Journal of Mathematical Research with Applications
Jul., 2019, Vol. 39, No. 4, pp. 408–432
DOI:10.3770/j.issn:2095-2651.2019.04.008
Http://jmre.dlut.edu.cn

The GPBiCG(m, l) Method for Solving General Matrix Equations

Basem I. Selim[1,2], **Lei DU**[1], **Bo YU**[1,*], **Xuanru ZHU**[1]
1. *School of Mathematical Sciences, Dalian University of Technology, Liaoning* 116024*, P. R. China;*
2. *Mathematics and Computer Science Department, Faculty of Science, Menoufia University, Shebin El-Kom* 32511*, Egypt*

Abstract The generalized product bi-conjugate gradient (GPBiCG(m, l)) method has been recently proposed as a hybrid variant of the GPBiCG and the BiCGSTAB methods to solve the linear system $Ax = b$ with non-symmetric coefficient matrix, and its attractive convergence behavior has been authenticated in many numerical experiments. By means of the Kronecker product and the vectorization operator, this paper aims to develop the GPBiCG(m, l) method to solve the general matrix equation

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i} A_{ij}X_iB_{ij} = C,$$

and the general discrete-time periodic matrix equations

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i}(A_{i,j,k}X_{i,k}B_{i,j,k} + C_{i,j,k}X_{i,k+1}D_{i,j,k}) = M_k, \quad k = 1, 2, \ldots, t,$$

which include the well-known Lyapunov, Stein, and Sylvester matrix equations that arise in a wide variety of applications in engineering, communications and scientific computations. The accuracy and efficiency of the extended GPBiCG(m, l) method assessed against some existing iterative methods are illustrated by several numerical experiments.

Keywords GPBiCG(m, l) method; Krylov Subspace method; matrix equations; Kronecker product; vectorization operator

MR(2010) Subject Classification 15A24; 65F10; 65F30

1. Introduction

In this paper, we first concern with the iterative solution of the general matrix equation

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i} A_{ij}X_iB_{ij} = C, \tag{1.1}$$

Received January 20, 2019; Accepted March 5, 2019
Supported by the National Natural Sciences Foundation of China (Grant Nos. 11501079; 11571061) and in Part by the Higher Education Commission of Egypt.
* Corresponding author
E-mail address: selim_basem@yahoo.com (Basem I. Selim); dulei@dlut.edu.cn (Lei DU); yubo@dlut.edu.cn (Bo YU)

where $A_{ij} \in \mathbf{R}^{l\times m_i}$, $B_{ij} \in \mathbf{R}^{n_i\times r}$, $C \in \mathbf{R}^{l\times r}$, for $j = 1, 2, \ldots, s_i$, $i = 1, 2, \ldots, p$, with the relation $lr = \sum_{i=1}^{p} m_i n_i$ are known matrices in which A_{ij} and B_{ij} are sparse matrices and $X_i \in \mathbf{R}^{m_i\times n_i}$, $i = 1, 2, \ldots, p$, are the matrices to be identified. Second, we discuss the iterative solution of the general discrete-time periodic matrix equations

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i}(A_{i,j,k}X_{i,k}B_{i,j,k} + C_{i,j,k}X_{i,k+1}D_{i,j,k}) = M_k, \quad k = 1, 2, \ldots, t, \tag{1.2}$$

in which the coefficient matrices $A_{i,j,k}, C_{i,j,k} \in \mathbf{R}^{l\times m_i}$, $B_{i,j,k}, D_{i,j,k} \in \mathbf{R}^{n_i\times r}$, $M_k \in \mathbf{R}^{l\times r}$ and the unknown matrices $X_{i,k} \in \mathbf{R}^{m_i\times n_i}$ are λ-cyclic matrices for $j = 1, 2, \ldots, s_i$, $i = 1, 2, \ldots, p$, with the relation $lr = \sum_{i=1}^{p} m_i n_i$ and $A_{i,j,k}, B_{i,j,k}, C_{i,j,k}$ and $D_{i,j,k}$ are sparse matrices. A λ-cyclic matrix is distinguished by repeating itself in a sequence of matrices every λ th time, e.g., $A_{i,j,\lambda+1} = A_{i,j,1}, A_{i,j,\lambda+2} = A_{i,j,2}$, etc.

The linear matrix equations have a wide range of applications in engineering, scientific computations, and communications such as system theory, stability theory, control theory, image filtering and restoration, signal processing, model reduction methods, and block diagonalization of matrices [1–10]. The discrete-time periodic matrix equations as a special case are encountered in many applications in design and analysis of many engineering and mechanical problems [3, 11–13].

Owing to these several applications, finding the solutions of various matrix equations have attracted much researchers' attention through many direct and iterative methods. As instance of the direct methods, the Sylvester matrix equation was studied using the Bartels-Stewart [14], the Hessenberg-Schur [15], and the Hessenberg [8,16] methods. Also, Penzl [17] proposed two algorithms for solving the generalized Lyapunov matrix equations. Hu and Cheng [18] presented a polynomial matrix method for solving the Sylvester matrix equation. Furthermore, Duan and Zhou [19] introduced explicit solutions to the second-order generalized Sylvester matrix equation and the generalized Sylvester matrix equation [20,21].

The iterative methods are one of the most important techniques for solving matrix equations. For instance, the gradient-based iterative (GI) method is a popular approach to solve matrix equations and was presented based on the hierarchical identification principle that considers the unknown matrix as the system parameter matrix to be determined [22–28].

Moreover, the GI method was developed to obtain some constrained solutions such as reflexive, anti-reflexive, symmetric, skew-symmetric, centro-symmetric solutions, and bisymmetric for the matrix equations [29–31].

Recently, many Krylov subspace methods that were initially presented to solve the linear systems have been developed to solve several forms of matrix equations. For instance, the conjugate gradient (CG) method has been generalized to construct iterative algorithms to deal with many forms of matrix equations over common, symmetric, skew-symmetric, reflexive and anti-reflexive, centro-symmetric, central anti-symmetric, and bisymmetric solutions [32–45].

More recently, the conjugate gradients squared (CGS), the bi-conjugate gradient stabilized (Bi-CGSTAB) [46], the quasi-minimal residual variant of the Bi-CGSTAB (QMRCGSTAB) [47], the generalized product bi-conjugate gradient (GPBiCG) [48] methods, the conjugate direction (CD)[49], the biconjugate residual (BCR) [50], and the generalized product-type BiCOR (GPBiCOR) [51] methods have been extended to obtain common and constrained solutions for many forms of linear matrix equations.

Moreover, the least-squares QR-factorization [52], the QMRCGSTAB [53], the CGS method [54,55], the GPBiCG [56,57], the biconjugate A-orthogonal residual (BiCOR) and the conjugate A-orthogonal residual squared (CORS) [58] methods have been developed to deal with several coupled matrix equations.

Furthermore, the discrete-time periodic matrix equations have been considered through some iterative methods such as the CG [59], the Bi-CGSTAB, the CGS [46], the CGLS method [60], the GI [61], the BCR [62], and the generalized product-type BiCOR (GPBiCOR)[51] methods.

Additionally, the QMRCGSTAB [53], the conjugate gradient method on the normal equations (CGNE) [63], the Bi-COR and the CORS [58], and the BCR method [64] methods have been developed for solving some types of the periodic discrete-time coupled matrix equations.

The main idea of the above approach is to transform the matrix equation into a matrix-vector form by applying the vectorization operator and the Kronecker product, then the vectorization operator is used again to express the matrix-vector multiplications in the form of matrix-matrix multiplications. Thus, the extended iterative methods may have the same advantages of the corresponding Krylov subspace iterative methods. However, they may have their disadvantages such as the possibility of breakdown and stagnation. It also can be derived that the extended CG and the extended CR families still have the same main differences.

Recently, the generalized product bi-conjugate gradient (GPBiCG(m,l)) method as a hybrid variant of the GPBiCG and the BiCGSTAB methods has been proposed by Fujino [65] to solve the linear system $Ax=b$ with non-symmetric coefficient matrix. The accuracy and efficiency of the GPBiCG(m,l) method have been approved in many applications compared to some other existing methods.

The objective of this paper is to develop the GPBiCG(m,l) method using the Kronecker product and the vectorization operator to solve the general matrix equations (1.1) and the general discrete-time periodic matrix equations (1.2). It should be indicated that these two forms of the matrix equations include many types of matrix equations.

The paper is organized as follows. We firstly give a brief review of the GPBiCG(m,l) method in Section 2. Then, we construct a matrix form of the GPBiCG(m,l) method to compute the solution of the general matrix equation (1.1) in Section 3. Moreover, we derive a matrix form of the GPBiCG(m,l) method to solve the general discrete-time periodic matrix equations (1.2) in Section 4. To investigate the convergence behaviors of the proposed methods assessed against some other existing methods, we provide some numerical examples in Section 5. Finally, we offer some concluding remarks in Section 6.

Throughout this paper, the following notations are utilized. Let $\mathbf{R}^{m\times n}$ denote the set of all $m\times n$ real matrices. A^T and $\mathrm{tr}(A)$ denote the transpose and the trace of the matrix A, respectively. For any matrices A and B with the same dimensions, the inner product of A and B

is defined as $\langle A, B\rangle = \mathrm{tr}(B^TA)$. The matrix norm of A induced by the inner product is known as Frobenius norm and referred to as $\|A\|$. $A \otimes B$ refers to the Kronecker product of matrices A and B. The vectorization operator $\mathrm{vec}(A)$ is defined as the column vector obtained by stacking up the columns of A. The vectorization is commonly applied with the Kronecker product to represent matrix multiplication as a linear transformation on matrices; for matrices A, B, and X with suitable dimensions, the following relation represents such transformation

$$\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X). \tag{1.3}$$

2. The GPBiCG(m, l) method for solving linear systems

In this section, we give a brief review of the GPBiCG(m, l) method which is one of the efficient iterative methods to determine the solution x of the linear system

$$Ax = b, \tag{2.1}$$

in which A is a given $n \times n$ matrix, b is a given n-vector, and x is an unknown n-vector.

It is well known that if the coefficient matrix A is Hermitian positive definite, the conjugate gradient (CG) method [66] is an effective tool for solving (2.1). However, the CG hardly obtains a solution if A is a large nonsymmetric matrix. The Bi-CG method, the nonsymmetric variant of the CG method by Fletcher [67], is considered a distinguished non-optimal Krylov subspace method for solving non-Hermitian systems. However, the BiCG method suffers some difficulties such as breakdowns of the first and second kind and the irregular convergence behavior in some practical applications.

Numerous variants of the BiCG method were presented to improve its behavior such as the conjugate gradient squared (CGS) method by Sonneveld [68], the biconjugate gradient stabilized (BiCGSTAB) method by van der Vorst [69], the BiCGSTAB2 method by Gutknecht [70], the BiCGSTAB(l) method by Sleijpen and Fokkema [71], and the generalized product-type method based on BiCG (GPBiCG) method by Zhang [72].

Although the CGS method is considerably faster than the BiCG method, the convergence behavior is much more irregular due to squaring the BiCG polynomial which affects the accuracy and the final convergence rate of the solution. The BiCGSTAB method converges rather smoothly and faster than the BiCG and CGS method. However, parameters' choice may lead to some practical problems such as stagnation or breakdown. The BiCGSTAB2 and the BiCGSTAB(l) methods employ both first and second (or higher) degree auxiliary polynomials to improve the convergence behavior of the BiCGSTAB method. On the other hand, the GPBiCG method uses only second degree auxiliary polynomials. Thus, the computational time per each iteration step in the GPBiCG method can be more expensive as compared with the other product-type iterative methods. However, the idea behind the GPBiCG method led to further hybridized variants by shifting between the product-type iterative methods through the choice of the parameters in a consecutive way to reduce the cost of implementing the algorithm. Motivated by this possibility, Fujino [65] proposed the GPBiCG(m, l) method as a hybrid variant of the GPBiCG and the

BiCGSTAB methods.

Algorithm 1 Algorithm of the GPBiCG method

Select initial guess x_0 and compute $r_0 = b - Ax_0$;
Choose $r_0^* = r_0$ such that $\langle r_0^*, r_0\rangle \neq 0$;
Set $t_{-1} = w_{-1} = 0,\ \beta_{-1} = 0$;
for $n = 0,\ 1,\ \ldots,$ until convergence **do**
  $p_n = r_n + \beta_{n-1}(p_{n-1} - u_{n-1})$; $q_n = Ap_n$
  $\alpha_n = \frac{\langle r_0^*, r_n\rangle}{\langle r_0^*, q_n\rangle}$;
  $t_n = r_n - \alpha_n q_n$; $y_n = t_{n-1} - t_n - \alpha_n w_{n-1}$; $s_n = At_n$;
  $\zeta_n = \frac{\langle y_n,\ y_n\rangle\langle s_n,\ t_n\rangle - \langle y_n,\ t_n\rangle\langle s_n,\ y_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$; $\eta_n = \frac{\langle s_n,\ s_n\rangle\langle y_n,\ t_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$;
  $u_n = \zeta_n q_n + \eta_n(t_{n-1} - r_n + \beta_{n-1}u_{n-1})$; $z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n$; $r_{n+1} = t_n - \eta_n y_n - \zeta_n s_n$;
  if $n = 0$ **then**
    $\zeta_n = \frac{\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle}$; (Hint: $\eta_n = 0$)
    $u_n = \zeta_n q_n$; $z_n = \zeta_n r_n - \alpha_n u_n$; $r_{n+1} = t_n - \zeta_n s_n$;
  end if
  $\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{\langle r_0^*, r_{n+1}\rangle}{\langle r_0^*, r_n\rangle}$;
  $x_{n+1} = x_n + \alpha_n p_n + z_n$; $w_n = s_n + \beta_n q_n$;
end for

Algorithm 2 Algorithm of the GPBiCG(m, l) method

Select initial guess x_0 and compute $r_0 = b - Ax_0$;
Choose $r_0^* = r_0$ such that $\langle r_0^*, r_0\rangle \neq 0$;
Set $t_{-1} = w_{-1} = 0,\ \beta_{-1} = 0$;
for $n = 0,\ 1,\ \ldots,$ until convergence **do**
  $p_n = r_n + \beta_{n-1}(p_{n-1} - u_{n-1})$; $q_n = Ap_n$;
  $\alpha_n = \frac{\langle r_0^*, r_n\rangle}{\langle r_0^*, q_n\rangle}$;
  $t_n = r_n - \alpha_n q_n$; $y_n = t_{n-1} - t_n - \alpha_n w_{n-1}$; $s_n = At_n$
  if $\mathrm{mod}(n, m + l) < m$ or $n = 0)$ **then**
    $\zeta_n = \frac{\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle}$; (Hint: $\eta_n = 0$)
    $u_n = \zeta_n q_n$; $z_n = \zeta_n r_n - \alpha_n u_n$; $r_{n+1} = t_n - \zeta_n s_n$;
  else
    $\zeta_n = \frac{\langle y_n,\ y_n\rangle\langle s_n,\ t_n\rangle - \langle y_n,\ t_n\rangle\langle s_n,\ y_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$; $\eta_n = \frac{\langle s_n,\ s_n\rangle\langle y_n,\ t_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$;
    $u_n = \zeta_n q_n + \eta_n(t_{n-1} - r_n + \beta_{n-1}u_{n-1})$; $z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n$; $r_{n+1} = t_n - \eta_n y_n - \zeta_n s_n$;
  end if
  $\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{\langle r_0^*, r_{n+1}\rangle}{\langle r_0^*, r_n\rangle}$;
  $x_{n+1} = x_n + \alpha_n p_n + z_n$; $w_n = s_n + \beta_n q_n$;
end for

In the GPBiCG(m, l) method, the parameters are computed by the BiCGSTAB method at successive m iteration steps and afterward, the parameters of the GPBiCG method are used in the subsequence. Thus, the method takes advantage of the low computational cost of the BiCGSTAB method and the good convergence behavior of the GPBiCG method. The GPBiCG and the GPBiCG(m, l) methods are presented in Algorithms 1 and 2, respectively [65,72,73]. Table 1 shows the choice of the parameters η_k and ζ_k of the BiCGSTAB, GPBiCG, BiCGSTAB2, and GPBiCG(m, l) methods. It can be noted that the GPBiCG(1, 0), GPBiCG(0, 1), and GPBiCG(1, 1) methods are corresponding to the BiCGSTAB, GPBiCG, and BiCGSTAB2 methods, respectively [65]. Also, the GPBiCG(m, l) method can be considered as a development of the BiCGSTAB2 method. It should be referred that the parameters m and l of the GPBiCG(m, l) method can be chosen regarding the difficulty of the problem.

Method	Choice of parameters ζ_k and η_k
BiCGSTAB	$\zeta_n = \frac{\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle}, \quad \eta_n = 0$
GPBiCG	$\zeta_n = \frac{\langle y_n,\, y_n\rangle\langle s_n,\, t_n\rangle - \langle y_n,\, t_n\rangle\langle s_n,\, y_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}, \eta_n = \frac{\langle s_n,\, s_n\rangle\langle y_n,\, t_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}$
BiCGSTAB2	at even iteration step:
	$\zeta_n = \frac{\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle}, \quad \eta_n = 0$
	at odd iteration step:
	$\zeta_n = \frac{\langle y_n,\, y_n\rangle\langle s_n,\, t_n\rangle - \langle y_n,\, t_n\rangle\langle s_n,\, y_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}, \eta_n = \frac{\langle s_n,\, s_n\rangle\langle y_n,\, t_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}$
GPBiCG(m, l)	at consecutive m iteration steps:
	$\zeta_n = \frac{\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle}, \quad \eta_n = 0$
	afterwards at consecutive iteration steps:
	$\zeta_n = \frac{\langle y_n,\, y_n\rangle\langle s_n,\, t_n\rangle - \langle y_n,\, t_n\rangle\langle s_n,\, y_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}, \eta_n = \frac{\langle s_n,\, s_n\rangle\langle y_n,\, t_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, t_n\rangle}{\langle s_n,\, s_n\rangle\langle y_n,\, y_n\rangle - \langle y_n,\, s_n\rangle\langle s_n,\, y_n\rangle}$

Table 1 Choice of parameters η_k and ζ_k of some product-type iterative methods

In many applications, the GPBiCG(m, l) method is indeed a considerable variant for solving large-scale problems. Based on these findings, we shall develop this method to solve the general matrix equation (1.1) and the general discrete-time periodic matrix equations (1.2) in the next two sections.

3. Matrix form of the GPBiCG(m, l) method for solving the general matrix equation

In this section, we develop an iterative algorithm based on the GPBiCG(m, l) method to obtain the solution of the general matrix equation (1.1) which can be expressed in the form

$$\sum_{j=1}^{s_1} A_{1,j}X_1B_{1,j} + \sum_{j=1}^{s_2} A_{2,j}X_2B_{2,j} + \cdots + \sum_{j=1}^{s_p} A_{p,j}X_pB_{p,j} = C. \tag{3.1}$$

The GPBiCG(m, l) method can be applied to determine the solutions of the general matrix equation (1.1), but first we must transform it to a linear system. By means of the vectorization operator and the Kronecker product, the general matrix equation (1.1) can be rewritten in the form of the linear system $Ax = b$ as follows

$$\mathrm{vec}\Big(\sum_{j=1}^{s_1} A_{1,j}X_1B_{1,j} + \sum_{j=1}^{s_2} A_{2,j}X_2B_{2,j} + \cdots + \sum_{j=1}^{s_p} A_{p,j}X_pB_{p,j}\Big) = \mathrm{vec}(C), \tag{3.2}$$

$$\underbrace{\begin{bmatrix} \sum_{j=1}^{s_1} B_{1,j}^T \otimes A_{1,j} & \sum_{j=1}^{s_2} B_{2,j}^T \otimes A_{2,j} & \cdots & \sum_{j=1}^{s_p} B_{p,j}^T \otimes A_{p,j} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \mathrm{vec}(X_1) \\ \mathrm{vec}(X_2) \\ \vdots \\ \mathrm{vec}(X_p) \end{bmatrix}}_{x} = \underbrace{\mathrm{vec}(C)}_{b}, \tag{3.3}$$

where $A \in \mathbf{R}^{lr \times u}, x, b \in \mathbf{R}^u$ and $u = \sum_{i=1}^p m_i n_i$.

It can be noticed that the dimension of the associate matrix A of the above system is large when the size of the matrices of (1.1) is large. Hence, applying Algorithm 2 of the GPBiCG(m,l) method directly to solve the above system will cause some computational difficulties due to the excessive computer memory and CPU time needed to obtain the solution. To overcome this problem, we utilize the vectorization operator again to express the vectors $r_n,\ r_0^*,\ p_n, q_n, t_n, y_n, s_n, u_n, z_n, x_n$ and w_n of Algorithm 2 as follows:

$$r_0^* = \mathrm{vec}(R_0^*), \tag{3.4}$$

$$r_n = \mathrm{vec}(R_n) = \begin{bmatrix} \mathrm{vec}(R_{1,n}) \\ \mathrm{vec}(R_{2,n}) \\ \vdots \\ \mathrm{vec}(R_{p,n}) \end{bmatrix}, q_n = \mathrm{vec}(Q_n) = \begin{bmatrix} \mathrm{vec}(Q_{1,n}) \\ \mathrm{vec}(Q_{2,n}) \\ \vdots \\ \mathrm{vec}(Q_{p,n}) \end{bmatrix}, \tag{3.5}$$

$$s_n = \mathrm{vec}(S_n) = \begin{bmatrix} \mathrm{vec}(S_{1,n}) \\ \mathrm{vec}(S_{2,n}) \\ \vdots \\ \mathrm{vec}(S_{p,n}) \end{bmatrix}, p_n = \begin{bmatrix} \mathrm{vec}(P_{1,n}) \\ \mathrm{vec}(P_{2,n}) \\ \vdots \\ \mathrm{vec}(P_{p,n}) \end{bmatrix}, \tag{3.6}$$

$$t_n = \begin{bmatrix} \mathrm{vec}(T_{1,n}) \\ \mathrm{vec}(T_{2,n}) \\ \vdots \\ \mathrm{vec}(T_{p,n}) \end{bmatrix}, y_n = \begin{bmatrix} \mathrm{vec}(Y_{1,n}) \\ \mathrm{vec}(Y_{2,n}) \\ \vdots \\ \mathrm{vec}(Y_{p,n}) \end{bmatrix}, u_n = \begin{bmatrix} \mathrm{vec}(U_{1,n}) \\ \mathrm{vec}(U_{2,n}) \\ \vdots \\ \mathrm{vec}(U_{p,n}) \end{bmatrix}, \tag{3.7}$$

$$z_n = \begin{bmatrix} \mathrm{vec}(Z_{1,n}) \\ \mathrm{vec}(Z_{2,n}) \\ \vdots \\ \mathrm{vec}(Z_{p,n}) \end{bmatrix}, x_n = \begin{bmatrix} \mathrm{vec}(X_{1,n}) \\ \mathrm{vec}(X_{2,n}) \\ \vdots \\ \mathrm{vec}(X_{p,n}) \end{bmatrix}, w_n = \begin{bmatrix} \mathrm{vec}(W_{1,n}) \\ \mathrm{vec}(W_{2,n}) \\ \vdots \\ \mathrm{vec}(W_{p,n}) \end{bmatrix}, \tag{3.8}$$

where R_0^*, R_n, Q_n, and $S_n \in \mathbf{R}^{l \times r}$ and $R_{i,n}, Q_{i,n},\ S_{i,n}, P_{i,n}, T_{i,n}, Y_{i,n}, U_{i,n}, Z_{i,n}, X_{i,n}$ and $W_{i,n} \in \mathbf{R}^{m_i \times n_i}$ for $i = 1, 2, \ldots, p$.

By considering the linear system (3.3) and the definitions (3.4)–(3.8), the vectors r_0, q_n, and s_n of Algorithm 3 can be obtained as

$$r_0 = \ b - Ax_0 \to R_0 = \ C - \sum_{i=1}^{p} \sum_{j=1}^{s_i} A_{ij} X_{i,0} B_{ij}, \tag{3.9}$$

$$q_n = \ Ap_n \to Q_n = \sum_{i=1}^{p}\sum_{j=1}^{s_i} A_{ij}P_{i,n}B_{ij}, \tag{3.10}$$

$$s_n = At_n \to S_n = \sum_{i=1}^{p}\sum_{j=1}^{s_i} A_{ij}T_{i,n}B_{ij}. \tag{3.11}$$

Also, the parameters ζ_n and η_n can be derived as

$$\begin{aligned}\zeta_n &= \frac{\langle y_n,\ y_n\rangle\langle s_n,\ t_n\rangle - \langle y_n,\ t_n\rangle\langle s_n,\ y_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}\\ &= \frac{\sum\limits_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle \sum\limits_{i=1}^{p}\langle S_{i,n},\ T_{i,n}\rangle - \sum\limits_{i=1}^{p}\langle Y_{i,n},\ T_{i,n}\rangle \sum\limits_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle}{\sum\limits_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle - \sum\limits_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle},\end{aligned} \tag{3.12}$$

$$\begin{aligned}\eta_n &= \frac{\langle s_n,\ s_n\rangle\langle y_n,\ t_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}\\ &= \frac{\sum\limits_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle Y_{i,n},\ T_{i,n}\rangle - \sum\limits_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle S_{i,n},\ T_{i,n}\rangle}{\sum\limits_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle - \sum\limits_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum\limits_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle}.\end{aligned} \tag{3.13}$$

While α_n and β_n take the forms

$$\alpha_n = \frac{\langle R_0^*,\ R_n\rangle}{\langle R_0^*,\ Q_n\rangle}, \tag{3.14}$$

$$\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \ \frac{\langle R_0^*, R_{n+1}\rangle}{\langle R_0^*, R_n\rangle}. \tag{3.15}$$

Here, by considering the definitions of the vectors (3.4)–(3.8), Eqs. (3.9)–(3.15), and Algorithm 2, we can construct Algorithm 3 as a matrix form of the GPBiCG(m, l) method for solving (1.1).

4. Matrix form of the GPBiCG(m, l) method for solving the general discrete-time periodic matrix equations

In this section, we generalize a matrix form of the GPBiCG(m, l) method to solve the general discrete-time periodic matrix equations (1.2). First, we show how Eq. (1.2) can be rewritten in the form of the general matrix equation, then we either transform it to a linear system and use Algorithm 2 to solve it, or transform it to a form of Eq. (1.1) and apply Algorithm 3 to solve it. Second, we extend Algorithm 2 directly to obtain the solutions of Eq. (1.2).

We can rewrite Eq. (1.2) in the form

$$\begin{aligned}&\sum_{j=1}^{s_1}(A_{1,j,k}X_{1,k}B_{1,j,k} + C_{1,j,k}X_{1,k+1}D_{1,j,k}) + \sum_{j=1}^{s_2}(A_{2,j,k}X_{2,k}B_{2,j,k} + C_{2,jk}X_{2,k+1}D_{2,j,k}) + \cdots +\\ &\quad \sum_{j=1}^{s_p}(A_{p,j,k}X_{p,k}B_{p,j,k} + C_{p,j,k}X_{p,k+1}D_{p,j,k}) = M_k, \quad k = 1, 2, \ldots, t.\end{aligned} \tag{4.1}$$

Algorithm 3 Algorithm of matrix form of the GPBiCG(m,l) method for solving the general matrix equation

Select initial guess $X_{i,0} \in \mathbf{R}^{m_i \times n_i}$, for $i = 1,2,\ldots,p$ and compute $R_0 = C - \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i} A_{ij}X_{i,0}B_{ij}$, $R_0 \in \mathbf{R}^{l\times r}$

Choose $R_0^* = R_0$ such that $\langle R_0^*, R_0\rangle \neq 0$.

Set $T_{i,-1} = W_{i,-1} = 0$, for $i = 1,2,\ldots,p$ and $\beta_{-1} = 0$.

for $n = 0,\ 1,\ \ldots,$ until convergence **do**

$P_{i,n} = R_{i,n} + \beta_{n-1}(P_{i,n-1} - U_{i,n-1})$, for $i = 1,2,\ldots,p$

$Q_n = \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i} A_{ij}P_{i,n}B_{ij}$

$\alpha_n = \frac{\langle R_0^*,\ R_n\rangle}{\langle R_0^*,\ Q_n\rangle}$

$T_{i,n} = R_{i,n} - \alpha_n Q_{i,n}$, for $i = 1,2,\ldots,p$

$Y_{i,n} = T_{i,n-1} - T_{i,n} - \alpha_n W_{i,n-1}$, for $i = 1,2,\ldots,p$

$S_{i,n} = \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i} A_{ij}T_{i,n}B_{ij}$

if mod$(n, m+l) < m$ or $n = 0)$ **then**

$\zeta_n = \dfrac{\sum_{i=1}^{p}\langle S_{i,n},\ T_{i,n}\rangle}{\sum_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle}$ (Hint: $\eta_n = 0$)

$U_{i,n} = \zeta_n Q_{i,n}$, for $i = 1,2,\ldots,p$

$Z_{i,n} = \zeta_n R_{i,n} - \alpha_n U_{i,n}$, for $i = 1,2,\ldots,p$

$R_{i,n+1} = T_{i,n} - \zeta_n S_{i,n}$, for $i = 1,2,\ldots,p$

else

$\zeta_n = \dfrac{\sum_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle \sum_{i=1}^{p}\langle S_{i,n},\ T_{i,n}\rangle - \sum_{i=1}^{p}\langle Y_{i,n},\ T_{i,n}\rangle \sum_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle}{\sum_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle - \sum_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle}$

$\eta_n = \dfrac{\sum_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle Y_{i,n},\ T_{i,n}\rangle - \sum_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle S_{i,n},\ T_{i,n}\rangle}{\sum_{i=1}^{p}\langle S_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle Y_{i,n},\ Y_{i,n}\rangle - \sum_{i=1}^{p}\langle Y_{i,n},\ S_{i,n}\rangle \sum_{i=1}^{p}\langle S_{i,n},\ Y_{i,n}\rangle}$

$U_{i,n} = \zeta_n Q_{i,n} + \eta_n(T_{i,n-1} - R_{i,n} + \beta_{n-1}U_{i,n-1})$, for $i = 1,2,\ldots,p$

$Z_{i,n} = \zeta_n R_{i,n} + \eta_n Z_{i,n-1} - \alpha_n U_{i,n}$, for $i = 1,2,\ldots,p$

$R_{i,n+1} = T_{i,n} - \eta_n Y_{i,n} - \zeta_n S_{i,n}$, for $i = 1,2,\ldots,p$

end if

$\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{\langle R_0^*, R_{n+1}\rangle}{\langle R_0^*, R_n\rangle}$

$X_{i,n+1} = X_{i,n} + \alpha_n P_{i,n} + Z_{i,n}$, for $i = 1,2,\ldots,p$

$W_{i,n} = S_{i,n} + \beta_n Q_{i,n}$, for $i = 1,2,\ldots,p$

end for

By defining the block matrices, we can transform Eq. (4.1) into the below general matrix equation

$$\sum_{j=1}^{s_1}(\mathcal{A}_{1,j}\mathcal{X}_1\mathcal{B}_{1,j}+\mathcal{C}_{1,j}\mathcal{X}_1\mathcal{D}_{1,j})+\sum_{j=1}^{s_2}(\mathcal{A}_{2,j}\mathcal{X}_2\mathcal{B}_{2,j}+\mathcal{C}_{2,j}\mathcal{X}_2\mathcal{D}_{2,j})+\cdots+$$
$$\sum_{j=1}^{s_p}(\mathcal{A}_{p,j}\mathcal{X}_p\mathcal{B}_{p,j}+\mathcal{C}_{p,j}\mathcal{X}_p\mathcal{D}_{p,j})=\mathcal{M}, \tag{4.2}$$

where

$$\mathcal{A}_{i,j}=\begin{bmatrix} 0 & \cdots & 0 & A_{i,j,1} \\ A_{i,j,2} & & & 0 \\ & \ddots & & \vdots \\ 0 & & A_{i,j,\lambda} & 0 \end{bmatrix},\ \mathcal{B}_{i,j}=\begin{bmatrix} 0 & B_{i,j,2} & & 0 \\ \vdots & & \ddots & 0 \\ 0 & & & B_{i,j,\lambda} \\ B_{i,j,1} & 0 & \cdots & 0 \end{bmatrix},$$

$$\mathcal{C}_{i,j}=\mathrm{diag}(\ C_{i,j,1},\ C_{i,j,2},\ \ldots,\ C_{i,j,\lambda}),\ \mathcal{D}_{i,j}=\mathrm{diag}(\ D_{i,j,1},\ D_{i,j,2},\ldots,\ D_{i,j,\lambda}),$$
$$\mathcal{M}=\mathrm{diag}(\ M_1,\ M_2,\ldots,\ M_\lambda),\ \mathcal{X}_i=\mathrm{diag}(\ X_{i,2},\ X_{i,3},\ldots,\ X_{i,\lambda},\ X_{i,1}),$$

$\mathcal{A}_{i,j},\mathcal{C}_{i,j}\in\mathbf{R}^{\lambda l\times\lambda m_i},\mathcal{B}_{i,j},\mathcal{D}_{i,j}\in\mathbf{R}^{\lambda n_i\times\lambda r},\mathcal{M}\in\mathbf{R}^{\lambda l\times\lambda r}$ and $\mathcal{X}_i\in\mathbf{R}^{\lambda m_i\times\lambda n_i}$ for $i=1,2,\ldots,p$.

Consequently, Eq. (1.2) takes the form

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i}(\mathcal{A}_{i,j}\mathcal{X}_i\mathcal{B}_{i,j}+\mathcal{C}_{i,j}\mathcal{X}_i\mathcal{D}_{i,j})=\mathcal{M}. \tag{4.3}$$

Here, by using the Kronecker product and the vectorization operator we can convert Eq. (4.3) into the nonsymmetric linear system

$$Ax=b, \tag{4.4}$$

with

$$A=\begin{bmatrix}A_1 & A_2 & \cdots & A_p\end{bmatrix},$$

where

$$A_i=\sum_{j=1}^{s_i}(\mathcal{B}_{i,j}^T\otimes\mathcal{A}_{i,j}+\mathcal{D}_{i,j}^T\otimes\mathcal{C}_{i,j})\ \text{for}\ i=1,2,\ldots,p, \tag{4.5}$$

and

$$x=\left[\mathrm{vec}(\mathcal{X}_1)^T,\mathrm{vec}(\mathcal{X}_2)^T,\ldots,\mathrm{vec}(\mathcal{X}_p)^T\right]^T,\quad b=\mathrm{vec}(\mathcal{M}), \tag{4.6}$$

where $A\in\mathbf{R}^{\lambda^2 lr\times\lambda^2 u}$, $x\in\mathbf{R}^{\lambda^2 u},b\in\mathbf{R}^{\lambda^2 lr}$ and $u=\sum_{i=1}^{p}m_in_i$.

Therefore, we can utilize Algorithm 2 for the system (4.4) to determine the solutions of (1.2).

Also, we can easily transform Eq. (4.3) into the following block matrix form

$$\sum_{i=1}^{p}\sum_{j=1}^{s_i}\begin{bmatrix}\mathcal{A}_{i,j} & \mathcal{C}_{i,j}\end{bmatrix}\begin{bmatrix}\mathcal{X}_i & 0 \\ 0 & \mathcal{X}_i\end{bmatrix}\begin{bmatrix}\mathcal{B}_{i,j} \\ \mathcal{D}_{i,j}\end{bmatrix}=\mathcal{M}, \tag{4.7}$$

which is converted to the form of Eq. (1.1). Consequently, we can implement Algorithm 3 to find the solutions. In these both ways Eqs. (3.3)–(3.8) can be concluded as below

$$R_0=\ \mathcal{M}-\sum_{i=1}^{p}\sum_{j=1}^{s_i}(\mathcal{A}_{i,j}\mathcal{X}_{i,0}\mathcal{B}_{i,j}+\mathcal{C}_{i,j}\mathcal{X}_{i,0}\mathcal{D}_{i,j}), \tag{4.8}$$

$$Q_n = \sum_{i=1}^{p}\sum_{j=1}^{s_i}(\mathcal{A}_{i,j}P_{i,n}\mathcal{B}_{i,j} + \mathcal{C}_{i,j}P_{i,n}\mathcal{D}_{i,j}), \tag{4.9}$$

$$S_n = \sum_{i=1}^{p}\sum_{j=1}^{s_i}(\mathcal{A}_{i,j}T_{i,n}\mathcal{B}_{i,j} + \mathcal{C}_{i,j}T_{i,n}\mathcal{D}_{i,j}). \tag{4.10}$$

Although the solutions of the general discrete-time periodic matrix equations (1.2) can be identified through the above two ways, the large size of the coefficient matrices will need more CPU time and excessive memory space.

To avoid these difficulties, we use the following approach to generalize Algorithm 2 of the GPBiCG(m,l) method to find the solutions of Eq. (1.2).

By using the vectorization operator and the Kronecker product, Eq. (4.1) can be transformed into the nonsymmetric linear system

$$Ax = b, \tag{4.11}$$

with a coefficient matrix A of the form

$$\begin{bmatrix}
M_{11} & N_{11} & 0 & \cdots & 0 & M_{21} & N_{21} & 0 & \cdots & 0 & M_{31} & \cdots & M_{P,1} & N_{P,1} & 0 & \cdots & 0 \\
0 & M_{12} & N_{12} & & \vdots & 0 & M_{22} & N_{22} & & \vdots & 0 & \cdots & 0 & M_{P,2} & N_{P,2} & & \vdots \\
\vdots & & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 & \vdots & \ddots & \vdots & & \ddots & \ddots & 0 \\
0 & \cdots & 0 & M_{1,\lambda-1} & N_{1,\lambda-1} & 0 & & & M_{2,\lambda-1} & N_{2,\lambda-1} & 0 & \cdots & 0 & \cdots & 0 & M_{p,\lambda-1} & N_{p,\lambda-1} \\
N_{1,\lambda} & 0 & \cdots & 0 & M_{1,\lambda} & N_{2,\lambda} & 0 & \cdots & 0 & M_{2,\lambda} & N_{3,\lambda} & \cdots & N_{P,\lambda} & 0 & \cdots & 0 & M_{P,\lambda}
\end{bmatrix},$$

and

$$x = \left[\mathrm{vec}(X_{11})^T, \ldots, \mathrm{vec}(X_{1\lambda})^T, \ldots, \mathrm{vec}(X_{p1})^T, \ldots, \mathrm{vec}(X_{p\lambda})^T\right]^T,$$

$$b = \left[\mathrm{vec}(M_1)^T, \ldots, \mathrm{vec}(M_\lambda)^T\right]^T, \tag{4.12}$$

where

$$M_{i,k} = \sum_{j=1}^{s_i} B_{i,j,k}^T \otimes A_{i,j,k}, N_{i,k} = \sum_{j=1}^{s_i} D_{i,j,k}^T \otimes C_{i,j,k}, X_{i,\lambda+1} = X_{i,1}, i = 1, 2, \ldots, p, k = 1, 2, \ldots, \lambda,$$

$$M_{i,k}, N_{i,k} \in \mathbf{R}^{lr\times m_i n_i}, A \in \mathbf{R}^{\lambda lr \times \lambda u}, x \in \mathbf{R}^{\lambda u}, b \in \mathbf{R}^{\lambda lr} \text{and } u = \sum_{i=1}^{p} m_i n_i.$$

It is obvious that the size of the associate matrix A of the linear system (4.11) is large once the size of the matrices of (1.2) is large. Therefore, applying Algorithm 2 of the GPBiCG(m,l) method directly to solve this system will cause some computational difficulties. To avoid this issue, we utilize the vectorization operator to rewrite the vectors $r_0^*, r_n, p_n, u_n, q_n, t_n, y_n, s_n, z_n$, and w_n of Algorithm 2 in the forms:

$$r_0^* = \left[\mathrm{vec}(R_{1,0}^*)^T, \mathrm{vec}(R_{2,0}^*)^T, \ldots, \mathrm{vec}(R_{\lambda,0}^*)^T\right]^T, \tag{4.13}$$

$$r_n = \begin{bmatrix} \mathrm{vec}(R_{1,n}) \\ \vdots \\ \mathrm{vec}(R_{\lambda,n}) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(R_{1,1,n}) \\ \vdots \\ \mathrm{vec}(R_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(R_{p,1,n}) \\ \vdots \\ \mathrm{vec}(R_{p,\lambda,n}) \end{bmatrix}, q_n = \begin{bmatrix} \mathrm{vec}(Q_{1,n}) \\ \vdots \\ \mathrm{vec}(Q_{\lambda,n}) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(Q_{1,1,n}) \\ \vdots \\ \mathrm{vec}(Q_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(Q_{p,1,n}) \\ \vdots \\ \mathrm{vec}(Q_{p,\lambda,n}) \end{bmatrix}, \tag{4.14}$$

$$s_n = \begin{bmatrix} \mathrm{vec}(S_{1,n}) \\ \vdots \\ \mathrm{vec}(S_{\lambda,n}) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(S_{1,1,n}) \\ \vdots \\ \mathrm{vec}(S_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(S_{p,1,n}) \\ \vdots \\ \mathrm{vec}(S_{p,\lambda,n}) \end{bmatrix}, p_n = \begin{bmatrix} \mathrm{vec}(P_{1,1,n}) \\ \vdots \\ \mathrm{vec}(P_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(P_{p,1,n}) \\ \vdots \\ \mathrm{vec}(P_{p,\lambda,n}) \end{bmatrix}, t_n = \begin{bmatrix} \mathrm{vec}(T_{1,1,n}) \\ \vdots \\ \mathrm{vec}(T_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(T_{p,1,n}) \\ \vdots \\ \mathrm{vec}(T_{p,\lambda,n}) \end{bmatrix}, \tag{4.15}$$

$$y_n = \begin{bmatrix} \mathrm{vec}(Y_{1,1,n}) \\ \vdots \\ \mathrm{vec}(Y_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(Y_{p,1,n}) \\ \vdots \\ \mathrm{vec}(Y_{p,\lambda,n}) \end{bmatrix}, u_n = \begin{bmatrix} \mathrm{vec}(U_{1,1,n}) \\ \vdots \\ \mathrm{vec}(U_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(U_{p,1,n}) \\ \vdots \\ \mathrm{vec}(U_{p,\lambda,n}) \end{bmatrix}, z_n = \begin{bmatrix} \mathrm{vec}(Z_{1,1,n}) \\ \vdots \\ \mathrm{vec}(Z_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(Z_{p,1,n}) \\ \vdots \\ \mathrm{vec}(Z_{p,\lambda,n}) \end{bmatrix}, w_n = \begin{bmatrix} \mathrm{vec}(W_{1,1,n}) \\ \vdots \\ \mathrm{vec}(W_{1,\lambda,n}) \\ \vdots \\ \mathrm{vec}(W_{p,1,n}) \\ \vdots \\ \mathrm{vec}(W_{p,\lambda,n}) \end{bmatrix}, \tag{4.16}$$

where $R^*_{k,0}, R_{k,n}, Q_{k,n}$, and $S_{k,n} \in \mathbf{R}^{l\times r}$ and $R_{i,k,n}, Q_{i,k,n}, S_{i,k,n}, P_{i,k,n}, T_{i,k,n}, Y_{i,k,n}, U_{i,k,n}, Z_{i,k,n}$, and $W_{i,k,n} \in \mathbf{R}^{m_i\times n_i}$, $i = 1, 2, \ldots, p$, $k = 1, 2, \ldots, \lambda$.

Hence, by considering the linear system (4.11) and the definitions above in Eqs. (4.12)– (4.16), we can conclude the vectors r_0, q_n, and s_n of Algorithm 2 as below

$$r_0 = b - Ax_0 \ \rightarrow R_{k,0} = M_k - \sum_{i=1}^{p}\sum_{j=1}^{s_i}(A_{i,j,k}X_{i,k,0}B_{i,j,k} + C_{i,j,k}X_{i,k+1,0}D_{i,j,k}), \tag{4.17}$$

$$q_n = Ap_n \ \rightarrow Q_{k,n} = \sum_{i=1}^{p}\sum_{j=1}^{s_i}(A_{i,j,k}P_{i,k,n}B_{i,j,k} + C_{i,j,k}P_{i,k+1,n}D_{i,j,k}), \tag{4.18}$$

$$s_n = At_n \ \rightarrow S_{k,n} = \sum_{i=1}^{p}\sum_{j=1}^{s_i}(A_{i,j,k}T_{i,k,n}B_{i,j,k} + C_{i,j,k}T_{i,k+1,n}D_{i,j,k}). \tag{4.19}$$

Moreover, the parameters α_n and β_n can be concluded as

$$\alpha_n = \frac{\langle r_0^*, r_n\rangle}{\langle r_0^*, q_n\rangle} = \left\langle \begin{bmatrix} \mathrm{vec}(R_{1,0}^*) \\ \mathrm{vec}(R_{2,0}^*) \\ \vdots \\ \mathrm{vec}(R_{\lambda,0}^*) \end{bmatrix}, \begin{bmatrix} \mathrm{vec}(R_{1,n}) \\ \mathrm{vec}(R_{2,n}) \\ \vdots \\ \mathrm{vec}(R_{\lambda,n}) \end{bmatrix} \right\rangle \Bigg/ \left\langle \begin{bmatrix} \mathrm{vec}(R_{1,0}^*) \\ \mathrm{vec}(R_{2,0}^*) \\ \vdots \\ \mathrm{vec}(R_{\lambda,0}^*) \end{bmatrix}, \begin{bmatrix} \mathrm{vec}(Q_{1,n}) \\ \mathrm{vec}(Q_{2,n}) \\ \vdots \\ \mathrm{vec}(Q_{\lambda,n}) \end{bmatrix} \right\rangle$$

$$= \sum_{k=1}^{\lambda} \langle R_{k,0}^*,\ R_{k,n}\rangle \Bigg/ \sum_{k=1}^{\lambda} \langle R_{k,0}^*,\ Q_{k,n}\rangle, \tag{4.20}$$

$$\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{\langle r_0^*, r_{n+1}\rangle}{\langle r_0^*, r_n\rangle} = \frac{\alpha_n}{\zeta_n} \cdot \frac{\sum_{k=1}^{\lambda} \langle R_{k,0}^*, R_{k,n+1}\rangle}{\sum_{k=1}^{\lambda} \langle R_{k,0}^*, R_{k,n}\rangle}. \tag{4.21}$$

In addition, for the parameters ζ_n, η_n, we have

$$\zeta_n = \frac{\langle y_n,\ y_n\rangle\langle s_n,\ t_n\rangle - \langle y_n,\ t_n\rangle\langle s_n,\ y_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$$

$$= \frac{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ T_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ T_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}, \tag{4.22}$$

$$\eta_n = \frac{\langle s_n,\ s_n\rangle\langle y_n,\ t_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ t_n\rangle}{\langle s_n,\ s_n\rangle\langle y_n,\ y_n\rangle - \langle y_n,\ s_n\rangle\langle s_n,\ y_n\rangle}$$

$$= \frac{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ T_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ T_{i,k,n}\rangle}{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}. \tag{4.23}$$

From the relations (4.17)–(4.23), Algorithm 2 can be generalized to construct a matrix form of the GPBiCG(m, l) method for the solutions of Eq. (1.2) as shown in Algorithm 4.

5. Numerical results

In this section, the following experiments are presented to illustrate some properties of the extended GPBiCG(m, l) method (GPBiCG(m, l)_M) when applied to solve seven test problems. For comparison, we consider the extended GPBiCG (GPBiCG_M), the extended BiCGSTAB (BiCGSTAB_M), the extended BiCGSTAB2 (BiCGSTAB2_M), the extended CGS (CGS_M), the extended CORS (CORS_M), and the extended BiCOR (BiCOR_M) methods. We should refer to that the GPBiCG(0,1)_M, GPBiCG(1,0)_M, and GPBiCG(1,1)_M methods are corresponding to the GPBiCG_M, BiCGSTAB_M, and BiCGSTAB2_M methods, respectively. The experiments aim to show the potential of the proposed method to solve efficiently sparse matrix equations.

The experiments have been carried out using MATLAB 2017b with a Windows (64 bit) on PC-Intel(R) Core(TM) i7-3612QM CPU 2.10 GHz, 8 GB of RAM. The performance is examined in four aspects: number of iterations (referred to as Iters), CPU time in seconds (referred to as

Algorithm 4 Algorithm of matrix form of the GPBiCG(m,l) method for solving the general discrete-time periodic matrix equations

Select initial guess $X_{i,k,0} \in \mathbf{R}^{m_i \times n_i}$ and set $X_{i,\lambda+1,0} = X_{i,1,0}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$.
Compute $R_{k,0} = M_k - \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i}(A_{i,j,k}X_{i,k,0}B_{i,j,k} + C_{i,j,k}X_{i,k+1,0}D_{i,j,k})$, $R_{k,0} \in \mathbf{R}^{l\times r}$ for $k = 1, \ldots, \lambda$,
and set $R_{i,\lambda+1,0} = R_{i,1,0}$, for $i = 1, 2, \ldots, p$.
Choose $R^*_{k,0} = R_{k,0}$ such that $\left\langle R^*_{k,0}, R_{k,0} \right\rangle \neq 0$,
and set $T_{i,k,-1} = W_{i,k,-1} = 0$, $\beta_{-1} = 0$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
for $n = 0,\ 1,\ \ldots,$ until convergence **do**
  $P_{i,k,n} = R_{i,k,n} + \beta_{n-1}(P_{i,k,n-1} - U_{i,k,n-1})$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  $Q_{k,n} = \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i}(A_{i,j,k}P_{i,k,n}B_{i,j,k} + C_{i,j,k}P_{i,k+1,n}D_{i,j,k})$, for $k = 1, \ldots, \lambda$,
  $\alpha_n = \dfrac{\sum_{k=1}^{\lambda}\langle R^*_{k,0}, R_{k,n}\rangle}{\sum_{k=1}^{\lambda}\langle R^*_{k,0}, Q_{k,n}\rangle}$
  $T_{i,k,n} = R_{i,k,n} - \alpha_n P_{i,k,n}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  $Y_{i,k,n} = T_{i,k,n-1} - T_{i,k,n} - \alpha_n W_{i,k,n-1}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  $S_{k,n} = \sum\limits_{i=1}^{p}\sum\limits_{j=1}^{s_i}(A_{i,j,k}T_{i,k,n}B_{i,j,k} + C_{i,j,k}T_{i,k+1,n}D_{i,j,k})$
  if $\mathrm{mod}(n, m+l) < m$ or $n = 0$) **then**
    $\zeta_n = \dfrac{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ T_{i,k,n}\rangle}{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle}$ (Hint: $\eta_n = 0$)
    $U_{i,k,n} = \zeta_n Q_{i,k,n}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
    $Z_{i,k,n} = \zeta_n R_{i,k,n} - \alpha_n U_{i,k,n}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
    $R_{i,k,n+1} = T_{i,k,n} - \zeta_n S_{i,k,n}$ and set $R_{i,\lambda+1,n+1} = R_{i,1,n+1}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  else
    $\zeta_n = \dfrac{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ T_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ T_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}$
    $\eta_n = \dfrac{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ T_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ T_{i,k,n}\rangle}{\sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ Y_{i,k,n}\rangle - \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle Y_{i,k,n},\ S_{i,k,n}\rangle \sum_{i=1}^{p}\sum_{k=1}^{\lambda}\langle S_{i,k,n},\ Y_{i,k,n}\rangle}$
    $U_{i,k,n} = \zeta_n Q_{i,k,n} + \eta_n(T_{i,k,n-1} - R_{i,k,n} + \beta_{n-1}U_{i,k,n-1})$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
    $Z_{i,k,n} = \zeta_n R_{i,k,n} + \eta_n Z_{i,k,n-1} - \alpha_n U_{i,k,n}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
    $R_{i,k,n+1} = T_{i,k,n} - \eta_n Y_{i,k,n} - \zeta_n S_{i,k,n}$ and set $R_{i,\lambda+1,n+1} = R_{i,1,n+1}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  end if
  $\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \dfrac{\sum_{k=1}^{\lambda}\langle R^*_{k,0}, R_{k,n+1}\rangle}{\sum_{k=1}^{\lambda}\langle R^*_{k,0}, R_{k,n}\rangle}$
  $X_{i,k,n+1} = X_{i,k,n} + \alpha_n P_{i,k,n} + Z_{i,k,n}$ and set $X_{i,\lambda+1,n+1} = X_{i,1,n+1}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
  $W_{i,k,n} = S_{i,k,n} + \beta_n Q_{i,k,n}$, for $i = 1, 2, \ldots, p$, and $k = 1, \ldots, \lambda$
end for

Time), $\log_{10}$ of the updated and final true relative residual Frobenius norms (referred to as Relres and TRR). The CPU time is computed as the time average of implementing the algorithms for one hundred times. Relres and TRR are defined, respectively, for Algorithm 3 as $\sqrt{\sum_{i=1}^{p} \|R_{i,n}\|^2} \Big/ \|C\|$ and $\|C - \sum_{i=1}^{p} \sum_{j=1}^{s_i} A_{ij} X_{i,n} B_{ij}\| / \|C\|$, and for Algorithm 4 as $\sqrt{\sum_{i=1}^{p} \sum_{k=1}^{\lambda} \|R_{i,k,n}\|^2} \Big/ \sqrt{\sum_{k=1}^{\lambda} \|M_k\|^2}$ and $\sqrt{\sum_{k=1}^{\lambda} \|M_k - \sum_{i=1}^{p} \sum_{j=1}^{s_i} (A_{i,j,k} X_{i,k,n} B_{i,j,k} + C_{i,j,k} X_{i,k+1,n} D_{i,j,k})\|^2} \Big/ \sqrt{\sum_{k=1}^{\lambda} \|M_k\|^2}$.

In all examples, we take the zero matrix as an initial guess and the stopping criterion for successful convergence is that the Relres is less than a given tolerance, referred to as TOL, which is set as $\mathrm{TOL} = 10^{-10}$. The maximal number of iterations referred to as MAXIT, which is taken as $\mathrm{MAXIT} = 5000$. The computational results are reported in Tables 2 and 3 and are displayed in Figures 1–8 which indicate the iteration number on the horizontal axis versus the Frobenius norms of the relative residuals on the vertical axis. A symbol "max" refers to that the method did not meet the required TOL before MAXIT.

Example 1 First, we study the Sylvester matrix equation $AX + XB = C$ for the next two cases

Case 1.1 Refer to [57], where the parameters have been modified to take the forms

$$A = \mathrm{triu}(\mathrm{rand}(n), 1) + \mathrm{diag}(3 + \mathrm{diag}(\mathrm{rand}(n))),$$

$$B = \mathrm{tril}(\mathrm{rand}(n), 1) + \mathrm{diag}(2 + \mathrm{diag}(\mathrm{rand}(n))), \ C = \mathrm{rand}(n),$$

where $n = 500$. The numerical results of the stated iterative algorithms are reported in Table 2. It can be seen that the GPBiCG(1,2)_M and the GPBiCG(1,1)_M methods are more efficient than the other methods regarding the number of Iters and the CPU time. Also, it can be noticed that the BiCOR_M method is more expensive regarding the number of Iters and the CPU time while the GPBiCG(4,1)_M method is more expensive regarding the CPU time. The accuracy of all the obtained solutions regarding TRR is roughly identical to the tolerance value as a stopping criterion. In addition, Figure 1 shows the convergence behavior of some of the iterative solvers. It can be seen that the CORS_M and the CGS_M methods show irregular (oscillating) convergence curve, while the convergence behavior of the remaining methods shows reasonably smooth decreasing residual.

Case 1.2 Here, another case is also considered after some adjustments in the parameters stated in [74]

$$A = M + rN + \frac{100}{(n+1)^2} I, \ B = M + 3rN + \frac{100}{(n+1)^2} I, \ C = \mathrm{rand}(n),$$

where $M = \mathrm{tridiag}(-1, 2, -1)$, $N = \mathrm{tridiag}(0.5, 0, -0.5)$, for $n = 500$ and $r = 1.5$. Through implementing the above methods for acquiring the approximations, we obtain the computational results that are shown in Table 2. It can be recognized that most of the GPBiCG(m, l)_M converges faster than the other methods. On contrary, the BiCGSTAB_M method shows a high number of Iters and a high cost in CPU time. While the accuracy of the computed solutions by

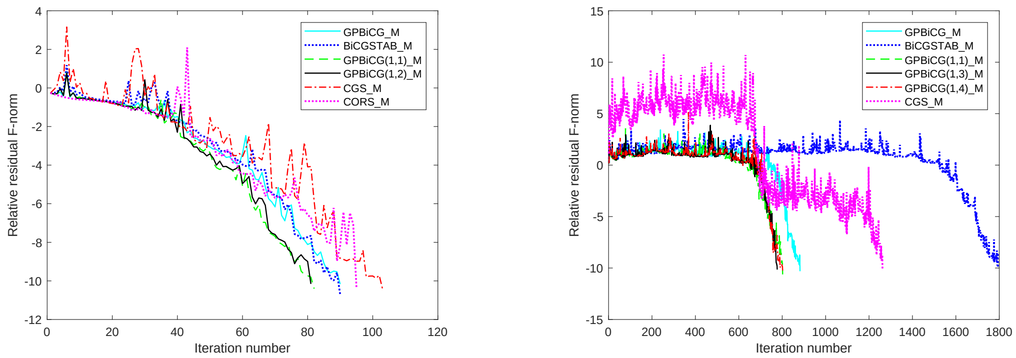

Figure 1 Convergence histories of different iterative methods for Case 1.1(Left) and Case 1.2(Right).

applying most of the iterative solvers (in terms of TRR) is roughly equivalent to the tolerance value that is taken as a stopping criterion, the CGS_M solver exhibits a low accuracy. In Figure 1, the convergence histories of some of the iterative methods are also shown. We can notice the smooth convergence behavior of most of the plotted methods except for the CGS_M and the BiCGSTAB_M methods.

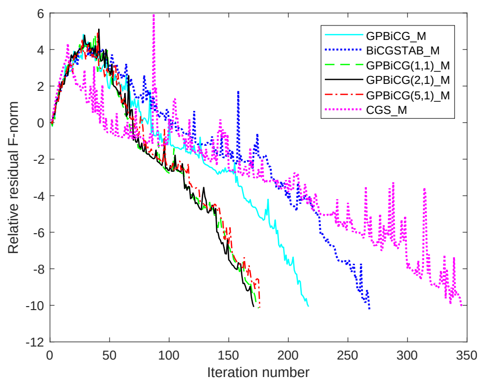

Figure 2 Convergence histories of different iterative methods for Example 2

Example 2 Next, we test the matrix equation $AXB = C$, after some minor modifications to the parameters in [58]

$$A = \mathrm{triu}(\mathrm{rand}(n), 1) + \mathrm{diag}(3 + \mathrm{diag}(\mathrm{rand}(n))),$$

$$B = \mathrm{tril}(\mathrm{rand}(n), 1) + \mathrm{diag}(8 + \mathrm{diag}(\mathrm{rand}(n))), \quad C = \mathrm{rand}(n),$$

where $n = 500$. The numerical results are stated in Table 3 which shows that the GPBiCG$(2, 1)$_M outperforms the other methods in CPU time and number of Iters. One can notice that the BiCOR_M method still needs more CPU time and Iters than the other methods followed by the CGS_M, the BiCGSTAB_M, and the CORS_M methods. The accuracy of the obtained solutions is roughly identical to the TOL value that was set as a stopping criterion except for the BiCOR_M

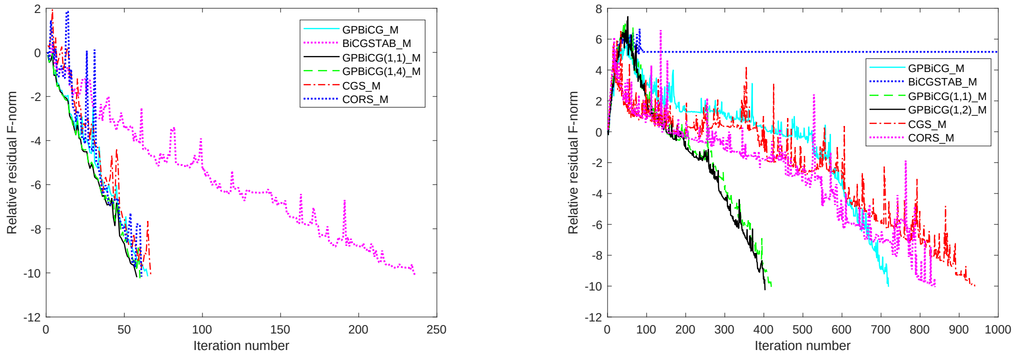

Figure 3 Convergence histories of different iterative methods for Case 3.1(Left) and Case 3.2(Right).

solver which reduces to half of the TOL. Moreover, Figure 2 shows the plots of the convergence histories of some of the iterative methods. The GPBiCG(m, l)_M and the GPBiCG_M methods show fairly smooth convergence behaviors. The BiCGSTAB method exhibits some extent acceptable convergence behaviors, while the CGS_M method shows an irregular (oscillating) convergence curve.

Example 3 In this example, we investigate the matrix equation $AXB + CXD = E$, for the following two cases

Case 3.1 Consider the parameters below

$$A = M + 2rN + \frac{100}{(n+1)^2}I,\ B = M + 3rN + \frac{100}{(n+1)^2}I,$$

$$C = M + rN + \frac{100}{(n+1)^2}I,\ D = M + 3rN + \frac{100}{(n+1)^2}I,\ E = \text{rand}(n),$$

where $M = \text{tridiag}(-1, 2, 0.5)$, $N = \text{tridiag}(0.5, 0, -0.5)$, for $n = 500$ and $r = 1.5$. Table 2 summarizes the characteristics of the numerical results for the mentioned iterative methods. It is apparent that the GPBiCG$(1, 1)$_M method converges faster than the other methods in terms of the CPU time and the number of Iters. It can be seen that the BiCGSTAB_M method has a slow convergence rate concerning the high number of Iters and the CPU time followed by the BiCOR_M method. The accuracy of the approximated solutions concerning TRR is roughly equal to the tolerance value as a stopping criterion. For simplicity, Figure 3 illustrates plots of the convergence behaviors to some of the mentioned iterative methods, where the GPBiCG(m, l)_M and the GPBiCG_M methods still have fairly smooth convergence behaviors. It is apparent that the remaining methods show typically erratic convergence behaviors especially the BiCGSTAB_M method.

Case 3.2 Here, we examine the parameters stated in [75] with some slight modifications

$$A = \text{triu}(\text{rand}(n), 1) + \text{diag}(8 + \text{diag}(\text{rand}(n))),\ B = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),$$

$$C = \text{triu}(\text{rand}(n), 1) + \text{diag}(8 + \text{diag}(\text{rand}(n))),\ D = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),$$

$$E = \text{rand}(n),$$

for $n = 500$. The computational results of different iterative solvers are stated in Table 2. The GPBiCG$(1, 2)$_M is more efficient than the other methods concerning the smaller number of Iters and the CPU time. On contrary, the CGS_M method needs a high cost of CPU time and a high number of Iters followed by the CORS_M method. one can notice that the BiCGSTAB_M and BiCOR_M fail to converge. The accuracy of the obtained approximations concerning TRR by implementing the iterative methods is reduced to two-thirds of the tolerance value that is chosen as a stopping criterion, but is still acceptable. The convergence histories to some of the iterative methods are also shown in Figure 3, where the GPBiCG(m, l)_M and the GPBiCG_M methods have preferable convergence behaviors assessed against the other methods which have typically erratic convergence behaviors.

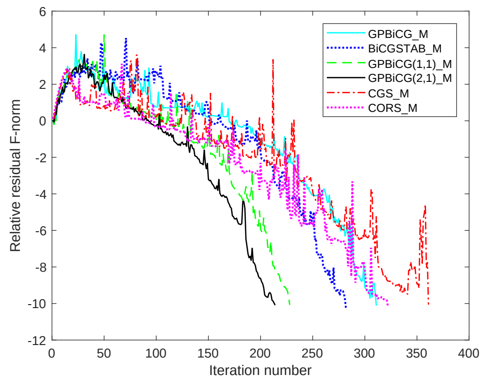

Figure 4 Convergence histories of different iterative methods for Example 4

Example 4 Next, we consider the discrete-time periodic Sylvester matrix equations $A_k X_k B_k + X_{k+1} = E_k, k = 1, 2$, where

$$A1 = \text{triu}(\text{rand}(n), 1) + \text{diag}(9 + \text{diag}(\text{rand}(n))), \quad A2 = \text{triu}(\text{rand}(n), 1) + \text{diag}(9 + \text{diag}(\text{rand}(n))),$$

$$B1 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))), \quad B2 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),$$

$$E1 = \text{rand}(n), \quad E2 = \text{rand}(n),$$

for $n = 300$. We also apply the mentioned methods with the initial matrices $X_i = 0, i = 1, 2, 3$ to get the approximations. The numerical outputs are listed in Table 3. It can be observed that the GPBiCG(2,1)_M method converges faster than the other methods regarding the CPU time and the number of Iters. It is apparent that the accuracy of the obtained approximations by applying the stated iterative methods regarding TRR is appropriately equal to the tolerance value that is taken as a stopping criterion except the BiCOR_M method which reduces to half of the TOL. Figure 4 presents typical plots of the convergence histories of some of the stated methods. It can be seen that the GPBiCG and the GPBiCG(m, l)_M methods still have reasonably smooth convergence behaviors. While the BiCGSTAB_M solver exhibits somewhat acceptable convergence behavior, the CGS_M and the CORS_M solvers exhibit erratic convergence behaviors.

Method	Iters	TRR	Time (s)	Iters	TRR	Time (s)
	Case 1.1			Case 1.2		
GPBiCG_M	90	-10.1682	3.2480	882	-9.9122	39.3278
BiCGSTAB_M	90	-10.7092	3.1910	1795	-9.8190	68.4095
GPBiCG(1, 1)_M	82	-10.4009	2.8951	802	-10.2622	37.3127
GPBiCG(1, 2)_M	**81**	-10.1563	**2.8867**	824	-10.2119	38.6519
GPBiCG(1, 3)_M	83	-10.5285	3.5428	**777**	-9.5408	**36.6381**
GPBiCG(1, 4)_M	85	-10.2910	3.7667	802	-9.2074	37.6052
GPBiCG(1, 5)_M	85	-10.0950	3.8858	836	-9.7079	39.1702
GPBiCG(2, 1)_M	82	-10.5286	3.7167	810	-10.0644	37.9198
GPBiCG(3, 1)_M	82	-10.3187	3.5602	838	-9.9531	38.8946
GPBiCG(4, 1)_M	84	-10.1160	4.9719	951	-9.4509	43.8428
GPBiCG(5, 1)_M	84	-10.1940	4.1231	888	-7.6950	41.4315
CGS_M	103	-10.4126	3.5223	1262	-4.6573	56.7263
CORS_M	95	-10.3435	3.4824	1167	-9.0450	54.1815
BiCOR_M	154	-10.0163	5.3884	1416	-9.8843	62.4768
	Case 3.1			Case 3.2		
GPBiCG_M	65	-10.1559	5.8609	719	-6.8431	52.7736
BiCGSTAB_M	236	-10.0982	20.3152	max	5.2137	485.0885
GPBiCG(1, 1)_M	**58**	-10.1968	**4.9507**	419	-6.9104	30.9961
GPBiCG(1, 2)_M	59	-10.0897	5.2623	**403**	-6.5491	**29.0221**
GPBiCG(1, 3)_M	60	-10.0830	5.3802	420	-6.4781	30.3230
GPBiCG(1, 4)_M	60	-10.2453	4.3017	484	-7.3202	36.8407
GPBiCG(1, 5)_M	64	-10.5004	5.4994	486	-5.1670	37.3621
GPBiCG(2, 1)_M	60	-10.0044	5.1857	466	-6.7697	35.6664
GPBiCG(3, 1)_M	69	-10.2790	6.0241	396	-6.9157	30.8324
GPBiCG(4, 1)_M	72	-10.0690	6.2875	449	-6.7088	35.1902
GPBiCG(5, 1)_M	78	-10.3402	6.8212	415	-7.3548	31.6364
CGS_M	67	-10.1578	5.7652	940	-8.5477	79.9300
CORS_M	61	-10.1718	5.1379	838	-8.1346	76.9950
BiCOR_M	118	-10.0242	9.6436	max	7.1937	444.8939

Table 2 The numerical results of different iterative solvers for Examples 1, 3.

Example 5 After considering some slight changes to the parameters in [46], the discrete-time periodic Sylvester matrix equations $X_k + C_k X_{k+1} D_k = E_k, k = 1, 2$ is also studied with

$$C1 = \text{triu}(\text{rand}(n), 1) + \text{diag}(50 + \text{diag}(\text{rand}(n))), \ C2 = \text{triu}(\text{rand}(n), 1) + \text{diag}(50 + \text{diag}(\text{rand}(n))),$$

$$D1 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))), \ D2 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),$$

$$E1 = \text{rand}(n), \quad E2 = \text{rand}(n),$$

for $n = 200$. The computational results obtained with the initial matrices $X_i = 0, i = 1, 2, 3$ are displayed in Table 3. It can be seen that the GPBiCG$(1,3)$_M methods performs better than the other methods regarding the smaller number of Iters and the CPU time. It is apparent the BiCGSTAB_M and the CGS_M solvers need more Iters and CPU time. The accuracy of the approximations calculated concerning the TRR values is slightly less than the TOL value that is set as a stopping criterion and falls to half of the TOL in case of the GPBiCG$(1,2)$_M and BiCOR_M methods. The results of some of the iterative methods are depicted in Figure 5 where the GPBiCG(m,l)_M and the GPBiCG_M methods exhibit somewhat acceptable convergence behaviors and the CGS_M method shows an irregular convergence behavior.

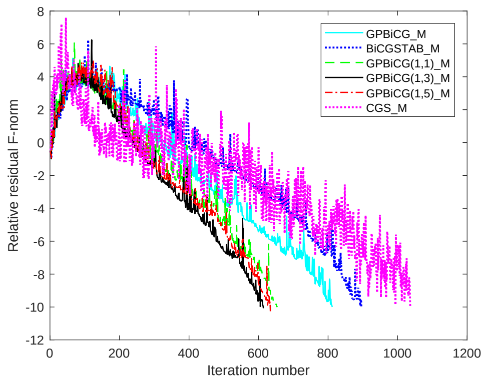

Figure 5 Convergence histories of different iterative methods for Example 5

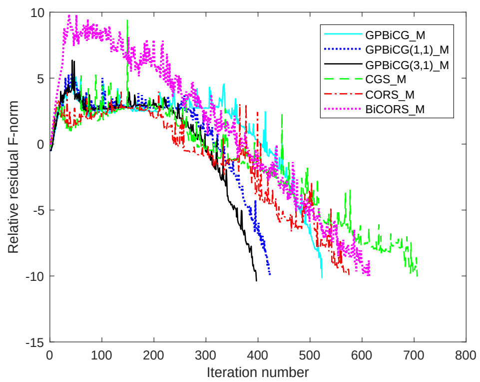

Figure 6 Convergence histories of different iterative methods for Example 6

Method	Iters	TRR	Time (s)	Iters	TRR	Time (s)
	Example 2			Example 4		
GPBiCG_M	217	-9.8005	8.1096	311	-8.8653	5.0144
BiCGSTAB_M	268	-10.0611	19.5060	282	-10.2224	4.5279
GPBiCG(1,1)_M	175	-8.9137	8.2357	228	-7.4375	3.6334
GPBiCG(1,2)_M	183	-9.0456	8.4352	230	-9.0614	3.7989
GPBiCG(1,3)_M	182	-8.9671	8.4225	245	-9.8697	3.9361
GPBiCG(1,4)_M	198	-9.7652	9.0272	246	-10.2205	3.9938
GPBiCG(1,5)_M	191	-9.5658	8.5559	232	-9.5446	3.6835
GPBiCG(2,1)_M	**171**	-8.9795	**7.7013**	**214**	-9.6084	**3.4358**
GPBiCG(3,1)_M	178	-8.5294	8.1977	262	-9.8470	4.2266
GPBiCG(4,1)_M	181	-9.3082	8.4152	226	-10.4067	3.5701
GPBiCG(5,1)_M	176	-9.3872	8.1607	228	-9.9946	3.6752
CGS_M	346	-8.9476	20.9887	361	-10.0607	4.9810
CORS_M	292	-10.3601	18.3857	322	-10.1091	5.1551
BiCOR_M	363	-5.5092	21.5462	380	-5.9163	5.5939
	Example 5			Example 6		
GPBiCG_M	812	-7.7253	4.6909	523	-7.3641	18.4198
BiCGSTAB_M	896	-8.5606	5.0627	479	-8.0667	12.1076
GPBiCG(1,1)_M	654	-6.7986	3.7031	424	-7.0825	15.1521
GPBiCG(1,2)_M	738	-5.0604	4.1772	417	-7.8694	14.6532
GPBiCG(1,3)_M	**614**	-8.6150	**3.5903**	448	-6.5951	15.9368
GPBiCG(1,4)_M	665	-7.5128	3.7525	501	-8.0137	17.8798
GPBiCG(1,5)_M	634	-7.4066	3.5589	474	-8.2086	16.8103
GPBiCG(2,1)_M	674	-6.1781	3.8092	420	-7.3106	14.5930
GPBiCG(3,1)_M	714	-6.9155	3.9762	**397**	-8.0684	**10.7976**
GPBiCG(4,1)_M	695	-8.0637	3.9288	417	-7.4675	11.4000
GPBiCG(5,1)_M	747	-7.1971	4.1525	438	-8.4071	12.0270
CGS_M	1037	-7.4055	4.8949	706	-4.8637	17.8298
CORS_M	773	-7.2565	4.1121	575	-9.3350	15.2128
BiCOR_M	903	-4.1335	4.5196	614	-3.7364	15.2135

Table 3 The numerical results of different iterative solvers for Examples 2, 4:6

Example 6 Finally, the discrete-time periodic Sylvester matrix equations $A_kX_kB_k+C_kX_{k+1}D_k = E_k, k = 1, 2$, stated in [62], is investigated after considering some changes to the parameters

$$A1 = \text{triu}(\text{rand}(n), 1) + \text{diag}(8 + \text{diag}(\text{rand}(n))),\ \ A2 = \text{triu}(\text{rand}(n), 1) + \text{diag}(8 + \text{diag}(\text{rand}(n))),$$

$$B1 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),\ \ B2 = \text{tril}(\text{rand}(n), 1) + \text{diag}(1 + \text{diag}(\text{rand}(n))),$$

$C1 = \mathrm{triu}(\mathrm{rand}(n),1) - \mathrm{diag}(1+\mathrm{diag}(\mathrm{rand}(n))),\ C2 = \mathrm{triu}(\mathrm{rand}(n),1) - \mathrm{diag}(1+\mathrm{diag}(\mathrm{rand}(n))),$

$D1 = \mathrm{tril}(\mathrm{rand}(n),1) + \mathrm{diag}(1+\mathrm{diag}(\mathrm{rand}(n))),\ D2 = \mathrm{tril}(\mathrm{rand}(n),1) + \mathrm{diag}(1+\mathrm{diag}(\mathrm{rand}(n))),$

$$E1 = \mathrm{rand}(n), \quad E2 = \mathrm{rand}(n),$$

for $n = 300$. By applying the stated iterative algorithms with the initial matrices $X_i = 0, i = 1,2,3$, we obtain the numerical results reported in Table 3. One can notice that the GPBiCG$(3,1)$_M is more efficient regarding the number of Iters and the CPU time while the CGS_M solver and the GPBiCG$(1,4)$_M solvers are more expensive than the other solvers. The accuracy of the approximations, concerning the TRR values by using the BiCOR_M and CGS_M methods declines to less than half of the TOL. Figure 6 presents the convergence histories for some of the mentioned solvers where the GPBiCG(m,l)_M and the GPBiCG_M methods show fairly attractive convergence behaviors assessed against the other methods. Also, the remaining solvers exhibit irregular convergence behaviors, especially the CGS_M and the CORS_M methods.

The above experiments indicated that the proposed GPBiCG(m,l)_M method has faster convergence rate and higher accuracy than the other stated methods. The obtained numerical results illustrate that Algorithms 3 and 4 are efficient.

6. Conclusions

In summary, by means of the the vectorization operator and the Kronecker product, we have generalized two matrix forms of the GPBiCG(m,l) method that was initially proposed to solve the nonsymmetric linear system problems to obtain the solutions of the general matrix equation (1.1) and the general discrete-time periodic matrix equations (1.2), which include many forms of matrix equations arising in several applications.

Several numerical examples of common linear matrix equations have been presented to demonstrate the accuracy and efficiency of the proposed algorithms assessed against some existing methods. The numerical results have revealed that the proposed method tends to show smoother convergence behaviors, often faster convergence, and can be competitive with the other existing methods. These results of the extended GPBiCG(m,l) are in accordance with the original GPBiCG(m,l) method by Fujino [65] for solving linear systems.

References

[1] B. ANDERSON, P. AGATHOKLIS, E. JURY, et al. *Stability and the matrix Lyapunov equation for discrete 2-dimensional systems.* IEEE Trans. Circuits and Systems, 1986, **33**(3): 261–267.

[2] Tongwen CHEN, B. A. FRANCIS. *Optimal Sampled-Data Control Systems.* Springer, London, UK, 1995.

[3] A. VARGA. *Periodic Lyapunov equations: Some applications and new algorithms.* Internat. J. Control, 1997, **67**(1): 69–88.

[4] Bin ZHOU, Guangren DUAN. *An explicit solution to the matrix equation $AX - XF = BY$.* Linear Algebra Appl., 2005, **402**: 345–366.

[5] Jianzhou LIU, Juan ZHANG, Yu LIU. *New solution bounds for the continuous algebraic Riccati equation.* J. Franklin Inst., 2011, **348**(8): 2128–2141.

[6] Jiaofen LI, Xiyan HU, Lei ZHANG. *New symmetry preserving method for optimal correction of damping and stiffness matrices using measured modes.* J. Comput. Appl. Math., 2010, **234**(5): 1572–1585.

[7] Aiguo WU, Gang FENG, Guangren DUAN, et al. *Iterative solutions to the Kalman-Yakubovich-conjugate matrix equation*. Appl. Math. Comput., 2011, **217**(9): 4427–4438.

[8] B. N. DATTA. *Numerical Methods for Linear Control Systems*. Elsevier Academic Press, San Diego, CA, USA, 2004.

[9] A. KAABI. *On the numerical solution of generalized Sylvester matrix equations*. Bull. Iranian Math. Soc., 2014, **40**(1): 101–113.

[10] V. SIMONCINI. *Computational methods for linear matrix equations*. SIAM Rev., 2016, **58**(3): 377–441.

[11] T. STYKEL. *Low-rank iterative methods for projected generalized Lyapunov equations*. Electron. Trans. Numer. Anal., 2008, **30**: 187–202.

[12] S. BITTANTI, P. COLANERI. *Periodic Systems: Filtering and Control*. Springer-Verlag, London, UK, 2009.

[13] E. K. W. CHU, H. Y. FAN, Wenwei LIN. *Projected generalized discrete-time periodic Lyapunov equations and balanced realization of periodic descriptor systems*. SIAM J. Matrix Anal. Appl., 2007, **29**(3): 982–1006.

[14] R. BARTELS, G. W. STEWART. *Solution of the matrix equation $AX + XB = C$*. Communications of the ACM, 1972, **15**(9): 820–826.

[15] G. H. GOLUB, S. NASH, C. VAN LOAN. *A Hessenberg-Schur method for the problem $AX + XB = C$*. IEEE Trans. Automat. Control, 1979, **24**(6): 909–913.

[16] M. A. RAMADAN, N. M. EL-SHAZLY. B. I. SELIM. *A Hessenberg method for the numerical solutions to types of block Sylvester matrix equations*. Math. Comput. Modelling, 2010, **52**(9-10): 1716–1727.

[17] T. PENZL. *Numerical solution of generalized Lyapunov equations*. Adv. Comput. Math., 1998, **8**(1-2): 33–48.

[18] Qingxi HU, Daizhan CHENG. *The polynomial solution to the Sylvester matrix equation*. Appl. Math. Lett., 2006, **19**(9): 859–864.

[19] Guangren DUAN, Bin ZHOU. *Solution to the second-order Sylvester matrix equation $MVF^2+DVF+KV = BW$*. IEEE Trans. Automat. Control, 2006, **51**(5): 805–809.

[20] Bin ZHOU, Guangren DUAN. *A new solution to the generalized Sylvester matrix equation $AV-EVF = BW$*. Systems Control Lett., 2006, **55**(3): 193–198.

[21] Bin ZHOU, Guangren DUAN. *Solutions to generalized Sylvester matrix equation by Schur decomposition*. Internat. J. Systems Sci., 2007, **38**(5): 369–375.

[22] Feng DING, Tongwen CHEN. *Gradient based iterative algorithms for solving a class of matrix equations*. IEEE Trans. Automat. Control, 2005, **50**(8): 1216–1221.

[23] Feng DING, Tongwen CHEN. *Iterative least-squares solutions of coupled Sylvester matrix equations*. Systems Control Lett., 2005, **54**(2): 95–107.

[24] Feng DING, Tongwen CHEN. *On iterative solutions of general coupled matrix equations*. SIAM J. Control Optim., 2006, **44**(6): 2269–2284.

[25] Feng DING, P. X. LIU and Jie DING. *Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle*. Appl. Math. Comput., 2008, **197**(1): 41–50.

[26] Bin ZHOU, Zhao Yan LI, Guangren DUAN, et al. *Weighted least squares solutions to general coupled Sylvester matrix equations*. J. Comput. Appl. Math., 2009, **224**(2): 759–776.

[27] Aiguo WU, Gang FENG, Guangren DUAN, et al. *Iterative solutions to coupled Sylvester-conjugate matrix equations*. Comput. Math. Appl., 2010, **60**(1): 54–66.

[28] Caiqin SONG, Guoliang CHEN, Linlin ZHAO. *Iterative solutions to coupled Sylvester-transpose matrix equations*. Appl. Math. Model., 2011, **35**(10): 4675–4683.

[29] M. DEHGHAN, M. HAJARIAN. *The generalised Sylvester matrix equations over the generalised bisymmetric and skew-symmetric matrices*. Internat. J. Systems Sci., 2012, **43**(8): 1580–1590.

[30] M. HAJARIAN. *A gradient-based iterative algorithm for generalized coupled Sylvester matrix equations over generalized centro-symmetric matrices*. Transactions of the Institute of Measurement and Control, 2014, **36**(2): 252–259.

[31] F. P. A. BEIK, D. K. SALKUYEH, M. M. MOGHADAM. *Gradient-based iterative algorithm for solving the generalized coupled Sylvester-transpose and conjugate matrix equations over reflexive (anti-reflexive) matrices*. Transactions of the Institute of Measurement and Control, 2014, **36**(1): 99–110.

[32] Yaxin PENG, Xiyan HU, Lei ZHANG. *An iteration method for the symmetric solutions and the optimal approximation solution of the matrix equation $AXB = C$*. Appl. Math. Comput., 2005, **160**(3): 763–777.

[33] Yaxin PENG, Xiyan HU, Lei ZHANG. *An iterative method for symmetric solutions and optimal approximation solution of the system of matrix equations $A_1XB_1 = C_1$, $A_2XB_2 = C_2$*. Appl. Math. Comput., 2006, **183**(2): 1127–1137.

[34] Zhenyun PENG, Yaxin PENG. *An efficient iterative method for solving the matrix equation $AXB + CYD = E$*. Numer. Linear Algebra Appl., 2006, **13**(6): 473–485.

[35] Yaxin PENG, Xiyan HU, Lei ZHANG. *An Iterative Method for Bisymmetric Solutions and Optimal Approximation Solution of $AXB = C$.* In: Third International Conference on Natural Computation (ICNC 2007), IEEE, Haikou, China, Aug. 2007, **4**: 829–832.

[36] Minghui WANG, Xuehan CHENG and Musheng WEI. *Iterative algorithms for solving the matrix equation $AXB + CX^TD = E$.* Appl. Math. Comput., 2007, **187**(2): 622–629.

[37] Guangxin HUANG, Feng YIN, Ke GUO. *An iterative method for the skew-symmetric solution and the optimal approximate solution of the matrix equation $AXB = C$.* J. Comput. Appl. Math., 2008, **212**(2): 231–244.

[38] Fanliang LI, Xiyan HU, Lei ZHANG. *The generalized anti-reflexive solutions for a class of matrix equations* $(BX = C, XD = E)$. Comput. Appl. Math., 2008, **27**(1): 31–46.

[39] Jiaofen LI, Xiyan HU, Xuefeng DUAN, et al. *Iterative method for mirror-symmetric solution of matrix equation $AXB + CYD = E$.* Bull. Iranian Math. Soc., 2010, **36**(2): 35–55.

[40] M. DEHGHAN, M. HAJARIAN. *An iterative algorithm for the reflexive solutions of the generalized coupled Sylvester matrix equations and its optimal approximation.* Appl. Math. Comput., 2008, **202**(2): 571–588.

[41] M. DEHGHAN, M. HAJARIAN. *An iterative method for solving the generalized coupled Sylvester matrix equations over generalized bisymmetric matrices.* Appl. Math. Model., 2010, **34**(3): 639–654.

[42] Xiang WANG, Wuhua WU. *A finite iterative algorithm for solving the generalized (P, Q)-reflexive solution of the linear systems of matrix equations.* Math. Comput. Modelling, 2011, **54**(9-10): 2117–2131.

[43] Aiguo WU, Bin LI, Ying ZHANG, et al. *Finite iterative solutions to coupled Sylvester-conjugate matrix equations.* Appl. Math. Model., 2011, **35**(3): 1065–1080.

[44] M. DEHGHAN, M. HAJARIAN. *Iterative algorithms for the generalized centro-symmetric and central anti-symmetric solutions of general coupled matrix equations.* Engineering Computations, 2012, **29**(5): 528–560.

[45] M. DEHGHAN, M. HAJARIAN. *Finite iterative methods for solving systems of linear matrix equations over reflexive and anti-reflexive matrices.* Bull. Iranian Math. Soc., 2014, **40**(2): 295–323.

[46] M. HAJARIAN. *Matrix iterative methods for solving the Sylvester-transpose and periodic Sylvester matrix equations.* J. Franklin Inst., 2013, **350**(10): 3328–3341.

[47] M. HAJARIAN. *The generalized QMRCGSTAB algorithm for solving Sylvester-transpose matrix equations.* Appl. Math. Lett., 2013, **26**(10): 1013–1017.

[48] M. HAJARIAN. *Extending the GPBiCG algorithm for solving the generalized Sylvester-transpose matrix equation.* International Journal of Control, Automation and Systems, 2014, **12**(6): 1362–1365.

[49] M. HAJARIAN. *New finite algorithm for solving the generalized nonhomogeneous Yakubovich-transpose matrix equation.* Asian J. Control, 2017, **19**(1): 164–172.

[50] M. HAJARIAN. *Convergence properties of BCR method for generalized Sylvester matrix equation over generalized reflexive and anti-reflexive matrices.* Linear Multilinear Algebra, 2018, **66**(10): 1975–1990.

[51] B. I. SELIM, Lei DU, Bo YU. *The GPBiCOR method for solving the general matrix equation and the general discrete-time periodic matrix equations.* IEEE Access, 2018, **6**: 68649–68674.

[52] Shengkun LI, Tingzhu HUANG. *LSQR iterative method for generalized coupled Sylvester matrix equations.* Appl. Math. Model., 2012, **36**(8): 3545–3554.

[53] M. HAJARIAN. *Solving the general coupled and the periodic coupled matrix equations via the extended QMRCGSTAB algorithms.* Comput. Appl. Math., 2014, **33**(2): 349–362.

[54] M. HAJARIAN. *Matrix form of the CGS method for solving general coupled matrix equations.* Appl. Math. Lett., 2014, **34**: 37–42.

[55] F. TOUTOUNIAN, D. KHOJASTEH SALKUYEH, M. MOJARRAB. *LSMR iterative method for general coupled matrix equations.* J. Appl. Math. 2015, Art. ID 562529, 12 pp.

[56] Yajun XIE, Changfeng MA. *The MGPBiCG method for solving the generalized coupled Sylvester-conjugate matrix equations.* Appl. Math. Comput., 2015, **265**: 68–78.

[57] M. HAJARIAN. *Matrix GPBiCG algorithms for solving the general coupled matrix equations.* IET Control Theory Appl., 2015, **9**(1): 74–81.

[58] M. HAJARIAN. *Developing BiCOR and CORS methods for coupled Sylvester-transpose and periodic Sylvester matrix equations.* Appl. Math. Model., 2015, **39**(19): 6073–6084.

[59] Guangbin CAI, Changhua HU. *Solving periodic Lyapunov matrix equations via finite steps iteration.* IET Control Theory Appl., 2012, **6**(13): 2111–2119.

[60] M. HAJARIAN. *Extending the CGLS method for finding the least squares solutions of general discrete-time periodic matrix equations.* Filomat, 2016, **30**(9): 2503–2520.

[61] M. HAJARIAN. *Solving the general Sylvester discrete-time periodic matrix equations via the gradient based iterative method.* Appl. Math. Lett., 2016, **52**: 87–95.

[62] M. HAJARIAN. *Matrix form of biconjugate residual algorithm to solve the discrete-time periodic Sylvester matrix equations.* Asian J. Control, 2018, **20**(1): 49–56.

[63] M. HAJARIAN. *Developing CGNE algorithm for the periodic discrete-time generalized coupled Sylvester matrix equations.* Comput. Appl. Math., 2015, **34**(2): 755–771.

[64] M. HAJARIAN. *Finding solutions for periodic discrete-time generalized coupled Sylvester matrix equations via the generalized BCR method.* Transactions of the Institute of Measurement and Control, 2018, **40**(2): 647–656.

[65] S. FUJINO. *GPBiCG(m, l): A hybrid of BiCGSTAB and GPBiCG methods with efficiency and robustness.* Appl. Numer. Math., 2002, **41**(1): 107–117.

[66] M. R. HESTENES, E. STIEFEL. *Methods of conjugate gradients for solving linear systems.* J. Research Nat. Bur. Standards, 1952, **49**(6): 409–436.

[67] R. FLETCHER. *Conjugate Gradient Methods for Indefinite Systems.* Springer, Berlin, 1976.

[68] P. SONNEVELD. *CGS, A fast Lanczos-type solver for nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput., 1989, **10**(1): 36–52.

[69] H. A. VAN DER VORST. *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput., 1992, **13**(2): 631–644.

[70] M. H. GUTKNECHT. *Variants of BICGSTAB for matrices with complex spectrum.* SIAM J. Sci. Comput., 1993, **14**(5): 1020–1033.

[71] G. L. SLEIJPEN, D. R. FOKKEMA. *BiCGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum.* Electron. Trans. Numer. Anal., 1993, **1**(11): 11–32

[72] S. L. ZHANG. *GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems.* SIAM Journal on Scientific Computing, 1997, **18**(2): 537–551.

[73] Xianming GU, Tingzhu HUANG, B. CARPENTIERI, et al. *A hybridized iterative algorithm of the Bi-CORSTAB and GPBiCOR methods for solving non-Hermitian linear systems.* Comput. Math. Appl., 2015, **70**(12): 3019–3031.

[74] Zhongzhi BAI. *On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equations.* J. Comput. Math., 2011, **29**(2): 185–198.

[75] M. HAJARIAN. *Convergence results of the biconjugate residual algorithm for solving generalized Sylvester matrix equation.* Asian J. Control, 2017, **19**(3): 961–968.