Advances in Difference Equations
a SpringerOpen Journal

RESEARCH Open Access

Developing CRS iterative methods for periodic Sylvester matrix equation

Linjie Chen[1] and Changfeng Ma[1*]

*Correspondence:
macf@fjnu.edu.cn
[1]College of Mathematics and Informatics, Fujian Normal University, Fuzhou, P.R. China

Abstract

In this paper, by applying Kronecker product and vectorization operator, we extend two mathematical equivalent forms of the conjugate residual squared (CRS) method to solve the periodic Sylvester matrix equation

$$A_jX_jB_j + C_jX_{j+1}D_j = E_j \quad \text{for } j = 1, 2, \dots, \lambda.$$

We give some numerical examples to compare the accuracy and efficiency of the matrix CRS iterative methods with other methods in the literature. Numerical results validate that the proposed methods are superior to some existing methods and that equivalent mathematical methods can show different numerical performance.

Keywords: Conjugate residual squared; Iterative method; Periodic Sylvester matrix equation; Kronecker product; Vectorization operator

1 Introduction

We consider the iterative solution of the periodic Sylvester matrix equation

$$A_jX_jB_j + C_jX_{j+1}D_j = E_j \quad \text{for } j = 1, 2, \dots, \tag{1.1}$$

where the coefficient matrices $A_j, B_j, C_j, D_j, E_j \in \mathbf{R}^{m\times m}$ and the solutions $X_j \in \mathbf{R}^{m\times m}$ are periodic with period λ, that is, $A_{j+\lambda} = A_j$, $B_{j+\lambda} = B_j$, $C_{j+\lambda} = C_j$, $D_{j+\lambda} = D_j$, $E_{j+\lambda} = E_j$, and $X_{j+\lambda} = X_j$. The periodic Sylvester matrix equation (1.1) attracts considerable attention because it comes from a variety of fields of control theory and applied mathematics [1–12].

In recent years, many efficient iterative methods have been proposed to solve the periodic Sylvester matrix equation (1.1). For example, Hajarian [13, 14] developed the conjugate gradient squared (CGS), biconjugate gradient stabilized (BiCGSTAB) and biconjugate residual methods for solving the periodic Sylvester matrix equation (1.1). Lv and Zhang [15] proposed a new kind of iterative algorithm for constructing the least square solution for the periodic Sylvester matrix equation. Hajarian [16] studied the biconjugate A-orthogonal residual and conjugate A-orthogonal residual squared (CORS) methods for solving coupled periodic Sylvester matrix equation, and so forth; see [17–28] and the references therein.

As we know, by applying Kronecker product and vectorization operator, some iterative algorithms for solving linear system $Ax = b$ can be extended to solve linear matrix equations. Recently, Sogabe et al. [29] proposed a conjugate residual squared (CRS) method for solving linear systems $Ax = b$ with nonsymmetric coefficient matrix. Independently, Zhang et al. [30] presented another form of CRS method. It can be proved that these CRS methods are mathematically equivalent. Chen and Ma [31] used the matrix CRS iterative method to solve a class of coupled Sylvester-transpose matrix equations. In this work, we obtain a matrix form of the CRS methods for solving the periodic Sylvester matrix equation (1.1).

The rest of this paper is organized as follows. In Sect. 2, we extend the CRS methods to solve the periodic Sylvester matrix equation (1.1). We give some numerical examples and comparison results in Sect. 3. In Sect. 4, we draw a brief conclusion.

Throughout this paper, we use the following notations. The set of all real m-vectors and the set of all $m \times n$ real matrices are denoted by $\mathbf{R}^m$ and $\mathbf{R}^{m\times n}$, respectively. The usual inner product in $\mathbf{R}^m$ is denoted by (u, v) for $u, v \in \mathbf{R}^m$. For a matrix $A \in \mathbf{R}^{m\times n}$, we denote its trace and transpose by $\operatorname{tr}(A)$ and A^T, respectively. The inner product of $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{m\times n}$ is defined by $\langle A, B\rangle = \operatorname{tr}(B^T A)$. Then the norm of a matrix generated by this inner product is the matrix Frobenius norm $\|\cdot\|$. For a matrix $A \in \mathbf{R}^{m\times n}$, the vectorization operator is defined as $\operatorname{vec}(A) = (a_1^T\ a_2^T\ \cdots\ a_n^T)^T$, where a_i is the ith column of A. The Kronecker product of matrices $A = [a_{ij}] \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times q}$ is defined as $A \otimes B = [a_{ij}B] \in \mathbf{R}^{mp\times nq}$. For matrices A, B, and X of appropriate dimensions, we have the following well-known property related to the Kronecker product and vectorization operator:

$$\operatorname{vec}(AXB) = \left(B^T \otimes A\right) \operatorname{vec}(X).$$

2 Matrix forms of the CRS iterative methods

In this section, we first briefly recall the CRS iterative methods for solving a large sparse nonsymmetric linear system $Ax = b$, where $A \in \mathbf{R}^{N\times N}$ and $x, b \in \mathbf{R}^N$. As described in the introduction, the CRS iterative methods are presented in [29] and [30], which are summarized as the following Algorithms 2.1 and 2.2, respectively. For more detail about the CRS methods, see [32–34].

Algorithm 2.1 (The first form of CRS method (CRS1) [29])

1. x_0 is an initial guess; $r_0 = b - Ax_0$; choose r_0^* (for example, $r_0^* = r_0$).
2. Set $e_0 = r_0$, $d_0 = Ae_0$, $\beta_{-1} = 0$. Let $t = A^T r_0^*$.
3. For $n = 0, 1, \ldots$, until convergence **Do**:

$$
\begin{aligned}
& s_n = d_n + \beta_{n-1}(f_{n-1} + \beta_{n-1}s_{n-1}); \\
& \alpha_n = (t, r_n)/(t, s_n); \\
& h_n = e_n - \alpha_n s_n; \\
& f_n = d_n - \alpha_n A s_n; \\
& x_{n+1} = x_n + \alpha_n(e_n + h_n); \\
& r_{n+1} = r_n - \alpha_n(d_n + f_n); \\
& \beta_n = (t, r_{n+1})/(t, r_n); \\
& e_{n+1} = r_{n+1} + \beta_n h_n;
\end{aligned}
$$

$$d_{n+1} = Ar_{n+1} + \beta_n f_n;$$

4. **EndDo**.

Algorithm 2.2 (The second form of CRS method (CRS2) [30])

1. Compute $r_0 = b - Ax_0$; choose r_0^* such that $(Ar_0, r_0^*) \neq 0$ (for example, $r_0^* = r_0$).
2. Set $p_0 = u_0 = r_0$. Let $t = A^T r_0^*$.
3. For $n = 0, 1, \ldots,$ until convergence **Do**:

$$\alpha_n = (t, r_n)/(t, Ap_n);$$

$$q_n = u_n - \alpha_n Ap_n;$$

$$x_{n+1} = x_n + \alpha_n(u_n + q_n);$$

$$r_{n+1} = r_n - \alpha_n A(u_n + q_n);$$

$$\beta_n = (t, r_{n+1})/(t, r_n);$$

$$u_{n+1} = r_{n+1} + \beta_n q_n;$$

$$p_{n+1} = u_{n+1} + \beta_n(q_n + \beta_n p_n);$$

4. **EndDo**.

Let

$$h_n \leftrightarrow q_n, \qquad e_n \leftrightarrow u_n, \qquad s_n \leftrightarrow Ap_n, \qquad d_n \leftrightarrow Au_n, \qquad f_n \leftrightarrow Aq_n.$$

Then we can verify that CRS1 and CRS2 are mathematically equivalent. The CRS methods were proposed mainly to avoid using the transpose of A in the BiCR algorithm and gain a faster convergence for roughly the same computational costs [30]. Indeed, in many cases, the CRS methods converge twice as fast as the BiCR method [33, 35]. On the other hand, the BiCR method can be derived from the preconditioned conjugate residual (CR) method [36]. Furthermore, the CR and conjugate gradient (CG) methods exhibit typically similar convergence [37]. In exact arithmetic, they terminate after a finite number of iterations. In conclusion, we can expect that the CRS methods also terminate after a finite number of iterations in exact arithmetic.

In the following, we want to use the CRS algorithms to solve the periodic Sylvester matrix equations (1.1). For this purpose, we can easily show that the periodic Sylvester matrix equation (1.1) is equivalent to the following generalized Sylvester matrix equation [13]:

$$\mathbf{AXB} + \mathbf{CXD} = \mathbf{E}, \tag{2.1}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & \cdots & 0 & A_1 \\ A_2 & & & 0 \\ & \ddots & & \vdots \\ 0 & & A_\lambda & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & B_2 & & 0 \\ \vdots & & \ddots & \\ 0 & & & B_\lambda \\ B_1 & 0 & \cdots & 0 \end{bmatrix},$$

$$\mathbf{C} = \mathrm{diag}[C_1, C_2, \ldots, C_\lambda], \qquad \mathbf{D} = \mathrm{diag}[D_1, D_2, \ldots, D_\lambda],$$
$$\mathbf{E} = \mathrm{diag}[E_1, E_2, \ldots, E_\lambda], \qquad \mathbf{X} = \mathrm{diag}[X_2, X_3, \ldots, X_\lambda, X_1].$$

Then we need to transform the generalized Sylvester matrix equations (2.1) into a linear system $Ax = b$. We should mention that the following derivation borrows much of that used in [13].

By applying the Kronecker product and vectorization operator we can change the generalized Sylvester matrix equations (2.1) into the following linear system of equations:

$$\left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) \mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{E}). \tag{2.2}$$

Denote

$$A := \mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}, \qquad x := \mathrm{vec}(\mathbf{X}), \qquad b := \mathrm{vec}(\mathbf{E}).$$

Then (2.2) can be written as

$$Ax = b,$$

where $A \in R^{\lambda^2 m^2 \times \lambda^2 m^2}$ and $x, b \in R^{\lambda^2 m^2}$. Then we are in position to present the matrix forms of Algorithms 2.1 and 2.2 for solving the generalized Sylvester matrix equation (2.1), and we just discuss Algorithm 2.1 in detail since the discussion of Algorithm 2.2 is similar.

From Algorithm 2.1 and the linear system of equation (2.2) we have

$$r_0 = b - Ax_0 \to r_0 = \mathrm{vec}(\mathbf{E}) - \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) x_0, \tag{2.3}$$
$$d_0 = Ae_0 \to d_0 = \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) e_0, \tag{2.4}$$
$$t = A^T r_0^* \to t = \left(\mathbf{B} \otimes \mathbf{A}^T + \mathbf{D} \otimes \mathbf{C}^T\right) r_0^*, \tag{2.5}$$
$$As_n \to \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) s_n, \tag{2.6}$$
$$Ar_{n+1} \to \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) r_{n+1}. \tag{2.7}$$

According to (2.3)–(2.7), we define

$$x_n = \mathrm{vec}\big(\mathbf{X}(n)\big), \qquad r_n = \mathrm{vec}\big(\mathbf{R}(n)\big), \qquad s_n = \mathrm{vec}\big(\mathbf{S}(n)\big), \tag{2.8}$$
$$h_n = \mathrm{vec}\big(\mathbf{H}(n)\big), \qquad f_n = \mathrm{vec}\big(\mathbf{F}(n)\big), \qquad e_n = \mathrm{vec}\big(\mathbf{E}(n)\big), \tag{2.9}$$
$$d_n = \mathrm{vec}\big(\mathbf{D}(n)\big), \qquad r_0^* = \mathrm{vec}\big(\mathbf{R}^*(0)\big), \qquad t = \mathrm{vec}(\mathbf{T}), \tag{2.10}$$

where $\mathbf{X}(n), \mathbf{R}(n), \mathbf{S}(n), \mathbf{H}(n), \mathbf{F}(n), \mathbf{E}(n), \mathbf{D}(n), \mathbf{R}^*(0), \mathbf{T} \in \mathbf{R}^{\lambda m \times \lambda m}$ for $n = 0, 1, 2, \ldots$. Substituting (2.8)–(2.10) into (2.3)–(2.7), we get

$$\mathrm{vec}\big(\mathbf{R}(0)\big) = \mathrm{vec}(\mathbf{E}) - \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) \mathrm{vec}\big(\mathbf{X}(0)\big),$$
$$\mathrm{vec}\big(\mathbf{D}(0)\big) = \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) \mathrm{vec}\big(\mathbf{E}(0)\big),$$
$$\mathrm{vec}(\mathbf{T}) = \left(\mathbf{B} \otimes \mathbf{A}^T + \mathbf{D} \otimes \mathbf{C}^T\right) \mathrm{vec}\big(\mathbf{R}^*(0)\big),$$

$$As_n = \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) \operatorname{vec}\left(\mathbf{S}(n)\right),$$

and

$$Ar_{n+1} = \left(\mathbf{B}^T \otimes \mathbf{A} + \mathbf{D}^T \otimes \mathbf{C}\right) \operatorname{vec}\left(\mathbf{R}(n+1)\right).$$

In addition, for the parameters α_n and β_n, we have

$$\alpha_n = \left\langle \mathbf{R}(n), \mathbf{T}\right\rangle / \left\langle \mathbf{V}(n), \mathbf{T}\right\rangle$$

and

$$\beta_n = \left\langle \mathbf{R}(n+1), \mathbf{T}\right\rangle / \left\langle \mathbf{R}(n), \mathbf{T}\right\rangle.$$

From this discussion it follows that the matrix form of CRS1 method for solving the generalized Sylvester matrix equation (2.1) can be constructed as the following Algorithm 2.3. Analogously, the matrix form of CRS1 method for solving the generalized Sylvester matrix equation (2.1) is summarized as Algorithm 2.4.

Algorithm 2.3 (Matrix CRS1 method for solving (2.1))

1. Compute $\mathbf{R}(0) = \mathbf{E} - \mathbf{A}\mathbf{X}(0)\mathbf{B} - \mathbf{C}\mathbf{X}(0)\mathbf{D}$ for an initial guess $\mathbf{X}(0) \in \mathbf{R}^{\lambda m \times \lambda m}$. Set $\mathbf{R}^*(0) = \mathbf{E}(0) = \mathbf{R}(0)$ and $\mathbf{D}(0) = \mathbf{A}\mathbf{E}(0)\mathbf{B} + \mathbf{C}\mathbf{E}(0)\mathbf{D}$. Let $\beta_{-1} = 0$.
2. Set $\mathbf{T} = \mathbf{A}^T\mathbf{R}^*(0)\mathbf{B}^T + \mathbf{C}^T\mathbf{R}^*(0)\mathbf{D}^T$.
3. For $n = 0, 1, \ldots,$ until convergence **Do**:

$$\mathbf{S}(n) = \mathbf{D}(n) + \beta_{n-1}\left(\mathbf{F}(n-1) + \beta_{n-1}\mathbf{S}(n-1)\right);$$

$$\alpha_n = \left\langle \mathbf{R}(n), \mathbf{T}\right\rangle / \left\langle \mathbf{S}(n), \mathbf{T}\right\rangle;$$

$$\mathbf{H}(n) = \mathbf{E}(n) - \alpha_n\mathbf{S}(n);$$

$$\mathbf{F}(n) = \mathbf{D}(n) - \alpha_n\left(\mathbf{A}\mathbf{S}(n)\mathbf{B} + \mathbf{C}\mathbf{S}(n)\mathbf{D}\right);$$

$$\mathbf{X}(n+1) = \mathbf{X}(n) + \alpha_n\left(\mathbf{E}(n) + \mathbf{H}(n)\right);$$

$$\mathbf{R}(n+1) = \mathbf{R}(n) - \alpha_n\left(\mathbf{D}(n) + \mathbf{F}(n)\right);$$

$$\beta_n = \left\langle \mathbf{R}(n+1), \mathbf{T}\right\rangle / \left\langle \mathbf{R}(n), \mathbf{T}\right\rangle;$$

$$\mathbf{E}(n+1) = \mathbf{R}(n+1) + \beta_n\mathbf{H}(n);$$

$$\mathbf{D}(n+1) = \mathbf{A}\mathbf{R}(n+1)\mathbf{B} + \mathbf{C}\mathbf{R}(n+1)\mathbf{D}) + \beta_n\mathbf{F}(n);$$

4. **EndDo**.

Algorithm 2.4 (Matrix CRS2 method for solving (2.1))

1. Compute $\mathbf{R}(0) = \mathbf{E} - \mathbf{A}\mathbf{X}(0)\mathbf{B} - \mathbf{C}\mathbf{X}(0)\mathbf{D}$ for an initial guess $\mathbf{X}(0) \in \mathbf{R}^{\lambda m \times \lambda m}$. Set $\mathbf{R}^*(0) = \mathbf{P}(0) = \mathbf{U}(0) = \mathbf{R}(0)$.
2. Set $\mathbf{T} = \mathbf{A}^T\mathbf{R}^*(0)\mathbf{B}^T + \mathbf{C}^T\mathbf{R}^*(0)\mathbf{D}^T$.
3. For $n = 0, 1, \ldots,$ until convergence **Do**:

$$\mathbf{V}(n) = \mathbf{A}\mathbf{P}(n)\mathbf{B} + \mathbf{C}\mathbf{P}(n)\mathbf{D};$$

$$\alpha_n = \langle \mathbf{R}(n), \mathbf{T}\rangle / \langle \mathbf{V}(n), \mathbf{T}\rangle;$$

$$\mathbf{Q}(n) = \mathbf{U}(n) - \alpha_n \mathbf{V}(n);$$

$$\mathbf{X}(n+1) = \mathbf{X}(n) + \alpha_n \big(\mathbf{U}(n) + \mathbf{Q}(n)\big);$$

$$\mathbf{W}(n) = \mathbf{A}\big(\mathbf{U}(n) + \mathbf{Q}(n)\big)\mathbf{B} + \mathbf{C}\big(\mathbf{U}(n) + \mathbf{Q}(n)\big)\mathbf{D};$$

$$\mathbf{R}(n+1) = \mathbf{R}(n) - \alpha_n \mathbf{W}(n);$$

$$\beta_n = \langle \mathbf{R}(n+1), \mathbf{T}\rangle / \langle \mathbf{R}(n), \mathbf{T}\rangle;$$

$$\mathbf{U}(n+1) = \mathbf{R}(n+1) + \beta_n \mathbf{Q}(n);$$

$$\mathbf{P}(n+1) = \mathbf{U}(n+1) + \beta_n \big(\mathbf{Q}(n) + \beta_n \mathbf{P}(n)\big);$$

4. **EndDo**.

From Algorithms 2.3 and 2.4 by using the equivalent relationships of periodic Sylvester matrix equation (1.1) and generalized Sylvester matrix equation (2.1) we can derive the CRS methods for solving periodic Sylvester matrix equation (1.1) as Algorithms 2.5 and 2.6, respectively.

Algorithm 2.5 (Matrix CRS1 method for solving (1.1))

1. Choose $X_j(0) \in \mathbf{R}^{m\times m}$ for $j = 1, 2, \ldots, \lambda$ and set $X_{\lambda+1}(0) = X_1(0)$.
2. Compute $R_j(0) = E_j - A_j X_j(0) B_j - C_j X_{j+1}(0) D_j$ and set $R_j^*(0) = E_j(0) = R_j(0)$ for $j = 1, 2, \ldots, \lambda$. Let $E_{\lambda+1}(0) = E_1(0)$ and $R_{\lambda+1}^*(0) = R_1^*(0)$. Set $D_j(0) = A_j E_j(0) B_j + C_j E_{j+1}(0) D_j$. Let $\beta_{-1} = 0$.
3. Set $T_j = A_j^T R_j^*(0) B_j^T + C_j^T R_{j+1}^*(0) D_j^T$ for $j = 1, 2, \ldots, \lambda$.
4. For $n = 0, 1, \ldots,$ until convergence **Do**:

$$S_j(n) = D_j(n) + \beta_{n-1}\big(F_j(n-1) + \beta_{n-1} S_j(n-1)\big) \quad \text{for } j = 1, 2, \ldots, \lambda.$$

Let $S_{\lambda+1}(n) = S_1(n)$;

$$\alpha_n = \left(\sum_{j=1}^{\lambda} \langle R_j(n), T_j\rangle\right) \Big/ \left(\sum_{j=1}^{\lambda} \langle S_j(n), T_j\rangle\right);$$

$$H_j(n) = E_j(n) - \alpha_n S_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$F_j(n) = D_j(n) - \alpha_n (A_j S_j(n) B_j + C_j S_{j+1}(n) D_j \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$X_j(n+1) = X_j(n) + \alpha_n \big(E_j(n) + H_j(n)\big) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$R_j(n+1) = R_j(n) - \alpha_n \big(D_j(n) + F_j(n)\big) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

Let $R_{\lambda+1}(n+1) = R_1(n+1)$;

$$\beta_n = \left(\sum_{j=1}^{\lambda} \langle R_j(n+1), T_j\rangle\right) \Big/ \left(\sum_{j=1}^{\lambda} \langle R_j(n), T_j\rangle\right);$$

$$E_j(n+1) = R_j(n+1) + \beta_n H_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$D_j(n+1) = A_j R_j(n+1) B_j + C_j R_{j+1}(n+1) D_j) + \beta_n F_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

5. **EndDo**.

Algorithm 2.6 (Matrix CRS2 method for solving (1.1))

1. Choose $X_j(0) \in \mathbf{R}^{m\times m}$ for $j = 1, 2, \ldots, \lambda$ and set $X_{\lambda+1}(0) = X_1(0)$.
2. Compute $R_j(0) = E_j - A_j X_j(0) B_j - C_j X_{j+1}(0) D_j$ and set $R_j^*(0) = P_j(0) = U_j(0) = R_j(0)$ for $j = 1, 2, \ldots, \lambda$. Let $R_{\lambda+1}^*(0) = R_1^*(0)$.
3. Set $T_j = A_j^T R_j^*(0) B_j^T + C_j^T R_{j+1}^*(0) D_j^T$ for $j = 1, 2, \ldots, \lambda$.
4. For $n = 0, 1, \ldots,$ until convergence **Do**:

$$\text{Let } P_{\lambda+1}(n) = P_1(n);$$

$$V_j(n) = A_j P_j(n) B_j + C_j P_{j+1}(n) D_j \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$\alpha_n = \left(\sum_{j=1}^{\lambda} \langle R_j(n), T_j \rangle\right) \Big/ \left(\sum_{j=1}^{\lambda} \langle V_j(n), T_j \rangle\right);$$

$$Q_j(n) = U_j(n) - \alpha_n V_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$X_j(n+1) = X_j(n) + \alpha_n \big(U_j(n) + Q_j(n)\big) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$\text{Let } U_{\lambda+1}(n) = U_1(n) \quad \text{and} \quad Q_{\lambda+1}(n) = Q_1(n);$$

$$W_j(n) = A_j \big(U_j(n) + Q_j(n)\big) B_j + C_j \big(U_{j+1}(n) + Q_{j+1}(n)\big) D_j \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$R_j(n+1) = R_j(n) - \alpha_n W_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$\beta_n = \left(\sum_{j=1}^{\lambda} \langle R_j(n+1), T_j \rangle\right) \Big/ \left(\sum_{j=1}^{\lambda} \langle R_j(n), T_j \rangle\right);$$

$$U_j(n+1) = R_j(n+1) + \beta_n Q_j(n) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

$$P_j(n+1) = U_j(n+1) + \beta_n \big(Q_j(n) + \beta_n P_j(n)\big) \quad \text{for } j = 1, 2, \ldots, \lambda;$$

5. **EndDo**.

Based on the earlier discussion, we know that Algorithms 2.5 and 2.6 are just the matrix forms of the original CRS method. Hence, generally speaking, Algorithms 2.5 and 2.6 have the same properties as Algorithms 2.1 and 2.2. For instance, in exact arithmetic, Algorithms 2.5 and 2.6 will also terminate after a finite number of iterations.

3 Numerical experiments

In this section, we present two numerical examples to show the accuracy and efficiency of the proposed methods. We compare the performances of CRS methods to those of the CGS, BiCGSTAB [13], and CORS [16] methods.

In our experiments, all runs are started from the zero initial guess and implemented in MATLAB(R2014b) with a machine precision 10^{-16} on a personal computer with Intel(R) Core(TM) i7-6500U CPU 2.50 GHz 2.60 GHz, 16.0 GB memory.

Example 3.1 ([13]) Consider the periodic Sylvester matrix equation

$$X_j + C_j X_{j+1} D_j = E_j \quad \text{for } j = 1, 2$$

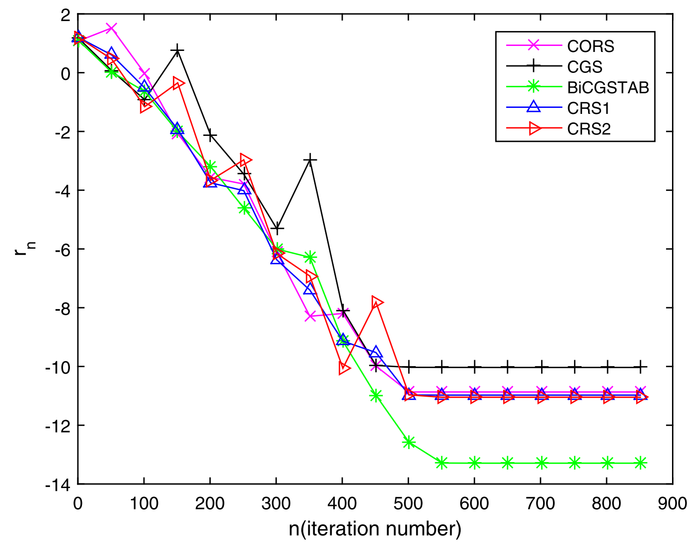

Figure 1 The residual for Example 3.1

with parameters

$$C_1 = \operatorname{tril}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(2 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$D_1 = \operatorname{triu}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(1.75 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$C_2 = \operatorname{triu}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(1.75 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$D_2 = \operatorname{tril}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(2 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$E_1 = E_2 = \operatorname{rand}(m,m).$$

In this example, we set $m = 20$. The numerical results are shown in Fig. 1, where

$$r_n = \log_{10}\sqrt{\bigl\|E_1 - X_1(n) - C_1X_2(n)D_1\bigr\|^2 + \bigl\|E_2 - X_2(n) - C_2X_1(n)D_2\bigr\|^2}.$$

From Fig. 1 we find that the CRS1 and CRS2 methods are superior to the CGS and CORS methods, and the BiCGSTAB method is the best among them for Example 3.1. In addition, the residual history of the CRS1 method seems smoother than that of the CRS2 method.

Example 3.2 ([16]) Consider the periodic Sylvester matrix equations

$$X_j + C_jX_{j+1}D_j = E_j, \quad \text{for } j = 1,2$$

with parameters

$$C_1 = -\operatorname{triu}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(1.75 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$D_1 = \operatorname{tril}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(1.25 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$
$$C_2 = \operatorname{triu}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(1.75 + \operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr),$$

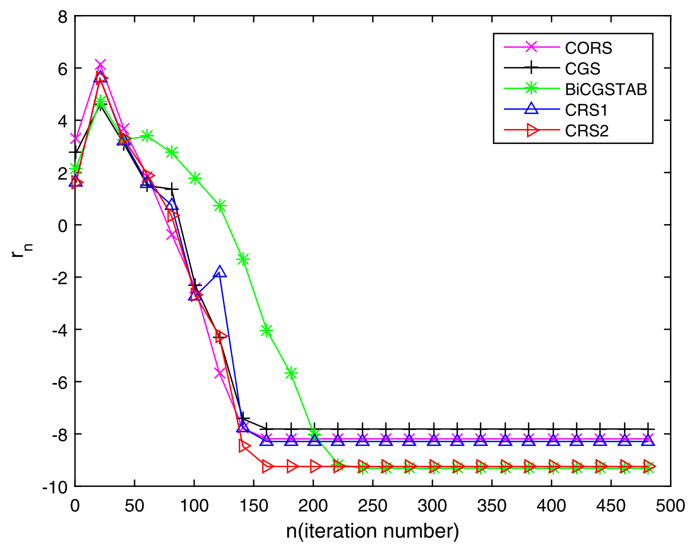

Figure 2 The residual for Example 3.2

$$\begin{aligned} D_2 &= \operatorname{tril}\bigl(\operatorname{rand}(m,m),1\bigr) + \operatorname{diag}\bigl(2+\operatorname{diag}\bigl(\operatorname{rand}(m)\bigr)\bigr), \\ E_1 &= E_2 = \operatorname{rand}(m,m). \end{aligned}$$

In this example, let $m = 40$. The numerical results are shown in Fig. 2, where

$$r_n = \log_{10}\sqrt{\bigl\|E_1 - X_1(n) - C_1X_2(n)D_1\bigr\|^2 + \bigl\|E_2 - X_2(n) - C_2X_1(n)D_2\bigr\|^2}.$$

From Fig. 2, for Example 3.2, we see that the CRS2 method is the best one among the five methods mentioned. The BiCGSTAB method can achieve higher accuracy than the CGS, CORS, and CRS1 methods.

4 Conclusions

In this paper, we present two matrix forms of the CRS iterative method for solving the periodic Sylvester matrix equation (1.1). Numerical examples and comparison with the CGS, BiCGSTAB, and CORS methods have illustrated that the CRS methods can work quite well in some situations. In addition, numerical results show that the CRS1 and CRS2 methods show different numerical performance, though they are mathematically equivalent.

Acknowledgements

The authors deeply thank the anonymous referees for helping to improve the original manuscript by valuable suggestions.

Funding

This work is supported by National Key Research and Development Program of China (No. 2018YFC0603500) and National Science Foundation of China (Nos. 41725017, 41590864).

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

Both authors contributed equally to this work. Both authors read and approved the final manuscript.

Publisher's Note

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

Received: 9 April 2018 Accepted: 21 February 2019 Published online: 04 March 2019

References

1. Varga, A.: Periodic Lyapunov equations: some applications and new algorithms. Int. J. Control **67**, 69–88 (1997)
2. Chu, E.K.W., Fan, H.-Y., Lin, W.-W.: Projected generalized discrete-time periodic Lyapunov equations and balanced realization of periodic descriptor systems. SIAM J. Matrix Anal. Appl. **29**, 982–1006 (2017)
3. Liu, C., Peng, Y.J.: Stability of periodic steady-state solutions to a non-isentropic Euler–Maxwell system. Z. Angew. Math. Phys. **68**, 105 (2017)
4. Zheng, X.X., Shang, Y.D., Peng, X.M.: Orbital stability of periodic traveling wave solutions to the generalized Zakharov equations. Acta Math. Sci. **37B**, 1–21 (2017)
5. Zheng, X.X., Shang, Y.D., Di, H.F.: The time-periodic solutions to the modified Zakharov equations with a quantum correction. Mediterr. J. Math. **14**, 152 (2017)
6. Tian, H.H., Han, M.A.: Bifurcation of periodic orbits by perturbing high-dimensional piecewise smooth integrable systems. J. Differ. Equ. **263**(11), 7448–7474 (2017)
7. Liu, B.M., Liu, L.S., Wu, Y.H.: Existence of nontrivial periodic solutions for a nonlinear second order periodic boundary value problem. Nonlinear Anal. **72**(7–8), 3337–3345 (2010)
8. Hao, X.N., Liu, L.S., Wu, Y.H.: Existence and multiplicity results for nonlinear periodic boundary value problems. Nonlinear Anal. **72**(9–10), 3635–3642 (2010)
9. Liu, A.J., Chen, G.L.: On the Hermitian positive definite solutions of nonlinear matrix equation $X^s + \sum_{i=1}^{m} A_i^* X^{-t_i} A_i = Q$. Appl. Math. Comput. **243**, 950–959 (2014)
10. Liu, A.J., Chen, G.L., Zhang, X.Y.: A new method for the bisymmetric minimum norm solution of the consistent matrix equations $A_1XB_1 = C_1$, $A_2XB_2 = C_2$. J. Appl. Math. **2013**, Article ID 125687 (2013)
11. Stykel, T.: Low-rank iterative methods for projected generalized Lyapunov equations. Electron. Trans. Numer. Anal. **30**, 187–202 (2008)
12. Bittanti, S., Colaneri, P.: Periodic Systems: Filtering and Control. Springer, London (2009)
13. Hajarian, M.: Matrix iterative methods for solving the Sylvester-transpose and periodic Sylvester matrix equations. J. Franklin Inst. **350**, 3328–3341 (2013)
14. Hajarian, M.: Matrix form of biconjugate residual algorithm to solve the discrete-time periodic Sylvester matrix equations. Asian J. Control **20**, 49–56 (2018)
15. Lv, L.-L., Zhang, Z., Zhang, L., Wang, W.-S.: An iterative algorithm for periodic Sylvester matrix equations. J. Ind. Manag. Optim. **14**, 413–425 (2018)
16. Hajarian, M.: Developing BiCOR and CORS methods for coupled Sylvester-transpose and periodic Sylvester matrix equations. Appl. Math. Model. **39**, 6073–6084 (2015)
17. Kressner, D.: Large periodic Lyapunov equations: algorithms and applications. In: Proc. ECC03, Cambridge, UK, pp. 951–956 (2003)
18. Hajarian, M.: Solving the general Sylvester discrete-time periodic matrix equations via the gradient based iterative method. Appl. Math. Lett. **52**, 87–95 (2016)
19. Hajarian, M.: Gradient based iterative algorithm to solve general coupled discrete-time periodic matrix equations over generalized reflexive matrices. Math. Model. Anal. **21**, 533–549 (2016)
20. Hajarian, M.: A finite iterative method for solving the general coupled discrete-time periodic matrix equations. Circuits Syst. Signal Process. **34**, 105–125 (2015)
21. Hajarian, M.: Developing CGNE algorithm for the periodic discrete-time generalized coupled Sylvester matrix equations. Comput. Appl. Math. **34**, 755–771 (2015)
22. Hajarian, M.: Convergence analysis of the MCGNR algorithm for the least squares solution group of discrete-time periodic coupled matrix equations. Trans. Inst. Meas. Control **39**, 29–42 (2017)
23. Cai, G.-B., Hu, C.-H.: Solving periodic Lyapunov matrix equations via finite steps iteration. IET Control Theory Appl. **6**, 2111–2119 (2012)
24. Lv, L., Zhang, Z., Zhang, L.: A periodic observers synthesis approach for LDP systems based on iteration. IEEE Access **6**, 8539–8546 (2018)
25. Lv, L., Zhang, Z.: Finite iterative solutions to periodic Sylvester matrix equations. J. Franklin Inst. **354**(5), 2358–2370 (2017)
26. Lv, L., Zhang, Z., Zhang, L.: A parametric poles assignment algorithm for second-order linear periodic systems. J. Franklin Inst. **354**, 8057–8071 (2017)
27. Lv, L., Zhang, L.: Robust stabilization based on periodic observers for LDP systems. J. Comput. Anal. Appl. **20**, 487–498 (2016)
28. Lv, L., Zhang, L.: On the periodic Sylvester equations and their applications in periodic Luenberger observers design. J. Franklin Inst. **353**, 1005–1018 (2016)
29. Sogabe, T., Fujino, S., Zhang, S.-L.: A product-type Krylov subspace method based on conjugate residual method for nonsymmetric coefficient matrices. Trans. IPSJ **48**, 11–21 (2007)
30. Zhang, L.-T., Zuo, X.-Y., Gu, T.-X., Huang, T.-Z., Yue, J.-H.: Conjugate residual squared method and its improvement for non-symmetric linear systems. Int. J. Comput. Math. **87**, 1578–1590 (2010)
31. Chen, C.-R., Ma, C.-F.: A matrix CRS iterative method for solving a class of coupled Sylvester-transpose matrix equations. Comput. Math. Appl. **74**, 1223–1231 (2017)
32. Zhang, L.-T., Huang, T.-Z., Gu, T.-X., Zuo, X.-Y.: An improved conjugate residual squared algorithm suitable for distributed parallel computing. Microelectron. Comput. **25**, 12–14 (2008) (in Chinese)
33. Zhao, J., Zhang, J.-H.: A smoothed conjugate residual squared algorithm for solving nonsymmetric linear systems. In: 2009 Second International Conference on Information and Computing Science, ICIC, vol. 3, pp. 364–367 (2009)
34. Zuo, X.-Y., Zhang, L.-T., Gu, T.-X.: An improved generalized conjugate residual squared algorithm suitable for distributed parallel computing. J. Comput. Appl. Math. **271**, 285–294 (2014)

35. Sogabe, T., Zhang, S.-L.: Extended conjugate residual methods for solving nonsymmetric linear systems. Numerical Linear Algebra and Optimization 88–99 (2003)
36. Sogabe, T., Sugihara, M., Zhang, S.-L.: An extension of the conjugate residual method to nonsymmetric linear systems. J. Comput. Appl. Math. **226**, 103–113 (2009)
37. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM, Philadelphia (2003)