AN ANALYSIS OF THE COMPOSITE STEP BICONJUGATE
GRADIENT METHOD

RANDOLPH E. BANK* AND TONY F. CHAN†

Abstract. The composite step biconjugate gradient method (CSBCG) is a simple modification
of the standard biconjugate gradient algorithm (BCG) which smoothes the sometimes erratic con-
vergence of BCG by computing only a subset of the iterates. We show that 2 × 2 composite steps
can cure breakdowns in the biconjugate gradient method caused by (near) singularity of principal
submatrices of the tridiagonal matrix generated by the underlying Lanczos process. We also prove
a "best approximation" result for the method. Some numerical illustrations showing the effect of
roundoff error are given.

Key words. Biconjugate Gradients, Nonsymmetric Linear Systems.

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

1. Introduction. In this paper, we give an analysis of the composite step bi-
conjugate gradient method, for solving linear systems of the form

\[ Ax = r \]

where \( A \) is a large, sparse, nonsymmetric and indefinite, but nonsingular matrix. The
composite step biconjugate gradient method was introduced in [4] as a method for
improving the performance of the biconjugate gradient method.

As is well known [24], [10], [17], [8], [14], [12], [13], the biconjugate gradient
method can suffer from two sources of failure, both of which can be traced to the
underlying Lanczos process. One type, which we call a failure of the second kind or
a serious breakdown, is caused by a breakdown of the underlying Lanczos process.
The other type of failure, which we call a failure of the first kind, is simply due to
the fact that the biconjugate gradient method implicitly computes and uses the \( LDU \)
factorization of an indefinite tridiagonal matrix arising from the underlying Lanczos
process. Since no pivoting is used, there is the possibility of encountering small or
zero pivots in this factorization. The use of small pivots often appears as apparently
erratic convergence of the method. When a small pivot is encountered, typically the
residual norm will increase by a large amount on one iteration, only to be reduced by
a similar amount on the next step, creating a “spike” in the convergence history. Such
spikes can cause large cancellation errors and render the method numerically unstable.
In looking at such convergence histories, often littered with many such spikes, it is
clear that simply using 2 \( \times \) 2 updates to reduce the number of these spikes should go a
long way towards stabilizing the behavior of the method. It is in fact this observation
which forms the basis of the composite step biconjugate gradient method.

* Department of Mathematics, University of California at San Diego, La Jolla, CA 92037, USA.
E-mail: rbank@ucsd.edu. The work of this author was supported by the Office of Naval Research
under contract N00014-89-J-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 N00014-90-J-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 ASC 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.
The composite step biconjugate gradient method is just a simple algebraic modification of the regular biconjugate gradient method which allows one to proceed from an iterate $x_k$ to the iterate $x_{k+2}$ without computing $x_{k+1}$ (or the residual $r_{k+1}$). The cost is negligible; $2 \times 2$ composite steps cost about twice as much as $1 \times 1$ steps, which are in turn essentially the same as steps in the regular biconjugate gradient method. We show that the use of composite steps can prevent breakdown in the biconjugate gradient method due to failures of the $LDU$ factorization. Its impact on the serious breakdown of the Lanczos process is more problematic. On the one hand, our theory has nothing directly to say about this case, as we assume at the start that Lanczos breakdown does not occur. On the other hand, using a $2 \times 2$ update has a great deal of similarity to look ahead Lanczos procedures [19], [17], [9], [8]. Since looking ahead for more than two steps may be required to avoid a Lanczos failure, a simple composite step is certainly not sufficient in all cases to cure this type of breakdown. However, some Lanczos failures can be averted using just a double step. So while a simple composite step cannot cure this breakdown in all cases, it might still reduce the potential for Lanczos failure, and in any event, should not make the situation worse.

In section 2, we consider the factorization of general nonsingular tridiagonal matrices. There we show that such matrices can successfully be factored without pivoting if one allows the occasional use of $2 \times 2$ pivots. This becomes the theoretical basis of the composite step method. In section 3 we make a derivation of the composite step biconjugate gradient method from the underlying nonsymmetric Lanczos process. Under the assumption that there is no failure of the Lanczos process, we see that the use of $2 \times 2$ pivots or composite steps solves the problem of small or zero pivots in the biconjugate gradient method. It is simple to see that the composite step biconjugate gradient method reduces to the regular biconjugate gradient method if just $1 \times 1$ steps are used, and further, that it reduces to the regular conjugate gradient algorithm if the matrix $A$ and the preconditioner $B$ are symmetric and the starting conditions are appropriately chosen. Similarly, the composite step biconjugate gradient method reduces to a composite step conjugate gradient algorithm when $A$ and $B$ are symmetric (but possibly indefinite). This algorithm can be used for symmetric indefinite linear systems to address the problem of computing the $LDL^T$ factorization of the symmetric indefinite tridiagonal matrix forming the foundation of that process. Since the symmetric Lanczos process cannot have a serious failure, the composite step conjugate gradient method can be applied without as much qualification in these cases, and can be used as an alternative to the SYMMLQ class of methods [18], which are based on orthogonal factorizations of the tridiagonal matrix.

In section 4, we present an analysis of the convergence of the composite step biconjugate gradient method. With minor modification, our theorems can also be applied to the regular biconjugate gradient method. The theory applies for systems of the form (1) and allows general nonsingular preconditioners. The only assumption is that the underlying Lanczos process does not breakdown; i.e. no breakdowns of the second kind. We show that the (composite step) biconjugate gradient method produces iterates that are within a fixed constant factor of being optimal within the Krylov subspace, a so-called "best approximation" result. The key to our analysis is the use of the Babuška-Brezzi inf-sup condition. Using this condition, the convergence behavior of the biconjugate gradient method can be analyzed in a fashion analogous to the convergence of Petrov-Galerkin finite element methods [1]. In fact, the analysis is easier in the present case because all the spaces involved are of finite dimension.
To our knowledge, this is the first "best approximation" convergence result to be given for the biconjugate gradient method [8]. When specialized to the symmetric case, it is similar in content, but not quite as sharp, as the well-known convergence results for the conjugate gradient method [6]. One interesting point is that in the symmetric case, our theory includes the case of a symmetric indefinite matrix preconditioned by a symmetric indefinite matrix (as opposed to a symmetric positive definite preconditioner).

It is important to emphasize that mathematically, the composite step biconjugate gradient method is really a relatively simple modification of the regular biconjugate gradient method which allows the computation of a subset of the iterates. In exact arithmetic, it perhaps should not be regarded so much as a "new" algorithm as an interesting variant of an old one. On the other hand, smoothing the convergence history through the reduction of the number of spikes is practically a very desirable improvement in the procedure. In section 5, we present three of many possible implementations of the composite step biconjugate gradient method. These differ in their choice of the basis vectors for the two dimensional spaces used for the $2 \times 2$ composite steps. We also discuss how we decide between taking a regular or a composite step. In section 6, we present some numerical examples indicating the influence of roundoff error on the biconjugate gradient and composite step methods. Interestingly, while all three variants are identical mathematically, they often exhibit different convergence histories when applied to the same problem. Practically, details of implementation appear to be rather critical with respect to roundoff error. Trying to determine the best (or at least a very good) implementation with respect to roundoff from a large number of reasonable choices is an area of current interest for us.

2. The Factorization of a Tridiagonal Matrix. In this section we analyze the possible breakdown in the factorization without pivoting of an nonsingular tridiagonal matrix, and show how the problem can be corrected by the occasional use of $2 \times 2$ block pivots. This idea is similar to one used by Bunch [5] for the case of symmetric indefinite matrices. Most of the analysis is elementary, and is included mainly for completeness.

Let $T_n$ be the $n \times n$ nonsingular tridiagonal matrix given by

\begin{equation}
T_n = \begin{bmatrix}
\alpha_1 & \beta_1 \\
\gamma_1 & \alpha_2 & \beta_2 \\
& \ddots & \ddots & \ddots \\
& & \gamma_{n-2} & \alpha_{n-1} & \beta_{n-1} \\
& & & \gamma_{n-1} & \alpha_n
\end{bmatrix}.
\end{equation}

**Theorem 2.1.** Let $T_k$, $1 \leq k \leq n$ be the upper left principal submatrices of a nonsingular tridiagonal matrix $T_n$ as in (2). Then $T_{k-1}$ and $T_k$ cannot both be singular.

**Proof.** Let

$$
\rho_k = \text{Det}(T_k)
$$

for $1 \leq k \leq n$. It is easy to check [24], using expansion by minors, that the $\rho_k$'s satisfy the well known recurrence relation

$$
\rho_k = \alpha_k \rho_{k-1} - \beta_{k-1} \gamma_{k-1} \rho_{k-2}
$$
for $1 \leq k \leq n$, with the conventions $\rho_{-1} = 0$, $\rho_0 = \beta_0 = \gamma_0 = 1$. If $\rho_{k-2} = \rho_{k-1} = 0$, then $\rho_k = \rho_{k+1} = \cdots = \rho_n = 0$, contradicting the supposed nonsingularity of $T_n$. 

The main result in this section is:

**Theorem 2.2.** Let $T_n$ be the nonsingular tridiagonal matrix given in (2). Then $T_n$ can be factored as

$$T_n = L_n D_n U_n$$

where $L_n$ is unit lower block bidiagonal, $U_n$ is unit upper block bidiagonal, and $D_n$ is block diagonal, with $1 \times 1$ and $2 \times 2$ diagonal blocks.

**Proof.** The proof is by induction. The cases $n = 1$ and $n = 2$ are clear. There are two possibilities for the induction step. First, suppose $\alpha_1 \neq 0$. Then one has

$$T_n = \begin{bmatrix} 1 & 0 \\ c_{n-1} & I_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_1 & 0 \\ 0 & T_{n-1} \end{bmatrix} \begin{bmatrix} 1 & r_{n-1}^t \\ 0 & I_{n-1} \end{bmatrix}$$

where

$$c_{n-1} = [\gamma_1/\alpha_1 0 \ldots 0]$$

$$r_{n-1}^t = [\beta_1/\alpha_1 0 \ldots 0]$$

and

$$T_{n-1} = \begin{bmatrix} \alpha_2 - \beta_2 \gamma_1/\alpha_1 & \beta_2 \\ \gamma_2 & \alpha_3 & \beta_3 \\ \vdots & \vdots & \ddots \\ \gamma_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ \gamma_{n-1} & \alpha_n \end{bmatrix}.$$ 

Since $\text{Det}(T_n) = \alpha_1 \cdot \text{Det}(T_{n-1})$, $T_{n-1}$ is nonsingular and the induction hypothesis yields

$$T_{n-1} = L_{n-1} D_{n-1} U_{n-1}$$

and it follows that

$$T_n = \begin{bmatrix} 1 & 0 \\ c_{n-1} & I_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_1 & 0 \\ 0 & D_{n-1} \end{bmatrix} \begin{bmatrix} 1 & r_{n-1}^t \\ 0 & I_{n-1} \end{bmatrix}.$$ 

On the other hand, suppose that $\alpha_1 = 0$. Then $\gamma_1 \beta_1 \neq 0$; otherwise, $T_n$ would be singular. In this case we can use a $2 \times 2$ pivot and factor $T_n$ as

$$T_n = \begin{bmatrix} I_2 & 0 \\ C_{n-2} & I_{n-2} \end{bmatrix} \begin{bmatrix} D_2 & 0 \\ 0 & T_{n-2} \end{bmatrix} \begin{bmatrix} I_2 & R_{n-2}^t \\ 0 & I_{n-2} \end{bmatrix}$$

where

$$D_2 = \begin{bmatrix} \alpha_1 & \beta_1 \\ \gamma_1 & \alpha_2 \end{bmatrix}$$

$$C_{n-2} = D_2^{-1} \begin{bmatrix} 0 & 0 \ldots & 0 \\ \gamma_2 & 0 \ldots & 0 \end{bmatrix}$$

$$R_{n-2}^t = D_2^{-1} \begin{bmatrix} 0 & 0 \ldots & 0 \\ \beta_2 & 0 \ldots & 0 \end{bmatrix}$$
and

\[
T_{n-2} = \begin{bmatrix}
\alpha_3 & \beta_3 & \ldots & \beta_{n-1} \\
\beta_3 & \alpha_4 & \beta_4 & \ldots & \beta_{n-1} \\
\gamma_3 & \alpha_{n-1} & \beta_{n-1} & \ldots & \\
\alpha_{n-1} & \gamma_{n-2} & \alpha_{n-1} & \beta_{n-1} & \\
\gamma_{n-1} & \alpha_{n-1} & \gamma_{n-1} & \alpha_{n-1} & \beta_{n-1}
\end{bmatrix}.
\]

We have included terms with \( \alpha_1 \) to illustrate its impact when \( \alpha_1 \) is "small" but nonzero. In any event, \( \text{Det}(T_n) = \text{Det}(D_2) \cdot \text{Det}(T_{n-2}) \), so \( T_{n-2} \) is nonsingular and it follows from the induction hypothesis that

\[
T_{n-2} = L_{n-2}D_{n-2}U_{n-2}
\]

and

\[
T_n = \begin{bmatrix} I_2 & 0 \\ C_{n-2} & L_{n-2} \end{bmatrix} \begin{bmatrix} D_2 & 0 \\ 0 & D_{n-2} \end{bmatrix} \begin{bmatrix} I_2 & R_{n-2}^t \\ 0 & U_{n-2} \end{bmatrix}.
\]

\[
\square
\]

**Corollary 2.3.** Suppose \( T_n \) is singular, but \( T_{n-1} \) is nonsingular, where \( T_{n-1} \) is the upper left principal submatrix of order \( n-1 \). Then we have

\[
T_n = L_nD_nU_n
\]

where \( L_n \) is unit lower block bidiagonal, \( U_n \) is unit upper block bidiagonal, and \( D_n \) is block diagonal, with \( 1 \times 1 \) and \( 2 \times 2 \) diagonal blocks. In particular, \( d_n = 0 \) and corresponds to a \( 1 \times 1 \) block.

**Proof.** We know \( T_{n-1} = L_{n-1}D_{n-1}U_{n-1} \) by Theorem 2.2. Thus

\[
T_n = \begin{bmatrix} T_{n-1} & \beta_{n-1}c_{n-1} \\ \gamma_{n-1}c_{n-1}^t & \alpha_n \end{bmatrix} = \begin{bmatrix} L_{n-1} & 0 \\ r_{n-1}^t & 1 \end{bmatrix} \begin{bmatrix} D_{n-1} & 0 \\ 0 & d_n \end{bmatrix} \begin{bmatrix} U_{n-1} & c_{n-1} \\ 0 & 1 \end{bmatrix}
\]

where

\[
U_{n-1}^tD_{n-1}r_{n-1} = \gamma_{n-1}c_{n-1}
\]

\[
L_{n-1}D_{n-1}c_{n-1} = \beta_{n-1}c_{n-1}
\]

\[
d_n = \alpha_n - r_{n-1}^tD_{n-1}c_{n-1}
\]

\[
= \alpha_n - \beta_{n-1}\gamma_{n-1}c_{n-1}^tT_{n-1}\gamma_{n-1}c_{n-1}.
\]

Since \( T_n \) is singular, \( 0 = \text{det}(T_n) = \text{det}(D_n) = \text{det}(D_{n-1})d_n \). Since \( \text{det}(D_{n-1}) \neq 0 \), it follows that \( d_n = 0 \). \( \square \)

In the biconjugate gradient method, one does not have the complete matrix \( T_n \) given \textit{a priori}; rather, it is (implicitly) computed in bordered form and simultaneously factored. Only the most current part of the factorization is on hand, as earlier parts are discarded (overwritten) when they are no longer needed (see Sec. 3). Corollary 2.3 insures us that if \( T_k \) is singular, it can still be factored, and we can recognize its singularity by examining only the last diagonal entry (and potential next pivot) \( d_k \).
If \( d_k = 0 \), then we can "wait" until the bordering (Lanczos) process provides the last row and column of \( T_{k+1} \), knowing that the 2 \times 2 block
\[
\begin{bmatrix}
    d_k & \beta_k \\
    \gamma_k & \alpha_{k+1}
\end{bmatrix}
\]
will be nonsingular (by Theorem 2.2) and can be used as a 2 \times 2 pivot.

We next prove a technical result concerning a special choice of the \( \gamma_k \)'s in \( T_n \) which leads to a particularly simple form of \( L_n \). We will use this result in Sec. 3 to make the connection between the Lanczos procedure and the BCG algorithm.

**Corollary 2.4.** Let \( T_k \) denote the upper left principal submatrix of order \( k \) of \( T_n \). Suppose that for those values of \( k \) for which \( T_k \) is nonsingular, the subdiagonal entry \( \gamma_k \) of \( T_n \) satisfies
\[
(5) \quad \gamma_k = -(e_k^t T_k^{-1} e_1)^{-1}.
\]
Then the diagonal blocks of \( L_n \) are either 1 \times 1 or 2 \times 2 identity matrices and the subdiagonal blocks have the forms
\[
(6) \quad \begin{bmatrix} -1 & 0 \\ -1 & \gamma_k \end{bmatrix}, \begin{bmatrix} 0 & d_k/\gamma_k \\ -1 & 0 \end{bmatrix}.
\]

**Proof.**

We shall only give a sketch of the proof, which is based on induction. First note, that for those values of \( k \) for which \( T_k \) is nonsingular, by Theorem 2.2 we have the factorization \( T_k = L_k D_k U_k \) where \( D_k \) is nonsingular. The case of the first matrix \( (L_1 \text{ or } L_2 \text{ depending on whether the first block is } 1 \times 1 \text{ or } 2 \times 2) \) trivially satisfies the corollary. We thus assume \( T_k \) is nonsingular, with \( L_k \) having subdiagonal blocks of the form (6). We must show that if \( \gamma_k \) satisfies (5), then the next nonzero subdiagonal block in \( L_{k+1} \) (or \( L_{k+2} \)) has one of the forms given in (6).

First, it is easy to check that the vector \( h_k \) satisfying \( L_k h_k = e_1 \) has blocks of the form \([1, 0]^t\); note in particular that the entries \( d_k/\gamma_k \) in any 1 \times 2 or 2 \times 2 subdiagonal blocks of \( L_k \) have no influence on \( h_k \). Since \( e_k^t U_k^{-1} = e_k^t \), (6) reduces to one of the forms
\[
\begin{align*}
\gamma_k &= -d_k \\
\gamma_k &= -\begin{bmatrix} 0 & 1 \\ -1 & \gamma_k \end{bmatrix}^{-1} \begin{bmatrix} d_{k-1} & \beta_{k-1} \\ \gamma_{k-1} & \alpha_k \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{(d_k \alpha_k - \beta_{k-1} \gamma_{k-1})}{\gamma_{k-1}}
\end{align*}
\]
depending on whether the last diagonal block in \( T_k \) was 1 \times 1 or 2 \times 2. Then we have the factorization
\[
T_{k+1} = \begin{bmatrix} T_k & 0 \\
\gamma_k e_k^t & \alpha_{k+1}
\end{bmatrix} = \begin{bmatrix} L_k & D_k \\
0 & D_{k+1}
\end{bmatrix} \begin{bmatrix} U_k & c_k \\
0 & 1
\end{bmatrix}
\]
where \( r_k^t D_k U_k = \gamma_k e_k^t \), \( L_k D_k e_k = \beta_k e_k \), and \( d_k e_k = \alpha_{k+1} - r_k^t D_k e_k \). Note that this factorization exists even if \( T_{k+1} \) is singular, by Corollary 2.3. In any event, it follows
that

\[ r_k^i = \gamma_k e_k^i U_k^{-1} D_k^{-1} \]
\[ = \gamma_k e_k^i D_k^{-1}. \]

Thus the non-zeros in \( r_k \) must have one of the forms

\[
\begin{bmatrix}
-1 \\
-1 & d_{k-1}/\gamma_{k-1}
\end{bmatrix}
\]

depending again on whether the last block of \( T_k \) is \( 1 \times 1 \) or \( 2 \times 2 \). \( \Box \)

We remark that the condition \( d_k = 0 \) forces a \( 2 \times 2 \) step, but we have explicitly included \( d_k \) in the subdiagonal blocks to indicate what happens if a \( 2 \times 2 \) step is chosen when \( d_k \) is small but nonzero. In such cases, we need not assume \( \gamma_k \) satisfies (5), even though \( T_k \) is formally nonsingular.

3. The Preconditioned Biconjugate Gradient Algorithm in Relation to the Lanczos Algorithm. In this section, we develop the preconditioned composite step biconjugate gradient method (CSBCG) from the underlying Lanczos process. The Lanczos process and biconjugate gradient method are described in detail in [24], [8] [29], [14], [12], [13], and elsewhere. Our treatment here was inspired by the analysis given in Paige and Saunders [18] for the symmetric indefinite case.

We consider the solution of the problems

\[
Ax = r_0
\]
\[
A^t \tilde{x} = \tilde{r}_0
\]

by a preconditioned version of the biconjugate gradient method. Here \( A \) is an \( n \times n \) nonsingular matrix. Our real interest is in the solution of (7), but the system (8) is also solved as a byproduct of the biconjugate gradient method.

Let

\[
V_k = [v_1 \ v_2 \ \cdots \ v_k]
\]
\[
W_k = [w_1 \ w_2 \ \cdots \ w_k]
\]

be \( n \times k \) matrices, \( k \leq n \), of rank \( k \). We seek an approximate solution to (7) (respectively (8)) in the subspaces spanned by the columns of \( V_k \) (respectively \( W_k \)) using the Galerkin equations

\[
W_k^t A V_k u_k = W_k^t r_0
\]
\[
V_k^t A^t W_k \tilde{u}_k = V_k^t \tilde{r}_0
\]

and setting

\[
x_k = V_k u_k
\]
\[
\tilde{x}_k = W_k \tilde{u}_k
\]

When \( A \) is symmetric, positive definite and one chooses \( r_0 = \tilde{r}_0 \), and \( V_k = W_k \), then the Galerkin equations (9)-(10) and (11)-(12) become equivalent, and can be found by formally minimizing the functional

\[
f(u_k) = (AV_k u_k - r_0)^t A^{-1} (AV_k u_k - r_0). \]
Let $B$ be an $n \times n$ nonsingular preconditioner for $A$; in this derivation we are not assuming that $B$ is necessarily symmetric, positive definite. The Krylov subspace corresponding to $V_k$ is generated by the Lanczos process

$$
\begin{align*}
  u_0 &= 0 \\
  Bu_1 &= r_0 \\
  \gamma_j Bu_{j+1} &= Av_j - \alpha_j Bu_j - \beta_{j-1} Bu_{j-1}
\end{align*}
$$

for $j = 1, 2, \ldots$, and that corresponding to $W_k$ is generated by

$$
\begin{align*}
  w_0 &= 0 \\
  B^t w_1 &= r_0 \\
  \gamma_j B^t w_{j+1} &= A^t w_j - \alpha_j B^t w_j - \beta_{j-1} B^t w_{j-1}
\end{align*}
$$

The normalization constant $\gamma_0 = 1$; the $\gamma_j$ for $j \geq 1$ are nonzero scalars specified later. The scalars $\alpha_j$, $\beta_j$ for $j \geq 1$ are defined by

$$
\begin{align*}
  \alpha_j &= \frac{w_j^t Av_j}{w_j^t Bv_j} \\
  \beta_j &= \frac{w_{j+1}^t Bv_{j+1}}{w_j^t Bv_j}
\end{align*}
$$

and are chosen so that $V_k$ and $W_k$ are biorthogonal in the sense that

$$
W_k^t B V_k = \Lambda_k
$$

where $\Lambda_k$ is a nonsingular diagonal matrix. For completeness, we set $\beta_0 = 1$. Note that the assumption that the Lanczos process does not fail is equivalent to the assumption that $\Lambda_k$ is nonsingular.

Rearranging (13) and (14), we have

$$
\begin{align*}
  AV_k &= BV_{k+1} T_{k+1} E_k \\
  A^t W_k &= B^t W_{k+1} T_{k+1} E_k
\end{align*}
$$

where $T_k$ is the $k \times k$ tridiagonal matrix

$$
T_k = \begin{bmatrix}
  \alpha_1 & \beta_1 \\
  \gamma_1 & \alpha_2 & \beta_2 \\
  & \ddots & \ddots & \ddots \\
  & & \gamma_{k-2} & \alpha_{k-1} & \beta_{k-1} \\
  & & & \gamma_{k-1} & \alpha_k
\end{bmatrix}
$$

and $E_k$ is the $k+1 \times k$ matrix

$$
E_k = \begin{bmatrix}
  I_k \\
  0
\end{bmatrix}
$$

and $I_k$ is the $k \times k$ identity matrix.

From (16) and (17), it follows that

$$
W_k^t A V_k = W_k^t B V_{k+1} T_{k+1} E_k = \Lambda_k T_k
$$
and similarly that

\[ V_k^T A^T W_k = \Lambda_k T_k. \]

It follows that the matrix \( \Lambda_k T_k \) is symmetric. Moreover, from Theorem 2.2 and Corollary 2.3, \( T_k \) can be factored as \( T_k = L_k D_k U_k \), where \( L_k \) is unit lower block bidiagonal, \( U_k \) is unit upper block bidiagonal, and \( D_k \) is block diagonal with either 1 \( \times \) 1 or 2 \( \times \) 2 diagonal blocks. This factorization is defined as long as the Lanczos process is well defined, even if \( T_k \) happens to be singular. From this factorization and the fact that \( \Lambda_k T_k \) is symmetric, one can easily derive that \( \Lambda_k T_k \) has a triangular factorization given by

\[ \Lambda_k T_k = U_k^T (\Lambda_k D_k) U_k \]

where \( U_k = \Lambda_k L_k \Lambda_k^{-1} \), and \( \Lambda_k D_k \) is symmetric and block diagonal.

We now define the sequences of direction vectors \( p_k \) and \( \tilde{p}_k \) by

\[ P_k = [p_1 \ p_2 \ \cdots \ p_k] \]
\[ = V_k U_k^{-1}, \]
\[ \tilde{P}_k = [\tilde{p}_1 \ \tilde{p}_2 \ \cdots \ \tilde{p}_k] \]
\[ = W_k U_k^{-1}. \]

Suppose that \( T_k \) is nonsingular. Then, from (9) and (10), we have \( T_k u_k = \epsilon_1 \). Similarly, \( T_k \tilde{u}_k = \epsilon_1 \). Thus

\[ x_k = V_k u_k \]
\[ = V_k T_k^{-1} \epsilon_1 \]
\[ = P_k D_k^{-1} L_k^{-1} \epsilon_1 \]
\[ = P_k c_k \]
\[ \tilde{x}_k = W_k \tilde{u}_k \]
\[ = W_k T_k^{-1} \epsilon_1 \]
\[ = \tilde{P}_k c_k \]

where \( c_k = D_k^{-1} L_k^{-1} \epsilon_1 \).

We now derive the standard equations for the biconjugate gradient method. When \( T_k \) is nonsingular, we define \( \gamma_k \) by (recall that \( \gamma_k \) was left arbitrary in (13) and (14))

\[ \gamma_k = -((\epsilon_k^T T_k^{-1} \epsilon_1)^{-1}). \]

Note that for this choice of \( \gamma_k \), the structure of the lower triangular matrices \( L_k \) is given by Lemma 2.4; in particular, all nonzero off diagonal elements of \( L_k \) which are actually needed for the CSBCG algorithm are equal to \(-1\).

Then

\[ r_k = r_0 - A x_k \]
\[ = r_0 - A V_k u_k \]
\[ = r_0 - B V_{k+1} T_{k+1} F_k u_k \]
\[ = (r_0 - B V_k \epsilon_1) - (\gamma_k \epsilon_k^T u_k) B \epsilon_{k+1} \]
\[ = -(\gamma_k \epsilon_k^T T_k^{-1} \epsilon_1) B \epsilon_{k+1} \]
\[ = B \epsilon_{k+1}. \]
Similarly, we have
\[
\tilde{r}_k = \tilde{r}_0 - A^t \tilde{x}_k \\
= B^t w_{k+1},
\]
showing that the Lanczos vectors are the preconditioned residuals. Thus we define
\[
R_k = [r_0 \ r_1 \cdots r_{k-1}] \\
= BV_k \\
\tilde{R}_k = [\tilde{r}_0 \tilde{r}_1 \cdots \tilde{r}_{k-1}] \\
= B^t W_k.
\]

Notice using (21)-(22) that the left and right "residuals" are defined even for iteration steps for which \( T_k \) is singular, although the scaling is arbitrary for those steps.

From (15), we see that
\[
W_k^t R_k = \tilde{R}_k^t V_k = \tilde{R}_k^t B^{-1} R_k = \Lambda_k.
\]

Next note that, by using (18),
\[
\tilde{P}_k^t A P_k = U_k^t W_k^t A V_k U_k^{-1} \\
= U_k^t \Lambda_k T_k U_k^{-1} \\
= \Lambda_k D_k
\]
showing the direction vectors are biconjugate. If all blocks are \( 1 \times 1 \), this is just the usual relationship. If some blocks are \( 2 \times 2 \), direction vectors corresponding to a \( 2 \times 2 \) subspace are not biconjugate to each other. However, the biconjugate relationship is maintained at the subspace level. Note that \( \Lambda_k D_k \) is symmetric.

The basic Lanczos iteration summarized in (16) may be rewritten in terms of the residuals and direction vectors as
\[
A P_k U_k = R_{k+1} T_{k+1} E_k.
\]

We now assume that the lower right block of \( T_{k+1} \) is \( 1 \times 1 \). If it remains a \( 1 \times 1 \) block or becomes the first member of a \( 2 \times 2 \) block in \( T_{k+2} \), then the residual \( r_k \) will be updated on this step. Otherwise, if the lower right block of \( T_{k+1} \) is \( 2 \times 2 \), then the residual will not be updated on this step by the biconjugate gradient method. For steps when the residual is updated, the lower right block of \( D_{k+1} \) is \( 1 \times 1 \), \( D_k \) is nonsingular, and \( D_{k+1} \tilde{U}_{k+1} E_k = E_k D_k U_k \). Thus (26) can be written
\[
A P_k D_k^{-1} = R_{k+1} L_{k+1} E_k.
\]

Similarly, from (17),
\[
A^t \tilde{P}_k D_k^{-1} = \tilde{R}_{k+1} L_{k+1} E_k.
\]

We also have from (19)-(20)
\[
P_k U_k = B^{-1} R_k, \\
\tilde{P}_k U_k = B^{-1} \tilde{R}_k.
\]
Equations (26, 27) give the updates of $r_k$ and $p_k$ in the BCG algorithm in terms of the entries in $D_k$ and $L_{k+1}$, whereas equations (28), (29) give the updates for $p_k$ and $p_k$ in terms of the entries of $U_k$. We next derive the inner product relationships used these entries. Using (23)-(24), (28)-(29) and (18), we have

\begin{align*}
U_k &= (P_k^t A P_k)^{-1} \left\{ P_k^t A (B^{-1} R_k) \right\} \\
&= (P_k^t A P_k)^{-1} \left\{ P_k^t A^t (B^{-1} L_k) \right\} \\
&= \Lambda_k^{-1} L_k A_k
\end{align*}

and from (23)-(24) and (26)-(27)

\begin{align*}
\Lambda_k &= P_k^t L_k R_k = P_k^t R_k L_k = \tilde{R}_k B^{-1} R_k.
\end{align*}

Equation (31) can be combined with (24) to obtain

\begin{align*}
D_k^{-1} &= (P_k^t A P_k)^{-1} \left\{ P_k^t R_k L_k \right\} \\
&= (P_k^t A P_k)^{-1} \left\{ P_k^t R_k L_k \right\} \\
&= (P_k^t A P_k)^{-1} \left\{ \tilde{R}_k B^{-1} R_k \right\}.
\end{align*}

The coefficients for the residual updates in (26)-(27) can be obtained from one of the possibilities given in (32). Possibilities for computing coefficients for the direction vector updates in (28)-(29) are given in in the first two lines of (30). One can also obtain these coefficients as ratios of diagonal elements in $\Lambda_k$, using the last line of (30) with some form of (31).

4. A Best Approximation Result. In this section, we prove a best approximation result for the composite step biconjugate gradient method. Our analysis is based on the the Lax-Milgram theorem as developed by Babuška and Aziz in [1].

Let $V_n = \text{span}(v_1, v_2, \ldots, v_k)$ and $W_n = \text{span}(w_1, w_2, \ldots, w_k)$ denote the Krylov subspaces generated by the Lanczos method in (13) and (14) respectively. Let

\begin{align*}
\|v\|_r &= v^t M_r v \\
\|v\|_\ell^2 &= w^t M_{\ell} w
\end{align*}

where $M_r$ and $M_{\ell}$ are symmetric and positive definite, denote the (possibly different) norms associated with $V_n \equiv \mathcal{R}^n$ and $W_n \equiv \mathcal{R}^n$. As in the other sections, we consider the solution of (7).

**Theorem 4.1.** Suppose that for all $v \in V_n$ and for all $w \in W_n$, we have

\begin{align*}
|w^t A v| \leq \Gamma \|v\|_r \|w\|_\ell,
\end{align*}

where $\Gamma$ is a constant independent of $v$ and $w$. Further, suppose that for those steps in the composite step biconjugate gradient method in which we compute an approximation $x_k$, we have

\begin{align*}
\inf_{v \in V_k} \sup_{w \in W_k} w^t A v \geq \delta_k \geq \delta > 0 \\
\|v\|_r = 1 \quad \|w\|_\ell \leq 1
\end{align*}
Then
\begin{equation}
\|x - x_k\|_r \leq (1 + \Gamma/\delta) \inf_{v \in V_k} \|x - v\|_r.
\end{equation}

Proof. Our proof is a simplified (and specialized) version of arguments used in proving theorems 5.2.1 and 6.2.1 in [1]. Inequality (35) is the famous Babuška-Brezzi inf-sup condition as it applies to the current situation.

From the Galerkin equation (9) we have for \( w \in W_k \),
\begin{equation}
w^t A(x - x_k) = 0.
\end{equation}
Let \( v \in V_k \) be arbitrary. Then from (37)
\begin{equation}
w^t A(x_k - v) = w^t A(x - v)
\end{equation}
for all \( v \in W_k \). We now take the sup of both sides of (38) for all \( \|w\|_k \leq 1 \). We use (35) to bound the left hand side, noting \( x_k - v \in V_k \), and (34) to bound the right hand side. Thus we obtain
\begin{equation}
\delta_k \|x_k - v\|_r \leq \Gamma \|x - v\|_r.
\end{equation}
Using the triangle inequality and (39) we obtain
\begin{equation}
\|x - x_k\|_r \leq \|x - v\|_r + \|x_k - v\|_r \leq (1 + \Gamma/\delta) \|x - v\|_r.
\end{equation}
Since \( v \in V_k \) in (40) is arbitrary, (36) follows immediately. \( \square \)

Corollary 4.2. Let (34) and (35) hold. Then
\begin{equation}
\|\bar{x} - \bar{x}_k\|_k \leq (1 + \Gamma/\delta) \inf_{w \in W_k} \|\bar{x}_k - w\|_k.
\end{equation}

Proof. The proof is analogous to the proof of theorem 4.1. \( \square \)

Equation (34) is a standard continuity assumption for the linear operator \( A \). The inf-sup condition (35) asserts the nonsingularity of \( A \) for the case \( k = n \), and of \( \Lambda_k T_k \) for those steps in which we solve for an approximate solution \( x_k \). If \( v \in V_k \), then \( v = V_k \hat{v} \) for some \( \hat{v} \in \mathbb{R}^k \). Similarly, \( w = W_k \hat{w} \) for some \( \hat{w} \in \mathbb{R}^k \). We define \( \hat{T}_k = \Lambda_k T_k \). Then for this \( v \) and \( w \),
w^t A v = \hat{v}^t W_k^t A V_k \hat{v} = \hat{w}^t \hat{T}_k \hat{v},
so that (35) could be formulated directly in terms of \( \hat{T}_k \), although it is less convenient for the proof.

The inf-sup condition gives a lower bound on the (generalized) singular values of \( A \) and its restrictions to the subspaces \( V_k \) and \( W_k \). To see this, we first consider the case \( k = n \) for simplicity. A straightforward calculation shows that \( \delta_n \) is a lower bound on the generalized eigenvalues for
\begin{equation}
A^t M_k^{-1} A v = \lambda^2 M_k v
\end{equation}
or, equivalently,
\begin{equation}
A M_k^{-1} A^t w = \lambda^2 M_k w.
\end{equation}
If \( k < n \), a similar calculation shows \( \delta_k \) is a lower bound for the eigenvalues of

\[
\bar{T}_k(W_k^T M_k W_k)^{-1} \bar{T}_k \hat{v} = \lambda^2 (W_k^T M_k W_k) \hat{v}
\]

and

\[
\bar{T}_k(V_k^T M_k V_k)^{-1} \bar{T}_k \hat{w} = \lambda^2 (W_k^T M_k W_k) \hat{w}.
\]

Finally, the continuity condition (34) gives an upper bound on the eigenvalues in (42)-(43) (and (44)-(45) as well).

When \( A \) is symmetric and positive definite, a natural choice for \( M_r \) and \( M_t \) is

\[
M_r = M_t = A.
\]

Then one has trivially \( \Gamma = \delta = 1 \). Estimate (36) is not sharp for this case (but only by a factor of 2), as it does not make use of the additional minimization property present when \( A \) is symmetric and positive definite.

For the case of general nonsymmetric and indefinite \( A \), the situation is less clear. For any choice of \( M_r \), we can take \( M_t = A M_r^{-1} A^T \). This yields \( \Gamma = \delta = 1 \), but not necessarily simple estimates for \( \delta_k \). An obvious example of this type is

\[
M_r = (A^T A)^{1/2} \text{ and } M_t = (A A^T)^{1/2}.
\]

A potentially better choice is

\[
M_r = B^T W_n \Lambda_n^{-1/2} U_n^T \bar{D}_n \bar{D}_n U_n^{-1} W_n B
\]

\[
M_t = B^T W_n \Lambda_n^{-1/2} U_n^T \bar{D}_n \bar{D}_n U_n^{-1} V_n^T,
\]

where \( \bar{D}_n = \Lambda_n D_n \). With these definitions, \( \Gamma = \delta = 1 \) for all \( k \) for which \( x_k \) is defined. Therefore, in these two norms, each iterate \( x_k \) computed by CSBCG is "optimal" to within a factor of 2 in error. Note that

\[
A = B V_n \Lambda_n^{-1/2} \bar{D}_n \Lambda_n^{-1/2} W_n^T B
\]

so that when \( A \) and \( B \) are symmetric and positive definite, and \( V_n = W_n \) as in the conjugate gradient method, we have \( M_r = M_t = A \).

Let \( e_k = x - x_k \) and \( \bar{e}_k = \bar{x} - \bar{x}_k \) denote the error. Then standard manipulations [6] show that

\[
e_k = P_k (B^{-1} A) e_0
\]

where \( P_k \) is a polynomial of degree \( k \) such that \( P_k(0) = 1 \). An immediate consequence of Theorem 4.1 is

**Theorem 4.3.** Let \( e_k = x - x_k \) as above. Then

\[
\| e_k \|_r \leq (1 + \Gamma / \delta) \inf_{P_k} \| P_k (M_r^{1/2} B^{-1} A M_r^{-1/2}) \| \| e_0 \|_r
\]

where the inf is taken over all polynomials of degree \( k \) such that \( P_k(0) = 1 \), and \( \| \cdot \|_r \) is the usual \( \ell^r \) matrix norm.

**Proof.** Estimate (48) is an immediate consequence of theorem 4.1 and (47).

**Corollary 4.4.** Let \( \bar{e}_k = \bar{x} - \bar{x}_k \) as above. Then

\[
\| \bar{e}_k \|_\ell \leq (1 + \Gamma / \delta) \inf_{P_k} \| P_k (M_t^{1/2} B^{-1} A M_t^{-1/2}) \| \| \bar{e}_0 \|_\ell
\]

where the inf is taken over all polynomials of degree \( k \) such that \( P_k(0) = 1 \).

**Proof.** The proof is similar to theorem 4.3. \( \square \)

It doesn’t seem possible to derive any simple estimates for the rate of convergence without making further assumptions. For example, when \( A \) and \( B \) are symmetric and
positive definite and $M_r = A$, we must estimate $\| P_k(A^{1/2}B^{-1}A^{1/2}) \|$. The standard approach is to use Chebyshev polynomials and bounds for the generalized Rayleigh quotient $z^*Az/z^*Bz$. This leads to the estimate

$$\|\varepsilon_k\|_r \leq 4 \left( \frac{\sqrt{K} - 1}{\sqrt{K} + 1} \right)^k \|\varepsilon_0\|_r,$$

where $K$ is the (generalized) condition number of $A^{1/2}B^{-1}A^{1/2}$. This is the standard result [6], except for the factor 4, which is due to our use of theorem 4.1.

For the general case, we note that

$$\| P_k(M_r^{1/2}B^{-1}AM_r^{-1/2}) \| = \| M_r^{1/2}P_k(B^{-1}A)M_r^{-1/2} \|
\| (M_r^{1/2}V_0)P_k(T_0)(M_r^{1/2}V_0)^{-1} \|
$$

giving some alternative formulations which might prove useful in obtaining bounds. We note here the appearance of the nonsymmetric matrix $T_0$ rather than the symmetric matrix $T_0$. For example, if the eigenvalues of $T_0$ can be enclosed by an ellipse in the complex plane which does not contain the origin, then an estimate for the rate of convergence can again be made in terms scaled and translated Chebyshev polynomials in the complex plane as in Manteuffel [15], [16].

Since $T_0$ is real, its eigenvalues will be real or complex conjugate pairs. We assume that all eigenvalues lie strictly in the right half of the complex plane, so that their convex hull will not contain the origin. Suppose all the eigenvalues are enclosed in an ellipse centered at the point $d$ in the complex plane, with foci at $d \pm c$. We assume the ellipse does not contain the origin and that $\lambda$ is an eigenvalue of $T_0$ lying on the boundary of the given ellipse. By symmetry, we may assume that $d$ is real and that $c$ is either real or purely imaginary. Then Manteuffel’s estimates imply

$$\|\varepsilon_k\|_r \leq C \left| \frac{d - \lambda + (d - \lambda)^2 - c^2)^{1/2}}{d + (d^2 - c^2)^{1/2}} \right|^k \|\varepsilon_0\|_r,$$

where $C$ is a constant independent of $k$. Manteuffel gives an algorithm for computing optimal choices of the parameters $d$ and $c$ from knowledge of the convex hull of the spectrum of $T_0$. He used them as the basis of an adaptive Chebyshev acceleration algorithm, whereas we require them for theoretical purposes only, to improve our estimate for the rate of convergence of the composite step biconjugate gradient method.

5. Implementation. In this section we consider some practical aspects of the composite step biconjugate gradient algorithm. We assume that $A$, $B$, $r_0$, $x_0$, $z_0 = 0$, and $\varepsilon_0 = 0$ are given. An implementation of the composite step algorithm, based on
Algorithm CSBCG:

\[ \psi_0 = \| r_0 \| \]
\[ Bp_1 = r_0 / \psi_0; \quad B^T \tilde{p}_1 = \tilde{r}_0 / \psi_0 \]
\[ q_1 = A^T p_1; \quad \tilde{q}_1 = A^T \tilde{p}_1 \]
\[ \rho_1 = \tilde{r}_1 / r_0 \]
\[ k \leftarrow 1 \]

Begin LOOP:

\[ \sigma_k = B^{-1} q_k \]
\[ s_k = \sigma_k r_{k-1} - \rho_k q_k; \quad \tilde{s}_k = \sigma_k \tilde{r}_{k-1} - \rho_k \tilde{q}_k \]
\[ \xi_k = \| s_k \| \]
\[ Bz_{k+1} = s_k / \xi_k; \quad B^T \tilde{z}_{k+1} = \tilde{s}_k / \xi_k \]
\[ y_{k+1} = A z_{k+1}; \quad \tilde{y}_{k+1} = A^T \tilde{z}_{k+1} \]
\[ \theta_{k+1} = \tilde{z}_{k+1} / s_k \]
\[ \zeta_{k+1} = \tilde{z}_{k+1} / y_{k+1} \]

If 1 \times 1 step, Then

\[ \alpha_k = \rho_k / \sigma_k \]
\[ \rho_{k+1} = \tilde{h}_{k+1} / \sigma_k \]
\[ \beta_k = \rho_{k+1} / \rho_k \]
\[ x_k = x_{k-1} + \alpha_k p_k; \quad \tilde{x}_k = \tilde{x}_{k-1} + \alpha_k \tilde{p}_k \]
\[ r_k = r_{k-1} - \alpha_k q_k; \quad \tilde{r}_k = \tilde{r}_{k-1} - \alpha_k \tilde{q}_k \]
\[ \psi_k = \| r_k \| \]
\[ p_{k+1} = \tilde{z}_{k+1} + \beta_k \tilde{p}_k; \quad \tilde{p}_{k+1} = \tilde{z}_{k+1} + \beta_k \tilde{p}_k \]
\[ q_{k+1} = \tilde{y}_{k+1} + \beta_k q_k; \quad \tilde{q}_{k+1} = \tilde{y}_{k+1} + \beta_k \tilde{q}_k \]
\[ k \leftarrow k + 1 \]

Else

\[ \begin{bmatrix} \alpha_k \\ \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} \sigma_k & -\theta_{k+1} / \rho_k \\ -\theta_{k+1} / \rho_k & \zeta_{k+1} \end{bmatrix}^{-1} \begin{bmatrix} \rho_k \\ 0 \end{bmatrix} \]
\[ s_{k+1} = s_k + \alpha_k p_k + \alpha_{k+1} x_{k+1}; \quad \tilde{s}_{k+1} = \tilde{s}_{k-1} + \alpha_k \tilde{p}_k + \alpha_{k+1} \tilde{x}_{k+1} \]
\[ r_{k+1} = r_{k-1} - \alpha_k q_k - \alpha_{k+1} y_{k+1}; \quad \tilde{r}_{k+1} = \tilde{r}_{k-1} - \alpha_k \tilde{q}_k - \alpha_{k+1} \tilde{y}_{k+1} \]
\[ \psi_{k+1} = \| r_{k+1} \| \]
\[ Bz_{k+2} = r_{k+1} / \psi_{k+1}; \quad B^T \tilde{z}_{k+2} = \tilde{r}_{k+1} / \psi_{k+1} \]
\[ p_{k+2} = \tilde{z}_{k+2} + \beta_k \tilde{p}_k; \quad \tilde{p}_{k+2} = \tilde{z}_{k+2} + \beta_k \tilde{p}_k \]
\[ q_{k+2} = \tilde{y}_{k+2} + \beta_k q_k; \quad \tilde{q}_{k+2} = \tilde{y}_{k+2} + \beta_k \tilde{q}_k \]
\[ k \leftarrow k + 2 \]

End If

End LOOP

At this point we make more precise the correspondence between the more abstract matrix formulation of the CSBCG algorithm given in section 3 and that given here. For 2 \times 2 steps, the two direction vectors used by CSBCG are \( p_k \) and \( z_{k+1} \), (and \( \tilde{p}_k \) and \( \tilde{z}_{k+1} \)) the current BCG direction vectors and the next Lanczos vectors. In section 3, it was more convenient to call them simply \( p_k \) and \( p_{k+1} \) (respectively \( \tilde{p}_k \) and \( \tilde{p}_{k+1} \)), but here that notation would lead to some confusion. Similarly, for 2 \times 2 steps, the two residual vectors in \( R_k \) are denoted \( s_k \) and \( r_{k+1} \) rather than \( r_k \) and \( r_{k+1} \), and those in \( R_k \) are denoted by \( \tilde{s}_k \) and \( \tilde{r}_{k+1} \).
The residual update coefficients, denoted by $\alpha_k$ here, are computed using the third form of (32). These formulae are symmetrical with respect to their use of the vectors corresponding to the systems for $A$ and $A^T$, and reduce to the usual choices for the regular conjugate gradient method when $A$ and $B$ are symmetric and $r_0 = \tilde{r}_0$.

For $1 \times 1$ steps, the diagonal block of $\tilde{P}_k^T A P_k$ of (32) is given by $\sigma_k$ here, and the diagonal entry of $\tilde{R}_k^T B^{-1} R_k = \Lambda_k$ in (32) is given by $\rho_k$.

For $2 \times 2$ steps, the relevant $2 \times 2$ block of $\tilde{P}_k^T A P_k$ in (32) is given by

\[
\begin{bmatrix}
\tilde{p}_k^T A p_k & \tilde{p}_k^T A z_{k+1} \\
\tilde{z}_{k+1}^T A p_k & \tilde{z}_{k+1}^T A z_{k+1}
\end{bmatrix} = \begin{bmatrix}
\sigma_k & -\theta_{k+1}/\rho_k \\
-\theta_{k+1}/\rho_k & \zeta_{k+1}
\end{bmatrix}.
\]

The off diagonal entries of the $2 \times 2$ block are computed using the identity

\[
\begin{align*}
\tilde{p}_k^T A z_{k+1} &= \tilde{z}_{k+1}^T A p_k \\
&= \tilde{z}_{k+1}^T q_k \\
&= -\tilde{z}_{k+1}^T (s_k - \sigma_k r_{k-1})/\rho_k \\
&= -\theta_k/\rho_k.
\end{align*}
\]

The relevant part of the diagonal matrix $\tilde{R}_k^T B^{-1} R_k = \Lambda_k$ in (32) is given by

\[
\begin{bmatrix}
\tilde{r}_{k-1}^T B^{-1} r_{k-1} \\
\tilde{s}_{k-1}^T B^{-1} r_{k-1}
\end{bmatrix} = \begin{bmatrix}
\rho_k \\
0
\end{bmatrix}.
\]

The coefficients for the direction vector updates, here denoted by $\beta_k$, are given by the third form in (30), $\hat{U}_k = \Lambda_k^{-1} L_k^T A_k$; that is, as ratios of the diagonal elements of $A_k$. As with the coefficients $\alpha_k$, these are symmetrical formulae which reduce to the usual choice for the conjugate gradient algorithm in the symmetric case. For $2 \times 2$ steps we have

\[
\begin{bmatrix}
\beta_k \\
\beta_{k+1}
\end{bmatrix} = \begin{bmatrix}
\rho_{k+2}/\rho_k \\
\rho_{k+2}/\rho_{k+1}
\end{bmatrix}
\]

where we have (formally) made the identification $\rho_{k+1} = \theta_{k+1}/\sigma_k$.

Note that the vectors $s_k, \tilde{s}_k$ are scaled versions of $r_k, \tilde{r}_k$, respectively ($s_k = \sigma_k r_k$ and $\tilde{s}_k = \sigma_k \tilde{r}_k$) when $r_k$ and $\tilde{r}_k$ are defined for the $1 \times 1$ update. Thus for $1 \times 1$ updates, these vectors could be computed from simple rescaling, rather than from the more standard formula given in algorithm CSBGC. Also note the nonstandard update formulae for $q_{k+1}$ and $\tilde{q}_{k+1}$ for the case of a $1 \times 1$ step. These vectors are updated by recurrence relations rather than the more standard $q_{k+1} = A p_{k+1}$ in order to save a matrix multiplication. Either $y_{k+1}$ or $\tilde{y}_{k+1}$ is required to compute $\zeta_{k+1}$, the $(2, 2)$ element of the current $2 \times 2$ block. This element is typically needed in the process of deciding whether to use a $1 \times 1$ or $2 \times 2$ update. Using a recurrence relation for $q_{k+1}$ and $\tilde{q}_{k+1}$ allows us to recycle this matrix multiplication. Also note that in this implementation, a $1 \times 1$ step requires one multiplication by $A$ and one by $A^T$, and one preconditioning by $B$ and one by $B^T$. A $2 \times 2$ update requires two of each of these matrix operations, and approximately twice as many inner products and vector-scalar multiplications, so that the algorithm is balanced, in the sense that a $2 \times 2$ update costs approximately twice as much as a $1 \times 1$ update. Thus there is no significant efficiency advantage to be gained by choosing $1 \times 1$ or $2 \times 2$ steps. Finally, we note that when $A$ and $B$ are symmetric, and $r_0 = \tilde{r}_0$, then $x_k = \tilde{x}_k$, $p_k = \tilde{p}_k$,
etc., and algorithm CSBCG can be simplified to a composite step conjugate gradient algorithm, saving about half of the computational work.

Before preconditioning, we scale the right hand sides such that the vector preconditioned by $B$ has unit length. We do this in a simple attempt to keep the components of the direction vectors near the center of the floating point number range. The mathematical theory is clearly independent of such scalings.

We next consider the issue of deciding between $1 \times 1$ and $2 \times 2$ 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 $2 \times 2$ matrix

$$
\begin{bmatrix}
\sigma_k & -\theta_{k+1}/\rho_k \\
-\theta_{k+1}/\rho_k & \zeta_{k+1}
\end{bmatrix}
$$

and deciding locally whether to choose the $1 \times 1$ pivot $\sigma_k$ or to use the matrix itself as a $2 \times 2$ 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 $1 \times 1$ update satisfies $\| r_k \| \leq \| r_{k-1} \|$, then we choose a $1 \times 1$ update. Otherwise, we consider the (potential) residual $r_{k+1}$ for a $2 \times 2$ update, and choose a $2 \times 2$ update if $\| r_{k+1} \| < \| r_k \|$. We don’t directly compute $r_k$ and $r_{k+1}$ but rather scaled versions to guard against small pivots. Thus we have $\| r_k \| / \| \sigma_k \| = \xi_k$. $\| r_{k+1} \|$ is not immediately available, but we compute (as necessary) a scaled version, where the scaling factor is the determinant of the $2 \times 2$ pivot. The following code fragment implements our test:

If $\xi_k \leq \psi_{k-1} / \| \sigma_k \|$, Then

1 \times 1 Step

Else

$\delta_k = \sigma_k \xi_{k+1} - (\theta_{k+1} / \rho_k)^2$

$\nu_{k+1} = \| \delta_k r_{k-1} - \rho_k \xi_{k+1} q_k - \theta_{k+1} y_{k+1} \|

If $\nu_{k+1} / \| \sigma_k \| \leq \xi_k / \| \delta_k \|$, Then

2 \times 2 Step

Else

1 \times 1 Step

End If

End If

This test mathematically simplifies to choosing a $2 \times 2$ update when

$$
\| r_k \| > \max \{ \| r_{k-1} \|, \| r_{k+1} \| \}.
$$

When (50) is satisfied, taking two $1 \times 1$ steps would result in a “spike” in the convergence history of the residual norm. By making a $2 \times 2$ update in such circumstances, we effectively cut off such spikes. 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 in $T_k$.

A second implementation issue concerns the choice of basis vectors for the two dimensional subspaces used in $2 \times 2$ update steps. The "natural" choice is $(p_k, \delta_{k+1})$ and $(\tilde{p}_k, \tilde{\xi}_{k+1})$ that we used in algorithm CSBCG. This basis consists of the $k$-th direction vectors for the biconjugate gradient iteration, and the $(k+1)$-st Lanczos
vectors. However, there is clearly a great deal of freedom in choosing the basis for these spaces. One interesting class of basis vectors we have considered are those of the form \((p_k + \tau z_{k+1}, \bar{z}_{k+1} + \omega p_k)\) and \((\bar{p}_k + \tau \bar{z}_{k+1}, \bar{\bar{z}}_{k+1} + \omega \bar{p}_k)\), where \(\tau \neq \omega^{-1}\) is chosen such that the resulting \(2 \times 2\) matrix in Algorithm CSBCG will be diagonal.

This requires that
\[
(\bar{p}_k + \tau \bar{z}_{k+1})^T A (z_{k+1} + \omega p_k) = \tau \zeta_{k+1} + \omega \sigma_k - (1 + \tau \omega) \theta_{k+1} / \rho_k = 0
\]
giving a one parameter family of basis vectors.

One member of this family corresponds to the choice \(\tau = 0\), \(\omega = \theta_{k+1} / (\rho_k \sigma_k)\). For this choice, the basis vectors are \((p_k, p_{k+1})\) and \((\bar{p}_k, \bar{p}_{k+1})\), the direction vectors for the standard biconjugate gradient method. Using this choice of basis vectors, algorithm CSBCG becomes:

Algorithm CSBCG/BCG:

\[
\psi_0 = \| r_0 \|
\]

\[
B p_1 = r_0 / \psi_0; \quad B^T \bar{p}_1 = \bar{r}_0 / \psi_0
\]

\[
q_1 = A \bar{p}_1; \quad \bar{q}_1 = A^T p_1
\]

\[
\rho_1 = \bar{p}_1^T r_0
\]

\[
k = 1
\]

Begin LOOP:

\[
\sigma_k = \bar{p}_k^T q_k
\]

\[
\xi_k = \sigma_k \xi_{k-1} - \rho_k q_k; \quad \zeta_k = \sigma_k \zeta_{k-1} - \rho_k \bar{q}_k
\]

\[
\zeta_k = \| s_k \|
\]

\[
B z_{k+1} = s_k / \xi_k; \quad B^T \bar{z}_{k+1} = \bar{s}_k / \zeta_k
\]

\[
\theta_{k+1} = \bar{s}_k^T \xi_k
\]

\[
\rho_{k+1} = \theta_{k+1} / \sigma_k
\]

\[
\bar{p}_k = \rho_{k+1} / \rho_k
\]

\[
p_{k+1} = z_{k+1} + \beta_k p_k; \quad \bar{p}_{k+1} = \bar{z}_{k+1} + \beta_k \bar{p}_k
\]

\[
q_{k+1} = A \bar{p}_{k+1}; \quad \bar{q}_{k+1} = A^T \bar{p}_{k+1}
\]

If 1 \times 1 step, Then

\[
\alpha_k = \rho_k / \sigma_k
\]

\[
x_k = x_{k-1} + \alpha_k p_k; \quad \bar{x}_k = \bar{x}_{k-1} + \alpha_k \bar{p}_k
\]

\[
r_k = r_{k-1} - \alpha_k q_k; \quad \bar{r}_k = \bar{r}_{k-1} - \alpha_k \bar{q}_k
\]

\[
\psi_k = \| r_k \|
\]

\[
k \leftarrow k + 1
\]

Else

\[
\sigma_{k+1} = \bar{p}_{k+1}^T q_{k+1}
\]

\[
\begin{bmatrix}
\alpha_k \\
\alpha_{k+1}
\end{bmatrix} = \sigma_{k+1} = \begin{bmatrix}
\theta_{k+1} / \sigma_{k+1} \\
\rho_{k+1}
\end{bmatrix}
\]

\[
x_{k+1} = x_k + (\alpha_k p_k + \alpha_{k+1} p_{k+1}); \quad \bar{x}_{k+1} = \bar{x}_k + (\alpha_k \bar{p}_k + \alpha_{k+1} \bar{p}_{k+1})
\]

\[
r_{k+1} = r_k - (\alpha_k q_k + \alpha_{k+1} q_{k+1}); \quad \bar{r}_{k+1} = \bar{r}_k - (\alpha_k \bar{q}_k + \alpha_{k+1} \bar{q}_{k+1})
\]

\[
\psi_{k+1} = \| r_{k+1} \|
\]

\[
B z_{k+2} = r_{k+1} / \psi_{k+1}; \quad B^T \bar{z}_{k+2} = \bar{r}_{k+1} / \psi_{k+1}
\]

\[
\rho_{k+2} = \bar{s}_{k+1} / \rho_k
\]

\[
\theta_{k+1} = \rho_{k+2} / \rho_{k+1}
\]

\[
p_{k+2} = z_{k+2} + \beta_{k+1} p_{k+1}; \quad \bar{p}_{k+2} = \bar{z}_{k+2} + \beta_{k+1} \bar{p}_{k+1}
\]

\[
q_{k+2} = A \bar{p}_{k+2}; \quad \bar{q}_{k+2} = A^T \bar{p}_{k+2}
\]

\[
k \leftarrow k + 2
\]

End If

End LOOP
Initially, this may appear to be a poor choice. After all, $p_{k+1}$ and $\tilde{p}_{k+1}$ are computed using the small pivot $\sigma_k$, and it is the division by $\sigma_k$ we seek to avoid in making a $2 \times 2$ update. Furthermore, since (50) is satisfied for a $2 \times 2$ update, there is certain to be strong cancellation in the computation of $\alpha_k q_k + \alpha_{k+1} q_{k+1}$ in the $2 \times 2$ update step in algorithm CBBGG/BCG. At the moment we do not have a theoretical justification for this choice, but can only say that despite our own misgivings, empirically it has proven to be a very robust choice. We will present some empirical evidence of this in the next section. In any event, the simplifications afforded by this choice of basis vectors help make clear the connection between the CSBCG method and the standard biconjugate gradient method.

Another set of basis vectors corresponds to the choice $\omega = 0, \tau = \theta_{k+1}/(p_k \zeta_{k+1})$. We will call this the Look Ahead Lanczos basis. A version of algorithm CSBCG using this basis is given below.
Algorithm CSBCG/LAL:

\[
\begin{align*}
\psi_0 &= \| r_0 \|, \quad Bp_1 = r_0/\psi_0; \quad B^t \hat{p}_1 = \hat{r}_0/\psi_0 \\
q_1 &= Ap_1; \quad \hat{q}_1 = A^t \hat{p}_1 \\
\rho_1 &= \hat{p}_1^t r_0 \\
k &\leftarrow 1 \\
\begin{aligned}
\text{Begin LOOP:} \\
\sigma_k &= \hat{p}_k^t q_k \\
s_k &= \sigma_k r_{k-1} - \rho_k q_k; \quad \tilde{s}_k = \sigma_k \tilde{r}_{k-1} - \rho_k \tilde{q}_k \\
\xi_k &= \| s_k \| \\
Bz_{k+1} &= s_k/\xi_k; \quad B^t \tilde{z}_{k+1} = \tilde{s}_k/\xi_k \\
y_{k+1} &= Az_{k+1}; \quad \tilde{y}_{k+1} = A^t \tilde{z}_{k+1} \\
\theta_{k+1} &= \tilde{z}_{k+1}^t \xi_k \\
C_{k+1} &= \tilde{z}_{k+1} y_{k+1} \\
\text{If } 1 \times 1 \text{ step, then} \text{ Then} \\
\alpha_k &= \rho_k/\sigma_k \\
\rho_{k+1} &= \hat{p}_{k+1}^t s_k /\alpha_k \\
\beta_k &= \rho_{k+1} / \rho_k \\
\varphi_k &= \varphi_{k-1} + \alpha_k \rho_k; \quad \tilde{\varphi}_k = \tilde{\varphi}_{k-1} + \alpha_k \tilde{\rho}_k \\
r_k &= r_{k-1} - \alpha_k q_k; \quad \tilde{r}_k = \tilde{r}_{k-1} - \alpha_k \tilde{q}_k \\
\psi_k &= \| r_k \| \\
p_{k+1} &= s_{k+1} + \beta_k p_k; \quad \hat{p}_{k+1} = \tilde{s}_{k+1} + \beta_k \hat{p}_k \\
q_{k+1} &= y_{k+1} + \beta_k q_k; \quad \hat{q}_{k+1} = \tilde{y}_{k+1} + \beta_k \hat{q}_k \\
k &\leftarrow k + 1 \\
\text{Else} \text{ } \\
\tau_{k+1} &= \theta_{k+1} / (\rho_k \xi_{k+1}) \\
f_k &= p_k + \tau_{k+1} \tilde{z}_{k+1}; \quad \tilde{f}_k = \tilde{p}_k + \tau_{k+1} \tilde{z}_{k+1} \\
g_k &= q_k + \tau_{k+1} y_{k+1}; \quad \tilde{g}_k = \tilde{q}_k + \tau_{k+1} \tilde{y}_{k+1} \\
\mu_k &= f_k^t g_k \\
\alpha_k &= \rho_k / \mu_k \\
\varphi_k &= \varphi_{k-1} + \alpha_k f_k; \quad \tilde{\varphi}_k = \tilde{\varphi}_{k-1} + \alpha_k \tilde{f}_k \\
r_k &= r_{k-1} - \alpha_k g_k; \quad \tilde{r}_k = \tilde{r}_{k-1} - \alpha_k \tilde{g}_k \\
\psi_{k+1} &= \| r_{k+1} \| \\
Bz_{k+2} &= \tau_{k+1} / \psi_{k+1}; \quad B^t \tilde{z}_{k+2} = \tilde{r}_{k+1} / \psi_{k+1} \\
\rho_{k+2} &= \tilde{z}_{k+2}^t \xi_{k+1} \\
\begin{bmatrix} \rho_{k+2} / \rho_k \\ \rho_{k+2} / \rho_k \end{bmatrix} &= \begin{bmatrix} \rho_{k+2} / \rho_k \\ \rho_{k+2} / \rho_k \end{bmatrix} \\
p_{k+2} &= \tilde{p}_{k+2} + \beta_k \tilde{f}_k + \beta_{k+1} \tilde{z}_{k+1} \\
q_{k+2} &= Ap_{k+2}; \quad \hat{q}_{k+2} = A^t \hat{p}_{k+2} \\
k &\leftarrow k + 2 \\
\text{End If} \\
\text{End LOOP}
\end{aligned}
\]

This choice has several interesting properties. First, for $2 \times 2$ update steps, the residuals are updated using only one of the basis vectors. In some sense this minimizes the potential for cancellation, in contrast to algorithm CSBCG/BCG. Also for this choice, as well as other cases where $p_k$ and $\hat{p}_k$ are not chosen as basis vectors, the matrix $A_k T_k$ in (18) becomes truly block tridiagonal, with an additional off diagonal
entry (bulge) in the second co-diagonal band to mark each $2 \times 2$ update. This is a characteristic property of the Look Ahead Lanczos method, and helps make clear the connection between CSBCG and the Look Ahead Lanczos process [19], [17], [9].

6. The Effect Of Roundoff Error. In this section, we will present a few numerical results for the CSBCG methods discussed in section 5. Here we will focus mainly on one aspect of the numerical behavior, the properties of CSBCG with respect to roundoff error. In exact arithmetic, CSBCG computes selected iterates of BCG and hence has the same convergence rate as BCG. Several more general illustrations of the effectiveness of the CSBCG method are given in [4]. In [22], biconjugate gradient and many related methods [21], [23], [17], [8], [7], [9], [10] are compared on a series of test problems.

All of our 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 an adaptively created triangulation with 492 vertices [2] using continuous piecewise linear finite elements and Petrov-Galerkin methods based on the divergence-free upwinding scheme described in [3]. 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 $\beta = 100$, leading to a more difficult problem. Although many good pre conditioners are available for this problem, because we are interested in studying the effects of roundoff error, we have elected to have no pre conditioning ($B = I$).

For each problem ($\beta = 10$ and $\beta = 100$) we generated six different minimum degree orderings [11] of the equations using the minimum degree routine from [2]. Because of the wide variety of tie breaking strategies and the nonuniqueness of a minimum degree ordering, a minimum degree code called with a minimum degree ordering often won't recognize it as such, but instead will return with a different minimum degree ordering. This property was used to generate the six different orderings. One linear system differs from another by a permutation make $P$ which converts $Ax = r$ into $(PAP^T)(Px) = (Pr)$. Since $B = I$, such permutations can have no effect on the preconditioning. The only significant effects are on the ordering of the calculations in forming the products $Av$ and $A^T\tilde{v}$, and in the ordering of the sums in the inner products used in computing parameters for the algorithms. To enhance the effect of roundoff, all calculations were performed in single precision arithmetic, except where otherwise noted. All calculations were done on a DECstation 5000/240 using the standard F77 compiler.

We compared four different algorithms: the composite step methods CSBCG, CSBCG/BCG, and CSBCG/LAL as given in section 5, and the standard biconjugate gradient method (BCG). The algorithm BCG was implemented using the same code as for CSBCG/BCG, with the pivot test modified to always choose $1 \times 1$ update steps. Mathematically, the three CSBCG variants should produce identical iterates, and these should be a subset of the iterates produced by BCG. Any further differences must therefore be attributed to roundoff.

We chose to measure error in the $H^1(\Omega)$ norm, given by

$$\| u \|^2_{H^1} = \int_\Omega |\nabla u|^2 + u^2 \, dx.$$
If $U \in \mathbb{R}^n$ corresponds of the finite element function $u_h$, then there is an $n \times n$ (stiffness) matrix $M$ such that

$$
\int_{\Omega} |\nabla u_h|^2 + u_n^2 \, dx = U^t M U = \| U \|_M^2.
$$

We begin with the initial conditions $x_0 = \bar{x}_0 = 0$ and $r_0 = \bar{r}_0$. We iterated either 200 steps (where a step could be either $1 \times 1$ or $2 \times 2$) or until the error $x_k - x_{\infty}$ satisfied

$$
\| x_k - x_{\infty} \|_M \leq 10^{-4} \| x_{\infty} \|_M.
$$

We measured the number of correct digits by the formula

$$
\text{digits} = -\log_{10} \left\{ \frac{\| x_k - x_{\infty} \|_M}{\| x_{\infty} \|_M} \right\}.
$$

In Table 1, we record the results for the case $\beta = 10$.

**Table 1**

Results for $\beta = 10$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>BCG iterations</th>
<th>BCG digits</th>
<th>CSBCG iterations</th>
<th>CSBCG digits</th>
<th>CSBCG/BCG iterations</th>
<th>CSBCG/BCG digits</th>
<th>CSBCG/LAL iterations</th>
<th>CSBCG/LAL digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
<td>23/9</td>
<td>4.48</td>
</tr>
</tbody>
</table>

The index $i$ refers to different minimum degree orderings. For the composite step methods, under the column labeled “iterations”, we record the total number of steps, and the number of $2 \times 2$ steps. In this example, all the composite step methods took 23 steps, of which 9 were $2 \times 2$ steps, for the equivalent of $23 + 9 = 32$ steps of the standard biconjugate gradient method.

There is really not much surprise in these results. All six problems were the same system of equations up to the application of a permutation matrix. All methods did approximately the same amount of computation and all obtained essentially the same answers, the sole exception being the BCG method for the first ordering.

In Table 2, we present the results for the case $\beta = 100$, the more difficult problem.

Here we note that there are substantial differences in the behavior of the algorithms. In the cases when an algorithm took 200 steps without satisfying (51), we included, in parenthesis, the number of correct digits at the 200-th step. In these cases, the procedure had stalled sometime before step 200, and was no longer making significant progress towards a solution.

One rather striking feature of the results is the extent to which convergence depends on the ordering of the equations. Different orderings introduce different roundoff errors into the computation of inner products, which in turn influences the update coefficients based on those inner products. This seems to have affected all the algorithms.
Table 2  
Results for $\beta = 100$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>BCG iterations</th>
<th>digits</th>
<th>CSBCG iterations</th>
<th>digits</th>
<th>CSBCG/BCG iterations</th>
<th>digits</th>
<th>CSBCG/LAL iterations</th>
<th>digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>72</td>
<td>4.21</td>
<td>200/21 (2.97)</td>
<td></td>
<td>55/18</td>
<td>4.01</td>
<td>200/24 (1.49)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>77</td>
<td>4.05</td>
<td>200/28 (0.80)</td>
<td></td>
<td>46/19</td>
<td>4.07</td>
<td>200/22 (1.38)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>4.07</td>
<td>200/39 (-0.26)</td>
<td></td>
<td>56/24</td>
<td>4.26</td>
<td>200/17 (0.32)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>96</td>
<td>4.11</td>
<td>71/26</td>
<td>4.04</td>
<td>92/31</td>
<td>4.12</td>
<td>200/17 (1.53)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>4.16</td>
<td>200/17 (-0.15)</td>
<td></td>
<td>54/17</td>
<td>4.11</td>
<td>200/25 (-1.02)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>4.34</td>
<td>200/28 (-0.62)</td>
<td></td>
<td>54/20</td>
<td>4.15</td>
<td>200/0 (-1.78)</td>
<td></td>
</tr>
</tbody>
</table>

For this example, the standard BCG algorithm looks surprisingly good. Despite a very erratic and oscillatory convergence behavior, it is in fact working steadily toward convergence. This observation is consistent with the behavior of the BCG algorithm in the extensive tests of Tong in [22]. It is unknown to us how the iterates of the algorithm compare to those computed in exact arithmetic, but we doubt that they are close. On the other hand, failure to compute accurate direction vectors, poor approximation of a Krylov subspace, loss of orthogonality and near failures in the Lanczos process do not necessarily translate into failure of the BCG algorithm, since the only quantity of interest obtained form the calculation is the solution vector $x_k$. And it seems, at least in this case, the BCG does a good job of computing an approximation to $x$, while perhaps doing a poor job in other respects.

Among the CSBCG algorithms, the CSBCG/BCG variant seems to be the most robust on this problem. We attribute this to the fact that this implementation is closest to the standard BCG algorithm in its computation of subspaces, and whatever good properties are inherent in these spaces seem to be inherited by the CSBCG/BCG variant. On the other hand, the convergence history of the CSBCG/BCG algorithm, while not monotonic, does not have the severe oscillations of the BCG method; the spikes have been clipped by the criteria (50).

The other two CSBCG variants do not perform well on this problem. The reader should not infer from this that these algorithms are inferior. With good preconditioners, double precision arithmetic, etc., one would expect them to perform comparably to CSBCG/BCG. Since this is the scenario in which such procedures are typically used, the effects of roundoff error will tend to be minimized. In Table 3, we report the results for the second problem using double precision arithmetic.

Here we see that all methods solve all problems in the equivalent of 51-52 BCG steps. However, the convergence histories differ for different orderings, and between different variants for the same ordering, so roundoff error still is having some influence.

It is also likely that the behavior of all methods with respect to roundoff error could be improved. For example, in the CSBCG/LAL algorithm, the coefficient $\beta_{k+1}$ for $2 \times 2$ update steps is computed by the formula

$$\beta_{k+2} = \rho_{k+2}\mu_k/\theta_{k+1}. \tag{52}$$

We experimented with replacing (52) with the mathematically equivalent formula

$$\beta_{k+2} = \rho_{k+2}\sigma_k/\theta_{k+1} - \rho_{k+2}\theta_{k+1}/(\rho_{k+2}\mu_{k+1}). \tag{53}$$

leaving all other aspects of the implementation unchanged. The results of this single change, which affects only the $2 \times 2$ updates, are reported in Table 4.
Here we see a decided improvement in the behavior of CSBCG/LAL. We do not know if (53) is generally better than (52) with respect to roundoff error. Indeed, our initial intuition suggested that (52) would be superior. However, the point of this demonstration is less to suggest a particular algorithm or formula than it is to illustrate the sensitivity of BCG-like algorithms to roundoff error, with slightly different implementations producing drastically different results in situations where roundoff error plays a significant role. CSBCG algorithms are especially vulnerable because of the large number of choices one has in their implementation. One has the choice of criteria for deciding between $1 \times 1$ and $2 \times 2$ update steps, the choice of basis for the $2 \times 2$ updates, as well as a multitude of reasonable looking formulae for computing the update coefficients. As in the case of the BCG algorithm, it might be that the "best" algorithm is not one which necessarily produces the "correct" sequence of direction vectors, good approximation of the Krylov spaces, or even satisfies the biorthogonality properties best; it is the method which produces the best $a_k$ at the least cost.

REFERENCES


