Research Article

LSMR Iterative Method for General Coupled Matrix Equations

F. Toutounian,1,2 D. Khojasteh Salkuyeh,3 and M. Mojarrab1

1 Department of Applied Mathematics, School of Mathematical Sciences, Ferdowsi University of Mashhad, Iran
2 The Center of Excellence on Modelling and Control Systems, Ferdowsi University of Mashhad, Iran
3 Faculty of Mathematical Sciences, University of Guilan, Iran

Correspondence should be addressed to M. Mojarrab; maryam.modjarrab@gmail.com

Received 18 March 2014; Accepted 28 July 2014

Academic Editor: D. R. Sahu

Copyright © F. Toutounian et al. 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.

By extending the idea of LSMR method, we present an iterative method to solve the general coupled matrix equations

$$\sum_{k=1}^{q} A_{ik} X_k B_{ik} = C_i, \quad i = 1, 2, \ldots, p,$$

including the generalized (coupled) Lyapunov and Sylvester matrix equations as special cases over some constrained matrix groups \((X_1, X_2, \ldots, X_q)\), such as symmetric, generalized bisymmetric, and \((R, S)\)-symmetric matrix groups. By this iterative method, for any initial matrix group \((X_1^{(0)}, X_2^{(0)}, \ldots, X_q^{(0)})\), a solution group \((X_1^*, X_2^*, \ldots, X_q^*)\) can be obtained within finite iteration steps in absence of round-off errors, and the minimum Frobenius norm solution or the minimum Frobenius norm least-squares solution group can be derived when an appropriate initial iterative matrix group is chosen. In addition, the optimal approximation solution group to a given matrix group \((X_1, X_2, \ldots, X_q)\) in the Frobenius norm can be obtained by finding the least Frobenius norm solution group of new general coupled matrix equations. Finally, numerical examples are given to illustrate the effectiveness of the presented method.

1. Introduction

In control and system theory [1–7], we often encounter Lyapunov and Sylvester matrix equations which have been playing a fundamental role. Due to the important roles of the matrix equations, the studies on the matrix equations have been addressed in a large body of papers [8–27]. By using the hierarchical identification principle [9–11, 28–32], a gradient-based iterative (GI) method was presented to solve the solutions and the least-squares solutions of the general coupled matrix equations. In [19, 33], Zhou et al. deduced the optimal parameter of the GI method for solving the solutions and the weighted least-squares solutions of the general coupled matrix equations. Dehghan and Hajarian [34–36] introduced several iterative methods to solve various linear matrix equations.

In [12, 17], Huang et al. presented finite iterative algorithms for solving generalized coupled Sylvester systems. Li and Huang [37] proposed a matrix LSQR iterative method to solve the constrained solutions of the generalized coupled Sylvester matrix equations. Hajarian [38] presented the generalized QMRCGSTAB algorithm for solving Sylvester-transpose matrix equations. Recently, Lin and Simoncini [39] established minimal residual methods for large scale Lyapunov equations. They explored the numerical solution of this class of linear matrix equations when a minimal residual (MR) condition is used during the projection step.

In this paper, we construct a matrix iterative method based on the LSMR algorithm [40] to solve the constrained solutions of the following problems.

Compatible matrix equations are as follows

$$\sum_{k=1}^{q} A_{ik} X_k B_{ik} = C_i, \quad i = 1, 2, \ldots, p.$$
Least-squares problem is as follows:

$$\min \left( \sum_{i=1}^{p} \sum_{k=1}^{q} A_{ik} X_k B_{ik} - C_{ij} \right)^{1/2}.$$  \hfill (2)

Matrix nearness problem is as follows:

$$\min \left\{ \sum_{k=1}^{q} \|X_k - X_i\|^2 \right\},$$ \hfill (3)

where $A_{ik}, B_{ik}, C_{ij}, i = 1, 2, \ldots, p; k = 1, 2, \ldots, q$, are constant matrices with suitable dimensions, $X_k, k = 1, 2, \ldots, q$, are unknown matrices to be solved, $X_k, k = 1, 2, \ldots, q$, are given matrices, and $S_P$ is the solution set of (1) or problem (2).

This paper is organized as follows. In Section 2, we will briefly review some fundamental properties of the LSMR algorithm \cite{40}, which is an iterative method for computing a solution $x$ to either of the following problems.

Compatible linear systems are as follows:

$$Mx = f.$$ \hfill (5)

Least-squares problem is as follows:

$$\min \|Mx - f\|_2.$$ \hfill (6)

where $M \in \mathbb{R}^{m \times n}$ and $f \in \mathbb{R}^m$. The LSMR algorithm uses an algorithm of Golub and Kahan \cite{44}, which states as procedure Bidiag 4, to reduce $M$ to lower bidiagonal form. The procedure Bidiag 4 can be described as follows.

**Bidiag 4** (starting vector $f$; reduction to lower bidiagonal form).

$$\beta_i u_1 = f, \quad \alpha_i v_1 = M^T u_1, \quad \beta_{i+1} u_{i+1} = M v_i - \alpha_i u_i, \quad \alpha_{i+1} v_{i+1} = M^T u_{i+1} - \beta_{i+1} v_i,$$ \hfill (7)

$i = 1, 2, \ldots$.

The scalers $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen such that $\|u_1\|_2 = \|v_1\|_2 = 1$.

The following properties presented in \cite{7} illustrate that the procedure Bidiag 4 has finite termination property.

**Property 1.** Suppose that $k$ steps of the procedure Bidiag 4 have been taken; then the vectors $v_1, v_2, \ldots, v_k$ and $u_1, u_2, \ldots, u_k, u_{k+1}$ are the orthonormal basis of the Krylov subspaces $K_k(M^T M, v_i)$ and $K_{k+1}(M M^T, u_j)$, respectively.

**Property 2.** The procedure Bidiag 4 will stop at step $m$ if and only if $\min \{|\mu|, |\lambda|\} = m$, where $\mu$ is the grade of $v_1$ with respect to $M^T M$ and $\lambda$ is the grade of $u_1$ with respect to $M M^T$.

By using the procedure Bidiag 4, the LSMR method constructs an approximation solution of the form $x_k = V_k y_k$, where $V_k = (v_1, v_2, \ldots, v_k)$, which solves the least-squares problem, $\min \|M^T r_k\|_2$, where $r_k = f - M x_k$ is the residual for the approximate solution $x_k$. The main steps of the LSMR algorithm can be summarized as shown in Algorithm 1.

More details about the LSMR algorithm can be found in \cite{40}.

The stopping criterion may be used as $\|M^T r_k\| = \|M^T (f - M x_k)\|_2 = \|M^T (f - M x_k)\|_b = \|[\tilde{K}]_{k+1}\|$ for the compatible linear systems (5) and for the least-squares problem (6). Other stopping criteria can also be used and are not listed here. Reader can see \cite{40} for details. Clearly, the sequence $x_k \in \text{Range}(M^T)$ generated by the LSMR algorithm converges to the unique minimum norm solution of (5) or the unique minimum norm least-squares solution of problem (6).
3. A Matrix LSMR Iterative Method

In this section, we will present our matrix iterative method based on the LSMR algorithm, for solving (1) and problem (2). For the unknown matrices $X_i \in \mathbb{R}^{n \times n}, i = 1, 2, \ldots, q$, by using the Kronecker product, (1) and problem (2) are equivalent to (5) and problem (6), respectively, with

$$M = \begin{pmatrix}
B_{11}^T \otimes A_{11} & B_{12}^T \otimes A_{12} & \cdots & B_{1q}^T \otimes A_{1q} \\
B_{21}^T \otimes A_{21} & B_{22}^T \otimes A_{22} & \cdots & B_{2q}^T \otimes A_{2q} \\
\vdots & \vdots & & \vdots \\
B_{p1}^T \otimes A_{p1} & B_{p2}^T \otimes A_{p2} & \cdots & B_{pq}^T \otimes A_{pq}
\end{pmatrix},$$

$$x = \begin{pmatrix}
\text{vec}(X_1) \\
\text{vec}(X_2) \\
\vdots \\
\text{vec}(X_q)
\end{pmatrix}, \quad f = \begin{pmatrix}
\text{vec}(C_1) \\
\text{vec}(C_2) \\
\vdots \\
\text{vec}(C_p)
\end{pmatrix}. \quad (8)$$

Hence, by using the invariance of the Frobenius norm under unitary transformations, it is easy to prove that the vector form $\beta_i u_i = f, \alpha_i v_i = M^T u_i, \alpha_i, \gamma_i = \alpha_i \beta_i, \rho_i = 1, \overline{p}_0 = 1, \overline{c}_0 = 1, \overline{z}_0 = 0, h_i = v_i, \overline{r}_0 = 0, x_0 = 0$

For $k = 1, 2, \ldots$, until convergence Do:

$$\beta_{k+1} u_{k+1} = M v_k - \alpha_k u_k$$
$$\alpha_k v_k = M^T u_{k+1} - \beta_{k+1} v_k$$
$$\rho_k = \left(\overline{c}_k^2 + \beta_{k+1}^2 \right)^{1/2}$$
$$s_k = \beta_{k+1} / \rho_k$$
$$\theta_{k+1} = s_k \alpha_{k+1}$$
$$\overline{c}_{k+1} = c_k \alpha_{k+1}$$
$$\overline{r}_k = \overline{c}_{k+1} / \rho_k$$
$$\beta_k = \left(\overline{c}_{k-1}^2 + \theta_{k+1}^2 \right)^{1/2}$$
$$\xi_k = \overline{c}_{k-1} \rho_k$$
$$s_k = \xi_k / \rho_k$$
$$\theta_k = \xi_k / \rho_k$$
$$\overline{r}_k = h_k - \overline{c}_{k-1} / \rho_k$$
$$x_k = x_{k+1} + \left(\xi_k / \rho_k \right) h_k$$
$$h_{k+1} = v_k - \left(\theta_k / \rho_k \right) h_k$$

If $|\overline{r}_{k+1}|$ is small enough then stop

End Do.

Algorithm 1: LSMR algorithm.

$$\alpha_i v_i^{(j)} = \sum_{i=1}^{p} A_{ij}^T U_i^{(j)} B_{ji}^{T}, \quad j = 1, 2, \ldots, q,$$

$$\alpha_1 = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A_{ij}^T U_i^{(j)} B_{ji}^{T} \right) \right)^{1/2},$$

$$\beta_i v_i^{(j)} = \left( \sum_{k=1}^{q} A_{jk} V_i^{(k)} B_{jk} - \alpha_i U_i^{(j)} \right), \quad j = 1, 2, \ldots, p,$$

$$\beta_{i+1} = \left( \sum_{j=1}^{p} \left( \sum_{k=1}^{q} A_{jk} V_i^{(k)} B_{jk} - \alpha_i U_i^{(j)} \right) \right)^{1/2},$$

$$\alpha_{i+1} v_i^{(k)} = \sum_{j=1}^{p} A_{jk} V_i^{(j)} B_{jk} - \beta_{i+1} V_i^{(k)}, \quad k = 1, 2, \ldots, q,$$

$$\alpha_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{j=1}^{p} A_{jk} V_i^{(j)} B_{jk} - \beta_{i+1} V_i^{(k)} \right) \right)^{1/2}. \quad (9)$$
\[
M = 
\begin{pmatrix}
B_{11}^T \otimes A_{11} & B_{12}^T \otimes A_{12} & \cdots & B_{1q}^T \otimes A_{1q} \\
B_{21}^T \otimes A_{11} & B_{22}^T \otimes A_{22} & \cdots & B_{2q}^T \otimes A_{2q} \\
\vdots & \vdots & \ddots & \vdots \\
B_{p1}^T \otimes A_{p1} & B_{p2}^T \otimes A_{p2} & \cdots & B_{pq}^T \otimes A_{pq} \\
A_{11} \otimes B_{11}^T & A_{12} \otimes B_{12}^T & \cdots & A_{1q} \otimes B_{1q}^T \\
A_{21} \otimes B_{21}^T & A_{22} \otimes B_{22}^T & \cdots & A_{2q} \otimes B_{2q}^T \\
\vdots & \vdots & \ddots & \vdots \\
A_{p1} \otimes B_{p1}^T & A_{p2} \otimes B_{p2}^T & \cdots & A_{pq} \otimes B_{pq}^T \\
(B_{11}^T P_1) \otimes (A_{11} P_1) & (B_{12}^T P_1) \otimes (A_{12} P_1) & \cdots & (B_{1q}^T P_1) \otimes (A_{1q} P_1) \\
(B_{21}^T P_1) \otimes (A_{21} P_1) & (B_{22}^T P_1) \otimes (A_{22} P_1) & \cdots & (B_{2q}^T P_1) \otimes (A_{2q} P_1) \\
\vdots & \vdots & \ddots & \vdots \\
(B_{p1}^T P_1) \otimes (A_{p1} P_1) & (B_{p2}^T P_1) \otimes (A_{p2} P_1) & \cdots & (B_{pq}^T P_1) \otimes (A_{pq} P_1) \\
(A_{11} P_1) \otimes (B_{11}^T P_1) & (A_{12} P_1) \otimes (B_{12}^T P_1) & \cdots & (A_{1q} P_1) \otimes (B_{1q}^T P_1) \\
(A_{21} P_1) \otimes (B_{21}^T P_1) & (A_{22} P_1) \otimes (B_{22}^T P_1) & \cdots & (A_{2q} P_1) \otimes (B_{2q}^T P_1) \\
\vdots & \vdots & \ddots & \vdots \\
(A_{p1} P_1) \otimes (B_{p1}^T P_1) & (A_{p2} P_1) \otimes (B_{p2}^T P_1) & \cdots & (A_{pq} P_1) \otimes (B_{pq}^T P_1) 
\end{pmatrix}
\]

\[
x = \begin{pmatrix}
\text{vec}(X_1) \\
\text{vec}(X_2) \\
\vdots \\
\text{vec}(X_q)
\end{pmatrix}, \quad f = \begin{pmatrix}
\text{vec}(C_1) \\
\text{vec}(C_2) \\
\vdots \\
\text{vec}(C_p) \\
\text{vec}(C_T^1) \\
\text{vec}(C_T^2) \\
\vdots \\
\text{vec}(C_T^q)
\end{pmatrix},
\]

(10)

where \(P_1, P_2, \ldots, P_q \in SOR^{\text{vec}}\). Hence, the vector forms of \(\beta_i u_i = f\), \(\alpha_{i1} v_1 = M^T u_1\), \(\beta_{i+1} u_{i+1} = M v_i - \alpha_i u_i\), and \(\alpha_{i+1} v_{i+1} = M^T u_{i+1} - \beta_i v_i\) \((i = 1, 2, \ldots)\) in LSMR algorithm can be rewritten in matrix forms, respectively, as

\[
\beta_1 U_1^{(j)} = C_j, \quad j = 1, 2, \ldots, p,
\]

\[
\alpha_{1j} V_1^{(j)} = \sum_{i=1}^{p} (A_{ij}^T U_1^{(i)}) B_{ij} + B_{ij} U_1^{(0) T} A_{ij} + (P_j A_{ij}^T) U_1^{(j)} (B_{ij}^T P_j) + (P_j B_{ij}) U_1^{(0) T} (A_{ij} P_j), \quad j = 1, 2, \ldots, q.
\]
\[
\alpha_1 = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} U^{(i)}_{ik} B^T_{ik} + B^T_{ik} U^{(i)T}_{ik} A_{ik} \right) + (P_k A^T_{ik}) U^{(i)}_{ik} (B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{ik} (A_{ik} P_k) \right)^{1/2},
\]

\[
\beta_{i+1} U^{(i)}_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} V^T_{ik} B_{jk} \right) - \alpha_i U^{(i)}_i \right), \quad j = 1, 2, \ldots, p,
\]

\[
\beta_{i+1} = \left( \sum_{j=1}^{p} \left( \sum_{k=1}^{q} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right)^{1/2},
\]

\[
\alpha_{i+1} V^{(k)}_{i+1} = \sum_{j=1}^{p} \left( A^T_{ik} U^{(i)}_{ik} B^T_{ik} + B^T_{ik} U^{(i)T}_{ik} A_{ik} + (P_k A^T_{ik}) \right) x(U^{(i)}_{i+1}, B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{i+1} (A_{ik} P_k)
\]

\[
\alpha_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} U^{(i)}_{ik} B^T_{ik} + B^T_{ik} U^{(i)T}_{ik} A_{ik} \right) + (P_k A^T_{ik}) U^{(i)}_{ik} (B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{i+1} (A_{ik} P_k) \right)^{1/2},
\]

(11)

If the unknown matrices \(X_1, X_2, \ldots, X_q \in \text{RSR}^{m \times n}\), then (1) and problem (2) are equivalent to (5) and problem (6), respectively, with

\[
M = \left( \begin{array}{cccc}
B^T_{11} \otimes A_{11} & B^T_{12} \otimes A_{12} & \cdots & B^T_{1q} \otimes A_{1q} \\
B^T_{21} \otimes A_{21} & B^T_{22} \otimes A_{22} & \cdots & B^T_{2q} \otimes A_{2q} \\
\vdots & \vdots & \ddots & \vdots \\
B^T_{p1} \otimes A_{p1} & B^T_{p2} \otimes A_{p2} & \cdots & B^T_{pq} \otimes A_{pq} \\
(B^T_{11} S_1) \otimes (A_{11} R_1) & (B^T_{12} S_2) \otimes (A_{12} R_2) & \cdots & (B^T_{1q} S_q) \otimes (A_{1q} R_q) \\
(B^T_{21} S_1) \otimes (A_{21} R_1) & (B^T_{22} S_2) \otimes (A_{22} R_2) & \cdots & (B^T_{2q} S_q) \otimes (A_{2q} R_q) \\
\vdots & \vdots & \ddots & \vdots \\
(B^T_{p1} S_1) \otimes (A_{p1} R_1) & (B^T_{p2} S_2) \otimes (A_{p2} R_2) & \cdots & (B^T_{pq} S_q) \otimes (A_{pq} R_q)
\end{array} \right),
\]

(12)

where \(P_i, R_i \in \text{SOR}^{m \times n}, i = 1, 2, \ldots, q\). Hence, the vector forms of \(\beta_i u_1 = f, \alpha_i v_1 = M^T u_i,\) \(\beta_{i+1} u_{i+1} = M v_i - \alpha_i u_i,\) and \(\alpha_{i+1} v_{i+1} = M^T u_{i+1} - \beta_{i+1} v_i (i = 1, 2, \ldots)\) in LSQR algorithm can be rewritten in matrix forms, respectively, as

\[
\alpha_i V^{(j)}_i = \sum_{j=1}^{q} \left( A^T_{ij} U^{(i)}_{ij} B^T_{ij} + R_j \left( A^T_{ij} U^{(i)T}_{ij} B^T_{ij} \right) S_j \right), \quad j = 1, 2, \ldots, q,
\]

\[
\beta_i = \sqrt{2} \left( \sum_{j=1}^{p} \left| C_j \right|^2 \right)^{1/2},
\]

where \(P_i, R_i \in \text{SOR}^{m \times n}, i = 1, 2, \ldots, q\). Hence, the vector forms of \(\beta_i u_1 = f, \alpha_i v_1 = M^T u_i,\) \(\beta_{i+1} u_{i+1} = M v_i - \alpha_i u_i,\) and \(\alpha_{i+1} v_{i+1} = M^T u_{i+1} - \beta_{i+1} v_i (i = 1, 2, \ldots)\) in LSQR algorithm can be rewritten in matrix forms, respectively, as

\[
\beta_{i+1} U^{(i)}_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} V^T_{ik} B_{jk} \right) - \alpha_i U^{(i)}_i \right), \quad j = 1, 2, \ldots, p,
\]

\[
\beta_{i+1} = \sqrt{2} \left( \sum_{j=1}^{p} \left( \sum_{k=1}^{q} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right)^{1/2},
\]

\[
\alpha_{i+1} V^{(k)}_{i+1} = \sum_{j=1}^{p} \left( A^T_{ik} U^{(i)}_{ik} B^T_{ik} + R_k \left( A^T_{ik} U^{(i)T}_{ik} B^T_{ik} \right) S_k \right) x(U^{(i)}_{i+1}, B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{i+1} (A_{ik} P_k)
\]

\[
\alpha_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} U^{(i)}_{ik} B^T_{ik} + R_k \left( A^T_{ik} U^{(i)T}_{ik} B^T_{ik} \right) S_k \right) \right)^{1/2},
\]

\[
\beta_{i+1} U^{(i)}_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right), \quad j = 1, 2, \ldots, p,
\]

\[
\beta_{i+1} = \sqrt{2} \left( \sum_{j=1}^{p} \left( \sum_{k=1}^{q} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right)^{1/2},
\]

\[
\alpha_{i+1} V^{(k)}_{i+1} = \sum_{j=1}^{p} \left( A^T_{ik} U^{(i)}_{ik} B^T_{ik} + R_k \left( A^T_{ik} U^{(i)T}_{ik} B^T_{ik} \right) S_k \right) x(U^{(i)}_{i+1}, B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{i+1} (A_{ik} P_k)
\]

\[
\alpha_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} U^{(i)}_{ik} B^T_{ik} + R_k \left( A^T_{ik} U^{(i)T}_{ik} B^T_{ik} \right) S_k \right) \right)^{1/2},
\]

\[
\beta_{i+1} U^{(i)}_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{i=1}^{p} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right), \quad j = 1, 2, \ldots, p,
\]

\[
\beta_{i+1} = \sqrt{2} \left( \sum_{j=1}^{p} \left( \sum_{k=1}^{q} A^T_{ik} V^T_{ik} B_{jk} - \alpha_i U^{(i)}_i \right) \right)^{1/2},
\]

\[
\alpha_{i+1} V^{(k)}_{i+1} = \sum_{j=1}^{p} \left( A^T_{ik} U^{(i)}_{ik} B^T_{ik} + R_k \left( A^T_{ik} U^{(i)T}_{ik} B^T_{ik} \right) S_k \right) x(U^{(i)}_{i+1}, B^T_{ik} P_k) + (P_k B_{ik}) U^{(i)T}_{i+1} (A_{ik} P_k)
\]
\[
\alpha_{i+1} = \left( \sum_{k=1}^{q} \left( \sum_{j=1}^{p} (A_{jk} U_{i+1}^{(j)} B_{jk} + R_k (A_{jk} U_{i+1}^{(j)} B_{jk}) S_k) \right) - \beta_{i+1} V_{i+1}^{(j)} \right)^{1/2},
\]

(13)

If the unknown matrices \(X_1, X_2, \ldots, X_q \in SR^{n \times n}\) (the set of symmetric matrices), then (1) and problem (2) are equivalent to (5) and problem (6), respectively, with

\[
M = \begin{pmatrix}
B_{11}^T \otimes A_{11} & B_{12}^T \otimes A_{12} & \cdots & B_{1q}^T \otimes A_{1q} \\
B_{21}^T \otimes A_{21} & B_{22}^T \otimes A_{22} & \cdots & B_{2q}^T \otimes A_{2q} \\
\vdots & \vdots & & \vdots \\
B_{pq}^T \otimes A_{p1} & B_{pq}^T \otimes A_{p2} & \cdots & B_{pq}^T \otimes A_{pq}
\end{pmatrix},
\]

(14)

\[
x = \begin{pmatrix}
\text{vec}(X_1) \\
\text{vec}(X_2) \\
\vdots \\
\text{vec}(X_q)
\end{pmatrix}, \quad f = \begin{pmatrix}
\text{vec}(C_1) \\
\text{vec}(C_2) \\
\vdots \\
\text{vec}(C_p)
\end{pmatrix}.
\]

(15)

Hence, the vector forms of \(\beta_i u_i = f, \alpha_i v_i = M^T u_i, \beta_{i+1} u_{i+1} = M v_i - \alpha_i u_i, \) and \(\alpha_{i+1} v_{i+1} = M^T u_{i+1} - \beta_{i+1} v_i \) \((i = 1, 2, \ldots)\) in LSMR algorithm can be rewritten in matrix forms, respectively, as

\[
\beta_{i+1} U_{i+1}^{(j)} = C_j, \quad j = 1, 2, \ldots, p,
\]

\[
\beta_i = \sqrt{2 \left( \sum_{j=1}^{p} \| C_j \|^2 \right)}^{1/2},
\]

\[
\alpha_i V_i^{(j)} = \sum_{j=1}^{p} (A_{ij} U_i^{(j)} B_{ij} + B_{ij} U_i^{(j)} A_{ij}), \quad j = 1, 2, \ldots, q,
\]

\[
\alpha_1 = \left( \sum_{k=1}^{q} \left( \sum_{j=1}^{p} (A_{jk} U_{i+1}^{(j)} B_{jk} + B_{jk} U_{i+1}^{(j)} A_{jk}) \right)^2 \right)^{1/2},
\]

(16)

From above results, we can obtain the matrix form iteration method of LSMR algorithm for solving the constrained solution group of (1) and problem (2). When the unknown matrices \(X_1, X_2, \ldots, X_q \in SR^{n \times n}\), the matrix form iterative method is given as shown in Algorithm 2.

### 4. The Solution Group of Problem (3)

Now, we consider the solution group of the matrix nearness problem (3) for given matrix group \((\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_q)\), where \(\overline{X}_k \in \mathbb{R}^{n \times n}, k = 1, 2, \ldots, q\). If \(X_k \in SR^{n \times n}\), it is easy to prove that

\[
\min_{X_k \in SR^{n \times n}, k=1,2,\ldots,q} \left\{ \sum_{k=1}^{q} \| X_k - \overline{X}_k \|_F^2 \right\}
\]

\[
= \min_{X_k \in SR^{n \times n}, k=1,2,\ldots,q} \left\{ \sum_{k=1}^{q} \| X_k - \frac{\overline{X}_k + \overline{X}_k^T}{2} \|_F^2 \right\}
\]

\[
+ \left\{ \sum_{k=1}^{q} \| \overline{X}_k - \frac{\overline{X}_k + \overline{X}_k^T}{2} \|_F^2 \right\}.
\]

(17)

Let

\[
\tilde{X}_k = X_k - \frac{\overline{X}_k + \overline{X}_k^T}{2}, \quad k = 1, 2, \ldots, q,
\]

(18)

\[
\tilde{C}_j = C_j - \sum_{k=1}^{q} A_{jk} \tilde{X}_k B_{jk}, \quad j = 1, 2, \ldots, p,
\]

then problem (3) is equivalent to finding the minimum Frobenius norm symmetric solution group or minimum Frobenius norm least-squares symmetric solution group of the following problems, respectively.

Compatible matrix equations are as follows:

\[
\sum_{k=1}^{q} A_{ik} \tilde{X}_k B_{ik} = \tilde{C}_i, \quad i = 1, 2, \ldots, p.
\]
Suppose that the matrices $A_{ij}$, $B_{ij}$, $C_{ij}$, $i$, $j = 1, 2$, are given by the following matrices:

$$A_{11} = \begin{pmatrix} 8 & -1 & 2 & -3 & 4 \\ 0 & 2 & 2 & 1 \\ -2 & -1 & -1 & 2 & 4 \\ 4 & 1 & 1 & 2 & 1 \end{pmatrix},$$

$$B_{11} = \begin{pmatrix} 9 & -2 & -3 & 1 & 1 \\ 0 & 5 & 4 & -2 & -2 \\ 2 & 5 & -4 & -8 & 1 \\ 9 & 0 & 2 & 5 & 1 \\ -3 & -3 & 1 & 1 & 9 \end{pmatrix},$$

$$A_{12} = \begin{pmatrix} 4 & -3 & 4 & -4 & 1 \\ -2 & -2 & 6 & -4 & 4 \\ 9 & 6 & 0 & 5 & 1 \\ 5 & 4 & 5 & 3 & 3 \\ 9 & 1 & 7 & -5 & 9 \end{pmatrix}.$$
\[
B_{12} = \begin{pmatrix}
-2 & -1 & -2 & -3 & 2 \\
1 & 5 & 4 & 9 & 6 \\
9 & -3 & 1 & 1 & 1 \\
-5 & 2 & 5 & 4 & 1 \\
4 & -8 & 2 & 2 & 1
\end{pmatrix},
\]
\[
A_{21} = \begin{pmatrix}
1 & 2 & 3 & -5 & 2 \\
4 & 6 & 1 & -3 & 9 \\
4 & 4 & -2 & 2 & 3 \\
2 & 2 & 8 & 1 & -9 \\
3 & 1 & 4 & -3 & 8
\end{pmatrix},
\]
\[
B_{21} = \begin{pmatrix}
9 & 2 & 3 & -1 & -1 \\
3 & -1 & 8 & 8 & 5 \\
-1 & -2 & -8 & -5 & -2 \\
-1 & -1 & 2 & 7 & 3 \\
1 & -1 & -8 & 8 & 2
\end{pmatrix},
\]
\[
A_{22} = \begin{pmatrix}
-9 & 2 & 3 & 2 & 1 \\
-3 & 3 & 9 & 4 & 3 \\
8 & 2 & 6 & -8 & 8 \\
1 & 2 & -9 & 4 & -2 \\
2 & -1 & -2 & -2 & -3
\end{pmatrix},
\]
\[
B_{22} = \begin{pmatrix}
-1 & -2 & 9 & 8 & 9 \\
1 & -3 & -3 & 3 & 9 \\
-1 & -3 & -9 & 5 & 9 \\
1 & -9 & -6 & 5 & -1 \\
2 & -7 & -4 & 5 & 8
\end{pmatrix}.
\]

(21)

The \(C_1\) and \(C_2\) matrices are chosen such that \(X_1 = I_n\) and \(X_2 = E_n\), where \(I_n\) and \(E_n\) are the \(n \times n\) identity matrix and the \(n \times n\) matrix whose entries are all one, respectively.

In Figure 1, we display the convergence curves of the function \(\log_{10} \delta_k\), with
\[
\delta_k = \max \left\{ \frac{\| R^{(k)} \|}{\| R^{(0)} \|}, \frac{\| R^{(k)} \|}{\| R^{(0)} \|} \right\},
\]
where \(R^{(k)}, i, k = 1, 2\), is the residual matrix of the \(i\)th equation in \(k\)th iteration. The initial iterative matrices in all the iterative methods are chosen as zero matrices of suitable size. Figure 1 confirms that the proposed algorithm has faster convergence rate and higher accuracy than the CGNE method and similar behavior to the matrix LSQR iterative method.

**Example 2.** Suppose that the matrices \(A_{ij}, B_{ij}, C_{ij}, i, j = 1, 2\), are given by the following matrices:
\[
A_{11} = \text{tridiag}(-1, 6, -1), \\
B_{11} = \text{tridiag}(1, 8, -1), \\
A_{12} = (0.1) I_n, \\
B_{12} = \text{tridiag}(1, 0, 1), \\
A_{21} = (0.1) I_n, \\
B_{21} = \text{tridiag}(-2, 1, -2), \\
A_{22} = \text{tridiag}(-1, -3, -1), \\
B_{22} = \text{tridiag}(1, 6, 2).
\]

(23)

As Example 1, the \(C_1\) and \(C_2\) matrices are chosen such that \(X_1 = I_n\) and \(X_2 = E_n\) with \(n = 400\) and the initial iterative matrices in all the iterative methods are chosen as zero matrices of suitable size. In Figure 2, as Figure 1, we display the convergence curves of the function \(\log_{10} \delta_k\). This figure shows that the LSMR method outperforms the CGNE and LSQR methods.

**Example 3** (see [45]). Consider the convection diffusion equation with the Dirichlet boundary conditions
\[
L(u) := -\Delta u + 2 \frac{\partial u}{\partial x} + 2 \frac{\partial u}{\partial y} = f \quad \text{on} \quad \Omega,
\]
\[
u = g \quad \text{on} \quad \partial \Omega.
\]

(24)

Here \(\Omega\) is the unit square \([0, 1] \times [0, 1]\). The operator \(L\) was discretized using central finite differences on \(\Omega\), with mesh size \(h = 1/(n + 1)\) in the “\(x\)” direction and \(k = 1/(p + 1)\) in the “\(y\)” direction. This yields a linear system of algebraic equations that can be written as a Sylvester matrix equation
\[
AX - XB = C,
\]

(25)

(as a particular case of (1) with \(A_{11} = A, A_{12} = -I_n, B_{11} = I_p, B_{12} = B, C_1 = C,\) and \(X_1 = X_2 = X\)) where tridiagonal matrices \(A\) and \(B\) are given by
\[
A = \frac{-1}{h^2} \text{tridiag}((1 + \nu h), -2, (1 - \nu h)),
\]
\[
B = \frac{1}{k^2} \text{tridiag}((1 + \nu k), -2, (1 - \nu k)).
\]

(26)
The right-hand side matrix \( C \) is obtained as follows:

\[
C(i, j) = f(x_{i+1}, y_{j+1}), \quad \text{for} \ i = 2, \ldots, n-1,
\]

\[
j = 2, \ldots, p-1,
\]

\[
C(1, 1) = f(x_2, y_2) + 1 + \frac{yh}{h^2} g(0, y_2)
\]

\[
+ \frac{1 + yk}{k^2} g(x_2, 0),
\]

\[
C(1, p) = f(x_2, y_{p+1}) + 1 + \frac{yh}{h^2} g(0, y_{p+1})
\]

\[
+ \frac{1 - yk}{k^2} g(x_2, 1),
\]

\[
C(n, 1) = f(x_{n+1}, y_2) + 1 + \frac{yh}{h^2} g(1, y_2)
\]

\[
+ \frac{1 + yk}{k^2} g(x_{n+1}, 0),
\]

\[
C(n, p) = f(x_{n+1}, y_{p-1}) + 1 - \frac{yh}{h^2} g(1, y_{p-1})
\]

\[
+ \frac{1 - yk}{k^2} g(x_{n+1}, 1),
\]

\[
C(1, j) = f(x_2, y_{j+1}) + 1 - \frac{yh}{h^2} g(0, y_{j+1})
\]

\[
\text{for} \ j = 2, \ldots, p-1,
\]

\[
C(n, j) = f(x_{n+1}, y_{j+1}) + 1 - \frac{yh}{h^2} g(1, y_{j+1})
\]

\[
\text{for} \ j = 2, \ldots, p-1,
\]

\[
C(i, 1) = f(x_{i+1}, y_2) + \frac{1 + yk}{k^2} g(x_{i+1}, 0),
\]

\[
\text{for} \ i = 2, \ldots, n-1,
\]

\[
C(i, p) = f(x_{i+1}, y_{p-1}) + \frac{1 - yk}{k^2} g(x_{i+1}, 1),
\]

\[
\text{for} \ i = 2, \ldots, n-1.
\]

(27)

In this example, the functions \( f \) and \( g \) were chosen such that the exact solution is

\[
u(x, y) = xe^{-x^2-y^2}
\]

(28)
on the domain \( \Omega \). In addition, we used the symmetric successive overrelaxation (SSOR) preconditioner for the matrix equation (25) to increase the convergence rate. It is easy to prove that the matrix equation (25) is equivalent to the \( np \times np \) linear system:

\[
\overline{\mathcal{A}} \tilde{x} = \tilde{c},
\]

(29)

where \( \overline{\mathcal{A}} = I_p \otimes A - B^T \otimes I_n \), \( \tilde{c} = \text{vec}(C) \), and \( \tilde{x} = \text{vec}(X) \).

The matrices \( A \) and \( B \) can be written as

\[
A = D_A - E_A - F_A,
\]

\[
B = D_B - E_B - F_B
\]

(30)

where \( D_A \) is the diagonal of \( A \) and \( -E_A \) and \( -F_A \) are the strict lower and upper part of \( A \), respectively. Then the splitting of the matrix \( \mathcal{A} \) is given as

\[
\overline{\mathcal{A}} = \overline{\mathcal{D}} - \overline{\mathcal{E}} - \overline{\mathcal{F}},
\]

(31)

with

\[
\overline{\mathcal{D}} = I_p \otimes D_A - D_B \otimes I_n,
\]

\[
\overline{\mathcal{E}} = I_p \otimes E_A - F_B^T \otimes I_n,
\]

\[
\overline{\mathcal{F}} = I_p \otimes F_A - E_B^T \otimes I_n.
\]

(32)

Now instead of solving the matrix equation (25), we will apply the LSMR-M algorithm to the preconditioned system

\[
\overline{\mathcal{A}} \tilde{x} = \tilde{c} \quad \text{with} \quad \tilde{y} = \tilde{x},
\]

(33)

where \( \tilde{x} \) is a preconditioner. As said, we use the SSOR preconditioner defined by

\[
\tilde{x} = \omega \left( \overline{\mathcal{D}} - \omega \overline{\mathcal{E}} \right)^{-1} \left( \overline{\mathcal{D}} - \omega \overline{\mathcal{F}} \right).
\]

(34)

We note that the \( np \times np \) matrix \( \overline{\mathcal{A}} \) is not used explicitly. We only use the action of the linear operator \( \mathcal{A} \) on a matrix \( V \in \mathbb{R}^{np \times p} \), defined by \( \mathcal{A}(V) = AV - VB \). In addition, we use only matrix-by-vector products; then when using the SSOR
The preconditioner we have to compute, for a given $V \in \mathbb{R}^{n \times p}$, the matrix $W \in \mathbb{R}^{n \times p}$ such that
\[ \tilde{w} = \mathcal{A} \mu_{\text{SSOR}}^{-1} \tilde{v} \quad \text{with} \quad \tilde{w} = \text{vec}(W), \quad \tilde{v} = \text{vec}(V), \quad (35) \]
or
\[ \tilde{w} = \left( \mathcal{A} \mu_{\text{SSOR}}^{-1} \right)^T \tilde{v} = \left( \mu_{\text{SSOR}}^T \right)^{-1} \mathcal{A}^T \tilde{v} \quad \text{with} \quad \tilde{w} = \text{vec}(W), \quad \tilde{v} = \text{vec}(V). \quad (36) \]

With setting
\[ \bar{v} = \mu^{-1} \tilde{v} \iff \bar{v} = \mu \bar{v} \quad \text{with} \quad \bar{v} = \text{vec}(R) \quad (37) \]
the linear system (35) is equivalent to
\[ \tilde{w} = \mathcal{A} \mu_{\text{SSOR}}^{-1} \tilde{v} \iff \tilde{w} = \mathcal{A} \bar{v}. \quad (38) \]

For computing $R$ such that $\tilde{w} = \mathcal{A} \mu_{\text{SSOR}}^{-1} \tilde{v}$, we have to solve the following matrix equations:
\[
(D_A - \omega E_A) Y - Y (D_B - \omega F_B) = \omega (2 - \omega) V, \quad (39) \\
D_A Y - YD_B = Z, \quad (40) \\
(D_A - \omega F_A) R - R (D_B - \omega E_B) = Z. \quad (41)
\]
The matrix equations (39) and (41) are also Sylvester matrix equations. But as was stated in [45], since the matrices involved in these equations are triangular, they are solved easily. In (39), the matrix $Y$ can be computed from left to right and from top to bottom in each column; this corresponds to backward substitution. Equation (41) is solved in the opposite sense and this corresponds to forward substitution. Now, to compute $W$ in (35), it is sufficient to use the action of the operator $\mathcal{A}$ on the matrix $\bar{v}$, defined by $\mathcal{A}(\bar{v}) = A\bar{v} - \bar{v}B$.

To compute $W$ in (36), first, we use the action of the operator $\mathcal{A}^T$ on the matrix, defined by $\mathcal{A}^T(\tilde{v}) = A^T\tilde{v} - \tilde{v}B^T$. Then, by setting
\[ \tilde{v} = A^T\bar{v} - \bar{v}B^T, \quad (42) \]
the linear system (36) is equivalent to
\[ \tilde{w} = \left( \mu_{\text{SSOR}}^T \right)^{-1} \tilde{v} \iff \mu_{\text{SSOR}}^T \tilde{w} = \bar{v}. \quad (43) \]

Therefore, $W$ can be obtained by solving the following matrix equations:
\[
(D_A - \omega E_A)^T Y - Y (D_B - \omega F_B)^T = \omega (2 - \omega) \bar{v}, \quad (44) \\
(D_A Y - YD_B) = Z, \quad (45) \\
(D_A - \omega F_A)^T \tilde{w} - \tilde{w} (D_B - \omega E_B) = Z. \quad (46)
\]

Similarly, the matrix equations (44) and (46) are also Sylvester matrix equations. But since the matrices involved in these equations are triangular, in (44), the matrix $Y$ can be computed from right to left and from bottom to top in each row; this corresponds to forward substitution. Equation (46) is solved in the opposite sense and this corresponds to backward substitution.

In Figure 3, we exhibited the function $\log_{10} \delta_k$ with
\[
\delta_k = \frac{\|C - (AX_k - X_k B)\|_F}{\|C - (AX_0 - X_0 B)\|_F}, \quad (47)
\]
versus the number of iterations for LSMR-M and the SSOR-LSMR-M. Furthermore, we note that for computing the quantity $\|R_k\|_F$, ($R_k$ is the residual matrix in $k$th iteration) we used the pseudocode stated in [40]. These results were obtained for $\nu = 100$, $n = 300$, $p = 300$, and $\omega = 0.9$. The initial iterative matrix was chosen as zero matrix of suitable size. As we observe by using the SSOR preconditioner the convergence rate of the LSMR-M algorithm has increased, effectively.

6. Conclusion

Solving the linear matrix equations is an attractive part of research. By extending the idea of LSMR method, we have proposed Algorithm 2 to solve the coupled matrix equations (1) or the least-squares problem (2) over generalized symmetric matrices. By this new iterative method on the selection of special initial matrix group, we obtain the minimum Frobenius norm solutions or the minimum Frobenius norm least-squares solutions over generalized symmetric matrices. All the presented results show that the matrix LSMR iterative method is efficient to compute the solution group of the general coupled matrix equations.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgment

The authors would like to thank the referees for their valuable remarks and helpful suggestions.

References


Submit your manuscripts at http://www.hindawi.com