UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

The Composite Step Family of Nonsymmetric Conjugate Gradient Methods

Tony F. Chan
Tedd Szeto

April 1994

CAM Report 94-11

This paper appears in the PROCEEDINGS FOR THE INTERNATIONAL SYMPOSIUM PCG'94 ON "MATRIX ANALYSIS AND PARALLEL COMPUTING". KEIO UNIVERSITY, YOKOHAMA, JAPAN, MARCH 1994

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

(RG'94) Matrix Analysis and Parallel Computing
Keio University, March 14 to 16, 1994

THE COMPOSITE STEP FAMILY OF NONSYMMETRIC CONJUGATE GRADIENT METHODS

TONY F. CHAN* AND TEDD SZETO†

Abstract. The Composite Step Biconjugate Gradient method (CSBCG), introduced recently by Bank and Chan [2, 3], is a stabilized variant of the Biconjugate Gradient (BCG) method which cures one of two possible causes of numerical instability in the BCG algorithm. Specifically, the composite step idea is to avoid breakdowns due to undefined iterates by skipping over those steps. This composite step idea can be incorporated into any algorithm which involves the BCG polynomial thereby leading to a family of composite step methods. In this paper we present a survey of these methods. For example, in [11], the CSBCG method is squared to obtain CSCGS, and in [12], composite step is applied to other product methods including Bi-CGSTAB [32], and some of its variants. Doing this not only cures the breakdown mentioned above, but also takes on the advantages of these product methods, namely, no multiplications by the transpose matrix, a faster convergence rate than BCG, and only two matrix-vector products per step to advance two degrees in the Krylov subspace. 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 methods in this family do produce improved performance over those without composite step on practical problems. Furthermore, we extend the "best approximation" result in [2] to obtain convergence proofs for CGS and Bi-CGSTAB.

1. Introduction.

The Biconjugate Gradient (BCG) algorithm [26] is the "natural" generalization of the classical Conjugate Gradient method [24] to nonsymmetric linear systems. It is an attractive method because of its simplicity and its good practical convergence properties. 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 the linear system $Ax = b$ and an auxiliary system, BCG produces iterates $x_n = x_0 + y_n$, with corresponding residuals of the form $r_n = b - Ax_n$, where $y_n \in K_n(r_0, A) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}$, and similarly for $\tilde{x}_n$ and $\tilde{r}_n$, and such that the following Galerkin conditions are satisfied:

$$r_n \perp K_n(\tilde{r}_0, A^T); \qquad \tilde{r}_n \perp K_n(r_0, A). \tag{1}$$

If we define K_n, K_n^* to be matrices whose columns span the Krylov spaces $K_n(r_0, A)$ and $K_n(\tilde{r}_0, A^T)$, respectively, condition (1) implies that the BCG iterate $x_n = x_0 + K_n v_n$ exists if we can find a solution to $(K_n^*)^T A K_n v_n = (K_n^*)^T r_0$. In other words, the iterate exists when the Hankel moment matrix

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

* 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.

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

It is well known that the BCG method is closely related to the nonsymmetric Lanczos process for computing the basis for the Krylov subspaces $K_n(r_0, A)$ and $K_n(\tilde{r}_0, A^T)$. One standard way to compute the BCG iterates is as follows [26, 16]:

Algorithm BCG

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

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 nonexistence 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 [4], 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 into the singularity of Hankel moment matrix $H_n^{(0)}$ for $K_n^*(\tilde{r}_0) K_n(r_0)$ [22] defined by:

$$H_n^{(0)} = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & \cdots & \mu_n \\ \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-2} \end{pmatrix}.$$

To overcome the pivot breakdown, we can "look ahead" in $H_n^{(1)}$ by not computing x_n where it is not defined. Rather, we build $H_i^{(1)}$ until it is no longer singular and we have an iterate $x_{n+m}, m \geq 1$. There are several approaches to handling this. In the case where A is symmetric, for which Lanczos breakdown cannot occur, it can be treated by the method of hyperbolic pairs due to Luenberger [27], and later expanded by Fletcher [16].

For general nonsymmetric matrices, the pivot breakdown can be cured using a three-term recurrence as done in the unnormalized BIORES algorithm of Gutknecht

[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 [18], *hard* and *soft* breakdowns [25].

[22]. The QMR method, due to Freund and Nachtigal [18], if considered without look-ahead Lanczos, numerically stabilizes the BCG method by computing an iterate defined by a "quasi-minimized" solution (which always exists) instead of the Galerkin condition.

Although the methods described above can cure possible singularities in $H_n^{(1)}$, $H_n^{(0)}$ can still be singular and cause breakdown problems. These breakdowns are harder to fix and many look-ahead methods have been proposed to remedy them as well, see e.g., Freund, Gutknecht and Nachtigal [20], Brezinski, Redivo-Zaglia and Sadok [8, 9], Brezinski and Sadok [4], Joubert [25], and Parlett et al [30]. Although the step size needed to overcome an exact 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*).

Recently, Bank and Chan introduced the Composite Step Biconjugate Gradient (CSBCG) algorithm [2, 3], an alternative which cures only the pivot breakdown (assuming no Lanczos breakdowns) by skipping over steps for which the BCG iterate is not defined. This is done with a simple modification of BCG which needs only a maximum look-ahead step size of 2 to eliminate the (near) breakdown and to smooth the sometimes erratic convergence of BCG. Lemma 4.3 in [3] proves that only two steps are needed, 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 [13].) 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.

The composite step idea, then, can in principle be incorporated anywhere the BCG polynomial is used; in particular, in product methods such as CGS [31], Bi-CGSTAB [32], Bi-CGSTAB2 [23], and TFQMR [17]. Doing this not only cures the breakdown mentioned above, but also takes on the advantages of these product methods over BCG, namely, no multiplications by the transpose matrix and a faster convergence rate. For example, if we take the CSBCG polynomials and square them, we obtain the Composite Step CGS method as shown in [11]. The Bi-CGSTAB algorithm computes iterates that are constructed from a more stable basis for the Krylov subspace, thereby handling some of the instability of CGS. In applying composite step to Bi-CGSTAB (CS-CGSTAB), we compute products of the CSBCG polynomial with a steepest descent polynomial to handle similar instability in CSCGS while maintaining the desirable properties. Other techniques can also be employed to stabilize CS-CGSTAB. For example, the Bi-CGSTAB2 method (Gutknecht, [23]) employs an alternate minimization strategy which can be applied during a composite step to further improve on this method. We can also apply composite step to the entire Bi-CGSTAB2 algorithm. These ideas are explored in [12].

There are other methods which also employ look-ahead techniques for product methods. The unnormalized $BIORES^2$ [22] squares the $BIORES$ method to handle pivot breakdowns. MRZS and its variants [5, 7] treat both breakdowns in the CGS method, as does the Look-ahead TFQMR method, currently being developed by Freund and Nachtigal [19]. The composite step approach, thus, should be viewed as one in a spectrum of methods with varying degrees of breakdown protection and complexities of implementation. Granted that it tries to cure only one of the two possible break-

downs, the composite step approach makes a conscious decision in favor of having a simpler method instead of a version (some of which were mentioned in the previous section) which is less prone to breakdown but more complicated and may require more matrix vector products.

An advantage to the composite step approach is that since CSBCG is based on BCG, it inherits its nice properties when extended to product methods. Extending other methods may not be so straight forward. For example, although it was stated earlier that the QMR method is a pivot breakdown-free, more stable alternative to BCG, we mention that the QMRS (QMR-squared) [21] cannot perform analogously for CGS because it is based on two-term recurrences that may suffer pivot breakdown. Moreover, it requires one more matrix-vector product per iteration. The MRZ method [8, 9] has been extended to the product method MRZS [5, 7] but this, too, involves extra matrix-vextor multiplications.

In [2], Bank and Chan also prove a "best approximation" result which establishes a bound on the error of BCG. Having this bound enables us to extend this result to prove convergence results for CGS and Bi-CGSTAB since these product methods both involve the BCG polynomial ϕ_n.

2. The Basic Composite Step Idea: CSBCG. Suppose in running the BCG algorithm (see section 1), we encounter a situation where $\sigma_n = 0$ at step n, and therefore, the values $x_{n+1}, \tilde{x}_{n+1}, r_{n+1}, \tilde{r}_{n+1}$ are not 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

$$z_{n+1} = \sigma_n r_{n+1} \in K_{n+2}(r_0); \qquad \tilde{z}_{n+1} = \sigma_n \tilde{r}_{n+1} \in K^*_{n+2}(\tilde{r}_0).$$

These always exist and thus, can be used in looking for the step $n+2$ iterate

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

with corresponding residual and search direction

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

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

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 solving two 2×2 linear systems. This yields 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, Lemma 5.1 in [2] 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.

2.1. CSBCG Stepping Strategy. As far as deciding when to take a 2×2 step, we do so whenever

$$\|r_{n+1}\| > \max\{\|r_n\|, \|r_{n+2}\|\}, \tag{3}$$

as decribed in [2, 3]. Obviously, this will avoid exact breakdowns by skipping over the "peak" in the residual convergence. Moreover, this strategy will yield a smoother, more stable method. In order to avoid unnecessary computation of $\|r_{n+2}\|$, we first evaluate condition (3a): $\|r_{n+1}\| < \|r_n\|$. Since r_{n+1} may be nonexistent, we use the auxiliary vector and check instead the equivalent condition: $\|z_{n+1}\| < |\sigma_n|\|r_n\|$. If this condition is not met, then we evaluate condition (3b): $\|r_{n+1}\| < \|r_{n+2}\|$, by restating it as $|\delta_n|\|z_{n+1}\| < |\sigma_n|\|\nu_{n+2}\|$, where $\nu_{n+2} \equiv \delta_n r_{n+2} = \delta_n r_n - \delta_n f_n^{(1)} A p_n - \delta_n f_n^{(2)} A z_{n+1}$.

The extra cost is minimal and note that no user specified tolerance parameters are required. Moreover, this stepping strategy can easily be applied in the implementation of the composite step product methods. The CSBCG algorithm is given in Table 1.

2.2. The Symmetric Case. Ideas similar to the composite step approach were used earlier by Luenberger [27] and Fletcher [16] in the case where A is symmetric. Note that in this case, there are no Lanczos breakdowns and mathematically, CSBCG and the methods in [16, 27] are, in fact, breakdown-free, and all produce precisely the same iterate x_{n+2}.

However, the details of exactly how x_{n+2} is updated are different. Luenberger treats only the case of exact breakdowns ($\sigma_n = 0$) and computes the iterates from a set of basis vectors different from that of CSBCG. Fletcher, on the other hand, handles near breakdowns similar to CSBCG but varies in the actual computation of f_n and g_n. This yields different roundoff properties when practically applied and compared to CSBCG on symmetric matrices.

Thus, CSBCG can be viewed as a way of generalizing [16] to nonsymmetric matrices.

3. Composite Step CGS. The Conjugate Gradients Squared (CGS) method (Sonneveld, [31]) is an attractive alternate to BCG because it is transpose-free and it often has a faster convergence rate. The CGS residual is based on a squaring of the BCG polynomial, $r_n^{CGS} = [\phi_n^{BCG}(A)]^2 r_0$, and thus, breakdowns exist in the CGS algorithm analogous to the breakdowns encountered in BCG. Hence, applying the composite step technique can eliminate the pivot breakdown in CGS assuming no Lanczos breakdowns.

For a 1×1 step, CSCGS is equivalent to CGS. In the case where we need to take a composite step, we must first express the CSBCG quantities in polynomial form:

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

For CSCGS, we want to compute

$$r_n^{CSCGS} = \phi_n^2(A) r_0$$

which yields corresponding iterates of the form $x_n^{CSCGS} \in x_0 + K_{2n}(r_0)$. This can be done using auxiliary vectors $p_n^{CSCGS} = \psi_n^2(A) r_0$ and $s_{n+1} = \xi_{n+1}^2(A) r_0$. The details of this computation can be found in [11].

The same stepping algorithm as described in section 2.1 for CSBCG can be used here but the details are different. Condition (3a) is now replaced by $\|s_{n+1}\| < \sigma_n^2 \|r_n\|$. Condition (3b) is more difficult because in order to compute $\nu_{n+2} \equiv \delta_n^2 r_{n+2}$, a matrix-vector multiplication is needed, and this would be wasteful if a 1×1 step is actually taken. An approximation to ν_{n+2} which does not involve a matrix-vector multiplication

TABLE 1
Algorithm CSBCG

$p_0 = r_0$; $\tilde{p}_0 = \tilde{r}_0$; $q_0 = Ap_0$; $\tilde{q}_0 = A^T\tilde{p}_0$; $\rho_0 = \tilde{p}_0^T r_0$
$n \leftarrow 0$
While *method not converged yet* do:
$\sigma_n = \tilde{p}_n^T q_n$
$z_{n+1} = \sigma_n r_n - \rho_n q_n$; $\tilde{z}_{n+1} = \sigma_n \tilde{r}_n - \rho_n \tilde{q}_n$
$y_{n+1} = Az_{n+1}$; $\tilde{y}_{n+1} = A^T\tilde{z}_{n+1}$
$\theta_{n+1} = \tilde{z}_{n+1}^T z_{n+1}$; $\zeta_{n+1} = \tilde{z}_{n+1}^T y_{n+1}$
$\xi_{n+1} = \|z_{n+1}\|$; $\phi_n = \|r_n\|$
% Decide whether to take a 1×1 step or a 2×2 step.
If $\xi_{n+1} < |\sigma_n|\phi_n$, Then % $\|r_{n+1}\| < \|r_n\|$
one-step $= 1$
Else
$\nu_{n+2} = \|\delta_n r_n - \rho_n^3 \zeta_{n+1} q_n - \theta_{n+1}\rho_n^2 y_{n+1}\|$
If $|\delta_n|\xi_{n+1} < |\sigma_n|\nu_{n+2}$, Then % $\|r_{n+1}\| < \|r_{n+2}\|$
one-step $= 1$
Else
one-step $= 0$
End If
End If
% Compute next iterate.
If *one-step*, Then % Usual BCG
$\alpha_n = \rho_n/\sigma_n$
$\rho_{n+1} = \theta_{n+1}/\sigma_n^2$; $\beta_{n+1} = \rho_{n+1}/\rho_n$
$r_{n+1} = r_n - \alpha_n q_n$; $\tilde{r}_{n+1} = \tilde{r}_n - \alpha_n \tilde{q}_n$
$x_{n+1} = x_n + \alpha_n p_n$; $\tilde{x}_{n+1} = \tilde{x}_n + \alpha_n \tilde{p}_n$
$p_{n+1} = z_{n+1}/\sigma_n + \beta_{n+1} p_n$; $\tilde{p}_{n+1} = \tilde{z}_{n+1}/\sigma_n + \beta_{n+1}\tilde{p}_n$
$q_{n+1} = y_{n+1}/\sigma_n + \beta_{n+1} q_n$; $\tilde{q}_{n+1} = \tilde{y}_{n+1}/\sigma_n + \beta_{n+1}\tilde{q}_n$
$n \leftarrow n + 1$
Else % 2×2 step C SBCG
$\delta_n = \sigma_n \zeta_{n+1}\rho_n^2 - \theta_{n+1}^2$
$\alpha_n = \zeta_{n+1}\rho_n^3/\delta_n$; $\alpha_{n+1} = \theta_{n+1}\rho_n^2/\delta_n$
$r_{n+2} = r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}$; $\tilde{r}_{n+2} = \tilde{r}_n - \alpha_n \tilde{q}_n - \alpha_{n+1}\tilde{y}_{n+1}$
$x_{n+2} = x_n + \alpha_n p_n + \alpha_{n+1} z_{n+1}$; $\tilde{x}_{n+2} = \tilde{x}_n - \alpha_n \tilde{p}_n - \alpha_{n+1}\tilde{z}_{n+1}$
$z_{n+2} = r_{n+2}$; $\tilde{z}_{n+2} = \tilde{r}_{n+2}$
$\rho_{n+2} = \tilde{z}_{n+2}^T r_{n+2}$; $\beta_n = \rho_{n+2}/\rho_n$; $\beta_{n+1} = \sigma_n \rho_{n+2}/\theta_{n+1}$
$p_{n+2} = z_{n+2} + \beta_n p_n + \beta_{n+1} z_{n+1}$; $\tilde{p}_{n+2} = \tilde{z}_{n+2} + \beta_n \tilde{p}_n + \beta_{n+1}\tilde{z}_{n+1}$
$q_{n+2} = Ap_{n+2}$; $\tilde{q}_{n+2} = A\tilde{p}_{n+2}$
$n \leftarrow n + 2$
End If
End While

is used to overcome this problem. Estimating ν_{n+2} by $\tilde{\nu}_{n+2}$, condition (3b) can be written $\delta_n^2\|s_{n+1}\| < \sigma_n^2\|\tilde{\nu}_{n+2}\|$.

In Table 2, we give the complete CSCGS algorithm. Note that there are 5 matrix-vector multiplications required for a 2×2 step, whereas in two steps of CGS, only 4 are needed. This is the price we pay for the composite step. However, it is still a considerable savings from BCG and also significantly less work than QMRS [21] and TFiQMR [10], where an extra matrix-vector multiply is required at *every* step.

In [11], the Minimum Residual Smoothing technique described in [33] by Weiss and in [34] by Zhou and Walker is applied to CSCGS. Similarly we can apply the QMR Smoothing algorithm, also in [34], to obtain a smoother version of CSCGS. Another possibility is to apply the composite step approach to Freund's Transpose Free QMR [17].

4. Composite Step Bi-CGSTAB. Similar to CSCGS, we can apply the composite step idea to handle breakdowns in other product methods. One such method is the Bi-CGSTAB method proposed by Van der Vorst [32] to stabilize CGS by multiplying the BCG polynomial with another polynomial of equal degree formed from a carefully chosen set of basis vectors for $K_n(r_0)$. Specifically for Bi-CGSTAB, we let $\tau_n(A) = (I - \omega_1 A)(I - \omega_2 A)\cdots(I - \omega_n A)$, where the ω_i's are chosen to locally minimize the residual by a steepest descent method, and define the residuals: $r_n^{Bi-CGSTAB} = \phi_n^{BCG}(A)\tau_n(A)r_0$. To incorporate the composite step idea, we simply use the CSBCG residual polynomial ϕ_n and obtain the CS-CGSTAB polynomials:

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

For the 2×2 composite step, we will need to evaluate

$$r_{n+2}^{CS-CGSTAB} = \tau_{n+2}\phi_{n+2}r_0; \quad p_{n+2}^{CS-CGSTAB} = \tau_{n+2}\psi_{n+2}r_0$$

using the quantities obtained from the n^{th} step. The details are worked out in [12].

Bi-CGSTAB finds ω_{n+1} in $r_{n+1}^{Bi-CGSTAB} = \tau_{n+1}\phi_{n+1}r_0 = (1 - \omega_{n+1}A)\tau_n\phi_{n+1}r_0$ by choosing it to minimize $\|r_{n+1}^{Bi-CGSTAB}\|$. We employ the same steepest descent rule to compute ω_{n+1} and ω_{n+2} by a minimization of $\|r_{n+1}\|$ and $\|r_{n+2}\|$, without having to actually compute r_{n+1} and r_{n+2}. If we let $u_n = \tau_n\xi_{n+1}r_0$, then $\|r_{n+1}\|$ is minimized by choosing $\omega_{n+1} = (Au_n, u_n)/(Au_n, Au_n)$. Similarly, we let $w_n = \tau_{n+1}\phi_{n+2}r_0$ and minimize $\|r_{n+2}\|$ by choosing $\omega_{n+2} = (Aw_n, w_n)/(Aw_n, Aw_n)$.

Once again, to decide when to take a 2×2 step, we use the algorithm from section 2.1. From the relationship $\xi_{n+1} = \sigma_n\phi_{n+1}$, we multiply by τ_{n+1}, and obtain $\sigma_n r_{n+1} = (I - \omega_{n+1}A)u_n$. Hence, condition (3a) can be written: $\|(I - \omega_{n+1}A)u_n\| < |\sigma_n|\|r_n\|$. To estimate $\|r_{n+2}\|$, we rescale the r_{n+2} update and let $\nu_{n+2} \equiv \delta_n r_{n+2}$, where δ_n is the determinant of a 2×2 matrix used in finding the unknowns f_n and g_n in CS-CGSTAB. Again, evaluating ν_{n+2} exactly would involve extra matrix-vector multiplies, so for practical purposes, we use an upper bound approximation to estimate $\|\nu_{n+2}\|$ using multiplications with $\kappa \approx \|A\|$. Thus, the condition $\|r_{n+1}\| < \|r_{n+2}\|$ can be expressed as: $|\delta_n|\|s_{n+1}\| < |\sigma_n|\|\tilde{\nu}_{n+2}\|$. The same consequences of the approximation of CSCGS apply in this case.

In Table 3, we present the CS-CGSTAB algorithm. Once again, there are 5 matrix-vector multiplications required for a 2×2 step, whereas in two steps of Bi-CGSTAB, only 4 are needed.

TABLE 2
Algorithm CSCGS

$\rho_0 = \tilde{r}_0^T r_0$; $p_0 = u_0 = r_0$; $b_0 = e_0 = Ap_0$
Compute κ = estimate for $\|A\|$
$n \leftarrow 0$
While *method not converged yet* do:
$\sigma_n = \tilde{r}_0^T b_n$
$q_{n+1} = \sigma_n u_n - \rho_n b_n$; $c_{n+1} = Aq_{n+1}$
$s_{n+1} = \sigma_n^2 r_n - \rho_n \sigma_n e_n - \rho_n c_{n+1}$; $\xi_{n+1} = \|s_{n+1}\|$; $\phi_n = \|r_n\|$
% Decide whether to take a 1×1 step or a 2×2 step.
If $\xi_{n+1} < \sigma_n^2 \phi_n$, Then % (a) $\|r_{n+1}\| < \|r_n\|$
one-step $= 1$
Else
$\theta_{n+1} = \tilde{r}_0^T s_{n+1}$; $\hat{\zeta}_n = \kappa \phi_0 \xi_{n+1}$; $\hat{\delta}_n = \sigma_n \hat{\zeta}_n \rho_n^2 - \theta_{n+1}^2$
$\hat{\alpha}_n = \hat{\zeta}_n \rho_n^3$; $\hat{\alpha}_{n+1} = \theta_{n+1} \rho_n^2$
$t_{n+1} = \sigma_n r_n - \rho_n e_n$
$\hat{v}_n = \hat{\delta}_n u_n - \hat{\alpha}_n b_n - \hat{\alpha}_{n+1} c_{n+1}$; $\hat{w}_n = \hat{\delta}_n t_{n+1} - \hat{\alpha}_n c_{n+1} - \hat{\alpha}_{n+1} \kappa s_{n+1}$
$\hat{\nu}_{n+2} = \|\hat{\delta}_n^2 r_n\| + \kappa \|\hat{\alpha}_n(\hat{\delta}_n u_n + \hat{v}_n) + \alpha_{n+1}(\hat{\delta}_n t_{n+1} + \hat{w}_n)\|$
If $\hat{\delta}_n^2 \xi_{n+1} < \sigma_n^2 \hat{\nu}_{n+2}$, Then % (b) $\|r_{n+1}\| < \|r_{n+2}\|$
one-step $= 1$
Else
$d_{n+1} = As_{n+1}$; $\zeta_{n+1} = \tilde{r}_0^T d_{n+1}$; $\delta_n = \sigma_n \zeta_{n+1} \rho_n^2 - \theta_{n+1}^2$
If $\delta_n^2 \xi_{n+1} < \sigma_n^2 \hat{\nu}_{n+2}$, Then % Test again with true δ_n in case test (b) failed
one-step $= 1$
Else
one-step $= 0$
End If
End If
End If
% Compute next iterate.
If *one-step*, Then % Usual CGS
$\alpha_n = \rho_n / \sigma_n$
$r_{n+1} = r_n - \alpha_n(e_n + c_{n+1}/\sigma_n)$
$x_{n+1} = x_n + \alpha_n(u_n + q_{n+1}/\sigma_n)$
$\rho_{n+1} = \tilde{r}_0^T r_{n+1}$; $\beta_n = \rho_{n+1}/\rho_n$
$u_{n+1} = r_{n+1} + \beta_n q_{n+1}/\sigma_n$; $e_{n+1} = Au_{n+1}$
$p_{n+1} = u_{n+1} + \beta_n(q_{n+1}/\sigma_n + \beta_n p_n)$
$b_{n+1} = e_{n+1} + \beta_n(c_{n+1}/\sigma_n + \beta_n b_n)$
$n \leftarrow n + 1$
Else % 2×2 step CSCGS
$\alpha_n = \zeta_{n+1} \rho_n^3 / \delta_n$; $\alpha_{n+1} = \theta_{n+1} \rho_n^2 / \delta_n$
$v_n = u_n - \alpha_n b_n - \alpha_{n+1} c_{n+1}$; $w_{n+1} = t_{n+1} - \alpha_n c_{n+1} - \alpha_{n+1} d_{n+1}$
$r_{n+2} = r_n - A[\alpha_n(u_n + v_n) + \alpha_{n+1}(t_{n+1} + w_{n+1})]$
$x_{n+2} = x_n + [\alpha_n(u_n + v_n) + \alpha_{n+1}(t_{n+1} + w_{n+1})]$
$\rho_{n+2} = \tilde{r}_0^T r_{n+2}$; $\beta_n = \rho_{n+2}/\rho_n$; $\beta_{n+1} = \sigma_n \rho_{n+2}/\theta_{n+1}$
$u_{n+2} = r_{n+2} + \beta_n v_n + \beta_{n+1} w_{n+1}$; $e_{n+2} = Au_{n+2}$
$p_{n+2} = u_{n+2} + \beta_n(v_n + \beta_n p_n + \beta_{n+1} q_{n+1}) + \beta_{n+1}(w_{n+1} + \beta_n q_{n+1} + \beta_{n+1} s_{n+1})$
$b_{n+2} = Ap_{n+2}$
$n \leftarrow n + 2$
End If
End While

TABLE 3
Algorithm CS-CGSTAB

$\rho_0 = \tilde{r}_0^T r_0; \quad p_0 = r_0; \quad \phi_0 = \|r_0\|; \quad q_0 = Ar_0; \quad \mu_0 = 1$
$n \leftarrow 0$
While *method not converged yet* do:
$\sigma_n = (\tilde{r}_0^T q_n)\mu_n; \quad e_n = Ar_n; \quad c_n = Aq_n$
$u_{n+1} = \sigma_n r_n - \rho_n q_n; \quad y_{n+1} = \sigma_n e_n - \rho_n c_n$
$\omega_{n+1} = (y_{n+1}, u_{n+1})/(y_{n+1}, y_{n+1})$
$\hat{r}_{n+1} = u_{n+1} - \omega_{n+1} y_{n+1}; \quad \psi_{n+1} = \|\hat{r}_{n+1}\|$
% Decide whether to take a 1×1 step or a 2×2 step.
If $\psi_{n+1} < |\sigma_n|\phi_n$, Then % (a) $\|r_{n+1}\| < \|r_n\|$
one-step $= 1$
Else
$d'_{n+1} = \kappa y_{n+1}$
$a_{11} = \tilde{r}_0^T q_n; \quad a_{12} = \tilde{r}_0^T y_{n+1}; \quad a_{21} = \tilde{r}_0^T c_{n+1}; \quad a'_{22} = \tilde{r}_0^T d'_{n+1}$
$\delta'_n = a_{11}a'_{22} - a_{12}a_{21}; \quad b_1 = \rho_n/\mu_n; \quad b_2 = \tilde{r}_0^T e_{n+1}$
$\alpha'_n = a'_{22}b_1 - a_{12}b_2; \quad \hat{\alpha}_{n+1} = -a_{21}b_1 + a_{11}b_2$
$s' = \delta'_n r_n - \alpha'_n q_n - \hat{\alpha}_{n+1} y_{n+1}; \quad t' = \delta'_n e_n - \alpha'_n c_n - \hat{\alpha}_{n+1} d_{n+1}; \quad v' = \kappa t'$
$w' = s' - \omega_{n+1} t'; \quad z' = t' - \omega_{n+1} v'$
$\omega'_{n+2} = (z', w')/(z', z'); \quad \gamma'_1 = -(\omega_{n+1} + \omega'_{n+2}); \quad \gamma'_2 = \omega_{n+1}\omega'_{n+2}$
$r'_{n+2} = s + \gamma'_1 t' + \gamma'_2 v'; \quad \xi'_{n+2} = \|r'_{n+2}\|$
If $|\delta'_n|\psi_{n+1} < |\sigma_n|\xi'_{n+2}$, Then % (b) $\|r_{n+1}\| < \|r_{n+2}\|$
one-step $= 1$
Else
$d_{n+1} = Ay_{n+1}$
$a_{22} = \tilde{r}_0^T d_{n+1}; \quad \delta_n = a_{11}a_{22} - a_{12}a_{21}; \quad \hat{\alpha}_n = a_{22}b_1 - a_{12}b_2$
$s = \delta_n r_n - \hat{\alpha}_n q_n - \hat{\alpha}_{n+1} y_{n+1}; \quad t = \delta_n e_n - \hat{\alpha}_n c_n - \hat{\alpha}_{n+1} d_{n+1}; \quad v = At$
$w = s - \omega_{n+1} t; \quad z = t - \omega_{n+1} v$
$\omega_{n+2} = (z, w)/(z, z); \quad \gamma_1 = -(\omega_{n+1} + \omega_{n+2}); \quad \gamma_2 = \omega_{n+1}\omega_{n+2}$
$\hat{r}_{n+2} = s + \gamma_1 t + \gamma_2 v; \quad \xi_{n+2} = \|\hat{r}_{n+2}\|$
If $|\delta_n|\psi_{n+1} < |\sigma_n|\xi_{n+2}$, Then % Test again with true ξ_{n+2} in case test (b) failed
one-step $= 1$
Else
one-step $= 0$
End If; End If; End If
% Compute next iterate.
If *one-step*, Then % Usual Bi-CGSTAB
$r_{n+1} = \hat{r}_{n+1}/\sigma_n; \quad \phi_{n+1} = \psi_{n+1}/\sigma_n$
$x_{n+1} = x_n + (\rho_n p_n + \omega_{n+1} u_{n+1})/\sigma_n$
$\mu_{n+1} = (\mu_n \rho_n)/(\sigma_n \omega_{n+1}); \quad \rho_{n+1} = (\tilde{r}_0^T r_{n+1})\mu_{n+1}$
$\beta_{n+1} = \rho_{n+1}/\rho_n$
$p_{n+1} = r_{n+1} + \beta_{n+1}(p_n - \omega_{n+1} q_n)$
$q_{n+1} = e_{n+1} + \beta_{n+1}(q_n - \omega_{n+1} c_n)$
$n \leftarrow n + 1$
Else % 2×2 step CSCGSTAB
$r_{n+2} = \hat{r}_{n+2}/\delta_n; \quad \phi_{n+2} = \xi_{n+2}/\delta_n$
$x_{n+2} = x_n + (\hat{\alpha}_n p_n + \hat{\alpha}_{n+1} u_{n+1} - \gamma_1 s - \gamma_2 t)/\delta_n$
$\mu_{n+2} = -\mu_n(\hat{\alpha}_{n+1}\rho_n/\delta_n\gamma_2); \quad \rho_{n+2} = (\tilde{r}_0^T r_{n+2})\mu_{n+2}$
$b_1 = \tilde{r}_0^T t_{n+1}; \quad b_2 = \tilde{r}_0^T v_{n+1}$
$\beta_n = (a_{22}b_1 - a_{12}b_2)/\delta_n; \quad \beta_{n+1} = (-a_{21}b_1 + a_{11}b_2)/\delta_n$
$p_{n+2} = r_{n+2} - \beta_n(p_n + \gamma_1 q_n) - \beta_{n+1}(u_{n+1} + \gamma_1 y_{n+1}) - \gamma_2 A(\beta_n q_n + \beta_{n+1} y_{n+1})$
$q_{n+2} = Ap_{n+2}$
$n \leftarrow n + 2$
End If
End While

5. Variants of CS-CGSTAB. We can also apply the composite step idea to the Bi-CGSTAB2 algorithm due to Gutknecht [23]. In this method, a two dimensional residual minimization is performed to handle problems in the steepest descent part of Bi-CGSTAB caused by eigenvalues of A with large imaginary part. In [12], we consider two such variants.

5.1. Hybrid CS-CGSTAB/CGSTAB2. A modification to the CS-CGSTAB method can be made in the minimization process similar to the odd steps of Bi-CGSTAB2. When we take a 2×2 step, we need not choose ω_{n+1} to behave as in Bi-CGSTAB. Instead, we minimize $\|r_{n+2}\| = \|(I - \omega_{n+2}A)(I - \omega_{n+1}A)r_n\phi_{n+2}r_0\|$ over the two degrees of freedom in ω_{n+1} and ω_{n+2}. This yields a noticeable improvement in the convergence behavior in practice.

5.2. CS-CSCGSTAB2. Another possibility is to consider the Bi-CGSTAB2 algorithm as a whole. This is a product method that takes two steps at a time. The first part is like Bi-CGSTAB, and the second part involves the minimization mentioned above. However, in Bi-CGSTAB2, the BCG polynomial ϕ_n can still break down in either part of the two-step since the method was designed to cure the r_n breakdowns only. We can overcome this by applying the composite step idea to Bi-CGSTAB2 when we forsee a nonexistent ϕ_n polynomial and taking a 3×3 step in cases where both may occur. The details of these variants are given in [12].

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 [2], 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 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 let the symmetric positive definite matrix $M_k = W_k U_k^T (D_k^T D_k)^{1/2} U_k W_k^T$ define the norm $|||v|||_*^2 = v^t M_k v$, 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:

$$|||e_k^{BCG}|||_* \leq (1 + \Gamma/\delta) \inf_{\phi_k : \phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}} A M_k^{-\frac{1}{2}})\|_2 |||e_0|||_*, \tag{4}$$

where Γ, δ are constants independent of k.

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 [3]. In principal, 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

$$\inf_{\phi_k : \phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}} A M_k^{-\frac{1}{2}})\|_2 \tag{5}$$

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, using a result by Manteuffel [28], the quantity (5) 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 (4) to establish bounds on these methods as well. These bounds are given in the following two theorems, the proofs of which can be found in [12].

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

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

THEOREM 2. *Let* $e_k^{BiCGSTAB} = \tau_k(A)\phi_k(A)e_0$. *Also, let* $\tilde{A} = M_k^{\frac{1}{2}} A M_k^{-\frac{1}{2}}$, *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,*

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

7. Numerical Experiments. All experiments are run in MATLAB 4.0 on a SUN Sparc station with machine precision about 10^{-16}. In most cases, composite step methods behave similarly to their non-composite step counterparts. Here, we present a few selected examples where composite step does make a significant difference. More numerical examples are found in [2, 3, 11, 12].

7.1. 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 [29]:

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

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 methods such as BCG, CGS, and Bi-CGSTAB 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 4 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 CSBCG, CSCGS, and CS-CGSTAB is insensitive to ϵ and these methods all converge in two steps with errors $\leq 10^{-16}$.

7.2. Example 2. We now show the effect of composite step when applied to the CGS algorithm using an example which comes from the Harwell-Boeing set of sparse test matrices [14]. The matrix is a discretization of the convection-diffusion equation:

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

TABLE 4
Example 1

Rel. error in the soln. after 2 steps (N=40)			
	BCG	CGS	Bi-CGSTAB
$\epsilon = 10^{-4}$	2.0×10^{-12}	3.6×10^{-8}	1.5×10^{-12}
$\epsilon = 10^{-8}$	4.7×10^{-9}	2.2×10^{0}	4.8×10^{-9}
$\epsilon = 10^{-12}$	3.7×10^{-4}	3.0×10^{8}	1.0×10^{-4}

on the unit square for a 63×63 grid. We use a random right hand side, zero initial guess, and left diagonal preconditioning. Figure 1 plots the number of iterations versus the true residual norm for CSCGS, CGS, and TFQMR. This illustrates the effect of cutting off the "peaks" of CGS. Specifically, note that the maximum point of the CGS curve is around 10^{10} whereas it is only 10^{6} for the composite step version. For this particular example, the CGS residual stagnates at 10^{-6} and so does TFQMR, whereas CSCGS reaches the stopping criterion $\|r_n\|/\|r_0\| < 10^{-8}$. The total number of 2×2 steps taken is 23, whereas 133 1×1 steps are taken.

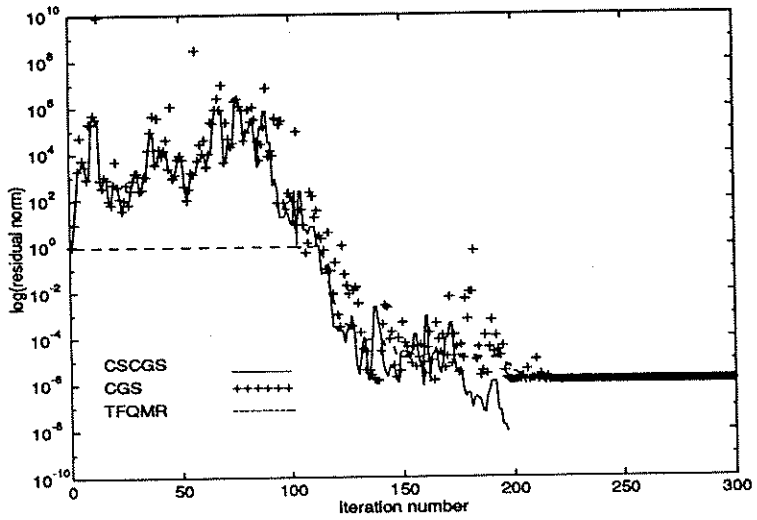

FIG. 1. *Example 2*

7.3. Example 3. Using the same matrix example, without preconditioning, we look at Bi-CGSTAB versus its composite step counterparts. We use a different right hand side from Example 2, one yielding some numerical instability for Bi-CGSTAB around step 140 which results in convergence stagnation. Taking composite steps in this case overcomes this problem. We also show the advantage of the hybrid variant discussed in section 5.1 over the standard CS-CGSTAB for this case.

REFERENCES

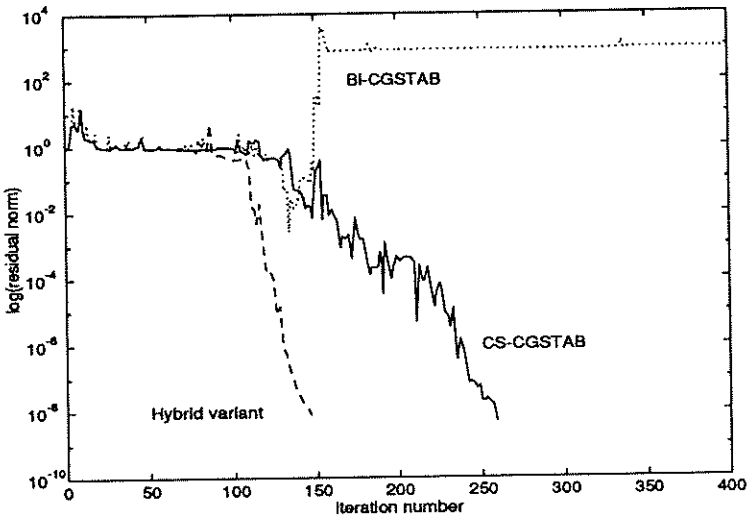

FIG. 2. *Example 3*

[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, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 93-21 (1993). Numer. Alg., to appear.

[3] R. E. BANK AND T. F. CHAN, *An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 92-53(1992). Numer. Math., 66(1993), pp. 295-319.

[4] C. BREZINSKI AND H. SADOK, *Lanczos-type algorithms for solving systems of linear equations*, Applied Numerical Mathematics, 11(1993), pp. 443-473.

[5] C. BREZINSKI AND H. SADOK, *Avoiding breakdown in the CGS algorithm*, Numer. Alg. 1(1991) 199-206.

[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, *Treatment of near-breakdown in the CGS algorithm*, USTL Publication ANO - 302 (1993). Numer. Alg., to appear.

[8] C. BREZINSKI, M. REDIVO-ZAGLIA AND H. SADOK, *A breakdown-free Lanczos type algorithm for solving linear systems*, Numer. Math. 63, 29-38 (1992).

[9] C. BREZINSKI, M. REDIVO-ZAGLIA AND H. SADOK, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Alg. 1(1991) 261-284.

[10] T. F. CHAN, L. DEPILLIS, AND H. VAN DER VORST, *A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems*, UCLA CAM Tech. Report 91-17 (1991).

[11] 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). Numer. Alg., to appear.

[12] T. F. CHAN AND T. SZETO, *Composite step product methods for solving nonsymmetric linear systems*, UCLA CAM Tech. Report (1994), in preparation.

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

[14] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math.

Softw., 15(1989), pp. 1-14.

[15] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for non-symmetric systems of linear equations*, SIAM J. of Numer. Anal., 20(1983), pp. 345-357.

[16] 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.

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

[18] 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.

[19] R. W. Freund and N. M. Nachtigal, *A Look ahead TFQMR Method*, Presented at the Cornelius Lanczos International Centenary Conference, Raleigh, NC, December 1993.

[20] 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.

[21] R. W. Freund and T. Szeto, *A Quasi-minimal residual squared algorithm for non-Hermitian linear systems*, UCLA CAM Tech. Report 92-19 (1992). Presented at the Copper Mountain Conference on Iterative Methods, April 1992.

[22] 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.

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

[24] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49(1952), pp. 409-436.

[25] 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).

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

[27] D. G. Luenberger, *Hyperbolic pairs in the method of conjugate gradients*, SIAM J. Appl. Math. 17(1969), pp.1263-1267.

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

[29] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13(1992), pp. 778-795.

[30] 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.

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

[32] 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.

[33] R. Weiss, *Convergence behavior of generalized conjugate gradient methods*, Ph. D. thesis, University of Karlsruhe (1990).

[34] L. Zhou and H. F. Walker, *Residual smoothing techniques for iterative methods*, Tech. rep., Dept. of Mathematics and Statistics, Utah State University (1992).