Journal of Computational Mathematics, Vol.26, No.2, 2008, 240–249.

THE RESTRICTIVELY PRECONDITIONED CONJUGATE GRADIENT METHODS ON NORMAL RESIDUAL FOR BLOCK TWO-BY-TWO LINEAR SYSTEMS*

Junfeng Yin and Zhongzhi Bai
LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China
Email: yinjf@lsec.cc.ac.cn, bzz@lsec.cc.ac.cn

Abstract

The *restrictively preconditioned conjugate gradient* (RPCG) method is further developed to solve large sparse system of linear equations of a block two-by-two structure. The basic idea of this new approach is that we apply the RPCG method to the normal-residual equation of the block two-by-two linear system and construct each required approximate matrix by making use of the incomplete orthogonal factorization of the involved matrix blocks. Numerical experiments show that the new method, called the *restrictively preconditioned conjugate gradient on normal residual* (RPCGNR), is more robust and effective than either the known RPCG method or the standard *conjugate gradient on normal residual* (CGNR) method when being used for solving the large sparse saddle point problems.

Mathematics subject classification: 65F10, 65W05.
Key words: Block two-by-two linear system, Saddle point problem, Restrictively preconditioned conjugate gradient method, Normal-residual equation, Incomplete orthogonal factorization.

1. Introduction

Consider an iterative solution of the *block two-by-two* (BTT) system of linear equations

$$Ax = b, \quad \text{where} \quad A = \begin{bmatrix} B & E \\ F & C \end{bmatrix} \in \mathbb{R}^{n\times n} \quad \text{and} \quad x, b \in \mathbb{R}^n, \tag{1.1}$$

with $B \in \mathbb{R}^{m\times m}$ and $C \in \mathbb{R}^{l\times l}$ being large, sparse, square and nonsymmetric matrices, $E \in \mathbb{R}^{m\times l}$ and $F \in \mathbb{R}^{l\times m}$ being sparse matrices, with $m \geq l$, such that $A \in \mathbb{R}^{n\times n}$ is nonsingular, where $n = m + l$. The BTT linear system (1.1) frequently arises in many areas of scientific computing and engineering applications such as the constrained least-squares problems, the Navier-Stokes equations in fluid computations, and the Maxwell equations in computational electromagnetics; see [1, 7, 2] for more details. Obviously, the saddle point problems form a subset of the BTT linear systems.

When the matrix blocks B is symmetric positive definite, C is symmetric positive semidefinite (e.g., $C = 0$) and $F = \pm E^T$, Bai and Li [4] recently proposed the *restrictively preconditioned conjugate gradient* (RPCG) methods for these special forms of the BTT linear system (1.1) and studied their convergence properties. Numerical results showed that this class of methods are robust and effective solvers for iteratively computing the solutions of the symmetric positive definite and the Hamiltonian systems of linear equations. We remark that the RPCG methods

* Received January 20, 2006 / Revised version received April 2, 2007 / Accepted September 19, 2007 /

were also developed to general nonsingular and nonsymmetric linear systems, resulting in a rather general framework of iterative methods, which not only covers many standard Krylov subspace methods such as the conjugate gradient [8, 11], the conjugate residual [8, 9], the conjugate gradient on normal residual (CGNR) [8, 9, 13] and the conjugate gradient on normal equation (CGNE) [8, 9, 13], as well as their preconditioned variants, but also yields many new ones. For details, we refer to [4] and references therein. Latter, Bai and Wang [5] further developed the RPCG method and obtained an inexact variant for the symmetric positive definite case of the BTT linear system (1.1), in which both B and C are symmetric positive definite and $F = E^T$.

In this paper, we use the inexact RPCG method presented in [5] to solve the BTT linear system (1.1). This new approach first forms the normal-residual equation

$$\mathcal{A}x \equiv A^T A x = A^T b \equiv \mathbf{b}, \tag{1.2}$$

where

$$\mathcal{A} = \begin{bmatrix} \mathcal{B} & \mathcal{E} \\ \mathcal{E}^T & \mathcal{C} \end{bmatrix} \equiv \begin{bmatrix} B^T B + F^T F & B^T E + F^T C \\ E^T B + C^T F & C^T C + E^T E \end{bmatrix} \tag{1.3}$$

is symmetric positive definite, with

$$\mathcal{B} = B^T B + F^T F, \quad \mathcal{C} = C^T C + E^T E \quad \text{and} \quad \mathcal{E} = B^T E + F^T C, \tag{1.4}$$

and then straightforwardly apply the inexact RPCG method developed in [5] to (1.2)-(1.4), as now $\mathcal{B} \in \mathbb{R}^{m\times m}$ and $\mathcal{C} \in \mathbb{R}^{l\times l}$ are symmetric positive definite and $\mathcal{E} \in \mathbb{R}^{m\times l}$ is of full column rank.

For the saddle point problems of the form

$$Ax = b, \quad \text{where} \quad A = \begin{bmatrix} B & E \\ E^T & 0 \end{bmatrix} \in \mathbb{R}^{n\times n} \quad \text{and} \quad x, b \in \mathbb{R}^n, \tag{1.5}$$

with $B \in \mathbb{R}^{m\times m}$ being a positive definite matrix, $E \in \mathbb{R}^{m\times l}$ being a matrix of full column rank and $m \geq l$, we give a practical way for constructing approximations to the submatrices $\mathcal{B}$ and $\mathcal{S} = \mathcal{C} - \mathcal{E}^T \mathcal{B}^{-1} \mathcal{E}$ by utilizing the incomplete orthogonal factorization technique in [3]; see also [12, 6]. The resulting method, called *RPCG on normal residual* (RPCGNR), is algorithmically described in detail. Numerical examples are used to show that the RPCGNR method outperforms the *preconditioned CGNR* (PCGNR) method [8] when they are employed to solve the large sparse saddle point problem (1.5). Moreover, RPCGNR also shows better numerical behaviour than both RPCG [4] and PCGNR when the matrix block B is symmetric positive definite and when RPCG is directly applied to the saddle point problem (1.5).

The organization of this paper is as follows. After establishing the RPCGNR method for solving the BTT linear system (1.1) and its special case (1.5) in Section 2, we present practical choices for approximating matrix blocks involved in the saddle point problem (1.5) in Section 3. In Section 4, numerical results are used to show the feasibility and effectiveness of our new method. Finally, in Section 5, we end this paper with a brief conclusion.

2. The RPCGNR Method

To establish the RPCGNR method for solving the BTT linear system (1.1), according to [5] we first decompose the block two-by-two matrix $\mathcal{A} \in \mathbb{R}^{n\times n}$ in (1.3) as $\mathcal{A} = \mathcal{PHQ}$, where

$\mathcal{P}, \mathcal{Q} \in \mathbb{R}^{n\times n}$ are two nonsingular matrices, and $\mathcal{H} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix. We then construct a restrictive preconditioner $\mathcal{M} = \mathcal{PWQ}$ for the matrix $\mathcal{A}$, where $\mathcal{W} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix approximating the matrix $\mathcal{H}$. See also [1, 2, 4].

It follows from the symmetric positive definiteness of the matrix $\mathcal{A} \in \mathbb{R}^{n\times n}$ that both submatrices $\mathcal{B} \in \mathbb{R}^{m\times m}$ and $\mathcal{C} \in \mathbb{R}^{l\times l}$ are symmetric positive definite. By direct computations, we know that $\mathcal{A} \in \mathbb{R}^{n\times n}$ adopts the block triangular factorization

$$\mathcal{A} = \underbrace{\begin{bmatrix} \mathcal{I} & 0 \\ \mathcal{E}^T\mathcal{B}^{-1} & \mathcal{I} \end{bmatrix}}_{\mathcal{P}} \underbrace{\begin{bmatrix} \mathcal{B} & 0 \\ 0 & \mathcal{S} \end{bmatrix}}_{\mathcal{H}} \underbrace{\begin{bmatrix} \mathcal{I} & \mathcal{B}^{-1}\mathcal{E} \\ 0 & \mathcal{I} \end{bmatrix}}_{\mathcal{Q}} \equiv \mathcal{PHQ}, \tag{2.1}$$

where $\mathcal{I}$ is the identity matrix and

$$\mathcal{S} = \mathcal{C} - \mathcal{E}^T\mathcal{B}^{-1}\mathcal{E} \tag{2.2}$$

is the Schur complement. Evidently, $\mathcal{S} \in \mathbb{R}^{l\times l}$ is a symmetric positive definite matrix, too. Denote by $\mathcal{K} = \mathcal{Q}^{-1}\mathcal{P}^T$. Then, it holds that

$$\mathcal{K} = \mathcal{Q}^{-1}\mathcal{P}^T = \begin{bmatrix} \mathcal{I} & -\mathcal{B}^{-1}\mathcal{E} \\ 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \mathcal{I} & \mathcal{B}^{-1}\mathcal{E} \\ 0 & \mathcal{I} \end{bmatrix} = \mathcal{I}. \tag{2.3}$$

Let $\widehat{\mathcal{B}} \in \mathbb{R}^{m\times m}$ and $\widehat{\mathcal{S}} \in \mathbb{R}^{l\times l}$ be symmetric positive definite matrices approximating the matrices $\mathcal{B}$ and $\mathcal{S}$, respectively. Then we can choose the matrix $\mathcal{W}$ involved in the preconditioning matrix $\mathcal{M}$ to be the block diagonal matrix $\mathcal{W} = \mathrm{Diag}(\widehat{\mathcal{B}}, \widehat{\mathcal{S}})$.

Based on the above investigation, we can algorithmically describe the inexact RPCG method in [5] with respect to the symmetric positive definite BTT linear system (1.2)-(1.4) as follows.

Method 2.1. *(The RPCGNR Method for the BTT Linear System (1.2)-(1.4))*

```
1. **Choose** $x_0 \in \mathbb{R}^n, r_0 = A^T(b - Ax_0)$
2. **Let** $r_0 := (r_0^{(1)^T}, r_0^{(2)^T})^T$ and $z_0 := (z_0^{(1)^T}, z_0^{(2)^T})^T$
3.    **Solve** $\widehat{\mathcal{B}}t^{(1)} = r_0^{(1)}$
4.    **Solve** $\widehat{\mathcal{S}}z_0^{(2)} = r_0^{(2)} - \mathcal{E}^T t^{(1)}$
5.    **Solve** $\widehat{\mathcal{B}}\widetilde{t}^{(1)} = \mathcal{E}z_0^{(2)}$
6.    **Compute** $z_0^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
7. **Set** $p_0 := z_0$
8. **For** $k = 0, 1, 2, \cdots$
9.       $q_k = Ap_k$
10.      $\alpha_k = z_k^T r_k / q_k^T q_k$
11.      $x_{k+1} = x_k + \alpha_k p_k$
12.      $r_{k+1} = r_k - \alpha_k A^T q_k$
13.      **Let** $r_{k+1} := (r_{k+1}^{(1)^T}, r_{k+1}^{(2)^T})^T$ and $z_{k+1} = (z_{k+1}^{(1)^T}, z_{k+1}^{(2)^T})^T$
14.         **Solve** $\widehat{\mathcal{B}}t^{(1)} = r_{k+1}^{(1)}$
15.         **Solve** $\widehat{\mathcal{S}}z_{k+1}^{(2)} = r_{k+1}^{(2)} - \mathcal{E}^T t^{(1)}$
```

```
16.       **Solve** $\widehat{\mathcal{B}}\widetilde{t}^{(1)} = \mathcal{E}z_{k+1}^{(2)}$
17.       **Compute** $z_{k+1}^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
18.     $\beta_k = z_{k+1}^T r_{k+1}/z_k^T r_k$
19.     $p_{k+1} = z_{k+1} + \beta_k p_k$
```

In particular, when $F = E^T$ and $C = 0$, Method 2.1 automatically results in the RPCGNR method for solving the saddle point problem (1.5), which is precisely described as follows.

Method 2.2. *(The RPCGNR Method for the Saddle Point Problem (1.5))*

```
1. **Choose** $x_0 \in \mathbb{R}^n$ and $r_0 = A^T(b - Ax_0)$
2. **Let** $r_0 = (r_0^{(1)^T}, r_0^{(2)^T})^T$ and $z_0 = (z_0^{(1)^T}, z_0^{(2)^T})^T$
3.   **Solve** $\widehat{\mathcal{B}}t^{(1)} = r_0^{(1)}$
4.   **Solve** $\widehat{\mathcal{S}}z_0^{(2)} = r_0^{(2)} - E^T B t^{(1)}$
5.   **Solve** $\widehat{\mathcal{B}}\widetilde{t}^{(1)} = B^T E z_0^{(2)}$
6.   **Compute** $z_0^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
7. **Set** $p_0 := z_0$
8. **For** $k = 0, 1, 2, \cdots$
9.      $q_k = Ap_k$
10.     $\alpha_k = z_k^T r_k / q_k^T q_k$
11.     $x_{k+1} = x_k + \alpha_k p_k$
12.     $r_{k+1} = r_k - \alpha_k A^T q_k$
13.     **Let** $r_{k+1} = (r_{k+1}^{(1)^T}, r_{k+1}^{(2)^T})^T$ and $z_{k+1} = (z_{k+1}^{(1)^T}, z_{k+1}^{(2)^T})^T$
14.       **Solve** $\widehat{\mathcal{B}}t^{(1)} = r_{k+1}^{(1)}$
15.       **Solve** $\widehat{\mathcal{S}}z_{k+1}^{(2)} = r_{k+1}^{(2)} - E^T B t^{(1)}$
16.       **Solve** $\widehat{\mathcal{B}}\widetilde{t}^{(1)} = B^T E z_{k+1}^{(2)}$
17.       **Compute** $z_{k+1}^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
18.     $\beta_k = z_{k+1}^T r_{k+1}/z_k^T r_k$
19.     $p_{k+1} = z_{k+1} + \beta_k p_k$
```

The advantages of Method 2.2 over the RPCG method [5] is that it may be feasible and effective even for the saddle point problem with nonsymmetric or indefinite $(1,1)$ block, and over the CGNR method [8] is that it uses a structured preconditioner rather than a general one. In fact, good approximations $\widehat{\mathcal{B}}$ and $\widehat{\mathcal{S}}$ to the matrices $\mathcal{B}$ and $\mathcal{S}$ are very crucial for the effectiveness of Method 2.2. Some typical choices of these two approximate matrices will be discussed in detail in the next section.

3. Choices of Approximating Matrix Blocks

We now construct practical and accurate approximations to the matrix blocks $\mathcal{B}$ and $\mathcal{S}$. To this end, we note that

$$\mathcal{B} = B^T B + EE^T = [B^T \ E] \begin{bmatrix} B \\ E^T \end{bmatrix}.$$

If we decompose the matrix $\begin{bmatrix} B \\ E^T \end{bmatrix}$ into an incomplete orthogonal-triangular form, i.e.,

$$\begin{bmatrix} B \\ E^T \end{bmatrix} = Q_{\mathcal{B}} R_{\mathcal{B}},$$

where $Q_{\mathcal{B}}$ is an incomplete orthogonal matrix and $R_{\mathcal{B}}$ is an incomplete upper triangular matrix, then $\mathcal{B}$ can be approximated by the matrix

$$\widehat{\mathcal{B}} = R_{\mathcal{B}}^T R_{\mathcal{B}}. \tag{3.1}$$

The incomplete orthogonal-triangular factors can be obtained by the incomplete modified Gram-Schmidt process [14] or the incomplete Givens orthogonalization process [3].

Analogously, by noticing that

$$\begin{aligned} \mathcal{S} &= \mathcal{C} - \mathcal{E}^T \mathcal{B}^{-1} \mathcal{E} \\ &\approx E^T E - E^T B \widehat{\mathcal{B}}^{-1} B^T E \\ &\approx E^T (I - B R_{\mathcal{B}}^{-1} R_{\mathcal{B}}^{-T} B^T) E \\ &= E^T (I - \widetilde{B} \widetilde{B}^T) E, \qquad \text{where} \quad \widetilde{B} = B R_{\mathcal{B}}^{-1}, \end{aligned}$$

we may take the approximation $\widehat{\mathcal{S}}$ of $\mathcal{S}$ to be

$$\widehat{\mathcal{S}} = R_{\mathcal{S}}^T R_{\mathcal{S}},$$

where $R_{\mathcal{S}}$ is the Cholesky factor in the incomplete Cholesky factorization of the matrix $E^T(I - \overline{B})E$, with $\overline{B} \approx \widetilde{B}\widetilde{B}^T$, e.g., $\overline{B}$ is the diagonal or the block diagonal matrix of $\widetilde{B}\widetilde{B}^T$.

4. Numerical Results

Consider the Oseen equation

$$\begin{cases} -\nu \, \Delta u^{m+1} + (u^m \cdot \nabla) u^{m+1} + \nabla p^{m+1} = f, \\ \operatorname{div} u^{m+1} = 0, \end{cases}$$

which is obtained when the steady-state Navier-Stokes equation is linearized by the Picard iteration. Here ν is the viscosity. Many discretization schemes applied to the Oseen equation can lead to saddle point problems of the form (1.5); see, for instance, [10] and references therein. Now, the $(1,1)$-block B of the coefficient matrix corresponds to the discretization of the convection-diffusion term, and it is nonsymmetric but positive real for the conservative discretization.

We generated the test problems (leaky-lid driven cavity) with the IFISS software written by Elman, Ramage, Silvester and Wathen, implemented the numerical experiments for the following two problems:

Table 4.1: IT and CPU for Problem (P_1) when $0 < \nu < 1$.

ν		0.001	0.005	0.01	0.05	0.1
RPCGNR(a)	IT	65	25	20	23	34
	CPU	0.54	0.21	0.17	0.20	0.28
RPCGNR(b)	IT	71	29	22	26	46
	CPU	0.60	0.23	0.18	0.22	0.39
PCGNR(a)	IT	130	50	40	48	66
	CPU	0.66	0.25	0.20	0.25	0.33
PCGNR(b)	IT	134	52	43	49	77
	CPU	0.67	0.27	0.21	0.25	0.38

Table 4.2: IT and CPU for Problem (P_1) when $\nu \geq 1$.

ν		1	10	50	100	500
RPCGNR(a)	IT	76	37	25	20	12
	CPU	0.68	0.32	0.21	0.17	0.11
RPCGNR(b)	IT	101	48	32	27	15
	CPU	0.87	0.42	0.28	0.23	0.12
PCGNR(a)	IT	237	113	69	51	30
	CPU	1.23	0.58	0.36	0.27	0.16
PCGNR(b)	IT	–	282	207	167	101
	CPU	–	1.45	1.08	0.85	0.52

(P_1) On a 16×16 grid, and the first two rows of E are dropped to avoid its rank deficiency so that the resulted E is a full-rank matrix. Hence, the saddle point problem (1.5) is of size $m = 578$ and $l = 254$;

(P_2) On a 32×32 grid, and the first two rows of E are dropped to avoid its rank deficiency so that the resulted E is a full-rank matrix. Hence, the saddle point problem (1.5) is of size $m = 2178$ and $l = 1020$.

In our implementations, all programs are coded with C++ and run on an SGI Origin 3800. The right-hand-side vector $b \in \mathbb{R}^n$ is generated such that the exact solution x^* of the saddle point problem (1.5) has all components being equal to one. This allows us to easily check the accuracy of the obtained approximate solution by the residual norm $\|Ax_k - b\|_2$ and the error norm $\|x_k - x^*\|_2$. The initial guess for each iteration is $x_0 = 0$, and the iteration process is terminated once the current iterate satisfies either $\|b - Ax_k\|_2 \leq 10^{-5}\|b - Ax_0\|_2$ or the number of iteration steps is over 300. All the incomplete factorization are computed by using a drop tolerance $\tau = 0.01$.

Because the $(1,1)$-block B in the coefficient matrix A of (1.5) is nonsymmetric for both problems (P_1) and (P_2), the RPCG-type methods proposed in [4, 5] can not be directly applied to solve the saddle point problem (1.5). However, the new RPCGNR method and the PCGNR method [13, 9] can be applied to solve these two problems. We can show that the former is numerically more efficient and more robust than the latter.

In our computations, we take $\widehat{\mathcal{B}}$ as given in (3.1) and $\widehat{\mathcal{S}} = \mathcal{R}_{\mathcal{S}}^T \mathcal{R}_{\mathcal{S}}$, where $\mathcal{R}_{\mathcal{S}}$ is the incomplete Cholesky factor of the matrix $\widetilde{\mathcal{S}}$ defined by either of the following two cases:

(a) $\widetilde{\mathcal{S}} = E^T E$;

(b) $\widetilde{\mathcal{S}} = E^T(I - \mathrm{Diag}(\widetilde{B}\widetilde{B}^T))E$, with $\widetilde{B} = BR_{\mathcal{B}}^{-1}$,

Table 4.3: IT and CPU for Problem (P_2) when $0 < \nu < 1$.

ν		0.001	0.005	0.01	0.05	0.1
RPCGNR(a)	IT	55	33	34	47	67
	CPU	7.09	4.23	4.36	6.17	8.62
RPCGNR(b)	IT	65	36	37	63	92
	CPU	8.39	4.64	4.75	8.17	11.79
PCGNR(a)	IT	111	67	69	80	104
	CPU	8.05	4.91	5.05	5.78	7.55
PCGNR(b)	IT	117	68	70	97	141
	CPU	8.52	4.95	5.08	7.06	10.22

Table 4.4: IT and CPU for Problem (P_2) when $\nu \geq 1$.

ν		1	10	50	100	500
RPCGNR(a)	IT	93	48	30	25	10
	CPU	12.03	6.21	3.86	3.25	1.27
RPCGNR(b)	IT	127	68	43	31	12
	CPU	16.46	8.81	5.61	4.01	1.56
PCGNR(a)	IT	–	206	113	49	28
	CPU	–	15.09	8.28	3.58	2.02
PCGNR(b)	IT	–	–	259	173	104
	CPU	–	–	18.89	12.61	7.56

where Diag$(\cdot)$ denotes the block diagonal matrix of the corresponding matrix.

The RPCGNR methods corresponding to the two choices (a) and (b) of $\widetilde{\mathcal{S}}$ are denoted as RPCGNR(a) and RPCGNR(b), respectively.

For the PCGNR method, we use the block-diagonal preconditioner

$$P = \begin{pmatrix} \widehat{\mathcal{B}} & 0 \\ 0 & \widehat{\mathcal{S}} \end{pmatrix}, \tag{4.1}$$

where $\widehat{\mathcal{B}}$ is given in (3.1) and $\widehat{\mathcal{S}} = \mathcal{R}_{\mathcal{S}}^T\mathcal{R}_{\mathcal{S}}$ is computed through the incomplete Cholesky factorization of the matrix $\widetilde{\mathcal{S}}$ defined above. Analogously, the PCGNR methods corresponding to the two choices (a) and (b) of $\widetilde{\mathcal{S}}$ are denoted as PCGNR(a) and PCGNR(b), respectively.

In Table 4.1, we list the iteration numbers (denoted by **IT**) and the CPU times in seconds (denoted by **CPU**) of the above-mentioned methods for Problem (P_1) when $0 < \nu < 1$. From the numerical results we observe that the iteration numbers of RPCGNR(#) are evidently much less than those of PCGNR(#). However, the CPU times of RPCGNR(#) are only a little less than those of PCGNR(#); the reason is that preconditioning RPCGNR(#) needs three more matrix-vector multiplications and two more vector-vector additions than preconditioning PCGNR(#).

In Table 4.2, we list the iteration numbers and the CPU times of these tested methods for Problem (P_1) when $\nu > 1$. Obviously, RPCGNR(a) is the best among all of the four methods in the sense of both iteration number and CPU time, and RPCGNR(b) is also a competitive one, in particular, for a larger ν. We note that when $\nu = 1$ PCGNR(b) can not achieve the convergence criterion within the given maximum number of iteration steps. Clearly, when ν is increasing, the iteration number is decreasing. It is also evident that the error reduction rate of

Table 4.5: IT and CPU for Problem (P_3).

ν		0.001	0.005	0.01	0.05	0.1
RPCG(a)	IT	88	81	75	75	55
	CPU	0.56	0.52	0.46	0.46	0.34
RPCG(b)	IT	37	37	36	35	31
	CPU	0.25	0.24	0.24	0.23	0.20
RPCGNR(a)	IT	3	5	8	23	40
	CPU	0.03	0.05	0.06	0.18	0.35
RPCGNR(b)	IT	3	6	8	30	52
	CPU	0.03	0.05	0.06	0.26	0.44
PCGNR(a)	IT	6	11	15	46	79
	CPU	0.05	0.07	0.09	0.25	0.39
PCGNR(b)	IT	6	11	15	48	84
	CPU	0.05	0.07	0.09	0.25	0.42
ν		1	10	50	100	500
RPCG(a)	IT	69	38	31	30	26
	CPU	0.42	0.23	0.20	0.19	0.15
RPCG(b)	IT	28	27	27	27	27
	CPU	0.19	0.17	0.17	0.17	0.17
RPCGNR(a)	IT	63	36	23	20	12
	CPU	0.54	0.31	0.18	0.17	0.11
RPCGNR(b)	IT	89	48	31	26	15
	CPU	0.76	0.40	0.26	0.21	0.13
PCGNR(a)	IT	151	73	47	41	30
	CPU	0.76	0.36	0.24	0.22	0.16
PCGNR(b)	IT	269	192	153	131	101
	CPU	1.39	0.98	0.76	0.66	0.52

RPCGNR(#) is much more rapid than that of PCGNR(#), and RPCGNR(#) is numerically more stable than PCGNR(#).

For Problem (P_2), when $0 < \nu < 1$ and $\nu \geq 1$, we respectively list the iteration numbers and the CPU times in Tables 4.3 and 4.4. These numerical results further examine and confirm the superiority of RPCGNR(#) to PCGNR(#) shown by those of Problem (P_1).

We now compare the numerical behaviours of the RPCGNR methods with those of the RPCG methods in [4] and the PCGNR methods in [13, 9]. As the RPCG methods require that the considered saddle point problem is of a symmetric positive definite $(1,1)$-block B, we replace the nonsymmetric $(1,1)$-blocks of the coefficient matrices in Problems (P_1) and (P_2) by their symmetric parts, and, correspondingly, obtain two classes of saddle point problems, denoted respectively as Problem (P_3) and (P_4), whose $(1,1)$-blocks are symmetric positive definite.

According to the approximation matrices $\widehat{B}$ and $\widehat{S}$ to the matrices B and $S := E^T B^{-1} E$, respectively, involved in the RPCG method, we take $\widehat{B} = R_B^T R_B$, with R_B the incomplete Cholesky factor [13, 9] of B, and

(a) $\widehat{S} = I$, or (b) $\widehat{S} = E^T \mathrm{Diag}(B)^{-1} E$.

The RPCG methods corresponding to the two choices (a) and (b) of $\widehat{S}$ are denoted as RPCG(a) and RPCG(b), respectively.

In Table 4.5, we list the iteration numbers and the CPU times in seconds of the testing

Table 4.6: IT and CPU for Problem (P_4).

ν		0.001	0.005	0.01	0.05	0.1
RPCG(a)	IT	185	174	176	114	101
	CPU	14.56	13.47	13.66	8.81	7.83
RPCG(b)	IT	71	69	68	66	66
	CPU	5.30	5.09	5.08	4.86	4.86
RPCGNR(a)	IT	5	12	20	42	59
	CPU	0.65	1.55	2.62	5.44	7.65
RPCGNR(b)	IT	5	13	22	57	76
	CPU	0.65	1.70	2.85	7.42	9.81
PCGNR(a)	IT	10	24	39	69	119
	CPU	0.73	1.77	2.86	5.09	8.70
PCGNR(b)	IT	10	24	40	78	117
	CPU	0.73	1.77	2.96	5.70	8.56
ν		1	10	50	100	500
RPCG(a)	IT	78	44	32	34	23
	CPU	6.03	3.41	2.49	2.63	1.78
RPCG(b)	IT	65	64	65	67	65
	CPU	4.75	4.73	4.78	4.91	4.80
RPCGNR(a)	IT	87	49	31	25	11
	CPU	11.37	6.32	4.02	3.26	1.42
RPCGNR(b)	IT	115	67	41	31	12
	CPU	14.89	8.73	5.31	4.03	1.54
PCGNR(a)	IT	213	138	101	77	30
	CPU	15.58	10.16	7.39	5.61	3.67
PCGNR(b)	IT	–	–	–	227	104
	CPU	–	–	–	16.50	7.68

methods RPCG(#), RPCGNR(#) and PCGNR(#) for Problem (P_3). The corresponding numerical results for Problem (P_4) are given in Table 4.6. Roughly speaking, RPCGNR(#) always outperform both RPCG(#) and PCGNR(#) and, for $\nu \geq 1$, its iteration number and CPU time decrease much faster than the latter two methods when ν is increasing. Also, the total CPU time of RPCGNR(#) is competitive with that of either RPCG(#) or PCGNR(#), although its cost at each step is the most expensive. Evidently, the error reduction rate of RPCGNR(#) is much more rapid than that of RPCG(#) or PCGNR(#), and the former is numerically more stable than the latter two.

5. Conclusions

We have established a class of RPCG-based iterative methods for solving large sparse block two-by-two system of linear equations, including the saddle point problem as its special case. The new methods can avoid exact solutions of the linear sub-systems with respect to the involved matrix blocks or the Schur complement of the original coefficient matrix and, therefore, can result in practical and effective solvers for this class of problems. Numerical results have shown that our new methods outperform the existing approaches such as PCGNR for solving large sparse system of linear equations of a block two-by-two structure.

Acknowledgments. This work is supported by the National Basic Research Program (No. 2005CB321702), the China NNSF Outstanding Young Scientist Foundation (No. 10525102), and the National Natural Science Foundation (No. 10471146), P.R. China. The authors are very much indebted to the referees for their constructive comments and helpful suggestions, which greatly improved the original manuscript of this paper.

References

[1] Z.-Z. Bai, Construction and analysis of structured preconditioners for block two-by-two matrices, *J. Shanghai Univ., (English Edition)*, **8** (2004), 397-405.

[2] Z.-Z. Bai, Structured preconditioners for nonsingular matrices of block two-by-two structures, *Math. Comput.*, **75** (2006), 791-815.

[3] Z.-Z. Bai, I.S. Duff and A.J. Wathen, A class of incomplete orthogonal factorization methods. I: methods and theories, *BIT Numer. Math.*, **41** (2001), 53-70.

[4] Z.-Z. Bai and G.-Q. Li, Restrictively preconditioned conjugate gradient methods for systems of linear equations, *IMA J. Numer. Anal.*, **23** (2003), 561-580.

[5] Z.-Z. Bai and Z.-Q. Wang, Restrictive preconditioners for conjugate gradient methods for symmetric positive definite linear systems, *J. Comput. Appl. Math.*, **187** (2006), 202-226.

[6] Z.-Z. Bai, J.-F. Yin and Y.-F. Su, A shift-splitting preconditioner for non-Hermitian positive definite matrices, *J. Comput. Math.*, **24** (2006), 539-552.

[7] H.C. Elman, D.J. Silvester and A.J. Wathen, Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics, Oxford University Press, New York, 2005.

[8] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore and London, 1996.

[9] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, PA, 1997.

[10] V. Girault and P.A. Raviart, Finite Element Methods for Navier-Stokes Equations, Springer-Verlag, Berlin, 1986.

[11] M.R. Hestenes and E.L. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Research National Bureau Standards, Section B*, **49** (1952), 409-435.

[12] A.T. Papadopoulos, I.S. Duff and A.J. Wathen, A class of incomplete orthogonal factorization methods. II. implementation and results, *BIT Numer. Math.*, **45** (2005), 159-179.

[13] Y. Saad, Iterative Methods for Sparse Linear Systems, *PWS Publishing Company*, Boston, 1996.

[14] X. Wang, K.A. Gallivan and R. Bramley, CIMGS: An incomplete orthogonalization preconditioner, *SIAM J. Sci. Comput.*, **18** (1997), 516-536.