UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

QMRCGSTAB: A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems

T. F. Chan
E. Gallopoulos
V. Simoncini
T. Szeto
C. H. Tong

June 1992

CAM Report 92-26

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

A QUASI-MINIMAL RESIDUAL VARIANT OF THE BI-CGSTAB ALGORITHM FOR NONSYMMETRIC SYSTEMS

T. F. CHAN*, E. GALLOPOULOS†, V. SIMONCINI‡, T. SZETO§ AND C. H. TONG¶

Abstract. Motivated by a recent method of Freund [3], who introduced a quasi-minimal residual (QMR) version of the CGS algorithm, we propose a QMR variant of the Bi-CGSTAB algorithm of Van der Vorst, which we call QMRCGSTAB for solving nonsymmetric linear systems. The motivation for both QMR variants is to obtain smoother convergence behavior of the underlying method. We illustrate this by numerical experiments, which also show that for problems on which Bi-CGSTAB performs better than CGS, the same advantage carries over to QMRCGSTAB.

AMS(MOS) subject classifications. 65F10,65Y20

1. Introduction. In this note we propose a variation of the Bi-CGSTAB algorithm of Van der Vorst [15] for solving the linear system

$$Ax = b, \tag{1}$$

where A is a nonsymmetric sparse matrix of order n.

Various attempts have been made in the last twenty years to extend the highly successful conjugate gradient (CG) algorithm to the nonsymmetric case. One of the most natural extension is the Bi-Conjugate Gradient algorithm (BCG) [2] [9]. Although BCG is still quite competitive today, it also has several well-known drawbacks. Among these are (1) the need for matrix-vector multiplications with A^T (which can be inconvenient as well as doubling the number of matrix-vector multiplications compared to CG for each increase in the degree of the underlying Krylov subspace), (2) the possibility of breakdowns and (3) erratic convergence behavior.

Many recently proposed methods can be viewed as improvements over some of these drawbacks of BCG. The most notable of these is the ingenious conjugate gradients-squared method (CGS) proposed by Sonneveld [14], which cures the first drawback mentioned above by computing the square of the BCG polynomial without requiring A^T. Hence when BCG converges, CGS is an attractive, faster converging alternative. However, this relation between the residual polynomials also causes CGS to behave even more erratically than BCG, particularly in near-breakdown situations for BCG [8][15]. These observations led van der Vorst [15] to introduce Bi-CGSTAB, a more smoothly converging variant of CGS. The main idea is to form

* Dept. of Math., Univ. of Calif. Los Angeles, Los Angeles, CA 90024. E-mail: chan@math.ucla.edu. Partially supported by the Dept. of Energy grant DE-FG03-87ER25037, the Office of Naval Research grant N00014-90-J-1695, the National Science Foundation grant ASC90-03002, and the Army Research Office grant DAAL03-91-G-150.

† Department of Computer Science and Center for Supercomputing Research and Development, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois 61801. E-mail stratis@csrd.uiuc.edu. This research was supported by the the Department of Energy grant DOE DE-FG02-85ER25001 and the National Science Foundation under grants NSF CCR-9120105 and CCR-9024554. Additional support was provided by the State of Illinois Department of Commerce and Community Affairs, State Technology Challenge Fund under grant No. SCCA 91-108.

‡ Center for Supercomputing Research and Development, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois 61801. E-mail simoncin@csrd.uiuc.edu. Supported by fellowship #2043597 from the Consiglio Nazionale delle Ricerche, Italy.

§ Dept. of Math., Univ. of Calif. Los Angeles, Los Angeles, CA 90024. E-mail: szeto@math.ucla.edu. Supported by same grants as the first author.

¶ Cnt. Comp. Eng., Sandia Nat. Lab., Livermore, CA 94551. E-mail: chtong@snll-arpagw.llnl.gov.

a product of the BCG polynomial with another, locally defined polynomial. The Bi-CGSTAB method was further refined by Gutknecht [7] to handle complex matrices and also lead to better convergence for the case of complex eigenvalues. Nevertheless, although the Bi-CGSTAB algorithms were found to perform very well compared to CGS in many situations, there are cases where convergence is still quite erratic (see, for example, Section 5 and [11]).

In a recent paper [3], Freund proposed a new version of CGS, called TFQMR[1], which "quasi-minimizes" ([5]) the residual in the space spanned by the vectors generated by the CGS iteration. Numerical experiments show that QMRCGS in most cases retains the good convergence features of CGS while correcting its erratic behavior. The transpose free nature of QMRCGS, its low computational cost and its smooth convergence behavior make QMRCGS an attractive alternative to CGS. On the other hand, since the square of the residual polynomial for BCG is still in the space being quasi-minimized, QMRCGS inherits the same asymptotic behavior of CGS. In particular, QMRCGS breaks down whenever CGS does. Moreover, it is well-known that the CGS residual polynomial can be quite polluted by round-off error [17]; one possible remedy would be to combine QMRCGS with a look-ahead Lanczos technique as was done for the original QMR method [4].

In this paper, we take an alternative approach by deriving quasi-minimum residual extensions to Bi-CGSTAB. We call the basic method QMRCGSTAB and illustrate its smoothed convergence by means of numerical experiments. We note that it may appear redundant to combine the local minimization in Bi-CGSTAB with a global quasi-minimization. However, our view is that the local minimization is secondary in nature and is only used as a way of generating residual polynomials in the appropriate Krylov subspace over which the residual is being quasi-minimized. In fact, this view allows us some flexibility in modifying the local minimization step in Bi-CGSTAB which leads to other quasi-minimal residual variants.

2. Review of Bi-CGSTAB. In this section we review the Bi-CGSTAB algorithm on which the method proposed in this paper is based.

All of the algorithms discussed in this paper are derived from the BCG algorithm [2], which is a natural generalization of the classical CG algorithm for the solution of both $Ax = b$ and $A^T\tilde{x} = \tilde{b}$. Given x_0 and $\tilde{x}_0$ such that $(\tilde{r}_0, r_0) \neq 0$, the algorithm can be written as follows:

Algorithm BCG$(A, b, \tilde{b}, x_0, \tilde{x}_0, \epsilon)$
$r_0 = b - Ax_0$, $\tilde{r}_0 = \tilde{b} - A^T\tilde{x}_0$
$p_0 = r_0$, $\tilde{p}_0 = \tilde{r}_0$
$\rho_0 = (\tilde{r}_0, r_0)$
$i = 0$
while ($\|r_i\| > \epsilon$ and $\|\tilde{r}_i\| > \epsilon$)
$i = i + 1$
$\sigma_i = (\tilde{p}_i, Ap_i)$
$\alpha_i = \rho_i/\sigma_i$
$r_{i+1} = r_i - \alpha_i Ap_i$, $\tilde{r}_{i+1} = \tilde{r}_i - \alpha_i A^T\tilde{p}_i$
$x_{i+1} = x_i + \alpha_i p_i$, $\tilde{x}_{i+1} = \tilde{x}_i + \alpha_i\tilde{p}_i$
$\rho_{i+1} = (\tilde{r}_{i+1}, r_{i+1})$
$\beta_{i+1} = \rho_{i+1}/\rho_i$
$p_{i+1} = r_{i+1} + \beta_{i+1}p_i$; $\tilde{p}_{i+1} = \tilde{r}_{i+1} + \beta_{i+1}\tilde{p}_i$
end

[1] In this paper we shall refer to it as QMRCGS in order to more clearly illustrate its relationship to and distinguish it from our proposed methods. Both methods are transpose-free quasi-minimal residual methods.

The basic idea in CGS is the following: Let $r_i = \varphi_i(A)r_0$ be the residual at the i^{th} iteration of BCG. The residual in CGS is expressed by $\varphi_i^2(A)r_0$, so that A^T is not involved. The coefficient $\rho_{i+1} = (\varphi_i(A^T)\tilde{r}_0, \varphi_i(A)r_0)$ is calculated using:

$$\rho_{i+1} = (\tilde{r}_0, \varphi_i^2(A)r_0). \tag{2}$$

Instead of squaring the polynomial φ_i, Algorithm Bi-CGSTAB applies a different polynomial $\psi_i(A)$ to the residual r_i using the formula

$$\psi_0(A) = I, \quad \psi_{i+1}(A) = (I - \omega_{i+1}A)\psi_i(A),$$

where the parameters ω_i are chosen in a suitable way. Hence, during the next iteration, the residual is given by $r_{i+1} = \psi_{i+1}(A)\varphi_{i+1}(A)r_0$. Van der Vorst [15] shows that r_i can also be updated without requiring A^T. From the definition of $\varphi_{i+1}(A)$, it can be shown that the residual r_i can be updated in two successive steps:

$$\psi_i(A)\varphi_i(A)r_0 \rightarrow \psi_i(A)\varphi_{i+1}(A)r_0 \rightarrow \psi_{i+1}(A)\varphi_{i+1}(A)r_0.$$

The coefficient ρ_{i+1} is computed by $\rho_{i+1} = (\tilde{r}_0, \psi_{i+1}(A)\varphi_{i+1}(A)r_0)$. It remains to explain how to compute parameters $\alpha_i, \beta_i, \omega_i$. The first two are related to the original BCG algorithm and are given in [15] as

$$\alpha_i = \frac{\rho_i}{(\tilde{r}_0, A\psi_i(A)\varphi_i(A)r_0)}, \qquad \beta_i = \frac{\rho_i}{\rho_{i-1}}\frac{\alpha_{i-1}}{\omega_{i-1}}.$$

The factor $\frac{\alpha_{i-1}}{\omega_{i-1}}$ is added in order for the polynomial $\psi_{i+1}(A)$ to have the same leading coefficient as $\varphi_{i+1}^2(A)$. The parameter ω_{i+1} is calculated by the steepest descent rule to minimize the quantity $\|(I - \omega_{i+1}A)\psi_i(A)\varphi_{i+1}(A)r_0\|$. This gives

$$\omega_{i+1} = \frac{(As_{i+1}, s_{i+1})}{(As_{i+1}, As_{i+1})},$$

where $s_{i+1} = \psi_i(A)\varphi_{i+1}(A)r_0$.

Using these parameters the Bi-CGSTAB algorithm is as shown below:

Algorithm Bi-CGSTAB(A, b, x_0, ϵ)
$r_0 = b - Ax_0$
choose $\tilde{r}_0$ such that $(\tilde{r}_0, r_0) \neq 0$
$\rho_0 = \alpha_0 = \omega_0 = 1$
$v_0 = p_0 = 0$
$i = 0$
while $(\|r_i\| > \epsilon)$
$i = i + 1$
$\rho_i = (\tilde{r}_0, r_{i-1}), \beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$
$p_i = r_{i-1} + \beta_i(p_{i-1} - \omega_{i-1}v_{i-1})$
$v_i = Ap_i$
$\alpha_i = \rho_i/(\tilde{r}_0, v_i)$
$s_i = r_{i-1} - \alpha_i v_i$
$t_i = As_i$

$\omega_i = (t_i, s_i)/(t_i, t_i)$
$x_i = x_{i-1} + \alpha_i p_i + \omega_i s_i$
$r_i = s_i - \omega_i t_i$
end

As explained in [15], the first part of the loop corresponds to the BCG step for the matrix A, and s_i represents an intermediate residual which could be tested for convergence.

Since φ_i and ψ_j are orthogonal for $j < i$, that is $(\psi_j(A^T)\tilde{r}_0, \varphi_i(A)r_0) = 0$, for $j < i$, it follows that in exact arithmetic Bi-CGSTAB terminates with the true solution after $m \leq n$ steps [15]. Since both CGS and Bi-CGSTAB have the BCG polynomial built-in, they break down whenever BCG does.

3. The QMRCGSTAB Algorithm. The algorithm proposed in this paper is inspired by QMRCGS in that the three term recurrence $x_i = x_{i-1} + \alpha_i p_i + \omega_i s_i$ of Bi-CGSTAB is transformed into a quasi-minimization problem. In other words, we use Bi-CGSTAB to generate the vectors p_i and s_i and to quasi-minimize the residual over their span.

Our derivation follows closely that of QMRCGS. During each step of Bi-CGSTAB two vector relations hold:

$$s_i = r_{i-1} - \alpha_i A p_i, \qquad r_i = s_i - \omega_i A s_i. \tag{3}$$

Let $Y_k = \{y_1, y_2, ..., y_k\}$, where $y_{2l-1} = p_l$ for $l = 1, \ldots, [(k+1)/2]$ and $y_{2l} = s_l$ for $l = 1, \ldots, [k/2]$, ($[k/2]$ is the integer part of $k/2$). In the same way, let $W_{k+1} = \{w_0, w_1, \ldots, w_k\}$ with $w_{2l} = r_l$ for $l = 0, \ldots, [k/2]$ and $w_{2l-1} = s_l$ for $l = 1, \ldots, [(k+1)/2]$. We also define $\{\delta_1, \delta_2, \ldots, \delta_k\}$, as $\delta_{2l} = \omega_l$ for $l = 1, \ldots, [(k+1)/2]$ and $\delta_{2l-1} = \alpha_l$ for $l = 1, \ldots, [(k+1)/2]$. In this case, for each column of W_{k+1} and Y_k, Eq. (3) may be written as

$$A y_j = (w_{j-1} - w_j)\delta_j^{-1}, \; j = 1, \ldots, k, \tag{4}$$

or, using matrix notation,

$$A Y_k = W_{k+1} E_{k+1},$$

where E_{k+1} is a $(k+1) \times k$ bidiagonal matrix with diagonal elements δ_j^{-1} and lower diagonal elements $-\delta_j^{-1}$.

It can easily be checked that the degree of the polynomials corresponding to the vectors r_i, s_i and p_i are $2i, 2i-1$ and $2i-2$ respectively. Therefore, $\text{span}(Y_k) = \text{span}(W_k) = K_{k-1}$, where K_k is the Krylov subspace of degree k generated by r_0. The main idea in QMRCGSTAB is to look for an approximation to the solution of Eq. (1), using the Krylov subspace K_{k-1}, in the form

$$x_k = x_0 + Y_k g_k, \quad \text{with} \quad g_k \in \mathbb{C}^k.$$

Hence, we may write the residual $r_k = b - Ax_k$ as

$$r_k = r_0 - A Y_k g_k = r_0 - W_{k+1} E_{k+1} g_k.$$

Using the fact that the first vector of W_{k+1} is indeed r_0, it follows that

$$r_k = W_{k+1}(e_1 - E_{k+1} g_k),$$

where e_1 is the first vector of the canonical basis. Since the columns of W_{k+1} are not normalized, it was suggested in [3] to use a $(k+1)\times(k+1)$ scaling matrix Σ_{k+1}=diag$(\sigma_1,..,\sigma_{k+1})$, with $\sigma_i = \|w_i\|$, in order to make the columns of W_{k+1} to be of unit norm. Then

$$r_k = W_{k+1}\Sigma_{k+1}^{-1}\Sigma_{k+1}(e_1 - E_{k+1}g_k) = W_{k+1}\Sigma_{k+1}^{-1}(\sigma_1 e_1 - H_{k+1}g_k) \tag{5}$$

with $H_{k+1} = \Sigma_{k+1}E_{k+1}$.

The quasi-minimal residual approach consists of the minimization of $\|\sigma_1 e_1 - H_{k+1}g\|$ for some $g \in \mathbb{R}^k$. In Section 4 we will introduce a variant of QMRCGSTAB which generates W_{k+1} with pairwise orthogonal columns.

The least squares minimization of $\|\sigma_1 e_1 - H_{k+1}g\|$ is solved using QR decomposition of H_{k+1}. This is done in an incremental manner by means of Givens rotations. Since H_{k+1} is lower bidiagonal, only the rotation of the previous step is needed. We refer to [3] for a detailed description of the QR decomposition procedure.

The pseudocode for the QMRCGSTAB algorithm is as follows, in which the Givens rotations used in the QR decomposition are written out explicitly:

Algorithm QMRCGSTAB(A, b, x_0, ϵ)
(1) Initialization
$r_0 = b - Ax_0$
choose $\tilde{r}_0$ such that $(\tilde{r}_0, r_0) \neq 0$
$p_0 = v_0 = d_0 = 0$
$\rho_0 = \alpha_0 = \omega_0 = 1; \tau = \|r_0\|, \theta_0 = 0, \eta_0 = 0$
(2) for $k = 1, 2, \cdots$ do
$\rho_k = (\tilde{r}_0, r_{k-1}); \beta_k = (\rho_k\alpha_{k-1})/(\rho_{k-1}\omega_{k-1})$
$p_k = r_{k-1} + \beta_k(p_{k-1} - \omega_{k-1}v_{k-1})$
$v_k = Ap_k$
$\alpha_k = \rho_k/(\tilde{r}_0, v_k)$
$s_k = r_k - \alpha_k v_k$
(2.1) First quasi-minimization and update iterate
$\tilde{\theta}_k = \|s_k\|/\tau; c = 1/\sqrt{1+\tilde{\theta}_k^2}; \tilde{\tau} = \tau\tilde{\theta}_k c$
$\tilde{\eta}_k = c^2\alpha_k$
$\tilde{d}_k = p_k + \frac{\theta_{k-1}^2\eta_{k-1}}{\alpha_k}d_{k-1}$
$\tilde{x}_k = x_{k-1} + \tilde{\eta}_k\tilde{d}_k$
(2.2) compute t_k, ω_k and update r_k
$t_k = As_k$
$\omega_k = (s_k, t_k)/(t_k, t_k)$
$r_k = s_k - \omega_k t_k$
(2.3) Second quasi-minimization and update iterate
$\theta_k = \|r_k\|/\tilde{\tau}; c = 1/\sqrt{1+\theta_k^2}; \tau = \tilde{\tau}\theta_k c$
$\eta_k = c^2\omega_k$
$d_k = s_k + \frac{\tilde{\theta}_k^2\tilde{\eta}_k}{\omega_k}\tilde{d}_k$
$x_k = \tilde{x}_k + \eta_k d_k$
If x_k is accurate enough, then quit
(3) end

To check the convergence the estimate $\|\hat{r}_k\| \leq \sqrt{k+1}|\tau|$ was used, where $\hat{r}_k$ denotes the QMRCGSTAB residual at step k [3].

Note that the cost per iteration is slightly higher than for Bi-CGSTAB, since two additional inner products are needed to compute the elements of Σ_{k+1}. A more detailed discussion on computational costs is given in Section 5.

4. Some Variants of QMRCGSTAB. The use of quasi-minimization in the "product algorithms" (such as CGS and Bi-CGSTAB) introduces some flexibility. For example, the underlying product algorithm need not be constrained to generate a residual polynomial that has small norm since, presumably the quasi-minimization step will handle that. Instead, the basic iteration can be viewed as only generating a set of vectors spanning the Krylov subspace over which the quasi-minimization is applied. This leads us to several variants of QMRCGSTAB which we will briefly describe. Note however that only one of these variants will be used in the numerical experiments.

We make two observations on the QMRCGSTAB method:

1. It is not crucial that the steepest descent step reduces the norm of the residual as long as it increases the degree of the Krylov subspace associated with W_{k+1}.
2. If W_{k+1} were orthogonal, then quasi-minimization becomes true minimization of the residual.

Therefore, it is natural to choose ω_i to make W_{k+1} "more orthogonal". For example, one can choose ω_i to make r_i orthogonal to s_i and W_k pairwise orthogonal. This leads to the formula:

$$\omega_i = \frac{(s_i, s_i)}{(s_i, t_i)}$$

which replaces the corresponding formula in Algorithm QMRCGSTAB. We call this variant QMRCGSTAB2. We note that since the inner-product (s_i, s_i) is already needed to compute $\tilde{\theta}_i$, we save one inner-product compared to QMRCGSTAB.

We also note that similarly to Bi-CGSTAB, both QMRCGSTAB and QMRCGSTAB2 break down if $(s_i, t_i) = 0$ which is possible if A is indefinite (in fact it is always true if A is skew symmetric.) This is an additional breakdown condition over that of BCG. One possible strategy to overcome this is to set a lower bound for the quantity $|(s_i, t_i)|$. However, for matrices with large imaginary parts, Gutknecht [7] observed that Bi-CGSTAB does not perform well because the steepest descent polynomials have only real roots and thus cannot be expected to approximate the spectrum well. In principle, it is possible to derive a quasi-minimal residual version of Gutknecht's variant of Bi-CGSTAB, but we shall not pursue that here.

5. Numerical experiments. We next compare the performance of the QMRCGSTAB variants with that of Bi-CGSTAB, QMRCGS, and CGS.

Table 1 shows the cost per step of the methods under discussion, excluding however the inner product which may be necessary to compute the residual norm.

The number of matrix-vector multiplications is the same for all methods. Although the number of inner products for QMRCGSTAB and QMRCGSTAB2 is higher than that for the corresponding quasi-minimal residual version of CGS, there are fewer floating point operations (flops) for the vector updates.

We will also present several experiments to show that QMRCGSTAB does achieve a smoothing of the residual compared to Bi-CGSTAB. Note however that, because the Bi-CGSTAB method already improves the erratic residual convergence of BCG, the effect of QMRCGSTAB is not as impressive as the one of QMRCGS on the residual of CGS.

Unless stated otherwise, in all examples, the right hand side b was generated as a random vector with values distributed uniformly in $(0, 1)$, and the starting vector x_0 was taken to be zero. All matrices arising from a partial differential operator were obtained using centered,

TABLE 1
Cost per step for each method applied on a system of order n

	inner products	flops for vector updates	matrix-vector multiplications
Bi-CGSTAB	4	$12n$	2
CGS	2	$13n$	2
QMRCGSTAB	6	$16n$	2
QMRCGSTAB2	5	$16n$	2
QMRCGS	4	$20n$	2

second order finite differences. The methods were compared on the basis of the number of iterations necessary to achieve relative residual $\frac{\|r_k\|}{\|r_0\|} < 10^{-8}$. Hence, the figures were built with the abscissae representing the number of iterations and the ordinates representing $\frac{\|r_k\|}{\|r_0\|}$ graded with a logarithmic scale. Experiments were conducted using a Beta test version of Matlab 4.0 [6] running on a Sun Sparc workstation.

Example 1. This example was taken from [14] and corresponds to the discretization of the convection-diffusion operator

$$L(u) = -\varepsilon\triangle u + \cos(\alpha)u_x + \sin(\alpha)u_y \tag{6}$$

on the unit square with homogeneous Dirichlet conditions on the boundary and parameters $\varepsilon = 0.1$ and $\alpha = -30°$, using 40 grid points per direction, yielding a matrix of order $n = 1600$. Fig. 1 shows the convergence histories, from which we can see the smoothing effect of quasi-minimization on the CGS and Bi-CGSTAB residuals. We see that Bi-CGSTAB and its smoothed counterparts converge slightly faster than CGS and QMRCGS, with QMRCGSTAB2 showing the best performance by a small margin.

Example 2. This example was taken from [16] and corresponds to the discretization of

$$-(Du_x)_x - (Du_y)_y = 1 \tag{7}$$

on the unit square with homogeneous boundary conditions. We used a coarser grid than the one considered in [16], that is 50 grid points per direction yielding a matrix of order $n = 2500$. Parameter D takes the value $D = 10^5$ in $0 \leq x, y \leq 0.75$, $D = 0.1$ in $0.75 < x, y \leq 1$, and $D = 1$ everywhere else. Left diagonal preconditioning was applied. In [16], this matrix was used to illustrate the better convergence of Bi-CGSTAB over CGS and we see that this advantage carries over to the smoothed versions (Fig. 2). This is due to the fact that, for operator (7), the direction vectors of Conjugate Gradients methods lose orthogonality rapidly.

Example 3. This example comes from the discretization of the convection-diffusion equation

$$L(u) = -\triangle u + \gamma(xu_x + yu_y) + \beta u \tag{8}$$

on the unit square where $\gamma = 100, \beta = -100$, for a 63×63 grid, yielding a matrix of order $n = 3969$. No preconditioning was used. In this example, we see the CGS-based methods converge a little faster than Bi-CGSTAB and QMRCGSTAB, but the pairwise orthogonal variant, QMRCGSTAB2, is the fastest. See Fig. 3.

Example 4. Figure 4 shows the results of a 3-dimensional version of Example 3 without preconditioning:

$$L(u) = -\triangle u + \gamma(xu_x + yu_y + zu_z) + \beta u \tag{9}$$

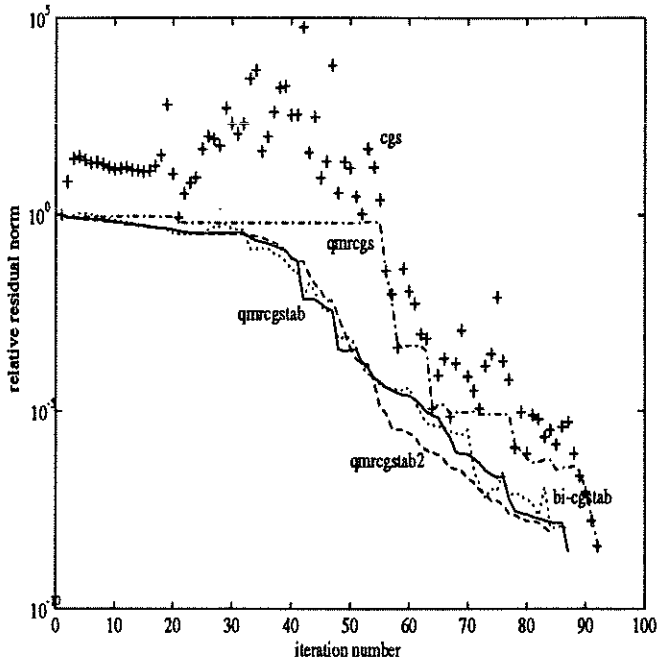

FIG. 1. *Example 1 : 2D conv-diff. operator (6)*

on the unit cube where $\beta = -100$, and $\gamma = 50$ for a $15 \times 15 \times 15$ grid, yielding a matrix of order $n = 3375$.

We note that in this example the improvement caused by Bi-CGSTAB over CGS and QMRCGS is impressive. Therefore it is not surprising that there is only little additional improvement brought by the variants proposed in this paper. We note that for this operator, the use of centered differences and large values of γ are unfavorable for Bi-CGSTAB-type methods, since the resulting matrices would have pronounced skew-symmetric component, and eigenvalues with large imaginary parts [7]; different discretization methods would be more attractive [13].

Example 5. The next example illustrates how all methods can be affected by the conditioning of the generated polynomial.

$$A = I_{n/2} \otimes \begin{pmatrix} \epsilon & 1 \\ -25 & 100 \end{pmatrix} \tag{10}$$

i.e., A is an $n \times n$ block diagonal matrix with 2×2 blocks and $n = 40$. We chose $b = (1\ 0\ 1\ 0\ \cdots)^T$, and $\tilde{r}_0 = r_0$. For such a b the norm of the resulting BCG polynomial satisfies $\|\varphi_n\| = \mathcal{O}(\epsilon^{-1})$. Thus, $\|\varphi_n^2\| = \mathcal{O}(\epsilon^{-2})$ in the squared methods and we can foresee numerical problems when ϵ is small.

Each entry of Table 2 shows (i) the number d of correct digits reached in the relative residual while running each algorithm for a maximum of 20 matrix vector multiplications or until the relative residual dropped below 10^{-8}, and (ii) in parentheses, the number of matrix vector multiplications, mv, that is a number, not greater than 20, needed to reach the relative residual 10^{-d}.

In exact arithmetic, finite termination occurs after the second BCG polynomial φ_2 is

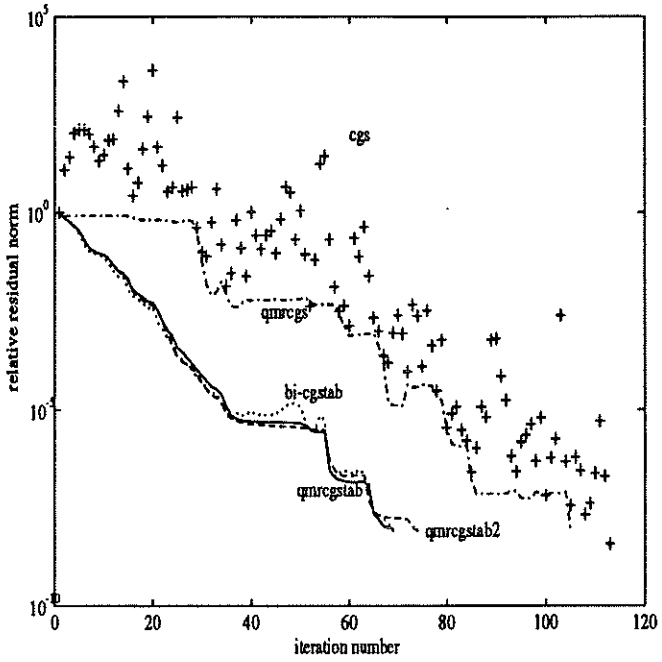

FIG. 2. *Example 2 : 2D discontinuous coefficients*

computed in both the CGS and Bi-CGSTAB algorithms. We see from Table 2 that all methods behave equally well for $\epsilon = 1.0$. As ϵ decreases, round-off error causes CGS and QMRCGS, which are based on squaring, to fail or not to converge within the expected time. Furthermore, both CGS and QMRCGS lose about twice as many digits as Bi-CGSTAB and its quasi-minimal variants. We also mark the instances of the quasi-minimal variants whose residuals stagnate before the maximum number of iterations has been reached.

We note that although the example is contrived, it does justify the implementation of a QMRCGSTAB-type method.

Example 6. We conclude the numerical experiments with an overall comparison of the methods on matrices from a standard test set, namely the Harwell-Boeing collection [1]. This experiment is conducted on a Silicon Graphics computer SGI IRIS 4D/240S which uses 64-bit IEEE standard floating point arithmetic. Table 3 displays the number of iterations to achieve a relative residual $\frac{\|r_k\|}{\|r_0\|} < 10^{-8}$. The right-hand side was computed in order to have a random solution vector. We also include iteration counts for preconditioned versions of all algorithms.

The preconditioner used is Saad's ILUT(p, τ), described in [12], with the actual subroutine borrowed from the implementation by Freund and Nachtigal. ILUT(p, τ) is based on two parameters controlling fill-in. The first parameter allows p elements in each row of the lower and upper triangular factors in addition to the number of nonzero elements originally in the lower and upper triangular parts of A. All elements smaller than the threshold τ times a normalization factor are also dropped. We used ILUT(0, 0) for all problems in this experiment. Since the factors from ILUT(0, 0) are as sparse as the problem matrix A, the number of operations in applying ILUT(0, 0) is ordinarily not more than two matrix vector

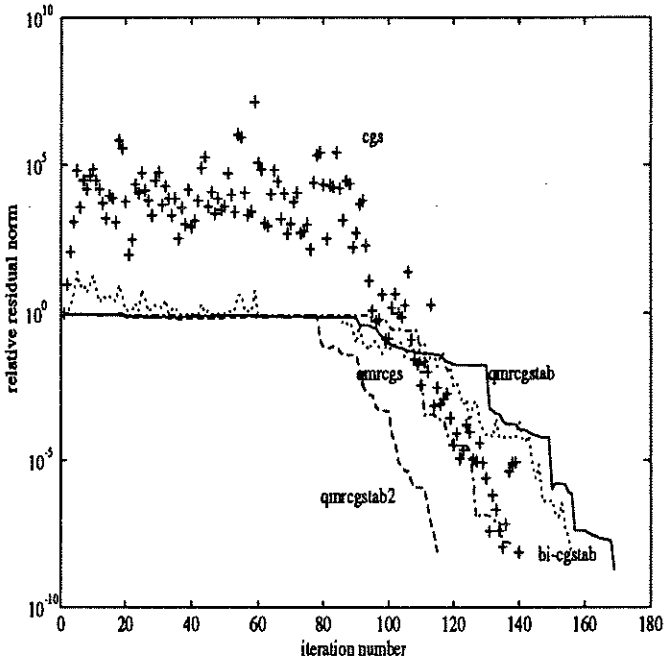

FIG. 3. *Example 3 : 2D conv-diff. operator (8)*

multiplications[2]. Thus ignoring the cost of the factorization, the (not surprising) observation one can make for Table 3 is that a good preconditioner greatly improves performance of all methods. More important for our comparison is that the Bi-CGSTAB and CGS variants give generally comparable performance in terms of number of iterations.

6. Conclusions and future work. We have derived two QMR variants of Bi-CGSTAB. Our motivation for these methods was to inherit any potential improvements on performance Bi-CGSTAB offers over CGS, while at the same time provide a smoother convergence behavior. We have shown numerically that this is indeed true for many realistic problems. Although in their present form, the two proposed methods still suffer from some numerical problems, they have many desirable properties: they are transpose-free, they use short recurrences, they make efficient use of matrix-vector multiplications and demonstrate smooth convergence behavior.

7. Acknowledgments. We are grateful to R. Freund and N. Nachtigal for letting us use the ILUT module from their QMR package, and to J. M. Hammond, C. Moler, and The Mathworks Inc. for providing us with the Beta Test version of Matlab 4.0. [6]. We also thank R. Freund and the referees for providing us with very useful suggestions and corrections.

REFERENCES

[1] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Softw., 15(1989),pp. 1-14.

[2] We emphasize that this statement discusses scalar computational cost without considering the effects of vector or parallel processing to performance.

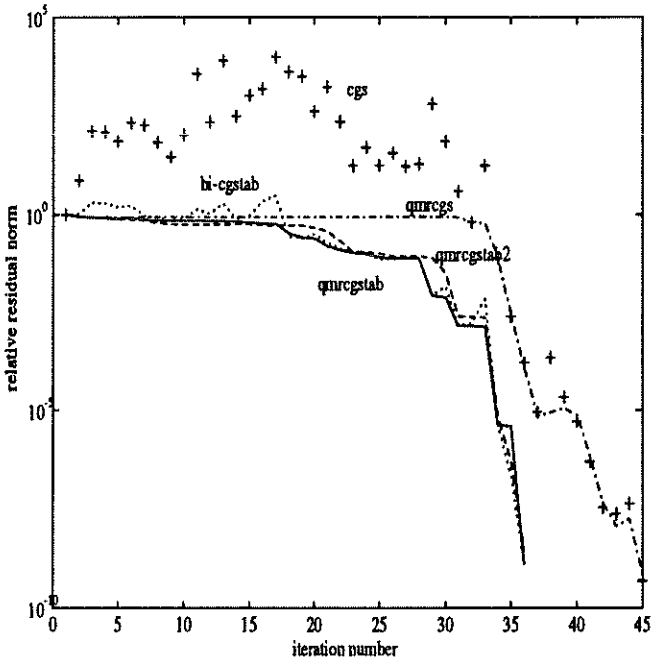

FIG. 4. *Example 4 : 3D conv-diffusion operator (9)*

[2] R. FLETCHER, *Conjugate gradient methods for indefinite linear systems*, in Proc. Dundee Biennial Conf. Numer. Anal., G. A. Watson, ed., vol. 506 of Lect. Notes Math., Springer-Verlag, Berlin, 1976, pp. 73–89.

[3] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, tech. rep., RIACS, NASA Ames Research Center, Sept. 1991. Presented at the Fourth SIAM Conf. Appl. Lin. Alg., Minneapolis.

[4] R. W. FREUND, M. H. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part I*, Tech. Rep. 90.45, RIACS, NASA Ames Research Center, Nov. 1990.

[5] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numerische Mathematik 60(1991), pp. 315-339.

[6] THE MATHWORKS, INC., *MATLAB User's Guide*, Natick, Mass. 01760, Beta 3 ed., Feb. 1991.

[7] M. H. GUTKNECHT, *Variants of BiCGStab for matrices with complex spectrum*, IPS Res. Rep. 91-14, Eidgenössische Technische Hochschule, Zürich, Aug. 1991.

[8] W. D. JOUBERT AND T. A. MANTEUFFEL, *Iterative methods for nonsymmetric linear systems*, in Iterative Methods for Large Linear Systems, D. R. Kincaid and L. J. Hayes, eds., Academic Press, Boston, 1990, pp. 149–171.

[9] C. LANCZOS, *Solution of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand. 49 (1952), 33-53.

[10] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (July 1992), pp. 778-795.

[11] C. POMMERELL AND W. FICHTNER, *PILS: An iterative linear solver package for ill-conditioned systems*, in Proc. Supercomputing'91, Albuquerque, New Mexico, Nov. 1991, IEEE, pp. 588–599.

[12] Y. SAAD, *Highly parallel preconditioners for general sparse matrices*, Tech. Rep. 92-087, Army High Performance Computing Research Center, Univ. Minnesota, Minneapolis, July 1992.

[13] A. SEGAL, *Aspects of numerical methods for elliptic singular perturbation problems*, SIAM J. Sci. Stat. Comput., 3 (Sep. 1982), pp. 327-349.

[14] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (Jan. 1989), pp. 36–52.

[15] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of*

TABLE 2

Correct digits and matrix vector multiplications at termination: d(mv). A max. 20 matrix vector mult. allowed.

	ϵ			
Method	1.0	10^{-4}	10^{-8}	10^{-12}
CGS	14(4)	5(4)‡	−3(20)∗	−1(4)†
QMRCGS	13(3)	5(4)‡	0(20)	0(20)
Bi-CGSTAB	16(3)	12(3)	7(3)‡	3(3)‡
QMRCGSTAB	16(3)	12(3)	7(3)‡	3(3)‡
QMRCGSTAB2	16(3)	12(3)	7(3)‡	3(3)‡

* Oscillatory behavior observed.

‡ Residual stagnated before max. number of *mv*'s was reached.

† Iterations stopped when division by zero was encountered.

TABLE 3

Iteration Counts for Some Problems in the Harwell-Boeing Collection

Problem	order	Bi-CGSTAB		QMRCGSTAB		QMRCGSTAB2		CGS		QMRCGS	
		(U)	(P)	(U)	(P)	(U)	(P)	(U)	(P)	(U)	(P)
orsreg_1	1030	303	16	308	16	312	16	172	16	171	16
orsirr_1	886	1329	12	1437	12	937	12	622	14	621	14
orsirr_2	2205	834	12	882	12	763	12	450	15	445	15
pores1	30	200	15	179	15	159	16	208	15	144	15
pores2	1224	**	32	**	32	**	32	**	34	**	34
pores3	532	1698	17	1795	17	1530	17	1627	20	1880	20
sherman1	1000	383	18	400	18	358	18	303	20	298	20
sherman2	1080	**	15	**	15	**	14	**	13	**	13
sherman3	5005	**	66	**	65	**	66	**	70	**	70
sherman4	1104	110	25	110	25	109	25	113	25	115	25
sherman5	3312	1846	15	1848	15	**	15	**	18	1293	17
saylor1	238	**	9	**	9	**	9	**	11	**	11
saylor3	1000	336	18	345	18	379	18	293	20	296	20
saylor4	3564	**	32	**	32	**	31	**	39	**	39

(U) - no preconditioning; (P) - ILUT(0, 0) preconditioning used;

** - max. number (2000) of iterations exceeded; $\tilde{r}_0 = r_0$ or $w_1 = v_1$.

nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 13 (Mar. 1992), pp. 631–644.

[16] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, Tech. Rep. No. 633, Dept. of Math., University of Utrecht, NL, Dec. 1990.

[17] H. A. VAN DER VORST, *The convergence behavior of preconditioned CG and CG-S in the presence of rounding errors*, in Proceedings of the PCG Conference, Nijmegen, O. Axelsson, ed., June 1989.