JOURNAL OF THE
CHUNGCHEONG MATHEMATICAL SOCIETY
Volume 8, June 1995

BCG–LIKE METHODS FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS

JAE HEON YUN AND MYUNG SUK JOO

ABSTRACT. This paper proposes two variants of BCG-like method for solving nonsymmetric linear sytems. It is shown that these new algorithms converge faster and more smoothly than the existing BCG and BiCGSTAB algorithms for problems tested in this paper.

1. Introduction

The classical Conjugate Gradient (CG) method of Hestenes and Stiefel [1] with some preconditioning technique is one of the most powerful iterative methods for solving large sparse linear systems $Ax = b$ with symmetric positive definite coefficient matrices A. However, this algorithm fails in general for nonsymmetric linear systems. In the last 15 years, a large number of generalizations of the CG method have been proposed for solving nonsymmetric linear systems [2,5,6,7,8,9]. The Bi-Conjugate Gradient (BCG) method [4], the Conjugate Gradient Squared (CGS) method [2], and the stabilized variant of BCG (Bi-CGSTAB) [3] are typical examples of those for solving nonsymmetric linear systems. Unfortunately, all iterative methods developed to date for nonsymmetric indefinite systems are either slow converging or converge fast only for very special case.

Throughout this paper, we consider variants of BCG-like method for solving nonsymmetric linear systems $Ax = b$, where A is a sparse

Received by the editors on June 30, 1995.
1991 *Mathematics subject classifications*: Primary 65F10.

matrix of order n. For simplicity, all coefficients of A and b are assumed to be real. BCG seems like an ideal algorithm because of its short recurrence (i.e., 3-term recurrence), but in practice it has a few disadvantages:

* The transpose of A is often not available for certain applications, such as linear systems arising in ordinary differential equation solvers.
* Each iteration step requires two matrix-vector multiplications (one with A and the other with the transpose of A) which are double the cost of CG.
* BCG often converges irregularly. In finite precision arithmetic, this irregular convergence behavior may slow down the speed of convergence.

In 1989, Sonneveld proposed the Conjugate Gradient Squared (CGS) method [2] which converges to the exact solution much faster than BCG whenever the CGS converges. One iteration step of the CGS algorithm requires two matrix-vector products with A, but no matrix-vector products at all with the transpose of A. The CGS seems like an attractive variant of the BCG, but, like BCG, it also exhibits rather erratic convergence behavior. This erratic convergence makes the method more sensitive to round-off errors on the finite precision arithmetic. To handle this problem, Van der Vorst [3] proposed the Bi-CGSTAB algorithm which uses local steepest descent steps to obtain a more smoothly convergent CGS-like process. In many cases, it was shown that Bi-CGSTAB converges more smoothly and also often faster than both BCG and CGS.

The purpose of this paper is to propose BCG-like algorithms which converge more smoothly and faster than Bi-CGSTAB. In Section 2, BCG and Bi-CGSTAB algorithms are briefly reviewed and then a new BCG-like algorithm MR-STAB is proposed. We also propose another

BCG-like algorithm COM-STAB which is a combined version of the Bi-CGSTAB and the MR-STAB. In Section 3, we report all computational results for MR-STAB and COM-STAB. The performance of MR-STAB and COM-STAB is compared with Bi-CGSTAB. Finally, some concluding remarks are drawn in Section 4. From now on, A^T denotes the transpose of A, and $(\cdot,\cdot)$ denotes the Euclidean inner product on $\mathbb{R}^n \times \mathbb{R}^n$ and $\|\cdot\|_2$ denotes the Euclidean norm on $\mathbb{R}^n$.

2. BCG-like Algorithms to be Proposed

We first review the BCG method [4] for solving $Ax = b$, where A is a nonsingular and nonsymmetric matrix of order n. The BCG algorithm on which the BCG-like algorithms to be proposed in this paper are based is described below.

```
ALGORITHM 1 : BCG
Choose $x_0$ and then compute $r_0 = b - Ax_0$
Choose $\hat{r}_0$ such that $(\hat{r}_0, r_0) \neq 0$, and set $p_0 = r_0$ and $\hat{p}_0 = \hat{r}_0$
For $i$ = 0, 1, ... , k, ... , until satisfied, do:
    $\alpha_i = (\hat{r}_i, r_i)/(Ap_i, \hat{p}_i)$
    $x_{i+1} = x_i + \alpha_i p_i$
    $r_{i+1} = r_i - \alpha_i A p_i$
    $\hat{r}_{i+1} = \hat{r}_i - \alpha_i A^T \hat{p}_i$
    $\beta_i = (\hat{r}_{i+1}, r_{i+1})/(\hat{r}_i, r_i)$
    $p_{i+1} = r_{i+1} + \beta_i p_i$
    $\hat{p}_{i+1} = \hat{r}_{i+1} + \beta_i \hat{p}_i$
```

From this scheme, $r_i = P_i(A)r_0$, $\hat{r}_i = P_i(A^T)\hat{r}_0$, $p_i = T_i(A)r_0$, and $\hat{p}_i = T_i(A^T)\hat{r}_0$, where $P_i(t)$ and $T_i(t)$ are the i-th degree polynomials with $P_i(0) = 1$. Moreover, $P_i(A)$ and $T_i(A)$ satisfy the following relations :

$$P_0(A) = T_0(A) = I \tag{2.1}$$

$$P_{i+1}(A) = P_i(A) - \alpha_i A T_i(A) \tag{2.2}$$

$$T_{i+1}(A) = P_{i+1}(A) + \beta_i T_i(A) \tag{2.3}$$

The basic properties among the vectors generated by the BCG algorithm which are proved in [4] are as follows:

$$(r_i, \hat{r}_j) = 0 \text{ if } i \neq j \tag{2.4}$$

$$(Ap_i, \hat{p}_j) = 0 \text{ if } i \neq j \tag{2.5}$$

For an arbitrary vector c, $\langle c, Ac, \ldots, A^{m-1}c \rangle$ denotes the subspace spanned by $\{c, Ac, \ldots, A^{m-1}c\}$. From properties (2.4) and (2.5), it follows that

$$r_i \perp \langle \hat{r}_0, A^T\hat{r}_0, \ldots, (A^T)^{i-1}\hat{r}_0 \rangle \tag{2.6}$$

$$Ap_i \perp \langle \hat{r}_0, A^T\hat{r}_0, \ldots, (A^T)^{i-1}\hat{r}_0 \rangle \tag{2.7}$$

Next, we briefly review the existent BCG-like method Bi-CGSTAB [3] which computes an approximation x_k whose residual is of the form $\tilde{r}_k = Q_k(A)P_k(A)r_0$, where $Q_0(t) = 1$ and for $k \geq 1$ $Q_k(t) = (1 - w_1t)(1 - w_2t)\ldots(1 - w_kt)$. The parameter w_k is determined at the k-th iteration so that $\| \tilde{r}_k \|_2$ is minimized. It is clear that $\tilde{r}_0 = r_0 = b - Ax_0$. If we define $\tilde{p}_k = Q_k(A)T_k(A)r_0$, then the Bi-CGSTAB algorithm is as follows:

Algorithm 2 : Bi-CGSTAB

Choose x_0 and then compute $\tilde{r}_0 = b - Ax_0$

```
Choose $\hat{r}_0$ such that $(\hat{r}_0, \tilde{r}_0) \neq 0$ and set $\tilde{p}_0 = \tilde{r}_0$
For $i = 0, 1, 2, \ldots, k, \ldots$, until satisfied, do:
    $\alpha_i = (\hat{r}_0, \tilde{r}_i)/(\hat{r}_0, A\tilde{p}_i)$
    $s_i = \tilde{r}_i - \alpha_i A\tilde{p}_i$
    $t_i = As_i$
    $w_{i+1} = (t_i, s_i)/(t_i, t_i)$
    $x_{i+1} = x_i + \alpha_i \tilde{p}_i + w_{i+1} s_i$
    $\tilde{r}_{i+1} = s_i - w_{i+1} t_i$
    If $\| \tilde{r}_{i+1} \|_2 <$ (tolerance), then stop
    $\beta_i = \{(\hat{r}_0, \tilde{r}_{i+1})/(\hat{r}_0, \tilde{r}_i)\}\{\alpha_i / w_{i+1}\}$
    $\tilde{p}_{i+1} = \tilde{r}_{i+1} + \beta_i(\tilde{p}_i - w_{i+1} A\tilde{p}_i)$
```

Now, we will consider the Minimal residual Bi-CGSTAB (called MR-STAB from now on) which is a new variant of BCG-like method. Notice that the polynomial $Q_k(t)$ in Bi-CGSTAB is a product of k linear polynomials with $Q_k(0) = 1$. However, the MR-STAB algorithm to be developed below uses a polynomial $Q_{2k}^*(t)$ which is a product of k quadratic polynomials. In other words, the MR-STAB computes an approximation x_{2k} whose residual is of the form $r_{2k}^* = Q_{2k}^*(A)P_{2k}(A)r_0$, where $Q_0^*(t) = 1$ and for $k \geq 1$ $Q_{2k}^*(t) = (1 + w_1 t + w_2 t^2)(1 + w_3 t + w_4 t^2) \cdots (1 + w_{2k-1} t + w_{2k} t^2)$, and the parameters w_{2k-1} and w_{2k} are determined at the k-th iteration so that $\| r_{2k}^* \|_2$ is minimized. Clearly, $r_0^* = r_0 = b - Ax_0$.

To derive the MR-STAB algorithm, for each i let

$$\begin{aligned} r_{2i}^* &= Q_{2i}^*(A) r_{2i} = Q_{2i}^*(A) P_{2i}(A) r_0 \\ p_{2i}^* &= Q_{2i}^*(A) p_{2i} = Q_{2i}^*(A) T_{2i}(A) r_0 \end{aligned} \tag{3.1}$$

where $P_{2i}(A)$ and $T_{2i}(A)$ are defined as in (2.1) - (2.3). We will describe how r_{2i+2}^* and p_{2i+2}^* can be generated from given vectors r_{2i}^* and p_{2i}^*.

Denote

$$\begin{aligned} \bar r_{2i+1} &= Q^*_{2i}(A) r_{2i+1} = r^*_{2i} - \alpha_{2i} A p^*_{2i} \\ \bar p_{2i+1} &= Q^*_{2i}(A) p_{2i+1} = \bar r_{2i+1} + \beta_{2i} p^*_{2i} \\ \bar r_{2i+2} &= Q^*_{2i}(A) r_{2i+2} = \bar r_{2i+1} - \alpha_{2i+1} A \bar p_{2i+1}. \end{aligned} \tag{3.2}$$

Using (3.1) and (3.2), it follows that

$$\begin{aligned} r^*_{2i+2} &= Q^*_{2i+2}(A) r_{2i+2} \\ &= (I + w_{2i+1}A + w_{2i+2}A^2) Q^*_{2i}(A) r_{2i+2} \\ &= \bar r_{2i+2} + w_{2i+1} A \bar r_{2i+2} + w_{2i+2} A^2 \bar r_{2i+2} \end{aligned} \tag{3.3}$$

$$\begin{aligned} p^*_{2i+2} &= Q^*_{2i+2}(A) p_{2i+2} \\ &= Q^*_{2i+2}(A) r_{2i+2} + \beta_{2i+1} Q^*_{2i+2}(A) p_{2i+1} \\ &= r^*_{2i+2} + \beta_{2i+1}(I + w_{2i+1}A + w_{2i+2}A^2) Q^*_{2i}(A) p_{2i+1} \\ &= r^*_{2i+2} + \beta_{2i+1}(\bar p_{2i+1} + w_{2i+1} A \bar p_{2i+1} + w_{2i+2} A^2 \bar p_{2i+1}) \end{aligned} \tag{3.4}$$

The parameters α_{2i}, β_{2i}, α_{2i+1}, and β_{2i+1} used in equations (3.2) and (3.4) are the same as those in the BCG. To express these 4 parameters in terms of new vectors - r^*_{2i}, p^*_{2i}, $\bar r_{2i+1}$, $\bar p_{2i+1}$, $\bar r_{2i+2}$, - which are denoted in (3.1) and (3.2), we use the properties (2.6) and (2.7) that the BCG algorithm satisfies. Notice that the highest order term of $Q^*_{2i}(A^T)$ is $w_2 w_4 \cdots w_{2i} (A^T)^{2i}$ by the definition of $Q^*_{2i}(t)$ and the highest order term of $P_{2i}(A^T)$ is $(-1)^{2i}\alpha_0\alpha_1 \cdots \alpha_{2i-1}(A^T)^{2i}$ from relations (2.1) to (2.3). Using these facts, one obtains

$$\begin{aligned} \alpha_{2i} &= (\hat r_{2i}, r_{2i})/(Ap_{2i}, \hat p_{2i}) \\ &= \frac{((A^T)^{2i}\hat r_0, r_{2i})(-1)^{2i}\alpha_0\alpha_1\cdots\alpha_{2i-1}}{((A^T)^{2i}\hat r_0, Ap_{2i})(-1)^{2i}\alpha_0\alpha_1\cdots\alpha_{2i-1}} \\ &= \frac{((A^T)^{2i}\hat r_0, r_{2i}) w_2 w_4 \cdots w_{2i}}{((A^T)^{2i}\hat r_0, Ap_{2i}) w_2 w_4 \cdots w_{2i}} \\ &= (r_{2i}, Q^*_{2i}(A^T)\hat r_0)/(Ap_{2i}, Q^*_{2i}(A^T)\hat r_0) \\ &= (r^*_{2i}, \hat r_0)/(Ap^*_{2i}, \hat r_0) \end{aligned} \tag{3.5}$$

$$
\begin{aligned}
\beta_{2i} &= (\hat{r}_{2i+1}, r_{2i+1})/(\hat{r}_{2i}, r_{2i}) \\
&= \frac{((A^T)^{2i+1}\hat{r}_0, r_{2i+1})(-1)^{2i+1}\alpha_0\alpha_1\dots\alpha_{2i}}{((A^T)^{2i}\hat{r}_0, r_{2i})(-1)^{2i}\alpha_0\alpha_1\dots\alpha_{2i-1}} \\
&= (-\alpha_{2i})\frac{((A^T)^{2i+1}\hat{r}_0, r_{2i+1})w_2w_4\cdots w_{2i}}{((A^T)^{2i}\hat{r}_0, r_{2i})w_2w_4\cdots w_{2i}} \\
&= (-\alpha_{2i})(Q^*_{2i}(A^T)\hat{r}_0, Ar_{2i+1})/(Q^*_{2i}(A^T)\hat{r}_0, r_{2i}) \\
&= (-\alpha_{2i})(A\bar{r}_{2i+1}, \hat{r}_0)/(r^*_{2i}, \hat{r}_0)
\end{aligned}
\tag{3.6}
$$

$$
\begin{aligned}
\alpha_{2i+1} &= (\hat{r}_{2i+1}, r_{2i+1})/(Ap_{2i+1}, \hat{p}_{2i+1}) \\
&= \frac{(r_{2i+1}, (A^T)^{2i+1}\hat{r}_0)}{(Ap_{2i+1}, (A^T)^{2i+1}\hat{r}_0)} \\
&= (A\bar{r}_{2i+1}, \hat{r}_0)/(A^2\bar{p}_{2i+1}, \hat{r}_0)
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
\beta_{2i+1} &= (\hat{r}_{2i+2}, r_{2i+2})/(\hat{r}_{2i+1}, r_{2i+1}) \\
&= (-\alpha_{2i+1})\frac{(r_{2i+2}, (A^T)^{2i+2}\hat{r}_0)}{(r_{2i+1}, (A^T)^{2i+1}\hat{r}_0)} \\
&= (-\alpha_{2i+1})(A^2\bar{r}_{2i+2}, \hat{r}_0)/(A\bar{r}_{2i+1}, \hat{r}_0)
\end{aligned}
\tag{3.8}
$$

It can be seen from equalities (3.5) to (3.8) that six matrix-vector products with A are required to determine four parameters α_{2i}, β_{2i}, α_{2i+1}, and β_{2i+1}. This is too expensive, so that an efficient way to find these parameters for given r^*_{2i} and p^*_{2i} is described below:

1. Compute Ap^*_{2i} and then determine α_{2i} using (3.5)
2. $\bar{r}_{2i+1} = r^*_{2i} - \alpha_{2i}Ap^*_{2i}$
3. Compute $A\bar{r}_{2i+1}$ and then determine β_{2i} using (3.6)
4. $\bar{p}_{2i+1} = \bar{r}_{2i+1} + \beta_{2i}p^*_{2i}$
5. $A\bar{p}_{2i+1} = A\bar{r}_{2i+1} + \beta_{2i}Ap^*_{2i}$
6. Compute $A(A\bar{p}_{2i+1})$ and then determine α_{2i+1} using (3.7)
7. $\bar{r}_{2i+2} = \bar{r}_{2i+1} - \alpha_{2i+1}A\bar{p}_{2i+1}$

```
8. $A\bar{r}_{2i+2} = A\bar{r}_{2i+1} - \alpha_{2i+1}A(A\bar{p}_{2i+1})$
9. Compute $A(A\bar{r}_{2i+2})$ and then determine $\beta_{2i+1}$ using (3.8)
```

As can be seen above, these procedures require only **four** matrix-vector products with A (i.e., each for steps 1, 4, 6, and 9) which is a significant reduction as compared with six matrix-vector products with A. The next thing to do is to determine w_{2i+1} and w_{2i+2} in order to compute r^*_{2i+2} in equation (3.3). The w_{2i+1} and w_{2i+2} are determined so that $\| r^*_{2i+2} \|_2$ is minimized. After r^*_{2i+2} is computed, p^*_{2i+2} can be easily computed using equation (3.4). Therefore, we obtain the following algorithm:

```
ALGORITHM 3 : MR-STAB
1. Choose $x_0$ and then compute $r^*_0 = b - Ax_0$
2. Choose $\hat{r}_0$ such that $(\hat{r}_0, r^*_0) \neq 0$ and set $p^*_0 = r^*_0$
For $i = 0, 2, 4, \ldots, 2k, \ldots,$ until satisfied, do:
3. Compute $Ap^*_i$
4. $\alpha_i = (r^*_i, \hat{r}_0)/(Ap^*_i, \hat{r}_0)$
5. $\bar{x}_{i+1} = x_i + \alpha_i p^*_i$
6. $\bar{r}_{i+1} = r^*_i - \alpha_i Ap^*_i$
7. Compute $A\bar{r}_{i+1}$
8. $\beta_i = (-\alpha_i)\{(A\bar{r}_{i+1}, \hat{r}_0)/(r^*_i, \hat{r}_0)\}$
9. $\bar{p}_{i+1} = \bar{r}_{i+1} + \beta_i p^*_i$
10. $A\bar{p}_{i+1} = A\bar{r}_{i+1} + \beta_i Ap^*_i$
11. Compute $A(A\bar{p}_{i+1})$
12. $\alpha_{i+1} = (A\bar{r}_{i+1}, \hat{r}_0)/(A(A\bar{p}_{i+1}), \hat{r}_0)$
13. $\bar{x}_{i+2} = \bar{x}_{i+1} + \alpha_{i+1}\bar{p}_{i+1}$
14. $\bar{r}_{i+2} = \bar{r}_{i+1} - \alpha_{i+1}A\bar{p}_{i+1}$
15. $A\bar{r}_{i+2} = A\bar{r}_{i+1} - \alpha_{i+1}A(A\bar{p}_{i+1})$
16. Compute $A(A\bar{r}_{i+2})$
17. Find $w_{i+1}$ and $w_{i+2}$ such that $\| r^*_{i+2} \|_2$ is minimized
18. $x_{i+2} = \bar{x}_{i+2} - w_{i+1}\bar{r}_{i+2} - w_{i+2}A\bar{r}_{i+2}$
```

```
19. $r^*_{i+2} = \bar{r}_{i+2} + w_{i+1}A\bar{r}_{i+2} + w_{i+2}A(A\bar{r}_{i+2})$
    If $\| r^*_{i+2} \|_2 <$ (tolerance), then stop
20. $\beta_{i+1} = (-\alpha_{i+1})\{(A(A\bar{r}_{i+2}), \hat{r}_0)/(A\bar{r}_{i+1}, \hat{r}_0)\}$
21. $p^*_{i+2} = r^*_{i+2} + \beta_{i+1}(\bar{p}_{i+1} + w_{i+1}A\bar{p}_{i+1} + w_{i+2}A^2\bar{p}_{i+1})$
```

Notice that $\bar{r}_{i+1} = b - A\bar{x}_{i+1}$ and $\bar{r}_{i+2} = b - A\bar{x}_{i+2}$ in the MR-STAB. Since the properties (2.6) and (2.7) in the BCG are utilized implicitly, the MR-STAB method is also a finite method, i.e., in exact arithmetic it will terminate in at most n iteration steps unless it breaks down.

It can be easily seen that the MR-STAB algorithm requires **four** matrix-vector products with A (i.e., each for steps 3, 7, 11, and 16) to execute two iteration steps. Here, execution of two iteration steps in the MR-STAB means all computational steps for computing new vectors x_{i+2}, p^*_{i+2}, and r^*_{i+2} from previously computed vectors x_i, p^*_i, and r^*_i (i.e., steps 3 through 21 in Algorithm 3). The computation of w_i and w_{i+1} in step 17 can be easily done using equation (3.3) and the following relations:

$$(r^*_{i+2}, A\bar{r}_{i+2}) = 0$$
$$(r^*_{i+2}, A^2\bar{r}_{i+2}) = 0$$

Thus, step 17 requires 5 Inner product operations. The operation counts of the Bi-CGSTAB and the MR-STAB listed in Table 1 are those required to execute two iteration steps.

Table 1 : The operation counts of Bi-CGSTAB and MR-STAB

Operation Type	Bi-CGSTAB	MR-STAB
Matrix-vector product	4	4
Inner product	8	10
Vector update	12	14

Since the $\| r^*_{i+2} \|_2$ in step 17 is minimized over two dimensional vector space R^2, it may be expected that the MR-STAB converges faster than the Bi-CGSTAB in which residual norm is minimized over one dimensional vector space R^1. Numerical experiments in Section 3 also show this fact. Table 1 shows that the MR-STAB requires two more operation counts for each of Inner product and Vector update than the Bi-CGSTAB. However, this is not a big problem because MR-STAB converges faster than Bi-CGSTAB (see Fig. 1 through Fig. 4) and the Inner product and Vector update are much less expensive operations than the Matrix-vector product.

To take advantages of both Bi-CGSTAB and MR-STAB, we propose a combined version of Bi-CGSTAB and MR-STAB which is now called the COM-STAB algorithm. In other words, the COM-STAB algorithm executes the Bi-CGSTAB and the MR-STAB alternately. By doing so, the average operation counts per iteration of COM-STAB become slightly smaller than those of MR-STAB. The COM-STAB algorithm is as follows:

```
ALGORITHM 4 : COM-STAB
Choose $x_0$ and then compute $\tilde{r}_0 = b - Ax_0$
Choose $\hat{r}_0$ such that $(\hat{r}_0, \tilde{r}_0) \neq 0$ and set $\tilde{p}_0 = \tilde{r}_0$
For $i = 0, 3, 6, \ldots, 3k, \ldots$, until satisfied, do:
   Find $x_{i+1}$, $\tilde{r}_{i+1}$, and $\tilde{p}_{i+1}$ using Bi-CGSTAB which starts
      with $x_i$, $\tilde{r}_i$, and $\tilde{p}_i$
   If $\| \tilde{r}_{i+1} \|_2 <$ (tolerance), then stop
   Set $r^*_{i+1} = \tilde{r}_{i+1}$ and $p^*_{i+1} = \tilde{p}_{i+1}$
   Find $x_{i+3}$, $r^*_{i+3}$, and $p^*_{i+3}$ using MR-STAB which starts
      with $x_{i+1}$, $r^*_{i+1}$, and $p^*_{i+1}$
   If $\| r^*_{i+3} \|_2 <$ (tolerance), then stop
   Set $\tilde{r}_{i+3} = r^*_{i+3}$ and $\tilde{p}_{i+3} = p^*_{i+3}$
```

3. Numerical Results

In this section, we consider linear systems $Ax = b$ with banded Toeplitz matrices A of order n, since the discretization of the partial differential equations often leads to such a matrix or a low-rank modification of one. All numerical experiments have been carried out in double precision floating point arithmetic. The right-hand side vector b was chosen so that the exact solution x_t to $Ax = b$ is $x_t = (1, 1, \ldots, 1)^T$. In all cases, the iteration was started with initial approximate solution $x_0 = (2, 2, \ldots, 2)^T$, and the iteration stopped when the norm of residual is less than 10^{-6}. All numerical results for $n = 200$ and 400 are summarized in Fig. 1 through Fig. 4.

EXAMPLE 4.1. This example is one in which Bi-CGSTAB converges fairly smoothly. The matrix A is given as follows.

$$A = \begin{pmatrix} 4 & -2 & 0 & 0 & \ldots & 0 \\ 1 & 4 & -2 & 0 & \ldots & 0 \\ 0 & 1 & 4 & -2 & \ldots & 0 \\ 0 & 0 & 1 & 4 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 4 \end{pmatrix}$$

As can be seen from Fig. 1 and Fig. 2, the MR-STAB and COM-STAB converge faster and more smoothly than the Bi-CGSTAB. In addition, the COM-STAB converges as fast as the MR-STAB, which means that the COM-STAB performs slightly better than the MR-STAB since the average operation counts per iteration of COM-STAB is less than those of MR-STAB.

EXAMPLE 4.2. This example is one in which Bi-CGSTAB converges irregularly. The matrix A is given as follows.

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 2 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 2 & 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & 2 & 1 & \dots & 0 \\ 0 & 0 & 1 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 2 \end{pmatrix}$$

As can be seen from Fig. 3 and Fig. 4, the MR-STAB and COM-STAB converge faster and more smoothly than the Bi-CGSTAB, and the COM-STAB converges faster than the MR-STAB.

4. Concluding Remarks

For the problems considered in this paper, the MR-STAB and COM-STAB converge faster and more smoothly than the Bi-CGSTAB which was recently proposed by Van der Vorst for the purpose of remedying disadvantages of the BCG. Also notice that the COM-STAB algorithm which executes the Bi-CGSTAB and MR-STAB alternately performs slightly better than the MR-STAB algorithm since the former converges as fast as or better than the latter. Therefore, we have some preference of the COM-STAB over the MR-STAB. However, we need further research to see how well the COM-STAB performs over the MR-STAB for other problems. On the basis of our work, we recommend the use of MR-STAB or COM-STAB instead of Bi-CGSTAB to solve large sparse nonsymmetric linear systems.

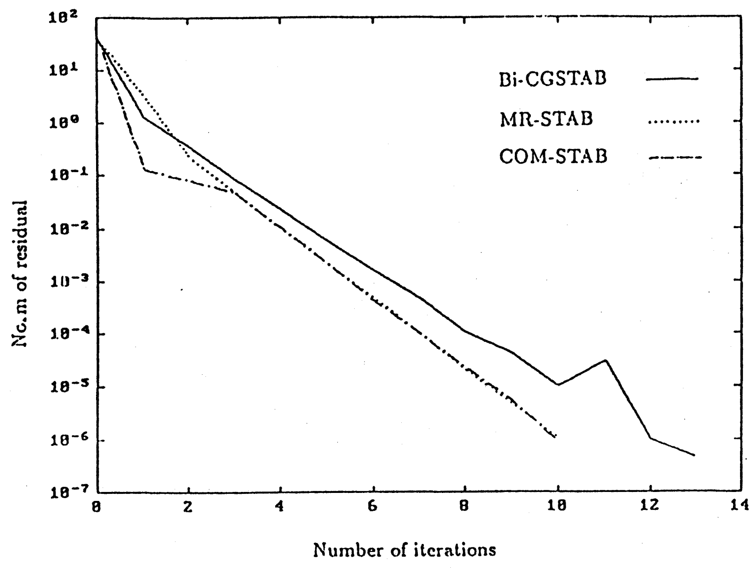

FIG. 1. PERFORMANCE OF BI-CGSTAB, MR-STAB, AND COM-STAB (EXAMPLE 4.1., $n = 200$)

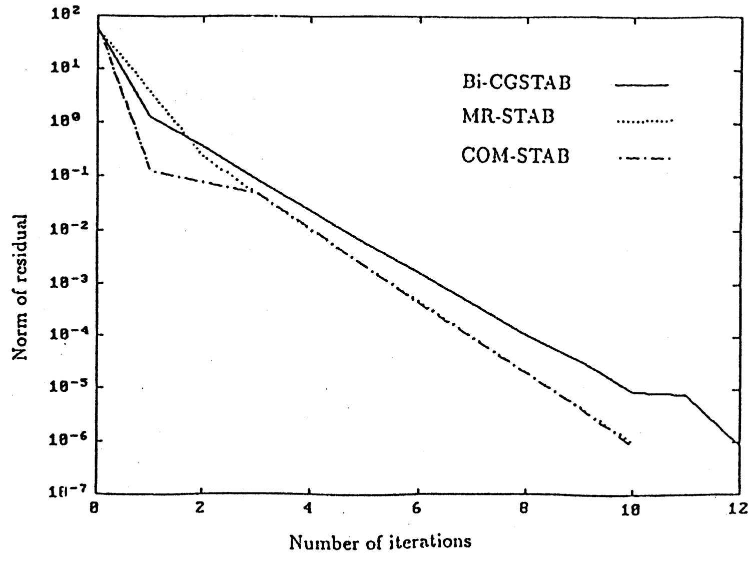

FIG. 2. PERFORMANCE OF BI-CGSTAB, MR-STAB, AND COM-STAB (EXAMPLE 4.1., $n = 400$)

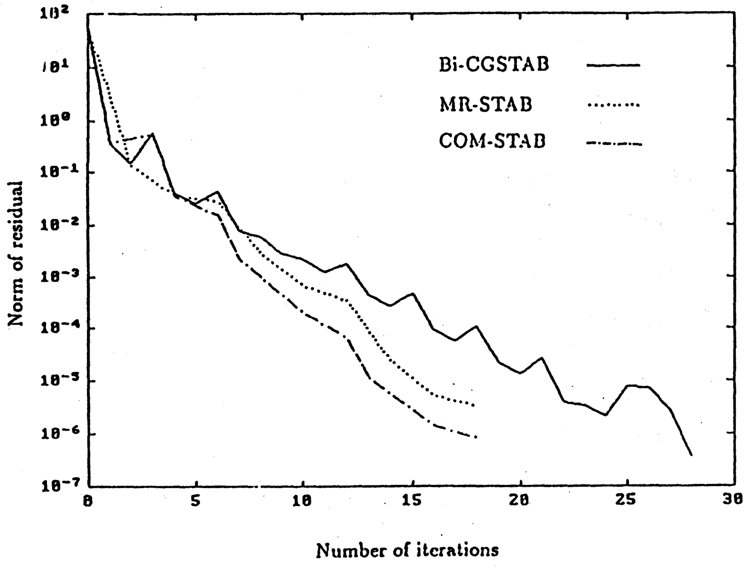

FIG. 3. PERFORMANCE OF BI-CGSTAB, MR-STAB, AND COM-STAB (EXAMPLE 4.2., $n = 200$)

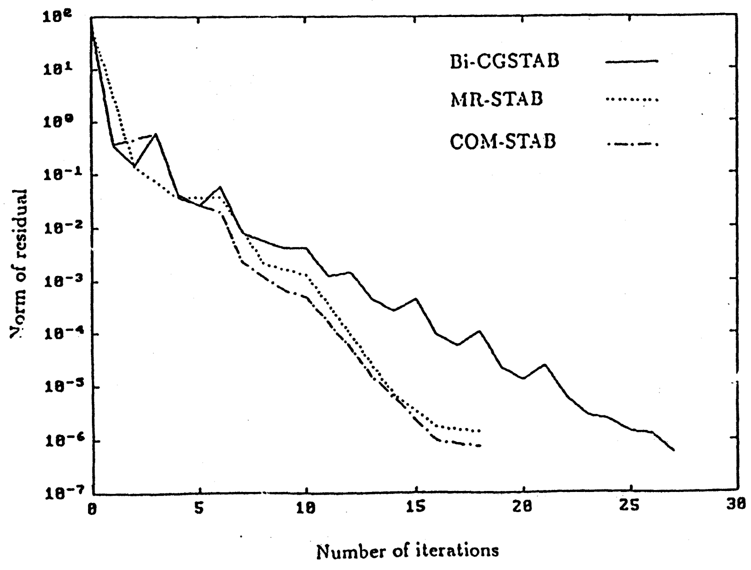

FIG. 4. PERFORMANCE OF BI-CGSTAB, MR-STAB, AND COM-STAB (EXAMPLE 4.2., $n = 400$)

REFERENCES

1. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards **49** (1952), 409-435.
2. P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **10** (1989), 36-52.
3. H. A. Van der Vorst, SIAM J. Sci. Stat. Comput **13** (1992), 631-644.
4. R. Fletcher, *Conjugate gradient methods for indefinite systems*, vol. 506, Lecture Notes in Math., Springer-Verlag, Berlin, New York, 1976, pp. 73-89.
5. S. C. Eisenstat, H. C. Elman, and M. H. Schultz, SIAM J. Numer. Anal. **20** (1983), 345-357.
6. H. C. Elman, *Iterative methods for large sparse nonsymmetric systems of linear equations, Research Report 229*, Yale University, 1982.
7. Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp. **37** (1981), 105-126.
8. Y. Saad and M. H. Schultz, *GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7** (1986), 856-869.
9. Jae H. Yun, *Efficient parallel iterative method for solving large nonsymmetric linear systems*, Comm. Korean Math. Soc. **9** (1994), 449-465.

DEPARTMENT OF MATHMETICS
COLLEGE OF NATURAL SCIENCES
CHUNGBUK NATIONAL UNIVERSITY
CHEONGJU 360-763, KOREA