the associated iteration matrix. We obtain the fastest convergence factor so that the spectral radius of the iteration matrix is minimized. In particular, we obtain iterative algorithms for the matrix equation $AXB = C$, the Sylvester equation, and the Kalman–Yakubovich equation. We give numerical experiments of the proposed algorithm to illustrate its applicability, effectiveness, and efficiency.

MSC: 15A60; 15A69; 47H09; 47H10; 65F45

Keywords: Generalized Sylvester matrix equation; Gradient; Linear difference vector equation; Banach contraction principle; Kronecker product; Matrix norms

1 Introduction

It is well known that linear matrix equations play crucial roles in control theory and related areas. Indeed, certain problems concerning analysis and design of control systems (e.g., existence of solutions or controllability/observability of the system) are converted to properties of associated matrix equations; see, for example, [1, Chs. 12–13] and [2]. Such matrix equations are particular cases of or closely related to the generalized Sylvester matrix equation

$$AXB + CXD = E, \tag{1}$$

© The Author(s) 2021. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

where A, B, C, D, E are given matrices, and X is an unknown matrix. This equation includes the equation $AXB = E$, the Lyapunov equation $AX + XA^T = E$, the (continuous-time) Sylvester equation $AX + XB = E$, and the Kalman–Yakubovich equation or the discrete-time Sylvester equation $AXB + X = E$. The generalized Sylvester equation naturally arises in robust control, singular system control, neural network, and statistics; see, for example, [3, 4]. An important particular case of (1), the Sylvester equation, also has applications to image restoration and numerical methods for implicit ordinary differential equations; see, for example, [5, 6].

Let us discuss how to solve (1) via the direct Kronecker linearization. We can convert the matrix equation (1) into a vector–matrix equation by taking the vector operator $\text{vec}(\cdot)$ so that (1) is reduced to $Px = b$, where

$$P = B^T \otimes A + D^T \otimes C, \quad x = \text{vec } X, \quad b = \text{vec } E.$$

Here the symbol \otimes denotes the Kronecker product. Thus Eq. (1) has a unique solution if and only if P is nonsingular. However, if the dimensions of matrices are large, then it will lead to computational difficulty, and so this approach is only applicable for small-dimensional matrices. For more information about analytical methods for solving such linear matrix equations, see, for example, [1, Ch. 12], [7, Ch. 4], and [8, Sect. 7.1]. Another technique is transforming the coefficient matrix into a Schur or Hessenberg form via an orthogonal transformation; see [9, 10].

For large matrix systems, iterative methods for solving matrix equations have received much attention. There are several ideas to formulate an iterative procedure for solving Eq. (1) and particular cases, for example, block successive overrelaxation [11], matrix sign function [12], block recursion [13, 14], Krylov subspace [15, 16], and truncated low-rank methods [17]. A group of iterative methods, called Hermitian and skew-Hermitian splitting (HSS) methods, relies on the fact that every square complex matrix can be written as the sum of its Hermitian and skew-Hermitian parts. Recently, there are several variants of HSS, namely, the generalized modified HSS (GMHSS) method [18], the accelerated double-step scale splitting (ADSS) method [19], the preconditioned HSS (PHSS) method [20], and the four-parameter positive skew-Hermitian splitting (FPPSS) method [21]. The idea of conjugate gradient (CG) also leads to several finite-step procedures to obtain the exact solution of the linear matrix equations. The principle of CG is constructing an orthogonal basis from the gradient of the associated quadratic function, consisting of vectors in the direction that approaches the fastest the exact solution. There are several variants of CG to solve such linear matrix equations, for example, the generalized conjugate direction (GCD) method [22], the conjugate gradient least-squares (CGLS) method [23], and generalized product-type methods based on biconjugate gradient (GPBiCG) method [24]. See more information in a survey [8] and references therein.

A group of gradient-based iterative algorithms relies on the ideas of hierarchical identification principle and minimization of quadratic norm-error functions; see, for example, [25–31]. Convergence analysis for such algorithms depends on the Frobenius norm $\|\cdot\|_F$, the spectral norm $\|\cdot\|_2$, the spectral radius $\rho(\cdot)$, and the condition number $\kappa(\cdot)$ of the associated iteration matrix. Let us focus on the following iterative algorithms to approximate the unique solution of Eq. (1) when all A, B, C, D are square matrices.

Algorithm 1.1 ([32]) *The gradient iterative algorithm (GI) for (1):*

$$\begin{aligned} X_1(k) &= X(k-1) + \tau A^T [E - AX(k-1)B - CX(k-1)D] B^T, \\ X_2(k) &= X(k-1) + \tau C^T [E - AX(k-1)B - CX(k-1)D] D^T, \\ X(k) &= \frac{1}{2} (X_1(k) + X_2(k)). \end{aligned}$$

If we choose the convergence factor τ such that

$$\tau = (\|A\|_2^2 \|B\|_2^2 + \|C\|_2^2 \|D\|_2^2)^{-1} \quad \text{or} \quad \tau = (\|A\|_F^2 \|B\|_F^2 + \|C\|_F^2 \|D\|_F^2)^{-1}, \tag{2}$$

then $X(k)$ converges to the exact solution for any initial values $X_1(0)$ and $X_2(0)$. Numerical simulations in [25] reveal that Algorithm 1.1 is more efficient than the B-Q algorithm [12]. In [33], Algorithm 1.1 was shown to be applicable if

$$0 < \tau < \frac{2}{\|A\|_2^2 \|B\|_2^2 + \|C\|_2^2 \|D\|_2^2}, \tag{3}$$

so that the range of τ is wider than that of (2).

Algorithm 1.2 ([33]) *The least-squares iterative algorithm (LSI) for (1):*

$$\begin{aligned} X_1(k) &= X(k-1) + \mu (A^T A)^{-1} A^T (E - AX(k-1)B - CX(k-1)D) B^T (BB^T)^{-1}, \\ X_2(k) &= X(k-1) + \mu (C^T C)^{-1} C^T (E - AX(k-1)B - CX(k-1)D) D^T (DD^T)^{-1}, \\ X(k) &= \frac{1}{2} (X_1(k) + X_2(k)). \end{aligned}$$

To make Algorithm 1.2 applicable for any initial values $X_1(0)$ and $X_2(0)$, the convergence factor μ must satisfy $0 < \mu < 4$.

There are many iterative algorithms for the Sylvester equation. The first solver is the gradient iterative algorithm (GI), introduced by Ding and Chen [32]. Niu et al. [34] introduced the relaxed gradient iterative algorithm (RGI), that is, GI with relaxation parameter $\omega \in (0, 1)$. Wang et al. [35] modified the GI algorithm in such a way that the information $X_1(k)$ has been fully considered to update $X(k-1)$; the result is called the MGI algorithm. Recently, Tian et al. introduced JGI [36, Algorithm 4], AJGI1 [36, Algorithm 5], and AJGI2 [36, Algorithm 6] algorithms based on GI and the idea of extracting the diagonal part from each coefficient matrix. Moreover, there are other iterative methods for solving the generalized coupled Sylvester matrix equations; see, for example, [37, 38].

In this paper, we introduce an iterative method for solving the generalized Sylvester equation (1), for which matrices A, B, C, D, E are not necessarily square. Our algorithm is based on gradients and hierarchical identification principle. This algorithm consists of only one parameter, the convergence factor θ , and the only initial value. To perform a convergence analysis of the algorithm, we use analysis on a complete metric space, together with matrix analysis. We convert the matrix iteration process to a first-order linear difference vector equation with matrix coefficient. Then we apply the Banach contraction principle to show that the proposed algorithm converges to the unique solution for any

initial value if and only if $0 < \theta < 2/\|P\|_2^2$. The range of the parameter is wider than those of [25, 33]. The convergence rate of the proposed algorithm is governed by the spectral radius of the associated iteration matrix. We also discuss error estimates; in particular, the error at each iteration becomes smaller than the previous one. The fastest convergence factor is determined so that the spectral radius of the iteration matrix is minimal. Moreover, we make convergence analysis of gradient-based iterative algorithms for the equation $AXB = C$, the Sylvester equation, and the Kalman–Yakubovich equation. We also provide numerical simulations to illustrate our results for the matrix equation (1) and the Sylvester equation. We compare the efficiency of our algorithm with the direct Kronecker linearization and recent algorithms, namely, GI, LSI, RGI, MGI, JGI, AJGI1, and AJGI2 algorithms.

The rest of the paper is organized as follows. We derive a gradient-based iterative algorithm in Sect. 2. We then analyze the convergence of the algorithm in Sect. 3. Iterative algorithms for particular cases of (1) are investigated in Sect. 4. We illustrate and discuss numerical simulations of the algorithm in Sect. 5. Finally, we conclude the work in Sect. 6.

2 Deriving a gradient-based iterative algorithm for the generalized Sylvester matrix equation

We denote by $\mathbb{M}_{m,n}$ the set of $m \times n$ real matrices and set $\mathbb{M}_n := \mathbb{M}_{n,n}$. In this section, we derive an iterative algorithm based on gradients to find a matrix $X \in \mathbb{M}_{n,p}$ satisfying

$$AXB + CXD = E. \tag{4}$$

Here we are given $A, C \in \mathbb{M}_{m,n}$, $B, D \in \mathbb{M}_{p,q}$, and $E \in \mathbb{M}_{m,q}$, where $m, n, p, q \in \mathbb{N}$ are such that $mq = np$.

Recall that equation (4) has a unique solution if and only if the square matrix $P = B^T \otimes A + D^T \otimes C$ is invertible. In this case, the (vector) solution is given by $\text{vec } X = P^{-1} \text{vec } E$.

To derive an iterative procedure for solving (4), we recall the hierarchical identification principle from [33]. Define two matrices

$$M := E - CXD \quad \text{and} \quad N := E - AXB.$$

In view of (4), we would like to solve two subsystems

$$AXB = M \quad \text{and} \quad CXD = N. \tag{5}$$

We will minimize the following quadratic norm-error functions:

$$L_1(X) := \|AXB - M\|_F^2 \quad \text{and} \quad L_2(X) := \|CXD - N\|_F^2. \tag{6}$$

Now we deduce their gradients as follows:

$$\begin{aligned} \frac{\partial}{\partial X} L_1(X) &= \frac{\partial}{\partial X} \text{tr}[(AXB - M)^T(AXB - M)] \\ &= \frac{\partial}{\partial X} \text{tr}(XBB^T X^T A^T A) - \frac{\partial}{\partial X} \text{tr}(X^T A^T M B^T) - \frac{\partial}{\partial X} \text{tr}(B M^T A X) \\ &= (A^T A)^T X (B B^T) + A^T A X B B^T - A^T M B^T - (B M^T A)^T \end{aligned}$$

$$= 2A^T(AXB - M)B^T. \tag{7}$$

Similarly, we have

$$\frac{\partial}{\partial X}L_2(X) = 2C^T(CXD - N)D^T. \tag{8}$$

Let $X_1(k)$ and $X_2(k)$ be the estimates or iterative solutions of system (5) at iteration k . We introduce a step-size parameter $\tau \in \mathbb{R}$ and a relaxation parameter $\omega \in (0, 1)$. We can derive recursive formulas for $X_1(k)$ and $X_2(k)$ from the gradient formulas (7) and (8) as follows:

$$\begin{aligned} X_1(k) &= X(k - 1) + \tau(1 - \omega)A^T[M - AX(k - 1)B]B^T \\ &= X(k - 1) + \tau(1 - \omega)A^T[E - AX(k - 1)B - CXD]B^T, \\ X_2(k) &= X(k - 1) + \tau\omega C^T[N - CX(k - 1)D]D^T \\ &= X(k - 1) + \tau\omega C^T[E - AXB - CX(k - 1)D]D^T. \end{aligned}$$

By the hierarchical identification principle the unknown variable X is replaced by its estimates at iteration $k - 1$. Instead of taking the arithmetic mean of $X_1(k)$ and $X_2(k)$ as in Algorithm 1.1, our algorithm computes the weighted arithmetic mean $\omega X_1(k) + (1 - \omega)X_2(k)$. By introducing the parameter $\theta = \tau\omega(1 - \omega)$ we get the following iterative algorithm.

Algorithm 2.1 *Input* $A, C \in \mathbb{M}_{m,n}$, $B, D \in \mathbb{M}_{p,q}$, and $E \in \mathbb{M}_{m,q}$. *Set* $A' = A^T$, $B' = B^T$, $C' = C^T$, and $D' = D^T$. *Choose an initial matrix* $X(0) \in \mathbb{M}_{n,p}$. *For each* $k = 0, 1, 2, \dots$ *until End, do:*

$$\begin{aligned} F(k) &= E - AX(k)B - CX(k)D, \\ X(k + 1) &= X(k) + \theta[A'F(k)B' + C'F(k)D']. \end{aligned}$$

Note that our algorithm avoids duplicate computations by introducing $F(k)$ at each iteration. To stop the process, we can impose a stopping rule such as $\|F(k)\|_F < \epsilon$ or $\|F(k)\|_F/\|E\|_F < \epsilon$, where ϵ is a chosen permissible error. The convergence of Algorithm 2.1 relies on the convergence factors θ , which will be determined in the next section. Note that the algorithm requires only one parameter and one initial value and uses less computing time than other gradient-based algorithms mentioned in Introduction.

3 Convergence analysis of the algorithm

In this section, we analyze the convergence of Algorithm 2.1. We convert the matrix iteration process to a first-order linear difference vector equation with contraction matrix as the coefficient. It follows that the contraction reflects the convergence criteria, convergence rate, and error estimates of the algorithm.

To analyze this algorithm, we recall useful facts in matrix analysis.

Lemma 3.1 (e.g. [7]) *For any matrices* A *and* B *of conforming dimensions, we have*

- (i) $\|A^T A\|_2 = \|A\|_2^2$;
- (ii) $\|AB\|_F \leq \|A\|_2 \|B\|_F$;
- (iii) *if* A *is symmetric, then* $\|A\|_2 = \rho(A)$;
- (iv) $\|A \otimes B\|_2 = \|A\|_2 \|B\|_2$.

3.1 Convergence criteria

From Algorithm 2.1 we start with considering the error matrix

$$\widehat{X}(k) = X(k) - X.$$

We will show that $\widehat{X}(k) \rightarrow 0$ or, equivalently, $\text{vec } \widehat{X}(k) \rightarrow 0$ as $k \rightarrow \infty$. Now we convert the matrix iteration process to a first-order linear difference vector equation with matrix coefficient. Indeed, we have

$$\begin{aligned} F(k) &= (AXB + CXD) - AX(k)B - CX(k)D \\ &= -A\widehat{X}(k)B - C\widehat{X}(k)D, \end{aligned}$$

and thus

$$\text{vec } F(k) = -(B^T \otimes A + D^T \otimes C) \text{vec } \widehat{X}(k) = -P \text{vec } \widehat{X}(k).$$

It follows that

$$\begin{aligned} \text{vec } \widehat{X}(k + 1) &= \text{vec} \{ \widehat{X}(k) + \theta [A^T F(k) B^T + C^T F(k) D^T] \} \\ &= \text{vec } \widehat{X}(k) + \theta [\text{vec}(A^T F(k) B^T) + \text{vec}(C^T F(k) D^T)] \\ &= \text{vec } \widehat{X}(k) + \theta P^T \text{vec } F(k) \\ &= \text{vec } \widehat{X}(k) - \theta P^T P \text{vec } \widehat{X}(k) \\ &= P_\theta \text{vec } \widehat{X}(k), \end{aligned} \tag{9}$$

where $P_\theta = I_{np} - \theta P^T P$. Denoting $u(k) = \text{vec } \widehat{X}(k)$ for $k \in \mathbb{N}$, we obtain a first-order linear difference vector equation, as desired.

Note that iteration (9) is also the Picard iteration

$$u(k + 1) = Tu(k), \quad k \in \mathbb{N}, \tag{10}$$

where T is the self-mapping on \mathbb{R}^n defined by $Tx = P_\theta x$. We will find some properties of T yielding that the iteration converges to the fixed point $u^* = 0$ of T for arbitrary initial point $u(0)$. In fact, this can be guaranteed by the Banach contraction principle:

Theorem 3.2 (e.g., [39, Sect. 5.1]) *Let (\mathbb{X}, d) be a nonempty complete metric space. Let $T : \mathbb{X} \rightarrow \mathbb{X}$ be a contraction, that is, there is a constant $\alpha \in [0, 1)$ such that*

$$d(Tx, Ty) \leq \alpha d(x, y) \quad \forall x, y \in \mathbb{X}.$$

Then T has a unique fixed point x^ . The following estimates are equivalent and describe the convergence rate:*

- (i) $d(x_{n+1}, x^*) \leq \alpha d(x_n, x^*)$;
- (ii) *prior estimate:* $d(x_n, x^*) \leq \frac{\alpha^n}{1-\alpha} d(x_1, x_0)$;
- (iii) *posterior estimate:* $d(x_{n+1}, x^*) \leq \frac{\alpha}{1-\alpha} d(x_{n+1}, x_n)$.

Now we look for some conditions on P_θ making the mapping T a contraction. For each $x \in \mathbb{R}^n$, we have by Lemma 3.1 that

$$\|Tx\|_F = \|P_\theta x\|_F \leq \|P_\theta\|_2 \|x\|_F = \rho(P_\theta) \|x\|_F.$$

The last equality holds since P_θ is a symmetric matrix. It follows that

$$\|Tx - Ty\|_F = \|T(x - y)\|_F \leq \rho(P_\theta) \|x - y\|_F \quad \forall x, y \in \mathbb{R}^n.$$

Thus, if $\rho(P_\theta) < 1$, then T is a contraction relative to the metric induced by $\|\cdot\|_F$. Note that further characterizations of matrix contractions, involving (induced) matrix norms, are given, for example, in [40]. Since P_θ is a symmetric matrix, all its eigenvalues are real, and thus

$$\rho(P_\theta) = \max\{|1 - \theta\lambda_{\min}(P^T P)|, |1 - \theta\lambda_{\max}(P^T P)|\}. \tag{11}$$

It follows that $\rho(P_\theta) < 1$ if and only if

$$0 < \theta\lambda_{\min}(P^T P) < 2 \quad \text{and} \quad 0 < \theta\lambda_{\max}(P^T P) < 2. \tag{12}$$

Since P is invertible and $P^T P$ is positive semidefinite, we have that $P^T P$ is positive definite and $\lambda_{\min}(P^T P) > 0$. The positive definiteness of $P^T P$ and Lemma 3.1(i) imply

$$\lambda_{\max}(P^T P) = \|P^T P\|_2 = \|P\|_2^2.$$

Hence condition (12) holds if and only if

$$0 < \theta < \frac{2}{\|P\|_2^2}. \tag{13}$$

Therefore, if (13) holds, then the sequence $X(k)$ generated by Algorithm 2.1 converges to the solution of (4) for any initial value $X(0)$.

Conversely, suppose that θ does not satisfy (13). The above discussion implies that $\rho(P_\theta) \geq 1$, that is, there is an eigenvalue λ of P_θ such that $|\lambda| \geq 1$. We can choose an eigenvector $v \in \mathbb{R}^n - \{0\}$ such that $P_\theta v = \lambda v$. The Picard iteration (10) with initial point $u(0) = v$ yields

$$u(k) = T^k u(0) = T^k v = \lambda^k v \not\rightarrow 0.$$

Thus $\widehat{X}(k) \not\rightarrow 0$ or $X(k) \not\rightarrow X$.

We summarize a necessary and sufficient condition for the convergence criteria as follows.

Theorem 3.3 *Let $\theta \in \mathbb{R}$ be given. Then the sequence $X(k)$ generated by Algorithm 2.1 converges to the solution of (4) for any initial value $X(0)$ if and only if θ satisfies (13).*

Thus, if $\theta \leq 0$ or $\theta \geq \frac{2}{\|P\|_2^2}$, then Algorithm 2.1 is not applicable for some initial values.

3.2 Convergence rate and error estimates

We now apply the Banach contraction principle to analyze the convergence rate and error estimates of Algorithm 2.1. Note that the error at each step of the associated Picard iteration is equal to that of the original matrix iterative algorithm. Indeed, for any $k \in \mathbb{N} \cup \{0\}$, we have

$$\begin{aligned} \|u^{(k)} - u^*\|_F &= \|\text{vec } \widehat{X}(k)\|_F = \|\widehat{X}(k)\|_F = \|X(k) - X\|_F, \\ \|u^{(k+1)} - u^{(k)}\|_F &= \|\text{vec } \widehat{X}(k+1) - \text{vec } \widehat{X}(k)\|_F \\ &= \|\widehat{X}(k+1) - \widehat{X}(k)\|_F = \|X(k+1) - X(k)\|_F. \end{aligned}$$

Thus by Theorem 3.2(i) we obtain

$$\|X(k+1) - X\|_F \leq \rho(P_\theta) \|X(k) - X\|_F. \tag{14}$$

It follows inductively that for each $k \in \mathbb{N}$,

$$\|X(k) - X\|_F \leq \rho^k(P_\theta) \|X(0) - X\|_F. \tag{15}$$

Hence $\rho(P_\theta)$ describes how fast the approximate solutions $X(k)$ converge to the exact solution X . The smaller the spectral radius, the faster $X(k)$ goes to X . In that case, since $\rho(P_\theta) < 1$, if $\|X(k) - X\|_F \neq 0$, then

$$\|X(k+1) - X\|_F < \|X(k) - X\|_F. \tag{16}$$

Moreover, from the prior and posterior estimates in Theorem 3.4 we obtain the bounds

$$\|X(k) - X\|_F \leq \frac{\rho^k(P_\theta)}{1 - \rho(P_\theta)} \|X(1) - X(0)\|_F, \tag{17}$$

$$\|X(k+1) - X\|_F \leq \frac{\rho(P_\theta)}{1 - \rho(P_\theta)} \|X(k+1) - X(k)\|_F. \tag{18}$$

we summarize our discussion in the following theorem.

Theorem 3.4 *Suppose that the parameter θ is chosen as in Theorem 3.3 so that the sequence $X(k)$ generated by Algorithm 2.1 converges to X for any initial value. Then we have:*

- *The convergence rate of the algorithm is governed by the spectral radius (11).*
- *The error estimates $\|X(k+1) - X\|_F$ compared to the previous step and the first step are provided by (14) and (15), respectively. In particular, the error at each iteration gets smaller than the (nonzero) previous one, as in (16).*
- *The prior estimate (17) and the posterior estimate (18) hold.*

From (11), if the eigenvalues of $\theta P^T P$ are close to 1, then the spectral radius of the iterative matrix is close to 0, and hence the errors $\widehat{X}(k)$ converge faster to 0.

Remark 3.5 The convergence criteria and the convergence rate of Algorithm 2.1 depend on A, B, C, D but not on E . However, the matrix E can be used for a stopping criteria.

The following proposition determines the iteration number for which the approximated solution $X(k)$ is close to the exact solution X so that $\|X(k) - X\|_F < \epsilon$.

Proposition 3.6 *According to Algorithm 2.1, for each given error $\epsilon > 0$, we have $\|X(k) - X\|_F < \epsilon$ after k^* iterations for any*

$$k^* > \frac{\log \epsilon - \log \|X(0) - X\|_F}{\log \rho(P_\theta)}. \tag{19}$$

Proof From estimate (15) we have

$$\|X(k) - X\|_F \leq \rho^k(P_\theta) \|X(0) - X\|_F \rightarrow 0 \text{ as } k \rightarrow \infty.$$

This precisely means that for each given $\epsilon > 0$, there is $k^* \in \mathbb{N}$ such that for all $k \geq k^*$,

$$\rho^k(P_\theta) \|X(0) - X\|_F < \epsilon.$$

Taking logarithms, we have that this condition is equivalent to (19). Thus, if we run Algorithm 2.1 k^* times, then we get $\|X(k) - X\|_F < \epsilon$, as desired. \square

3.3 Optimal parameter

We now discuss the fastest convergence factors for Algorithm 2.1.

Theorem 3.7 *Let $0 < \theta < \frac{2}{\|P\|_2^2}$. Denote κ as the condition number of the matrix P . Then the optimal value of θ for which Algorithm 2.1 is applicable for any initial value is determined by*

$$\theta_{\text{opt}} = \frac{2}{\lambda_{\min}(P^T P) + \lambda_{\max}(P^T P)}. \tag{20}$$

In this case the spectral radius of the iteration matrix is given by

$$\rho(P_{\theta_{\text{opt}}}) = \frac{\lambda_{\max}(P^T P) - \lambda_{\min}(P^T P)}{\lambda_{\max}(P^T P) + \lambda_{\min}(P^T P)} = \frac{\kappa^2 - 1}{\kappa^2 + 1}. \tag{21}$$

Proof Theorem 3.3 tells us that the convergence of the algorithm implies (13). The convergence rate of the algorithm is the same as that of the linear iteration (9) and thus is governed by the spectral radius (11). Hence we would like to minimize the spectral radius $\rho(P_\theta)$ subject to condition (13). Putting $a = \lambda_{\min}(P^T P)$ and $b = \lambda_{\max}(P^T P)$, we make the following optimization:

$$\min_{0 < \theta < \frac{2}{b}} \{ \max \{ |1 - a\theta|, |1 - b\theta| \} \}.$$

The minimality is reached at $\theta_{\text{opt}} = 2/(a + b)$, so that the minimum value of $\rho(P_\theta)$ is equal to $(b - a)/(b + a)$. \square

From Theorem 3.7, Algorithm 2.1 has a fast convergence if the condition number of P is close to 1 or, equivalently, $\lambda_{\max}(P^T P)$ is close to $\lambda_{\min}(P^T P)$.

4 Iterative algorithms for particular cases of the generalized Sylvester equation

In this section, we discuss iterative algorithms for solving interesting particular cases of the generalized Sylvester equation.

4.1 The equation $AXB = E$

Assume that the equation $AXB = E$ has a unique solution or, equivalently, the square matrix $Q := B^T \otimes A$ is invertible. In the particular case where A and B are square matrices, this condition is reduced to the invertibility of both A and B . The following algorithm is proposed to find the solution X .

Algorithm 4.1 Set $A' = A^T$ and $B' = B^T$. Choose $X(0) \in \mathbb{M}_{n,p}$. For each $k = 0, 1, 2, \dots$ until End, do:

$$X(k + 1) = X(k) + \theta A'(C - AX(k)B)B'.$$

Note that $Q^T Q = BB^T \otimes A^T A$ by the mixed-product property of the Kronecker product. Since Q is invertible, so is $Q^T Q$. It follows that $A^T A$ and BB^T are positive definite. Thus

$$\lambda_{\min}(Q^T Q) = \lambda_{\min}(A^T A)\lambda_{\min}(BB^T) = \lambda_{\min}(A^T A)\lambda_{\min}(B^T B),$$

and, similarly, $\lambda_{\max}(Q^T Q) = \lambda_{\max}(A^T A)\lambda_{\max}(B^T B)$. Now we obtain the following:

Corollary 4.2 Assume that the equation $AXB = E$ has a unique solution. Let $\theta \in \mathbb{R}$. Then we have:

- 1) Algorithm 4.1 is applicable for any initial value $X(0)$ if and only if

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

- 2) The convergence rate of the iteration is governed by the spectral radius

$$\begin{aligned} &\rho(I_{np} - \theta Q^T Q) \\ &= \max\{|1 - \theta \lambda_{\min}(A^T A)\lambda_{\min}(B^T B)|, |1 - \theta \lambda_{\max}(A^T A)\lambda_{\max}(B^T B)|\}. \end{aligned}$$

- 3) The optimal convergence factor for which Algorithm 4.1 is applicable for any initial value is given by

$$\theta_{\text{opt}} = [\lambda_{\min}(A^T A)\lambda_{\min}(B^T B) + \lambda_{\max}(A^T A)\lambda_{\max}(B^T B)]^{-1}/2.$$

4.2 The Sylvester equation

Suppose that $m = n$ and $p = q$. Assume that the Sylvester equation

$$AX + XD = E \tag{22}$$

has a unique solution. This condition is equivalent to the invertibility of the Kronecker sum $D^T \oplus A := D^T \otimes I_n + I_p \otimes A$, or all possible sums of eigenvalues of A and D are nonzero.

Algorithm 4.3 Set $A' = A^T$ and $D' = D^T$. Choose $X(0) \in \mathbb{M}_{n,p}$. For each $k = 0, 1, 2, \dots$ until End, do:

$$F(k) = E - AX(k) - X(k)D,$$

$$X(k + 1) = X(k) + \theta[A'F(k) + F(k)D'].$$

Corollary 4.4 Assume that the equation $AX + XD = E$ has a unique solution X . Then the iterative sequence $X(k)$ generated by Algorithm 4.3 converges to X for any initial value $X(0)$ if and only if

$$0 < \theta < \frac{2}{\|D^T \oplus A\|_2^2}.$$

Error estimates and the optimal convergence factor for Algorithm 4.3 can also be obtained from Theorem 2.1 when B and C are the identity matrices.

Remark 4.5 If A and D are positive semidefinite, then $\|D^T \oplus A\|_2$ is reduced to $\|A\|_2 + \|D\|_2$.

4.3 The Kalman–Yakubovich equation

Suppose that $m = n$ and $p = q$. Assume that the Kalman–Yakubovich equation

$$AXB + X = E \tag{23}$$

has a unique solution. This condition is equivalent to the invertibility of $B^T \otimes A + I_{np}$ or, equivalently, to that all possible products of eigenvalues of A and B are not equal to -1 .

Algorithm 4.6 Set $A' = A^T$ and $B' = B^T$. Choose $X(0) \in \mathbb{M}_{n,p}$. For each $k = 0, 1, 2, \dots$ until End, do:

$$F(k) = E - AX(k)B - X(k),$$

$$X(k + 1) = X(k) + \theta[A'F(k)B' + F(k)].$$

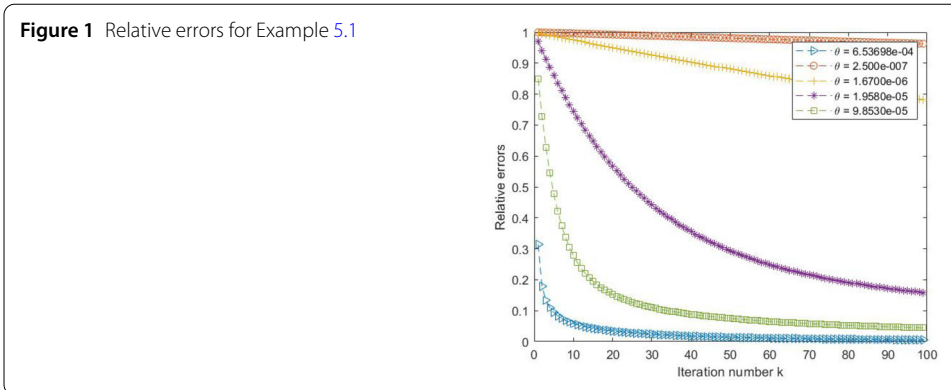
Corollary 4.7 Assume that the equation $AXB + X = E$ has a unique solution X . The iterative sequence $X(k)$ generated by Algorithm 4.6 converges to X for any initial value $X(0)$ if and only if

$$0 < \theta < \frac{2}{\|B^T \otimes A + I_{n^2}\|_2^2}.$$

Remark 4.8 If A and B are positive semidefinite, then $\|B^T \otimes A + I_{n^2}\|_2$ is reduced to $\|A\|_2\|B\|_2 + 1$.

5 Numerical simulations

In this section, we report numerical results illustrating the effectiveness of Algorithm 2.1. We consider matrix systems from small dimensions (say, 2×2) to large dimensions (say, 120×120). We investigate the effect of changing convergence factors (see Example 5.1)



and initial points (Example 5.2). We compare the performance of the algorithm to the direct Kronecker linearization (Example 5.1) and recent iterative algorithms (Example 5.3). We show that Algorithm 2.1 is still effective when dealing with a nonsquare problem (see Example 5.4). To measure errors at step k of the iteration, we consider the following relative error:

$$\frac{\|AX(k)B + CX(k)D - E\|_F}{\|E\|_F}.$$

All iterations have been carried out on the same PC environment: MATLAB R2018a, Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz, RAM 8.00 GB. To measure the computational time (in seconds) taken for a program, we use the *tic* and *toc* functions in MATLAB and abbreviate CT for it. The readers are recommended to consider both reported errors and CTs while comparing the performance of any algorithms.

Example 5.1 Consider the equation $AXB + CXD = E$, where A, B, C, D, E, X of sizes 100×100 are given by

$$\begin{aligned} A &= \text{tridiag}(-1, 2, -1), & B &= \text{tridiag}(6, 4, -1), & C &= \text{tridiag}(1, 2, 3), \\ D &= \text{tridiag}(4, 2, -5), & E &= \text{heptadiag}(2, -22, 16, 92, 36, -58, -42). \end{aligned}$$

We run Algorithm 2.1 with five convergence factors; one of them is the optimal convergence factor $\theta_{\text{opt}} = 6.5398\text{e-}04$, determined by Theorem 3.7. According to (13), the range of appropriate θ is given by $0 < \theta < 2/\|P\|_2^2 \approx 6.5398\text{e-}04$ (in this case, $\lambda_{\min}(P^T P) \approx 0$), that is, Algorithm 2.1 is applicable for every chosen θ . The result after 100 iterations is presented by Fig. 1 and Table 1. From the relative error plot versus iteration time in Fig. 1 we see that the optimal convergence factor gives the fastest convergence. Table 1 shows that the computational times with the five convergence factors are significantly less than that of the direct method $\text{vec } X = P^{-1} \text{vec } E$. The relative errors after 100 iterations are very small in comparison with the dimensions of the coefficient matrices.

Example 5.2 In this example, we consider the equation $AXB + CXD = E$ with different sizes of matrices, say, 2×2 , 10×10 , 100×100 , and 120×120 . For each case, we take $A = \text{tridiag}(7, -2, 5)$, $B = \text{tridiag}(1, 6, 8)$, $C = \text{tridiag}(3, -9, 1)$, $D = \text{tridiag}(9, -2, 5)$, and $E = \text{heptadiag}(34, 21, 99, 8, 252, -9, 135)$ of corresponding sizes. We denote by $\text{ones}(n)$ the $n \times n$

Table 1 Relative error and computational time for Example 5.1

θ	Relative error	CT
$\theta = 6.5398e-04$	5.800×10^{-3}	0.0324
$\theta = 2.500e-07$	9.629×10^{-1}	0.0311
$\theta = 1.6700e-06$	7.810×10^{-1}	0.0322
$\theta = 1.9580e-05$	1.585×10^{-1}	0.0316
$\theta = 9.8530e-05$	4.460×10^{-2}	0.0303
Direct method		20.8804

Table 2 Relative error and computational time for Example 5.2

Initial value	Dimension 2 × 2		Dimension 10 × 10		Dimension 100 × 100		Dimension 120 × 120	
	$\theta_{opt} = 9.8701e-05$		$\theta_{opt} = 1.6800e-05$		$\theta_{opt} = 1.4951e-05$		$\theta_{opt} = 1.4945e-05$	
	CT	Error	CT	Error	CT	Error	CT	Error
X_1	3.4340e-04	0.2823	6.1930e-04	0.0934	0.0134	0.1096	0.0236	0.1098
X_2	3.1870e-04	0.2770	3.7480e-04	0.0928	0.0151	0.1095	0.0240	0.1098
X_3	3.0990e-04	0.2610	5.6450e-04	0.0915	0.0143	0.1096	0.0231	0.1099
X_4	3.1040e-04	0.2250	4.5500e-04	0.0925	0.0147	0.1110	0.0228	0.1113
X_5	3.1020e-04	0.2102	3.8710e-04	0.0947	0.0146	0.1120	0.0233	0.1124
X_6	3.1550e-04	0.1867	3.7830e-04	0.1004	0.0145	0.1144	0.0225	0.1149

matrix that contains 1 at every position. For each $n \in \{2, 10, 100, 120\}$, we run Algorithm 2.1 with distinct initial candidates:

$$\begin{aligned}
 X_1 &= 0.1 \times \text{ones}(n), & X_2 &= 0.2 \times \text{ones}(n), & X_3 &= 0.5 \times \text{ones}(n), \\
 X_4 &= 1.2 \times \text{ones}(n), & X_5 &= 1.5 \times \text{ones}(n), & X_6 &= 2 \times \text{ones}(n).
 \end{aligned}$$

We run 50 iterations for the matrices of dimensions 2 × 2 and 10 × 10, whereas we run 100 iterations for large dimensions 100 × 100 and 120 × 120. The optimal convergence factors θ for each case are provided in Table 2. The computational times and errors reported in Table 2 show that our algorithm is satisfactorily applicable for all initial candidates and different sizes of coefficient matrices.

Example 5.3 We consider the equation $AX + XB = C$ with three cases of coefficient matrix sizes, namely 2 × 2, 10 × 10, and 100 × 100. We set

$$A_0 = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}, \quad B_0 = \begin{bmatrix} 8 & 0 \\ -5 & -6 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} 2 & 3 \\ -6 & 9 \end{bmatrix}.$$

For each $n \in \{2, 10, 100\}$, we consider the coefficient matrices $A = A_0 \otimes I_{n/2}$, $B = B_0 \otimes I_{n/2}$, and $X^* = Z \otimes I_{n/2}$ together with initial condition $X(0) = 10^{-6} \times \text{ones}(n)$. We compare Algorithm 2.1 with GI [25], RGI [34], MGI [35], JGI [36, Algorithm 4], AJGI1 [36, Algorithm 5], AJGI2 [36, Algorithm 6], and LSI [33] algorithms.

We run 50 iterations for dimension 2 × 2, 100 iterations for dimension 10 × 10, and 200 iterations for dimension 100 × 100. The relative error and the computational time in Table 3 reveal that our algorithm well performs comparing to the other algorithms. Figure 2 displays the error plots of the first 100 iterations for each case; 2 × 2 (a), 10 × 10 (b), and 100 × 100 (c). We can see that our algorithm gives the fastest convergence.

Example 5.4 We consider the generalized Sylvester matrix equation with rectangular unknown matrix $X \in M_{50,100}$. Let $A, C \in M_{50}$ and $B, D \in M_{100}$ be defined by $A =$

Table 3 Relative error and computational time for Example 5.3

Method	Dimension 2×2		Dimension 10×10		Dimension 100×100	
	CT	Error	CT	Error	CT	Error
GI opt	2.7170e-04	9.4674e-04	4.5690e-04	0.0018	0.0188	0.0036
GI	2.3330e-04	0.0996	4.3030e-04	0.2125	0.0160	0.0709
RGI	2.6100e-04	0.0254	5.3710e-04	0.8040	0.0205	0.0456
MGI	2.7400e-04	0.0074	7.2790e-04	0.0072	0.0195	0.0240
JGI	2.6410e-04	0.0772	7.6200e-04	0.1649	0.0200	0.0267
AJGI 1	2.8810e-04	0.1601	7.8790e-04	0.1465	0.0267	0.1936
AJGI 2	2.8420e-04	0.0167	8.2090e-04	0.0251	0.0262	0.0229
LSI	2.5810e-04	0.2567	7.9300e-04	0.2935	0.0261	0.2329

Figure 2 Natural logarithm relative errors for Example 5.3

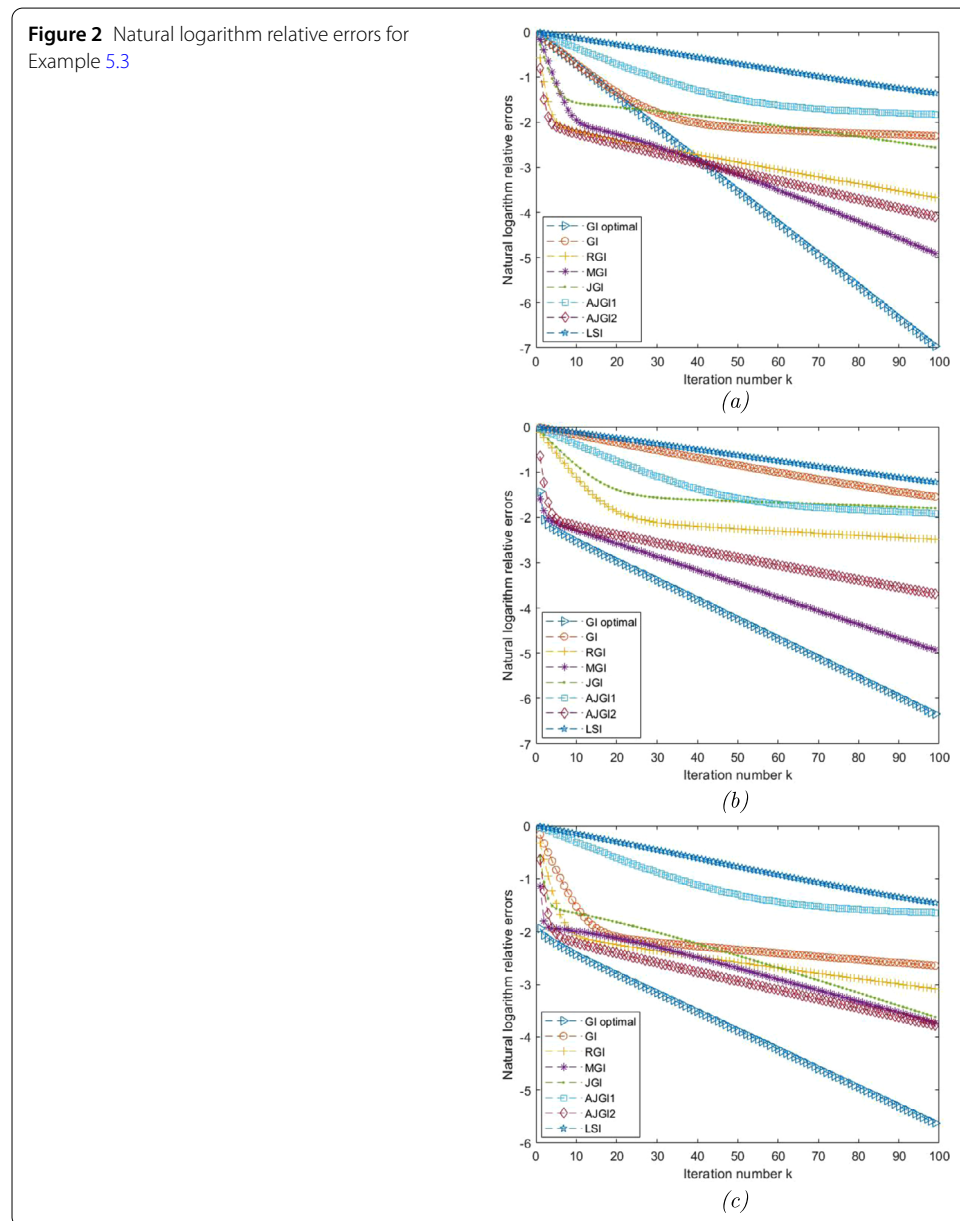
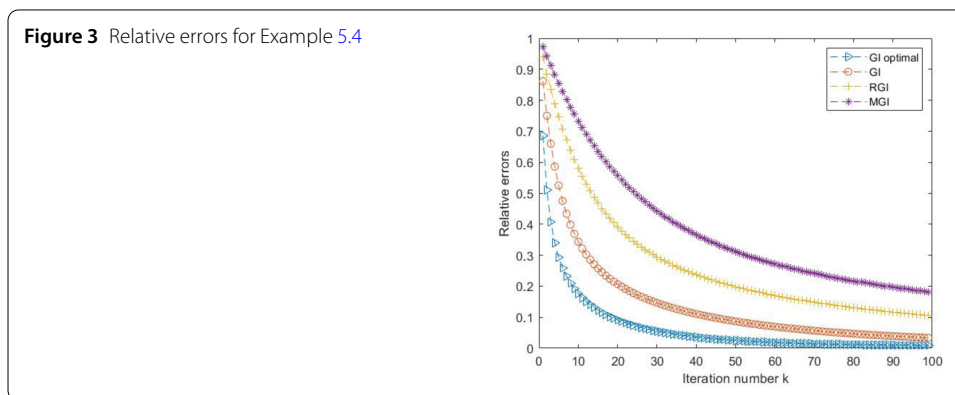


Table 4 Relative error and computational time for Example 5.4

Method	CT	Error
GI opt	0.0185	0.0089
GI	0.0176	0.0336
RGI	0.0199	0.1053
MGI	0.0196	0.1819
direct method	3.6177	



$\text{tridiag}(-1, 2, -1)$, $B = \text{tridiag}(1, 4, -3)$, $C = \text{tridiag}(3, -1, 2)$, and $D = \text{tridiag}(3, 5, 7)$. The exact solution is given by $X^* = \tilde{X} \otimes I_{10}$, where

$$\tilde{X} = \begin{bmatrix} 1 & 3 & -5 & 9 & 5 & 7 & 4 & -6 & 9 & 10 \\ 2 & -8 & 9 & -7 & 4 & 5 & -6 & 1 & 2 & 3 \\ 2 & 3 & 5 & 7 & 9 & -8 & -5 & 0 & 1 & 2 \\ 6 & 9 & -8 & 7 & 5 & 4 & -2 & 0 & 3 & 6 \\ -8 & -9 & 6 & 5 & -1 & 2 & 0 & 3 & -4 & -7 \end{bmatrix}.$$

The constant matrix E is determined by $E = AX^*B + CX^*D$.

We compare Algorithm 2.1 ($\theta_{\text{opt}} = 6.4000\text{e-}05$) and the algorithms compatible with non-square matrices, that is, GI [25], RGI [34], and MGI [35], with 100 iterations. The relative error at terminal iteration and the computational time of each algorithm and of the direct method are shown in Table 4, whereas Fig. 3 displays the error plots. Both of them reveal the effectiveness in performance of our algorithm. The computational times of GI-opt and GI are less than those of RGI and MGI. The reason that GI-opt takes little more time than GI is that our algorithm needs more time to compute θ_{opt} in Theorem 3.7. On the other hand, Algorithm 2.1 obtains a significantly better error than GI algorithm.

6 Conclusion

We propose a gradient-based iterative algorithm (Algorithm 2.1) for solving the rectangular generalized Sylvester matrix equation (4) and its famous particular cases. Theorem 3.3 informs us that the parameter θ must be chosen properly to have the proposed algorithm applicable for any initial matrices. Moreover, we determine the optimal convergence factors, which makes the algorithm reach the fastest convergence rate. The asymptotic convergence rate of the algorithm is governed by the spectral radius of $I_{np} - \theta P^T P$. If the eigenvalue $\theta P^T P$ is close to 1, then the algorithm converges faster in the long run. The numerical

simulations reveal that our algorithm is suitable for both small and large matrix systems, both square and nonsquare problems, and any initial points. In addition, the algorithm always gives the effective performance comparing to the errors obtained from recent methods, namely, GI, RGI, MGI, JGI, AJGI1, AJGI2, and LSI algorithms. There are two reasons that cause our algorithm to perform well. The first reason is that the algorithm requires only one parameter and one initial value, and avoids duplicated computations. The second is that the convergence factor is chosen by an optimization method.

Acknowledgements

This work was supported by Thailand Science Research and Innovation (Thailand Research Fund). The authors would like to thank referees for useful comments and suggestions.

Funding

This second author expresses his gratitude to Thailand Science Research and Innovation (Thailand Research Fund), Grant No. MRG6280040, for financial support.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

All authors contributed equally and significantly in writing this article. All authors read and approved the final manuscript.

Author details

¹Department of Mathematics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, 10520, Bangkok, Thailand. ²Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, 10330, Bangkok, Thailand.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 2 July 2020 Accepted: 14 December 2020 Published online: 07 January 2021

References

1. Lancaster, P., Tismenetsky, M.: *The Theory of Matrices*, 2nd edn. Academic Press, San Diego (1985)
2. Dullerud, G.E., Paganini, F.: *A Course in Robust Control Theory: A Convex Approach*. Springer, New York (1999)
3. Varga, A.: Robust pole assignment via Sylvester equation based state feedback parametrization. pp. 13–18 (2000)
4. Magnus, J.R., Neudecker, H.: *Matrix Differential Calculus with Applications in Statistics and Econometrics*, 3rd edn. Wiley, Chichester (2007)
5. Epton, M.: Methods for the solution of $AXD - BXC = E$ and its applications in the numerical solution of implicit ordinary differential equations. *BIT Numer. Math.* **20**, 341–345 (1980)
6. Calvetti, D., Reichel, L.: Application of ADI iterative methods to the restoration of noisy images. *SIAM J. Matrix Anal. Appl.* **17**, 165–186 (1996)
7. Horn, R.A., Johnson, C.R.: *Topics in Matrix Analysis*. Cambridge University Press, New York (1991)
8. Simoncini, V.: Computational methods for linear matrix equations. *SIAM Rev.* **58**(3), 377–441 (2016). <https://doi.org/10.1137/130912839>
9. Bartels, R.H., Stewart, G.W.: Solution of the matrix equation $AX + XB = C$. *Commun. ACM* **15**(9), 820–826 (1972). <https://doi.org/10.1145/361573.361582>
10. Golub, G., Nash, S., Van Loan, C.: A Hessenberg–Schur method for the problem $AX + XB = C$. *IEEE Trans. Autom. Control* **24**(6), 909–913 (1979). <https://doi.org/10.1109/TAC.1979.1102170>
11. Starke, G., Niethammer, W.: SOR for $AX - XB = C$. *Linear Algebra Appl.* **154–156**, 355–375 (1991). [https://doi.org/10.1016/0024-3795\(91\)90384-9](https://doi.org/10.1016/0024-3795(91)90384-9)
12. Benner, P., Quintana-Orti, E.S.: Solving stable generalized Lyapunov equations with the matrix sign function. *Numer. Algorithms* **20**, 75–100 (1999). <https://doi.org/10.1023/A:1019191431273>
13. Jonsson, I., Kagstrom, B.: Recursive blocked algorithms for solving triangular systems—part I: one-sided and coupled Sylvester-type matrix equations. *ACM Trans. Math. Softw.* **28**(4), 392–415 (2002). <https://doi.org/10.1145/592843.592845>
14. Jonsson, I., Kagstrom, B.: Recursive blocked algorithms for solving triangular systems—part II: two-sided and generalized Sylvester and Lyapunov matrix equations. *ACM Trans. Math. Softw.* **28**(4), 416–435 (2002). <https://doi.org/10.1145/592843.592846>
15. Kaabi, A., Kerayechian, A., Toutounian, F.: A new version of successive approximations method for solving Sylvester matrix equations. *Appl. Math. Comput.* **186**(1), 638–648 (2007). <https://doi.org/10.1016/j.amc.2006.08.007>
16. Lin, Y.Q.: Implicitly restarted global FOM and GMRES for nonsymmetric matrix equations and Sylvester equations. *Appl. Math. Comput.* **167**(2), 1004–1025 (2005). <https://doi.org/10.1016/j.amc.2004.06.141>

17. Kressner, D., Sirkovic, P.: Truncated low-rank methods for solving general linear matrix equations. *Numer. Linear Algebra Appl.* **22**(3), 564–583 (2015). <https://doi.org/10.1002/nla.1973>
18. Dehghan, M., Shirilord, A.: A generalized modified Hermitian and skew-Hermitian splitting (GMHSS) method for solving complex Sylvester matrix equation. *Appl. Math. Comput.* **348**, 632–651 (2019)
19. Dehghan, M., Shirilord, A.: Solving complex Sylvester matrix equation by accelerated double-step scale splitting (ADSS) method. *Eng. Comput.* (2019). <https://doi.org/10.1007/s00366-019-00838-6>
20. Li, S.Y., Shen, H.L., Shao, X.H.: PHSS iterative method for solving generalized Lyapunov equations. *Mathematics* **7**(1), Article ID 38 (2019). <https://doi.org/10.3390/math7010038>
21. Shen, H.L., Li, Y.R., Shao, X.H.: The four-parameter PSS method for solving the Sylvester equation. *Mathematics* **7**(1), Article ID 105 (2019). <https://doi.org/10.3390/math7010105>
22. Hajarian, M.: Generalized conjugate direction algorithm for solving the general coupled matrix equations over symmetric matrices. *Numer. Algorithms* **73**(3), 591–609 (2016)
23. 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)
24. 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)
25. Ding, F., Chen, T.W.: Iterative least-squares solutions of coupled Sylvester matrix equations. *Syst. Control Lett.* **54**(2), 95–107 (2005). <https://doi.org/10.1016/j.sysconle.2004.06.008>
26. Ding, F., Chen, T.W.: 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>
27. Ding, F., Chen, T.W.: 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>
28. Zhang, X.D., Sheng, X.P.: The relaxed gradient based iterative algorithm for the symmetric (skew symmetric) solution of the Sylvester equation $AX + XB = C$. *Math. Probl. Eng.* **2017**, 1624969 (2017). <https://doi.org/10.1155/2017/1624969>
29. 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>
30. Hajarian, M.: Matrix iterative methods for solving the Sylvester-transpose and periodic Sylvester matrix equations. *J. Franklin Inst.* **350**, 3328–3341 (2013)
31. Hajarian, M.: Solving the general Sylvester discrete-time periodic matrix equations via the gradient based iterative method. *Appl. Math. Lett.* **52**, 87–95 (2015)
32. Ding, F., Chen, T.W.: 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>
33. Ding, F., Liu, P.X., 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>
34. Niu, Q., Wang, X., Lu, L.Z.: A relaxed gradient based algorithm for solving Sylvester equations. *Asian J. Control* **13**(3), 461–464 (2011). <https://doi.org/10.1002/asjc.328>
35. Wang, X., Dai, L., Liao, D.: A modified gradient based algorithm for solving Sylvester equations. *Appl. Math. Comput.* **218**(9), 5620–5628 (2012). <https://doi.org/10.1016/j.amc.2011.11.055>
36. Tian, Z.L., Tian, M.Y., Gu, C.Q., Hao, X.N.: An accelerated Jacobi-gradient based iterative algorithm for solving Sylvester matrix equations. *Filomat* **31**(8), 2381–2390 (2017). <https://doi.org/10.2298/FIL1708381T>
37. Dehghan, M., Shirilord, A.: An iterative method for solving the generalized coupled Sylvester matrix equations over generalized bisymmetric matrices. *Appl. Math. Model.* **34**(3), 639–654 (2010)
38. Dehghan, M., Shirilord, A.: An iterative algorithm for the reflexive solutions of the generalized coupled Sylvester matrix equations and its optimal approximation. *Appl. Math. Comput.* **202**(2), 571–588 (2008)
39. Kreyszig, E.: *Introductory Functional Analysis with Applications*. Wiley, New York (1978)
40. Lim, T.C.: Nonexpansive matrices with applications to solutions of linear systems by fixed point iterations. *Fixed Point Theory Appl.* **2010**, Article ID 821928 (2009). <https://doi.org/10.1155/2010/821928>

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)