Research Article

A Parallel Preconditioned Modified Conjugate Gradient Method for Large Sylvester Matrix Equation

Junxia Hou, Quanyi Lv, and Manyu Xiao

Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an 710129, China

Correspondence should be addressed to Manyu Xiao; manyuxiao@gmail.com

Received 2 December 2013; Accepted 15 January 2014; Published 26 March 2014

Academic Editor: Balaji Raghavan

Copyright © 2014 Junxia Hou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Computational effort of solving large-scale Sylvester equations $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$ is frequently hindered in dealing with many complex control problems. In this work, a parallel preconditioned algorithm for solving it is proposed based on combination of a parameter iterative preconditioned method and modified form of conjugate gradient (MCG) method. Furthermore, Schur's inequality and modified conjugate gradient method are employed to overcome the involved difficulties such as determination of parameter and calculation of inverse matrix. Several numerical results finally show that high performance of proposed parallel algorithm is obtained both in convergent rate and in parallel efficiency.

1. Introduction

Sylvester equations play vital roles in a number of applications such as control theory [1], matrix eigendecompositions [2], model reduction [3–5], and image processing [6]. Particularly, when solving parabolic and elliptic PDEs by the discretization method, the issue of solving a large scale Sylvester matrix equation is often encountered. Nowadays, there are two types of methods for solving these matrix equations: the first common one is the standard direct method proposed by Bartels and Stewart [7]. However, this is not applicable in large-scale settings. In order to overcome this limitation, the other concerning iterative form is discussed such as a few simple iterative schemes proposed by Ding and Chen [8]. ADI method was first introduced by Peaceman and Rachford [9] for solving parabolic and elliptic PDEs and was later adapted solving the Sylvester equation by Wachspress [10]. Although using ADI method needs to choose optimal parameters, it is too difficult to find them for a large scale problem. Therefore, the parameter iterative method was first proposed in [11], and the available choice of parameter was also given for a large scale problem. Recently, a new effective method based on Krylov subspace [12–14] is discussed. In particular, conjugate gradient (CG) method [15] is its essential one, which has the advantages of small store, small computing, and the fact

that it is suitable for parallel. But it is only applied for the case that the coefficient matrix is positive definite symmetric matrices. So we can choose the modified conjugate gradient method (MCG) [16, 17] in other cases. The convergence speed of MCG has been closely related to the singular values of the coefficient matrix, and the more concentrated the singular values are, the better the convergence speed of MCG is. In order to enhance the speed of convergence, the idea of preconditioning [18] is introduced. By this way, it will concentrate the singular values of the coefficient matrix. All of these brought about a need for the development of efficient parallel algorithm as a computational time is prohibitive, which motivates us to propose a parallel preconditioned algorithm for solving large Sylvester matrix equation.

The remainder of the paper is organized as follows. In Section 2, there is the introduction of MCG algorithm for solving large matrix equation $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$, and then our proposed preconditioned MCG algorithm is presented in Section 3, followed by the implementation of parallel algorithm (in Section 4). Convergence Analysis is given in Section 5. Finally, some numerical examples are discussed with the results analysis and comparison, followed by concluding comments.

Throughout this paper we use the following notations. Let $\mathbf{A}$ and $\mathbf{B}$ stand for two $n \times n$ matrices and $\mathbf{A}^T$, $\rho(\mathbf{A})$,

and $\mathbf{I}$ represent the transpose of matrix $\mathbf{A}$, spectral radius of matrix, $\mathbf{A}$ and the $n \times n$ identity matrix, respectively. At the same time, we define the inner product as $[\mathbf{A}, \mathbf{B}] = \text{tr}(\mathbf{A}^T \mathbf{B})$; then the matrix norm of $\mathbf{A}$ induced by the inner product is Frobenius norm and denoted by $\|\mathbf{A}\|$; that is, $\|\mathbf{A}\| = \sqrt{[\mathbf{A}, \mathbf{A}]} = \sqrt{\text{tr}(\mathbf{A}^T \mathbf{A})}$.

2. Introduction of MCG Algorithm for Solving $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$

An $n \times n$ Sylvester matrix equation takes the form

$$\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}, \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times n}$, and $\mathbf{F} \in \mathbb{R}^{n \times n}$ are given matrices, and $\mathbf{X} \in \mathbb{R}^{n \times n}$ is unknown matrix. MCG algorithm [15, 19, 20] can be expressed as follows.

Algorithm 1. (1) Choose initial vector $\mathbf{X}^{(0)} \in \mathbb{R}^{n \times n}$; let $k := 0$, and calculate

$$\begin{aligned} \mathbf{R}^{(0)} &= -\mathbf{F} - \mathbf{AX}^{(0)} - \mathbf{X}^{(0)}\mathbf{B}, \\ \mathbf{Q}^{(0)} &= \mathbf{A}^T \mathbf{R}^{(0)} + \mathbf{R}^{(0)} + \mathbf{R}^{(0)} \mathbf{B}^T, \quad \mathbf{Z}^{(0)} = \mathbf{Q}^{(0)}. \end{aligned} \quad (2)$$

(2) If $\mathbf{R}^{(k)} = \mathbf{O}$, or $\mathbf{R}^{(k)} \neq \mathbf{O}$ but $\mathbf{Z}^{(k)} = \mathbf{O}$ stop; otherwise, calculate

$$\gamma_k = \frac{[\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]}{[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)}]}, \quad \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \gamma_k \mathbf{Z}^{(k)}. \quad (3)$$

(3) Then perform

$$\begin{aligned} \mathbf{R}^{(k+1)} &= -\mathbf{F} - \mathbf{AX}^{(k+1)} - \mathbf{X}^{(k+1)}\mathbf{B}, \\ \mathbf{Q}^{(k+1)} &= \mathbf{A}^T \mathbf{R}^{(k+1)} + \mathbf{R}^{(k+1)} + \mathbf{R}^{(k+1)} \mathbf{B}^T. \end{aligned} \quad (4)$$

Furthermore, the convergent condition of MCG algorithm referred in [15, 19, 20] is illustrated as Theorem 2.

Theorem 2. Assume that $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$ is consistent; then, for any arbitrary initial matrix $\mathbf{X}^{(0)}$, a solution of $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$ can be obtained with finite iteration steps in the absence of round-off error.

It is important to accelerate the speed of the convergence. Form the matrix equation $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$, we know that the speed of convergence depends on the singular values centralization of the coefficient matrix $(\mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{B}^T)$. With the point of theorem, it is difficult to calculate its singular values. In order to easily enhance the speed of convergence, we propose a preconditioned MCG algorithm mentioned in the following section.

3. Proposed Preconditioned MCG Algorithm

In order to choose preconditioner for MCG, we introduce the parameter iterative method in [11] for solving (1), which is formed as follows:

$$\mathbf{X}^{(k+1)} = \tilde{\mathbf{A}}\mathbf{X}^{(k)}\tilde{\mathbf{B}} + \tilde{\mathbf{F}}, \quad (5)$$

where

$$\begin{aligned} \tilde{\mathbf{A}} &= (\alpha\mathbf{A} - \mathbf{I})^{-1}(\alpha\mathbf{A} + \mathbf{I}) = \mathbf{I} + 2(\alpha\mathbf{A} - \mathbf{I})^{-1}, \\ \tilde{\mathbf{B}} &= (\alpha\mathbf{B} - \mathbf{I})^{-1}(\alpha\mathbf{B} + \mathbf{I}) = \mathbf{I} + 2(\alpha\mathbf{B} - \mathbf{I})^{-1}, \\ \tilde{\mathbf{F}} &= 2\alpha(\alpha\mathbf{A} - \mathbf{I})^{-1}\mathbf{F}(\alpha\mathbf{B} - \mathbf{I})^{-1}, \end{aligned} \quad (6)$$

and α is a parameter. The speed of convergence depends on the value of $\rho(\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}})^T$, since $\tilde{\mathbf{A}} = (\alpha\mathbf{A} - \mathbf{I})^{-1}(\alpha\mathbf{A} + \mathbf{I})$, $\tilde{\mathbf{B}} = (\alpha\mathbf{B} - \mathbf{I})^{-1}(\alpha\mathbf{B} + \mathbf{I})$. It is much simple to obtain the best convergence effect after decision of an optimal value of α . However, two inverse matrices of $\alpha\mathbf{A} + \mathbf{I}$ and $\alpha\mathbf{B} + \mathbf{I}$ must be solved to get the matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$. Let $\mathbf{Y}_1 = (\alpha\mathbf{A} - \mathbf{I})^{-1}$ and $\mathbf{Y}_2 = (\alpha\mathbf{B} - \mathbf{I})^{-1}$; we can find $\mathbf{Y}_1$ and $\mathbf{Y}_2$ by solving the matrix equations $(\alpha\mathbf{A} - \mathbf{I})\mathbf{Y}_1 = \mathbf{I}$ and $(\alpha\mathbf{B} - \mathbf{I})\mathbf{Y}_2 = \mathbf{I}$.

Next subsection will give the methods determining the parameter α and solving the matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{F}}$, respectively.

3.1. Technique for Getting the Parameter α and the Inverse of Matrix $(\alpha\mathbf{A} - \mathbf{I})$

3.1.1. Determination of the Parameter α . If the negative real numbers $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ are the eigenvalues of $\mathbf{A}$ and the negative real numbers $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$ are the eigenvalues of $\mathbf{B}$, based on the optimal parameter measurement in [11], we have either the optimal parameter $\alpha = 1/\sqrt{\lambda_1 \lambda_n}$ or $\alpha = 1/\sqrt{\mu_1 \mu_n}$.

On basis of Schur's inequality in [21], we determine α approximate value.

Theorem 3 (Schur's inequality). Let $\mathbf{A} = (a_{rs}) \in \mathbb{C}^{n \times n}$, and let the eigenvalues of $\mathbf{A}$ be $\lambda_1, \lambda_2, \dots, \lambda_n$; then it holds that:

$$\sum_{r=1}^n |\lambda_r|^2 \leq \sum_{r,s=1}^n |a_{rs}|^2 = \|\mathbf{A}\|_F^2. \quad (7)$$

If $\lambda_1, \lambda_2, \dots, \lambda_n$ satisfy $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$, from Schur's inequality, one has

$$|\lambda_1|^2 \geq \frac{\|\mathbf{A}\|_F^2}{n}, \quad |\lambda_n|^2 \leq \frac{\|\mathbf{A}\|_F^2}{n}; \quad (8)$$

then we choose either $\alpha = \sqrt{n}/\|\mathbf{A}\|_F$ or $\alpha = \sqrt{n}/\|\mathbf{B}\|_F$; it is very easy for us to find α .

3.1.2. Algorithm for Solving Matrix Equation $\alpha(\mathbf{A} - \mathbf{I})\mathbf{Y}_1 = \mathbf{I}$. We use the MCG algorithm to solve the matrix equations $(\alpha\mathbf{A} - \mathbf{I})\mathbf{Y}_1 = \mathbf{I}$ and $(\alpha\mathbf{B} - \mathbf{I})\mathbf{Y}_2 = \mathbf{I}$. The specific algorithm for it will be given as follows.

Algorithm 4. (1) Give an initial matrix $\mathbf{Y}_1^{(0)} \in \mathbb{R}^{n \times n}$, let $k := 0$ calculate

$$\begin{aligned} \mathbf{R}^{(0)} &= \mathbf{I} - (\alpha\mathbf{A} - \mathbf{I})\mathbf{Y}_1^{(0)}, \quad \mathbf{Q}^{(0)} = (\alpha\mathbf{A} - \mathbf{I})^T \mathbf{R}^{(0)}, \\ \mathbf{Z}^{(0)} &= \mathbf{Q}^{(0)}. \end{aligned} \quad (9)$$

(2) If $\mathbf{R}^{(k)} = \mathbf{O}$, or $\mathbf{R}^{(k)} \neq \mathbf{O}$ but $\mathbf{Z}^{(k)} = \mathbf{O}$ stop; otherwise, do the form

$$\gamma_k = \frac{[\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]}{[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)}]}, \quad \mathbf{Y}_1^{(k+1)} = \mathbf{Y}_1^{(k)} + \gamma_k \mathbf{Z}^{(k)}. \quad (10)$$

(3) Then perform

$$\begin{aligned} \mathbf{R}^{(k+1)} &= \mathbf{I} - (\alpha \mathbf{A} - \mathbf{I}) \mathbf{Y}_1^{(k+1)}, & \mathbf{Q}^{(k+1)} &= (\alpha \mathbf{A} - \mathbf{I})^T \mathbf{R}^{(k+1)}, \\ \eta_k &= \frac{[\mathbf{R}^{(k+1)}, \mathbf{R}^{(k+1)}]}{[\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]}, & \mathbf{Z}^{(k+1)} &= \mathbf{Q}^{(k+1)} + \eta_k \mathbf{Z}^{(k)}. \end{aligned} \quad (11)$$

(4) Let $k := k + 1$ and turn back to (2).

Through Algorithm 4, we can get the matrix $(\alpha \mathbf{A} - \mathbf{I})^{-1}$. In similar manner, the inverse of $\alpha \mathbf{B} - \mathbf{I}$ can be obtained. Therefore, the matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{F}}$ can be obtained.

3.2. Algorithm for Solving the New Matrix Equation $\tilde{\mathbf{A}}\mathbf{X}\tilde{\mathbf{B}} - \mathbf{X} + \tilde{\mathbf{F}} = \mathbf{O}$. We still use MCG algorithm for solving the matrix equation $\tilde{\mathbf{A}}\mathbf{X}\tilde{\mathbf{B}} - \mathbf{X} + \tilde{\mathbf{F}} = \mathbf{O}$, which is obtained from the parameter iterative method of the above introduced, for solving (5). Thereby the precondition MCG is developed; it can be expressed as follows.

Algorithm 5. (1) Given initial matrix $\mathbf{X}^{(0)} \in \mathbb{R}^{n \times n}$, calculate

$$\begin{aligned} \mathbf{R}^{(0)} &= -\tilde{\mathbf{F}} + \mathbf{X}^{(0)} - \tilde{\mathbf{A}}\mathbf{X}^{(0)}\tilde{\mathbf{B}}, & \mathbf{Q}^{(0)} &= \tilde{\mathbf{A}}^T \mathbf{R}^{(0)} \tilde{\mathbf{B}}^T - \mathbf{R}^{(0)}; \\ \mathbf{Z}^{(0)} &= \mathbf{Q}^{(0)}. \end{aligned} \quad (12)$$

(2) For $k = 0, 1, \dots$, calculate

Step 1: If $\|\mathbf{R}^{(k)}\| < \varepsilon$, or $\|\mathbf{R}^{(k)}\| > \varepsilon$ but $\|\mathbf{Z}^{(k)}\| < \varepsilon$ stop; otherwise, turn to Step 2;

Step 2: $\gamma_k = [\mathbf{R}^{(k)}, \mathbf{R}^{(k)}] / [\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)}]$;

Step 3: $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \gamma_k \mathbf{Z}^{(k)}$;

Step 4: $\mathbf{R}^{(k+1)} = -\tilde{\mathbf{F}} + \mathbf{X}^{(k+1)} - \tilde{\mathbf{A}}\mathbf{X}^{(k+1)}\tilde{\mathbf{B}}$, $\mathbf{Q}^{(k+1)} = \tilde{\mathbf{A}}^T \mathbf{R}^{(k+1)} \tilde{\mathbf{B}}^T - \mathbf{R}^{(k+1)}$;

Step 5: $\eta_k = [\mathbf{R}^{(k+1)}, \mathbf{R}^{(k+1)}] / [\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]$;

Step 6: $\mathbf{Z}^{(k+1)} = \mathbf{Q}^{(k+1)} + \eta_k \mathbf{Z}^{(k)}$.

From every step of the previously mentioned algorithms, we can find that there are only matrix multiplication, addition, and inner product, which shows that the algorithm has very good parallelism. Furthermore, at each iteration, amount of calculation with preconditioned modified conjugate gradient method is the same as that with modified conjugate gradient method. There is only computational cost on obtaining the inverse of two matrices. The numerical results will also confirm its advantage both in convergent rate and parallel efficiency.

4. Parallel Implementation of Proposed Algorithm

For describing convenience, let the processor number be p , $n = pl$, where l is integer; p_i ($i = 1, 2, \dots, p$) represents i th processor. And mark

$$\begin{aligned} \mathbf{A} &= (\mathbf{A}_1^T, \mathbf{A}_2^T, \dots, \mathbf{A}_p^T)^T, & \tilde{\mathbf{A}} &= (\tilde{\mathbf{A}}_1^T, \tilde{\mathbf{A}}_2^T, \dots, \tilde{\mathbf{A}}_p^T)^T, \\ \tilde{\mathbf{C}} &= (\tilde{\mathbf{C}}_1, \tilde{\mathbf{C}}_2, \dots, \tilde{\mathbf{C}}_p), \\ \mathbf{B} &= (\mathbf{B}_1^T, \mathbf{B}_2^T, \dots, \mathbf{B}_p^T)^T, & \tilde{\mathbf{B}} &= (\tilde{\mathbf{B}}_1^T, \tilde{\mathbf{B}}_2^T, \dots, \tilde{\mathbf{B}}_p^T)^T, \\ \mathbf{I} &= (\mathbf{I}_1^T, \mathbf{I}_2^T, \dots, \mathbf{I}_p^T)^T, \\ (\alpha \mathbf{A} - \mathbf{I})^{-1} &= (\mathbf{D}_1^T, \mathbf{D}_2^T, \dots, \mathbf{D}_p^T)^T, \\ (\alpha \mathbf{B} - \mathbf{I})^{-1} &= (\mathbf{E}_1^T, \mathbf{E}_2^T, \dots, \mathbf{E}_p^T)^T, \\ \mathbf{F} &= (\mathbf{F}_1^T, \mathbf{F}_2^T, \dots, \mathbf{F}_p^T)^T, \\ \mathbf{X}^{(k)} &= \left((\mathbf{X}_1^{(k)})^T, (\mathbf{X}_2^{(k)})^T, \dots, (\mathbf{X}_p^{(k)})^T \right)^T, \\ \mathbf{R}^{(k)} &= \left((\mathbf{R}_1^{(k)})^T, (\mathbf{R}_2^{(k)})^T, \dots, (\mathbf{R}_p^{(k)})^T \right)^T, \\ \mathbf{Z}^{(k)} &= \left((\mathbf{Z}_1^{(k)})^T, (\mathbf{Z}_2^{(k)})^T, \dots, (\mathbf{Z}_p^{(k)})^T \right)^T, \end{aligned} \quad (13)$$

where $\mathbf{A}_i$, $\tilde{\mathbf{A}}_i$, $\mathbf{B}_i$, $\tilde{\mathbf{B}}_i$, $\mathbf{I}_i$, $\mathbf{D}_i$, $\mathbf{E}_i$, $\mathbf{F}_i$, $\mathbf{X}_i^{(k)}$, $\mathbf{R}_i^{(k)}$, and $\mathbf{Z}_i^{(k)}$ are $l \times n$ subblock matrices, and $\tilde{\mathbf{C}}_i$ is $n \times l$ subblock matrix ($i = 1, 2, \dots, p$). Because of subblock matrices $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{C}}_i$ of $\tilde{\mathbf{A}}$ need be stored on p_i ($i = 1, 2, \dots, p$); then, the data of $\tilde{\mathbf{A}}$ will be redistributed during the precondition process. And the details can be seen in precondition process. In addition, since the storage manner directly influences the ways of matrix multiplication in parallel, two parallel computing algorithms of matrix multiplication proposed in [22] are used in the paper:

Multiplication 1: matrix multiplication with block row-row matrices stored;

Multiplication 2: matrix multiplication with block row-column matrices stored.

Its detailed implementation can be expressed as follows.

(1) *Storage Method.* Store $\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{I}_i$, $\mathbf{F}_i$, $\mathbf{X}_i^{(0)}$, α on processor p_i ($i = 1, 2, \dots, p$).

(2) *Precondition Process.* Solve matrix equations $(\alpha \mathbf{A} - \mathbf{I})\mathbf{Y}_1 = \mathbf{I}$, $(\alpha \mathbf{B} - \mathbf{I})\mathbf{Y}_2 = \mathbf{I}$ by Algorithm 4 in parallel, then we can get the matrices $(\alpha \mathbf{A} - \mathbf{I})^{-1}$ and $(\alpha \mathbf{B} - \mathbf{I})^{-1}$. Calculate $\mathbf{I}_i + 2\mathbf{D}_i$ on p_i ($i = 1, 2, \dots, p$), the subblock matrices $\tilde{\mathbf{A}}_i$ of $\tilde{\mathbf{A}}$ can be obtained by p_i ($i = 1, 2, \dots, p$); all the p processors pass p times in turn, and the sub-block matrices $\tilde{\mathbf{C}}_i$ of $\tilde{\mathbf{A}}$ can be obtained by p_i ($i = 1, 2, \dots, p$). Thus, the two storage ways

of $\tilde{\mathbf{A}}$ are realized. Calculate $\mathbf{I}_i + 2\mathbf{E}_i$ on p_i ($i = 1, 2, \dots, p$); the subblock matrices $\tilde{\mathbf{B}}_i$ of $\tilde{\mathbf{B}}$ can be obtained by p_i ($i = 1, 2, \dots, p$).

(3) *Calculate Process.* Calculate matrix multiplication $\tilde{\mathbf{A}}\mathbf{X}^{(0)}$ by Multiplication 1; then calculate matrix multiplication $(\tilde{\mathbf{A}}\mathbf{X}^{(0)})\tilde{\mathbf{B}}$ by Multiplication 1. Finally, we can obtain $\mathbf{R}^{(0)}$; then, get matrix multiplication $\tilde{\mathbf{A}}^T\mathbf{R}^{(0)}$ by Multiplication 1, and calculate $(\tilde{\mathbf{A}}^T\mathbf{R}^{(0)})\tilde{\mathbf{B}}^T$ by Multiplication 2, so $\mathbf{Q}^{(0)}$ can be obtained.

(4) *Cycle Process:*

- (1) On p_i ($i = 1, 2, \dots, p$), calculate $[\mathbf{R}_i^{(k)}, \mathbf{R}_i^{(k)}]$ and $[\mathbf{Z}_i^{(k)}, \mathbf{Z}_i^{(k)}]$; we can get $[\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]$ and $[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)}]$ after all-reduce. If $\|\mathbf{R}^{(k)}\| < \varepsilon$, or $\|\mathbf{R}^{(k)}\| > \varepsilon$ but $\|\mathbf{Z}^{(k)}\| < \varepsilon$ stop; otherwise, turn back to (2).
- (2) Calculate $\gamma_k = [\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]/[\mathbf{Z}^{(k)}, \mathbf{Z}^{(k)}]$ in each processor.
- (3) On p_i ($i = 1, 2, \dots, p$) calculate $\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \gamma_k \mathbf{Z}_i^{(k)}$.
- (4) Then get $\tilde{\mathbf{A}}\mathbf{X}^{(k+1)}$ and $(\tilde{\mathbf{A}}\mathbf{X}^{(k+1)})\tilde{\mathbf{B}}$; Processor p_i ($i = 1, 2, \dots, p$) calculates $\mathbf{R}_i^{(k+1)}$ and $[\mathbf{R}_i^{(k+1)}, \mathbf{R}_i^{(k+1)}]$, and finally obtain $[\mathbf{R}^{(k+1)}, \mathbf{R}^{(k+1)}]$ after all-reduce.
- (5) After that $\tilde{\mathbf{A}}^T\mathbf{R}^{(k+1)}$ and $(\tilde{\mathbf{A}}^T\mathbf{R}^{(k+1)})\tilde{\mathbf{B}}^T$ are done; then calculate $\mathbf{Q}_i^{(k+1)}$ on p_i ($i = 1, 2, \dots, p$).
- (6) Calculate the factor $\eta_k = [\mathbf{R}^{(k+1)}, \mathbf{R}^{(k+1)}]/[\mathbf{R}^{(k)}, \mathbf{R}^{(k)}]$ in each processor; then on processor p_i ($i = 1, 2, \dots, p$), renew the value $\mathbf{Z}_i^{(k+1)} = \mathbf{Q}_i^{(k+1)} + \eta_k \mathbf{Z}_i^{(k+1)}$.
- (7) Let $k := k + 1$; turn back to (1).

Note that if n is not divisible by p , some processors store $[n/p] + 1$ rows-block of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{I}$, sequentially, and others store $[n/p]$ rows-block to make the load balance of each processor and short wait-time.

5. Convergent Analysis

Let the matrices $\mathbf{A}$ and $\mathbf{B}$ be both positive definite symmetric matrices. λ_i, μ_j ($i, j = 1, 2, \dots, n$) are the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$, respectively. λ_i, μ_j ($i, j = 1, 2, \dots, n$) satisfy $\lambda_n \geq \dots \geq \lambda_2 \geq \lambda_1 > 0, \mu_n \geq \dots \geq \mu_2 \geq \mu_1 > 0$.

For the matrix equations $\mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{B} + \mathbf{F} = \mathbf{O}$, the speed of the convergence depends on

$$\frac{\max |\lambda_i + \mu_j|}{\min |\lambda_i + \mu_j|} = \frac{\lambda_n + \mu_n}{\lambda_1 + \mu_1}. \quad (14)$$

While our proposed matrix forms $\mathbf{X} = \tilde{\mathbf{A}}\mathbf{X}\tilde{\mathbf{B}} + \tilde{\mathbf{F}}$, its convergent rate is decided by

$$\frac{\max_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|}{\min_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|}. \quad (15)$$

The following comparison could indicate that our algorithm is more effective.

Theorem 6. *If the eigenvalues of matrices $\mathbf{A}$ and $\mathbf{B}$ satisfy*

$$\frac{\lambda_1}{\lambda_n} \leq \left(\frac{\sqrt{5} - 1}{2} \right)^2, \quad \frac{\mu_1}{\mu_n} \leq \frac{\sqrt{5} - 1}{2}, \quad (16)$$

then there exists α such that the inequality

$$\frac{\max_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|}{\min_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|} < \frac{\lambda_n + \mu_n}{\lambda_1 + \mu_1} \quad (17)$$

holds.

Proof. Since $|((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))| < 1$, for $\alpha < 0$, we have

$$\begin{aligned} & \frac{\max_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|}{\min_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|} \\ & \leq \frac{2}{\min_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|} \\ & \leq \frac{2}{1 - \rho(\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}^T)}. \end{aligned} \quad (18)$$

From [17], we have known that $\rho(\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}^T) = ((\sqrt{\lambda_n} - \sqrt{\lambda_1}) / (\sqrt{\lambda_n} + \sqrt{\lambda_1})) ((\sqrt{\lambda_1\lambda_n} - \mu_1) / (\sqrt{\lambda_1\lambda_n} + \mu_1))$, for $\alpha = -1 / \sqrt{\lambda_1\lambda_n}$. Since $\lambda_1 / \lambda_n \leq ((\sqrt{5} - 1) / 2)^2$ and $\mu_1 / \mu_n \leq (\sqrt{5} - 1) / 2$, we have that

$$\sqrt{\lambda_n} (\lambda_n - \sqrt{\lambda_1\lambda_n} - \lambda_1) + \sqrt{\lambda_n} (\mu_n - \mu_1 - \mu_1 \sqrt{\frac{\lambda_1}{\lambda_n}}) > 0. \quad (19)$$

So

$$\frac{\sqrt{\lambda_n}}{(\sqrt{\lambda_n} + \sqrt{\lambda_1})(\sqrt{\lambda_1\lambda_n} + \mu_1)} \geq \frac{1}{\lambda_n + \mu_n}, \quad (20)$$

that is,

$$\frac{2}{1 - \rho(\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}^T)} \leq \frac{\lambda_n + \mu_n}{\lambda_1 + \mu_1}, \quad \text{for } \alpha = -\frac{1}{\sqrt{\lambda_1\lambda_n}}. \quad (21)$$

Therefore

$$\begin{aligned} & \frac{\max_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|}{\min_{i,j} |1 - ((\alpha\lambda_i + 1) / (\alpha\lambda_i - 1)) ((\alpha\mu_j + 1) / (\alpha\mu_j - 1))|} \\ & < \frac{\lambda_n + \mu_n}{\lambda_1 + \mu_1} \quad \text{for } \alpha = -\frac{1}{\sqrt{\lambda_1\lambda_n}}. \end{aligned} \quad (22)$$

□

TABLE 1: Numerical results of Example 7.

	p	8	12	16	20
Parameter iterative method ($\alpha = -0.3015$)	T (s)	571.2402	411.2114	310.7856	251.4239
	K	33	33	33	33
	$E1$	0.53	0.49	0.48	0.48
	Δ	$7.21e - 11$	$7.21e - 11$	$7.21e - 11$	$7.21e - 11$
Parameter iterative method ($\alpha = -0.4083$)	T (s)	564.0080	394.3398	301.6541	246.2671
	K	31	31	31	31
	$E1$	0.53	0.51	0.50	0.49
	Δ	$7.21e - 11$	$7.21e - 11$	$7.21e - 11$	$7.21e - 11$
MCG algorithm	T (s)	641.1601	427.8546	332.7490	271.4428
	K	79	79	79	79
	$E1$	0.47	0.47	0.46	0.45
	Δ	$9.80e - 11$	$9.80e - 11$	$9.80e - 11$	$9.80e - 11$
The algorithm in the paper ($\alpha = -0.3015$)	T (s)	303.8564	220.1811	176.6607	151.9282
	K	20	20	20	20
	$E1$	1	0.92	0.86	0.80
	Δ	$9.82e - 11$	$9.82e - 11$	$9.82e - 11$	$9.82e - 11$
The algorithm in the paper ($\alpha = -0.4083$)	T (s)	301.6773	216.2561	167.5985	141.9657
	K	18	18	18	18
	$E1$	1	0.93	0.90	0.85
	Δ	$6.39e - 11$	$6.39e - 11$	$6.39e - 11$	$6.39e - 11$

From previously mentioned theorem, we can observe that proposed preconditioned MCG is better than traditional MCG if λ_1/λ_n and μ_1/μ_n are less, that is, if the singular values of $\mathbf{A}$ and $\mathbf{B}$ are centralized slightly.

6. Numerical Examples and Results Analysis

In this section, the comparison with three methods the parameter iterative method, the MCG algorithm, and the proposed algorithm (PMCG) is discussed for solving two examples. The parallel platform is the parallel machine Lenovo Shenteng 1800 cluster. In the following test cases, $\mathbf{X}^{(0)} = \mathbf{O}$ and termination condition $\varepsilon = 10^{-10}$.

Example 7. Consider the elliptical partial differential equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \sin 2\pi x \frac{\partial u}{\partial x} + \sin 2\pi y \frac{\partial u}{\partial y} + u = 0, \quad (23)$$

$$0 \leq x, \quad y \leq 1,$$

the boundary condition are

$$\begin{aligned} u|_{x=0} &= u|_{x=1} = 10 + \cos \pi y, \\ u|_{y=0} &= u|_{y=1} = 10 + \cos \pi x. \end{aligned} \quad (24)$$

Let the step size be $h = 1/1001$, and five-point difference form is used to discrete above equation, which is finally transformed to solving a Sylvester matrix equation seen in (1). Since the eigenvalues of matrix $\mathbf{A}$ and matrix $\mathbf{B}$ are all greater than zero, then we have $\alpha = -\sqrt{n}/\|\mathbf{A}\|_F = -0.3015$

and $\alpha = -\sqrt{n}/\|\mathbf{B}\|_F = -0.4083$, respectively. The numerical results are shown in Table 1.

Example 8. If matrices $\mathbf{A} = (a_{ij})_{n \times n}$ and $\mathbf{B} = (b_{ij})_{n \times n}$, where

$$\begin{aligned} a_{ij} &= \begin{cases} 3 + \sin(i+j) & i = j \\ \cos(i+j) & \text{others,} \end{cases} \\ b_{ij} &= \begin{cases} 3 + 2 \sin(i+j) & i = j \\ 2 \cos(i+j) & \text{others,} \end{cases} \end{aligned} \quad (25)$$

$\mathbf{F}$ is any given $n \times n$ matrix, $n = 500$. The comparison of three different methods for solving Sylvester matrix equation $\mathbf{AX} + \mathbf{XB} + \mathbf{F} = \mathbf{O}$ is discussed here.

Since the eigenvalues of matrix $\mathbf{A}$ and matrix $\mathbf{B}$ have not only negative real numbers but also positive ones, so there are four choices for α . We have $\alpha = -\sqrt{n}/\|\mathbf{A}\|_F = -0.0621$, $\alpha = \sqrt{n}/\|\mathbf{A}\|_F = 0.0621$, $\alpha = -\sqrt{n}/\|\mathbf{B}\|_F = -0.0314$, and $\alpha = \sqrt{n}/\|\mathbf{B}\|_F = 0.0314$, respectively. The numerical results are shown in Table 2.

6.1. Results Analysis. From the results shown in the above tables, we observed the following.

- (i) Compared with the modified conjugate gradient method and parameter iterative method, the iteration number and the computational time by our algorithm are significantly reduced. It is shown that the preconditioned method enhances its validity and convergent rate. Obviously, better parallel efficiency can be obtained with our algorithm.
- (ii) When the eigenvalues of matrix $\mathbf{A}$ and matrix $\mathbf{B}$ are all positive real numbers (seen in Example 7),

TABLE 2: Numerical results of Example 8.

	p	4	8	12	16
Parameter iterative method ($\alpha = -0.0621, 0.0621, -0.0314, 0.0314$)	T (s)	—	—	—	—
	K	—	—	—	—
	$E1$	—	—	—	—
	Δ	—	—	—	—
MCG algorithm	T (s)	612.0169	336.6452	237.2380	179.1261
	K	331	331	331	331
	$E1$	0.23	0.21	0.20	0.20
	Δ	$8.35e - 11$	$8.35e - 11$	$8.35e - 11$	$8.35e - 11$
The algorithm in the paper ($\alpha = -0.0621$)	T (s)	144.6455	79.1208	54.7878	41.6944
	K	80	79	81	81
	$E1$	1	0.91	0.88	0.87
	Δ	$3.64e - 11$	$5.31e - 11$	$9.44e - 11$	$9.44e - 11$
The algorithm in the paper ($\alpha = 0.0621$)	T (s)	221.1802	120.1056	84.8724	64.2965
	K	134	133	133	133
	$E1$	1	0.92	0.87	0.86
	Δ	$7.70e - 11$	$9.87e - 11$	$9.87e - 11$	$9.87e - 11$
The algorithm in the paper ($\alpha = -0.0314$)	T (s)	169.8777	91.0232	65.0475	48.8154
	K	104	103	103	103
	$E1$	1	0.93	0.87	0.87
	Δ	$8.07e - 11$	$6.30e - 11$	$6.30e - 11$	$6.30e - 11$
The algorithm in the paper ($\alpha = 0.0314$)	T (s)	245.1565	132.7123	93.1277	70.7914
	K	157	154	154	154
	$E1$	1	0.92	0.88	0.87
	Δ	$7.03e - 11$	$5.91e - 11$	$5.91e - 11$	$5.91e - 11$

Note:

- (1) Here p , T (s), K , $E1$, and Δ represent the process number, time, iteration number, absolute parallel efficiency, and error, respectively.
(2) Because the computational time of one processor is too long, we take the minimal time with one of multiprocessors as a reference to calculate the relative speedup and relative parallel efficiency.

the parallel efficiency of our algorithm is much higher than it with traditional MCG. Furthermore, the numerical results are consistent with the theory; in the case that the eigenvalues have not only negative real numbers but also positive ones in Example 8, the parallel efficiency of our algorithm is still higher than that with MCG. That means the preconditioned factor α is efficient.

- (iii) From the results in the above Tables, we can observe that the parameter method is ineffective in Example 8. This illustrates that our algorithm has a wider range of applications than parameter method.

Conflict of Interests

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

Acknowledgments

This work has been supported by the Basic Project Foundation of Northwestern Polytechnical University (Grants no.

JC20120241), by the Natural Science Foundation of Shan'xi Province (Grants no. 2009JM1008), and by the National Natural Science Foundation of China (Grants no. 11302173). The authors also acknowledge the High Performance Computing Center of Northwestern Polytechnical University for implementation of above two examples.

References

- [1] B. N. Datta, *Numerical Methods for Linear Control Systems*, Elsevier Academic Press, San Diego, Calif, USA, 2004.
- [2] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.
- [3] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, vol. 6 of *Advances in Design and Control*, SIAM, Philadelphia, Pa, USA, 2005.
- [4] U. Baurt and P. Benner, "Cross-gramian based model reduction for datasparse systems," *Electronic Transactions on Numerical Analysis*, vol. 31, pp. 256–270, 2008.
- [5] D. C. Sorensen and A. C. Antoulas, "The Sylvester equation and approximate balanced reduction," *Linear Algebra and its Applications*, vol. 351–352, pp. 671–700, 2002.

- [6] D. Calvetti and L. Reichel, "Application of ADI iterative methods to the restoration of noisy images," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 1, pp. 165–186, 1996.
- [7] R. H. Bartels and G. W. Stewart, "Solution of the matrix equation $AX + XB = C$ [F4]," *Communications of the ACM*, vol. 15, no. 9, pp. 820–826, 1972.
- [8] F. Ding and T. Chen, "On iterative solutions of general coupled matrix equations," *SIAM Journal on Control and Optimization*, vol. 44, no. 6, pp. 2269–2284, 2006.
- [9] D. W. Peaceman and H. H. Rachford Jr., "The numerical solution of parabolic and elliptic differential equations," *Journal of the Society for Industrial and Applied Mathematics*, vol. 3, no. 1, pp. 28–41, 1955.
- [10] E. L. Wachspress, "The adi minimax problem for complex spectra," *Applied Mathematics Letters*, vol. 1, no. 3, pp. 311–314, 1988.
- [11] K.-Y. Zhang and T.-J. Wang, "The parameter iterative method for matrix equation $AX + XB = F$," *Chinese Journal of Engineering Mathematics*, vol. 21, no. 8, pp. 6–10, 2004.
- [12] S.-F. Xu, *Matrix Computations in Control Theory*, Higher Education Press, Beijing, China, 2011.
- [13] A. El Guennouni, K. Jbilou, and A. J. Riquet, "Block Krylov subspace methods for solving large Sylvester equations," *Numerical Algorithms*, vol. 29, no. 1–3, pp. 75–96, 2002.
- [14] X.-M. Li and J.-P. Wu, *Numerical Parallel Algorithm and Software*, Science Press, Beijing, China, 2007.
- [15] K.-Y. Zhang and Z. Xu, *Numerical Algebra*, Science Press, Beijing, China, 2010.
- [16] L. Bao, Y. Lin, and Y. Wei, "A new projection method for solving large Sylvester equations," *Applied Numerical Mathematics*, vol. 57, no. 5–7, pp. 521–532, 2007.
- [17] Y.-X. Peng, X.-Y. Hu, and L. Zhang, "An iteration method for the symmetric solutions and the optimal approximation solution of the matrix equation $AXB = C$," *Applied Mathematics and Computation*, vol. 160, no. 3, pp. 763–777, 2005.
- [18] B.-L. Zhang, T.-X. Gu, and Z.-Y. Mo, *Principles and Methods of Numerical Parallel Computation*, Science Press, Beijing, China, 1999.
- [19] M. Dehghan and M. Hajarani, "An iterative algorithm for solving a pair of matrix equations $AYB = E$, $CYD = F$ over generalized centro-symmetric matrices," *Computers & Mathematics with Applications*, vol. 56, no. 12, pp. 3246–3260, 2008.
- [20] S.-J. Chen and K.-Y. Zhang, "Symmetric solution of a system of matrix equations and its optimal approximation," *Chinese Journal of Engineering Mathematics*, vol. 26, no. 4, pp. 711–715, 2009.
- [21] Y.-P. Cheng, *Matrix Theory*, Northwestern Polytechnical University Press, Xi'an, China, 2002.
- [22] G.-L. Chen, A. Hong, J. Chen, Q.-L. Zheng, and J.-L. Shan, *The Parallel Algorithm*, Higher Education Press, Beijing, China, 2004.