Research Article

Numerical Solution of Sylvester Equation Based on Iterative Predictor-Corrector Method

Ovgu Cidar Iyikal

Department of Mathematics, Faculty of Arts and Sciences, Eastern Mediterranean University, North Cyprus, Via Mersin 10, Famagusta, Turkey

Correspondence should be addressed to Ovgu Cidar Iyikal; ovgu.cidar@emu.edu.tr

Received 4 May 2022; Accepted 23 July 2022; Published 22 August 2022

Academic Editor: Qingkai Zhao

Copyright © 2022 Ovgu Cidar Iyikal. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The inspiration of the study concerns an iterative predictor-corrector method with order of convergence \( p = 45 \) for computing the inverse of the coefficient matrix \( \Lambda = \left( I_m \otimes A \right) + \left( B^T \otimes I_m \right) \), which is obtained by the Sylvester equation \( AX + XB = C \). The numerical solutions of three examples by predictor-corrector algorithm are given. The final numerical results also support the applicability, fast convergency, and high accuracy of the method for finding matrix inverses.

1. Introduction

Consider the Sylvester matrix equations of the form \( AX + XB = C \) where \( A \in \mathbb{R}^{m \times m} \), \( B \in \mathbb{R}^{n \times n} \), \( C \in \mathbb{R}^{m \times n} \) are the constant coefficient matrices and \( X \in \mathbb{R}^{m \times n} \) is the unknown solution matrix. These equations play important roles in matrix decompositions of the eigenvalues, in control theory, in reduction models, in numerical solutions of the matrix differential Riccati equations, and in image processing (see [1–6]).

Solution of Sylvester matrix equations follows the transformation to the linear form \( \Lambda \text{col}(X) = \text{col}(C) \), where \( \Lambda = \left( I_m \otimes A \right) + \left( B^T \otimes I_m \right) \) is an \( mn \times mn \) matrix [7]. So, higher-order dimensions of \( \Lambda \) can create problems. These problems can be avoided with iterative algorithms for finding matrix inverses. An iterative optimization-based method on partial swarm theory to solve Sylvester equation was given in [8]. Iterative gradient-based algorithm for solving the equations by minimizing specific criterion functions was presented in [7]. A preconditioned iterative gradient-based method is obtained by selection of two auxiliary matrices by using the Newton method, and it can be acceptable as a generalization of the iterative splitting method to solve system of linear equations in [9]. Later, a preconditioned positive-definite/skew-Hermitian iteration method of splitting for continuous Sylvester equations with positive-definite/semi-definite matrices was presented in [10]. The implicit iteration alternating direction for solving the continuous Sylvester equation, where the coefficient matrices are taking as positive semi-definite matrices, and at least one of them to be positive definite is given in [11]. The implicit iteration alternating direction method for solving the continuous Sylvester equation, when the coefficient matrices are taking as positive semidefinite matrices (at least one of them to be positive definite) is given in [12]. A method based on the sign function iteration for solving large-scale Sylvester equation was presented in [13]. A transformation method such as Hessenberg–Schur algorithm was also used to solve Sylvester equation by reducing the coefficient matrices \( A \) and \( B \) to triangular form in [14]. A numerical Arnoldi-based method for solving Sylvester equation when \( A \) is sparse and large for partial pole-assignment problem for large matrices was given in [15]. A method consisting of orthogonal reduction of the coefficient matrix \( A \) of the Sylvester matrix equation to a block-upper-Hessenberg form was given in [16].

Other than these methods, direct methods generally take time and require a lot of storage. Special interest is to use an iterative method for computing the Moore–Penrose inverse \( \Lambda^+ \in \mathbb{C}^{n_2 \times n_2} \) of the matrix \( \Lambda \in \mathbb{C}^{n_1 \times n_2} \). This method is derived by the Schulz method of 2nd order. A \( p^{th} \) order iterative method for computing the Moore–Penrose inverse was
studied in [17, 18] for the orders \( p \geq 2 \), and it is known as the hyperpower iterative method. Let \( \Lambda \in \mathbb{C}^{n \times n} \), and we surveyed on iterative methods by which \( V_k \) presents the approximate Moore–Penrose inverse of \( \Lambda \) at the \( k \)th iteration. It is usually presented as

\[
V_{k+1} = V_k \sum_{j=0}^{p-1} T_k^j, \quad k = 0, 1, \ldots ,
\]

for \( n_1 \leq n_2 \). \( T_k = I - \Lambda V_k \) performs \( p \) matrix-by-matrix multiplications per step. There are different representations of (1) which differ in the calculation of the power sum \( \sum_{j=0}^{p-1} T_k^j \). If the nested loops are used through the factorizations, then the computational effort in the \( p \)th order method decreases and the number of matrix-by-matrix multiplications and additions in the polynomial is reduced (see [19, 20]). Most recently, in [21], for a given integer \( t \geq 1 \), two different classes of iterative methods for finding matrix inverses of square nonsingular matrices were proposed and they were used as approximate inverse preconditioners for solving linear systems. Among these classes of methods, class 1 converges with order \( p = 3 \cdot 2^t + 1 \) and class 2 converges with order \( p = 5 \cdot 2^t - 1 \), requiring \( 2t + 4 \) and \( 3t + 4 \) matrix-by-matrix multiplications per step, respectively. The method of orders \( p = 7, 11, 15, 19 \) to approximate matrix inversion was applied for approximating the Schur complement matrices. The given algorithm was applied to solve Fredholm integral equations of first kind in [22]. Another recursive approach for constructing incomplete matrix factorization of the \( M \)-matrices by iterative two-step method for the approximation of the inverse of pivoting the diagonal block-matrices at each stage of the recursion was given in [23].

In this study, an iterative predictor-corrector method of order \( p = 45 \) is used for computing the inverse of nonzero coefficient matrix \( \Lambda \in \mathbb{R}^{mn \times mn} \) of the linear form \( A \text{col}(X) = \text{col}(C) \) of the Sylvester equations. It was established in [24]. The method requires 5 matrix-by-matrix multiplications per step for the predictor step and 5 matrix-by-matrix multiplications per step for the corrector step. Then, to verify the predictor-corrector method’s fast convergence, experimental analysis is conducted for the three examples, whereas for large dimensional matrices, the computation of the Moore–Penrose inverse is costly when the direct method as singular value decomposition is used. Therefore, fast converging iterative methods for approximating the Moore–Penrose inverses that also computationally efficient are essential. The need to compute Moore-Penrose inverse for the solution of the Sylvester equation leads to one of the application area. As stated before, these equations appear in control theory and other important real-life phenomena.

### 2. Sylvester Matrix Equation

Consider the Sylvester matrix equations of the form

\[
AX + XB = C, \tag{2}
\]

where \( A \in \mathbb{R}^{mn \times mn} \), \( B \in \mathbb{R}^{mn \times mn} \), and \( C \in \mathbb{R}^{mn \times mn} \) are constant matrices and \( X \in \mathbb{R}^{mn \times mn} \) is the matrix of unknowns. Linear system (2) has a unique solution given as

\[
\text{col}[X] = [A]^{-1}\text{col}[C], \tag{3}
\]

if and only if \( \lambda_i[A] + \lambda_j[B] \neq 0 \) for any \( i \) and \( j \), where \( \otimes \) presents the Kronecker product for

\[
\Lambda = (I_n \otimes A) + (B^T \otimes I_m) \in \mathbb{R}^{mn \times mn}, \tag{4}
\]

\[
\text{col}[X] = \begin{bmatrix} x_1^T & x_2^T & \cdots & x_{mn}^T \end{bmatrix}^T, \tag{5}
\]

\[
\text{col}[C] = \begin{bmatrix} c_1^T & c_2^T & \cdots & c_{mn}^T \end{bmatrix}^T. \tag{6}
\]

In practice, if \( B^T = A \), then equation (2) becomes continuous time Lyapunov equation. For numerical solution of

\[
A \text{col}(X) = \text{col}(C), \tag{7}
\]

the following iterative predictor-corrector method is used to find inverse of \( \Lambda \) in (4) for solving (7).

### 3. The Iterative Predictor-Corrector Method for Approximating Moore–Penrose Inverse of the Matrix

Let \( \tilde{\Lambda} \in \mathbb{C}^{m \times mn} \), \( I \) denote the unit matrix, \( \tilde{\Lambda} \) denote the conjugate transpose of \( \Lambda \), \( R(\Lambda) \) denote the range of \( \Lambda \), and \( P_{R(\Lambda)} \) denote the orthogonal projection of \( R(\Lambda) \). The following matrix-valued functions are defined:

\[
\Omega(T_k) = T_k + T_k^2, \tag{8}
\]

\[
\Psi(T_k) = I + T_k^2, \tag{9}
\]

\[
\Gamma(T_k) = T_k^4, \tag{10}
\]

where \( T_k = (I - \tilde{\Lambda} V_k) \). We propose the predictor-corrector method in [24] as follows.

**Predictor-corrector method:**

\[
\text{In: } V_0 = a\tilde{\Lambda}^+, \quad \text{G: } T_k = I - \tilde{\Lambda} V_k, \Phi(T_k) = \Psi(T_k)\Omega(T_k), \quad \text{P: } V_{k+1/2} = V_k(I + \Phi(T_k)), \quad T_{k+1/2} = (I - \tilde{\Lambda} V_{k+1/2}), \tag{11}
\]

\[
\text{C: } V_{k+1} = V_{k+1/2}(I + \Phi(T_{k+1/2}))(I + \Gamma(T_{k+1/2})), \quad k = 0, 1, 2, \ldots ,
\]

where \( \text{In} \) represents the initial step, \( \text{G} \) represents the generator, \( \text{P} \) represents the predictor at the fractional step \( k + 1/2 \), and \( \text{C} \) represents the corrector for approximating the Moore–Penrose inverse \( V_{k+1} \) of \( \tilde{\Lambda} \) at the \( k + 1 \)th step in (11). The predictor-corrector method given in (11) is effective in computational aspects and is useful for \( n_1 \leq n_2 \). If \( n_1 > n_2 \), then dual version of the predictor-corrector iterative algorithm (11) can be used.
Choose an initial matrix $V_0 = a\Lambda^T$ under the assumption $0 < \alpha < 2/\lambda_1(\Lambda^T \Lambda)$ where $\lambda_1(\Lambda^T \Lambda) \geq \lambda_2(\Lambda^T \Lambda) \geq \ldots \geq \lambda_r(\Lambda^T \Lambda) > 0$.

Step 1. Let $k = 0$. Evaluate $\text{col}(X)_0 = V_0 \text{col}(C)$ and $\tau_0 = \text{col}(C) - \text{col}(X)_0$; then, calculate $\|\tau_0\|_2/\|\text{col}(C)\|_2$.

Do Step 2–Step 6 until $\|\tau_k\|_2/\|\text{col}(C)\|_2 \leq \epsilon$.

Step 2. Evaluate $T_k = I - \Lambda V_k$.

Step 3. Apply $(P)$, the predictor step of the predictor-corrector method (11), to find $V_{k+1/2}$.

Step 4. Evaluate $T_{k+1/2} = I - \Lambda V_{k+1/2}$.

Step 5. Apply $(C)$, the corrector step of the predictor-corrector method (11), to find $V_{k+1}$.

Step 6. Evaluate $\text{col}(X)_{k+1} = V_{k+1} \text{col}(C)$ and $\tau_{k+1} = \text{col}(C) - \text{col}(X)_{k+1}$, respectively. Then, calculate $\|\tau_{k+1}\|_2$.

Let $k = k + 1$.

Step 7. If $k$ is the performed iteration number, then the approximate solution is $y_k = \text{col}(X)_k$ that satisfies $\|\tau_k\|_2/\|\text{col}(C)\|_2 \leq \epsilon$.

**Algorithm 1:** Predictor-corrector algorithm

**Theorem 1** (see [24]). Let $\Lambda \in \mathbb{C}^{n \times n}$ and let the nonzero eigenvalues of $\Lambda^* \Lambda$ satisfy $\lambda_1(\Lambda^* \Lambda) \geq \lambda_2(\Lambda^* \Lambda) \geq \ldots \geq \lambda_r(\Lambda^* \Lambda) > 0$ where the real scalar $\alpha$ satisfies $0 < \alpha < 2/\lambda_1(\Lambda^* \Lambda)$. Then, the sequence $\{V_{k+1}\}$ obtained by the predictor-corrector method (11) converges to the Moore-Penrose inverse $\Lambda^*$ in the $L_2$ norm when $k \to \infty$ with order of convergence $p = 45$ and the asymptotic convergence factor is [21] ACF $= 10/\ln 45 \approx 2.627$. Moreover, the following error estimate is valid:

$$\|\Lambda^* - V_k\|_2 \leq \frac{d\|R_0\|_2}{1 - d\|R_0\|_2} \left| \frac{(\|\Lambda^*\|_2)^2}{\|\Lambda^*\|_2} \right|,$$

where degree of practicality increases in the given order and degree of precision decreases in the same order and $R_0 = P_{\Lambda^*} - a\Lambda\Lambda^*$.

**3.1. Error Analysis.** The minimum norm of the solution to the linear system:

$$\text{col}(X) = \text{col}(C), \quad \Lambda \in \mathbb{R}^{m \times m}, \quad \text{col}(C) \in \mathbb{R}^{m \times n},$$

where $\text{col}(C) \in R(\Lambda)$ with $R(\Lambda)$ denotes the range of $\Lambda$, $\text{col}(X) \in R^{m \times n}$ denotes the unknown solution, and rank$(\Lambda) = n$. The general least squares solution of (13) or minimum norm least squares solution is the solution of the problem

$$\min_{X \in \mathbb{C}^{n \times m}} \|\text{col}(X)\|_2, \quad S = \{\text{col}(X) \in \mathbb{R}^{m \times n} | \|\text{col}(X) - \text{col}(C)\|_2 = \min\}.$$

System (14) has a unique solution obtained by the equation $\text{col}(X) = \Lambda^* \text{col}(C)$, and it is called the pseudoinverse solution [25].

If $\Lambda$ is nonsingular matrix, then $\Lambda^* = \Lambda^{-1}$. Moreover, the condition number of $\Lambda$ is defined by $\text{cond}(\Lambda) = \|\Lambda\|_2 \|\Lambda^*\|_2$.

**3.2. Algorithm for Approximating Inverse of Matrix $\Lambda$ to Solve Sylvester Equation.** Let $\text{col}(X)_{k+1} = V_{k+1} \text{col}(C)$ be the approximate solution of (14), where $V_{k+1}$ is the inverse of $\Lambda$ obtained by the predictor-corrector method given in (11) at the $k + 1$-th iteration by using initial approximation $V_0 = \alpha\Lambda^T$, and $\alpha$ satisfies $0 < \alpha < 2/\lambda_1(\Lambda^T \Lambda)$ where $\lambda_1(\Lambda^T \Lambda) \geq \lambda_2(\Lambda^T \Lambda) \geq \ldots \geq \lambda_r(\Lambda^T \Lambda) > 0$. Therefore, the residual error at the corresponding step is $\tau_{k+1} = \text{col}(C) - \text{col}(X)_{k+1}$. Given a prespecified accuracy $\epsilon > 0$, we propose the following algorithm based on Algorithm A in [24] that uses the predictor-corrector method (11) and approximates the pseudoinverse for $\Lambda$ (See Algorithm 1).

**4. Numerical Examples**

In this section, all the calculations are carried out by Mathematica program in double precision for Examples 1–3. The tables adopt the following notation:

- TCP: total CPU time in seconds.

**Example 1.** Consider the following coefficient matrices for Sylvester equation (2) (see [7]):

$$A = \begin{bmatrix} 1 & 1 \\ 2 & -4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 3 & 10 \\ -12 & -8 \end{bmatrix},$$

with exact solution

$$X = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \end{bmatrix}.$$

Here, $X_0 = \text{ones}(2,2) \times 10^{-6}$ (ones$(2,2)$ represent the $2 \times 2$ matrix of ones) and apply the predictor-corrector method to find the inverse of $\Lambda$ in (4).

It is obtained as

$$\Lambda = \begin{bmatrix} 2 & -1 & 0 \\ 2 & 3 & 0 \\ 1 & 0 & 2 \\ 1 & -2 & -3 \end{bmatrix},$$

and then, compute approximate solution of the Sylvester equation (2). In this example, the right side vector $C$ is calculated by $AX + XB$. We apply the proposed predictor-corrector algorithm to obtain the inverse of $\Lambda$, by performing $k$ iterations for an accuracy of $\epsilon \leq 1 \times 10^{-6}$ for the corresponding Sylvester matrix equation (7). Table 1 presents $L_2$ norm of the errors using the predictor-corrector algorithm and the condition numbers.
Example 3. Consider the following randomly generated coefficient matrices $A, B, C \in \mathbb{R}^{6 \times 6}$ [16] to solve Sylvester equation (2) via (7) by predictor-corrector algorithm with stopping criteria $\epsilon = 1 \times 10^{-15}$:

$$a_{11} = 0.846221417824,$$
$$a_{12} = 0.202647357650,$$
$$a_{13} = 0.891298966149,$$
$$a_{21} = 0.525152496305,$$
$$a_{22} = 0.672137468474,$$
$$a_{23} = 0.762096833027,$$
$$a_{31} = 0.921812970745,$$
$$a_{32} = 0.176266144495,$$
$$a_{33} = 0.93546969108,$$
$$a_{41} = 0.738207245811,$$
$$a_{42} = 0.405706213062,$$
$$a_{43} = 0.916904439913,$$
$$a_{51} = 0.138890881957,$$
$$a_{52} = 0.198721742661,$$
$$a_{53} = 0.272187924970,$$
$$a_{61} = 0.20276521856,$$
$$a_{62} = 0.603792479194,$$
$$a_{63} = 0.198814267761,$$
$$a_{64} = 0.456467651618,$$
$$a_{65} = 0.821407164295,$$
$$a_{66} = 0.615432348100,$$
$$a_{67} = 0.018503643248,$$
$$a_{25} = 0.444703364353,$$
$$a_{26} = 0.791937037427,$$
$$a_{44} = 0.410270206991,$$
$$a_{35} = 0.057891304784,$$
$$a_{46} = 0.813166497304,$$
$$a_{44} = 0.893649530914,$$
$$a_{45} = 0.352868132217,$$
$$a_{46} = 0.009861300661.

Table 1: $L_2$ norm of the errors obtained by predictor-corrector algorithm and condition numbers for Example 1.

| $\alpha$ | $\frac{\|r_1\|_2}{\|\text{col}(C)\|_2}$ | $\frac{\|\text{col}(X) - V_1\text{col}(C)\|_2}{\|\text{col}(X)\|_2}$ | $\text{cond}(A)$ | $\text{cond}(A)$ |
|----------|---------------------------------|---------------------------------|-----------------|-----------------|
| 0.1      | 4.893274 × 10^{-5}              | 5.814136 × 10^{-5}              | 5.000000        | 3.365853        |
| 0.5      | 6.30926 × 10^{-4}               | 8.14136 × 10^{-5}              | 10.000000       | 3.365853        |
| 1        | 8.93274 × 10^{-5}               | 10.000000                      | 12.000000       | 3.365853        |
| 5        | 10.000000                       | 16.000000                      | 16.000000       | 3.365853        |

Figure 1: Error versus iteration number $k$ for Example 2 when $(p, t) = (10, 10)$.

The best error of the approximate solution for Example 1 obtained in [7] was $2.5974 \times 10^{-4}$ after $k = 15$ iterations. When we compare results in Table 1 with the literature [7], the predictor-corrector method is a more accurate, fast, and effective method.

Example 2. Sylvester equation (2) with the following coefficient matrices is considered:

$$A = \text{triu}(\text{rand}(p, p), 1) + \text{diag}(\alpha + \text{diag} (\text{rand}(p)))$$
$$B = \text{triu}(\text{rand}(t, t), 1) + \text{diag}(\alpha + \text{diag} (\text{rand}(p)))$$

where triu represents a tridiagonal matrix, rand represents random numbers, diag represents the diagonal matrix, and $\alpha$ is used for the weights of the diagonal entries [9]. For Example 2, the exact solution is

$$X = \text{rand}(p, t) + \text{speye}(p, t) \times 2,$$

with $X_0 = \text{ones}(p, t) \times 10^{-6}$, where speye represents a sparse identity matrix and $\alpha = 2$. Stopping criterion is $\epsilon = 1 \times 10^{-9}$.

Table 2 presents TCP, total iteration number $k$, and $L_2$ norm errors for given values of the dimensions $p$ and $t$.

| $\alpha$ | $\epsilon$ | TCP | $k$ | $L_2$ norm |
|----------|-------------|-----|-----|------------|
| 0.1      | 10^{-15}    | 6   | 15  | 2.5974 × 10^{-4} |
| 0.5      | 10^{-15}    | 6   | 15  | 2.5974 × 10^{-4} |
| 1        | 10^{-15}    | 6   | 15  | 2.5974 × 10^{-4} |
| 5        | 10^{-15}    | 6   | 15  | 2.5974 × 10^{-4} |

According to Figure 1, fast and accurate result can be obtained when $\alpha$ increases.
\[ a_{54} = 0.015273927029, \]
\[ a_{55} = 0.445096432288, \]
\[ a_{56} = 0.465994341675, \]
\[ a_{64} = 0.746785676564, \]
\[ a_{65} = 0.93181458462, \]
\[ a_{66} = 0.418649467728. \]
\[ b_{11} = 0.546571151829, \]
\[ b_{12} = 0.522590349081, \]
\[ b_{13} = 0.875741899818, \]
\[ b_{21} = 0.444880204673, \]
\[ b_{22} = 0.880142207411, \]
\[ b_{23} = 0.737305988466, \]
\[ b_{31} = 0.694567240426, \]
\[ b_{32} = 0.172956141275, \]
\[ b_{33} = 0.136518742260, \]
\[ b_{41} = 0.621310130795, \]
\[ b_{42} = 0.979746896789, \]
\[ b_{43} = 0.011756687353, \]
\[ b_{51} = 0.794821080201, \]
\[ b_{52} = 0.271447258642, \]
\[ b_{53} = 0.893897966445, \]
\[ b_{61} = 0.956843448445, \]
\[ b_{62} = 0.252329346874, \]

Figure 2: Error versus iteration number \( k \) for Example 3.

| \( p, t \) | TCP | \( k \) | \( \| \tau_k \|_2 / \| \text{col}(C) \|_2 \) | \( \text{cond}(A) \) |
|---|---|---|---|---|
| 5, 3 | 0.01 | 2 | \( 4.57222 \times 10^{-17} \) | 2.36803 |
| 10, 6 | 0.17 | 2 | \( 3.05786 \times 10^{-16} \) | 3.81110 |
| 15, 10 | 2.25 | 2 | \( 4.70474 \times 10^{-16} \) | 4.03430 |
| 20, 15 | 17.79 | 2 | \( 2.54129 \times 10^{-10} \) | 12.9925 |
| 50, 30 | 2167.15 | 3 | \( 8.37652 \times 10^{-16} \) | 40.3054 |

Table 3: \( L_2 \) norm of the errors obtained by predictor-corrector algorithm for Example 3.

| \( \| \tau_5 \|_2 / \| \text{col}(C) \|_2 \) | \( \| \tau_5 \|_2 \) | \( \text{cond}(A) \) | \( \text{cond}(\Lambda) \) |
|---|---|---|---|
| \( 4.23589 \times 10^{-15} \) | \( 1.50175 \times 10^{-14} \) | 70.80 | 1051.51 |

\[ b_{63} = 0.199138067206, \]
\[ b_{14} = 0.298723012102, \]
\[ b_{15} = 0.582791681561, \]
\[ b_{16} = 0.579806873250, \]
\[ b_{24} = 0.661442576382, \]
\[ b_{25} = 0.423496256851, \]
\[ b_{26} = 0.760365009804, \]
\[ b_{34} = 0.284408589750, \]
\[ b_{35} = 0.515511752141, \]
\[ b_{36} = 0.529823116717, \]
\[ b_{44} = 0.469224285211, \]
\[ b_{45} = 0.333951479972, \]
\[ b_{46} = 0.640526498990, \]
\[ b_{54} = 0.064781122963, \]
\[ b_{55} = 0.432906596107, \]
\[ b_{56} = 0.209069404438, \]
\[ b_{64} = 0.988334938278, \]
\[ b_{65} = 0.225949868144, \]
\[ b_{66} = 0.379818370351. \]
\[ c_{12} = 0.709471392703, \]
\[ c_{11} = 0.838118445052, \]
\[c_{13} = -0.52465110045,\]
\[c_{21} = 0.019639513865,\]
\[c_{22} = 0.428892365341,\]
\[c_{23} = -0.157705064793,\]
\[c_{31} = 0.681277161282,\]
\[c_{32} = 0.304617366869,\]
\[c_{33} = -0.25907521287,\]
\[c_{41} = 0.379481018028,\]
\[c_{42} = 0.189653747547,\]
\[c_{43} = 0.045079530894,\]
\[c_{51} = 0.831796017610,\]
\[c_{52} = 0.19341156405,\]
\[c_{53} = -0.195440134055,\]
\[c_{61} = 0.502812883996,\]
\[c_{62} = 0.682223223591,\]
\[c_{63} = 0.128597436672,\]
\[c_{14} = 0.883859755846,\]
\[c_{15} = 0.92062135311,\]
\[c_{16} = 0.337873490541,\]
\[c_{24} = 0.906034170972,\]
\[c_{25} = 0.486903569691,\]
\[c_{26} = 0.30731827939,\]
\[c_{34} = 0.356617663467,\]
\[c_{35} = 0.936266830386,\]
\[c_{36} = 0.135805928663,\]
\[c_{44} = 0.742419230344,\]
\[c_{45} = 0.836601295625,\]
\[c_{46} = 0.670950124615,\]
\[c_{54} = 0.592369870448,\]
\[c_{55} = 0.589358440912,\]
\[c_{56} = 0.57915246589,\]
\[c_{64} = 0.975132462127,\]
\[c_{65} = 0.918056131072,\]
\[c_{66} = 0.352654884696.\]

Figure 2 illustrates the error \(\|r_k\|_2/\|\text{col}(C)\|_2\) for the test problems in Example 3, and Table 3 presents \(L_2\) norm of the errors using the predictor-corrector algorithm and the condition numbers. The accuracy of the predictor-corrector algorithm is remarkable.

5. Concluding Remarks

An iterative predictor-corrector method of convergence order \(p = 45\) is used for computing the inverse of a nonzero matrix \(\Lambda \in \mathbb{R}^{mn \times mn}\) in (4) to solve Sylvester equation (2). It converges fast, and it is a highly accurate method. It can also be useful when \(\Lambda\) is a rectangular matrix or an ill-conditioned matrix. The predictor-corrector iterative method may also be applied to precondition the coefficient matrices of the linear algebraic system of equations obtained by using finite difference method to solve boundary value problems, for example, for the solution of Laplace’s equation with singularities (see [26, 27]), for the solution of the heat equation on hexagonal grid (see [28]), and for the approximation of the derivatives of the solution of the heat equation (see [29, 30]). Moreover, the iterative method of predictor-corrector for finding matrix inverses may be applied to precondition the coefficient matrices of the system of equations obtained by using finite element method to solve parabolic partial differential equations (see [31]).

Data Availability

No data were used in this study.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] Z. Z. Bai, X. X. Guo, and S. F. Xu, “Alternately linearized implicit iteration methods for the minimal nonnegative solutions of the nonsymmetricalalgebraic riccati equations,” Numerical Linear Algebra with Applications, vol. 13, pp. 655–674, 2006.

[2] Z. Z. Bai, “On Hermitian and skew-hermitian splitting iteration methods for continuous Sylvester equations,” Journal of Computational Mathematics, vol. 29, pp. 185–198, 2011.

[3] P. Benner, R. C. Li, and N. Truhar, “On the ADI method for Sylvester equations,” Journal of Computational and Applied Mathematics, vol. 233, no. 4, pp. 1035–1045, 2009.
[4] R. Bhatia and P. Rosenthal, “How and why to solve the operator equation $AX - XB = Y$,” Bulletin of the London Mathematical Society, vol. 29, pp. 1–21, 1997.

[5] G. H. Golub and V. Loan CF, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

[6] E. L. Wachspress, “Trail to a lyapunov equation solver,” Computers and Mathematics with Applications, vol. 55, no. 8, pp. 1653–1659, 2008.

[7] F. Ding and T. Chen, “Gradient based iterative algorithms for solving a class of matrix equations,” IEEE Transactions on Automatic Control, vol. 50, no. 8, pp. 1216–1221, 2005.

[8] S. S. Asari and M. Amirfakhrian, “Numerical solution of Sylvester matrix equations: application to dynamic systems,” Journal of Interpolation and Approximation in Scientific Computing, vol. 2016, no. 1, 13 pages, Article ID 00097, 2016.

[9] J. Zhou, R. Wang, and Q. Niu, “A preconditioned iteration method for solving Sylvester equations,” Journal of Applied Mathematics, vol. 2012, Article ID 401059, 1 page, 2012.

[10] R. Zhou, X. Wang, and X. B. Tang, “Preconditioned positive-definite and skew-hermitian splitting iteration methods for continuous Sylvester equations $AX + XB = C$,” East Asian Journal on Applied Mathematics, vol. 7, no. 1, pp. 55–69, 2017.

[11] Z. Liu, Y. Zhou, and Y. Zhang, “On inexact alternating direction implicit iteration for continuous Sylvester equations,” Numerical Linear Algebra with Applications, vol. 27, no. 5, pp. 1–13, 2020.

[12] F. Ding and T. Chen, “On iterative solutions of general coupled matrix equations,” SIAM Journal on Control and Optimization, vol. 44, no. 6, pp. 2269–2284, 2006.

[13] U. Baur, “Low rank solution of data-sparse Sylvester equations,” Numerical Linear Algebra with Applications, vol. 15, no. 9, pp. 837–851, 2008.

[14] G. Golub, S. Nash, and C. Van Loan, “A hessenberg-schur method for the problem $AX + XB = C$,” IEEE Transactions on Automatic Control, vol. 24, no. 6, pp. 909–913, 1979.

[15] B. N. Datta and Y. Saad, “Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment,” Linear Algebra and its Applications, vol. 154–156, pp. 225–244, 1991.

[16] M. A. Ramadan, N. M. El-Shazly, and B. I. Selim, “A hessenberg method for the numerical solutions to types of block Sylvester matrix equations,” Mathematical and Computer Modelling, vol. 52, no. 9–10, pp. 1716–1727, 2010.

[17] A. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, PA, USA, 1996.

[18] S. C. Buranay, D. Subasi, and O. C. Iyikal, “On the two classes of high order convergent methods of approximate inverse preconditioners for solving linear systems,” Numerical Linear Algebra with Applications, vol. 24, no. 6, 2017.

[19] S. C. Buranay and O. C. Iyikal, “Approximate Schur-block ILU preconditioners for regularized solution of discrete ill-posed problems,” Mathematical Problems in Engineering, vol. 2019, Article ID 912535, 18 pages, 2019.