UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Mixed Product Krylov Subspace Method for Solving Nonsymmetric Linear Systems

Tony F. Chan
Qiang Ye

September 1997

CAM Report 97-41

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

A Mixed Product Krylov Subspace Method for Solving Nonsymmetric Linear systems

Tony F. Chan * Qiang Ye †

Abstract

In this paper, a product Krylov subspace method that we call mixed BiCGSTAB-CGS is derived. The method is built on the idea of the standard CGS and BiCGSTAB iterations but allows switching between the two at each iteration. This flexibility can be used, for example, to address the difficulty of excessive increase in residual norm in CGS, which may cause instability. In particular, a CGS based implementation will be presented, which can be regarded as another way of using the BiCGSTAB idea to improve the stability of CGS. Numerical examples are given to demonstrate the stabilizing effect of the mixed algorithm.

1 Introduction

Iterative methods for solving a large nonsymmetric linear systems $Ax = b$ that extract approximate solutions from the Krylov subspace $K_n = \text{span}\{b, Ab, A^2b, \cdots, A^nb\}$ are usually called Krylov subspace methods. The BiCG algorithm [4, 9] is a classical Krylov subspace method that produces an approximation x_n with a residual reduction $r_n = b - Ax_n = P_n(A)r_0$ ($r_0 = b$) where P_n is a polynomial of degree n and called the BiCG polynomial.

In the past few years, several efficient product Krylov subspace methods such as CGS [13] and BiCGSTAB [15] have been developed to accelerate convergence in BiCG and to avoid multiplications by the transpose of A (see [1, 8, 10] for a general review and comparison). These product type methods are based on constructing the product of the BiCG polynomial and some other accelerating polynomials. In comparison with BiCG, which requires $2n$ matrix-vector multiplications at iteration n, CGS uses the same amount of matrix-vector multiplications to produce a reduction with P_n^2, i.e., $r_n = P_n^2(A)r_0$. This potentially accelerates the convergence by a factor of 2. Unfortunately, during the early stage of iteration, $P_n(A)$ may not be a contraction and thus r_n may grow substantially or vary significantly in magnitude, causing instability or stagnation of the true residual [12]. A remedy to this problem is BiCGSTAB [15] (or BiCGSTAB(l) [7, 11]), which constructs $r_n = P_n(A)Q_n(A)r_0$ with Q_n being a product of polynomials of degree 1 (or l resp.) constructed locally to minimize the residual. Indeed, the BiCGSTAB type methods have proved to be very efficient in smoothing and accelerating the convergence and is currently one of the most popular methods. However, there are

*Department of Mathematics, UCLA, Los Angeles, CA 90095-1555. E-mail : chan@math.ucla.edu. Research supported by NSF DMS-9626755 and ARO contract DAAL-03-91-C-0047 (Univ. Tenn. subcontract ORA4466.04 Amendment 1 and 2)

†Department of Applied Mathematics, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2. E-mail: ye@newton.amath.umanitoba.ca. Research supported by a grant from Natural Sciences and Engineering Research Council of Canada

many cases where CGS remains competitive. For example, BiCGSTAB may suffer from a type of near-breakdown that is associated with BiCGSTAB only and may result in stagnation of residuals.

Since the difficulties encountered in CGS and BiCGSTAB are of different type and usually occur only at a small subset of the iteration steps, it might be advantageous to consider a combination of the two that can choose either of the two kinds of construction at each iteration and avoid using the one for which difficulties arise. Also local steepest descent is used throughout BiCGSTAB iterations, which may not be optimal in exploiting the global information available in the Krylov sequences. Therefore, it might still be desirable to construct residual reduction like $P_n^2(A)r_0$ but in a way that excessive increase in the residual norm is controlled. In this paper, we introduce a general concept of switching product Krylov subspace methods from one type to another through appropriately defining the sequence of polynomials. In this regard, we shall derive a mixed method that is based on the CGS and BiCGSTAB iterations but has the freedom at each iteration to carry out either a CGS step or a BiCGSTAB step. This is done without resorting to restart, which would lose all global information that has been built up in the iterations. In particular, we present a CGS based implementation that takes a BiCGSTAB step only when it is necessary, i.e., when there is a large increase in the residual norm. In this way, it can be regarded as another way of using the BiCGSTAB idea to improve the stability of CGS.

There is a recent work [16] on a general product method that constructs residuals of the form $r_n = P_n(A)S_n(A)r_0$ with S_n being any polynomial sequence satisfying a three term recurrence. The mixed methods to be presented here also constructs residuals of the form $r_n = P_n(A)S_n(A)r_0$ but S_n does not satisfy any three term recurrence when there is a switching. Therefore, it does not fall into the framework of [16]. There are also other works such as those of [2, 6, 12] that address various difficulties in CGS. Here, we shall concentrate on the approach of constructing product polynomials that are slightly different from CGS.

The paper is organized as follows. In section 2, we present the derivation of the general mixed algorithm. In section 3, we briefly describe a CGS based implementation. Then, we give in section 4 some numerical examples to illustrate the stabilizing effect of the mixed algorithm and finally in section 5 concluding remarks.

2 The mixed product algorithm

Consider applying BiCG to the matrix A with the initial left and right residuals $\tilde{r}_0$, r_0 and denote the nth residual vector by r_n^{BiCG} and the nth search direction vector by p_n^{BiCG}. Let P_n be the nth BiCG polynomial corresponding to r_n^{BiCG} , and T_n be the nth polynomial corresponding to p_n^{BiCG} (see [15] for example), i.e.,

$$P_n(A)r_0 = r_n^{BiCG}, \quad T_n(A)r_0 = p_n^{BiCG}.$$

Then, from the BiCG recurrence, we have

$$P_{n+1}(t) = P_n(t) - \alpha_n t T_n(t),$$

$$T_{n+1}(t) = P_{n+1}(t) + \beta_{n+1} T_n(t),$$

where

$$\alpha_n = \frac{(P_n(A^T)\tilde{r}_0, P_n(A)r_0)}{(T_n(A^T)\tilde{r}_0, AT_n(A)r_0)} \quad \text{and} \quad \beta_{n+1} = \frac{(P_{n+1}(A^T)\tilde{r}_0, P_{n+1}(A)r_0)}{(P_n(A^T)\tilde{r}_0, P_n(A)r_0)}.$$

The choice of α_n and β_n ensures the following orthogonality properties

$$(p(A^T)\tilde{r}_0, P_n(A)r_0) = 0 \quad \text{and} \quad (p(A^T)\tilde{r}_0, AT_n(A)r_0) = 0 \tag{1}$$

where p is any polynomial of degree less than n.

To accelerate the convergence in BiCG and to avaid multiplications by the transpose of A, CGS constructs an approximation x_n such that the residual is $r_n = b - Ax_n = P_n^2(A)r_0$, and in BiCGSTAB, it is $r_n = P_n(A)Q_n(A)r_0$ where $Q_n(t) = (1-\omega_1 t)\cdots(1-\omega_n t)$.

We propose a general mixed product method that constructs approximations x_n such that its residual has the form

$$r_n = b - Ax_n = S_n(A)P_n(A)r_0,$$

where

$$S_n(t) = Q_k(t)P_{n-k}(t) \text{ and } Q_k(t) = (1-\omega_1 t)\cdots(1-\omega_k t)$$

and k is an integer parameter that determines what kind of residual reduction is used. When constructing $r_{n+1} = S_{n+1}(A)P_{n+1}(A)r_0$ from r_n, we can choose either $S_{n+1}(t) = Q_k(t)P_{n+1-k}(t)$ or $S_{n+1}(t) = Q_{k+1}(t)P_{n-k}(t)$. We call the fromer a CGS step and the latter a BiCGSTAB step. So, essentially, in the first n iterations, we take k steps of BiCGSTAB and $n-k$ steps of CGS. With $k > 0$, this approach avoids squaring P_n in the CGS and thus may have a stabilizing effect. We note that Fokkema, Sleijpen and van der Vorst [5] have also suggested the use of $P_{n-1}(A)P_n(A)r_0$ as a possible way to improve the stability of CGS.

As mentioned in the introduction, [16] gives a general algorithm for constructing $r_n = S_n(A)P_n(A)r_0$ if S_n satisfies a three term recurrence. It is easy to see that, when there is switching from a CGS step to a BiCGSTAB step and back, S_n in our case no longer satisfies a three term recurrence. Therefore, the general algorithm of [16] does not apply. The question now is whether r_n defined above can still be constructed by some simple recurrence. It turns out that this can be done with a little extra cost. We now proceed to derive such a recurrence. The derivation is similar to those used in CGS and BiCGSTAB and will be in terms of polynomials, from which the corresponding vector recurrence follows.

Since k is a parameter that may vary from step to step (thus a function of n), we denote its value at the nth step by $k(n)$. As in CGS and BiCGSTAB, we define the following auxiliary polynomials and the corresponding vectors:

$$\phi_n(t) = P_{n-k}(t)Q_k(t)P_n(t), \quad \text{and} \quad r_n = \phi_n(A)r_0 = P_{n-k}(A)Q_k(A)P_n(A)r_0; \tag{2}$$

$$\xi_n(t) = P_{n-k}(t)Q_k(t)T_n(t), \quad \text{and} \quad u_n = \xi_n(A)r_0 = P_{n-k}(A)Q_k(A)T_n(A)r_0; \tag{3}$$

$$\zeta_n(t) = T_{n-k}(t)Q_k(t)P_n(t), \quad \text{and} \quad v_n = \zeta_n(A)r_0 = T_{n-k}(A)Q_k(A)P_n(A)r_0; \tag{4}$$

$$\eta_n(t) = T_{n-k}(t)Q_k(t)T_n(t), \quad \text{and} \quad p_n = \eta_n(A)r_0 = T_{n-k}(A)Q_k(A)T_n(A)r_0 \tag{5}$$

where $k = k(n)$.

Assume the above four polynomials have been constructed for a fixed n with $k = k(n)$. In proceeding to step $n+1$ from step n, we first choose $k(n+1)$ as either k or $k+1$, and then construct correspondingly either a CGS step

$$\phi_{n+1}(t) = P_{n+1-k(n+1)}(t)Q_{k(n+1)}(t)P_{n+1}(t) = P_{n+1-k}(t)Q_k(t)P_{n+1}(t)$$

or a BiCGSTAB step

$$\phi_{n+1}(t) = P_{n+1-k(n+1)}(t)Q_{k(n+1)}(t)P_{n+1}(t) = (I - \omega_{k+1}t)P_{n-k}(t)Q_k(t)P_{n+1}(t).$$

We now derive the recurrence for the construction of the four polynomials for each of the two cases.

Case 1: $k(n+1) = k(n) = k$ (i.e. a CGS step). In this case, we first generate $\psi_n = T_{n-k}(t)Q_k(t)P_{n+1}(t)$ by

$$\psi_n = T_{n-k}Q_k(P_n - \alpha_n t T_n) = \zeta_n - \alpha_n t \eta_n.$$

Then

$$\begin{aligned}
\phi_{n+1} &= P_{n+1-k}Q_k P_{n+1} \\
&= P_{n-k}Q_k P_{n+1} - \alpha_{n-k} t T_{n-k}Q_k P_{n+1} \\
&= P_{n-k}Q_k(P_n - \alpha_n t T_n) - \alpha_{n-k} t T_{n-k}Q_k P_{n+1} \\
&= \phi_n - t(\alpha_n \xi_n + \alpha_{n-k}\psi_n),
\end{aligned}$$

$$\begin{aligned}
\xi_{n+1} &= P_{n+1-k}Q_k T_{n+1} \\
&= P_{n+1-k}Q_k P_{n+1} + \beta_{n+1}P_{n+1-k}Q_k T_n \\
&= P_{n+1-k}Q_k P_{n+1} + \beta_{n+1}(P_{n-k} - \alpha_{n-k}tT_{n-k})Q_k T_n \\
&= \phi_{n+1} + \beta_{n+1}(\xi_n - \alpha_{n-k}t\eta_n),
\end{aligned}$$

$$\begin{aligned}
\zeta_{n+1} &= T_{n+1-k}Q_k P_{n+1} \\
&= P_{n+1-k}Q_k P_{n+1} + \beta_{n+1-k}T_{n-k}Q_k P_{n+1} \\
&= \phi_{n+1} + \beta_{n+1-k}\psi_n,
\end{aligned}$$

$$\begin{aligned}
\eta_{n+1} &= T_{n+1-k}Q_k T_{n+1} \\
&= P_{n+1-k}Q_k T_{n+1} + \beta_{n+1-k}T_{n-k}Q_k T_{n+1} \\
&= P_{n+1-k}Q_k T_{n+1} + \beta_{n+1-k}T_{n-k}Q_k(P_{n+1} + \beta_{n+1}T_n) \\
&= \xi_{n+1} + \beta_{n+1-k}(\psi_n + \beta_{n+1}\eta_n).
\end{aligned}$$

Immediately, by writing $q_n = \psi_n(A)r_0$, these recurrence can be expressed in the vector form

$$\begin{aligned}
q_n &= v_n - \alpha_n A p_n, \\
r_{n+1} &= r_n - A(\alpha_n u_n + \alpha_{n-k}q_n), \\
u_{n+1} &= r_{n+1} + \beta_{n+1}(u_n - \alpha_{n-k}Ap_n), \\
v_{n+1} &= r_{n+1} + \beta_{n+1-k}q_n, \\
p_{n+1} &= u_{n+1} + \beta_{n+1-k}(q_n + \beta_{n+1}p_n).
\end{aligned}$$

Case 2: $k(n+1) = k(n)+1 = k+1$ (i.e. a BiCGSTAB step). In this case, $Q_{k+1} = (1-\omega_{k+1}t)Q_k$.

$$\begin{aligned}
\phi_{n+1} &= P_{n+1-(k+1)}Q_{k+1}P_{n+1} \\
&= P_{n-k}Q_{k+1}(P_n - \alpha_n t T_n)
\end{aligned}$$

$$
\begin{aligned}
&= (1-\omega_{k+1}t)(P_{n-k}Q_kP_n - \alpha_n t P_{n-k}Q_kT_n) \\
&= (1-\omega_{k+1}t)(\phi_n - \alpha_n t\xi_n),
\end{aligned}
$$

$$
\begin{aligned}
\xi_{n+1} &= P_{n+1-(k+1)}Q_{k+1}T_{n+1} \\
&= P_{n-k}Q_{k+1}P_{n+1} + \beta_{n+1}P_{n-k}Q_{k+1}T_n \\
&= \phi_{n+1} + \beta_{n+1}(1-\omega_{k+1}t)\xi_n,
\end{aligned}
$$

$$
\begin{aligned}
\zeta_{n+1} &= T_{n+1-(k+1)}Q_{k+1}P_{n+1} \\
&= T_{n-k}Q_{k+1}P_n - \alpha_n t T_{n-k}Q_{k+1}T_n \\
&= (1-\omega_{k+1}t)(\zeta_n - \alpha_n t\eta_n),
\end{aligned}
$$

$$
\begin{aligned}
\eta_{n+1} &= T_{n+1-(k+1)}Q_{k+1}T_{n+1} \\
&= T_{n-k}Q_{k+1}P_{n+1} + \beta_{n+1}T_{n-k}Q_{k+1}T_n \\
&= \zeta_{n+1} + \beta_{n+1}(1-\omega_{k+1}t)\eta_n.
\end{aligned}
$$

Again, we have the corresponding vector recurrence

$$
\begin{aligned}
r_{n+1} &= (I-\omega_{k+1}A)(r_n - \alpha_n Au_n), \\
u_{n+1} &= r_{n+1} + \beta_{n+1}(I-\omega_{k+1}A)u_n, \\
v_{n+1} &= (I-\omega_{k+1}A)(v_n - \alpha_n Ap_n), \\
p_{n+1} &= v_{n+1} + \beta_{n+1}(I-\omega_{k+1}A)p_n.
\end{aligned}
$$

In this case, ω_{k+1} is determined to minimize $r_{n+1} = (I-\omega_{k+1}A)(r_n - \alpha_n Au_n)$.

This completes the recurrence from step n to step $n+1$. Of course, a recurrence for the approximate solution x_n can be easily obtained from that of r_n (see Algorithm 2.1 below).

To finish the construction, we also need to recover the BiCG coefficients α_{n+1} and β_{n+1}. First note that by the orthogonality (1), we have

$$
(\tilde{r}_0, \zeta_n(A)r_0) - (\tilde{r}_0, \phi_n(A)r_0) = \beta_{n-k}(T_{n-k-1}(A^T)Q_k(A^T)\tilde{r}_0, P_n(A)r_0) = 0
$$

and

$$
(\tilde{r}_0, A\eta_n(A)r_0) - (\tilde{r}_0, A\xi_n(A)r_0) = \beta_{n-k}(T_{n-k-1}(A^T)Q_k(A^T)\tilde{r}_0, AT_n(A)r_0) = 0.
$$

Then

$$
(\tilde{r}_0, v_n) = (\tilde{r}_0, r_n) \text{ and } (\tilde{r}_0, Ap_n) = (\tilde{r}_0, Au_n).
$$

Again we need to separate the two cases.

Case 1: From the definition of q_n, we have the following orthogonality

$$
\tilde{r}_0^T q_n = (\tilde{r}_0, \psi_n(A)r_0) = (T_{n-k}(A^T)Q_k(A^T)\tilde{r}_0, P_{n+1}(A)r_0) = 0,
$$

where we have used (1). Thus, $\tilde{r}_0^T v_n - \alpha_n \tilde{r}_0^T Ap_n = \tilde{r}_0^T q_n = 0$, i.e.,

$$
\alpha_n = \frac{\tilde{r}_0^T v_n}{\tilde{r}_0^T Ap_n} = \frac{\tilde{r}_0^T r_n}{\tilde{r}_0^T Ap_n}. \tag{6}
$$

Similarly,

$$(\tilde{r}_0, AT_{n-k}(A)Q_k(A)T_{n+1}(A)r_0) = (T_{n-k}(A^T)Q_k(A^T)\tilde{r}_0, AT_{n+1}(A)r_0) = 0.$$

Using $\psi_n(t) + \beta_{n+1}\eta_n(t) = T_{n-k}(t)Q_k(t)T_{n+1}(t)$, we obtain

$$(\tilde{r}_0, A\psi_n(A)r_0) + \beta_{n+1}(\tilde{r}_0, A\eta_n(A)r_0) = 0,$$

and thus

$$\beta_{n+1} = -\frac{\tilde{r}_0^T A q_n}{\tilde{r}_0^T A p_n} = -\frac{\tilde{r}_0^T A q_n}{\tilde{r}_0^T A u_n}. \tag{7}$$

Furthermore, it follows from (6) that $\tilde{r}_0^T r_{n+1} = \tilde{r}_0^T r_n - \alpha_n \tilde{r}_0^T A p_n - \alpha_{n-k}\tilde{r}_0^T A q_n = -\alpha_{n-k}\tilde{r}_0^T A q_n$. Substituting this and (6) into (7), we obtain a second formula for β_{n+1}

$$\beta_{n+1} = \frac{\tilde{r}_0^T r_{n+1}}{\tilde{r}_0^T r_n}\frac{\alpha_n}{\alpha_{n-k}}. \tag{8}$$

Case 2: From

$$(\tilde{r}_0, P_{n-k}(A)Q_k(A)P_{n+1}(A)r_0) = (P_{n-k}(A^T)Q_k(A^T)\tilde{r}_0, P_{n+1}(A)r_0) = 0,$$

and $P_{n-k}(t)Q_k(t)P_{n+1}(t) = \phi_n(t) - \alpha_n t\xi_n(t)$, we obtain

$$\alpha_n = \frac{\tilde{r}_0^T r_n}{\tilde{r}_0^T A u_n}. \tag{9}$$

On the other hand, noting that

$$(\tilde{r}_0, AP_{n-k}(A)Q_k(A)T_{n+1}(A)r_0) = (P_{n-k}(A^T)Q_k(A^T)\tilde{r}_0, AT_{n+1}(A)r_0) = 0.$$

and $P_{n-k}(t)Q_k(t)T_{n+1}(t) = v(t) + \beta_{n+1}\xi_n(t)$ with $v(t) = P_{n-k}(t)Q_k(t)P_{n+1}(t) = \phi_n(t) - \alpha_n t\xi_n(t)$, we have

$$\beta_{n+1} = -\frac{(\tilde{r}_0, Av(A)r_0)}{(\tilde{r}_0, A\xi_n(A)r_0)} = -\frac{\tilde{r}_0^T A(r_n - \alpha_n A u_n)}{\tilde{r}_0^T A u_n}. \tag{10}$$

Furthermore, $(\tilde{r}_0, v(A)r_0) = 0$ and hence

$$\tilde{r}_0^T r_{n+1} = (\tilde{r}_0, \phi_{n+1}(A)r_0) = (\tilde{r}_0, v(A)r_0) - \omega_{k+1}(\tilde{r}_0, Av(A)r_0) = -\omega_{k+1}(\tilde{r}_0, Av(A)r_0).$$

Using this and (9), we obtain again a second formula

$$\beta_{n+1} = \frac{\alpha_n}{\omega_{k+1}}\frac{\tilde{r}_0^T r_{n+1}}{\tilde{r}_0^T r_n}. \tag{11}$$

We remark that (6, 8, 9, 11) can also be derived by the method of [13, 15]. The derivation given here is based on some orthogonality relations among the vectors and leads to alternative formulas (7, 10) for β_n.

Finally, we summarize the above derivation in the following algorithm:

Mixed-BiCGSTAB-CGS Algorithm:
Input an initial approximation x_0 and an auxiliary vector $\tilde{r}_0$;
Initialize $r_0 = u_0 = v_0 = p_0 = b - Ax_0$; $k = 0$; $\rho_0 = \tilde{r}_0^T r_0$.

For $n = 0, 1, 2, \cdots$ until convergence
Determine whether $k \leftarrow k$ (CGS step) or $k \leftarrow k+1$ (BiCGSTAB step);
If (CGS), then

$$\alpha_n = \rho_n / \tilde{r}_0^T A p_n$$
$$q_n = v_n - \alpha_n A p_n$$
$$r_{n+1} = r_n - A(\alpha_n u_n + \alpha_{n-k} q_n)$$
$$x_{n+1} = x_n + \alpha_n u_n + \alpha_{n-k} q_n$$
$$\rho_{n+1} = \tilde{r}_0^T r_{n+1};$$
$$\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\alpha_{n-k} \rho_n} \quad (\text{or} = -\frac{\tilde{r}_0^T A q_n}{\tilde{r}_0^T A p_n})$$
$$u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \alpha_{n-k} A p_n)$$
$$v_{n+1} = r_{n+1} + \beta_{n+1-k} q_n$$
$$p_{n+1} = u_{n+1} + \beta_{n+1-k}(q_n + \beta_{n+1} p_n)$$

End if
If (BiCGSTAB), then

$$\alpha_n = \rho_n / \tilde{r}_0^T A u_n$$
$$v = r_n - \alpha_n A u_n$$
$$\omega = v^T A v / (Av)^T Av;$$
$$r_{n+1} = v - \omega A v$$
$$x_{n+1} = x_n + \alpha_n u_n + \omega v$$
$$\rho_{n+1} = \tilde{r}_0^T r_{n+1};$$
$$\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\omega \rho_n} \quad (\text{or} = -\frac{\tilde{r}_0^T A v}{\tilde{r}_0^T A u_n})$$
$$u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \omega A u_n)$$
$$v_{n+1} = (I - \omega A)(v_n - \alpha_n A p_n)$$
$$p_{n+1} = v_{n+1} + \beta_{n+1}(p_n - \omega A p_n)$$

End if
End for

We present some remarks concerning the algorithm.

Remark 1: It is easy to see that one iteration of the mixed algorithm requires two multiplications by A, if it is a CGS step, and four multiplications, if it is a BiCGSTAB step.

Remark 2: If all iterations are CGS steps, i.e., $k(n) = 0$ for all n, then the recurrence derived is equivalent to the standard CGS.

Remark 3: If all iterations are BiCGSTAB steps, i.e. $k(n) = n$ for all n, then the recurrences for r_n and u_n are identical to BiCGSTAB. Note that the recurrences for p_n and v_n are independent of r_n and u_n and therefore in the case of BiCGSTAB, p_n and v_n need not be constructed. Indeed, p_n and v_n are generated here solely for the use in later CGS steps. This leads to two extra multiplications by A, resulting in the four multiplications by A as opposed to two in the standard BiCGSTAB.

Remark 4: We shall discuss in the next section a criterion for switching between the two kinds of steps. However, the algorithm can also be implemented by first taking p consecutive steps of BiCGSTAB ($k(n) = 0$ for $n \leq p$) to reduce the residual norm to certain level, and then switch to CGS steps completely ($k(n) = n - p$ for $n > p$). In that case, for the first p BiCGSTAB steps, $v_n = r_n$ and $p_n = u_n$ since $r_0 = u_0 = v_0 = p_0$. Thus v_n and p_n need not be constructed, which reduces the multiplications by A to two. For example, by taking one BiCGSTAB step and switch to CGS, we recover the method of [5]. On the other hand, it is also possible to take the first p steps as CGS steps ($k(n) = 0$ for $n \leq p$) and then switch to BiCGSTAB steps completely ($k(n) = n - p$ for $n > p$). In this case, again p_n and v_n in the remaining BiCGSTAB steps need not be constructed. We have examples where a complete switching like this achieves better convergence than CGS and BiCGSTAB; but at present, we do not see a practical implementation emerging from this approach as some *a priori* information is needed in determining the step at which to carry out the switching.

A preconditioned version of the algorithm can easily be worked out. The following is one such form that uses $K = K_1K_2$ as a preconditioner, K_1 being the left preconditioner and K_2 being the right one. This includes the case that K itself is used as the left preconditioner (i.e. $K = K_1I$) or as the right preconditioner (i.e. $K = IK_2$). Note that the choice of K_1 and K_2 plays no explicit role in the algorithm except that the left initial vector $\tilde{r}_0$ used in the explicitly preconditioned version is replaced by $(K_1^{-1})^T\tilde{r}_0$. However, if there is no particular choice for $\tilde{r}_0$ in the explicitly preconditioned version, then such a replacement is not necessary. The detailed derivation is similar to [15].

Preconditioned Mixed-BiCGSTAB-CGS Algorithm:
Input an initial approximation x_0 and an auxiliary vector $\tilde{r}_0$; $\tilde{r}_0 \leftarrow (K_1^{-1})^T\tilde{r}_0$;
Initialize $r_0 = u_0 = v_0 = p_0 = b - Ax_0$; $k = 0$; $\rho_0 = \tilde{r}_0^T r_0$.

For $n = 0, 1, 2, \cdots$ until convergence
Determine whether $k \leftarrow k$ (CGS step) or $k \leftarrow k + 1$ (BiCGSTAB step);
If (CGS), then

$$\alpha_n = \rho_n / \tilde{r}_0^T AK^{-1}p_n$$
$$q_n = v_n - \alpha_n AK^{-1}p_n$$
$$r_{n+1} = r_n - AK^{-1}(\alpha_n u_n + \alpha_{n-k}q_n)$$
$$x_{n+1} = x_n + K^{-1}(\alpha_n u_n + \alpha_{n-k}q_n)$$
$$\rho_{n+1} = \tilde{r}_0^T r_{n+1}$$
$$\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\alpha_{n-k}\rho_n}$$
$$u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \alpha_{n-k}AK^{-1}p_n)$$
$$v_{n+1} = r_{n+1} + \beta_{n+1-k}q_n$$
$$p_{n+1} = u_{n+1} + \beta_{n+1-k}(q_n + \beta_{n+1}p_n)$$

End if
If (BiCGSTAB), then

$$\alpha_n = \rho_n / \tilde{r}_0^T AK^{-1}u_n$$
$$v = r_n - \alpha_n AK^{-1}u_n$$
$$z = AK^{-1}v$$
$$\omega = (K_1^{-1}z)^T K_1^{-1}v / (K_1^{-1}z)^T K_1^{-1}z;$$
$$r_{n+1} = v - \omega z$$
$$x_{n+1} = x_n + K^{-1}(\alpha_n u_n + \omega v)$$

$$\rho_{n+1} = \tilde{r}_0^T r_{n+1};$$
$$\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\omega \rho_n}$$
$$u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \omega A K^{-1} u_n)$$
$$v_{n+1} = (I - \omega A K^{-1})(v_n - \alpha_n A K^{-1} p_n)$$
$$p_{n+1} = v_{n+1} + \beta_{n+1}(p_n - \omega A K^{-1} p_n)$$

End if

End for

We remark that there are two ways for choosing ω in the BiCGSTAB part. The one given above is to minimize the preconditioned residual $K_1^{-1} r_{n+1}$. It can also be chosen as $\omega = z^T v / z^T z$ to minimize the original residual r_{n+1} (the two are the same when $K_1 = I$). See [15].

3 Switching Criterion

In the mixed method, we can switch between BiCGSTAB and CGS at will at every step. Except in some special situations (See Remark 3 in Section 2), however, four matrix-vector multiplications are usually needed at each BiCGSTAB step. Therefore, to be cost effective, it is necessary to minimize the number of switchings. We shall consider CGS based implementations that switch to BiCGSTAB only occasionally.

A main purpose of introducing the mixed method is to improve the stability of CGS (convergence of the computed residual vector). While the behaviour of CGS in finite precision is not well understood, it is generally believed that large increase in the residual is a main factor in causing instability. What is interesting in our numerical testing, however, is that a large local variation in the residual norm, i.e. $\|r_{n+1}\|/\|r_n\|$, seems to have most significant effects in the convergence and the absolute magnitude of the relative residual $\|r_{n+1}\|/\|r_0\|$ has less apparent effect in the convergence, at least for the computed (or called updated) residuals. Specifically, there seems to be very little correlation between convergence of CGS and $\max_n \|r_{n+1}\|/\|r_0\|$. On the other hand, the convergence is usually improved by controlling the local residual increase (see Section 4). To this end, we advocate an implementation that controls the local increase in the residual $\|r_{n+1}\|/\|r_n\|$. Namely, given a tolerance Tol, we compute r_{n+1} by CGS and test the switching criterion

$$\|r_{n+1}\|/\|r_n\| \leq Tol. \tag{12}$$

If this is satisfied (there is no large local increase in the residual norm), a CGS step is taken; otherwise, a BiCGSTAB step will be taken. In order to minimize the extra cost (two extra matrix-vector multiplications) associated with each switching, a too small Tol should be avoided. It is also sensible in this regard not to implement the switching when r_n starts to converge ($\|r_{n+1}\|/\|r_0\| < 0.1$, say). From our tests, a value around $Tol = 10^2$ seems to be sufficient in improving stability of CGS yet leads to only limited switchings to BiCGSTAB.

It should be pointed out that a large growth in the absolute magnitude of the relative residual $\|r_{n+1}\|/\|r_0\|$ could lead to stagnation of the true residual $b - Ax_n$ at certain level even when the computed residual continues to converge. We note that BiCG type recurrences are designed to reduce the computed residual which in turn drives the convergence of the true residual (see [14]). In finite precision, however, when $\|r_{n+1}\|/\|r_0\|$ is large, there will be large differences between r_n

and $b - Ax_n$ at the convergence owing to the error accumulation. In that case, convergence of r_n no longer guarantees the convergence of $b - Ax_n$. However, it is possible to deal with this kind of problem separately through residual updating, see [12]. The present work is focused on convergence of the computed residual r_n, which is the source of convergence of the true residual $b - Ax_n$.

4 Numerical Examples

In this section, we present some numerical examples to demonstrate the stabilizing effect of the mixed algorithm. Specifically, we shall consider the switching criterion discussed in section 3 and compare the mixed method with CGS and BiCGSTAB.

Throughout the examples, Tol used will be 10^2 unless otherwise specified. In the convergence plots below, at the point at which a switching incurs, we use "x" to mark the norm of the CGS step without switching.

Example 1: The matrix is a finite-difference discretization (center difference) on a 40×40 grids of the following convection diffusion equation

$$-\triangle u + \beta u_x + \gamma u_y = f(x,y) \quad \text{on} \quad (0,1)^2;$$

with the homogeneous Dirichlet boundary condition. f is chosen such that the vector of ones is the solution. Figure 1 gives the convergence history of the computed residuals for two sets of parameters (a): $\beta = -200$, $\gamma = 200$ and (b): $\beta = -122$ and $\gamma = 190$. For the mixed method, 14 switchings incurred in (a) and 6 switchings in (b). There is a clear improvement of the mixed method over CGS.

Example 2: The matrix is a finite-difference discretization on a 40×40 grids of the following convection diffusion equation

$$-\triangle u + \alpha(xu_x + yu_y) + \beta u = f(x,y) \quad \text{on} \quad (0,1)^2;$$

with the homogeneous Dirichlet boundary condition. f is a constant. Figure 3 is the convergence history of the computed residuals for two sets of parameters (a): $\alpha = 100$, $\beta = -100$, and (b): $\alpha = 100$, $\beta = -360$. For the mixed method, 3 switchings incurred in (a) and 4 switchings in (b).

Example 3: The matrix is ORSREG1 from the OILGEN group of the Harwell-Boeing collection [3] of sparse test matrices. The order of the matrix is 2205 and it has 14133 nonzero entries. The right-hand side b is chosen to be the vector of ones. We implement it (a) without any preconditioning and (b) with the ILU(0) preconditioning. The results are given in Figure 3. In case (b) with the ILU(0) preconditioning, we choose $Tol = 10$ as the regular $Tol = 10^2$ does not lead to any switching for this relatively easy problem. For the mixed method, 6 switchings incurred in (a) and 1 switching in (b).

In all three examples, we have seen the dramatic stabilizing effect of the mixed method over CGS. In particular, it is capable of turning a divergent CGS into a convergent one with a few switchings. However, this is by no means typical. For most cases, the performance of Example 1 (a) and Example 3 (b) (the preconditioning case) is more typical in that CGS itself converges and the mixed method improves it slightly with a few switchings.

Through our testing, we also found that in some cases the choice of Tol could have significant effect in the performance of the mixed method. It sometimes needs to be fine tuned in order to achieve better balance between stability and the number of switchings. For some problems, the

Figure 1: **Convergence History for Example 1**

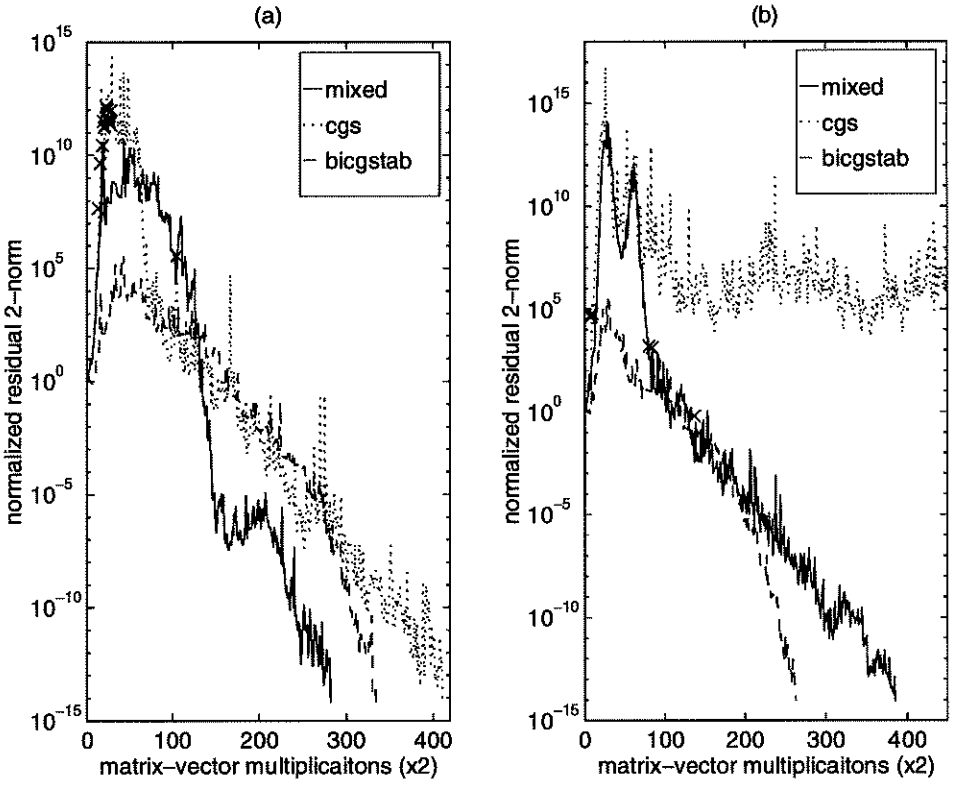
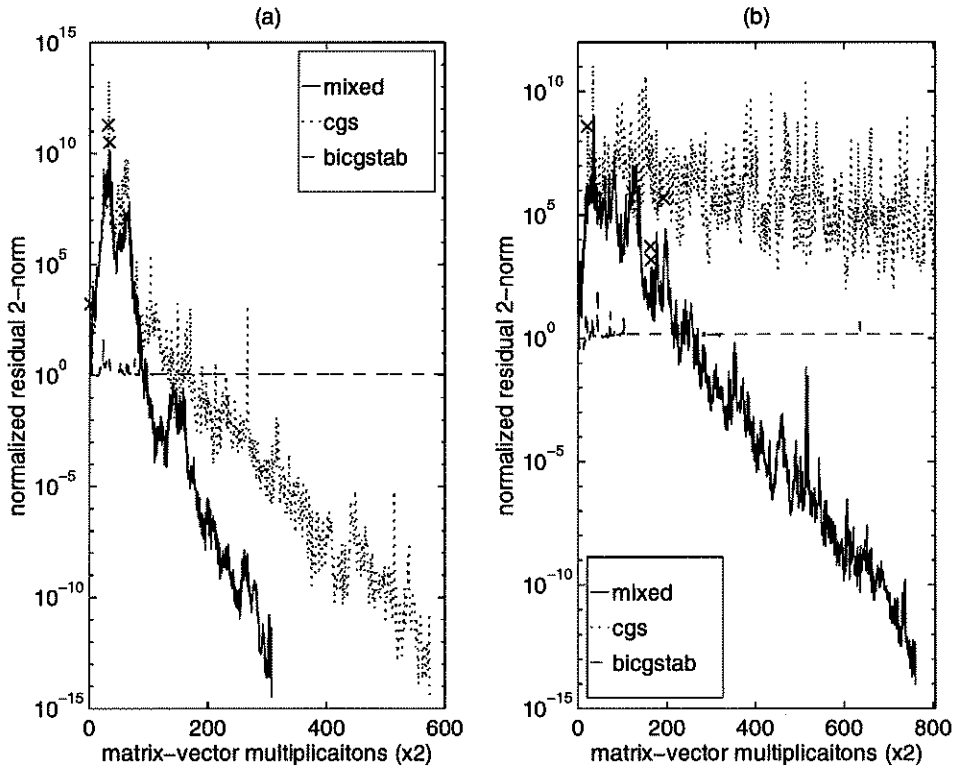
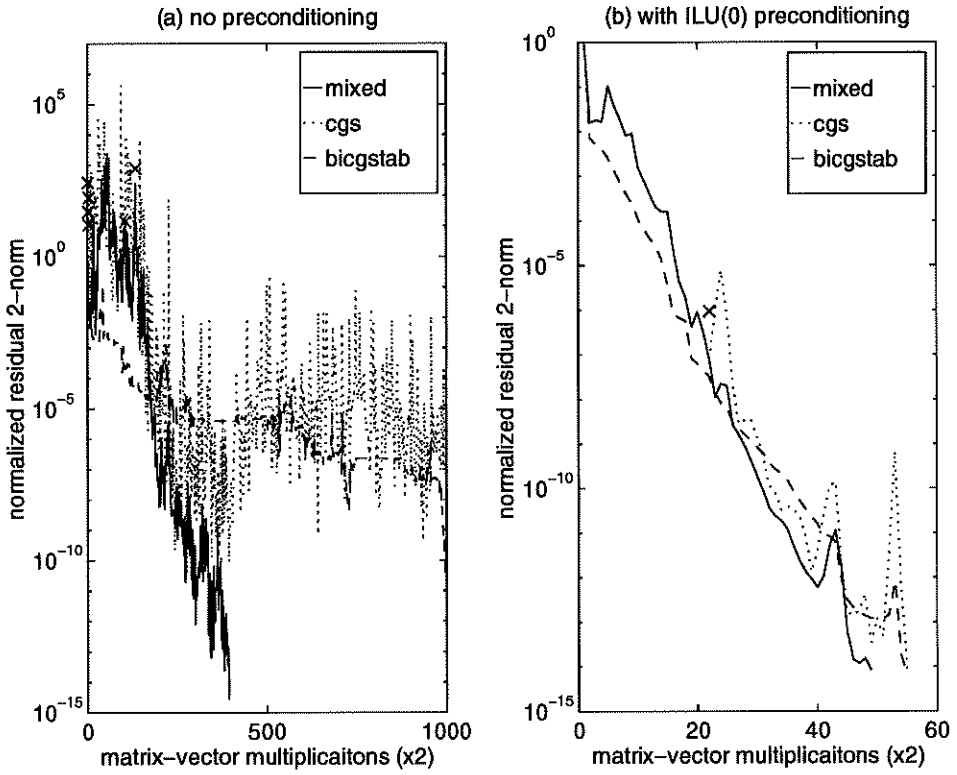


Figure 2: **Convergence History for Example 2**



number of switchings may be too large to compensate any gain in stability. Some of these are due to the limitation of a single cut switching criterion (12). For example, consider a CGS run where $\|r_{n+1}\|/\|r_n\|$ just exceeds Tol. Then, use of (12) will be less efficient in a case it occurs just once than in a case it occurs, say, ten times. We plan to study these issues in the future and carry out more extensive testing with different switching criteria.

Figure 3: **Convergence History for Example 3**



5 Concluding Remarks

We have presented the idea of switching between two different type product methods and demonstrated in principle its potential as a competitive product method. The mixed CGS-BiCGSTAB method is derived and implemented to control local increase in the residual norm. The numerical examples show that this implementation could provide a competitive alternative for the class of problems where CGS is competitive or where both CGS and BiCGSTAB diverge.

We point out that the idea of switching may have applications in other contexts. Its ultimate success will depend on the switching strategy, the best of which is not clear at the moment. The preliminary success of the one used through controlling the local increase may also indicate the relation between large local variation in the residual norm and stability of CGS. However, there is no theoretical results to confirm this. It would still be interesting to carry out some error analysis to determine the precise cause for the possible instability in CGS. A complete understanding in

this regard may lead to a better implementation of the present idea that picks the best of the two methods.

References

[1] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, V. Pozo, Romime C., and H. van der Vorst, *Templates for the solution of linear systems: Building blocks for iterative methods*, SIAM, Philadelphia, PA, 1994.

[2] C. Brezinski and H. Sadok, *Avoiding Breakdown in the CGS Algorithm,* Numerical Algorithms 1 (1991), pp. 199-206.

[3] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse Matrix Test Problems,* ACM Trans. Math. Softw., 15 (1989), pp.1-14.

[4] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems,* in Proc. Dundee Conference on Numerical Analysis, 1975, Lecture Notes in Mathematics 506, G. A. Watson, ed., Springer-Verlag, Berlin, 1976, pp. 73-89.

[5] D.R. Fokkema, G.L.G. Sleijpen and H.A. van der Vorst, *Generalized conjugate gradient squared,* J. Comp. and Appl. Math., 71 , 125-146, 1996.

[6] R. Freund, *A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems* SIAM J. Sci. Stat. Comput. 14, pp. 470-482 (1993).

[7] M. Gutknecht, *Variants of BiCGStab for matrices with complex spectrum,* SIAM J. Sci. Stat. Comput. 14, pp. 1020-1033 (1993).

[8] M. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations,* Acta Numerica 6, pp. 271-397 (1997).

[9] C. Lanczos, *Solution of Systems of Linear Equations by Minimized Iterations,* J. Res. Natl. Bur. Stand. 49, pp. 33-53 (1952).

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

[11] G. Sleijpen and D. Fokkema, *BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum* Electronic Trans. Numer. Anal. 1, pp. 11-32 (1993).

[12] G. Sleijpen and H. van der Vorst, *Reliable updated residuals in hybrid B-CG methods* Preprint 886, Dept. of Math. University Utrecht, 1994.

[13] P. Sonnefeld, *CGS, A Fast Lanczos-type Solver for nonsymmetric Linear Systems* SIAM J. Sci. Stat. Comput. 10, p. 36-52 (1989).

[14] C. H. Tong and Q. Ye, *Analysis of the Finite Precision Bi-Conjugate Gradient algorithm for Nonsymmetric Linear Systems,* Stanford SCCM Report 95-11, Stanford University, Stanford, CA, October 1995.

[15] H. van der Vorst, *BiCGSTAB, A fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems* SIAM J. Sci. Stat. Comput. 13, 631-644 (1992).

[16] S. Zhang, *GPBi-CG: Generalized product-type methods based on BiCG for solving nonsymmetric linear systems* SIAM J. Sci. Stat. Comput. 13, 631-644 (1992).