A note on an SOR-like method for augmented systems

CHANGJUN LI
Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110015, People’s Republic of China and Department of Mathematics, Northeastern University, Shenyang 110006, People’s Republic of China

ZHENG LI
School of Information Science & Engineering, Northeastern University, Shenyang 110006, People’s Republic of China

DAVID J. EVANS
Department of Computing, Nottingham Trent University, Norttingham NG1 4BU, UK

AND

TIE ZHANG
Department of Mathematics, Northeastern University, Shenyang 110006, People’s Republic of China

[Received on 23 August 2001; revised on 4 December 2002]

Golub et al. (2001, BIT, 41, 71–85) gave a generalized successive over-relaxation method for the augmented systems. In this paper, the connection between the SOR-like method and the preconditioned conjugate gradient (PCG) method for the augmented systems is investigated. It is shown that the PCG method is at least as accurate (fast) as the SOR-like method. Numerical examples demonstrate that the PCG method is much faster than the SOR-like method.

Keywords: augmented systems; least squares problems; generalized successive over-relaxation method (GSOR); SOR-like method; preconditioned conjugate gradient method.

1. Introduction

The augmented system

\[
\begin{pmatrix}
A & B \\
B^T & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix}
=
\begin{pmatrix}
b \\
q
\end{pmatrix},
\]

(1.1)

where \(A \in \mathbb{R}^{m \times m}\) is symmetric and positive definite (SPD), and \(B \in \mathbb{R}^{m \times n}\), appears in many different applications such as the finite-element method for solving the Navier–Stokes equation (Elman & Golub, 1994; Elman & Silvester, 1996; Elman et al., 1997; Fischer et al., 1998). When \(A\) and \(B\) are large and sparse, iterative methods for solving system (1.1) are effective because of storage requirements and preservation of sparsity. Elman et al. (1997), and Fischer et al. (1998) have investigated conjugate gradient (CG)-like methods, the preconditioned MINRES method, the QMR method and the GMRES
method, for solving system (1.1). The difficulty in applying splitting iterative methods such as the Gauss–Seidel (GS) method and the successive over-relaxtion (SOR) method (Young, 1971) to system (1.1) is the singularity of the block diagonal part of the coefficient matrix of the system. Some methods have been developed to overcome this difficulty, such as the Uzawa iteration (Saad, 1996) and the Inexact Uzawa method (Elman & Golub, 1994). In 1998, Li et al. gave a generalized SOR (GSOR) method for solving the above system (1.1) with \(A\) being the identity matrix. The GSOR method involves an acceleration parameter and a preconditioning matrix. In 1999, Li et al. considered the optimum choice for the acceleration parameter. Recently, Golub et al. (2001) further generalized the SOR method for the augmented system (1.1) and gave a new algorithm called the SOR-like method.

Let \(Q\) be a nonsingular and symmetric matrix, and let

\[
\begin{pmatrix}
A & B \\
B^T & 0
\end{pmatrix} = D - L - U
\]

where

\[
D = \begin{pmatrix} A & 0 \\ 0 & Q \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 \\ -B^T & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & -B \\ 0 & Q \end{pmatrix}.
\]

Then, like the classical SOR method (Young, 1971), Golub et al. (2001) defined the following iterative scheme:

\[
(D - \omega L)\begin{pmatrix} x^{(k+1)}_\text{SOR} \\ y^{(k+1)}_\text{SOR} \end{pmatrix} = [(1 - \omega)D + \omega U]\begin{pmatrix} x^{(k)}_\text{SOR} \\ y^{(k)}_\text{SOR} \end{pmatrix} + \omega \begin{pmatrix} b \\ q \end{pmatrix}.
\]

Thus, the SOR-like method can be summarized in the following algorithm:

**Algorithm 1** (SOR-like algorithm Golub et al., 2001.)

1. Choose \(x^{(0)}_\text{SOR}, y^{(0)}_\text{SOR}\), optimum parameter \(\omega\), and preconditioning matrix \(Q\)
2. For \(k = 0, 1, \ldots, \) till convergence Do

   2.1 \(x^{(k+1)}_\text{SOR} = (1 - \omega)x^{(k)}_\text{SOR} + \omega A^{-1}(b - By^{(k)}_\text{SOR})\)
   2.2 \(y^{(k+1)}_\text{SOR} = y^{(k)}_\text{SOR} + \omega Q^{-1}(q - B^Tx^{(k+1)}_\text{SOR})\)

   End Do

The SOR-like method, like the GSOR method given by Li et al. (1998), has an acceleration parameter \(\omega\) and a preconditioning matrix \(Q\). They studied the convergence properties, the optimum choices for the acceleration parameter and the preconditioning matrix. Numerical results showed that the SOR-like method works quite well for the augmented system (1.1) arising from real problems.

Note that if we let \(Q_G\) be the preconditioning matrix used by Golub et al. (2001), then \(Q\) and \(Q_G\) are different by a sign, i.e.

\[
Q = -Q_G.
\]

Thus by choosing different preconditioning matrices \(Q\), Algorithm 1 can cover many methods, which are summarized in Table 1.
Table 1 Different choices for $Q$ results in different methods from SOR-like Algorithm 1

<table>
<thead>
<tr>
<th>$Q$</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = -\alpha B^T B$</td>
<td>Scaled SOR algorithm (Golub et al., 2001)</td>
</tr>
<tr>
<td>$Q = -B^T B$</td>
<td>SOR Algorithm 1 (Golub et al., 2001)</td>
</tr>
<tr>
<td>$Q = -B^T A^{-1} B$</td>
<td>SOR Algorithm 2 (Golub et al., 2001)</td>
</tr>
<tr>
<td>$Q = -\alpha I$</td>
<td>SOR Algorithm 3 (Golub et al., 2001)</td>
</tr>
<tr>
<td>$Q = (1 + \rho)P, \omega = 1/(1 + \rho), A = I$</td>
<td>GSOR method (Li et al., 1998)</td>
</tr>
<tr>
<td>$Q = -1/\alpha, \omega = 1$</td>
<td>Uzawa algorithm (Elman &amp; Golub, 1994; Saad, 1996)</td>
</tr>
</tbody>
</table>

Note that by the convergence analyses of Golub et al. (2001) and Li et al. (1998), the preconditioning matrix $Q$ for the SOR-like method must satisfy that $Q^{-1} B^T A^{-1} B$ has negative real eigenvalues only. It is clear that if $-Q$ is symmetric and positive definite, then $Q^{-1} B^T A^{-1} B$ has negative real eigenvalues only. In this paper we assume that all preconditioning matrices $Q$ for the SOR-like method satisfies that $-Q$ is symmetric and positive definite.

Note also that if we choose $\omega = 1$ in the Algorithm 1, the SOR-like method becomes the Gauss–Seidel method applied to system (1.1) with the coefficient matrix being split as in (1.3). In fact it is also equivalent to the Richardson iterative method (Young, 1971) applied to the linear system

$$ (Q^{-1}B^TA^{-1}B)y = Q^{-1}(B^TA^{-1}b - q). \quad (1.5) $$

It is clear that the above system can be considered to be the SPD system

$$ (B^TA^{-1}B)y = (B^TA^{-1}b - q) \quad (1.6) $$

preconditioned by choosing the preconditioning matrix $M = Q$. The above system (1.6) can be directly obtained from the system (1.1) by eliminating the vector $x$. Here it is assumed that the matrix $B$ is full column rank. Since the above system (1.6) is SPD, there are many iterative methods, especially the preconditioned conjugate gradient (PCG) method that can be directly applied to it. Thus by Algorithm 9.1 on page 247 of Saad (1996) the PCG method applied to system (1.6) is as follows.

**Algorithm 2 (Preconditioned conjugate gradient method.)**

1. Choose $y_{PCG}^{(0)}$, and preconditioning matrix $M$. Then compute $r^{(0)} = B^TA^{-1}b - q - B^TA^{-1}By_{PCG}^{(0)}$, and $z^{(0)} = p^{(0)} = M^{-1}r^{(0)}$
2. For $k = 0, 1, \ldots$, till convergence Do
   2.1 $\alpha_k = (r^{(k)}, z^{(k)})/(p^{(k)}, B^TA^{-1}Bp^{(k)})$
   2.2 $y_{PCG}^{(k+1)} = y_{PCG}^{(k)} + \alpha_k p^{(k)}$
   2.3 $r^{(k+1)} = r^{(k)} - \alpha_k B^TA^{-1}Bp^{(k)}$
   2.4 $z^{(k+1)} = M^{-1}r^{(k+1)}$
   2.5 $\beta_{k+1} = (r^{(k+1)}, z^{(k+1)})/(r^{(k)}, z^{(k)})$
   2.6 $p^{(k+1)} = z^{(k+1)} + \beta_{k+1} p^{(k)}$
End Do
3. Let \( y_{PCG}^{(N)} \) be the final iterate for the above, then solve: \( Ax = b - B y_{PCG}^{(N)} \) for \( x_{PCG}^{(N+1)} \). Thus \( x_{PCG}^{(N+1)} \) and \( y_{PCG}^{(N)} \) are the approximations to the exact solutions \( x^* \) and \( y^* \) respectively of the system (1.1).

Note that the reason we designate the approximations to \( x^* \) by \( x_{PCG}^{(N+1)} \) rather than \( x_{PCG}^{(N)} \) is the consideration of the computational work involved per iterate compared with the SOR-like method. We also note that the preconditioning matrix \( M \) must be symmetric positive definite for the convergence of the PCG method.

The main aim of this paper is to investigate the connection between the SOR-like method and the PCG method. This work is motivated by the early work of Freund (1987) and the present authors (Evans & Li, 1989; Li, 1989). It is shown in the next section that the PCG method is at least as accurate (fast) as the SOR-like method. This is confirmed by our numerical results.

Before we end this section we note that the iterates \( y_{PCG}^{(k)} \) of the PCG method satisfy (by Freund, 1987)

\[
y_{PCG}^{(k)} = y_{PCG}^{(0)} + \text{Span}\{z^{(0)}, (M^{-1}B^T A^{-1}B)^{k-1}z^{(0)}\} = y_{PCG}^{(0)} + K_k(z^{(0)}, M^{-1}B^T A^{-1}B)
\]

and

\[
\|y_{PCG}^{(k)} - y^*\|_{B^T A^{-1}B} = \min_{v \in y_{PCG}^{(0)} + K_k(z^{(0)}, M^{-1}B^T A^{-1}B)} \|y^* - v\|_{B^T A^{-1}B}. 
\]

Here, the norm \( \|y\|_P \) is defined by \( \|y\|_P = (y^T Py)^{1/2} \), where \( P \) is a symmetric positive definite matrix.

2. Main results

**Lemma 1** If we let \( x_{SOR}^{(0)} = A^{-1}b, y_{SOR}^{(0)} = 0 \) in Algorithm 1, then the iterates \( x_{SOR}^{(k)}, y_{SOR}^{(k)} \) of Algorithm 1 satisfy

\[
x_{SOR}^{(k)} \in A^{-1}b + A^{-1}B K_{k-1}(u, Q^{-1}B^T A^{-1}B), \ k \geq 1
\]

\[
y_{SOR}^{(k)} \in K_k(u, Q^{-1}B^T A^{-1}B), \ k \geq 1.
\]

Here,

\[
u = Q^{-1}(B^T A^{-1}b - q), \ K_0(u, Q^{-1}B^T A^{-1}B) = \{0\}.
\]

The proof of Lemma 1 is similar to the proof of the Lemma given by Freund (1987). Therefore its proof is omitted.

Now we can prove one of our main results.

**Theorem 1** Let \( y_{SOR}^{(k)} \) and \( y_{PCG}^{(k)} \) be the \( k \)th iterates of Algorithm 1 and Algorithm 2 respectively. Then we have

\[
\|y_{SOR}^{(k)} - y^*\|_{B^T A^{-1}B} \geq \|y_{PCG}^{(k)} - y^*\|_{B^T A^{-1}B}
\]
if $x^{(0)}_\text{SOR} = A^{-1}b$ and $y^{(0)}_\text{SOR} = 0$ in Algorithm 1, $y^{(k)}_\text{PCG} = 0$ in Algorithm 2, and the two preconditioners $Q$ and $M$ in Algorithms 1 and 2 respectively satisfy: $M = -Q$ and $-Q$ is symmetric and positive definite.

**Proof.** By the conditions of the Theorem, Lemma 1 holds and by properties (1.7) and (1.8) of the PCG method, we have

$$y^{(k)}_\text{PCG} \in K_k(z^{(0)}, M^{-1} B^T A^{-1} B)$$

$$||y^{(k)}_\text{PCG} - y^*||_{B^T A^{-1} B} = \min_{v \in K_k(z^{(0)}, M^{-1} B^T A^{-1} B)} ||y^* - v||_{B^T A^{-1} B}.$$  

Since $M = -Q, z^{(0)} = M^{-1}r^{(0)} = M^{-1}(B^T A^{-1}b - q) = -Q^{-1}(B^T A^{-1}b - q) = -u$ and $K_k(u, Q^{-1} B^T A^{-1} B) = K_k(z^{(0)}, M^{-1} B^T A^{-1} B)$, therefore

$$y^{(k)}_\text{SOR} \in K_k(z^{(0)}, M^{-1} B^T A^{-1} B).$$

Thus

$$\|y^{(k)}_\text{PCG} - y^*\|_{B^T A^{-1} B} \leq \|y^{(k)}_\text{SOR} - y^*\|_{B^T A^{-1} B},$$

concluding the proof of the theorem. □

**THEOREM 2** Let $x^{(k)}_\text{SOR}$ and $y^{(k)}_\text{PCG}$ be the $k$th iterates of Algorithms 1 and 2 respectively. Then under the assumptions of Theorem 1 we have

$$\|x^{(k)}_\text{SOR} - x^*\|_A \geq \|y^{(k)}_\text{PCG} - x^*\|_A.$$

**Proof.** By Lemma 1, Theorem 1 and Algorithm 2,

$$x^{(k)}_\text{SOR} \in A^{-1}b + A^{-1} B K_{k-1}(u, Q^{-1} B^T A^{-1} B)$$

$$= A^{-1}b - A^{-1} B y,$$

where

$$y \in K_{k-1}(u, Q^{-1} B^T A^{-1} B) = K_{k-1}(z^{(0)}, M^{-1} B^T A^{-1} B).$$

Thus,

$$x^{(k)}_\text{SOR} = A^{-1}b - A^{-1} B y$$

$$= x^* - A^{-1} B(y - y^*),$$

where $x^*$ and $y^*$ are the exact solution of the system 1.1. Let $A^{1/2}$ be SPD and satisfy

$$A = A^{1/2} A^{1/2}$$

then $A^{1/2}(x^* - x^{(k)}_\text{SOR}) = A^{-1/2} B(y - y^*)$ and

$$||(x^* - x^{(k)}_\text{SOR})||_A = ||(y - y^*)||_{B^T A^{-1} B}.$$
On the other hand, from Algorithm 2 we have

\[ x^{(k)}_{PCG} = A^{-1} b - A^{-1} B y^{(k-1)}_{PCG}. \]

Hence, we have

\[ A^{1/2} (x^* - x^{(k)}_{PCG}) = A^{-1/2} B (y^{(k-1)}_{PCG} - y^*). \] (2.6)

Therefore

\[ \|x^* - x^{(k)}_{PCG}\|_A = \|y^{(k-1)}_{PCG} - y^*\|_{B^T A^{-1} B}, \] by (2.6)
\[ \leq \|y - y^*\|_{B^T A^{-1} B}, \]
\[ = \|x^* - x^{(k)}_{SOR}\|_A, \] by (2.5)

which completes the proof of Theorem 2. \qed

3. Numerical comparisons

We use the example considered in Golub et al. (2001). Consider a flow in the unit square domain \( \Omega \) governed by the following linear Stokes equation:

\[-\tau \Delta u + p_x = 0 \text{ in } \Omega, \]
\[-\tau \Delta v + p_y = 0 \text{ in } \Omega, \]
\[u_x + v_y = 0 \text{ in } \Omega, \]
\[u = 1 \text{ on } y = 1, \ u = 0 \text{ on other parts of } \Gamma, \]
\[v = 0 \text{ on } \Gamma, \]

where \( \Gamma \) is the boundary of \( \Omega \), \( u \) and \( v \) are the velocity components in the \( x \) and \( y \) directions, and \( p \) is the pressure. The linear finite element method was used to discretize the equation with a triangulation similar to Fig. 1.

For the velocity, we used linear elements and pressure was chosen as a constant on each of the triangles with grid size \( h = 0.0625 \). The details of the discretization and its theoretical consideration can be found in Zhang (1992) The resulting linear system for the discrete solution has the form

\[
\begin{pmatrix}
A & B \\
B^T & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix} =
\begin{pmatrix}
b \\
q
\end{pmatrix}.
\] (3.1)

Here \( x = (u^T, v^T)^T \) gives the approximate velocities at the nodes and mid-points of each triangle, \( y \) gives the approximate pressures on each of the triangles. The matrix \( A \) in \( \mathbb{R}^{450 \times 450} \) is symmetric positive definite, and the matrix \( B \) in \( \mathbb{R}^{450 \times 127} \) has full column rank.

According to the numerical results in Golub et al. (2001),

\[ Q = -B^T B \]
works fine for the SOR-like method. Hence we choose \( Q \) as above for the preconditioner of the SOR-like method. Hence the preconditioner of the PCG method is

\[
M = -Q = B^T B.
\]

The matrices \( A, B \), vectors \( b \) and \( q \) were generated on a PC using the C language. All other computations were done using Matlab. Since we aim to compare the two methods, we did not pay any special attention to solving systems \( Au = v \) and \( Qs = t \). We simply used the Matlab command:

\[
u = A \backslash v \quad \text{and} \quad s = Q \backslash t.
\]

The viscosity constant \( \tau \) was chosen as \( \tau = 1, 0.1, \) and \( 0.01 \) respectively. The initial guesses were \( x_{SOR}^{(0)} = A^{-1} b \) and \( y_{SOR}^{(0)} = y_{PCG}^{(0)} = 0 \). Let \( x^* \) and \( y^* \) be the exact solution to system (3.1). For the SOR-like method, \( x_{SOR}^{(k)} \) and \( y_{SOR}^{(k)} \) are generated during iterations. The optimum parameters \( (\omega_b) \) for the SOR-like method in each case were calculated according to the results of Golub et al. (2001), and are listed in Table 2. But for the PCG method, it only generates vector sequences \( y_{PCG}^{(k)} \). After the method has converged, the corresponding \( x_{PCG}^{(k)} \) is calculated as an approximation to the exact solution. For comparison, the exact solution \( x^* \) and \( y^* \) were found using Matlab. \( x_{PCG}^{(k)} \) was also calculated during each iteration. The following stopping rule:

\[
\frac{\| \delta^{(k)} \|_2}{\| \delta^{(0)} \|_2} \leq \text{eps} \quad (3.2)
\]

was used for both methods. Here \( \text{eps} = 10^{-6} \), and the residual vector is defined as the following:

\[
\delta^{(k)} = \begin{pmatrix} b \\ q \end{pmatrix} - \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix}, \quad (3.3)
\]

where \( x^{(k)} \) and \( y^{(k)} \) are the iterates of the SOR-like method and PCG method. The numbers of iterations (IT) for the SOR-like and PCG methods in each case are listed in Table 2. The CPU times (in seconds) used for the two methods in each case are also given.

It can be seen from Table 2 that the PCG method is preferable over the SOR-like method in terms of the number of iterations and the CPU time needed for convergence. For example, when \( \tau = 1 \), it took 100 iterations and 40.32 s for the SOR-like method, while, it took 24 iterations and 9.83 s for the PCG method. Note that the CPU time does
TABLE 2 Number of iterations (IT) and CPU time needed for the convergence for the SOR-like and PCG methods

<table>
<thead>
<tr>
<th>τ</th>
<th>(ω_b)</th>
<th>SOR-like method (IT / CPU)</th>
<th>PCG method (IT / CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8887</td>
<td>100 / 40.32</td>
<td>24 / 9.83</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2117</td>
<td>172 / 69.10</td>
<td>21 / 8.63</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0697</td>
<td>575 / 231.24</td>
<td>20 / 8.24</td>
</tr>
</tbody>
</table>

FIG. 2. \(\log_{10} \|x^{(k)} - x^*\|_A\) versus the number of iterations for the SOR (thin line) and the PCG (thick line) when \(τ = 1\).

not include the time for generating the matrices \(A, B\), vectors \(b\) and \(q\), and the times for finding the optimum parameter \(ω_b\) for the SOR-like method. Also, the PCG method was not affected much by the choices of the viscosity constant \(τ\) compared with the SOR-like method. The smaller the viscosity \(τ\) is, the greater number of iterations is needed for the convergence of the SOR-like method.

In order to verify the conclusions of Theorems 1 and 2, the norms (errors) \(\|x^{(k)} - x^*\|_A\), \(\|y^{(k)} - y^*\|_{B^{-1}A^{-1}B}\) were recorded for each method; \(\log_{10} \|x^{(k)} - x^*\|_A\), and \(\log_{10} \|y^{(k)} - y^*\|_{B^{-1}A^{-1}B}\) versus the number of iterations when \(τ = 1\) were plotted and are shown in Figs 2 and 3 respectively. In each diagram, the thin line represents the SOR-like method and the thick line represents the PCG method. It is clear from Figs 2 and 3 that the thin lines are above the thick lines, which is consistent with Theorems 1 and 2. Surprisingly, the two figures demonstrate that the PCG method is much better than the SOR-like method. This is true for \(τ = 0.1\) and \(τ = 0.01\) as well.

Also, the norms \(\|x^{(k)} - x^*\|_2\), \(\|y^{(k)} - y^*\|_2\) and \(\|\frac{x^{(k)} - x^*}{y^{(k)} - y^*}\|_2\) were recorded for comparison of the two methods: \(\log_{10} \|x^{(k)} - x^*\|_2\), \(\log_{10} \|y^{(k)} - y^*\|_2\) and...
A NOTE ON AN SOR-LIKE METHOD FOR AUGMENTED SYSTEMS

Fig. 3. \( \log_{10} \| y^{(k)} - y^* \|_{B A - 1 B} \) versus the number of iterations for the SOR (thin line) and the PCG (thick line) when \( \tau = 1 \).

\[
\log_{10} \left\| y^{(k)} - y^* \right\|_2 \text{ versus the number of iterations when } \tau = 1 \text{ were plotted and are shown in Figs 4, 5, 6 respectively. Figures 5 and 6 clearly show that for each iteration }
\]

\[
\| y_{PCG}^{(k)} - y^* \|_2 < \| y_{SOR}^{(k)} - y^* \|_2 \text{ and } \left\| \begin{pmatrix} A_{PCG}^{(k)} - x^* \\ y_{PCG}^{(k)} - y^* \end{pmatrix} \right\|_2 < \left\| \begin{pmatrix} A_{SOR}^{(k)} - x^* \\ y_{SOR}^{(k)} - y^* \end{pmatrix} \right\|_2.
\]

While for the norm \( \| x^{(k)} - x^* \|_2 \), when \( k \geq 2 \), we also have

\[
\| x_{PCG}^{(k)} - y^* \|_2 < \| x_{SOR}^{(k)} - y^* \|_2.
\]

The above phenomena were also valid for the cases \( \tau = 0.1 \) and \( \tau = 0.01 \).

Concluding remarks

- Each of the considered methods needs a preconditioning matrix, which is important for the performance for each method.
- The work per iteration of the considered methods is comparable. This can be seen from Tables 3 and 4. The main work per iteration for each method involves two system solvers, one with coefficient matrix being \( A \) and the other one with coefficient matrix being the preconditioner. When \( m = 1.5n \), then both of them have the same amount of computational work per iteration. If \( m > 1.5n \), the SOR-like method is slightly more expensive; while if \( m < 1.5n \), the PCG method will be slightly more expensive.
- The SOR-like method (Algorithm 1) is simpler, easier to program, and suitable for parallel computations compared with the PCG method (Algorithm 2).
According to our results (Theorems 1 and 2), the PCG method is at least as accurate as the SOR-like method if they use the same number of iterations for the convergence.
Our (sequential) numerical results demonstrated that the PCG method converges much faster than the SOR-like method.

Acknowledgements

The authors would like to thank the referees for their valuable suggestions, which have improved the quality of the paper. The work of CL was supported by the Natural Science...
Foundation of China (grant no 19871011) and the Key Teacher’s Foundation of the Educational Ministry of China.

REFERENCES


