Article

Conjugate Gradient Algorithm for Least-Squares Solutions of a Generalized Sylvester-Transpose Matrix Equation

Kanjanaporn Tansri [1,†] and Pattrawut Chansangiam [1,*,†]

Department of Mathematics, School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
* Correspondence: pattrawut.ch@kmitl.ac.th; Tel.: +66-9352-666-00
† These authors contributed equally to this work.

Abstract: We derive a conjugate-gradient type algorithm to produce approximate least-squares (LS) solutions for an inconsistent generalized Sylvester-transpose matrix equation. The algorithm is always applicable for any given initial matrix and will arrive at an LS solution within finite steps. When the matrix equation has many LS solutions, the algorithm can search for the one with minimal Frobenius-norm. Moreover, given a matrix Y, the algorithm can find a unique LS solution closest to Y. Numerical experiments show the relevance of the algorithm for square/non-square dense/sparse matrices of medium/large sizes. The algorithm works well in both the number of iterations and the computation time, compared to the direct Kronecker linearization and well-known iterative methods.

Keywords: generalized Sylvester-transpose matrix equation; conjugate gradient algorithm; least-squares solution; orthogonality; Kronecker product

MSC: 65F45; 65F10; 15A60; 15A69

check for updates

Citation: Tansri, K.; Chansangiam, P. Conjugate Gradient Algorithm for Least-Squares Solutions of a Generalized Sylvester-Transpose Matrix Equation. *Symmetry* **2022**, *14*, 1868. https://doi.org/10.3390/sym14091868

Academic Editor: Dongfang Li

Received: 2 August 2022
Accepted: 2 September 2022
Published: 7 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Sylvester-type matrix equations are closely related to ordinary differential equations (ODEs), which can be adapted to several problems in control engineering and information sciences; see e.g., monographs [1–3]. The Sylvester matrix equation $AX + XB = E$ and the famous special case Lyapunov equation $AX + XA^T = E$ have several applications in numerical methods for ODEs, control and system theory, signal processing, and image restoration; see e.g., [4–9]. The Sylvester-transpose equation $AX + X^TB = C$ is utilized in eigenstructure assignment in descriptor systems [10], pole assignment [3], and fault identification in dynamic systems [11]. In addition, if we require that the solution X to be symmetric, then the Sylvester-transpose equation coincides with the Sylvester one. A generalized Sylvester equation $AXB + CXD = E$ can be applied to implicit ODEs, and a general dynamical linear model for vibration and structural analysis, robot and spaceship controls; see e.g., [12,13].

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

$$AXB + CX^TD = E, \tag{1}$$

or more generally

$$\sum_{i=1}^{s} A_iXB_i + \sum_{j=1}^{t} C_jX^TD_j = E. \tag{2}$$

A direct algebraic method to find a solution of Equation (2) is to use the Kronecker linearization transforming the matrix equation into an equivalent linear system; see e.g., [14] (Ch. 4). The same technique together with the notion of weighted Moore–Penrose inverse were adapted to solve a coupled inconsistent Sylvester-type matrix equations [15] for least-squares (LS) solutions. Another algebraic method is to apply a generalized Sylvester mapping [13], so that the solution is expressed in terms of polynomial matrices. However, when the sizes of coefficient matrices are moderate or large, it is inconvenient to use matrix factorizations or another traditional methods since they require a large memory to calculate an exact solution. Thus, the Kronecker linearization and other algebraic methods are only suitable for small matrices. That is why it is important to find solutions that are easy to compute, leading many researchers to come up with algorithms that can reduce the time and memory usage of solving large matrix equations.

In the literature, there are two notable techniques to derive iterative procedures for solving linear matrix equations; see more information in a monograph [16]. The conjugate gradient (CG) technique aims to create an approximate sequence of solutions so that the respective residual matrix creates a perpendicular base. The desired solution will come out in the final step of iterations. In the last decade, many authors developed CG-type algorithms for Equation (2) and its special cases, e.g., BiCG [17], BiCR [17], CGS [18], GCD [19], GPBiCG [20], and CMRES [21]. The second technique, known as gradient-based iterative (GI) technique, intends to construct a sequence of approximate solutions from the gradient of the associated norm-error function. If we carefully set parameters of GI algorithm, then the generated sequence would converge to the desired solution. In the last five years, many GI algorithms have been introduced; see e.g., GI [22,23], relaxed GI [24], accelerated GI [25], accelerated Jacobi GI [26], modified Jacobi GI [27], gradient-descent algorithm [28], and global generalized Hessenberg algorithm [29]. For LS solutions of Sylvester-type matrix equations, there are iterative solvers, e.g., [30,31].

Recently, the work [32] developed an effective gradient-descent iterative algorithm to produce approximated LS solutions of Equation (2). When Equation (2) is consistent, a CG-type algorithm was derived to obtain a solution within finite steps; see [33]. This work is a continuation of [33], i.e., we consider Equation (1) with rectangular coefficient matrices and a rectangular unknown matrix X. Suppose that Equation (1) is inconsistent. We propose a CG-type algorithm to approximate LS solutions, which will solve the following problems:

Problem 1. *Find a matrix $\hat{X} \in \mathbb{R}^{n \times p}$ that minimizes $\|E - AXB - CX^T D\|_F$.*

Let $\mathcal{L}$ be the set of least-squares solutions of Equation (1). The second problem is to find an LS solution with the minimal norm:

Problem 2. *Find the matrix X^* such that*

$$\|X^*\|_F = \min_{\hat{X} \in \mathcal{L}} \|\hat{X}\|_F. \tag{3}$$

The last one is to find an LS solution closest to a given matrix:

Problem 3. *Let $Y \in \mathbb{R}^{n \times p}$. Find the matrix $\check{X}$ such that*

$$\|\check{X} - Y\|_F = \min_{\hat{X} \in \mathcal{L}} \|\hat{X} - Y\|_F. \tag{4}$$

Moreover, we extend our studies to the matrix Equation (2). We verify the results from theoretical and numerical points of view.

The organization of this article is as follows. In Section 2, we recall preliminary results from matrix theory that will be used in later discussions. In Section 3, we explain how the Kronecker linearization can transform Equation (1) into an equivalent linear system to obtain LS solutions. In Section 4, we propose a CG-type algorithm to solve Problem 1 and verify the theoretical capability of the algorithm. After that, Problems 2 and 3 are investigated in Sections 5 and 6, respectively. To verify the theory, we provide numerical experiments in Section 7 to show the applicability and efficiency of the algorithm, compared to the Kronecker linearization and recent iterative algorithms. We summarize the whole work in the last section.

2. Auxiliary Results from Matrix Theory

Throughout, let us denote by $\mathbb{R}^{m\times n}$ the set of all m-by-n real matrices. Recall that the standard (Frobenius) inner product of $A, B \in \mathbb{R}^{m\times n}$ is defined by

$$\langle A, B\rangle \ := \ \operatorname{tr}(B^T A) \ = \ \operatorname{tr}(AB^T). \tag{5}$$

If $\langle A, B\rangle = 0$, we say that A is orthogonal to B. A well-known property of the inner product is that

$$\langle A, BCD\rangle \ = \ \langle B^T AD^T, C\rangle, \tag{6}$$

for any matrices A, B, C, D with appropriate dimensions. The Frobenius norm of matrix $A \in \mathbb{R}^{m\times n}$ is defined by

$$\|A\|_F \ := \ \sqrt{\operatorname{tr}(A^T A)}.$$

The Kronecker product $A \otimes B$ of $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ is defined to be the mp-by-nq matrix whose each (i, j)-th block is given by $a_{ij}B$.

Lemma 1 ([14])**.** *For any real matrices A and B, we have $(A \otimes B)^T \ = \ A^T \otimes B^T$.*

The vector operator $\operatorname{Vec}(\cdot)$ transforms a matrix $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ to the vector

$$\operatorname{Vec} A \ := \ [a_{11} \cdots a_{m1}\, a_{12} \cdots a_{m2} \cdots a_{1n} \cdots a_{mn}]^T \ \in \ \mathbb{R}^{mn}.$$

The vector operator is bijective, linear, and related to the usual matrix multiplication as follows.

Lemma 2 ([14])**.** *For any $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$ and $C \in \mathbb{R}^{p\times q}$, we have*

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

For each $m, n \in \mathbb{N}$, we define a commutation matrix

$$P(m, n) \ := \ \sum_{i=1}^{m}\sum_{j=1}^{n} E_{ij} \otimes E_{ij}^T \ \in \ \mathbb{R}^{mn\times mn}, \tag{7}$$

where the (i, j)-th position of $E_{ij} \in \mathbb{R}^{m\times n}$ is 1 and all other entries are 0. Indeed, $P(m, n)$ acts on a vector by permuting its entries as follows.

Lemma 3 ([14])**.** *For any matrix $A \in \mathbb{R}^{m\times n}$, we have*

$$\operatorname{Vec}(A^T) \ = \ P(m, n) \operatorname{Vec}(A). \tag{8}$$

Moreover, commutation matrices permute the entries of $A \otimes B$ as follows.

Lemma 4 ([14])**.** *For any $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, we have*

$$B \otimes A \ = \ P(m, p)^T (A \otimes B) P(n, q). \tag{9}$$

The next result will be used in the later discussions.

Lemma 5 ([14])**.** *For any matrices A, B, C, D with appropriate dimensions, we get*

$$\operatorname{tr}(A^T D^T BC) \;=\; (\operatorname{Vec} D)^T (A \otimes B) \operatorname{Vec} C. \tag{10}$$

3. Least-Squares Solutions via the Kronecker Linearization

From now on, we investigate the generalized Sylvester-transpose matrix Equation (1), with corresponding coefficient matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{m\times p}$, $D \in \mathbb{R}^{n\times q}$, $E \in \mathbb{R}^{m\times q}$, and a rectangular unknown matrix $X \in \mathbb{R}^{n\times p}$. We focus our attention when Equation (1) is inconsistent. In this case, we will seek for its LS solution, that is, a matrix X^* that solves the following minization problem:

$$\min_{X\in\mathbb{R}^{n\times p}} \left\| E - AXB - CX^T D \right\|_F. \tag{11}$$

A traditional algebraic way to solve a linear matrix equation is known as the Kronecker linearization–to transform the matrix equation into an equivalent linear system using the notions of vector operator and Kronecker products. Indeed, taking the vector operator to Equation (1) and applying Lemmas 2 and 3 yield

$$\begin{aligned} \operatorname{Vec} E \;&=\; \operatorname{Vec}(AXB + CX^T D) \;=\; (B^T \otimes A) \operatorname{Vec} X + (D^T \otimes C) \operatorname{Vec} X^T \\ &=\; (B^T \otimes A) \operatorname{Vec} X + (D^T \otimes C) P(n,p) \operatorname{Vec} X. \end{aligned} \tag{12}$$

Let us denote $x = \operatorname{Vec} X$, $e = \operatorname{Vec} E$, and

$$M \;=\; (B^T \otimes A) + (D^T \otimes C) P(n,p). \tag{13}$$

Thus, a matrix X is an LS solution of Equation (1) if and only if x is an LS solution of the linear system $Mx = e$, or equivalently, a solution of the associated normal equation

$$M^T M x \;=\; M^T e. \tag{14}$$

The linear system (14) is always consistent, i.e., Equation (1) always has an LS solution. From the normal Equation (14) and Lemmas 2 and 3, we can deduce:

Lemma 6 ([34])**.** *Problem 1 is equivalent to the following consistent matrix equation*

$$A^T(AXB + CX^T D)B^T + D(B^T X^T A^T + D^T X C^T)C \;=\; A^T E B^T + D E^T C. \tag{15}$$

Moreover, the normal Equation (14) has a unique solution if and only if the matrix M is of full-column rank, i.e., $M^T M$ is invertible. In this case, the unique solution is given by $x^* = (M^T M)^{-1} M^T e$, and the LS error can be computed as follows:

$$\|Mx^* - e\|^2 \;=\; \|e\|^2 - e^T M x^*. \tag{16}$$

If M is of not full-column rank (i.e., the kernel of M is nontrivial), then the system $Mx = e$ has many solutions. In this case, the LS solutions appear in the form $x^* = M^\dagger e + u$ where $M^\dagger$ is the Moore–Penrose inverse of M, and u is an arbitrary vector in the kernel of M. Among all these solutions,

$$x^* \;=\; M^\dagger e \tag{17}$$

is the unique one having minimal norm.

4. Least-Squares Solution via a Conjugate Gradient Algorithm

In this section, we propose a CG-type algorithm to solve Problem 1. We do not impose any assumption on the matrix M, so that LS solutions of Equation (1) may not be unique.

We shall adapt the conjugate-gradient technique to solve the equivalent matrix Equation (15). Recall that the set of LS solutions of Equation (1) is denoted by $\mathcal{L}$. From Lemma 6, observe that the residual of a matrix $X \in \mathbb{R}^{n\times p}$ according to Equation (1) is given by

$$R_X \ := \ A^TEB^T + DE^TC - A^T(AXB + CX^TD)B^T - D(B^TX^TA^T + D^TXC^T)C. \tag{18}$$

Lemma 6 states that $X \in \mathcal{L}$ if and only if $R_X = 0$. From this, we propose the following algorithm. Indeed, the next approximate solution X_{r+1} is equal to the current approximation X_r along with a search direction U_{r+1} of suitable step size.

Algorithm 1: A conjugate gradient iterative algorithm for Equation (1)

```
$A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{m\times p}$, $D \in \mathbb{R}^{n\times q}$ and $E \in \mathbb{R}^{m\times q}$ ;
Set $r = 0$, $U_0 = 0$. Choose $X_0 \in \mathbb{R}^{n\times p}$;
$R_0 = A^TEB^T + DE^TC - A^T(AX_0B + CX_0^TD)B^T - D(B^TX_0^TA^T + D^TX_0C^T)C$;
for $r = 0$ to $np$ do
    if $\|R_r\|_F \leqslant \epsilon$ then
        $X_r$ is an LS solution; break;
    else
        if $r = 0$ then
            set $U_{r+1} = R_r$;
        else
            set $U_{r+1} = R_r + \frac{\|R_r\|_F^2}{\|R_{r-1}\|_F^2} U_r$;
        end
        $H_{r+1} = A^T(AU_{r+1}B + CU_{r+1}^TD)B^T + D(B^TU_{r+1}^TA^T + D^TU_{r+1}C^T)C$;
        $\alpha_{r+1} = \operatorname{tr}(U_{r+1}^TH_{r+1})$;
        $X_{r+1} = X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}} U_{r+1}$;
        $R_{r+1} = A^TEB^T + DE^TC - A^T(AX_{r+1}B + CX_{r+1}^TD)B^T$
                  $- D(B^TX_{r+1}^TA^T + D^TX_{r+1}C^T)C$;
        update $r$;
    end
end
```

Remark 1. *To terminate the algorithm, one can alternatively set the stopping rule to be* $\|R_r\|_F - \delta \leqslant \epsilon'$ *where* $\delta := \|Mx^* - e\|$ *is the positive square root of the LS error described in Equation (16) and* $\epsilon' > 0$ *is a small tolerance.*

For any given initial matrix X_0, we will show that Algorithm 1 generates a sequence of approximate solutions X_r of Equation (1), so that the set of residual matrices R_r is orthogonal. It follows that a unique LS solution will be obtained within finite steps.

Lemma 7. *Assume that the sequences* $\{R_r\}$ *and* $\{H_r\}$ *are generated by Algorithm 1. We get*

$$R_{r+1} \ = \ R_r - \frac{\|R_r\|_F^2}{\alpha_{r+1}} H_{r+1}, \qquad \textit{for} \quad r \ = \ 1, 2, \ldots. \tag{19}$$

Proof. From Algorithm 1, we have that for any r,

$$\begin{aligned}
R_{r+1} &= A^TEB^T + DE^TC - A^T(AX_{r+1}B + CX_{r+1}^TD)B^T - D(B^TX_{r+1}^TA^T + D^TX_{r+1}C^T)C \\
&= A^TEB^T + DE^TC - A^T\left(A(X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}}U_{r+1})B + C(X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}}U_{r+1})^TD\right)B^T \\
&\quad - D\left(B^T(X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}}U_{r+1})^TA^T + D^T(X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}}U_{r+1})C^T\right)C \\
&= A^TEB^T + DE^TC - A^T\left(AX_rB + \frac{\|R_r\|_F^2}{\alpha_{r+1}}AU_{r+1}B + CX_r^TD + \frac{\|R_r\|_F^2}{\alpha_{r+1}}CU_{r+1}^TD\right)B^T \\
&\quad - D\left(B^TX_r^TA^T + \frac{\|R_r\|_F^2}{\alpha_{r+1}}B^TU_{r+1}^TA^T + D^TX_rC^T + \frac{\|R_r\|_F^2}{\alpha_{r+1}}D^TU_{r+1}C^T\right)C \\
&= A^TEB^T + DE^TC - A^T(AX_rB + CX_r^TD)B^T - D(B^TX_r^TA^T + D^TX_rC^T)C \\
&\quad - \frac{\|R_r\|_F^2}{\alpha_{r+1}}[A^T(AU_{r+1}B + CU_{r+1}^TD)B^T + D(B^TU_{r+1}^TA^T + D^TU_{r+1}C^T)C] \\
&= R_r - \frac{\|R_r\|_F^2}{\alpha_{r+1}}H_{r+1}. \quad \square
\end{aligned}$$

Lemma 8. *The sequences $\{U_r\}$ and $\{H_r\}$ generated by Algorithm 1 satisfy*

$$\mathrm{tr}(U_m^TH_n) = \mathrm{tr}(H_m^TU_n), \qquad \textit{for any} \quad m, n. \tag{20}$$

Proof. Using the properties of the Kronecker product and the vector operator in Lemmas 1–5, we have

$$\begin{aligned}
\mathrm{tr}(H_m^TU_n) &= (\mathrm{Vec}\,H_m)^T\,\mathrm{Vec}\,U_n \\
&= [\mathrm{Vec}(A^T(AU_mB + C_jU_m^TD)B^T + D(B^TU_m^TA^T + D^TU_mC^T)C)]^T\,\mathrm{Vec}\,U_n \\
&= [\mathrm{Vec}(A^TAU_mBB^T)]^T\,\mathrm{Vec}\,U_n + [\mathrm{Vec}(A^TCU_m^TDB^T)]^T\,\mathrm{Vec}\,U_n \\
&\quad + [\mathrm{Vec}(DB^TU_m^TA^TC)]^T\,\mathrm{Vec}\,U_n + [\mathrm{Vec}(DD^TU_mC^TC)]^T\,\mathrm{Vec}\,U_n \\
&= (\mathrm{Vec}\,U_m)^T(BB^T \otimes A^TA)\,\mathrm{Vec}\,U_n + (\mathrm{Vec}\,U_m^T)^T(DB_k^T \otimes C^TA)\,\mathrm{Vec}\,U_n \\
&\quad + (\mathrm{Vec}\,U_m^T)^T(A^TC \otimes BD^T)\,\mathrm{Vec}\,U_n + (\mathrm{Vec}\,U_m)^T(C^TC \otimes DD^T)\,\mathrm{Vec}\,U_n \\
&= \mathrm{tr}(BB^TU_m^TA^TAU_n) + \mathrm{tr}(A^TCU_m^TDB^TU_n^T) + \mathrm{tr}(DB^TU_m^TA^TCU_n^T) \\
&\quad + \mathrm{tr}(C^TCU_m^TDD^TU_n) \\
&= (\mathrm{Vec}(BB_k^TU_n^TA^TA))^T\,\mathrm{Vec}\,U_m^T + (\mathrm{Vec}(C^TAU_nBD^T))^T\,\mathrm{Vec}\,U_n^T \\
&\quad + (\mathrm{Vec}(BD^TU_nC^TA))^T\,\mathrm{Vec}\,U_m^T + (\mathrm{Vec}(C^TCU_n^TDD^T))^T\,\mathrm{Vec}\,U_m^T \\
&= [\mathrm{Vec}(A^TAU_nBB^T)]^T\,\mathrm{Vec}\,U_m + [\mathrm{Vec}(DB^TU_n^TA^TC)]^T\,\mathrm{Vec}\,U_m \\
&\quad + [\mathrm{Vec}(A^TCU_n^TDB^T)]^T\,\mathrm{Vec}\,U_m + [\mathrm{Vec}(DD^TU_nC^TC)]^T\,\mathrm{Vec}\,U_m \\
&= [\mathrm{Vec}(A^T(AU_nB + CU_n^TD)B^T + D(B^TU_n^TA^T + D^TU_nC^T)C)]^T\,\mathrm{Vec}\,U_m \\
&= (\mathrm{Vec}\,H_n)^T\,\mathrm{Vec}\,U_m \\
&= \mathrm{tr}(H_n^TU_m) \\
&= \mathrm{tr}(U_m^TH_n). \quad \square
\end{aligned}$$

Lemma 9. *The sequences $\{R_r\}$, $\{U_r\}$ and $\{H_r\}$ generated by Algorithm 1 satisfy*

$$\mathrm{tr}(R_r^TR_{r-1}) = 0, \quad \textit{and} \quad \mathrm{tr}(U_{r+1}^TH_r) = 0, \qquad \textit{for any} \quad r. \tag{21}$$

Proof. We use the induction principle to prove (21). In order to calculate related terms, we utilize Lemmas 7 and 8. For $r = 1$, we get

$$\begin{aligned} \operatorname{tr}(R_1^T R_0) &= \operatorname{tr}(R_0^T R_0) - \frac{\|R_0\|_F^2}{\alpha_1} \operatorname{tr}(H_1^T R_0) = 0, \\ \operatorname{tr}(U_2^T H_1) &= \operatorname{tr}(R_1^T H_1) + \frac{\|R_1\|_F^2}{\|R_0\|_F^2} \operatorname{tr}(U_1^T H_1) \\ &= -\frac{\alpha_1}{\|R_0\|_F^2} \operatorname{tr}(R_1^T R_1) + \alpha_1 \frac{\|R_1\|_F^2}{\|R_0\|_F^2} = 0. \end{aligned}$$

These imply that (21) holds for $r = 1$. Now, we proceed the inductive step by assuming that $\operatorname{tr}(R_r^T R_{r-1}) = 0$ and $\operatorname{tr}(U_{r+1}^T H_r) = 0$. Then

$$\begin{aligned} \operatorname{tr}(R_{r+1}^T R_r) &= \operatorname{tr}(R_r^T R_r) - \frac{\|R_r\|_F^2}{\alpha_{r+1}} \operatorname{tr}\left(H_{r+1}^T \left(U_{r+1} - \frac{\|R_r\|_F^2}{\|R_{r-1}\|_F^2} U_r \right) \right) \\ &= \|R_r\|_F^2 - \frac{\|R_r\|_F^2}{\alpha_{r+1}} \operatorname{tr}(H_{r+1}^T U_{r+1}) = 0, \\ \operatorname{tr}(U_{r+2}^T H_{r+1}) &= \operatorname{tr}\left(R_{r+1}^T \left(\frac{-\alpha_{r+1}}{\|R_r\|_F^2} (R_{r+1} - R_r) \right) \right) + \frac{\|R_{r+1}\|_F^2}{\|R_r\|_F^2} \alpha_{r+1} \\ &= \frac{\alpha_{r+1}}{\|R_r\|_F^2} \operatorname{tr}[(R_{r+1}^T R_r) - (R_{r+1}^T R_{r+1})] + \frac{\|R_{r+1}\|_F^2}{\|R_r\|_F^2} \alpha_{r+1} = 0. \end{aligned}$$

Hence, Equationd (21) holds for any r. □

Lemma 10. *Suppose the sequences $\{R_r\}$, $\{U_r\}$ and $\{H_r\}$ are constructed from Algorithm 1. Then*

$$\operatorname{tr}(R_r^T R_0) = 0, \quad \operatorname{tr}(U_{r+1}^T H_1) = 0, \qquad \text{for any} \quad r. \tag{22}$$

Proof. The initial step $r = 1$ holds due to Lemma 9. Now, assume that Equation (22) is valid for all $r = 1, \ldots, k$. From Lemmas 7 and 8, we get

$$\begin{aligned} \operatorname{tr}(R_{k+1}^T R_0) &= \operatorname{tr}\left(\left(R_k - \frac{\|R_k\|_F^2}{\alpha_{k+1}} V_{k+1} \right)^T R_0 \right) = \operatorname{tr}(R_k^T R_0) - \frac{\|R_k\|_F^2}{\alpha_{k+1}} \operatorname{tr}(H_{k+1}^T R_0) \\ &= -\frac{\|R_k\|_F^2}{\alpha_{k+1}} \operatorname{tr}(H_{k+1}^T U_1) = -\frac{\|R_k\|_F^2}{\alpha_{k+1}} \operatorname{tr}(U_{k+1}^T H_1) = 0, \end{aligned}$$

and

$$\begin{aligned} \operatorname{tr}(U_{k+2}^T H_1) &= \operatorname{tr}(H_{k+2}^T U_1) = \operatorname{tr}\left(\left(\frac{-\alpha_{k+2}}{\|R_{k+1}\|_F^2} (R_{k+2} - R_{k+1}) \right)^T U_1 \right) \\ &= \frac{-\alpha_{k+2}}{\|R_{k+1}\|_F^2} [\operatorname{tr}(R_{k+2}^T R_0) - \operatorname{tr}(R_{k+1}^T R_0)] = 0. \end{aligned}$$

Hence, Equation (22) holds for any r. □

Lemma 11. *Suppose the sequences $\{R_r\}$, $\{U_r\}$ and $\{H_r\}$ are constructed from Algorithm 1. Then for any integers m and n such that $m \neq n$, we have*

$$\operatorname{tr}(R_{m-1}^T R_{n-1}) = 0, \quad \text{and} \quad \operatorname{tr}(U_m^T H_n) = 0. \tag{23}$$

Proof. According to Lemma 8 and the fact that $\mathrm{tr}(R_{m-1}^T R_{n-1}) = \mathrm{tr}(R_{n-1}^T R_{m-1})$ for any integers m and n, it remains to prove two equalities in (23) for only m, n such that $m > n$. For $m = n+1$, the two equalities hold by Lemma 9. For $m = n+2$, we have

$$\begin{aligned}
\mathrm{tr}(R_{n+2}^T R_n) &= \mathrm{tr}\left(\left(R_{n+1} - \frac{\|R_{n+1}\|_F^2}{\alpha_{n+2}} H_{n+2}\right)^T R_n\right) \\
&= -\frac{\|R_{n+1}\|_F^2}{\alpha_{n+2}} \mathrm{tr}\left(H_{n+2}^T\left(U_{n+1} - \frac{\|R_n\|_F^2}{\|R_{n-1}\|_F^2} U_n\right)\right) \\
&= \frac{\|R_{n+1}\|_F^2}{\alpha_{n+2}} \frac{\|R_n\|_F^2}{\|R_{n-1}\|_F^2} \mathrm{tr}\left(\left(R_{n+1} + \frac{\|R_{n+1}\|_F^2}{\|R_n\|_F^2} U_{n+1}\right)^T H_n\right) \\
&= \frac{\|R_{n+1}\|_F^2}{\alpha_{n+2}} \frac{\|R_n\|_F^2}{\|R_{n-1}\|_F^2} \frac{\alpha_n}{\|R_{n-1}\|_F^2} \mathrm{tr}(R_{n+1}^T R_{n-1}),
\end{aligned}$$

and, similarly, we have

$$\begin{aligned}
\mathrm{tr}(R_{n+1}^T R_{n-1}) &= \frac{\|R_n\|_F^2}{\alpha_{n+1}} \frac{\|R_{n-1}\|_F^2}{\|R_{n-2}\|_F^2} \frac{\alpha_{n-1}}{\|R_{n-2}\|_F^2} \mathrm{tr}(R_n^T R_{n-2}), \\
\mathrm{tr}(R_n^T R_{n-2}) &= \frac{\|R_{n-1}\|_F^2}{\alpha_n} \frac{\|R_{n-2}\|_F^2}{\|R_{n-3}\|_F^2} \frac{\alpha_{n-2}}{\|R_{n-3}\|_F^2} \mathrm{tr}(R_{n-1}^T R_{n-3}).
\end{aligned}$$

Moreover,

$$\begin{aligned}
\mathrm{tr}(U_{n+2}^T H_n) &= \mathrm{tr}\left(\left(R_{n+1} - \frac{\|R_{n+1}\|_F^2}{\|R_n\|_F^2} U_{n+1}\right)^T H_n\right) \\
&= \mathrm{tr}\left(R_{n+1}^T\left(\frac{-\alpha_n}{\|R_{n-1}\|_F^2}(R_n - R_{n-1})\right)\right) \\
&= \frac{\alpha_n}{\|R_{n-1}\|_F^2} \mathrm{tr}\left(\left(R_n - \frac{\|R_n\|_F^2}{\alpha_{n+1}} H_{n+1}\right)^T R_{n-1}\right) \\
&= -\frac{\alpha_n}{\|R_{n-1}\|_F^2} \frac{\|R_n\|_F^2}{\alpha_{n+1}} \left[\mathrm{tr}(H_{n+1}^T U_n) - \frac{\|R_{n-1}\|_F^2}{\|R_{n-2}\|_F^2} \mathrm{tr}(H_{n+1}^T U_{n-1})\right] \\
&= \frac{\alpha_n}{\|R_{n-1}\|_F^2} \frac{\|R_n\|_F^2}{\alpha_{n+1}} \frac{\|R_{n-1}\|_F^2}{\|R_{n-2}\|_F^2} \mathrm{tr}(U_{n+1}^T H_{n-1}),
\end{aligned}$$

and, similarly,

$$\mathrm{tr}(U_{n+1}^T H_{n-1}) = \frac{\alpha_{n-1}}{\|R_{n-2}\|_F^2} \frac{\|R_{n-1}\|_F^2}{\alpha_n} \frac{\|R_{n-2}\|_F^2}{\|R_{n-3}\|_F^2} \mathrm{tr}(U_n^T H_{n-2}).$$

In a similar way, we can write $\mathrm{tr}(R_{n+1}^T R_{n-1})$ and $\mathrm{tr}(U_{n+2}^T H_n)$ in terms of $\mathrm{tr}(R_n^T R_{n-2})$ and $\mathrm{tr}(U_{n+1}^T H_{n-1})$, respectively. Continuing this process until the terms $\mathrm{tr}(R_2^T R_0)$ and $\mathrm{tr}(U_3^T H_1)$ appear. By Lemma 10, we get $\mathrm{tr}(R_{n+1}^T R_{n-1}) = 0$ and $\mathrm{tr}(U_{n+2}^T H_n) = 0$. Similarly, we have $\mathrm{tr}(R_m^T R_{n-1}) = 0$ and $\mathrm{tr}(U_m^T H_n) = 0$ for $m = n+3, \ldots, k$.

Suppose that $\operatorname{tr}(R_{m-1}^T R_{n-1}) = \operatorname{tr}(U_m^T H_n) = 0$ for $m \in \{n+1,\ldots,k\}$. Then for $m = k+1$, we have

$$\begin{aligned}
\operatorname{tr}(R_k^T R_{n-1}) &= \operatorname{tr}(R_{k-1}^T R_{n-1}) - \frac{\|R_{k-1}\|_F^2}{\alpha_k}\operatorname{tr}(H_k^T R_{n-1}) \\
&= -\frac{\|R_{k-1}\|_F^2}{\alpha_k}\operatorname{tr}\left(H_k^T\left(U_n - \frac{\|R_{n-1}\|_F^2}{\|R_{n-2}\|_F^2}U_{n-1}\right)\right) \\
&= -\frac{\|R_{k-1}\|_F^2}{\alpha_k}\left[\operatorname{tr}(H_k^T U_n) - \frac{\|R_{n-1}\|_F^2}{\|R_{n-2}\|_F^2}\operatorname{tr}(H_k^T U_{n-1})\right] = 0.
\end{aligned}$$

and

$$\begin{aligned}
\operatorname{tr}(U_{k+1}^T H_n) &= \operatorname{tr}(R_k^T H_n) + \frac{\|R_k\|_F^2}{\|R_{k-1}\|_F^2}\operatorname{tr}(U_k^T H_n) = \operatorname{tr}\left(R_k^T\left(\frac{-\alpha_n}{\|R_{n-1}\|_F^2}(R_n - R_{n-1})\right)\right) \\
&= \frac{-\alpha_n}{\|R_{n-1}\|_F^2}\operatorname{tr}(R_k^T R_n - R_k^T R_{n-1}) = 0.
\end{aligned}$$

Hence, $\operatorname{tr}(R_{m-1}^T R_{n-1}) = 0$ and $\operatorname{tr}(U_m^T H_n) = 0$ for any m, n such that $m \neq n$. □

Theorem 1. *Algorithm 1 solves Problem 1 within finite steps. More precisely, for any given initial matrix $X_0 \in \mathbb{R}^{n\times p}$, the sequence $\{X_r\}$ constructed from Algorithm 1 converges to an LS solution of Equation (1) in at most np iterations.*

Proof. Assume that $R_r \neq 0$ for $r = 0, 1, \ldots, np-1$. Assume that $R_{np} \neq 0$. By Lemma 11, the set $\{R_0, R_1, ..., R_{np}\}$ of residual matrices is orthogonal in $\mathbb{R}^{n\times p}$ with respect to the Frobenius inner product (5). Therefore, the set $\{R_0, R_1, ..., R_{np}\}$ of $np+1$ elements is linearly independent. This contradicts the fact that the dimension of $\mathbb{R}^{n\times p}$ is np. Thus, $R_{np} = 0$, and X_{np} satisfies Equation (15) in Lemma 6. Hence X_{np} is an LS solution of Equation (1). □

We adapt the same idea as that for Algorithm 1 to derive an algorithm for Equation (2) as follows:

Algorithm 2: A conjugate gradient iterative algorithm for Equation (2)

```
$A_i \in \mathbb{R}^{m\times n}$, $B_i \in \mathbb{R}^{p\times q}$, $C_j \in \mathbb{R}^{m\times p}$, $D_j \in \mathbb{R}^{n\times q}$ for any $i = 1,2,\dots,s$, $j = 1,2,\dots,t$
 and $E \in \mathbb{R}^{m\times q}$ ;
Given $\epsilon > 0$, set $r = 0$, $U_0 = 0$. Choose $X_0 \in \mathbb{R}^{n\times p}$;
$R_0 = \sum_{k=1}^{s} A_k^T E B_k^T + \sum_{l=1}^{t} D_l E^T C_l - \sum_{k=1}^{s} A_k^T \Big( \sum_{i=1}^{s} A_i X_0 B_i + \sum_{j=1}^{t} C_j X_0^T D_j \Big) B_k^T$
      $- \sum_{l=1}^{t} D_l \Big( \sum_{i=1}^{s} B_i^T X_0^T A_i^T + \sum_{j=1}^{t} D_j^T X_0 C_j^T \Big) C_l$;
for $r = 0$ to $np$ do
    if $\|R_r\|_F \leqslant \epsilon$ then
        $X_r$ is an LS solution; break;
    else
        if $r = 0$ then
            set $U_{r+1} = R_r$;
            else
                set $U_{r+1} = R_r + \frac{\|R_r\|_F^2}{\|R_{r-1}\|_F^2} U_r$;
            end
        end
        $H_{r+1} = \sum_{k=1}^{s} A_k^T \Big( \sum_{i=1}^{s} A_i U_{r+1} B_i + \sum_{j=1}^{t} C_j U_{r+1}^T D_j \Big) B_k^T$
            $+ \sum_{l=1}^{t} D_l \Big( \sum_{i=1}^{s} B_i^T U_{r+1}^T A_i^T + \sum_{j=1}^{t} D_j^T U_{r+1} C_j^T \Big) C_l$;
        $\alpha_{r+1} = \mathrm{tr}\Big( U_{r+1}^T H_{r+1} \Big)$;
        $X_{r+1} = X_r + \frac{\|R_r\|_F^2}{\alpha_{r+1}} U_{r+1}$;
        $R_{r+1} = \sum_{k=1}^{s} A_k^T E B_k^T + \sum_{l=1}^{t} D_l E^T C_l$
            $- \sum_{k=1}^{s} A_k^T \Big( \sum_{i=1}^{s} A_i X_{r+1} B_i + \sum_{j=1}^{t} C_j X_{r+1}^T D_j \Big) B_k^T$
            $- \sum_{l=1}^{t} D_l \Big( \sum_{i=1}^{s} B_i^T X_{r+1}^T A_i^T + \sum_{j=1}^{t} D_j^T X_{r+1} C_j^T \Big) C_l$;
        update $r$;
    end
end
```

The stopping rule of Algorithm 2 may be described as $\|R_r\|_F - \delta \leqslant \epsilon'$ where δ is the positive square root of the associated LS error and $\epsilon' > 0$ is a small tolerance.

Theorem 2. *Consider Equation (2) where $A_i \in \mathbb{R}^{m\times n}$, $B_i \in \mathbb{R}^{p\times q}$, $C_j \in \mathbb{R}^{m\times p}$, $D_j \in \mathbb{R}^{n\times q}$, $D \in \mathbb{R}^{m\times q}$, $E \in \mathbb{R}^{m\times q}$ are given constant matrices and $X \in \mathbb{R}^{n\times p}$ is an unknown matrix. Assume that the matrix*

$$M := \sum_{i=1}^{s} (B_i^T \otimes A_i) + \sum_{j=1}^{t} (D_j^T \otimes C_j) P(n,p) \tag{24}$$

is of full-column rank. Then, for any given initial matrix $X_0 \in \mathbb{R}^{n\times p}$, the sequence $\{X_r\}$ constructed from Algorithm 2 converges to a unique LS solution.

Proof. The proof of is similar to that of Theorem 1. □

5. Minimal-Norm Least-Squares Solution via Algorithm 1

In this section, we investigate Problem 2. That is, we consider the case when the matrix M may not have full-column rank, so that Equation (1) may have many LS solutions. We shall seek for an element of $\mathcal{L}$ with the minimal Frobenius norm.

Lemma 12. *Assume $\hat{X} \in \mathcal{L}$. Then, any arbitrary element $\tilde{X} \in \mathcal{L}$ can be expressed as $\hat{X} + Z$ for some matrix $Z \in \mathbb{R}^{n\times p}$ satisfying*

$$A^T(AZB + CZ^TD)B^T + D(B^TZ^TA^T + D^TZC^T)C \;=\; 0. \tag{25}$$

Proof. Let us denote the residual of the LS solutions $\hat{X}$ and $\tilde{X}$, according to Equation (18), by $R_{\hat{X}}$ and $R_{\tilde{X}}$, respectively. We consider the different $Z := \tilde{X} - \hat{X}$. Now, we compute

$$\begin{aligned} R_{\tilde{X}} \;&=\; A^T(A(\hat{X}+Z)B + C(\hat{X}+Z)^TD)B^T + D(B^T(\hat{X}+Z)^TA^T + D^T(\hat{X}+Z)C^T)C \\ &\quad - A^TEB^T - DE^TC \\ &=\; A^T(AZB + CZ^TD)B^T + D(B^TZ^TA^T + D^TZC^T)C - R_{\hat{X}}. \end{aligned}$$

Since $\hat{X}, \tilde{X} \in \mathcal{L}$, by Lemma 6 we have $R_{\hat{X}} = R_{\tilde{X}} = 0$. It follows that Equation (25) holds as desired. □

Theorem 3. *Algorithm 1 solves Problem 2 in at most np iterations by starting with the initial matrix*

$$X_0 \;=\; A^T(AV_0B + CV_0^TD)B^T + D(B^TV_0^TA^T + D^TV_0C^T)C, \tag{26}$$

where $V_0 \in \mathbb{R}^{n\times p}$ is an arbitrary matrix, or especially $X_0 = 0$.

Proof. If we run Algorithm 1 starting with (26), then we can write the solution X^* of Problem 2 so that

$$X^* \;=\; A^T(AV^*B + CV^{*T}D)B^T + D(B^TV^{*T}A^T + D^TV^*C^T)C,$$

for some matrix $V^* \in \mathbb{R}^{n\times p}$. Now, assume that $\tilde{X}$ is an arbitrary element in $\mathcal{L}$. By Lemma 12, there is a matrix $Z \in \mathbb{R}^{n\times p}$ such that $\tilde{X} = X^* + Z$ and

$$A^T(AZB + CZ^TD)B^T + D(B^TZ^TA^T + D^TZC^T)C \;=\; 0.$$

Using the property (6), we get

$$\begin{aligned} \langle X^*, Z\rangle \;&=\; \left\langle A^T(AV^*B + CV^{*T}D)B^T + D(B^TV^{*T}A^T + D^TV^*C^T)C, Z\right\rangle \\ &=\; \left\langle V^*, A^T(AZB + CZ^TD)B^T + D(B^TZ^TA^T + D^TZC^T)C\right\rangle \\ &=\; 0. \end{aligned}$$

Since X^* is orthogonal to Z, it follows from the Pythagorean theorem that

$$\|\tilde{X}\|_F^2 \;=\; \|X^* + Z\|_F^2 \;=\; \|X^*\|_F^2 + \|Z\|_F^2 \;\geq\; \|X^*\|_F^2.$$

This implies that X^* is the minimal-norm solution. □

Theorem 4. *Consider the sequence $\{X_r\}$ generated by Algorithm 2 starting with the initial matrix*

$$X_0 = \sum_{k=1}^{s} A_k^T\left(\sum_{i=1}^{s} A_iV_0B_i + \sum_{j=1}^{t} C_jV_0^TD_j\right)B_k^T + \sum_{l=1}^{t} D_l\left(\sum_{i=1}^{s} B_i^TV_0^TA_i^T + \sum_{j=1}^{t} D_j^TV_0C_j^T\right)C_l,$$

where $V_0 \in \mathbb{R}^{n\times p}$ is an arbitrary matrix, or especially $X_0 = 0 \in \mathbb{R}^{n\times p}$. Then the sequence $\{X_r\}$ converges to the minimal-norm LS solution of Equatiom (2) in at most np iterations.

Proof. The proof is similar to that of Theorem 3. □

6. Least-Squares Solution Closest to a Given Matrix

In this section, we investigate Problem 3. In this case, Equation (1) may have many LS solutions. We shall seek for one that closest to a given matrix with respect to the Frobenius norm.

Theorem 5. *Algorithm 1 solves Problem 3 by substituting E with $E_1 = E - (AYB + CY^TD)$, and choosing the initial matrix to be*

$$W_0 \;=\; A^T(AVB + CV^TD)B^T + D(B^TV^TA^T + D^TVC^T)C, \tag{27}$$

where $V \in \mathbb{R}^{n\times p}$ is arbitrary, or specially $W_0 = 0 \in \mathbb{R}^{n\times p}$.

Proof. Let $Y \in \mathbb{R}^{n\times p}$ be given. We can translate Problem 3 into Problem 2 as follows:

$$\begin{aligned}
&\min_{X\in\mathbb{R}^{n\times p}} \|AXB + CX^TD - E\|_F \\
&\quad= \min_{X\in\mathbb{R}^{n\times p}} \|AXB + CX^TD - E - AYB - CY^TD + AYB + CY^TD\|_F \\
&\quad= \min_{X\in\mathbb{R}^{n\times p}} \|A(X-Y)B + C(X-Y)^TD - E + AYB + CY^TD\|_F.
\end{aligned}$$

Now, substituting $E_1 = E - (AYB + CY^TD)$ and $W = X - Y$, we see that the solution $\check{X}$ of Problem 3 is equal to $W^* + Y$ where W^* is the minimal-norm LS solution of the equation

$$AWB + CW^TD \;=\; E_1,$$

in unknown W. By Theorem 3, the matrix W^* can be solved by Algorithm 1 with the initial matrix (27) where $V \in \mathbb{R}^{n\times p}$ is arbitrary matrix, or especially $W_0 = 0$. □

Theorem 6. *Suppose that the matrix Equation (2) is inconsistent. Let $Y \in \mathbb{R}^{n\times p}$ be given. Consider Algorithm 2 when we replace the matrix E by*

$$E_1 \;=\; E - \sum_{i=1}^{s} A_iYB_i - \sum_{j=1}^{t} C_jY^TD_j,$$

and choose the initial matrix

$$W_0 \;=\; \sum_{k=1}^{s} A_k^T\left(\sum_{i=1}^{s} A_iFB_i + \sum_{j=1}^{t} C_jF^TD_j\right)B_k^T + \sum_{l=1}^{t} D_l\left(\sum_{i=1}^{s} B_i^TF^TA_i^T + \sum_{j=1}^{t} D_j^TFC_j^T\right)C_l,$$

where $F \in \mathbb{R}^{n\times p}$ is arbitrary, or $W_0 = 0 \in \mathbb{R}^{n\times p}$. Then, the sequence $\{X_r\}$ obtained by Algorithm 2 converges to the LS solution of (2) closest to Y within np iterations.

Proof. The proof of the theorem is similar to that of Theorem 5. □

7. Numerical Experiments

In this section, we provide numerical results to show the efficiency and effectiveness of Algorithm 2 (denoted by CG), which is an extension of Algorithm 1. We perform experiments when the coefficients in a given matrix equation are dense/sparse rectangular matrices of moderate/large sizes. We denote by ones(m,n) the m-by-n matrix whose all entries are 1. Each random matrix rand(m,n) has all entries belonging to the interval $(0,1)$. Each experiment contains some comparisons of Algorithm 2 with the direct Kronecker linearization as well as well-known iterative algorithms. All iterations were performed by

MATLAB R2021a, on Mac operating system (M1 chip 8C CPU/8C GPU/8GB/512GB). The performance of algorithms is investigated through the number of iterations, the norm of residual matrices, and the CPU time. The latter is measured in seconds using the functions *tic* and *toc* on MATLAB.

The next is an example of Problem 1.

Example 1. *Consider a generalized Sylvester-transpose matrix equation*

$$A_1XB_1 + A_2XB_2 + A_3XB_3 + C_1X^TD_1 + C_2X^TD_2 \;=\; E, \tag{28}$$

where the coefficient matrices are given by

$$\begin{aligned}
&A_1 = 0.5\,\mathrm{ones}(m,n) - \mathrm{rand}(m,n) \in \mathbb{R}^{50\,\times\,50}, && A_2 = 0.5\,\mathrm{ones}(m,n) - \mathrm{rand}(m,n) \in \mathbb{R}^{50\,\times\,50},\\
&A_3 = 0.5\,\mathrm{ones}(m,n) - \mathrm{rand}(m,n) \in \mathbb{R}^{50\,\times\,50}, && B_1 = 0.5\,\mathrm{ones}(p,q) - \mathrm{rand}(p,q) \in \mathbb{R}^{40\,\times\,50},\\
&B_2 = 0.5\,\mathrm{ones}(p,q) - \mathrm{rand}(p,q) \in \mathbb{R}^{40\,\times\,50}, && B_3 = 0.5\,\mathrm{ones}(p,q) - \mathrm{rand}(p,q) \in \mathbb{R}^{40\,\times\,50},\\
&C_1 = 0.5\,\mathrm{ones}(m,p) - \mathrm{rand}(m,p) \in \mathbb{R}^{50\,\times\,40}, && C_2 = 0.5\,\mathrm{ones}(m,p) - \mathrm{rand}(m,p) \in \mathbb{R}^{50\,\times\,40},\\
&D_1 = 0.5\,\mathrm{ones}(n,q) - \mathrm{rand}(n,q) \in \mathbb{R}^{50\,\times\,50}, && D_2 = 0.5\,\mathrm{ones}(n,q) - \mathrm{rand}(n,q) \in \mathbb{R}^{50\,\times\,50},\\
&E = 0.5\,\mathrm{ones}(m,q) - \mathrm{rand}(m,q) \in \mathbb{R}^{50\,\times\,50}.
\end{aligned}$$

In fact, we have $\mathrm{rank}\,M = 2000 \neq 2001 = \mathrm{rank}\,[M \;\; e]$, *i.e., the matrix equation does not have an exact solution. However, M is of full-column rank, so this equation has a unique LS solution. We run Algorithm 2 using an initial matrix $X_0 = 0 \in \mathbb{R}^{50\times40}$ and a tolerance error $\epsilon = \|Mx^* - e\| = 6.4812$, where $x^* = (M^TM)^{-1}M^Te$. It turns out that Algorithm 2 takes* 20 *iterations to get a least-square solution, consuming around* 0.2 *s, while the direct method consumes around* 7 *s. Thus, Algorithm 2 takes* 35 *times less computational time than the direct method. We compare the performance of Algorithm 2 with other well-known iterative algorithms: GI method [31], LSI method [31], and TAUOpt method [32]. The numerical results are shown in Table 1 and Figure 1. We see that after running* 20 *iterations, Algorithm 2 consumes CTs slightly more than other methods, but the relative error $\|R_r\|_F$ is less than those of the others. Hence, Algorithm 2 is applicable and has a good performance.*

Table 1. Relative error and computational time for Example 1.

Method	Iterations	CPU	$\|R_r\|_F$
CG	20	0.199308	6.407766
GI	20	0.129715	10.907665
LSI	20	0.179449	14.390460
TAUOpt	20	0.073866	7.806273
Direct	–	7.048632	0

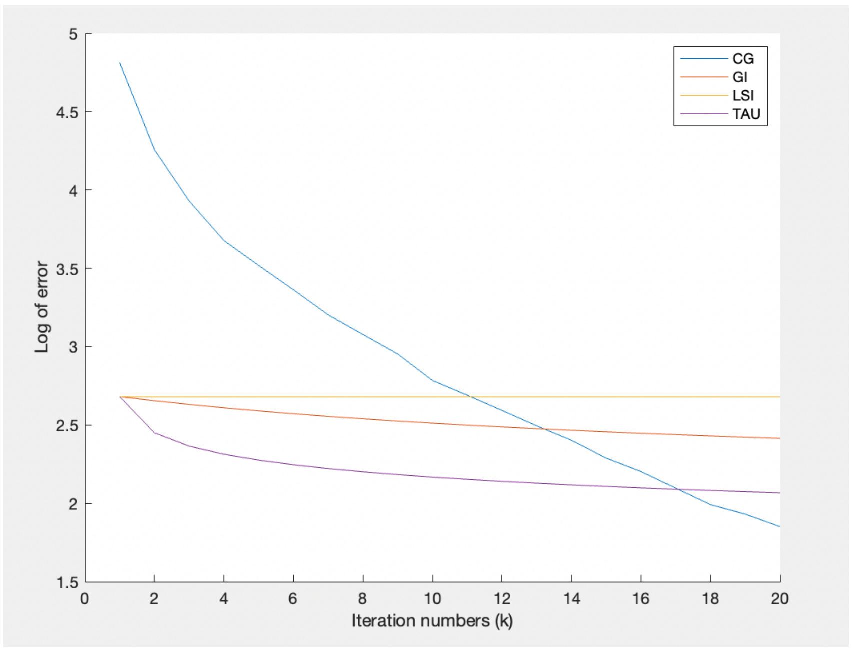

Figure 1. The logarithm of the relative error $\|R_r\|_F$ for Example 1.

The next is an example of Problem 2.

Example 2. *Consider a generalized Sylvester-transpose matrix equation*

$$A_1XB_1 + C_1X^TD_1 + C_2X^TD_2 \;=\; E, \tag{29}$$

where

$$\begin{aligned} &A_1 = -0.08 \times \text{ones}(m,n) \in \mathbb{R}^{30\,\times\,25}, && B_1 = \text{tridiag}(0.11, -0.61, -0.29) \in \mathbb{R}^{30\,\times\,30},\\ &C_1 = \text{tridiag}(-0.03, -0.22, -0.1) \in \mathbb{R}^{30\,\times\,30}, && C_2 = \text{tridiag}(0.38, 0.29, -0.41) \in \mathbb{R}^{30\,\times\,30},\\ &D_1 = -0.13 \times \text{ones}(n,q) \in \mathbb{R}^{25\,\times\,30}, && D_2 = 0.04 \times \text{ones}(n,q) \in \mathbb{R}^{25\,\times\,30},\\ &E = -0.01 \times I_{30} \in \mathbb{R}^{30\,\times\,30}. \end{aligned}$$

In this case, Equation (29) *is inconsistent and the associated matrix M is not of full-column rank. Thus, Equation* (29) *has many LS solutions. The direct method concerning Moore–Penrose inverse* (17) *takes* 0.627019 *s to get the minimal-norm LS solutions. Alternatively, MNLS method [35] can be also used to this kind of problem. However, some coefficient matrices are triangular matrices with multiple zeros, causing the MNLS algorithm diverges and cannot provide answer. Therefore, let us apply Algorithm 2 using a tolerance error $\epsilon = 10^{-5}$. According to Theorem 4, we choose three different matrices V_0 to generate the initial matrix X_0. The numerical results are shown in Table 2 and Figure 2.*

Table 2. Relative error and computational time for Example 2.

V_0	Iterations	CPU	$\|\|R_r\|\|_F$
0	6	0.036523	0.000008
0.02 × ones	11	0.036540	0.000009
$-0.01\ \times I$	10	0.038425	0.000009

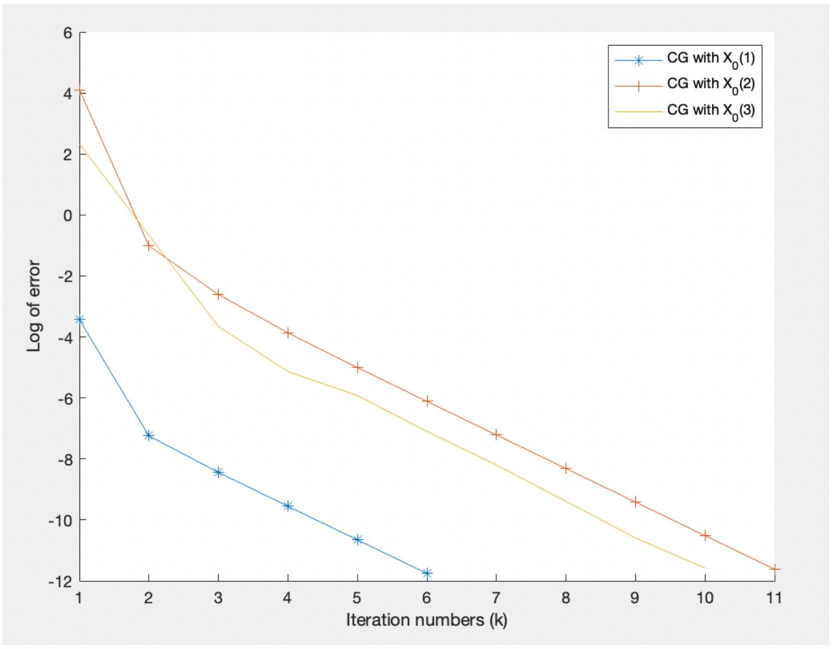

Figure 2. The logarithm of the relative error for Example 2.

Figure 2 shows that the logarithm of the relative errors $\|R_r\|_F$ for CG algorithms, using three initial matrices, are rapidly decreasing to zero. All of them consume around 0.037 s to arrive at the desired solution, which is 16 times less than the direct method.

The following is an example of Problem 3.

Example 3. *Consider a generalized Sylvester-transpose matrix equation*

$$A_1XB_1 + C_1X^TD_1 + C_2X^TD_2 = E, \tag{30}$$

where

$$\begin{aligned} &A_1 = 0.2 \times \text{ones}(m,n) \in \mathbb{R}^{50\,\times\,40}, && B_1 = \text{tridiag}(-0.2, 0.3, 0.3) \in \mathbb{R}^{50\,\times\,50}, \\ &C_1 = \text{tridiag}(0.4, -0.2, -0.1) \in \mathbb{R}^{50\,\times\,50}, && C_2 = \text{tridiag}(0.7, -0.2, 0.3) \in \mathbb{R}^{50\,\times\,50}, \\ &D_1 = -0.2 \times \text{ones}(n,q) \in \mathbb{R}^{40\,\times\,50}, && D_2 = 0.1 \times \text{ones}(n,q) \in \mathbb{R}^{40\,\times\,50}, \\ &E = I_{50} \in \mathbb{R}^{50\times 50}. \end{aligned}$$

In fact, Equation (30) is inconsistent and has many LS solutions. The first task is to find the LS solution of Equation (30) closest to $Y = 0.1 \times \text{ones}(n,p) \in \mathbb{R}^{40\times 50}$. According to Theorem 6, we apply Algorithm 2 with two different matrices V to construct the initial matrix W_0. Algorithm 2 with $V = 0$ and $V = -0.19 \times I$ are denoted in Figure 3 by CG_1 and CG_2, respectively.

The second task is to solve Problem 3 when we are given $Y = 0.1 \times \text{ones}(n,p) \in \mathbb{R}^{40\times 50}$. Similarly, we use two different matrices $V = 0$ and $V = 0.02 \times \text{ones}(n,p) \in \mathbb{R}^{40\times 50}$ to construct the initial matrix.

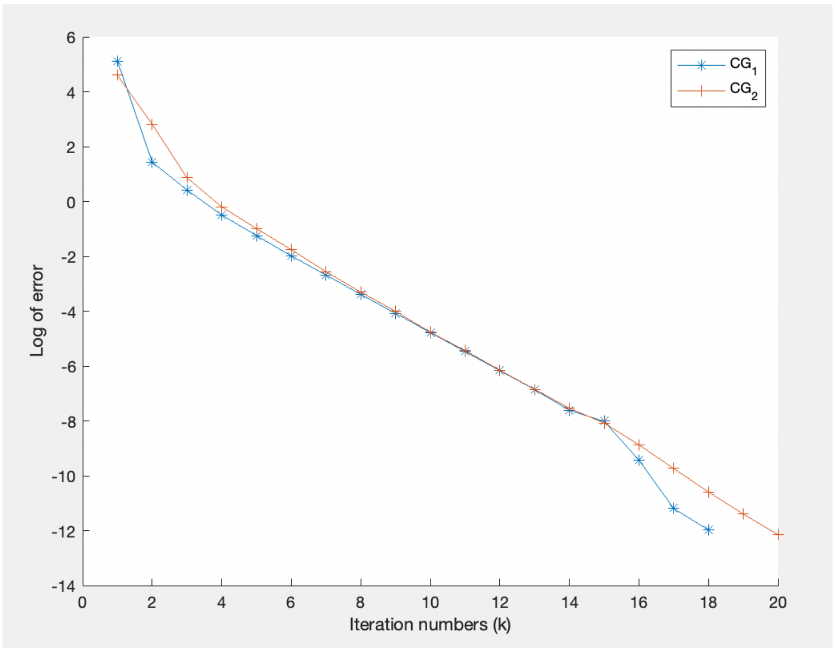

Figure 3. The logarithm of the relative error for Example 3 with Y = 0.1 $\times$ ones(n, p).

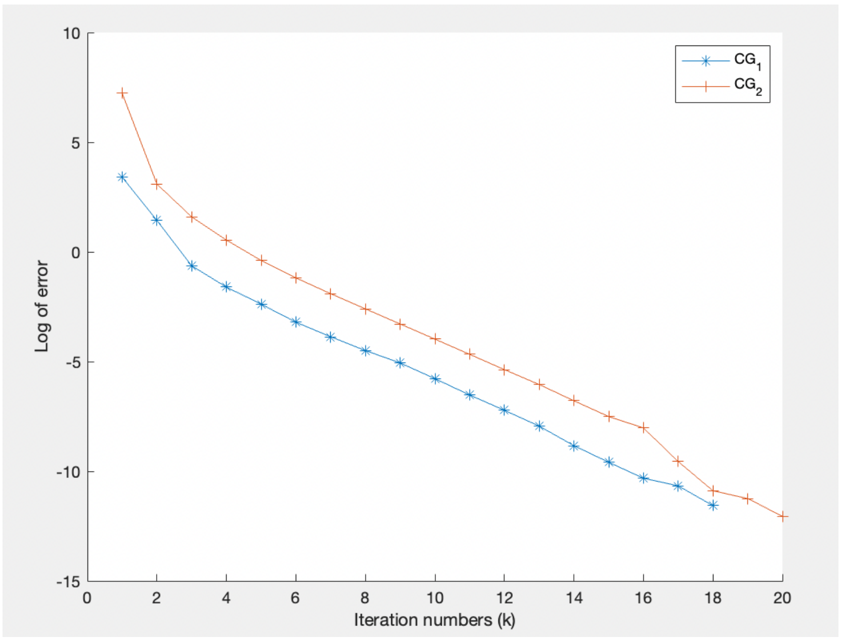

Figure 4. The logarithm of the relative error for Example 3 with Y = I.

Table 3. Relative error and computational time for Example 3.

Y	Initial V	Iterations	CPU	$\|\|R_r\|\|_F$	$\|\|X^* - Y\|\|_F$
$0.1 \times \text{ones}(n, p)$	0	18	0.104135	0.000006	4.3116
	$-0.19 \times I$	20	0.108153	0.000005	4.3116
I	0	18	0.113960	0.000009	0.8580
	$0.02 \times$ ones	20	0.108499	0.000006	0.8580

We apply Algorithm 2 with a tolerance error $\epsilon = 10^{-5}$. The numerical results in Figures 3 and 4, and Table 3 illustrate that, in each case, the relative error converges rapidly to zero within 20 iterations, consuming around 0.1 s. Thus, Algorithm 2 performs well in both the number of iterations and computational time. Moreover, changing initial matrix and the desired matrix Y does not siginificantly affect the performance of algorithm.

8. Conclusions

We propose CG-type iterative algorithms, namely, Algorithms 1 and 2, to generate approximate solutions for the generalized Sylvester-transpose matrix Equations (1) and (2), respectively. When the matrix equation is inconsistent, the algorithm will arrive at an LS solution within np iterations with the absence of round-off errors. When the matrix equation has many LS solutions, the algorithm can search for the one with minimal Frobenius norm within np steps. Moreover, given a matrix Y, the algorithm can find the LS solution closest to Y within np steps. The numerical simulations validate the relevance of the algorithm for medium/large sizes of squares/non-squares matrices. The algorithm is always applicable for any given initial matrix and the given matrix Y. The algorithm performs well in both the number of iterations and computational times, compared to the direct Kronecker linearization and well-known iterative methods.

Author Contributions: Writing—original draft preparation, K.T.; writing—review and editing, P.C.; data curation, K.T.; supervision, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research project is supported by National Research Council of Thailand (NRCT): (N41A640234).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare there is no conflict of interest.

References

1. Geir, E.D.; Fernando, P. *A Course in Robust Control Theory: A Convex Approach*; Springer: New York, NY, USA, 1999.
2. Lewis, F. A survey of linear singular systems. *Circ. Syst. Signal Process.* **1986**, *5*, 3–36. [CrossRef]
3. Dai, L. *Singular Control Systems*; Springer: Berlin, Germany, 1989.
4. Enright, W.H. Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations. *ACM Trans. Math. Softw.* **1978**, *4*, 127–136. [CrossRef]
5. Aliev, F.A.; Larin, V.B. *Optimization of Linear Control Systems: Analytical Methods and Computational Algorithms*; Stability Control Theory, Methods Applications; CRC Press: Boca Raton, FL, USA, 1998
6. Calvetti, D.; Reichel, L. Application of ADI iterative methods to the restoration of noisy images. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 165–186. [CrossRef]
7. Duan, G.R. Eigenstructure assignment in descriptor systems via output feedback: A new complete parametric approach. *Int. J. Control* **1999**, *72*, 345–364. [CrossRef]
8. Duan, G.R. Parametric approaches for eigenstructure assignment in high-order linear systems. *Int. J. Control Autom. Syst.* **2005**, *3*, 419–429.
9. Kim, Y.; Kim, H.S. Eigenstructure assignment algorithm for second order systems. *J. Guid. Control Dyn.* **1999**, *22*, 729–731. [CrossRef]
10. Fletcher, L.R.; Kuatsky, J.; Nichols, N.K. Eigenstructure assignment in descriptor systems. *IEEE Trans. Autom. Control* **1986**, *31*, 1138–1141. [CrossRef]
11. Frank, P.M. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy a survey and some new results. *Automatica* **1990**, *26*, 459–474. [CrossRef]
12. 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.* **1980**, *20*, 341–345. [CrossRef]
13. Zhou, B.; Duan, G.R. On the generalized Sylvester mapping and matrix equations. *Syst. Control Lett.* **2008**, *57*, 200–208. [CrossRef]
14. Horn, R.; Johnson, C. *Topics in Matrix Analysis*; Cambridge University Press: New York, NY, USA, 1991.
15. Kilicman, A.; Al Zhour, Z.A. Vector least-squares solutions for coupled singular matrix equations. *Comput. Appl. Math.* **2007**, *206*, 1051–1069. [CrossRef]
16. Simoncini, V. Computational methods for linear matrix equations. *SIAM Rev.* **2016**, *58*, 377–441. [CrossRef]
17. Hajarian, M. Developing BiCG and BiCR methods to solve generalized Sylvester-transpose matrix equations. *Int. J. Autom. Comput.* **2014**, *11*, 25–29. [CrossRef]
18. Hajarian, M. Matrix form of the CGS method for solving general coupled matrix equations. *Appl. Math. Lett.* **2014**, *34*, 37–42. [CrossRef]

19. Hajarian, M. Generalized conjugate direction algorithm for solving the general coupled matrix equations over symmetric matrices. *Numer. Algorithms* **2016**, *73*, 591–609. [CrossRef]
20. Dehghan, M.; Mohammadi-Arani, R. Generalized product-type methods based on Bi-conjugate gradient (GPBiCG) for solving shifted linear systems. *Comput. Appl. Math.* **2017**, *36*, 1591–1606. [CrossRef]
21. Zadeh, N.A.; Tajaddini, A.; Wu, G. Weighted and deflated global GMRES algorithms for solving large Sylvester matrix equations. *Numer. Algorithms* **2019**, *82*, 155–181. [CrossRef]
22. Kittisopaporn, A.; Chansangiam, P.; Lewkeeratiyukul, W. Convergence analysis of gradient-based iterative algorithms for a class of rectangular Sylvester matrix equation based on Banach contraction principle. *Adv. Differ. Equ.* **2021**, *2021*, 17. [CrossRef]
23. Boonruangkan, N.; Chansangiam, P. Convergence analysis of a gradient iterative algorithm with optimal convergence factor for a generalized Sylvester-transpose matrix equation. *AIMS Math.* **2021**, *6*, 8477–8496. [CrossRef]
24. 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**, *2017*. [CrossRef]
25. Xie, Y.J.; Ma, C.F. The accelerated gradient based iterative algorithm for solving a class of generalized Sylvester transpose matrix equation. *Appl. Math. Comp.* **2016**, *273*, 1257–1269. [CrossRef]
26. Tian, Z.; Tian, M.; Gu, C.; Hao, X. An accelerated Jacobi-gradient based iterative algorithm for solving Sylvester matrix equations. *Filomat* **2017**, *31*, 2381–2390. [CrossRef]
27. Sasaki, N.; Chansangiam, P. Modified Jacobi-gradient iterative method for generalized Sylvester matrix equation. *Symmetry* **2020**, *12*, 1831. [CrossRef]
28. Kittisopaporn, A.; Chansangiam, P. Gradient-descent iterative algorithm for solving a class of linear matrix equations with applications to heat and Poisson equations. *Adv. Differ. Equ.* **2020**, *2020*, 324. [CrossRef]
29. Heyouni, M.; Saberi-Movahed, F.; Tajaddini, A. On global Hessenberg based methods for solving Sylvester matrix equations. *Comp. Math. Appl.* **2018**, *2019*, 77–92. [CrossRef]
30. Hajarian, M. Extending the CGLS algorithm for least squares solutions of the generalized Sylvester-transpose matrix equations. *J. Franklin Inst.* **2016**, *353*, 1168–1185. [CrossRef]
31. Xie, L.; Ding, J.; Ding, F. Gradient based iterative solutions for general linear matrix equations. *Comput. Math. Appl.* **2009**, *58*, 1441–1448. [CrossRef]
32. Kittisopaporn, A.; Chansangiam, P. Approximated least-squares solutions of a generalized Sylvester-transpose matrix equation via gradient-descent iterative algorithm. *Adv. Differ. Equ.* **2021**, *2021*, 266. [CrossRef]
33. Tansri, K.; Choomklang, S.; Chansangiam, P. Conjugate gradient algorithm for consistent generalized Sylvester-transpose matrix equations. *AIMS Math.* **2022**, *7*, 5386–5407. [CrossRef]
34. Wang, M.; Cheng, X. Iterative algorithms for solving the matrix equation $AXB + CX^TD = E$. *Appl. Math. Comput.* **2007**, *187*, 622–629. [CrossRef]
35. Chen, X.; Ji, J. The minimum-norm least-squares solution of a linear system and symmetric rank-one updates. *Electron. J. Linear Algebra* **2011**, *22*, 480–489. [CrossRef]