SIAM J. SCI. STAT. COMPUT.
Vol. 13, No. 2, pp. 631–644, March 1992

012

BI-CGSTAB: A FAST AND SMOOTHLY CONVERGING VARIANT OF BI-CG FOR THE SOLUTION OF NONSYMMETRIC LINEAR SYSTEMS*

H. A. VAN DER VORST†

Abstract. Recently the Conjugate Gradients-Squared (CG-S) method has been proposed as an attractive variant of the Bi-Conjugate Gradients (Bi-CG) method. However, it has been observed that CG-S may lead to a rather irregular convergence behaviour, so that in some cases rounding errors can even result in severe cancellation effects in the solution. In this paper, another variant of Bi-CG is proposed which does not seem to suffer from these negative effects. Numerical experiments indicate also that the new variant, named Bi-CGSTAB, is often much more efficient than CG-S.

Key words. Bi-CG, CG-S, nonsymmetric linear systems, iterative solver, preconditioning

AMS(MOS) subject classification. 65F10

1. Introduction. In recent years the Conjugate Gradients-Squared (CG-S) method [8] has been recognized as an attractive variant of the Bi-Conjugate Gradient (Bi-CG) iterative method for the solution of certain classes of nonsymmetric linear systems. Recent studies indicate that the method is often competitive with other established methods, such as GMRES [7]. For such comparative studies see, e.g., [1], [2], [6].

In many situations, however, one is faced with a quite irregular convergence behaviour of CG-S, in particular in situations when starting the iteration close to the solution. This is a common situation in, e.g., the final stages of some nonlinear solvers and in time dependent problems. The then often occurring irregularities in the iteration process may even lead to severe cancellation, spoiling the solution delivered by the process.

This motivated our search for a more smoothly converging variant of Bi-CG, without giving up the attractive speed of convergence of CG-S. It appeared that an early predecessor of CG-S, named IDR [10], could be reformulated to a method that shows the desired properties (at least for a large number of test cases).

In this paper we give some relevant background for the CG-S method in §2, which provides a suitable framework for the presentation of the new variant Bi-CGSTAB in §3. The various ways in which preconditioning can be incorporated in the algorithm are discussed in §4. In §5 we show, by suitable examples, that Bi-CGSTAB may be very attractive in comparison with CG-S in many situations.

2. Conjugate gradients-squared (CG-S). The residual vectors r_i, generated by the CG method, satisfy a 3-term recurrence relation. This 3-term recurrence relation is not used explicitly in CG, but it is fundamental for the efficiency of the method. When A is not symmetric we lose this property for the r_i's. Moreover, when A is not positive definite, then $\| \ \|_A$ does not define a norm, so that it does not make sense to minimize $\| x - x_i \|_A$.

In the Bi-CG method [3], the approximations are constructed in such a way that the residual r_j is orthogonal with respect to another row of vectors $\hat{r}_0, \hat{r}_1, \cdots, \hat{r}_{j-1}$,

* Received by the editors May 21, 1990; accepted for publication (in revised form) February 18, 1991.

† Mathematical Institute, University of Utrecht, Budapestlaan 6, NL-3584 CD Utrecht, the Netherlands (vorst@math.ruu.nl).

and, vice versa, $\hat{r}_j$ is orthogonal with respect to $r_0, r_1, \cdots, r_{j-1}$. This can be accomplished by two 3-term recurrence relations for the rows $\{r_j\}$ and $\{\hat{r}_j\}$. The Bi-CG method also terminates within n steps at most (when A is an n by n matrix), but there is no minimization property as in CG for the intermediate steps.

Sonneveld [8] made the observation that, in the case of convergence, both the rows $\{r_j\}$ and $\{\hat{r}_j\}$ converge towards zero, but that only the convergence of the $\{r_j\}$ is exploited. He proposes the following modification to Bi-CG by which all the convergence effort is concentrated in the r_i vectors. For the Bi-CG vectors it is well known that they can be written as $r_j = P_j(A)r_0$ and $\hat{r}_j = P_j(A^T)\hat{r}_0$, and because of the bi-orthogonality relation we have that

$$\begin{aligned}(r_j, \hat{r}_i) &= (P_j(A)r_0, P_i(A^T)\hat{r}_0) \\ &= (P_i(A)P_j(A)r_0, \hat{r}_0) = 0 \quad \text{for } i < j.\end{aligned}$$

The iteration parameters for Bi-CG are computed from innerproducts like the above. Sonneveld observed that we can also construct the vectors $\tilde{r}_j = P_j^2(A)r_0$, using only the latter form of the innerproduct for recovering the Bi-CG parameters (which implicitly define the polynomial P_j). By doing so, it can be avoided that the vectors $\hat{r}_j$ have to be formed, nor is there any multiplication with the matrix A^T.

The resulting algorithm can be represented by the following scheme.

UNPRECONDITIONED CG-S ALGORITHM.

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\hat{r}_0$ is an arbitrary vector, such that $(r_0, \hat{r}_0) \neq 0$,
    e.g., $\hat{r}_0 = r_0$;
$\rho_0 = 1; p_0 = q_0 = 0$;
for $i = 1, 2, 3, \cdots$,
    $\rho_i = (\hat{r}_0, r_{i-1})$;
    $\beta = \rho_i / \rho_{i-1}$;
    $u = r_{i-1} + \beta q_{i-1}$;
    $p_i = u + \beta(q_{i-1} + \beta p_{i-1})$;
    $v = Ap_i$;
    $\alpha = \rho_i / (\hat{r}_0, v)$;
    $q_i = u - \alpha v$;
    $\hat{u} = u + q_i$;
    $x_i = x_{i-1} + \alpha w$;
    if x_i is accurate enough then quit;
    $r_i = r_{i-1} - \alpha A w$;
end

One iteration step of CG-S involves about as many arithmetical operations as one step of Bi-CG.

The last line of the Unpreconditioned CG-S Algorithm could be replaced by the computation of the true residual vector $r_i = b - Ax_i$. However, in many of the experiments, especially those with the typical jumping convergence behaviour of CG-S, it has been seen that this has a negative effect on the iteration process, i.e., it may take many more iteration steps for this *true residual* process to get at an x_i of similar accuracy as in the CG-S algorithm in its form described above. In some cases the *true residual* process did even not converge whereas CG-S did.

An example of such a situation is discussed in §5.4. A possible explanation might be that locally large variations in a current update direction overshadow variations in

other almost converged directions, so that the true residual vector does not necessarily satisfy the underlying orthogonality relations for the updated vectors $P_i(A)r_0$. This aspect needs further research.

If the Bi-CG polynomial $P_i(A)$ is viewed as a reductor working on r_0, then this reductor is now applied twice in the new method and this explains the name *conjugate gradients-squared.*

The weak point in this way of reasoning is that the reduction operator $P_i(A)$ in Bi-CG is very dependent on the starting initial residual r_0, and that it is not likely to be a reduction operator for any other vector, not even for the particular vector $P_i(A)r_0$ itself. Indeed, it is not very difficult to construct examples for which $P_i(A)r_0$ is small in norm and for which ${P^2}_i(A)r_0$ is even much bigger in norm then r_0 is.

Such a phenomenon may happen in the following situation. Suppose that r_0 has small coordinates in some eigenvector directions of the matrix A. Then $P_i(\lambda)$ may take very large values in the corresponding eigenvalues, in particular when these eigenvalues are more or less isolated from the others, without leading to a significant contribution of that eigenvector direction to the norm of r_i in Bi-CG. However, the value of ${P^2}_i(\lambda)$ may be so large that the contribution of the corresponding eigenvector direction even dominates in the residual r_i of CG-S.

These situations occur quite often, and even during one convergence history we may observe many local peaks in the convergence curve for CG-S (see, e.g., the examples in §5). These peaks do not seem to delay the convergence of CG-S. However, in practical situations they may have quite adverse effects, since they may be so large that the local corresponding corrections to the current iterate result in cancellation. As a result, the final solution may have little significance, which can be checked by computing the real residual (instead of the updated residual as is done usually in these processes). This is a very serious problem with CG-S and in our examples in §5 we will encounter significant losses in accuracy.

For a more detailed discussion on the convergence behaviour of CG-S, see [9].

3. A more smoothly converging variant of CG-S. We have mentioned that the residuals r_i in CG-S satisfy the relation $r_i = P_i(A)^2 r_0$, in which $P_i(A)r_0$ just defines the residual $r_{bi-CG,i}$ in the Bi-CG method:

$$r_{bi-CG,i} = P_i(A)r_0.$$

By construction we have that $(P_i(A)r_0, P_j(A^T)\hat{r}_0) = 0$ for $j < i$, which expresses the fact that $P_i(A)r_0$ is perpendicular to the subspace $K^i(A^T; \hat{r}_0)$, spanned by the vectors

$$\hat{r}_0, A^T\hat{r}_0, \cdots, (A^T)^{i-1}\hat{r}_0.$$

This implies that, in principle, we can also recover the Bi-CG iteration parameters by requiring that, e.g., r_i is perpendicular to $\tilde{P}_j(A^T)\hat{r}_0$, or, equivalently, that $(\tilde{P}_j(A)P_i(A)r_0, \hat{r}_0) = 0$, for another suitable set of polynomials $\tilde{P}_j$ of degree j. In Bi-CG one takes $\tilde{P}_j = P_j$, namely, $\hat{r}_j = P_j(A^T)\hat{r}_0$. This is exploited in CG-S, as we have indicated before, since recursion relations for the vectors $P_j^2(A)r_0$ can be derived from those for $P_j(A)r_0$.

Of course, we can construct other iteration methods, by which x_i are generated so that $r_i = \tilde{P}_i(A)P_i(A)r_0$ with other ith degree polynomials, like, e.g., Chebychev polynomials, which might be more suitable. Unfortunately, the optimal parameters for the Chebychev polynomials are in general not easily obtainable and also the recurrence

relations for the resulting method are more complicated than for CG-S. Another possibility is to take for $\tilde{P}_j$ a polynomial of the form

$$Q_i(x) = (1 - \omega_1 x)(1 - \omega_2 x) \cdots (1 - \omega_i x), \tag{1}$$

and to select suitable constants ω_j. This expression leads to an almost trivial recurrence relation for the Q_i.

An obvious possibility to select ω_j in the jth iteration step is to minimize r_j, with respect to ω_j, for residuals that can be written as $r_j = Q_j(A)P_j(A)r_0$. This leads to a method which is mathematically equivalent with the IDR method described in [10]. IDR, however, in the form as it is described in [10] may also suffer from severe cancellation and can therefore lead to unreliable results. These effects can be even much worse than in CG-S. In fact, the CG-S method has been derived as an improvement to IDR and the IDR method was not given further attention.

However, it is possible to rewrite the scheme, in which the residuals $Q_i(A)P_i(A)r_0$ are generated, to an apparently rather stable and more efficient one. Because of its similarity to CG-S, its favourable stability properties, and its relation with Bi-CG, we have named the method Bi-CGSTAB.

We will now derive the (unpreconditioned) Bi-CGSTAB Algorithm. This will be done in a similar way as followed in [8] for the derivation of CG-S.

The polynomial P_i and related polynomials are implicitly defined by the Bi-CG scheme.

UNPRECONDITIONED BI-CG ALGORITHM.

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\hat{r}_0$ is an arbitrary vector, such that
    $(\hat{r}_0, r_0) \neq 0$, e.g., $\hat{r}_0 = r_0$;
$\rho_0 = 1$;
$\hat{p}_0 = p_0 = 0$;
for $i = 1, 2, 3, \cdots$,
    $\rho_i = (\hat{r}_{i-1}, r_{i-1})$; $\beta_i = (\rho_i / \rho_{i-1})$;
    $p_i = r_{i-1} + \beta_i p_{i-1}$;
    $\hat{p}_i = \hat{r}_{i-1} + \beta_i \hat{p}_{i-1}$;
    $v_i = Ap_i$;
    $\alpha_i = \rho_i / (\hat{p}_i, v_i)$;
    $x_i = x_{i-1} + \alpha_i p_i$;
    if x_i is accurate enough then quit;
    $r_i = r_{i-1} - \alpha_i v_i$;
    $\hat{r}_i = \hat{r}_{i-1} - \alpha_i A^T \hat{p}_i$;
end

From this scheme it is straightforward to show that $r_i = P_i(A)r_0$ and $p_{i+1} = T_i(A)r_0$, in which $P_i(A)$ and $T_i(A)$ are ith degree polynomials in A. The Bi-CG Algorithm then defines the relations between these polynomials:

$$T_i(A)r_0 = (P_i(A) + \beta_{i+1} T_{i-1}(A))r_0,$$

and

$$P_i(A)r_0 = (P_{i-1}(A) - \alpha_i A T_{i-1}(A))r_0.$$

In the Bi-CGSTAB Algorithm we wish to have recurrence relations for

$$r_i = Q_i(A)P_i(A)r_0.$$

With Q_i as in (1) and the Bi-CG relation for the factor P_i and T_i, it then follows that

$$\begin{aligned} Q_i(A)P_i(A)r_0 &= (1-\omega_i A)Q_{i-1}(A)(P_{i-1}(A) - \alpha_i A T_{i-1}(A))r_0 \\ &= \{Q_{i-1}(A)P_{i-1}(A) - \alpha_i A Q_{i-1}(A)T_{i-1}(A)\}r_0 \\ &\quad -\omega_i A\{(Q_{i-1}(A)P_{i-1}(A) - \alpha_i A Q_{i-1}(A)T_{i-1}(A))\}r_0. \end{aligned}$$

Clearly, we also need a relation for the product $Q_i(A)T_i(A)r_0$. This can also be obtained from the Bi-CG relations:

$$\begin{aligned} Q_i(A)T_i(A)r_0 &= Q_i(A)(P_i(A) + \beta_{i+1}T_{i-1}(A))r_0 \\ &= Q_i(A)P_i(A)r_0 + \beta_{i+1}(1-\omega_i A)Q_{i-1}(A)T_{i-1}(A)r_0 \\ &= Q_i(A)P_i(A)r_0 + \beta_{i+1}Q_{i-1}(A)T_{i-1}(A)r_0 \\ &\quad -\beta_{i+1}\omega_i A Q_{i-1}(A)T_{i-1}(A)r_0. \end{aligned}$$

Finally we have to recover the Bi-CG constants ρ_i, β_i, and α_i by innerproducts in terms of the new vectors that we now have generated. For example, β_i can be computed as follows. First we compute

$$\tilde{\rho}_{i+1} = (\hat{r}_0, Q_i(A)P_i(A)r_0) = (Q_i(A^T)\hat{r}_0, P_i(A)r_0).$$

By construction, $P_i(A)r_0$ is orthogonal with respect to all vectors $U_{i-1}(A^T)\hat{r}_0$, where U_{i-1} is an arbitrary polynomial of degree $i-1$ at most. This means that we have to consider only the highest order term of $Q_i(A^T)$ when computing $\tilde{\rho}_{i+1}$. This term is given by $(-1)^i\omega_1\omega_2\cdots\omega_i(A^T)^i$. We actually wish to compute

$$\rho_{i+1} = (P_i(A^T)\hat{r}_0, P_i(A)r_0),$$

and since the highest order term of $P_i(A^T)$ is given by $(-1)^i\alpha_1\alpha_2\cdots\alpha_i(A^T)^i$, it follows that

$$\beta_i = (\tilde{\rho}_i/\tilde{\rho}_{i-1})(\alpha_{i-1}/\omega_{i-1}).$$

The other constants can be derived similarly.

Note that in our discussion we have focused on the recurrence relations for the vectors r_i and p_i, while in fact our main goal is to determine x_i. As in all CG-type methods, x_i itself is not required for continuing the iteration, but it can easily be determined as a "sideproduct" by realizing that an update of the form $r_i = r_{i-1} - \gamma Ay$ corresponds to an update $x_i = x_{i-1} + \gamma y$ for the current approximated solution.

By writing r_i for $Q_i(A)P_i(A)r_0$ and p_i for $Q_{i-1}(A)T_{i-1}(A)r_0$, we obtain the following scheme for Bi-CGSTAB (we trust that, with the foregoing observations, the reader will now be able to verify the relations in Bi-CGSTAB). In this scheme we have computed the ω_i so that $r_i = Q_i(A)P_i(A)r_0$ is minimized in 2-norm as a function of ω_i.

UNPRECONDITIONED BI-CGSTAB ALGORITHM.

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\hat{r}_0$ is an arbitrary vector, such that
$(\hat{r}_0, r_0) \neq 0$, e.g., $\hat{r}_0 = r_0$;
$\rho_0 = \alpha = \omega_0 = 1$;

$v_0 = p_0 = 0;$
for $i = 1, 2, 3, \cdots,$
$\rho_i = (\hat{r}_0, r_{i-1}); \beta = (\rho_i/\rho_{i-1})(\alpha/\omega_{i-1});$
$p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1}v_{i-1});$
$v_i = Ap_i;$
$\alpha = \rho_i/(\hat{r}_0, v_i);$
$s = r_{i-1} - \alpha v_i;$
$t = As;$
$\omega_i = (t, s)/(t, t);$
$x_i = x_{i-1} + \alpha p_i + \omega_i s;$
if x_i is accurate enough then quit;
$r_i = s - \omega_i t;$
end

In order to restrict on memory traffic, we have carried out both updates to the current solution x in one single step, while the updates to the residual r had to be done separately ($s = r_{i-1} - \alpha v_i$ and $r_i = s - \omega_i t$). So s represents the residual after a "Bi-CG step" and one might terminate the iteration as soon as $\|s\|$ is small enough, but in that case, before stopping the algorithm, the current solution has to be updated appropriately as $x_i = x_{i-1} + \alpha p_i$ in order to be compatible with the current residual s (and the computation of t, ω_i, as well as the second update $\omega_i s$ should be skipped).

From the orthogonality property $(P_i(A)r_0, Q_j(A^T)\hat{r}_0) = 0$, for $j < i$, it follows that Bi-CGSTAB is also a finite method, i.e., in exact arithmetic it will terminate after $m \leq n$ iteration steps. In this case we get $s = 0$ at iteration step m and ω_m is then not defined. This represents a lucky breakdown of the algorithm and the process should be terminated as indicated in the previous paragraph.

In the above form Bi-CGSTAB requires, for the solution of an N by N system $Ax = b$, evaluation of two matrix vector products with A, $12N$ flops for vector updates, and four innerproducts. This has to be compared with (unpreconditioned) CG-S which requires also two matrix vector products with A, and $13N$ flops, but only two innerproducts. In practical situations, however, the two additional innerproducts lead to only a small increase in computational work per iteration step and this is readily undone by almost any reduction in the number of iteration steps (especially on vector computers on which innerproducts are usually fast operations).

Except for memory locations for x, b, r, and A, we need for Bi-CGSTAB memory space for four additional N-vectors $\hat{r}_0, p, v$, and t (note that r may be overwritten by s). This is the same as for CG-S.

In §5 we will see, for some examples, the more smooth convergence properties of Bi-CGSTAB in comparison with CG-S. We will also see that, at least for our experiments, the new method is often more efficient than CG-S, in terms of the amount of computational work necessary to achieve a specified accuracy in the final result.

We have not discussed convergence problems that one can encounter with Bi-CG, and, hence, with CG-S and Bi-CGSTAB. These problems stem basically from the fact that for general matrices the bilinear form

$$[x, y] \equiv (P(A^T)x, P(A)y),$$

which is used to form the bi-orthogonality, does not define an innerproduct. In particular, it may occur that, by an unfavourable choice for $\hat{r}_0$, an iteration parameter ρ_i or $(\hat{r}_0, v_i)$ is zero (or very small), without convergence having taken place. In an

actual code, one should test for such situations and take appropriate measures, e.g., restart with a different $\hat{r}_0$ or switch to another method (for example, GMRES [7]).

Moreover, one should keep in mind that Bi-CGSTAB, in the form as presented, is one out of many possible variants, which are all equivalent in exact arithmetic but which may have different behaviour in finite precision arithmetic. One such variant, which is less straightforward than the presented one, is obtained by the following changes to the scheme:

- include $\rho_1 = (\hat{r}_0, r_0)$ in the initialization part of the scheme;
- skip the computation of ρ_i in the first line of the iteration part;
- add the computation of $\rho_{i+1} = -\omega_i(\hat{r}_0, t)$ immediately after the computation of ω_i.

We have seen, in some of our experiments, a markedly better convergence behaviour for this variant, but further research is necessary in order to determine the most satisfying variant.

4. Preconditioning. If we want to use preconditioning with a suitable preconditioning matrix K, i.e., $K \approx A$, then we write $K = K_1 K_2$ and we may apply any of the previously discussed iteration schemes to the explicitly preconditioned system

$$\tilde{A}\tilde{x} = \tilde{b}, \tag{2}$$

with $\tilde{A} = {K_1}^{-1} A {K_2}^{-1}, x = {K_2}^{-1}\tilde{x}$, and $\tilde{b} = {K_1}^{-1} b$. For example, for $K_1 = I$ we have preconditioning from the right (or postconditioning), for $K_2 = I$ we have preconditioning from the left, and for $K_1 = L, K_2 = U$, we have the well-known preconditioning from both sides.

Now we write the CG-S scheme (§2) for (2), and denote all the occurring vectors by $\tilde{}$, e.g., $\tilde{p}_i$. With the change of variables:

$$\tilde{p}_i \Rightarrow {K_1}^{-1} p_i, \tilde{q}_i \Rightarrow {K_1}^{-1} q_i, \tilde{v}_i \Rightarrow {K_1}^{-1} v_i,$$
$$\tilde{r}_i \Rightarrow {K_1}^{-1} r_i, \tilde{u}_i \Rightarrow {K_1}^{-1} u_i, \tilde{x}_i \Rightarrow K_2 x_i,$$
$$\hat{r}_0 \Rightarrow {K_1}^{T} \bar{r}_0,$$

this leads to the following scheme for *preconditioned CG-S.*

PRECONDITIONED CG-S ALGORITHM.

x_0 is an initial guess; $r_0 = b - Ax_0$;
$\bar{r}_0$ is an arbitrary vector, such that $(r_0, \bar{r}_0) \neq 0$,
  e.g., $\bar{r}_0 = r_0$;
$\rho_0 = 1; p_0 = q_0 = 0$;
for $i = 1, 2, 3, \cdots$,
  $\rho_i = (\bar{r}_0, r_{i-1})$;
  $\beta = \rho_i / \rho_{i-1}$;
  $u = r_{i-1} + \beta q_{i-1}$;
  $p_i = u + \beta(q_{i-1} + \beta p_{i-1})$;
  Solve y from $Ky = p_i$;
  $v = Ay$;
  $\alpha = \rho_i / (\bar{r}_0, v)$;
  $q_i = u - \alpha v$;
  Solve z from $Kz = u + q_i$
  $x_i = x_{i-1} + \alpha z$;

if x_i is accurate enough then quit;
$r_i = r_{i-1} - \alpha Az$;
end

Note that this scheme delivers the variables x_i and r_i corresponding to the original system $Ax = b$, i.e., $r_i = b - Ax_i$.

A remarkable observation is that K_1 and K_2 play no explicit role in the scheme, and that any of the forms of preconditioning (i.e., any of the choices for K_1 and K_2) *correspond only with a different choice for* $\hat{r}_0$ in the original, explicitly preconditioned scheme. Hence, if one of the forms of explicit preconditioning in the original scheme might lead to an advantage, then the same advantage can be obtained by the above scheme through an appropriate choice of $\bar{r}_0$. Or, in still other words, instead of studying the effect of the different forms of applying the preconditioner, one may as well study the effect of the choice for $\bar{r}_0$.

When we rewrite the Bi-CGSTAB scheme for equation (2), similarly as previously for CG-S, then with the change of variables:

$$\tilde{p}_i \Rightarrow {K_1}^{-1} p_i, \tilde{v}_i \Rightarrow {K_1}^{-1} v_i, \tilde{r}_i \Rightarrow {K_1}^{-1} r_i,$$
$$\tilde{s} \Rightarrow {K_1}^{-1} s, \tilde{t} \Rightarrow {K_1}^{-1} t, \tilde{x}_i \Rightarrow K_2 x_i,$$
$$\hat{r}_0 \Rightarrow {K_1}^{T} \bar{r}_0,$$

we obtain the following scheme for *preconditioned Bi-CGSTAB.*

PRECONDITIONED BI-CGSTAB ALGORITHM.
x_0 is an initial guess; $r_0 = b - Ax_0$;
$\bar{r}_0$ is an arbitrary vector, such that
$(\bar{r}_0, r_0) \neq 0$, e.g., $\bar{r}_0 = r_0$;
$\rho_0 = \alpha = \omega_0 = 1$;
$v_0 = p_0 = 0$;
for $i = 1, 2, 3, \cdots$,
$\rho_i = (\bar{r}_0, r_{i-1})$; $\beta = (\rho_i/\rho_{i-1})(\alpha/\omega_{i-1})$;
$p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$;
Solve y from $Ky = p_i$;
$v_i = Ay$;
$\alpha = \rho_i/(\bar{r}_0, v_i)$;
$s = r_{i-1} - \alpha v_i$;
Solve z from $Kz = s$;
$t = Az$;
$\omega_i = ({K_1}^{-1} t, {K_1}^{-1} s)/({K_1}^{-1} t, {K_1}^{-1} t)$;
$x_i = x_{i-1} + \alpha y + \omega_i z$;
if x_i is accurate enough then quit;
$r_i = s - \omega_i t$;
end

This scheme too delivers the variables x_i and r_i corresponding to the original system $Ax = b$.

For preconditioned CG-S we have seen that any of the forms of preconditioning can be regarded as a suitable choice for $\bar{r}_0$. However, for Bi-CGSTAB there is an explicit difference between the different forms, which cannot be attributed to a suitable choice for $\bar{r}_0$, because of the expression for ω_i. This expression does not seem to make

the preconditioned scheme very attractive, but we might as well compute another value for ω_i:

$$\omega_i = (t, s)/(t, t). \tag{3}$$

When computing ω_i as in (3), we are minimizing the current residual for the original system rather than the preconditioned one. In this case we have in fact the same effect as by explicit postconditioning in the original scheme (though for a different $\bar{r}_0$), and hence we lose potential (near-)symmetry of the preconditioned operator. In such a case we might prefer to apply the Unpreconditioned Bi-CGSTAB Algorithm, as in §3, to the explicitly preconditioned system $K_1^{-1}AK_2^{-1}\tilde{x} = K_1^{-1}b$. However, this has the obvious disadvantage that we have to construct a (near-)symmetric splitting of the preconditioner K.

We will refer to the scheme with expression (3) for the ω_i as Bi-CGSTAB-P.

Hence, if one compares Bi-CGSTAB-P with preconditioned CG-S then one can view this as a comparison between the two explicitly postconditioned schemes in §§2 and 3 (each with a different $\hat{r}_0$).

5. Numerical results. In our experiments we consider four different, but representative situations. These experiments have been carried out in double precision floating point arithmetic (about 15 decimal places) on a CONVEX C-240 computer. In order to avoid all confusion about the definition of $\bar{r}_0$ (see §4), all experiments have been carried out with CG-S and Bi-CGSTAB applied to the explicitly preconditioned system $L^{-1}AU^{-1}\tilde{x} = L^{-1}b$. However, based upon our experiments so far, our conclusions would be about the same when we compare the preconditioned CG-S scheme with Bi-CGSTAB-P (§4).

In all cases the iteration was started with $x_0 = 0$.

5.1. Example 1. The first situation is one in which Bi-CG converges quite smoothly and in which it was, apparently, a good idea to square the Bi-CG polynomial. This happens sometimes in the early phases of the iteration process or in situations where the eigenvalue distribution is quite uniform.

As an example we show in Fig. 1 the convergence behaviour for the discretized Poisson equation in two dimensions over a 150×150 grid (leading to a symmetric positive definite system), preconditioned by Modified Incomplete Choleski decomposition [4]. In this case CG-S converged about twice as fast as Bi-CG, as we might expect from heuristic arguments.

We see that in this case, though Bi-CGSTAB converges more smoothly, it does not improve the iteration process with respect to efficiency. In similar experiments Bi-CGSTAB requires roughly the same number of iteration steps as CG-S, sometimes slightly more and sometimes slightly less.

5.2. Example 2. Our second example is a preconditioned symmetric positive definite system for which Bi-CG (=Conjugate Gradients in this case) loses orthogonality among the residuals in a very early phase. For a discussion on this effect and its consequences, see [9]. In such a case one would expect some strong effects when squaring the Bi-CG polynomial, as is done in CG-S.

The linear system comes from a 5-point finite difference discretization of the partial differential equation

$$-(Du_x)_x - (Du_y)_y = 1$$

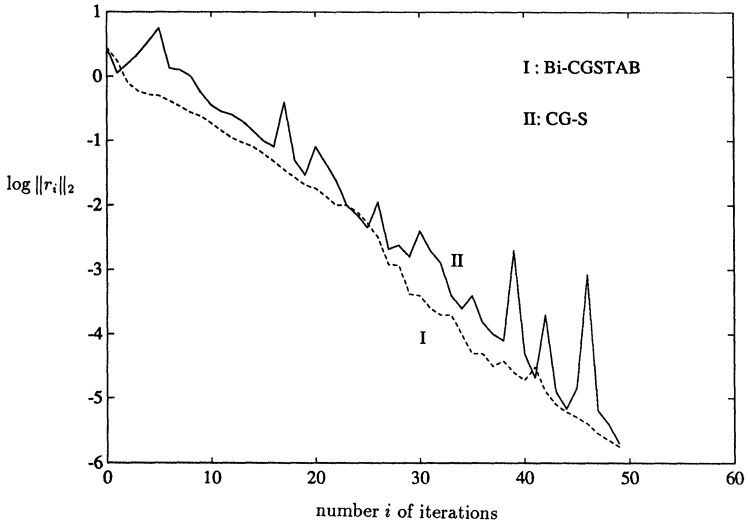

FIG. 1. *CG-S and Bi-CGSTAB for an ideal CG-S situation.*

over the unit square, with Dirichlet boundary conditions along $y = 0$ and Neumann conditions along the other parts of the boundary. Meshsizes have been chosen so that the resulting linear system has $150 \times 150 = 22,500$ unknowns.

The function D is defined as

$$D = 1000 \quad \text{for } 0.1 \leq x, y \leq 0.9, \quad \text{and } D = 1 \quad \text{elsewhere.}$$

Modified Incomplete Choleski Decomposition [4] was used as preconditioner.

In Fig. 2 we see the loss of orthogonality reflected in the convergence behaviour of Bi-CG by small irregularities and by the fact that superlinear convergence does not take place. CG-S converges faster eventually, but we see that the local effects in this method are much more violent.

Bi-CGSTAB seems to have about the same "asymptotical" speed of convergence as CG-S (if we discard the peaks in the CG-S curve), but its convergence behaviour is definitely smoother. In situations like these, we often see that the updated residual in Bi-CGSTAB has more significance than the one in CG-S. See also the discussion of example 4 (§5.4) for this effect.

5.3. Example 3. In our third example the nonsymmetric linear system comes from discretization of the partial differential equation

$$-u_{xx} - u_{yy} + ((au)_x + au_x)/2 = 1$$

over the unit square, with $a = 20\exp(3.5(x^2 + y^2))$. Along the boundaries we have Dirichlet conditions (this equation was taken from [11]). The equation was discretized over a rectangular grid, with central differences for the first order terms, with stepsize $1/201$ in each direction, leading to a system with $40,000$ unknowns. The linear system was preconditioned by a standard incomplete LU factorization [5].

In Fig. 3 we have displayed the norms of the iterated vectors r_i for both CG-S and Bi-CGSTAB, and again we note the much smoother convergence behaviour of

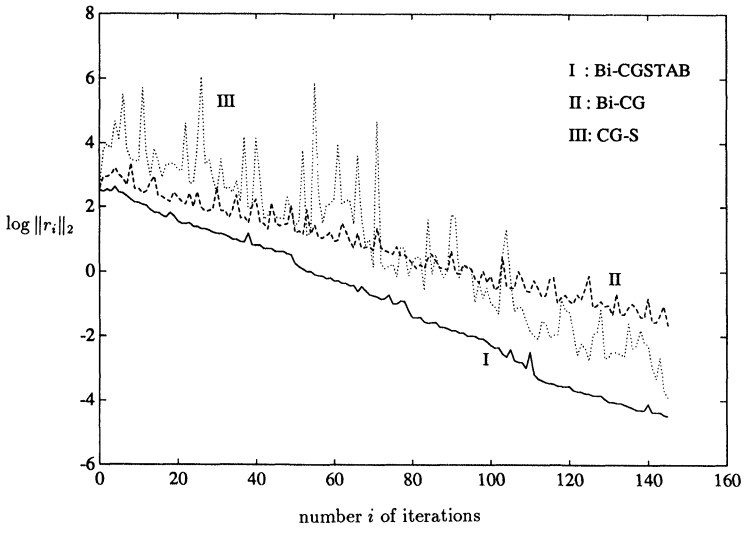

FIG. 2. *Less smooth convergence behaviour of Bi-CG and CG-S.*

Bi-CGSTAB. Also note that some residuals in CG-S may be quite large as compared with the starting residual. Since the updates to the corresponding iterate vectors x_i must have been large too, we may expect cancellation effects in the iterated vector. Indeed, after 200 CG-S iterations, the explicitly computed residual (using the delivered iteration vector x_i) was larger (in norm) than the updated residual vector r_i by three orders of magnitude. This implies that the approximated solution had an error which was about 1000 times as large as we might have thought. For Bi-CGSTAB both residuals were virtually equal to each other.

Finally note that in this case Bi-CGSTAB, apart from being more stable, also converges faster than CG-S. This was true for most of our test cases. We have also tested Bi-CG for this problem, but this method showed hardly any progress within 450 iteration steps.

5.4. Example 4. The fourth example is typical for a situation in which Bi-CGSTAB is much more efficient than CG-S. The nonsymmetric linear system comes from discretization of the partial differential equation

$$-(Au_x)_x - (Au_y)_y + B(x,y)u_x = F$$

over the unit square, with $B(x,y) = 2\exp(2(x^2+y^2))$. Along the boundaries we have Dirichlet conditions: $u = 1$, for $y = 0, x = 0$ and $x = 1$, and $u = 0$ for $y = 1$.

The function A is defined as shown in Fig. 4; $F = 0$ everywhere, except for the small subsquare in the centre where $F = 100$.

The equation was discretized over a rectangular grid, with central differences for the first order term, with stepsize $1/128$ in each direction, leading to a system with 127^2 unknowns. The linear system was preconditioned by an incomplete LU factorization [5].

Figure 5 shows the norms of the iteration vectors r_i and we observe similar effects as in example 3. At the 151st iteration Bi-CGSTAB was terminated with $\|r_i\|_2 \approx 10^{-8}$; at that point the true residual $\|b - Ax_i\|_2$ is virtually the same.

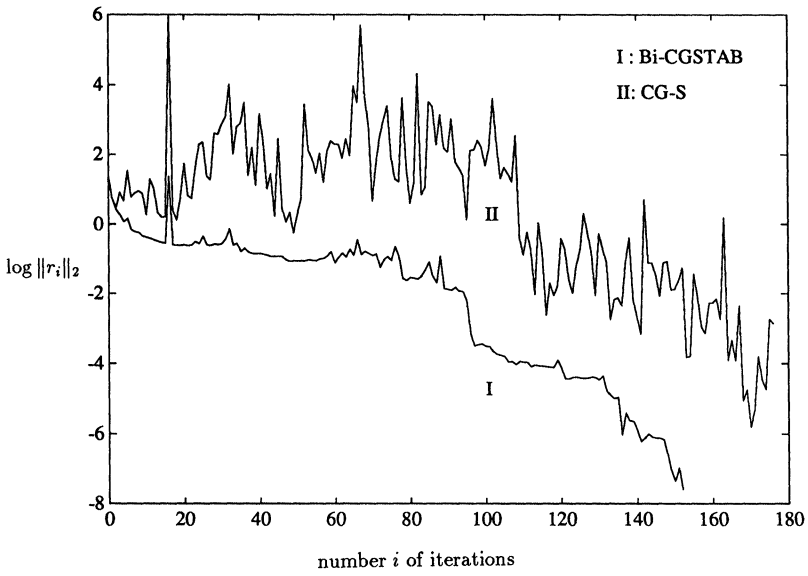

FIG. 3. *CG-S and Bi-CGSTAB for a nonsymmetric problem (example 3).*

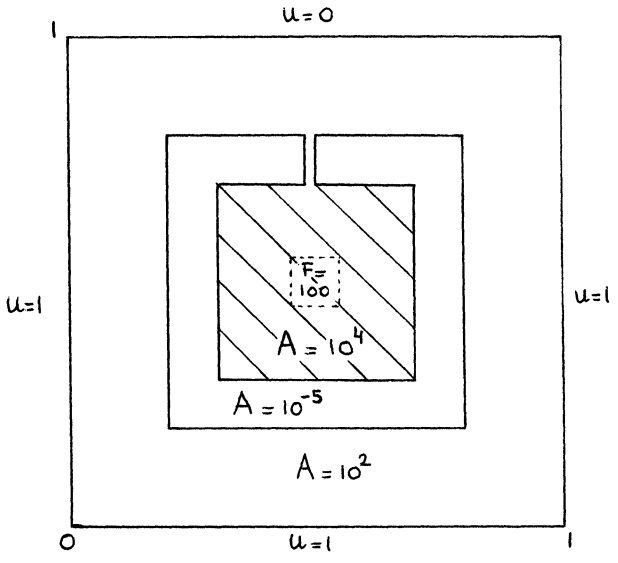

FIG. 4. *The coefficients for example 4.*

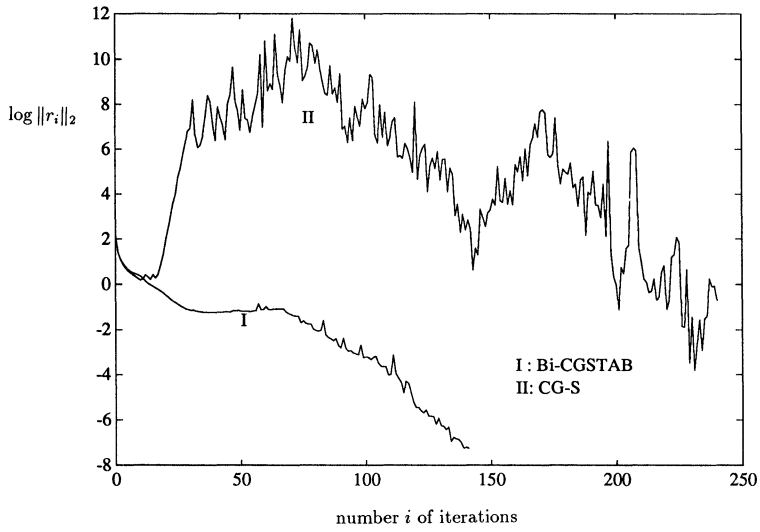

FIG. 5. *CG-S and Bi-CGSTAB for a nonsymmetric problem (example 4).*

It takes CG-S 382 iteration steps to obtain $\|r_i\|_2 \approx 10^{-8}$, but at that point the true residual $\|b - Ax_i\|_2$ is only about 10^{-4}. This means that in this case we have to iterate even further with CG-S, possibly after restarting the process, in order to get a result similar to that with Bi-CGSTAB.

For CG-S we have also checked what happens if we replace the *updated* vector r_i in the process by $r_i = b - Ax_i$ (see §2). It then turns out that both CG-S processes produce about the same residual vectors up to the 70th iteration step. But from then on the *updated* r_i process and the *true residual* process produce r_i vectors that differ more and more. After a while the *true residual* process delivers even much less accurate x_i vectors than Bi-CGSTAB. For example, at the 382nd iteration step the norm of the true residual $b - Ax_i$ in the *updated* r_i process is, as has been mentioned before, about 10^{-4}, whereas $\|b - Ax_i\|_2$ in the *true residual* process is about 10^2 for $i = 382$.

For this problem Bi-CG required 282 iteration steps to get the residual in norm below 10^{-8}. This example represents one of those rare examples in which CG-S does much worse than Bi-CG.

6. Conclusions. From many experiments we have learned that Bi-CGSTAB (and Bi-CGSTAB-P) is an attractive alternative for CG-S. Its convergence behaviour is much smoother so that it often produces much more accurate residual vectors (and, hence, more accurate solutions), and in most cases it converges considerably faster than CG-S.

The method has recently been tested for problems coming from semi-conductor device simulation and oil reservoir simulation and also in these circumstances our earlier findings have been confirmed (for many of these problems CG-S was the method of choice so far).

Therefore, we conclude that Bi-CGSTAB (with preconditioning, of course) is a very competitive method for solving relevant classes of nonsymmetric linear systems.

Acknowledgements. Peter Sonneveld (TU-Delft) suggested that I reconsider his old method IDR again, which turned out to be the key for Bi-CGSTAB, and I learned a lot from discussions with him.

I wish to thank Wolfgang Fichtner (ETH-Zürich), Koos Meijerink (Shell-Rijswijk), Marjan Driessen (Philips-Eindhoven), and Shun Doi (NEC-Kawasaki) for testing Bi-CGSTAB extensively on industrial problems and for reporting their results to me.

The referees have helped me to improve the presentation of this paper.

I further wish to express my thanks to Philips-Eindhoven for kindly permitting me access to their IBM 3090 system. Most of the numerical experiments, leading to understanding CG-S and to learning about the advantages of Bi-CGSTAB, have been carried out on that computer.

REFERENCES

[1] R. E. Bank, W. M. Coughran, M. A. Driscoll, W. Fichtner, and R. K. Smith, *Iterative methods in semiconductor device simulation*, Comput. Phys. Comm., 53 (1989), pp. 201–212.

[2] G. Brussino and V. Sonnad, *A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 801–815.

[3] R. Fletcher, *Conjugate gradient methods for indefinite systems*, Lecture Notes in Math., 506 (1976), pp. 73–89.

[4] I. Gustafsson, *A class of 1st order factorization methods*, BIT, 18 (1978), pp. 142–156.

[5] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[6] G. Radicati di Brozolo and Y. Robert, *Parallel conjugate gradient-like algorithms for solving sparse non-symmetric systems on a vector multiprocessor*, Parallel Comput., 11 (1989), pp. 223–239.

[7] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[8] P. Sonneveld, *CGS: a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[9] H. A. van der Vorst, *The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors*, Lecture Notes in Math., 1457 (1990), pp. 126–136.

[10] P. Wesseling and P. Sonneveld, *Numerical experiments with a multiple grid and a preconditioned Lanczos type method*, in Approximation Methods for Navier-Stokes Problems, Lecture Notes in Mathematics, R. Rautmann, ed., Springer-Verlag, Berlin, 1980.

[11] O. Widlund, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.