UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Composite Step Product Methods for Solving Nonsymmetric Linear Systems

Tony F. Chan
Tedd Szeto

July 1994

CAM Report 94-18

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

COMPOSITE STEP PRODUCT METHODS FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS

TONY F. CHAN* AND TEDD SZETO†

Abstract. First proposed in [2, 3] by Bank and Chan, the Composite Step method is a technique for curing the pivot breakdown (one of two possible breakdowns) in the Biconjugate Gradient (BCG) algorithm by skipping over steps in which the iterate is not defined. We show how to extend this method to cure the same breakdown inherited in product methods such as CGS [26], Bi-CGSTAB [28], Bi-CGSTAB2 [18], which are derived from a product of the BCG polynomial with another polynomial of the same degree. New methods introduced in this paper are CS-CGSTAB (composite step applied to Bi-CGSTAB) and a more stable variant of this, CS-CGSTAB2, which handles a possible additional breakdown problem due to the Bi-CGSTAB process in the case where A is skew-symmetric. CS-CGSTAB2 can be viewed as an improvement over the BiCGSTAB(l) algorithm (Sleijpen and Fokkema [25]) with $l = 2$ in that although BiCGSTAB(2) is designed to cure the skew-symmetric breakdown, it does not handle pivot breakdown as CS-CGSTAB2 does. The new methods require only a minor modification to the existing product methods. Moreover, the sizes of the steps taken in our methods can vary as opposed to fixed step methods like BiCGSTAB(l) and BiCGSTAB2. Our strategy for deciding whether to skip a step does not involve any machine dependent parameters and is designed to skip near breakdowns as well as produce smoother iterates. Numerical experiments show that the new methods do produce improved performance over those without composite step on practical problems. Furthermore, we extend the "best approximation" result in [3] to obtain convergence proofs for CGS and Bi-CGSTAB and their composite step stabilized versions.

AMS(MOS) subject classification. 65F10,65F25

Key Words. Lanczos method, Bi-conjugate gradient method, breakdowns, product methods, Bi-CGSTAB, composite step

1. Introduction. The Biconjugate Gradient (BCG) algorithm [21] is the "natural" generalization of the classical Conjugate Gradient method [19] to nonsymmetric linear systems

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

It is an attractive method because of its simplicity and its good practical convergence properties. Unfortunately, one of its drawbacks is that it requires multiplications with the transpose matrix A^T. Methods have been developed which overcome this by computing residuals characterized by a product of the BCG residual polynomial with another polynomial of equal degree. Hence, we term the class of these algorithms *product methods.*

The Conjugate Gradients Squared (CGS) algorithm (Sonneveld, [26]) is a product method whose residuals can be written $r_n^{CGS} = \phi_n^2(A)r_0$, where $\phi_n(A)r_0$ is the residual from the standard BCG method. However, as discussed in [29], the convergence behavior of CGS can sometimes be irregular (i.e., $\|r_i^{CGS}\|$ can vary wildly with i). This is

* Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: chan@math.ucla.edu. Partially supported by the Office of Naval Research grant N00014-92-J-1890, the National Science Foundation grant ASC92-01266, and the Army Research Office grant DAAL03-91-G-150.

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

due to the fact that if $\phi_n(A)$ is viewed as a reduction operator applied twice to r_0, since $\phi_n(A)$ is quite dependent on the initial residual r_0, it is not likely to be a reduction for any other vector, not even for $\phi_n(A)r_0$ itself.

The Bi-CGSTAB algorithm [28] due to Van der Vorst attempts to stabilize this by multiplying the BCG polynomial $\phi_n(A)$, instead, by another polynomial of equal degree,

$$\tau_n(A) = (I - \omega_1 A)(I - \omega_2 A) \cdots (I - \omega_n A), \tag{2}$$

where the ω_i's are chosen to locally minimize the residual by a steepest descent method. Thus, by computing residuals $r_n^{Bi-CGSTAB} = \tau_n(A)\phi_n(A)r_0$, we obtain a more smoothly converging algorithm.

Unfortunately, problems arise in this method if we encounter a matrix A having complex eigenvalues with large imaginary parts. Since τ_n has only real zeros, it is difficult to accurately handle such eigenvalues. Thus, the BiCGSTAB2 algorithm (Gutknecht, [18]) was developed to overcome this by performing two steps of BiCGSTAB at a time to allow for pairs of complex conjugate zeros but doing the local minimization at the n-th step over the two degrees of freedom in ω_n and ω_{n-1}. BiCGSTAB2, then, is also a product method.

In the case that A is (nearly) skew-symmetric, however, BiCGSTAB2 will suffer (near) breakdown due to the τ_{n-1} polynomial. The BiCGSTAB(2) method (a case of the BiCGSTAB(l) algorithm by Sleijpen and Fokkema [25] when $l = 2$) cures this by not involving τ_{n-1} in the intermediate step.

Since all of the product methods mentioned above involve the BCG residual polynomial ϕ_n, they not only inherit the good properties of BCG, but they also take on some of the problems of BCG. Specifically, it is well known that BCG suffers numerical breakdowns (attempts to divide by 0). There are two different possibilities of breakdown in the algorithm and many methods have been designed to cure them by "looking ahead" to avoid computing iterates where a breakdown can be predicted. The spectrum of these methods ranges from simple modifications of BCG to handle only one of the breakdowns to more complex algorithms which provide total breakdown protection using a variable look ahead step. (See e.g., [3, 5, 7, 8, 14, 16, 17, 20, 24].)

In this paper, we consider the *composite step* technique (Bank and Chan [2, 3]) which cures one of the breakdowns (assuming the other one does not occur) by simply looking ahead only one step when a (near) breakdown occurs. This technique is attractive because there is no need for user specified tolerance parameters, no variable step sizes, and requires only a minimal modification of the standard BCG algorithm. Moreover, the composite step technique can easily be extended to product methods. For example, the Composite Step CGS (CSCGS) algorithm was developed in by Chan and Szeto [9] and in this paper, we show how composite step can be applied to Bi-CGSTAB.

In [3], Bank and Chan also prove a "best approximation" result which establishes a bound on the error of BCG. Here, we extend this result to prove convergence results for CGS, Bi-CGSTAB, and their composite step variants since these product methods all involve the BCG polynomial ϕ_n.

Section 2 describes in detail the composite step idea as originally presented for BCG and in Section 3, this is extended to Bi-CGSTAB yielding the method CS-CGSTAB. We emphasize in this section the strategy used to decide when to take a look ahead composite step. We note that this stepping strategy not only skips exact breakdowns, but is also designed to yield smoother residuals and does not require any user specified tolerance parameters. A variant of this method, CS-CGSTAB2, which is a way of combining composite step with an idea from BiCGSTAB2, is presented in Section 4. The purpose of this variant is to cure additional breakdowns that may occur due to instability in the BiCGSTAB steepest descent polynomial in a manner similar to BiCGSTAB(l) with $l = 2$. The advantage is that our method cures pivot breakdowns as well. In addition, the step sizes in our new methods can vary as opposed to steps of fixed size in BiCGSTAB2 and BiCGSTAB(l). In Section 5, we show some numerical examples which compare these methods and show the advantages over their non-composite step counterparts. Finally, Section 6 details the proofs for convergence of CGS, Bi-CGSTAB, and their composite step variants,

2. The Composite Step Idea: CSBCG. It is well known that the Biconjugate Gradient method [19] is closely related to the nonsymmetric Lanczos process for computing the basis for the Krylov subspaces $K_n(r_0)$ and $K_n^*(\tilde{r}_0)$, defined as follows:

$$K_n(r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}; \tag{3}$$

$$K_n^*(\tilde{r}_0) \equiv \text{span}\{\tilde{r}_0, A^T\tilde{r}_0, \ldots, (A^T)^{n-1}\tilde{r}_0\}. \tag{4}$$

The BCG iterates are defined by a Galerkin method on the associated Krylov subspaces. Given initial guesses of x_0 and $\tilde{x}_0$ to the solutions of (1) and an auxiliary system, $A^T\tilde{x} = \tilde{b}$, BCG produces iterates

$$x_n = x_0 + y_n; \qquad \tilde{x}_n = \tilde{x}_0 + \tilde{y}_n,$$

with corresponding residuals of the form

$$r_n = b - Ax_n; \qquad \tilde{r}_n = \tilde{b} - A^T\tilde{x}_n,$$

where $y_n \in K_n(r_0)$ and $\tilde{y}_n \in K_n^*(\tilde{r}_0)$, and such that the following Galerkin conditions are satisfied:

$$r_n \perp K_n^*(\tilde{r}_0); \qquad \tilde{r}_n \perp K_n(r_0). \tag{5}$$

If we define $\bar{K}_n, \bar{K}_n^*$ to be matrices whose columns are as given in (3) and (4) and span the Krylov spaces $K_n(r_0)$ and $K_n^*(\tilde{r}_0)$, respectively, condition (5) implies that the BCG iterates $x_n = x_0 + \bar{K}_n v_n$ are defined by the solution to the linear system $(\bar{K}_n^*)^T A\bar{K}_n v_n = (\bar{K}_n^*)^T r_0$. We note that the iterate x_n exists whenever the Hankel moment matrix

$$H_n^{(1)} = (\bar{K}_n^*)^T A\bar{K}_n = \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_n \\ \mu_2 & \mu_3 & \cdots & \mu_{n+1} \\ \vdots & \vdots & & \vdots \\ \mu_n & \mu_{n+1} & \cdots & \mu_{2n-1} \end{pmatrix},$$

where $\mu_i \equiv \tilde{r}_0^T A^i r_0$, is nonsingular.

One standard way to compute the BCG iterates is as follows [21, 13]:

Algorithm BCG

$$
\begin{array}{lll}
\text{Set} & r_0 = b - Ax_0; & \tilde{r}_0 = \tilde{b} - A^T\tilde{x}_0 \\
 & p_0 = r_0; & \tilde{p}_0 = \tilde{r}_0 \\
 & \rho_0 = \tilde{r}_0^T r_0 & \\
\multicolumn{3}{l}{\text{For } n = 0, 1, \ldots} \\
 & \sigma_n = \tilde{p}_n^T A p_n; & \alpha_n = \rho_n/\sigma_n \\
 & r_{n+1} = r_n - \alpha_n A p_n; & \tilde{r}_{n+1} = \tilde{r}_n - \alpha_n A^T \tilde{p}_n \\
 & x_{n+1} = x_n + \alpha_n p_n; & \tilde{x}_{n+1} = \tilde{x}_n + \alpha_n \tilde{p}_n \\
 & \rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1}; & \beta_{n+1} = \rho_{n+1}/\rho_n \\
 & p_{n+1} = r_{n+1} + \beta_{n+1} p_n; & \tilde{p}_{n+1} = \tilde{r}_{n+1} + \beta_{n+1}\tilde{p}_n \\
\text{End} & &
\end{array}
$$

We can see that there are two possible kinds of numerical breakdowns (attempts to divide by 0) in the above routine: (1) $\sigma_n = 0$ *(pivot breakdown)*, and (2) $\rho_n = 0$, but $r_n \neq 0$ *(Lanczos breakdown)*. [1] Although such exact breakdowns are very rare in practice, near breakdowns can cause severe numerical instability.

We term the first kind of breakdown a *pivot breakdown* because it is due to the non-existence of the residual polynomial implicitly caused by encountering a zero pivot in the factorization of the tridiagonal matrix generated in the underlying Lanczos process. In terms of formally orthogonal polynomials [5], the BCG polynomial ϕ_n (defined from $r_n = \phi_n(A)r_0$) exists and is unique if and only if the $H_n^{(1)}$ is nonsingular. In other words, a pivot breakdown will occur at the n-th iteration of the BCG algorithm if $det(H_n^{(1)}) = 0$.

The second source of breakdown, *Lanczos breakdown*, is directly related to the breakdown of the underlying Lanczos process, and is tied to the singularity of another Hankel moment matrix $H_n^{(0)}$ [17] defined by:

$$
H_n^{(0)} \equiv \bar{K}_n^*(\tilde{r}_0)\bar{K}_n(r_0) = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & \cdots & \mu_n \\ \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-2} \end{pmatrix}.
$$

There are many methods designed to handle pivot breakdowns (see e.g., [3, 14, 17]), as well as methods which cure both types of breakdown (see e.g., [5, 7, 8, 14, 16, 20, 24]). Although the step size needed to overcome an exact Lanczos breakdown can be computed in principle, these methods can unfortunately be quite complicated for handling near breakdowns since the sizes of the look-ahead steps are variable (indeed, the breakdowns can be *incurable*).

The Composite Step Biconjugate Gradient (CSBCG) algorithm (Bank and Chan, [2, 3]) is an alternative which cures only the pivot breakdown (assuming no Lanczos breakdowns) by "looking ahead" in $H_n^{(1)}$ and not computing x_n where it is not defined. Looking ahead means that we build $H_i^{(1)}$ until it is no longer singular and we have an iterate $x_{n+m}, m \geq 1$. In CSBCG, this is done with a simple modification of BCG

[1] In other literature, what we term the *pivot* and *Lanczos* breakdowns, are also known as *true* and *ghost* breakdowns [6], *Galerkin* and *serious Lanczos* breakdowns [14], *hard* and *soft* breakdowns [20].

which needs only a maximum look-ahead step size of $m = 2$ to eliminate the (near) breakdown and to smooth the sometimes erratic convergence of BCG. It was shown in [2, Lemma 4.3] that only two steps are needed if we assume no Lanczos breakdown, but this can also be seen in the relationship between the two Hankel matrices defined above: assuming that $det(H_n^{(0)}) \neq 0$ for all n, then no two consecutive principal submatrices of $H_n^{(1)}$ can be singular. (The structure of these Hankel determinants was studied in detail by Draux [10].) Thus, instead of a more complicated (but less prone to breakdown) version, CSBCG cures only one kind of breakdown, but does so with a minimal modification to the usual implementation of BCG in the hope that its empirically observed stability will be inherited [27].

We shall next briefly review some of the details of CSBCG. Suppose in running the BCG algorithm, we encounter a situation where $\sigma_n = 0$ at step n, and therefore, the values $x_{n+1}, \tilde{x}_{n+1}, r_{n+1}, \tilde{r}_{n+1}$ cannot be defined. The composite step approach is to overcome this problem by skipping the $n+1$ update and computing the quantities in step $n+2$ by using scaled versions of r_{n+1} and $\tilde{r}_{n+1}$, which do not require divisions by σ_n. More specifically, we define the auxiliary vectors:

$$\begin{aligned} z_{n+1} &\equiv \sigma_n r_{n+1} = \sigma_n r_n - \rho_n A p_n &\in\quad & K_{n+2}(r_0); \\ \tilde{z}_{n+1} &\equiv \sigma_n \tilde{r}_{n+1} = \sigma_n \tilde{r}_n - \rho_n A^T \tilde{p}_n &\in\quad & K^*_{n+2}(\tilde{r}_0). \end{aligned} \tag{6}$$

These exist even if $\sigma_n = 0$ and thus, can be used in defining x_{n+2} :

$$x_{n+2} = x_n + [p_n, z_{n+1}] f_n,$$

with corresponding residual and search direction

$$r_{n+2} \;=\; r_n - A[p_n, z_{n+1}] f_n; \tag{7}$$

$$p_{n+2} \;=\; r_{n+2} + [p_n, z_{n+1}] g_n, \tag{8}$$

where $f_n, g_n \in R^2$, and similarly for $\tilde{x}_{n+2}, \tilde{r}_{n+2}, \tilde{p}_{n+2}, \tilde{f}_n$, and $\tilde{g}_n$.

To solve for the unknowns $f_n = \left(f_n^{(1)}, f_n^{(2)}\right)^T$ and $g_n = \left(g_n^{(1)}, g_n^{(2)}\right)^T$, we impose the Galerkin condition and conjugacy condition of BCG which result in two 2×2 linear systems:

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} f_n^{(1)} \\ f_n^{(2)} \end{bmatrix} = \begin{bmatrix} \tilde{p}_n^T r_n \\ \tilde{z}_{n+1}^T r_n \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = -\begin{bmatrix} \tilde{p}_n^T A r_{n+2} \\ \tilde{z}_{n+1}^T A r_{n+2} \end{bmatrix}. \tag{10}$$

This yields (after some algebra) the quantities:

$$f_n = (\zeta_{n+1}\rho_n, \theta_{n+1})\rho_n^2/\delta_n \text{ and } g_n = (\rho_{n+2}/\rho_n, \sigma_n\rho_{n+2}/\theta_{n+1}),$$

where $\zeta_{n+1} = \tilde{z}_{n+1}^T A z_{n+1}, \theta_{n+1} = \tilde{z}_{n+1}^T z_{n+1}, \delta_n = \sigma_n \zeta_{n+1} \rho_n^2 - \theta_{n+1}^2$. Furthermore, a lemma in [3, Lemma 5.1] shows $\tilde{f}_n = f_n$ and $\tilde{g}_n = g_n$. It is now possible to compute $x_{n+2}, \tilde{x}_{n+2}, r_{n+2}, \tilde{r}_{n+2}$ and thus, advance from step n to step $n+2$. The Composite Step BCG algorithm, then, is simply the combination of the 1×1 and 2×2 steps.

3. Composite Step Bi-CGSTAB. We now apply the composite step idea to handle the pivot breakdowns that occur in the Bi-CGSTAB method proposed by Van der Vorst [28]. Since the Bi-CGSTAB residual polynomials are formed from multiplying the BCG polynomial ϕ_n with another polynomial τ_n of the form (2), we can use the subset of well-defined ϕ_i's from CSBCG to multiply with τ_n instead. To do this, we first define

$$p_n^{BCG} = \phi_n(A)r_0; \quad z_{n+1}^{BCG} = \xi_{n+1}(A)r_0.$$

The CS-CGSTAB polynomials, then, take on the form:

$$r_n^{CS-CGSTAB} = \tau_n(A)\phi_n(A)r_0; \quad p_n^{CS-CGSTAB} = \tau_n(A)\psi_n(A)r_0.$$

In the case of a 1×1 step, we simply use the Bi-CGSTAB update. For the 2×2 composite step, we will need to evaluate

$$\begin{aligned} r_{n+2}^{CS-CGSTAB} &= \tau_{n+2}\phi_{n+2}r_0 \\ &= (I - \omega_{n+2}A)(I - \omega_{n+1}A)\tau_n\phi_{n+2}r_0 \end{aligned} \tag{11}$$

and

$$\begin{aligned} p_{n+2}^{CS-CGSTAB} &= \tau_{n+2}\psi_{n+2}r_0 \\ &= (I - \omega_{n+2}A)(I - \omega_{n+1}A)\tau_n\psi_{n+2}r_0 \end{aligned} \tag{12}$$

using the quantities obtained from the n-th step: $\tau_n\phi_n$, and $\tau_n\psi_n$. We first show how to compute the polynomials $\tau_n\phi_{n+2}$ and $\tau_n\psi_{n+2}$ appearing in (11) and (12). We use the CSBCG relationships from (6) - (8):

$$\xi_{n+1}(\vartheta) = \sigma_n\phi_n - \rho_n\vartheta\psi_n, \tag{13}$$

$$\phi_{n+2}(\vartheta) = \phi_n - \vartheta\psi_n f_n^{(1)} - \vartheta\xi_{n+1}f_n^{(2)}, \tag{14}$$

$$\psi_{n+2}(\vartheta) = \phi_{n+2} + \psi_n g_n^{(1)} + \xi_{n+1}g_n^{(2)} \tag{15}$$

and multiply by the τ_n polynomial to evaluate the needed quantities:

$$\tau_n(\vartheta)\xi_{n+1}(\vartheta) = \sigma_n\tau_n\phi_n - \rho_n\vartheta\tau_n\psi_n \tag{16}$$

$$\begin{aligned} \tau_n(\vartheta)\phi_{n+2}(\vartheta) &= \tau_n(\phi_n - \vartheta\psi_n f_n^{(1)} - \vartheta\xi_{n+1}f_n^{(2)}), \\ &= \tau_n\phi_n - \vartheta\tau_n\psi_n f_n^{(1)} - \vartheta\tau_n\xi_{n+1}f_n^{(2)}, \end{aligned} \tag{17}$$

$$\begin{aligned} \tau_n(\vartheta)\psi_{n+2}(\vartheta) &= \tau_n(\phi_{n+2} + \psi_n g_n^{(1)} + \xi_{n+1}g_n^{(2)}) \\ &= \tau_n\phi_{n+2} + \tau_n\psi_n g_n^{(1)} + \tau_n\xi_{n+1}g_n^{(2)}. \end{aligned} \tag{18}$$

We now show how to compute the unknowns f_n and g_n in (17) and (18). The CSBCG residual r_{n+2} in (7) and search direction p_{n+2} in (8) are, respectively, orthogonal and A-conjugate to $K_{n+1}^*(\tilde{r}_0)$ [2]. By imposing orthogonality and A-conjugacy conditions on two specific vectors $\tau_n(A^T)\tilde{r}_0 \in K_{n+1}^*(\tilde{r}_0)$ and $A^T\tau_n(A^T)\tilde{r}_0 \in K_{n+1}^*(\tilde{r}_0)$, we obtain two linear systems which give f_n and g_n.

Specifically, by writing equation (7) in polynomial form (14) and taking an inner product with $\tau_n(A^T)\tilde{r}_0$, we obtain the relation:

$$\begin{aligned} 0 &= (\tau_n(A^T)\tilde{r}_0, \phi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, \tau_n(A)\phi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, [\tau_n\phi_n - A\tau_n\psi_n f_n^{(1)} - A\tau_n\xi_{n+1}f_n^{(2)}](A)r_0). \end{aligned}$$

Similarly, for the A-conjugacy of the search direction (8),

$$\begin{aligned}0 &= (A^T\tau_n(A^T)\tilde{r}_0, \psi_{n+2}r_0) \\ &= (\tilde{r}_0, A\tau_n(A)\psi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, [A\tau_n\phi_{n+2} + A\tau_n\psi_n g_n^{(1)} + A\tau_n\xi_{n+1}g_n^{(2)}](A)r_0).\end{aligned}$$

We derive two similar relations by imposing the orthogonality and A-conjugacy conditions on $A^T\tau_n(A^T)\tilde{r}_0$. Combining the four relations, we obtain the following two 2×2 linear systems:

$$\begin{bmatrix}(\tilde{r}_0, \vartheta\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta\tau_n\xi_{n+1}r_0) \\ (\tilde{r}_0, \vartheta^2\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta^2\tau_n\xi_{n+1}r_0)\end{bmatrix}\begin{bmatrix}f_n^{(1)} \\ f_n^{(2)}\end{bmatrix} = \begin{bmatrix}(\tilde{r}_0, \tau_n\phi_n r_0) \\ (\tilde{r}_0, \vartheta\tau_n\phi_n r_0)\end{bmatrix} \tag{19}$$

$$\begin{bmatrix}(\tilde{r}_0, \vartheta\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta\tau_n\xi_{n+1}r_0) \\ (\tilde{r}_0, \vartheta^2\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta^2\tau_n\xi_{n+1}r_0)\end{bmatrix}\begin{bmatrix}g_n^{(1)} \\ g_n^{(2)}\end{bmatrix} = \begin{bmatrix}(\tilde{r}_0, \vartheta\tau_n\phi_{n+2}) \\ (\tilde{r}_0, \vartheta^2\tau_n\phi_{n+2})\end{bmatrix}. \tag{20}$$

These are easily solved since all of the entries in the 2×2 matrix and the right hand sides can be obtained from equations (16), (17), and quantities from the n-th step. Thus, we can update (17) and (18).

The next step in evaluating (11) and (12) is to choose ω_{n+1} and ω_{n+2} to satisfy some local minimization property. In the case of a 1×1 step, we imitate the Bi-CGSTAB update steps. Specifically, ω_{n+1} is chosen to minimize the norm of $r_{n+1} = \tau_{n+1}\phi_{n+1}r_0 = (I-\omega_{n+1}A)\tau_n\phi_{n+1}r_0$. For the 2×2 step, we employ the same steepest descent rule to compute ω_{n+1} and ω_{n+2} by minimizing $\|r_{n+1}\|$ and $\|r_{n+2}\|$. Note that r_{n+1} is not available in a 2×2 step, but we can use a scaled version for minimization. To do this, let $u_{n+1} = \tau_n\xi_{n+1}r_0 \equiv \frac{1}{\sigma_n}\,\tau_n\phi_{n+1}r_0$. Then we can write

$$r_{n+1} = \tau_{n+1}\phi_{n+1}r_0 = (I-\omega_{n+1}A)\tau_n\phi_{n+1}r_0 = (I-\omega_{n+1}A)\frac{1}{\sigma_n}\,u_{n+1}. \tag{21}$$

The vector u_{n+1} is already computed in the CS-CGSTAB algorithm (see relation (16)) and thus, we minimize

$$\|r_{n+1}^2\| = \frac{1}{\sigma_n^2}\,((I-\omega_{n+1}A)u_{n+1})^T\,((I-\omega_{n+1}A)u_{n+1})$$

by choosing $\omega_{n+1} = (Au_{n+1}, u_{n+1})/(Au_{n+1}, Au_{n+1})$, an orthogonal projection of u_{n+1} onto Au_{n+1}.

Similarly, let

$$u_{n+2} \equiv \tau_{n+1}\phi_{n+2}r_0 = (I-\omega_{n+1}A)\tau_n\phi_{n+2}r_0 \tag{22}$$

which can be computed because $\tau_n\phi_{n+2}$ is available from relation (17). Then write

$$r_{n+2} = \tau_{n+2}\phi_{n+2}r_0 = (I-\omega_{n+2}A)\tau_{n+1}\phi_{n+2}r_0 = (I-\omega_{n+2}A)u_{n+2} \tag{23}$$

and minimize

$$\|r_{n+2}^2\| = ((I-\omega_{n+2}A)u_{n+2})^T\,((I-\omega_{n+2}A)u_{n+2})$$

by choosing

$$\omega_{n+2} = (Au_{n+2}, u_{n+2})/(Au_{n+2}, Au_{n+2}). \tag{24}$$

Finally, in running CS-CGSTAB, we must be able to recover the BCG constants ρ_n^{BCG} and σ_n^{BCG} in order to update the BCG polynomial part of the residual. In [28], Van der Vorst defined the Bi-CGSTAB constant

$$\rho_n^{Bi-CGSTAB} = (\tilde{r}_0, r_n^{Bi-CGSTAB})$$

and showed its relation to

$$\rho_n^{BCG} = (\tilde{r}_n^{BCG}, r_n^{BCG}).$$

Using the property that by construction, $\phi_n(A)r_0$ is orthogonal with respect to all vectors $\chi_{n-1}(A^T)\tilde{r}_0$ where χ_{n-1} is an arbitrary polynomial of degree at most $n-1$, one gets:

$$\begin{aligned}
\rho_n^{BCG} &= (\phi_n(A^T)\tilde{r}_0, \phi_n(A)r_0) \\
&= \alpha_0 \cdots \alpha_{n-1}((-A^T)^n \tilde{r}_0, \phi_n(A)r_0), \\
\rho_n^{Bi-CGSTAB} &= (\tilde{r}_0, \tau_n(A)\phi_n(A)r_0) \\
&= (\tau_n(A^T)\tilde{r}_0, \phi_n(A)r_0) \\
&= \omega_1 \cdots \omega_n((-A^T)^n \tilde{r}_0, \phi_n(A)r_0).
\end{aligned}$$

The relationship between them follows:

$$\rho_{n+1}^{BCG} = \frac{\alpha_0 \cdots \alpha_n}{\omega_1 \cdots \omega_{n+1}}\, \rho_{n+1}^{Bi-CGSTAB}, \text{ where } \alpha_n = \frac{\rho_n^{BCG}}{\sigma_n^{BCG}}\ .$$

If we let $\mu_n = (\alpha_0 \cdots \alpha_{n-1})/(\omega_1 \cdots \omega_n)$, this reduces to the update formula:

$$\rho_{n+1}^{BCG} = \mu_{n+1}\rho_{n+1}^{Bi-CGSTAB} = \mu_n \left(\frac{\rho_n^{BCG}}{\sigma_n^{BCG}\omega_{n+1}} \right) \rho_{n+1}^{Bi-CGSTAB}.$$

In a 2×2 step, we determine ρ_{n+2}^{CSBCG} similarly. First we use relation (14):

$$\begin{aligned}
\rho_{n+2}^{CSBCG} &= (\phi_{n+2}(A^T)\tilde{r}_0, \phi_{n+2}(A)r_0) \\
&= ((\phi_n(A^T) - \alpha_n A^T \psi_n(A^T) - \alpha_{n+1} A^T \xi_{n+1}(A^T))\tilde{r}_0, \phi_{n+2}(A)r_0).
\end{aligned} \tag{25}$$

Then, the fact that $\phi_{n+2}r_0$ is orthogonal to all vectors $\chi_{n+1}(A^T)\tilde{r}_0$ implies that the inner product (25) picks out the coefficient of the highest order term of the $n+2$ degree polynomial that ϕ_{n+2} is being orthogonalized against. In this case, the coefficient of the highest order term for the polynomial $\xi_{n+1}(A^T) = \sigma_n^{BCG}\phi_{n+1}(A^T)$ is $\sigma_n^{BCG}\alpha_0 \cdots \alpha_n$, so (25) reduces to:

$$\rho_{n+2}^{CSBCG} = -\alpha_0 \cdots \alpha_{n+1}\sigma_n^{BCG}((-A^T)^{n+2}\tilde{r}_0, \phi_{n+2}(A)r_0)$$

which leads to the update formula:

$$\rho_{n+2}^{CSBCG} = \mu_{n+2}\rho_{n+2}^{CS-CGSTAB} = -\mu_n \left(\frac{\rho_n^{CSBCG}\alpha_{n+1}}{\omega_{n+1}\omega_{n+2}} \right) \rho_{n+2}^{CS-CGSTAB}. \tag{26}$$

The update for $\sigma_n^{BCG} \equiv (\psi_n(A^T)\tilde{r}_0, A\psi_n(A)r_0)$ can be calculated similarly. We define $\sigma_n^{CS-CGSTAB} \equiv (\tilde{r}_0, A\tau_n(A)\psi_n(A)r_0) = (\tilde{r}_0, Ap_n^{CS-CGSTAB})$ and obtain the analogous update formula:

$$\sigma_{n+2}^{CSBCG} = \mu_{n+2}\sigma_{n+2}^{CS-CGSTAB}. \tag{27}$$

TABLE 1
Notation for CS-CGSTAB

vector	polynomial	where/how derived	where used
r_n	$\tau_n \phi_n r_0$	(11)	(16) (17) (19)
p_n	$\tau_n \psi_n r_0$	(12)	(18)
u_{n+1}	$\tau_n \xi_{n+1} r_0$	(16)	(17) (18) (21)
s_{n+2}	$\tau_n \phi_{n+2} r_0$	(17)	(11) (18) (22)
u_{n+2}	$\tau_{n+1} \phi_{n+2} r_0$	(22)	(23)
e_n	$A\tau_n \phi_n r_0$	Ar_n	(19)
q_n	$A\tau_n \psi_n r_0$	Ap_n	(16) (17) (19) (20)
y_{n+1}	$A\tau_n \xi_{n+1} r_0$	Au_{n+1}	(19) (20) (21)
t_{n+2}	$A\tau_n \phi_{n+2} r_0$	As_{n+2}	(20) (22)
y_{n+2}	$A\tau_{n+1} \phi_{n+2} r_0$	Au_{n+2}	(23)
c_n	$A^2\tau_n \psi_n r_0$	Aq_n	(19) (20)
d_{n+1}	$A^2\tau_n \xi_{n+1} r_0$	Ay_{n+1}	(19) (20)
v_{n+2}	$A^2\tau_n \phi_{n+2} r_0$	At_{n+2}	(20)
α_n	$f_n^{(1)}$	(19)	(17)
α_{n+1}	$f_n^{(2)}$	(19)	(17)
β_n	$g_n^{(1)}$	(20)	(18)
β_{n_1}	$g_n^{(2)}$	(20)	(18)

3.1. Implementation Details. In Table 1, we summarize the notation we will be using in our implementation of the CS-CGSTAB algorithm. We list the vector used in the algorithm, its corresponding polynomial form, and the equations in which it is derived and used.

At every step, we must anticipate a 2×2 step even if we decide to take a 1×1 step. (The stepping strategy will be discussed in Section 3.2.) Recall that in one step of Bi-CGSTAB, only two matrix-vector multiplications are performed: $q_n = Ap_n$ and $y_{n+1} = Au_{n+1}$. From the third column of Table 1, we see that 8 matrix-vector products are required to do a 2×2 step which appears to be 4 more than 2 steps of Bi-CGSTAB.

However, the total 8 products can be reduced to 4 multiplications by precomputing certain values and absorbing them into vector updates rather than explicitly multiplying by A. Specifically, by precomputing $c_n \equiv Aq_n$, the multiplication $y_{n+1} \equiv Au_{n+1}$ can be written:

$$\begin{aligned} y_{n+1} &= \sigma_n e_n - \rho_n A q_n \\ &= \sigma_n e_n - \rho_n c_n. \end{aligned}$$

Similarly, if we have $d_{n+1} \equiv Ay_{n+1}$, then the product $e_{n+1} \equiv Ar_{n+1}$ can also be evaluated without having to multiply by A:

$$\begin{aligned} e_{n+1} &= (y_{n+1} - \omega_{n+1} A y_{n+1})/\sigma_n \\ &= (y_{n+1} - \omega_{n+1} d_{n+1})/\sigma_n. \end{aligned}$$

Furthermore, $q_{n+1} \equiv Ap_{n+1}$ can also be updated:

$$\begin{aligned} q_{n+1} &= Ar_{n+1} + \beta_{n+1}(Ap_n - \omega_{n+1} A q_n) \\ &= e_{n+1} + \beta_{n+1}(q_n - \omega_{n+1} c_n). \end{aligned}$$

Thus, the 1×1 step can still be performed with only 2 (pre)multiplications with A: Aq_n and Ay_{n+1}. Moreover, by precomputing $v_{n+2} \equiv At_{n+2}$ and $q_{n+2} \equiv Ap_{n+2}$, we can update the remaining values in the 2×2 step in a similar fashion.

However, using this precomputing strategy to update

$$e_{n+2} \equiv Ar_{n+2} = A(I - \omega_{n+1}A)(I - \omega_{n+2}A)s_{n+2}$$

implies that we need $A^3 s_{n+2}$ which we do not have. Thus, the 2×2 step requires an additional matrix-vector multiplication: $w_{n+2} \equiv A^3 s_{n+2} = A(v_{n+2})$.

Hence, as in CSCGS, there are 5 matrix-vector multiplications for a 2×2 step, whereas in two steps of Bi-CGSTAB, only 4 are needed. This is the price we must pay for composite step. If we do not need to take many 2×2 steps in practice, this price will not be too costly.

3.2. CS-CGSTAB Stepping Strategy. As far as deciding when to actually take a 2×2 step, we follow the principles in [2, 3, 9]. As with these methods, CS-CGSTAB employs a practical stepping strategy that will skip over exact breakdowns using the criterion:

$$\|r_{n+1}\| > \max\{\|r_n\|, \|r_{n+2}\|\}. \tag{28}$$

If this condition were met (e.g., at a near breakdown) and we performed two 1×1 steps, it would result in a "peak" in the residual convergence. By taking a 2×2 step, we skip over this and obtain a smoother, more stable method. In order to avoid unnecessary computation of $\|r_{n+2}\|$, we express condition (28) in the following algorithm:

```
If (||r_{n+1}|| < ||r_n||) then          <--- Condition (28a)
   choose 1 x 1 step
else
   if (||r_{n+1}|| < ||r_{n+2}||) then   <--- Condition (28b)
      choose 1 x 1 step
   else
      choose 2 x 2 step
   end
end
```

In order to avoid repeating work and to do this in a stable way, Condition (28a) can be written:

$$\|(I - \omega_{n+1}A)u_{n+1}\| < |\sigma_n| \|r_n\|.$$

For Condition (28b), we first rescale the r_{n+2} update in order to estimate $\|r_{n+2}\|$ stably by letting

$$\nu_{n+2} \equiv \delta_n r_{n+2} = \delta_n (I - \omega_{n+1}A)(I - \omega_{n+2}A)s_{n+2},$$

where δ_n is the determinant of the 2×2 matrix in (19). Evaluating ν_{n+2} exactly would involve the quantity $A^2 s_{n+2}$ which would require an additional matrix-vector multiplication if we decide to take a 1×1 step. Hence, for practical purposes, we use an upper bound approximation to estimate $\|\nu_{n+2}\|$. Note that although we do not have $A^2 s_{n+2}$, the quantity As_{n+2} can be made available even if we take a 1×1 step, and can be done without an additional matrix-vector product using the precomputed values e_n, c_n, and d_{n+1}:

$$
\begin{aligned}
t_{n+2} &\equiv As_{n+2} \\
&= A(\delta_n r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}) \\
&= \delta_n e_n - \alpha_n c_n - \alpha_{n+1} d_{n+1}.
\end{aligned} \tag{29}
$$

Thus, we would like to estimate $\|\nu_{n+2}\| = \|\delta_n(I - \omega_{n+1}A)(I - \omega_{n+2}A)s_{n+2}\|$ using (29) and without any further matrix-vector multiplications. Our strategy is to set $\omega_{n+2} = 0$ and minimize $\|\delta_n(I - \omega_{n+1}A)s_{n+2}\|$, thereby eliminating the need to compute $A^2 s_{n+2}$. Doing this gives an upper bound estimate to $\|\nu_{n+2}\|$ which can be shown by defining

$$h(\omega_{n+1}, \omega_{n+2}) \equiv \|\delta_n(I - \omega_{n+1}A)(I - \omega_{n+2}A)s_{n+2}\|,$$

and establishing the following inequality:

$$\min_{\omega_{n+1}, \omega_{n+2}} h(\omega_{n+1}, \omega_{n+2}) \le \min_{\omega_{n+1}} h(\omega_{n+1}, 0). \tag{30}$$

The minimization problem:

$$\min_{\omega_{n+1}} h(\omega_{n+1}, 0) = \min_{\omega_{n+1}} \|(I - \omega_{n+1}A)s_{n+2}\| = \min_{\omega_{n+1}} \|s_{n+2} - \omega_{n+1} t_{n+2}\|$$

is solved with $\tilde{\omega}_{n+1} = (t_{n+2}, s_{n+2})/(t_{n+2}, t_{n+2})$.

Hence, Condition (28b): $\|r_{n+1}\| < \|r_{n+2}\|$ is evaluated by the approximated condition

$$|\delta_n|\|(I - \omega_{n+1}A)u_{n+1}\| < |\sigma_n|\|\tilde{\nu}_{n+2}\|, \tag{31}$$

where $\|\tilde{\nu}_{n+2}\|$ is the estimated upper bound for $\|\nu_{n+2}/\delta_n\|$:

$$\|\nu_{n+2}/\delta_n\| < \|\tilde{\nu}_{n+2}\| \equiv \|s_{n+2} - \tilde{\omega}_{n+1} t_{n+2}\|. \tag{32}$$

If a 2×2 step is chosen, then the true ν_{n+2} is evaluated. After ν_{n+2} is computed, we add one final check to make sure the approximation was indeed valid. If not, we take a 1×1 step. In Table 2, we present the CS-CGSTAB algorithm. Note that if only 1×1 steps are taken, we have exactly the Bi-CGSTAB algorithm.

4. A Variant of CS-CGSTAB. A problem in the CS-CGSTAB method is that additional breakdowns may occur if A is skew-symmetric or near breakdowns if it has complex eigenvalues with large imaginary parts. Suppose we are at step n of the algorithm. In the case that A is skew-symmetric, it can be easily checked that $\omega_{n+1} = \omega_{n+2} = 0$. The 2×2 step attempts to divide by the quantity $\gamma_2 = \omega_{n+1}\omega_{n+2}$ which causes a breakdown. For nearly skew-symmetric A, γ_2 will be small, thus causing near breakdown and numerical instability.

This can be cured by modifying the local minimization at that particular 2×2 step. The idea is to require

$$\|r_{n+2}\| = \min_{\varphi \in \mathcal{P}_2} \|\varphi(A)\tau_n(A)\phi_{n+2}(A)r_0\|, \tag{33}$$

TABLE 2

Algorithm CS-CGSTAB

$\rho_0 = \tilde{r}_0^T r_0$; $p_0 = r_0$; $\phi_0 = \|r_0\|$; $e_0 = q_0 = Ar_0$; $\mu_0 = 1$

$n \leftarrow 0$

While *method not converged yet* do:

$\sigma_n = (\tilde{r}_0^T q_n)\mu_n$; $c_n = Aq_n$ % Evaluate (27)

$u_{n+1} = \sigma_n r_n - \rho_n q_n$; $y_{n+1} = \sigma_n e_n - \rho_n c_n$; $d_{n+1} = Ay_{n+1}$

$\omega_{n+1} = (y_{n+1}, u_{n+1})/(y_{n+1}, y_{n+1})$

$\hat{r}_{n+1} = u_{n+1} - \omega_{n+1} y_{n+1}$; $\hat{e}_{n+1} = y_{n+1} - \omega_{n+1} d_{n+1}$; $\psi_{n+1} = \|\hat{r}_{n+1}\|$

% Decide whether to take a 1×1 step or a 2×2 step.

If $\psi_{n+1} < |\sigma_n|\phi_n$,Then % Condition (28a): $\|r_{n+1}\| < \|r_n\|$

one-step $= 1$

Else

$a_{11} = \tilde{r}_0^T q_n$; $a_{12} = \tilde{r}_0^T y_{n+1}$; $a_{21} = \tilde{r}_0^T c_n$; $a_{22} = \tilde{r}_0^T d_{n+1}$

$\delta_n = a_{11}a_{22} - a_{12}a_{21}$; $b_1 = \rho_n/\mu_n$; $b_2 = \tilde{r}_0^T e_{n+1}$

$\alpha_n = a_{22}b_1 - a_{12}b_2$; $\alpha_{n+1} = -a_{21}b_1 + a_{11}b_2$ % Solve (19)

$s_{n+2} = \delta_n r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}$; $t_{n+2} = \delta_n e_n - \alpha_n c_n - \alpha_{n+1} d_{n+1}$

$\tilde{\omega}_{n+1} = (t_{n+2}, s_{n+2})/(t_{n+2}, t_{n+2})$; $\tilde{\nu}_{n+2} = \|s_{n+2} - \tilde{\omega}_{n+1} t_{n+2}\|$ % Evaluate (32)

If $|\delta_n|\psi_{n+1} < |\sigma_n|\tilde{\nu}_{n+2}$, Then % Condition (28b): $\|r_{n+1}\| < \|r_{n+2}\|$

one-step $= 1$

Else % True ν_{n+2}

$v_{n+2} = At_{n+2}$; $w_{n+2} = Av_{n+2}$; $z_{n+2} = t_{n+2} - \omega_{n+1} v_{n+2}$

$\omega_{n+2} = (z_{n+2}, u_{n+2})/(z_{n+2}, z_{n+2})$; $\gamma_1 = -(\omega_{n+1} + \omega_{n+2})$; $\gamma_2 = \omega_{n+1}\omega_{n+2}$

$\hat{r}_{n+2} = s_{n+2} + \gamma_1 t_{n+2} + \gamma_2 v_{n+2}$; $\nu_{n+2} = \|\hat{r}_{n+2}\|$

$\hat{e}_{n+2} = t_{n+2} + \gamma_1 v_{n+2} + \gamma_2 w_{n+2}$

If $|\delta_n|\psi_{n+1} < |\sigma_n|\nu_{n+2}$, Then % Re-test w/$\nu_{n+2}$

one-step $= 1$

Else

one-step $= 0$

End If; End If; End If

% Compute next iterate.

If *one-step*, Then % *** Usual Bi-CGSTAB ***

$r_{n+1} = \hat{r}_{n+1}/\sigma_n$; $e_{n+1} = \hat{e}_{n+1}/\sigma_n$; $\phi_{n+1} = \psi_{n+1}/\sigma_n$

$x_{n+1} = x_n + (\rho_n p_n + \omega_{n+1} u_{n+1})/\sigma_n$

$\mu_{n+1} = (\mu_n \rho_n)/(\sigma_n \omega_{n+1})$; $\rho_{n+1} = (\tilde{r}_0^T r_{n+1})\mu_{n+1}$

$\beta_{n+1} = \rho_{n+1}/\rho_n$

$p_{n+1} = r_{n+1} + \beta_{n+1}(p_n - \omega_{n+1} q_n)$

$q_{n+1} = e_{n+1} + \beta_{n+1}(q_n - \omega_{n+1} c_n)$

$n \leftarrow n + 1$

Else % *** 2×2 step CSCGSTAB ***

$r_{n+2} = \hat{r}_{n+2}/\delta_n$; $e_{n+2} = \hat{e}_{n+2}/\delta_n$; $\phi_{n+2} = \nu_{n+2}/\delta_n$

$x_{n+2} = x_n + (\alpha_n p_n + \alpha_{n+1} u_{n+1} - \gamma_1 s_{n+2} - \gamma_2 t_{n+2})/\delta_n$

$\mu_{n+2} = -\mu_n(\alpha_{n+1}\rho_n/\delta_n\gamma_2)$; $\rho_{n+2} = (\tilde{r}_0^T r_{n+2})\mu_{n+2}$ % Evaluate (26)

$b_1 = \tilde{r}_0^T t_{n+2}$; $b_2 = \tilde{r}_0^T v_{n+2}$

$\beta_n = (a_{22}b_1 - a_{12}b_2)/\delta_n$; $\beta_{n+1} = (-a_{21}b_1 + a_{11}b_2)/\delta_n$ % Solve (20)

$p_{n+2} = r_{n+2} - \beta_n(p_n + \gamma_1 q_n + \gamma_2 c_n) - \beta_{n+1}(u_{n+1} + \gamma_1 y_{n+1} + \gamma_2 d_{n+1})$

$q_{n+2} = Ap_{n+2}$

$n \leftarrow n + 2$

End If

End While

where $\mathcal{P}_2$ is the set of all polynomials of degree at most 2 and $\varphi(0) = 1$.

Performing this minimization over two degrees of freedom was first presented in the BiCGSTAB2 algorithm by Gutknecht, [18]. The purpose was to cure problems that arise in the steepest descent part of Bi-CGSTAB due to eigenvalues of A in the complex spectrum that are not approximated well with eq. (2). Hence, every other step of the BiCGSTAB2 method performs (33) in order to handle conjugate pairs of eigenvalues. (In the remaining steps of the BiCGSTAB2 algorithm, the usual Bi-CGSTAB update is taken.) The BiCGSTAB2 algorithm can be summarized:

$$\tau_n \phi_n \overset{\text{1-D min}}{\longrightarrow} \tau_{n+1}\phi_{n+1} \overset{\text{2-D min}}{\longrightarrow} \tau_{n+2}\phi_{n+2}.$$

Unfortunately, in the implementation presented in [18], the two-dimensional minimization steps of BiCGSTAB2 are computed based on the Bi-CGSTAB step immediately before them. For skew-symmetric A, this poses a similar breakdown problem to the one mentioned above because the Bi-CGSTAB step requires a division by ω_{n+1}, which will be zero in this case.

Note that BiCGSTAB2 is mathematically equivalent to the BiCGSTAB(l) algorithm, due to Sleijpen and Fokkema [25] in the case where $l = 2$, but the implementation is different. BiCGSTAB(2) does not compute the intermediate Bi-CGSTAB residual $r_{n+1} = \tau_{n+1}\phi_{n+1}r_0$. Instead, it uses an intermediate basis vector, $A\tau_n\phi_{n+1}r_0$, to generate the auxiliary residual $\hat{r}_{n+2} = \tau_n\phi_{n+2}r_0$. This algorithm can be summarized:

$$\tau_n \phi_n \overset{A\tau_n\phi_{n+1}}{\longrightarrow} \tau_n\phi_{n+2} \overset{\text{2-D min}}{\longrightarrow} \tau_{n+2}\phi_{n+2}.$$

By not involving τ_{n+1}, BiCGSTAB(2) will not suffer the breakdown mentioned above. However, note that it is still prone to breakdown in the BCG ϕ_{n+1} part.

We now show how to overcome this breakdown problem. Recall that CS-CGSTAB already cures the BCG pivot breakdown in step $n + 1$, so all we need to do now is to show how to modify CS-CGSTAB so that it will overcome the τ_{n+1} breakdowns as well. The idea is to note that CS-CGSTAB has access to the $A\tau_n\phi_{n+1}r_0$ vector through the relationship $\xi_{n+1} = \sigma_n\phi_{n+1}$. We use it to compute the auxiliary residual $s_{n+2} = \tau_n\phi_{n+2}r_0$ to yield :

$$\tau_n \phi_n \overset{A\tau_n\xi_{n+1}}{\longrightarrow} \tau_n\phi_{n+2} \overset{\text{2-D min}}{\longrightarrow} \tau_{n+2}\phi_{n+2}.$$

The quantity s_{n+2} is updated by (17) and the 2-dimensional minimization step can be performed by first writing the residual

$$r_{n+2} = s_{n+2} + R\begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix},$$

where $R = [\,As_{n+2} \quad A^2 s_{n+2}\,] \equiv [\,t_{n+2} \quad v_{n+2}\,]$, and then minimizing $\|r_{n+2}\|$ by solving the system:

$$R^T R\begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix} = -R^T s_{n+2}. \tag{34}$$

In the case where A is skew-symmetric, the attempt to divide by $\gamma_2 = \omega_{n+1}\omega_{n+2}$ in the 2×2 step can now be performed without breakdown. In particular, if $A = -A^T$, then $\gamma_2 \equiv -(v_{n+2}, s_{n+2})/(v_{n+2}, v_{n+2}) \neq 0$ because the numerator $v_{n+2}^T s_{n+2} = s_{n+2}^T A^2 s_{n+2} > 0$.

We can now form a variant of the CS-CGSTAB algorithm which follows the former method in allowing 1×1 and 2×2 steps and differs only in the 2×2 step, performing instead the minimization over the 2 degrees of freedom described above. We use the same stepping strategy and approximation scheme to estimate $\|\nu_{n+2}\|$. We incorporate this into the new variant, CS-CGSTAB2. This is a more stable implementation which will not breakdown when A is skew-symmetric. Rather, in this case, provided there are no pivot breakdowns, it will always take 2×2 steps and will be equivalent to the BiCGSTAB(2) method.

In fact, when only 2×2 steps are taken, the methods Bi-CGSTAB, BiCGSTAB(2), CS-CGSTAB, CS-CGSTAB2, are all mathematically equivalent. In general, the composite step methods differ because they are variable step methods.

Note that in CS-CGSTAB2, the composite step is used to skip over breakdowns in the τ_{n+1} polynomial as well as ϕ_{n+1}. However, if there was no ϕ_{n+1} breakdown, and we took a 2×2 step in CS-CGSTAB2, then it is still possible that there could be pivot breakdown due to ϕ_{n+2}. In principle, we can solve this by applying the composite step idea to Bi-CGSTAB2 and taking a 3×3 step when we forsee possible pivot breakdown because ϕ_{n+3} exists under our assumption of $det(H^{(0)}) \neq 0$. However, we will not pursue this in this paper.

5. Numerical Experiments. All experiments are run in MATLAB 4.0 on a SUN Sparc station with machine precision about 10^{-16}. In most cases, as expected, composite step methods behave similarly to their non-composite step counterparts. In terms of the number of iterations it takes to converge, composite step methods are never worse in almost all cases, and in terms of the number of matrix-vector products performed, the cost is minimal. Here, we present a few selected examples where composite step does make a significant improvement.

Example 1. We begin the numerical experiments with a contrived example to illustrate the superior numerical stability of composite step methods over those without composite step. Let A be a modification of an example found in [23]:

$$A = \begin{pmatrix} \epsilon & 1 \\ -1 & 2 \end{pmatrix} \otimes I_{N/2},$$

i.e., A is a $N \times N$ block diagonal with 2×2 blocks, and $N = 40$. By choosing $b = (1 \quad 0 \quad 1 \quad 0 \quad \cdots)^T$ and a zero initial guess, we set $\sigma_0 = \epsilon$, and thus, we can forsee numerical problems with BCG polynomial based methods such as Bi-CGSTAB, BiCGSTAB2, and CGS when ϵ is small. Although these methods converge in 2 steps in exact arithmetic when $\epsilon \neq 0$, in finite precision, convergence gets increasingly unstable as ϵ decreases. Table 3 shows the relative error in the solution after 2 steps of BCG, CGS, and Bi-CGSTAB. Note that the loss of significant digits in BCG and Bi-CGSTAB is approximately proportional to $O(\epsilon^{-1})$ and the loss of digits in CGS is proportional to $O(\epsilon^{-2})$. The accuracy of CS-CGSTAB, CS-CGSTAB2, and CSCGS, the composite step CGS algorithm [9] is insensitive to ϵ and these three methods all converge in two steps with errors $\leq 10^{-16}$.

Example 2. Next we alter Example 1 slightly to show the advantage of CS-CGSTAB2 over CS-CGSTAB. Recall, CS-CGSTAB was developed to overcome breakdowns in cases where A is (nearly) skew-symmetric. Hence, if we change the last

TABLE 3
Example 1

Rel. error in the soln. after 2 steps (N=40)				
ϵ	Bi-CGSTAB	BiCGSTAB2	BiCGSTAB(2)	CGS
10^{-4}	1.5×10^{-12}	8.6×10^{-13}	2.5×10^{-16}	3.6×10^{-8}
10^{-8}	4.9×10^{-9}	4.7×10^{-9}	1.0×10^{-9}	2.2×10^{0}
10^{-12}	3.0×10^{-5}	7.3×10^{-4}	2.5×10^{-4}	3.0×10^{8}
CS-CGSTAB, CS-CGSTAB2, and CSCGS all converge with errors $\leq 10^{-16}$.				

example so that

$$A = \begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} \otimes I_{N/2},$$

we see that as ϵ gets small, CS-CGSTAB exhibits poor numerical results whereas CS-CGSTAB2 converges in the first 2×2 step as in the Example 1.

TABLE 4
Example 2

Rel. error in the soln. after 2 steps (N=40)					
ϵ	Bi-CGSTAB	BiCGSTAB2	BiCGSTAB(2)	CGS	CS-CGSTAB
10^{-4}	2.1×10^{-12}	2.9×10^{-13}	2.9×10^{-13}	2.5×10^{-8}	2.3×10^{-13}
10^{-8}	2.0×10^{-8}	bd	1.0×10^{-8}	1.0×10^{0}	7.6×10^{-10}
10^{-12}	1.2×10^{-4}	2.7×10^{1}	1.0×10^{-12}	1.3×10^{8}	bd
bd: Encountered breakdown					
CS-CGSTAB2 converged each time with error $\leq 10^{-16}$.					

Example 3. To emphasize the point made in Example 2, we pick a random skew-symmetric matrix with dimension $N = 20$ and a random right hand side. All the methods mentioned above either diverge or break down, except for CS-CGSTAB2 and BiCGSTAB(2), which achieve residual tolerance 10^{-11} in 24 iterations.

Example 4. We now show an example using a matrix which comes from the Harwell-Boeing set of sparse test matrices [11]. It is a discretization of the convection-diffusion equation:

$$L(u) = -\Delta u + 100(xu_x + yu_y) - 100u$$

on the unit square for a 63×63 grid. We use a random right hand side, zero initial guess, and no preconditioning. Figure 1 plots the true residual norm for Bi-CGSTAB, BiCGSTAB(2), CS-CGSTAB, and CS-CGSTAB2 versus the number of iterations taken, and Figure 2 shows the number of matrix-vector products. We have chosen a right hand side which yields some numerical instability for Bi-CGSTAB due to a near pivot breakdown around step 135 which results in convergence stagnation. Taking composite steps in this case overcomes this problem. We also see the advantage of the 2-dimensional minimization steps in the convergence behavior of BiCGSTAB(2) and CS-CGSTAB2.

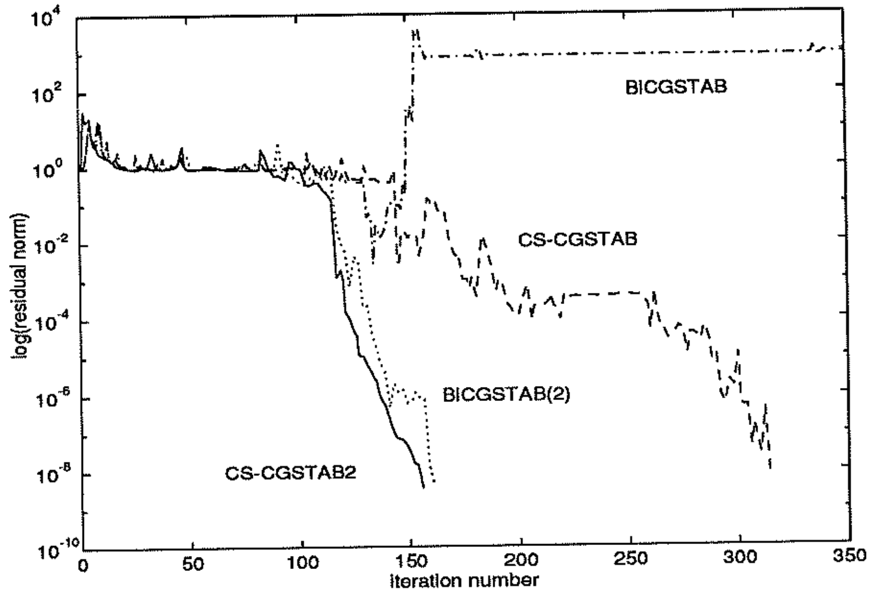

FIG. 1. *Example 4a*

The stepping strategy that we have described and implemented is conservative in that a 2×2 step is chosen whenever there is a peak in the residual convergence. If the result of the composite step is only a slight improvement, the extra cost it takes to perform a 2×2 step would be wasted. However, in practice, this increase in cost is relatively small. For example, in this particular problem, 96 of the steps taken in CS-CGSTAB are 2×2 steps and 50 2×2 steps are taken in CS-CGSTAB2. We see that the composite step methods require only about 15% more matrix-vector products in this example.

6. Best Approximation Results. Until recently, there has been very little theory known on the convergence of the Biconjugate gradient algorithm or other related methods. When Bank and Chan introduced CSBCG in [3], they also included a proof of a "best approximation" result for BCG. It is based on an analysis by Aziz and Babuška [1] and is similar to the analysis of the Petrov-Galerkin methods in finite element theory. Specifically, if we let M_k be any symmetric positive definite matrix and define the norm $|||v|||_*^2 \equiv v^t M_k v$, then Bank and Chan showed that the BCG error term $e_k^{BCG} = x - x_k^{BCG} = \phi_k(A)e_0$ can be bounded as follows:

$$|||e_k^{BCG}|||_* \leq (1 + \Gamma/\delta) \inf_{\phi_k : \phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2 |||e_0|||_*, \tag{35}$$

where Γ and δ are constants independent of k determined from inequalities involving $v \in \mathcal{V}_k$ and $w \in \mathcal{W}_k$, where $\mathcal{V}_k$ and $\mathcal{W}_k$ are the Krylov subspaces generated by the Lanczos method at the k^{th} step:

$$|w^T Av| \leq \Gamma |||v|||_* |||w|||_*;$$

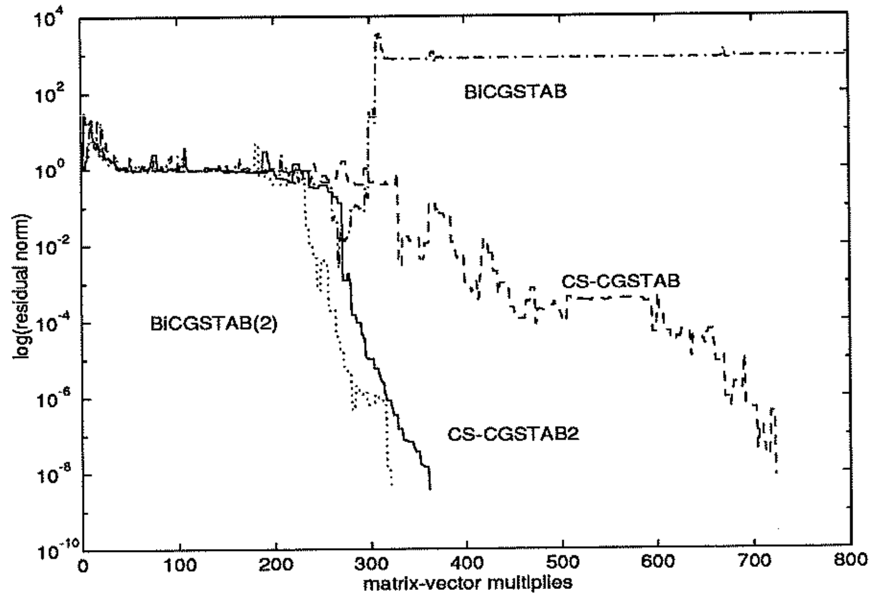

FIG. 2. *Example 4b*

$$\inf_{\substack{v \in \mathcal{V}_k \\ |||v|||_* = 1}} \sup_{\substack{w \in \mathcal{W}_k \\ |||w|||_* = 1}} w^T A v \geq \delta_k > 0.$$

Moreover, if we define the Lanczos tridiagonal matrix $T_k = W_k^T A V^k$ and its LU-factorization $T_k = L_k D_k U_k$, and define $M_k \equiv W_k U_k^T (D_k^T D_k)^{1/2} U_k W_k^T$, they showed that $\Gamma = \delta = 1$.

This result establishes convergence of BCG in the case where there are no breakdowns because then M_k is well-defined and symmetric positive definite. If this were not the case, the tridiagonal matrix T_k would be singular and such an M_k would not be positive definite. However, this result can be extended to cover situations with breakdown. For example, assuming no Lanczos breakdowns, the composite step approach does yield an M_k matrix based on a factorization of T_k which may involve 2×2 blocks, and hence, the above result applies to the error e_k corresponding to the well-defined iterates x_k [2]. In principle, if we add a look-ahead method to handle the Lanczos breakdowns to this, we can prove convergence of BCG for cases where both breakdowns occur.

Note that in general, simple upper bounds for the term

$$\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}} A M_k^{-\frac{1}{2}})\|_2 \tag{36}$$

are known only for special cases. For example, if we assume that the eigenvalues of A are contained in an ellipse in the complex plane which does not contain the origin, then, due to a result by Manteuffel [22], the quantity (36) can be bounded by a value dependent on the foci of the ellipse.

The product methods discussed earlier (CGS and Bi-CGSTAB) both involve the BCG polynomial. Hence, we can use the result in (35) to establish bounds on these methods as well. We first prove a lemma which will be used in the derivation of both bounds for CGS and Bi-CGSTAB.

LEMMA 1. *For any matrix $A \in R^{N\times N}$ and vector $v \in R^N$,*

$$|||Av|||_* \leq \|M^{\frac{1}{2}}\ AM^{-\frac{1}{2}}\ \|_2|||v|||_*$$

Proof. By definition, $|||w|||_* = \|M^{\frac{1}{2}}\ w\|_2$,
and thus,

$$\begin{aligned} |||Av|||_* &= \|M^{\frac{1}{2}}\ Av\|_2 = \|M^{\frac{1}{2}}\ AM^{-\frac{1}{2}}\ M^{\frac{1}{2}}\ v\|_2 \\ &\leq \|M^{\frac{1}{2}}\ AM^{-\frac{1}{2}}\ \|_2\|M^{\frac{1}{2}}\ v\|_2 = \|M^{\frac{1}{2}}\ AM^{-\frac{1}{2}}\ \|_2|||v|||_*. \ \square \end{aligned}$$

We now use this to estimate a bound on the CGS method and show the squaring effect on the convergence rate.

THEOREM 1. *Let $e_k^{CGS} = \phi_k^2(A)e_0$. Then*

$$|||e_k^{CGS}|||_* \leq c_1 \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2 \right)^2 |||e_0|||_*.$$

Proof. Applying Lemma 1 to $|||e_k^{CGS}|||_*$, we get

$$\begin{aligned} |||e_k^{CGS}|||_* &= |||\phi_k^2(A)e_0|||_* \\ &\leq \|\phi_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2|||\phi_k(A)e_0|||_* \\ &= \|\phi_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2|||e_k^{BCG}|||_* \\ &\leq c_1 \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2 \right)^2 |||e_0|||_*. \ \square \end{aligned}$$

Next we show convergence of the Bi-CGSTAB method and how the convergence rate of BCG is decreased further by the effect of the steepest descent.

THEOREM 2. *Let $e_k^{BiCGSTAB} = \tau_k(A)\phi_k(A)e_0$. Also, Let $\tilde{A} = M^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}$, and define S to be the symmetric part of $\tilde{A}$ (i.e., $S = \frac{1}{2}\ (\tilde{A} + \tilde{A}^T)$). Then if S is positive definite,*

$$|||e_k^{BiCGSTAB}|||_* \leq c_2 \left(1 - \frac{\lambda_{min}(S)^2}{\lambda_{max}(\tilde{A}^T\tilde{A})}\right)^{\frac{k}{2}} \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A})\|_2 \right) |||e_0|||_*.$$

Proof. First note that we can bound

$$\min_{\tau_k \in P_k} \|\tau_k(\tilde{A})\|_2 \ \leq \ \left(1 - \frac{\lambda_{min}(S)^2}{\lambda_{max}(\tilde{A}^T\tilde{A})}\right)^{\frac{k}{2}}$$

by applying the proof in Theorem 3.3 of (Eisenstat, Elman, and Schultz, [12]) to the matrix $\tilde{A}$. Combining this fact with Lemma 1, we can establish the following bound.

$$\begin{aligned} |||e_k^{BiCGSTAB}|||_* &= \min_{\tau_k \in P_k} |||\tau_k(A)\phi_k(A)e_0|||_* \\ &\leq \min_{\tau_k \in P_k} \left(\|\tau_k(M_k^{\frac{1}{2}}\ AM_k^{-\frac{1}{2}}\ )\|_2|||\phi_k(A)e_0|||_* \right) \end{aligned}$$

$$\begin{aligned}
&\leq \left(\min_{\tau_k \in P_k} \|\tau_k(\tilde{A})\|_2\right) |||\phi_k(A)e_0|||_* \\
&\leq \left(1 - \frac{\lambda_{min}(S)^2}{\lambda_{max}(\tilde{A}^T\tilde{A})}\right)^{\frac{k}{2}} |||e_k^{BCG}|||_* \\
&\leq c_2 \left(1 - \frac{\lambda_{min}(S)^2}{\lambda_{max}(\tilde{A}^T\tilde{A})}\right)^{\frac{k}{2}} \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A})\|_2\right) |||e_0|||_*. \quad \square
\end{aligned}$$

Recently, Barth and Manteuffel [4] have shown that the constant $(1 + \Gamma/\delta)$ in (35) can be improved to 1. In other words, e_k^{BCG} is minimized over K_n in the $|||\cdot|||_*$ norm. Correspondingly, the constants c_1 and c_2 in Theorems 1 and 2 can be improved to 1.

7. Conclusions. In conclusion, we have presented two new methods which are improvements on the Bi-CGSTAB algorithm using the composite step technique to increase numerical stability. They require only a minimal modification of the Bi-CGSTAB algorithm and cure the pivot breakdowns. One of the variants simultaneously cures the additional breakdown in Bi-CGSTAB due to skew-symmetric A. We employ an adaptive stepping strategy and the new methods do not require any user specified tolerance parameters. Numerical experiments support the improved convergence behavior of the new methods. Finally, convergence rate estimates were proved for the product methods Bi-CGSTAB, CGS, and their composite step counterparts.

REFERENCES

[1] A. K. Aziz and I. Babuška, *Part I, survey lectures on the mathematical foundations of the finite element method,* in The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, Academic Press, New York, 1972.

[2] R. E. Bank and T. F. Chan, *An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems,* Numer. Math., 66(1993), pp. 295-319.

[3] R. E. Bank and T. F. Chan, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems,* UCLA CAM Tech. Report 93-21 (1993). Numerical Algorithms, to appear.

[4] T. Barth and T. Manteuffel, *Variable metric conjugate gradient methods,* Paper presented at the Colorado Conference on Iterative Methods, Breckenridge, CO, 1994.

[5] C. Brezinski and H. Sadok, *Lanczos-type algorithms for solving systems of linear equations,* Applied Numerical Mathematics, 11(1993), pp. 443-473.

[6] C. Brezinski and M. Redivo-Zaglia, *Breakdowns in the computation of orthogonal polynomials,* Nonlinear Numerical Methods and Rational Approximation, A. Cuyt ed., Kluwer, Dordrecht. To appear.

[7] C. Brezinski and M. Redivo-Zaglia and H. Sadok, *A breakdown-free Lanczos type algorithm for solving linear systems,* Numer. Math. 63, 29-38 (1992).

[8] C. Brezinski and M. Redivo-Zaglia and H. Sadok, *Avoiding breakdown and near-breakdown in Lanczos type algorithms,* Numer. Alg. 1(1991) 261-284.

[9] T. F. Chan and T. Szeto, *A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems,* UCLA CAM Tech. Report 93-27 (1993). Numerical Alg., to appear.

[10] A. Draux, *Polynômes orthognaux formels,* Lect. Notes in Math., v. 974, Springer Verlag, Berlin, 1983.

[11] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems,* ACM Trans. Math. Softw., 15(1989), pp. 1-14.

[12] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations,...*

[13] R. Fletcher, *Conjugate gradient methods for indefinite systems,* in Numerical Analysis, Lecture Notes on Math. 506, G.A. Watson, ed., Springer-Verlag, Berlin, 1976, pp. 73-89.

[14] 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.

[15] R. W. FREUND AND N. M. NACHTIGAL, *A Look ahead TFQMR Method*, Presented at the Cornelius Lanczos International Centenary Conference, Raleigh, NC, December 1993.

[16] R. W. FREUND AND M.H. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comp., 13 (1992) 137-158.

[17] M. H. GUTKNECHT, *The unsymmetric Lanczos algorithms and their relations to Pade approximation, continued fraction and the QD algorithm*, in Proc. of the Copper Mt. Conf. on Iterative Methods, 1990.

[18] M. H. GUTKNECHT, *Variants of BiCGSTAB for matrices with complex spectrum*, IPS Research Report No. 91-14, ETH Zürich.

[19] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49(1952), pp. 409-436.

[20] W. JOUBERT, *Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations*, Ph.D. Thesis, The University of Texas at Austin, Austin, TX (1990).

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

[22] T. MANTEUFFEL, *An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters*, Ph.D. Thesis, University of Illinois, Urbana, 1975.

[23] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, Tech. Report, MIT Dept. of Math., Cambridge, MA (1990).

[24] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44(1985), pp.105-124.

[25] G. SLEIJPEN AND D. FOKKEMA, *BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum*, ETNA, 1(1993), pp. 11-32.

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

[27] C. H. TONG, *A comparative study of preconditioned Lanczos methods for nonsymmetric linear systems*, Tech. Rep. SAND91-8402 UC-404, Sandia National Laboratories, Albuquerque, 1992.

[28] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13(March 1992), pp. 631-644.

[29] H. A. VAN DER VORST, *The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Yu. Kolotilina (eds.), Lecture Notes in Mathematics 1457, Springer Verlag, Berlin, 1990.