UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Composite Step Bi-Conjugate Gradient Algorithm for Nonsymmetric Linear Systems

Randolph E. Bank
Tony F. Chan

June 1993

CAM Report 93-21

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

A COMPOSITE STEP BI-CONJUGATE GRADIENT ALGORITHM FOR NONSYMMETRIC LINEAR SYSTEMS

RANDOLPH E. BANK* AND TONY F. CHAN†

Abstract. The Bi-Conjugate Gradient (BCG) Algorithm is the simplest and most natural generalization of the classical conjugate gradient method for solving nonsymmetric linear systems. It is well-known that the method suffers from two kinds of breakdowns. The first is due to the breakdown of the underlying Lanczos process and the second is due to the fact that some iterates are not well-defined by the Galerkin condition on the associated Krylov subspaces. In this paper, we derive a simple modification of the BCG algorithm, the Composite Step BCG (CSBCG) algorithm, which is able to compute all the well-defined BCG iterates stably, assuming that the underlying Lanczos process doesn't breakdown. The main idea is to skip over a step for which the BCG iterate is not defined.

Key Words. Biconjugate Gradients, Nonsymmetric Linear Systems.

AMS(MOS) subject classification. 65N20, 65F10

1. Introduction. The Conjugate Gradient (CG) Method is one of the simplest and most elegant methods in numerical computation. The algorithm itself can be described precisely in only a few lines but yet it manages to possess many desirable properties, including optimal convergence rate (minimizing the residual in a given Krylov subspace) and a very short recurrence. For these and other reasons, the CG method is one of the most popular iterative methods today.

Many attempts have been made to extend the CG method to solve general nonsymmetric problems. A fundamental result of Faber and Manteuffel [8] states that in the general nonsymmetric case one cannot preserve both the optimality and the short recurrence. Most existing methods can thus be classified into two classes. One class of methods preserve the optimality at the expense of having a longer recurrence (and the associated increase in cost). The best known example is the GMRES method [20]. Another class of methods retains the short recurrence at the expense of optimality. Among methods in this class are: BCG [15], CGS [21], BiCGSTAB[23], QMR [12], TFQMR[10] and QMRCGSTAB[5].

Among the second class of methods, the BCG method is perhaps the most natural extension of the CG method. The BCG iterates are defined by a Galerkin condition with respect to two Krylov subspaces associated with A and A^T, and reduces to the CG method for symmetric positive definite A's. Although many other more sophisticated methods have been invented since, the BCG method still remains competitive for many practical problems [22].

* Department of Mathematics, University of California at San Diego, La Jolla, CA 92093, USA. E-mail: rbank@ucsd.edu. The work of this author was supported by the Office of Naval Research under contract N00014-89J-1440.

† Department of Mathematics, University of California at Los Angeles, Los Angeles, California 90024, USA. E-mail: chan@math.ucla.edu. The work of this author was supported by the Office of Naval Research under contracts N00014-90J-1695 and N00014-92-J-1890, the Department of Energy under contract DE-FG03-87ER25307, the National Science Foundation under contracts ASC 90-03002 and 92-01266, and the Army Research Office under contract DAAL03-91-G-0150. Part of this work was completed during a visit to the Computer Science Dept., The Chinese University of Hong Kong.

It is well-known that the BCG method is intimately related to the nonsymmetric Lanczos iterative process for computing the basis for the two Krylov subspaces associated with A and A^T. The BCG method, however, does suffer from two sources of numerical instability, both of which can be traced to the underlying nonsymmetric Lanczos process. The first kind of instability occurs if the Lanczos process itself breaks down (we refer to this as *Lanczos breakdowns*). Breakdowns of the second kind occur if a principal submatrix of the tridiagonal matrix generated by the Lanczos process is singular, because BCG pivots with that matrix implicitly (we refer to this as *pivot breakdowns*). Pivot breakdowns occur when the corresponding BCG iterates are not well-defined by the Galerkin condition, whereas the Lanczos breakdowns are due to the breakdown of the particular recurrence use to compute the iterates. Although such exact breakdowns are very rare in practice, near breakdowns can cause severe numerical instability.

In this paper, we propose a simple modification of the BCG algorithm which eliminates pivot breakdowns, *assuming that Lanczos breakdowns do not occur.* The basic idea is to skip over one step of the BCG method explicitly when the corresponding BCG iterate is not well-defined. The resulting algorithm is still very simple but requires the solution of a 2×2 linear system whenever a step is skipped.

There have been several related approaches proposed in the literature to stabilize BCG. For symmetric A's, Lanczos breakdowns cannot occur and there are several natural ways to overcome the pivot breakdowns. Among the earliest are the method of hyperbolic pairs of Leunberger [16] and a method proposed by Fletcher in [9]. Both use a composite step similar (but different in the details) to ours. The SYMMLQ method of Paige and Saunders [18] uses an orthogonal factorization method to solve and update the solution of the tridiagonal systems generated by the Lanczos process. In principle, all three methods above are mathematically equivalent in the sense that they all compute the same well-defined BCG iterates.

For general nonsymmetric matrices, Saad [19] proposed using Gaussian elimination with partial pivoting explicitly on the Lanczos tridiagonal matrices. More recently, Freund and Nachtigal [12] proposed the QMR method (here we refer only to the version without look-ahead Lanczos) which eliminates pivot breakdowns by a "quasi-minimization" principle. Although the QMR method produces iterates which are generally different from the BCG iterates, it can be viewed as an alternative method for stabilizing the BCG method. Indeed, the BCG iterates can be recovered from the QMR recurrences with little cost. Lanczos breakdowns are much more difficult to eliminate. Freund, Gutknecht and Nachtigal [11] used a look-ahead Lanczos technique to overcome the curable Lanczos breakdowns. The BMRZ method of Brezinski, Redivo Zaglia and Sadok [2, 3], together with the strategy described in [4], is also free from breakdowns except for the *incurable* ones. However, these look-ahead algorithms are usually much more complicated than the BCG method because delicate heuristics have to be employed to determine the best step size to use at each iteration and the algorithm for actually performing a composite step of arbitrary size is necessarily complicated. In our approach, a maximum step size of two is needed and this simplifies the implementation a great deal. Our goal here is to derive a simple modification to the BCG algorithm which is able to compute all the well-defined BCG iterates stably, assuming that the underlying Lanczos process does not breakdown. We note that the occurrence of the two kinds of breakdowns are related (see e.g. [13, 14]) and our approach does not cure the more complicated Lanczos breakdowns.

The organization of the paper is as follows. In Section 2, we review the BCG algo-

rithm and the breakdowns. In Section 3, we derive the composite step BCG (CSBCG) method and in Section 4 we prove that CSBCG is able to compute exactly those BCG iterates which are well-defined. In Section 5, we discuss our choice of some heuristic strategies for deciding when to take a composite step and some implementation details. Our stepping strategy, which does not involve any arbitrary machine-dependent or user-supplied tolerances, is designed to avoid near pivot breakdowns as well as to provide some smoothing of the convergence history of the norm of the residual. Finally, in Section 6, we present the results of some numerical experiments.

Our main emphasis in this paper is the introduction of the composite step idea and the derivation of the basic CSBCG algorithm. In a companion paper [1], we investigate different implementations of the CSBCG method, as well as study in detail the connection of the CSBCG algorithm with the underlying nonsymmetric Lanczos process, from which a convergence proof for the CSBCG algorithm can be derived. We emphasize that we do not view the CSBCG method as a competitor for other methods such as QMR, CGS, BiCGSTAB, TFQMR etc. Afterall, in exact arithmetic, its iterates are a subset of those of BCG. Instead, our view is that since the BCG method underlies many other *product* methods, its improvement will lead to analogous improvements for those methods as well. For an example of this, see [6] for an application of the composite step idea to the CGS method.

2. Breakdowns of the BCG Algorithm. Suppose we would like to solve the following two related linear systems:

$$Ax = b, \quad A^T\tilde{x} = \tilde{b},$$

where A is a general N by N nonsymmetric matrix and we are given initial guesses x_0 and $\tilde{x}_0$, with corresponding residuals $r_0 = b - Ax_0$ and $\tilde{r}_0 = \tilde{b} - A^T\tilde{x}_0$. We assume A is real in this paper; generalization to the complex case is straightforward. If one is only interested in solving the first system, then $\tilde{r}_0$ can be chosen arbitrarily, although the performance of the method may depend on the particular choice.

Define the following two Krylov subspaces:

$$K_n(r_0) = < r_0, Ar_0, A^2r_0, ...A^{n-1}r_0 >,$$

$$K_n^*(\tilde{r}_0) = < \tilde{r}_0, A^T\tilde{r}_0, (A^T)^2\tilde{r}_0, ...(A^T)^{n-1}\tilde{r}_0 >,$$

where $\tilde{r}_0$ is arbitrary. Then the BCG iterates $x_n \equiv x_0 + y_n, \tilde{x}_n \equiv \tilde{x}_0 + \tilde{y}_n$ are defined by the Galerkin conditions:

1. $y_n \in K_n(r_0), \tilde{y}_n \in K_n^*(\tilde{r}_0)$,
2. $r_n \perp K_n^*, \tilde{r}_n \perp K_n$.

The following algorithm computes the BCG iterates x_n's by an efficient and simple iteration [9]:

Algorithm BCG Solves $Ax = b$ and $A^T\tilde{x} = \tilde{b}$ given x_0 and $\tilde{x}_0$:
Set $r_0 = b - Ax_0, p_0 = r_0, \tilde{r}_0 = \tilde{b} - A^T\tilde{x}_0, \tilde{p}_0 = \tilde{r}_0, \rho_0 = \tilde{r}_0^T r_0$.
For $n = 0, 1, ...$

$$\sigma_n \quad = \quad \tilde{p}_n^T A p_n;$$

$$
\begin{aligned}
\alpha_n &= \rho_n/\sigma_n; \\
r_{n+1} &= r_n - \alpha_n A p_n; \\
x_{n+1} &= x_n + \alpha_n p_n; \\
\tilde{r}_{n+1} &= \tilde{r}_n - \alpha_n A^T \tilde{p}_n; \\
\tilde{x}_{n+1} &= \tilde{x}_n + \alpha_n \tilde{p}_n; \\
\rho_{n+1} &= \tilde{r}_{n+1}^T r_{n+1}; \\
\beta_{n+1} &= \rho_{n+1}/\rho_n; \\
p_{n+1} &= r_{n+1} + \beta_{n+1} p_n; \\
\tilde{p}_{n+1} &= \tilde{r}_{n+1} + \beta_{n+1} \tilde{p}_n.
\end{aligned}
$$

End

It is well-known that, provided the above algorithm runs successfully to step n (i.e. $\sigma_i \neq 0, \rho_i \neq o, i = 0, 1, ..., n-1$) , then the iterates satisfy the following properties [9]:

THEOREM 2.1. *Let* $R_n = [r_0, r_1, ..., r_{n-1}], \tilde{R}_n = [\tilde{r}_0, ..., \tilde{r}_{n-1}]$ *and similarly for* $P_n, \tilde{P}_n$. *We have:*

1. $Range(R_n) = Range(P_n) = K_n(r_0)$, $Range(\tilde{R}_n) = \tilde{P}_n = K_n^*(\tilde{r}_0)$.
2. $\tilde{R}_n^T R_n$ *is diagonal.*
3. $\tilde{P}_n^T A P_n$ *is diagonal.*

As can be seen from the outline above, Algorithm BCG suffers from two kinds of breakdown:

1. $\rho_n \equiv \tilde{r}_n^T r_n = 0$ but $r_n \neq 0$ (Lanczos breakdown),
2. $\sigma_n = 0$ (singular pivot).

The Lanczos breakdown can be cured via Look-Ahead Lanczos procedures [11] in principle, but since the size of the look ahead step needed can be arbitrarily large and in any case is not easily determined, such procedures are necessarily quite complicated.

In this paper, we try to cure the (simpler) $\sigma_n = 0$ breakdown, assuming that the underlying Lanczos process does not breakdown. In other words, *we shall assume for the rest of this paper that* $\rho_n \neq 0$.

3. The Composite Step BCG Algorithm. Suppose Algorithm BCG runs successfully up to step n and encounters a $\sigma_n = 0$. This implies that the updates of $x_{n+1}, r_{n+1}, \tilde{x}_{n+1}, \tilde{r}_{n+1}$ are not defined. Our main idea is to avoid division by $\sigma_n = 0$ and look for a *composite step* update to obtain directly $x_{n+2}, r_{n+2}, \tilde{x}_{n+2}, \tilde{r}_{n+2}$.

To this end, define:

$$
\begin{aligned}
z_{n+1} &= \sigma_n r_n - \rho_n A p_n; \\
\tilde{z}_{n+1} &= \sigma_n \tilde{r}_n - \rho_n A^T \tilde{p}_n;
\end{aligned}
$$

Note that z_{n+1} and $\tilde{z}_{n+1}$ are scaled versions of r_{n+1} and $\tilde{r}_{n+1}$, but which remain defined even if $\sigma_n = 0$. By construction and Theorem 1, we still have $z_{n+1} \in K_{n+2}(r_0)$ and $\tilde{z}_{n+1} \in K_{n+2}^*(\tilde{r}_0)$.

Next we look for $x_{n+2}, \tilde{x}_{n+2}$ of the form:

$$
\begin{aligned}
x_{n+2} &= x_n + [p_n, z_{n+1}] f_n, \\
\tilde{x}_{n+2} &= \tilde{x}_n + [\tilde{p}_n, \tilde{z}_{n+1}] \tilde{f}_n
\end{aligned}
$$

where $f_n, \tilde{f}_n \in R^2$. Note that $x_{n+1} \in K_{n+2}(r_0)$ and $\tilde{x}_{n+1} \in K_{n+2}^*(\tilde{r}_0)$.

It follows that:

$$\begin{aligned} r_{n+2} &= r_n - A[p_n, z_{n+1}]f_n, \\ \tilde{r}_{n+2} &= \tilde{r}_n - A^T[\tilde{p}_n, \tilde{z}_{n+1}]\tilde{f}_n. \end{aligned}$$

The Galerkin condition of BCG requires:

$$\begin{aligned} (\tilde{p}_n, \tilde{z}_{n+1})^T r_{n+2} &= 0, \\ (p_n, z_{n+1})^T \tilde{r}_{n+2} &= 0, \end{aligned}$$

which give the following 2×2 linear systems for f_n and $\tilde{f}_n$:

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

$$(2) \qquad \begin{bmatrix} p_n^T A^T \tilde{p}_n & p_n^T A^T \tilde{z}_{n+1} \\ z_{n+1}^T A^T \tilde{p}_n & z_{n+1}^T A^T \tilde{z}_{n+1} \end{bmatrix} \begin{bmatrix} \tilde{f}_n^{(1)} \\ \tilde{f}_n^{(2)} \end{bmatrix} = \begin{bmatrix} p_n^T \tilde{r}_n \\ z_{n+1}^T \tilde{r}_n \end{bmatrix}.$$

Analogously, we look for p_{n+2} and $\tilde{p}_{n+2}$ of the form:

$$\begin{aligned} p_{n+2} &= r_{n+2} + (p_n, z_{n+1})g_n; \\ \tilde{p}_{n+2} &= \tilde{r}_{n+2} + (\tilde{p}_n, \tilde{z}_{n+1})\tilde{g}_n, \end{aligned}$$

where $g_n, \tilde{g}_n \in R^2$. Obviously, we have $p_{n+2} \in K_{n+2}(r_0)$ and $\tilde{p}_{n+2} \in K_{n+2}^*(\tilde{r}_0)$.

The conjugacy conditions:

$$\begin{aligned} (\tilde{p}_n, \tilde{z}_{n+1})^T A p_{n+2} &= 0, \\ (p_n, z_{n+1})^T A^T \tilde{p}_{n+2} &= 0 \end{aligned}$$

give:

$$(3) \qquad \begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = -\begin{bmatrix} \tilde{p}_n^T A r_{n+2} \\ \tilde{z}_{n+1}^T A r_{n+2} \end{bmatrix},$$

and

$$(4) \qquad \begin{bmatrix} p_n^T A^T \tilde{p}_n & p_n^T A^T \tilde{z}_{n+1} \\ z_{n+1}^T A^T \tilde{p}_n & z_{n+1}^T A^T \tilde{z}_{n+1} \end{bmatrix} \begin{bmatrix} \tilde{g}_n^{(1)} \\ \tilde{g}_n^{(2)} \end{bmatrix} = -\begin{bmatrix} p_n^T A^T \tilde{r}_{n+2} \\ z_{n+1}^T A^T \tilde{r}_{n+2} \end{bmatrix}.$$

Therefore, solving the above four 2×2 linear systems (1), (2), (3) and (4) give the updates from step n to step $n+2$. With a combination of 1×1 and 2×2 steps, one obtains the *composite step* CSBCG algorithm.

4. Some Properties of the CSBCG Algorithm. We prove the following lemma which we shall use several times later on.

LEMMA 4.1. *Using either a 1×1 or a 2×2 step to obtain $p_n, \tilde{p}_n$, we have: $\tilde{r}_n^T A p_n = \tilde{p}_n^T A r_n = \tilde{p}_n^T A p_n \equiv \sigma_n$.*

Proof. Suppose first that $p_n, \tilde{p}_n$ are obtained via a 1×1 step, i.e. $\tilde{p}_n = \tilde{r}_n + \beta_n \tilde{p}_{n-1}$. Then it follows that $\tilde{r}_n^T A p_n = \tilde{p}_n^T A p_n - \beta_n \tilde{p}_{n-1}^T A p_n = \tilde{p}_n^T A p_n \equiv \sigma_n$, since $\tilde{p}_{n-1}^T A p_n = 0$. Similarly, we obtain $r_n A^T \tilde{p}_n = p_n^T A^T \tilde{p}_n \equiv \sigma_n$.

Now suppose that $p_n, \tilde{p}_n$ are obtained via a 2×2 step, i.e. $\tilde{p}_n = \tilde{r}_n + \tilde{g}_{n-2}^{(1)} \tilde{p}_{n-2} + \tilde{g}_{n-2}^{(2)} \tilde{z}_{n-1}$. Then it follows that $\tilde{r}_n^T A p_n = \tilde{p}_n^T A p_n - \tilde{g}_{n-2}^{(1)} \tilde{p}_{n-2}^T A p_n - \tilde{g}_{n-2}^{(2)} \tilde{z}_{n-1}^T A p_n = \tilde{p}_n^T A p_n \equiv \sigma_n$, since the last two terms vanish due to the conjugacy condition. □

We next show that

LEMMA 4.2. *The four 2×2 coefficient matrices in Equations (1), (2), (3) and (4) are identical and symmetric.*

Proof. It suffices to show $\tilde{z}_{n+1}^T A p_n = \tilde{p}_n^T A z_{n+1}$. From the definitions of z_{n+1} and $\tilde{z}_{n+1}$, we have $\tilde{z}_{n+1}^T A p_n = \sigma_n \tilde{r}_n^T A p_n - \rho_n \tilde{p}_n^T A^2 p_n$ and $z_{n+1}^T A \tilde{p}_n = \sigma_n \tilde{p}_n^T A r_n - \rho_n \tilde{p}_n^T A^2 p_n$. The lemma now follows from Lemma 4.1.

□

We can now show that a 2×2 step is always sufficient for skipping over the breakdown of $\sigma_n = 0$.

LEMMA 4.3. *Assume that the Lanczos process underlying BCG doesn't break down, i.e. $\rho_i \neq 0, i = 0, 1, .., n$ and $\tilde{z}_{n+1}^T z_{n+1} \neq 0$. If $\sigma_n \equiv \tilde{p}_n^T A p_n = 0$ then the 2×2 coefficient matrix in (1), (2), (3) and (4) is nonsingular.*

Proof.

It suffices to show that if $\sigma_n \equiv \tilde{p}_n^T A p_n = 0$, then $\tilde{z}_{n+1}^T A p_n \neq 0$. From above, we have $\tilde{z}_{n+1}^T A p_n = z_{n+1}^T A \tilde{p}_n = -\rho_n \tilde{p}_n^T A^2 p_n = -\tilde{z}_{n+1}^T z_{n+1} / \rho_n \neq 0$. □

An alternative proof can also be derived from noting that the tridiagonal matrix generated by the underlying Lanczos process cannot have two successive singular leading principal submatrices if A is nonsingular. For more information on the relation between the CSBCG algorithm and the underlying Lanczos process, we refer the reader to [1], where this relationship is used to obtain a convergence proof of CSBCG. There is also a connection with the Hankel matrices defined by:

$$H_k^{(1)} = \begin{pmatrix} c_1 & \cdots & c_k \\ \vdots & \vdots & \vdots \\ c_k & \cdots & c_{2k-1} \end{pmatrix}, \qquad H_k^{(0)} = \begin{pmatrix} c_0 & \cdots & c_{k-1} \\ \vdots & \vdots & \vdots \\ c_{k-1} & \cdots & c_{2k-2} \end{pmatrix},$$

where $c_i = \tilde{r}_0^T A^i r_0$. It can be shown that a pivot breakdown occurs at the k-th iteration of BCG if $det(H_k^{(1)}) = 0$ and a Lanczos breakdown occurs if $det(H_k^{(0)}) = 0$. Thus, the above lemma corresponds to the following result: *assuming that $det(H_k^{(0)}) \neq 0$ for all k, then no two consecutive principal submatrix of $H_k^{(1)}$ can be singular.* The structure of these Hankel determinants was studied in detail by Draux [7].

Thus, with an appropriate combination of 1×1 and 2×2 steps, the CSBCG algorithm is guaranteed to run to completion (provided that $\rho_n \neq 0$). In the next lemma, we show that, with some minor changes, a result similar to Theorem 2.1 still holds.

THEOREM 4.4. *For the CSBCG algorithm, let $R_n = [r_0, r_1, ..., r_{n-1}], \tilde{R}_n = [\tilde{r}_0, ..., \tilde{r}_{n-1}]$ where $r_i, \tilde{r}_i$ are replaced by $z_i, \tilde{z}_i$ appropriately if the i-th step is a composite step, and let $P_n, \tilde{P}_n$ be similarly defined. Assuming the underlying Lanczos process doesn't breakdown, we have:*

1. *$Range(R_n) = Range(P_n) = K_n(r_0)$, $Range(\tilde{R}_n) = Range(\tilde{P}_n) = K_n^*(\tilde{r}_0)$.*
2. *$\tilde{R}_n^T R_n$ is diagonal.*
3. *$\tilde{P}_n^T A P_n = diag(D_k), k = 1, ...m$, where D_k is either 1×1 or 2×2.*

Proof. We shall prove the theorem by induction. The theorem is obviously true for $n = 1$ or $n = 2$ depending on whether a 1×1 or a 2×2 step is taken initially. Assume the theorem is true for index n. We shall prove that it is true for index $n+1$ or $n+2$, again depending on whether a 1-step or a 2-step is taken.

If a 1×1 step is taken, the proof is very similar to the usual proof of the analogous Theorem 2.1 for BCG and we shall not repeat that here.

So now assume a 2×2 step is taken. First, the relationships in (1.) above hold by construction. Next, let us look at (2.) concerning R_{n+2} and $\tilde{R}_{n+2}$. Note that the 2 new vectors in R_{n+2} are z_{n+1} and r_{n+2} and similarly for $\tilde{R}_{n+2}$. First we check the orthogonality conditions for z_{n+1}. Note that since z_{n+1} is a scaled version of r_{n+1}, we have $z_{n+1} \perp \tilde{r}_i, i \leq n$ by the usual inductive proof for BCG. Next, we check the orthogonality conditions for r_{n+2}. By construction, $r_{n+2} \perp (\tilde{p}_n, \tilde{z}_{n+1})$. For $i < n$, we have $\tilde{r}_i^T r_{n+2} = \tilde{r}_i^T(r_n - f_n^{(1)} A p_n - f_n^{(2)} A z_{n+1}) = \tilde{r}_i^T r_n - f_n^{(1)} \tilde{r}_i^T A p_n - f_n^{(2)} \tilde{r}_i^T A z_{n+1} = 0$ since each of the 3 terms on the right hand side vanishes by the induction hypothesis. Therefore, we have $r_{n+2} \perp Span(\tilde{r}_0, ..., \tilde{r}_{n-1}, \tilde{p}_n, \tilde{z}_{n+1})$. But $Span(\tilde{r}_0, ..., \tilde{r}_{n-1}, \tilde{p}_n, \tilde{z}_{n+1}) = Span(\tilde{r}_0, ..., \tilde{r}_{n-1}, \tilde{r}_n, \tilde{z}_{n+1})$. Therefore, $r_{n+2} \perp Span(\tilde{r}_0, ..., \tilde{r}_{n-1}, \tilde{r}_n, \tilde{z}_{n+1}) \equiv K_{n+1}^*(\tilde{r}_0)$. A similar argument shows that $\tilde{r}_{n+2} \perp Span(r_0, ..., r_n, z_{n+1}) \equiv K_{n+1}(r_0)$.

Finally, we turn to (3.). Note that the 2 new vectors in P_{n+2} are z_{n+1} and p_{n+2} and similarly for $\tilde{P}_{n+2}$. First, we check the conjugacy condition for z_{n+1}. If $i < n$, we have $\tilde{p}_i^T A z_{n+1} = 0$ because $A^T \tilde{p}_i \in K_n^*(\tilde{r}_0)$ and $z_{n+1} \perp K_n^*(\tilde{r}_0)$ by the orthogonality relation proven earlier. In other words, z_{n+1} is A-conjugate to $Span(\tilde{p}_0, ..., \tilde{p}_{n-1})$. But note that $\tilde{p}_n^T A z_{n+1} \neq 0$, which corresponds to the 2×2 block in (3.). Next, we check the conjugacy condition for p_{n+2}. By construction, p_{n+2} is A-conjugate to $(\tilde{p}_n, \tilde{z}_{n+1})$. For $i < n$, $\tilde{p}_i^T A p_{n+2} = \tilde{p}_i^T A r_{n+2} + (\tilde{p}_i^T A p_n, \tilde{p}_i^T A z_{n+1}) g_n = 0$ because each of the 3 terms on the right hand side vanishes due to the induction hypothesis. Therefore, we have proved that p_{n+2} is A-conjugate to $Span(\tilde{p}_0, ..., \tilde{p}_n, \tilde{z}_{n+1})$. Similar relationships hold for $\tilde{z}_{n+1}$ and $\tilde{p}_{n+2}$. This completes the proof. □

Using the above results, it is easy to prove the main result of this section.

THEOREM 4.5. *Assume that the Lanczos process does not breakdown, then CS-BCG computes, without breakdown, exactly those BCG iterates x_i's which are well-defined.*

Proof. By construction, $x_{n+2} = x_n + f_n^{(1)} p_n + f_n^{(2)} z_{n+1}$. By induction, $x_n \in x_0 + K_{n-1}(r_0)$ and $p_n \in K_n(r_0)$. By definition, $z_{n+1} \in K_{n+1}(r_0)$. It follows that $x_{n+2} \in x_0 + K_{n+1}(r_0)$. From Theorem 4.4, we have $r_{n+2} \perp K_{n+1}$. Therefore, x_{n+2} satisfies exactly the Galerkin condition defining the BCG iterate. □

5. Implementation Issues. We first prove some relations which simplify the solution of the four 2×2 systems (1), (2), (3) and (4).

LEMMA 5.1. *For the systems (1), (2), (3) and (4), the following relations hold:*

1. $p_n^T \tilde{r}_n = \tilde{p}_n^T r_n = \tilde{r}_n^T r_n \equiv \rho_n$ *and* $z_{n+1}^T \tilde{r}_n = \tilde{z}_{n+1}^T r_n = 0$, $f_n = \tilde{f}_n$.
2. $p_n^T A^T \tilde{r}_{n+2} = \tilde{p}_n^T A r_{n+2} = 0$ *and* $z_{n+1}^T A^T \tilde{r}_{n+2} = \tilde{z}_{n+1}^T A r_{n+2} = -\rho_{n+2}/f_n^{(2)}$ *(where* $\rho_{n+2} \equiv \tilde{r}_{n+2}^T r_{n+2}$*)*, $g_n = \tilde{g}_n$.
3. $p_n^T A^T \tilde{z}_{n+1} = \tilde{p}_n^T A z_{n+1} = -\theta_{n+1}/\rho_n$, *where* $\theta_{n+1} \equiv \tilde{z}_{n+1}^T z_{n+1}$.

Proof.

1. If p_n were computed using a 1×1 step, then $p_n = r_n + \beta_n p_{n-1}$. From this it follows that $p_n^T \tilde{r}_n = r_n^T \tilde{r}_n + \beta_n p_{n-1}^T \tilde{r}_n$. The last term on the right hand side vanishes due to Theorem 4.4. Analogously, we can show that $\tilde{p}^T r_n = \tilde{r}_n^T r_n$. If p_n were computed using a 2×2 step, then $p_n = r_n + (p_{n-2}, z_{n-1}) g_{n-2}$. Therefore, $p_n^T \tilde{r}_n = r_n^T \tilde{r}_n + (p_{n-2}^T \tilde{r}_n, z_{n-1}^T \tilde{r}_n) g_{n-2} = r_n^T \tilde{r}_n$, since the last two terms on the right hand side vanish due to Theorem 4.4. Similarly, we can show $\tilde{p}^T r_n = \tilde{r}_n^T r_n$. The equalities $z_{n+1}^T \tilde{r}_n = \tilde{z}_{n+1}^T r_n = 0$ follow from Theorem 4.4 directly. Finally, since the systems governing f_n and $\tilde{f}_n$ have the same

coefficient matrix and the same right hand side, we must have $f_n = \tilde{f}_n$.

2. $p_n^T A^T \tilde{r}_{n+2} = \tilde{p}_n^T A r_{n+2} = 0$ follow directly from Theorem 4.4 because $Ap_n \in K_{n+1}$ and $A^T \tilde{p}_n \in \tilde{K}_{n+1}$. By construction, $f_n^{(2)} A z_{n+1} = r_n - r_{n+2} - f_n^{(1)} A p_n$. Therefore, we can write $\tilde{r}_{n+2} A z_{n+1} = (\tilde{r}_{n+2}^T r_n - r_{n+2}^T \tilde{r}_{n+2} - f_n^{(1)} \tilde{r}_{n+2}^T A p_n)/f_n^{(2)} = r_{n+2}^T \tilde{r}_{n+2}/f_n^{(2)}$, since the first and third term on the right hand side vanish due to Theorem 4.4. (Note that if $\sigma_n = 0$ then $f_n^{(2)} \neq 0$ otherwise the first equation for the f system cannot be satisfied.) Similarly, we can prove that $r_{n+2} A^T \tilde{z}_{n+1} = \tilde{r}_{n+2}^T r_{n+2}/\tilde{f}_n^{(2)}$. Since we just proved $f_n = \tilde{f}_n$, we have $z_{n+1}^T A^T \tilde{r}_{n+2} = \tilde{z}_{n+1}^T A r_{n+2} = -\rho_{n+2}/f_n^{(2)}$. Finally, since the systems governing g_n and $\tilde{g}_n$ have the same coefficient matrix and the same right hand side, we must have $g_n = \tilde{g}_n$.
3. Since $z_{n+1} = \sigma_n r_n - \rho_n A p_n$, it follows that $p_n^T A^T \tilde{z}_{n+1} = (\sigma_n r_n^T \tilde{z}_{n+1} - z_{n+1}^T \tilde{z}_{n+1})/\rho_n = z_{n+1}^T \tilde{z}_{n+1}/\rho_n$, since the first term vanishes by Theorem 4.4. We proved earlier that $p_n^T A^T \tilde{z}_{n+1} = \tilde{p}_n^T A z_{n+1}$ when we proved that the 2×2 coefficient matrices are symmetric.

□

With Lemma 5.1, we need to solve only the following *two* linear systems for f_n and g_n for a 2×2 step:

$$\begin{bmatrix} \sigma_n & -\theta_{n+1}/\rho_n \\ -\theta_{n+1}/\rho_n & \zeta_{n+1} \end{bmatrix} \begin{bmatrix} f_n^{(1)} \\ f_n^{(2)} \end{bmatrix} = \begin{bmatrix} \rho_n \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} \sigma_n & -\theta_{n+1}/\rho_n \\ -\theta_{n+1}/\rho_n & \zeta_{n+1} \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \rho_{n+2}/f_n^{(2)} \end{bmatrix},$$

where $\zeta_{n+1} \equiv \tilde{z}_{n+1}^T A z_{n+1}$.

These two systems can be solved explicitly for f_n and g_n giving:

$$f_n = (\zeta_{n+1}\rho_n, \theta_{n+1})\rho_n^2/\delta_n$$

and

$$g_n = (\rho_{n+2}/\rho_n, \sigma_n \rho_{n+2}/\theta_{n+1}),$$

where $\delta_n \equiv \sigma_n \zeta_{n+1} \rho_n^2 - \theta_{n+1}^2$.

We summarize the CSBCG Algorithm below. In the algorithmic description, we have included a possible preconditioner B for A, and use the notation s_{n+1}, z_{n+1} to denote the unpreconditioned and the preconditioned residuals respectively. Also, we use the more natural notation (corresponding to BCG): α_n, α_{n+1} for $f_n^{(1)}, f_n^{(2)}$ respectively and β_{n+1}, β_{n+2} for $g_n^{(1)}, g_n^{(2)}$ respectively.

Algorithm CSBCG:

$p_0 = r_0; \quad \tilde{p}_0 = \tilde{r}_0$

$q_0 = Ap_0; \quad \tilde{q}_0 = A^T \tilde{p}_0$

$\rho_0 = \tilde{p}_0^T r_0$

$n \leftarrow 0$

Begin LOOP:

$\sigma_n = \tilde{p}_n^T q_n$

$z_{n+1} = \sigma_n r_n - \rho_n q_n; \quad \tilde{z}_{n+1} = \sigma_n \tilde{r}_n - \rho_n \tilde{q}_n$

$y_{n+1} = Az_{n+1}; \quad \tilde{y}_{n+1} = A^T \tilde{z}_{n+1}$

$\theta_{n+1} = \tilde{z}_{n+1}^T z_{n+1}$

$\zeta_{n+1} = \tilde{z}_{n+1}^T y_{n+1}$

If 1×1 step, Then

$\alpha_n = \rho_n / \sigma_n$

$\rho_{n+1} = \theta_{n+1} / \sigma_n^2$

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

$x_{n+1} = x_n + \alpha_n p_n; \quad \tilde{x}_{n+1} = \tilde{x}_n + \alpha_n \tilde{p}_n$

$r_{n+1} = r_n - \alpha_n q_n; \quad \tilde{r}_{n+1} = \tilde{r}_n - \alpha_n \tilde{q}_n$

$p_{n+1} = z_{n+1}/\sigma_n + \beta_{n+1} p_n; \quad \tilde{p}_{n+1} = \tilde{z}_{n+1}/\sigma_n + \beta_{n+1} \tilde{p}_n$

$q_{n+1} = y_{n+1}/\sigma_n + \beta_{n+1} q_n; \quad \tilde{q}_{n+1} = \tilde{y}_{n+1}/\sigma_n + \beta_{n+1} \tilde{q}_n$

$n \leftarrow n + 1$

Else

$\delta_n = \sigma_n \zeta_{n+1} \rho_n^2 - \theta_{n+1}^2$

$\alpha_n = \zeta_{n+1} \rho_n^3 / \delta_n; \quad \alpha_{n+1} = \theta_{n+1} \rho_n^2 / \delta_n$

$x_{n+2} = x_n + \alpha_n p_n + \alpha_{n+1} z_{n+1}; \quad \tilde{x}_{n+2} = \tilde{x}_n + \alpha_n \tilde{p}_n + \alpha_{n+1} \tilde{z}_{n+1}$

$r_{n+2} = r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}; \quad \tilde{r}_{n+2} = \tilde{r}_n - \alpha_n \tilde{q}_n - \alpha_{n+1} \tilde{y}_{n+1}$

$Bz_{n+2} = r_{n+2}; \quad B^T \tilde{z}_{n+2} = \tilde{r}_{n+2}$

$\rho_{n+2} = \tilde{z}_{n+2}^T r_{n+2}$

$\beta_{n+1} = \rho_{n+2} / \rho_n; \quad \beta_{n+2} = \rho_{n+2} \sigma_n / \theta_{n+1}$

$p_{n+2} = z_{n+2} + \beta_{n+1} p_n + \beta_{n+2} z_{n+1}; \quad \tilde{p}_{n+2} = \tilde{z}_{n+2} + \beta_{n+1} \tilde{p}_n + \beta_{n+2} \tilde{z}_{n+1}$

$q_{n+2} = Ap_{n+2}; \quad \tilde{q}_{n+2} = A^T \tilde{p}_{n+2}$

$n \leftarrow n + 2$

End If

End LOOP

At every step we compute the matrix-vector product $y_{n+1} = Az_{n+1}$ in anticipation of a 2×2 step. In case a 1×1 step is taken, this may appear as an extra cost. However, note that

$$Ap_{n+1} = Az_{n+1}/\sigma_n + \beta_{n+1} Ap_n$$

and therefore the vector y_{n+1} can be used to update the vector $q_{n+1} \equiv Ap_{n+1}$ without an extra matrix-vector product. Therefore the number of matrix-vector remains two per step, the same as in the BCG algorithm.

We next consider the issue of deciding between 1×1 and 2×2 updates. Our goal is to choose the step size which maximizes numerical stability. We have experimented with several decision processes based on the sizes of the elements in the 2×2 matrix

$$\begin{bmatrix} \sigma_n & -\theta_{n+1}/\rho_n \\ -\theta_{n+1}/\rho_n & \zeta_{n+1} \end{bmatrix}$$

and deciding locally whether to choose the 1×1 pivot σ_k or to use the matrix itself as a 2×2 pivot. Such schemes usually make reasonable decisions with respect to the matrix factorization, but, based on our numerical experience, are somewhat less satisfying with respect to the behavior of the CSBCG algorithm itself. Thus we are led to develop a heuristic based on the magnitudes of the residuals. If the (potential) residual from a 1×1 update satisfies $\| r_{n+1} \| \leq \| r_n \|$, then we choose a 1×1 update. Otherwise, we consider the (potential) residual r_{n+2} for a 2×2 update, and choose a 2×2 update if $\| r_{n+2} \| < \| r_{n+1} \|$. This test mathematically simplifies to choosing a 2×2 update only when

$$\| r_{n+1} \| > \max\{\| r_n \|, \| r_{n+2} \|\}. \tag{5}$$

Care should be taken in implementing the above tests so that no overflow occurs in the computation of $||r_{n+1}||$ and $||r_{n+2}||$ unnecessarily. We shall describe a simple scaling procedure for achieving this. Define $v_{n+2} \equiv \delta_n r_{n+2} = \delta_n r_n - \zeta_{n+1}\rho_n^3 A p_n - \theta_{n+1}\rho_n^2 A z_{n+1}$. Then the test $||r_{n+1}|| \leq ||r_n||$ can be replaced by $||z_{n+1}|| \leq |\sigma_n|||r_n||$ and the test $||r_{n+1}|| < ||r_{n+2}||$ can be replaced by $|\delta_n|||z_{n+1}|| < |\sigma_n|||v_{n+2}||$. Note that the new tests have no possibility of overflow even if $||r_{n+1}||$ or $||r_{n+2}||$ can be arbitrarily large. The following code fragment implements our test:

```
If || z_{n+1} || <= || r_n || |σ_k|, Then
    1 × 1 Step
Else
    ν_{n+2} = || δ_n r_n − ρ_n^3 ζ_{n+1} q_k − θ_{n+1} ρ_n^2 y_{n+1} ||
    If ν_{n+2} |σ_k| < || z_{n+1} || |δ_n|, Then
        2 × 2 Step
    Else
        1 × 1 Step
    End If
End If
```

When (5) is satisfied, taking two 1×1 steps would result in a "spike" in the convergence history of the residual norm. By making a 2×2 update in such circumstances, we effectively cut off such spikes. Note that this strategy not only eliminates near pivot breakdowns, but also provides some smoothing of the convergence history of the norm of the residuals. Moreover, this is achieved without any machine-dependent or user-supplied tolerances. We emphasize that CSBCG does not make the residual norm decrease monotonically, i.e. it can't eliminate *all* spikes, only those that are due the small pivots.

Finally, we mentioned that in this paper we have only derived one particular implementation of the CSBCG algorithm. Other variants, all mathematically equivalent but with different rounding error properties, are presented in [1].

6. Numerical Experiments. In this section, we will present a few numerical results for the CSBCG method. We emphasize that the purpose of these experiments is to show the stabilizing effect of the composite step strategy on the BCG method and not to show that the CSBCG method is better than other competing methods. For comparisons of the BCG method with other nonsymmetric CG-like methods, we refer the reader to [22].

Our first examples concern the model convection diffusion equation

$$-\Delta u + \beta u_x = 1$$

in $\Omega = (0,1) \times (0,1)$ with the Dirichlet boundary condition $u = 0$ on $\partial\Omega$. This problem is discretized on a uniform 33×33 mesh with 1089 vertices. We used continuous piecewise linear finite elements with no upwinding. The standard nodal basis functions were used for the finite element space. We consider the cases $\beta = 10$, leading to a relatively easy problem, and $\beta = 1000$, leading to a more difficult problem. As a linear system arising from partial differential equations, these examples are somewhat artificial because the mesh is uniform, because no upwinding was used to stabilize the discretization, and because we used no preconditioner. Although this leads to a more standard test environment, we note that the resulting linear algebra problem is generally made more difficult to solve by these choices. These calculations were performed in double precision arithmetic on a DECstation 5000/240 using the standard F77 compiler.

For these two test problems, we compared the BCG and CSBCG algorithms. The BCG algorithm was implemented using the same code as for CSBCG, except the pivot test was modified to always choose 1×1 update steps. We chose to measure error in terms of the relative residual, given by

$$e_k = \log_{10}(\| r_k \|_{\ell_2} / \| r_0 \|_{\ell_2})$$

We began with the initial conditions $x_0 = \tilde{x}_0 = 0$ and $r_0 = \tilde{r}_0$, and iterated until the relative residual was less than 10^{-4}.

The convergence history for the cases $\beta = 10$ and $\beta = 1000$ are given in figures 1 and 2, respectively.

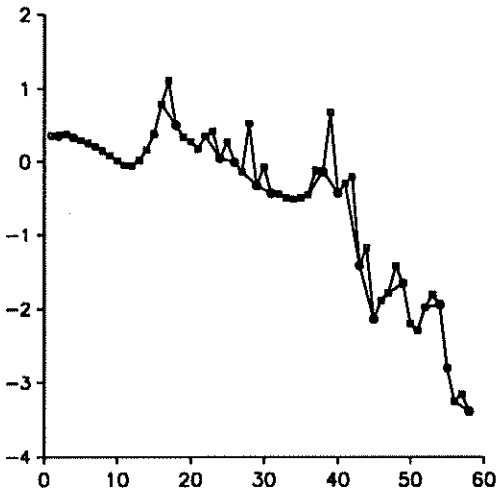

FIG. 1. *Convergence history e_k v. k for $\beta = 10$. BCG iterates are open boxes, CSBCG iterates are filled boxes (1×1 steps) and filled circles (2×2 steps).*

We note that for the case $\beta = 10$ the CSBCG algorithm works exactly as predicted, computing a subset of the BCG iterates, and smoothing the convergence history by skipping the "spikes" in the BCG history through the use of 2×2 steps. This is also true initially for the case $\beta = 1000$, but we note that after $k = 40$,

FIG. 2. *Convergence history e_k v. k for $\beta = 1000$. BCG iterates are open boxes, CSBCG iterates are filled boxes (1×1 steps) and filled circles (2×2 steps).*

the two methods begin to compute somewhat different iterates. We attribute this to accumulated roundoff error.

Finally, we consider a contrived example which is chosen to illustrate the potential superior numerical stability CSBCG has over BCG. The matrix A is a modification of an example presented in [17],

$$A = \begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} \otimes I_{N/2} \tag{6}$$

i.e., A is an $N \times N$ block diagonal matrix with 2×2 blocks. Furthermore, by choosing $b = (1\ 0\ 1\ 0\ \cdots)^T$, we set $\sigma_0 = \epsilon^{-1}$ and we can forsee numerical problems with BCG when ϵ is small. In exact arithmetic, BCG converges in exactly 2 steps if $\epsilon \neq 0$. In finite precision (in MATLAB on a SUN Sparc station IPX with machine precision about 10^{-16}), the relative error in the solution after 2 steps of BCG are (for $N = 40$) $1.5 \times 10^{-12}, 2.5 \times 10^{-8}, 4.9 \times 10^{-4}$ for $\epsilon = 10^{-4}, 10^{-8}, 10^{-12}$ respectively. Note that the number of digits lost are approximately $-\log_{10} \epsilon$, as expected. On the other hand, the corresponding relative errors for CSBCG are less than 10^{-16} for all 3 cases, demonstrating the effectiveness of the composite step strategy.

REFERENCES

[1] R. BANK AND T. CHAN, *An analysis of the composite step biconjugate gradient method*, Numer. Math., 66 (1993), pp. 295–319.

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

[3] ———, *Addendum to: Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Alg., 2 (1992), pp. 133–136.

[4] ———, *Breakdowns in the implementation of the Lanczos method for solving linear systems*, Int'l J. Math. with Applic., (1994). To appear.

[5] T. CHAN, E. GALLOPOULOS, V. SIMONCINI, T. SZETO, AND C. TONG, *QMRCGSTAB: A quasi-minimal residual version of the Bi-CGSTAB algorithm for nonsymmetric systems*, Tech. Rep. CAM 92-26, UCLA, Dept. of Math., Los Angeles, CA 90024-1555, 1992. To appear in SIAM J. Sci. Comp.

[6] T. F. CHAN AND T. SZETO, *A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems*, Numer. Alg., this issue (1994).

[7] A. DRAUX, *Polynômes orthognaux formels. Applications*, Lect. Notes in Math., vol. 974, Springer Verlag, Berlin, 1983.

[8] V. FABER AND T. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 315–339.

[9] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975 (Lecture Notes in Mathematics 506), G.A. Watsons (eds.), Springer (Berlin), 1976, pp. 73–89.

[10] R. W. FREUND, *A transpose-free quasi-minimum residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comp., 14 (1993), pp. 470–482.

[11] R. W. FREUND, M. H. GUTKNECHT, AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-hermitian matrices*, SIAM J. Sci. Comp., 14 (1993), pp. 137–158.

[12] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi residual residual method for non-Hermetian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[13] M. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal., 13 (1992), pp. 594–639.

[14] W. JOUBERT, *Generalized conjugate gradient and Lancoz methods for the solution of nonsymmetric systems of linear equations*, PhD thesis, Univ. of Texas at Austin, 1990.

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

[16] D. LUENBERGER, *Hyperbolic pairs in the method of conjugate gradients*, SIAM J. Applied Math., 17 (1969), pp. 1263–1267.

[17] N. NACHTIGAL, S. REDDY, AND L. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal., 13 (1992), pp. 778–795.

[18] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[19] Y. SAAD, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485–506.

[20] M. SCHULTZ AND Y. SAAD, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

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

[22] 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.

[23] 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. Comput., 13 (1992), pp. 631–644.