The Block Conjugate Gradient Algorithm and Related Methods*

Dianne P. O'Leary[†]
Mathematics Department
University of Michigan
Ann Arbor, Michigan

Submitted by G. W. Stewart

ABSTRACT

The development of the Lanczos algorithm for finding eigenvalues of large sparse symmetric matrices was followed by that of block forms of the algorithm. In this paper, similar extensions are carried out for a relative of the Lanczos method, the conjugate gradient algorithm. The resulting block algorithms are useful for simultaneously solving multiple linear systems or for solving a single linear system in which the matrix has several separated eigenvalues or is not easily accessed on a computer. We develop a block biconjugate gradient algorithm for general matrices, and develop block conjugate gradient, minimum residual, and minimum error algorithms for symmetric semidefinite matrices. Bounds on the rate of convergence of the block conjugate gradient algorithm are presented, and issues related to computational implementation are discussed. Variants of the block conjugate gradient algorithm applicable to symmetric indefinite matrices are also developed.

1. INTRODUCTION

Conjugate direction algorithms are important tools in computational linear algebra. The basic idea behind these methods is to accumulate information about the behavior of a symmetric matrix A of dimension n along A-conjugate directions p_j, $j=1,2,\ldots,n$. The unique features of conjugate direction algorithms make them the ideal and, in some cases, the only useful methods available for solving certain systems of linear equations and

*This work was supported in part by the National Science Foundation under Grant MCS76-06595.

[†]Current address: Computer Science Department, University of Maryland, College Park, MD 20742.

eigenvalue problems involving large sparse matrices. Storage requirements are a modest multiple of n, and operations counts for each iteration are linearly related to n plus the number of nonzero elements in the matrix A. The methods are useful iterative algorithms but also have a finite termination property. Because of these desirable features, generalizations of these algorithms have also been extensively studied as solution techniques for nonlinear problems.

Conjugate direction algorithms were proposed by Fox, Huskey, and Wilkinson [8] for the solution of systems of linear equations on computers. Hestenes and Stiefel [14] developed a family of algorithms for positive definite matrices A in which the direction p_j is related to the residual of the system of linear equations after $j-1$ steps. These are termed conjugate gradient algorithms, and they are by far the most widely used of the conjugate direction algorithms. Karush [17], Kaniel [16], Daniel [6], and others studied the convergence rate of conjugate gradient algorithms considered as iterative methods, and further contributions were made by Stewart [33] and Axelsson [2]. Rutishauser [31] and Reid [28] discussed alternate computational forms of conjugate gradient algorithms, and much attention has been devoted to acceleration techniques, for example in [1, 2, 3, 4, 6, 13, 22]. Paige and Saunders [27], Fletcher [7], and Chandra [3] considered the stable extension of conjugate gradient techniques to symmetric indefinite matrices.

There has been a parallel development of conjugate direction algorithms for the solution of linear eigenvalue problems. Lanczos [18] proposed an algorithm in 1950. Paige [24] and Kaniel [16] established bounds on the convergence rate, and Paige [24–26] and Kahan and Parlett [15] discussed stable and efficient implementation of the algorithm to determine a few of the extreme eigenvalues of a matrix. Cullum and Donath [5] and Golub and Underwood [10, 34] extended the method to block form, in which several directions are used at once as blocks P_j of dimension $n\times s$. Computational modifications of this algorithm were investigated by Lewis [20].

For matrices which have certain eigenvalue distributions, the block Lanczos algorithm is often a dramatic improvement over the standard Lanczos algorithm. The idea of developing a block conjugate gradient algorithm which would share this virtue has undoubtably occurred to many researchers, and the statement of a block conjugate direction algorithm appears in a paper by Stewart [32].

The purpose of the present work is to present various forms of block biconjugate and conjugate gradient algorithms, investigate the convergence rate of some of them, and provide numerical experience with these methods. These algorithms can be useful in three classes of problems:

(1) If there are s systems to be solved, the block conjugate gradient algorithm will solve them in at most $\lceil n/s\rceil$ iterations and may involve less work than applying the conjugate gradient algorithm s times.

(2) If the matrix has several extreme eigenvalues widely separated from the others, the block conjugate gradient algorithm may converge significantly faster than the conjugate gradient algorithm.

(3) If the matrix A is stored on a secondary storage device or needs to be regenerated at every use, the block algorithm can be significantly more efficient, since it forms the product of A with several vectors at once.

These statements will be made more precise in Secs. 3 and 4.

In Sec. 2 we present the block biconjugate gradient algorithm with a preconditioning operator. This algorithm is defined for a general $n\times n$ matrix A without the assumption of symmetry. We discuss the properties of the algorithm and the role of scaling and orthogonalization in a computationally practical algorithm. All of the algorithms in this paper can be derived from this one. In Sec. 3 we specialize this algorithm in several forms for a symmetric matrix A, obtaining various block conjugate gradient algorithms. We discuss properties and variants of the algorithms. In Sec. 4 we give bounds on the convergence rate of the block conjugate gradient algorithm. The relation of these algorithms to the block Lanczos method and some extensions to symmetric indefinite systems are discussed in Sec. 5, and some implementation issues are mentioned in Sec. 6.

For simplicity, we state our results for matrices with real elements, although the generalization to complex matrices, and in many cases to general linear operators over Hilbert spaces, will be clear.

Throughout the paper, uppercase letters denote matrices. A superscript on a matrix, vector, or scalar denotes an iteration number. Columns of a matrix are indexed by subscript; elements of a matrix have row and column indices as subscripts. Thus, for example $x_{i,j}^{(k)}$ is the element of the matrix $X^{(k)}$ in the ith row and jth column, and $x_j^{(k)}$ is the jth column. The letters α, β, γ, ε, η, ν, ρ, and ω denote $s\times s$ matrices, and iteration numbers for these are indicated by subscript. The letter σ denotes singular values, while λ denotes eigenvalues, and a subscript "max" or "min" denotes largest or smallest respectively.

We use the Euclidean vector and matrix norms: $\|x\|^2=x^Tx$ and $\|X\|=\sigma_{\max}(X)=[\lambda_{\max}(X^TX)]^{1/2}$. The span of the columns of a set of matrices $\{X^{(1)},X^{(2)},\dots,X^{(k)}\}$ will be denoted by $\mathrm{sp}\{X^{(1)},X^{(2)},\dots,X^{(k)}\}$, and the trace of an $n\times n$ matrix A by $\mathrm{tr}[A]=\sum_{i=1}^n a_{ii}$.

2. THE BLOCK BICONJUGATE GRADIENT ALGORITHM

In this section we present a basic algorithm and its properties. The computationally practical algorithms in the following sections all arise from special cases of this one. An alternate approach to these algorithms, proceeding from the block Lanczos algorithm, is discussed in Sec. 5.

We consider the following algorithm, which is a direct generalization of the biconjugate gradient algorithm given, for example, in Lanczos [19] and Fletcher [7]. We call it the block biconjugate gradient algorithm, abbreviated by B-BCG.

ALGORITHM B-BCG. Given matrices M and A, both of dimension $n\times n$, full rank matrices γ_k and $\bar{\gamma}_k$ of dimension $s\times s$, $k=0,1,\ldots,$ and matrices $R^{(0)}$ and $\bar{R}^{(0)}$ of dimension $n\times s$ and rank s, we define $P^{(0)}=MR^{(0)}\gamma_0$ and $\bar{P}^{(0)}=M^T\bar{R}^{(0)}\bar{\gamma}_0$, and iterate for $k=0,1,\ldots$:

$$R^{(k+1)}=R^{(k)}-AP^{(k)}\alpha_k, \tag{1a}$$

$$\bar{R}^{(k+1)}=\bar{R}^{(k)}-A^T\bar{P}^{(k)}\bar{\alpha}_k, \tag{1b}$$

$$P^{(k+1)}=\left(MR^{(k+1)}+P^{(k)}\beta_k\right)\gamma_{k+1}, \tag{2a}$$

$$\bar{P}^{(k+1)}=\left(M^T\bar{R}^{(k+1)}+\bar{P}^{(k)}\bar{\beta}_k\right)\bar{\gamma}_{k+1}, \tag{2b}$$

where

$$\alpha_k=\left(\bar{P}^{(k)T}AP^{(k)}\right)^{-1}\bar{\gamma}_k^T\bar{R}^{(k)T}MR^{(k)}, \tag{3a}$$

$$\bar{\alpha}_k=\left(P^{(k)T}A^T\bar{P}^{(k)}\right)^{-1}\gamma_k^TR^{(k)T}M^T\bar{R}^{(k)}, \tag{3b}$$

$$\beta_k=\gamma_k^{-1}\left(\bar{R}^{(k)T}MR^{(k)}\right)^{-1}\bar{R}^{(k+1)T}MR^{(k+1)}, \tag{4a}$$

$$\bar{\beta}_k=\bar{\gamma}_k^{-1}\left(R^{(k)T}M^T\bar{R}^{(k)}\right)^{-1}R^{(k+1)T}M^T\bar{R}^{(k+1)}. \tag{4b}$$

The matrices R, $\bar{R}$, P, and $\bar{P}$ have dimension $n\times s$, while the parameters α, $\bar{\alpha}$, β, and $\bar{\beta}$ are $s\times s$ matrices.

The algorithm is terminated when one of the matrices α_k, $\bar{\alpha}_k$, β_k, or $\bar{\beta}_k$ fails to exist or is singular. This algorithm reduces to the standard biconjugate gradient algorithm when $s=1$, $M=I$, and $\gamma_k=\bar{\gamma}_k=1$ for $k=0,1,\ldots.$ To use the block algorithm to solve a linear system of equations

$$AX^*=B,$$

where B is a given $n\times s$ matrix and X^* is to be determined, we choose an initial guess $X^{(0)}$ for the solution matrix, set $R^{(0)}=B-AX^{(0)}$, and update the X iterates as

$$X^{(k+1)}=X^{(k)}+P^{(k)}\alpha_k.$$

Then the matrices $R^{(k)}$ are the successive residuals of the equations; i.e.,

$$R^{(k)} = B - AX^{(k)}, \qquad k = 0, 1, \ldots.$$

In a similar way, the algorithm can be used to simultaneously solve a system $A^T\bar{X}^* = \bar{B}$.

The iterates $R^{(k)}$, $\bar{R}^{(k)}$, $X^{(k)}$, and $\bar{X}^{(k)}$ are invariant with respect to the choice of nonsingular matrices γ_k and $\bar{\gamma}_k$, and these parameter matrices are used to decrease roundoff in computational implementations of the algorithm. On the other hand, various choices of the matrix M yield different algorithms, and can change the rate of convergence of the sequence $\{X^{(k)}\}$ to X^*.

Some algebraic properties of the algorithm are established in the following two lemmas. This development parallels that of Fletcher [7] for the standard biconjugate gradient algorithm.

LEMMA 1. *For $j<k$, the iterates satisfy the biconjugacy conditions*

$$\bar{R}^{(k)T}MR^{(j)} = R^{(k)T}M^T\bar{R}^{(j)} = 0 \tag{5}$$

and

$$\bar{P}^{(k)T}AP^{(j)} = P^{(k)T}A^T\bar{P}^{(j)} = 0. \tag{6}$$

Proof. We use induction to establish the results. The trivial case is obvious, so we assume the properties hold for k and prove them for $k+1$.

Using Eq. (1b) and Eq. (2a) with j substituted for $k+1$, we have

$$\begin{aligned}\bar{R}^{(k+1)T}MR^{(j)} &= \bar{R}^{(k)T}MR^{(j)} - \bar{\alpha}_k^T\bar{P}^{(k)T}AMR^{(j)}\\ &= \bar{R}^{(k)T}MR^{(j)} - \bar{\alpha}_k^T\bar{P}^{(k)T}A\left(P^{(j)}\gamma_j^{-1} - P^{(j-1)}\beta_{j-1}\right)\end{aligned}$$

For $j<k$, the right hand side is zero by the induction hypotheses. For $j=k$, it is zero by the induction hypothesis and the definition of $\bar{\alpha}_k$. An exactly analogous argument, using Eqs. (1a) and (2b) and the definition of $\bar{\alpha}_k$, establishes the second part of Eq. (5).

The first part of Eq. (6) is established using Eq. (2b) and Eq. (1a) with j substituted for k:

$$\begin{aligned}\bar{P}^{(k+1)T}AP^{(j)} &= \bar{\gamma}_{k+1}^T\bar{R}^{(k+1)T}MAP^{(j)} + \bar{\gamma}_{k+1}^T\bar{\beta}_k^T\bar{P}^{(k)T}AP^{(j)}\\ &= -\bar{\gamma}_{k+1}^T\bar{R}^{(k+1)T}M(R^{(j+1)} - R^{(j)})\alpha_j^{-1} + \bar{\gamma}_{k+1}^T\bar{\beta}_k^T\bar{P}^{(k)T}AP^{(j)}.\end{aligned}$$

For $j<k$, we have already established that the first term is zero, and the second term is zero by the induction hypothesis. For $j=k$, we have, using the definitions of α_k and $\bar{\beta}_k$,

$$\begin{aligned}\bar{P}^{(k+1)T}AP^{(k)} &= -\bar{\gamma}_{k+1}^{T}\bar{R}^{(k+1)T}MR^{(k+1)}\alpha_k^{-1}+\bar{\gamma}_{k+1}^{T}\bar{\beta}_k^{T}\bar{P}^{(k)T}AP^{(k)}\\ &= -\bar{\gamma}_{k+1}^{T}\bar{R}^{(k+1)T}MR^{(k+1)}\alpha_k^{-1}\\ &\quad+\bar{\gamma}_{k+1}^{T}\left(R^{(k+1)T}M^{T}\bar{R}^{(k+1)}\right)^{T}\left(R^{(k)T}M^{T}\bar{R}^{(k)}\right)^{-T}\bar{\gamma}_k^{-T}\bar{P}^{(k)T}AP^{(k)}\\ &= \bar{\gamma}_{k+1}^{T}\bar{R}^{(k+1)T}MR^{(k+1)}\left(-\alpha_k^{-1}+\alpha_k^{-1}\right)=0.\end{aligned}$$

The second part of Eq. (6) is established by an analogous argument, using Eqs. (2a) and (1b), and the definitions of $\bar{\alpha}_k$ and β_k. ■

LEMMA 2. *Some further properties of the algorithm are as follows:*

$$R^{(k)T}M^{T}A^{T}\bar{P}^{(k)}=\gamma_k^{-T}P^{(k)T}A^{T}\bar{P}^{(k)}, \tag{7a}$$

$$\bar{R}^{(k)T}MAP^{(k)}=\bar{\gamma}_k^{-T}\bar{P}^{(k)T}AP^{(k)}, \tag{7b}$$

$$R^{(k)T}\bar{P}^{(j)}=0,\qquad j<k, \tag{8a}$$

$$\bar{R}^{(k)T}P^{(j)}=0,\qquad j<k, \tag{8b}$$

$$R^{(k)T}M^{T}\bar{R}^{(k)}=R^{(k)T}\bar{P}^{(k)}\bar{\gamma}_k^{-1}, \tag{9a}$$

$$\bar{R}^{(k)T}MR^{(k)}=\bar{R}^{(k)T}P^{(k)}\gamma_k^{-1}. \tag{9b}$$

Proof. We will establish (7a), (8a), and (9a). The other results follow similarly.

From Eq. (2a), with k substituted for $k+1$, and from Eq. (6),

$$\begin{aligned}R^{(k)T}M^{T}A^{T}\bar{P}^{(k)} &= \left(\gamma_k^{-T}P^{(k)T}-\beta_{k-1}^{T}P^{(k-1)T}\right)A^{T}\bar{P}^{(k)}\\ &= \gamma_k^{-T}P^{(k)T}A^{T}\bar{P}^{(k)},\end{aligned}$$

and thus (7a) is established.

Equation (8a) is established by observing that repeated use of the definition of $\bar{P}$ yields

$$\bar{P}^{(j)}=M^{T}\left(\bar{R}^{(j)}\bar{\gamma}_j+\bar{R}^{(j-1)}\bar{\gamma}_{j-1}\bar{\beta}_{j-1}\bar{\gamma}_j+\cdots+\bar{R}^{(0)}\bar{\gamma}_0\bar{\beta}_0\cdots\bar{\gamma}_{j-1}\bar{\beta}_{j-1}\bar{\gamma}_j\right), \tag{10}$$

and so, by Eq. (5), $R^{(k)T}\bar{P}^{(j)}=0$ for $j<k$.

Using Eq. (2b) with k substituted for $k+1$, and Eq. (8a), we have

$$R^{(k)T}M^T\bar{R}^{(k)}=R^{(k)T}\left(\bar{P}^{(k)}\bar{\gamma}_k^{-1}-\bar{P}^{(k-1)}\bar{\beta}_{k-1}\right)=R^{(k)T}\bar{P}^{(k)}\bar{\gamma}_k^{-1},$$

which is Eq. (9a). ■

These algebraic properties can now give us some insight into the use of the algorithm to solve the equation $AX^*=B$.

Theorem 1. *The columns of the matrix $R^{(k)}$ are orthogonal to*

$$\operatorname{sp}\left\{M^T\bar{R}^{(0)},(M^TA^T)M^T\bar{R}^{(0)},\ldots,(M^TA^T)^{k-1}M^T\bar{R}^{(0)}\right\},$$

and thus, if the algorithm does not terminate before $\bar{k}=\lceil n/s\rceil$ steps, $R^{(\bar{k})}=0$. Similarly, the columns of $\bar{R}^{(k)}$ are orthogonal to $\operatorname{sp}\{MR^{(0)},(MA)MR^{(0)},\ldots,(MA)^{k-1}MR^{(0)}\}$, and $\bar{R}^{(\bar{k})}=0$.

Proof. From Eq. (10), under the assumption that all of the parameter matrices are nonsingular, it is clear that

$$\operatorname{sp}\{\bar{P}^{(0)},\ldots,\bar{P}^{(k)}\}=\operatorname{sp}\{M^T\bar{R}^{(0)},\ldots,M^T\bar{R}^{(k)}\}. \tag{11}$$

By the definition of $\bar{R}^{(k)}$,

$$\bar{R}^{(k)}\in\operatorname{sp}\{\bar{R}^{(k-1)},A^T\bar{P}^{(k-1)}\}.$$

Using these two facts, a simple induction argument shows that

$$\operatorname{sp}\{M^T\bar{R}^{(0)},\ldots,M^T\bar{R}^{(k-1)}\}=\operatorname{sp}\left\{M^T\bar{R}^{(0)},\ldots,(M^TA^T)^{k-1}M^T\bar{R}^{(0)}\right\},$$

and this, plus the biconjugacy condition (5), establishes the first conclusion. If the parameter matrices exist, then the dimension of the kth of these subspaces is ks, and thus at $\bar{k}$ steps, $R^{(\bar{k})}$ must be orthogonal to every nonzero n-vector, and therefore must be zero. The result for $\bar{R}^{(\bar{k})}$ is established in a similar way. ■

The block (or standard) biconjugate gradient algorithm breaks down in theory if $\bar{R}^{(k)T}MR^{(k)}$ or $\bar{P}^{(k)T}AP^{(k)}$ is singular (zero) for some $k<\bar{k}$. In practice, failure also occurs if any of these matrices is ill conditioned. In that case, any roundoff errors in the computation may be magnified greatly, and the parameter matrices will be calculated inaccurately. In hopes of postponing this catastrophe, precautions can be taken; for example, B and $\bar{B}$ can be normalized so that all columns of $R^{(0)}$ and $\bar{R}^{(0)}$ have the same size.

Failure is inevitable if at any stage one of the matrices $P^{(k)}$ or $\bar{P}^{(k)}$ fails to have full rank. This can be monitored by choosing matrices γ_k and $\bar{\gamma}_k$ which orthonormalize the columns of $P^{(k)}$ and $\bar{P}^{(k)}$. A QR algorithm or a modified Gram-Schmidt algorithm might be performed on the matrices $MR^{(k)}+P^{(k-1)}\beta_{k-1}$ and $M^T\bar{R}^{(k)}+\bar{P}^{(k-1)}\bar{\beta}_{k-1}$, and the resulting orthonormal matrices used as $P^{(k)}$ and $\bar{P}^{(k)}$ respectively. When this orthogonalization procedure produces a matrix of less than full rank, we must restart the block biconjugate gradient algorithm.

3. SOME BLOCK CONJUGATE GRADIENT ALGORITHMS FOR SYMMETRIC POSITIVE DEFINITE MATRICES

As mentioned in the previous section, the block biconjugate gradient algorithm breaks down if one of the parameter matrices becomes singular or undefined before the matrix $R^{(k)}=0$. For matrices A and M which are symmetric and positive definite, however, there are block algorithms which cannot fail. With special choices of the matrix $\bar{R}^{(0)}$ and reduction of the blocksize if linear dependence arises, we can develop algorithms which always terminate successfully with a zero residual matrix. Two choices of the initial matrix $\bar{R}^{(0)}$ give particularly useful algorithms.

If $\bar{R}^{(0)}=R^{(0)}$, the B-BCG algorithm reduces to a block conjugate gradient (B-CG) algorithm.

ALGORITHM B-CG VERSION 1 (Hestenes and Stiefel form). Given $X^{(0)}$, let $R^{(0)}=B-AX^{(0)}$, define $P^{(0)}=MR^{(0)}\gamma_0$, and for $k=0,1,\ldots,$ update the iterates, residuals, and directions:

$$X^{(k+1)}=X^{(k)}+P^{(k)}\alpha_k,$$

$$R^{(k+1)}=R^{(k)}-AP^{(k)}\alpha_k,$$

$$P^{(k+1)}=\left(MR^{(k+1)}+P^{(k)}\beta_k\right)\gamma_{k+1},$$

where

$$\alpha_k=\left(P^{(k)T}AP^{(k)}\right)^{-1}\gamma_k^TR^{(k)T}MR^{(k)},$$

$$\beta_k=\gamma_k^{-1}\left(R^{(k)T}MR^{(k)}\right)^{-1}R^{(k+1)T}MR^{(k+1)}.$$

A version of the B-CG algorithm similar to this one has been developed independently by Richard R. Underwood, working from the block Lanczos algorithm rather than block biconjugate gradients.

Notice that as long as the matrices $P^{(k)}$ and $R^{(k)}$ retain full rank, the algorithm is well defined. By Eq. (11) of the previous section, for each value of k, the ranks of these two matrices are equal. Thus we can monitor the stability of the algorithm by calculating the matrices γ_k through an orthogonalization procedure. If the columns of $P^{(k)}$ lose their independence, we delete the zero or redundant column j of $P^{(k)}$ and the corresponding columns of $X^{(k)}$ and $R^{(k)}$, and continue the algorithm with $s-1$ vectors. The vectors $x_j^{(k)}$ and $r_j^{(k)}$ can be updated separately, and the resulting sequences retain all of the properties necessary to guarantee convergence.

Properties of the B-CG algorithm are derived as special cases of the results in the previous section, and are summarized in the following theorem, which also presents a minimization property which is a simple consequence of these results.

THEOREM 2. *For the block conjugate gradient algorithm,*

$$R^{(k)T}MR^{(j)}=0, \qquad j\neq k,$$

$$P^{(k)T}AP^{(j)}=0, \qquad j\neq k,$$

$$R^{(k)T}P^{(j)}=0, \qquad j\neq k,$$

$$R^{(k)T}MAP^{(k)}=\gamma_k^{-T}P^{(k)T}AP^{(k)},$$

$$R^{(k)T}MR^{(k)}=R^{(k)T}P^{(k)}\gamma_k^{-1}.$$

$R^{(k)}$ *is orthogonal to* $\operatorname{sp}\{MR^{(0)},(MA)MR^{(0)},\dots,(MA)^{k-1}MR^{(0)}\}$, *and thus* $X^{(k)}$ *minimizes* $\operatorname{tr}[(X-X^*)^TA(X-X^*)]$ *over all* X *such that* $X-X^{(0)}\in \operatorname{sp}\{MR^{(0)},\dots,(MA)^{k-1}MR^{(0)}\}$.

A second algorithm is obtained if we make the choice $\tilde{R}^{(0)}=AMR^{(0)}$. We call this the block minimum residual (B-MR) algorithm.

ALGORITHM B-MR. Given $X^{(0)}$, let $R^{(0)}=B-AX^{(0)}$ and $P^{(0)}=MR^{(0)}\gamma_0$, and for $k=0,1,\dots,$ update the iterates, residuals, and directions:

$$X^{(k+1)}=X^{(k)}+P^{(k)}\alpha_k,$$

$$R^{(k+1)}=R^{(k)}-AP^{(k)}\alpha_k,$$

$$P^{(k+1)}=\left(MR^{(k+1)}+P^{(k)}\beta_k\right)\gamma_{k+1},$$

where

$$\alpha_k = (P^{(k)T}AMAP^{(k)})^{-1}\gamma_k^T R^{(k)T}MAMR^{(k)},$$

$$\beta_k = \gamma_k^{-1}(R^{(k)T}MAMR^{(k)})^{-1}R^{(k+1)T}MAMR^{(k+1)}.$$

The properties of this algorithm are summarized below.

THEOREM 3. *For the block minimum residual algorithm,*

$$R^{(k)T}MAMR^{(j)} = 0, \qquad j \neq k,$$

$$P^{(k)T}AMAP^{(j)} = 0, \qquad j \neq k,$$

$$R^{(k)T}MAP^{(j)} = 0, \qquad j \neq k,$$

$$R^{(k)T}MAMAP^{(k)} = \gamma_k^{-T}P^{(k)T}AMAP^{(k)},$$

$$R^{(k)T}MAMR^{(k)} = R^{(k)T}MAP^{(k)}\gamma_k^{-1},$$

and $X^{(k)}$ *minimizes* $\mathrm{tr}[(X-X^*)^TAMA(X-X^*)]$ *over all* X *such that* $X-X^{(0)} \in \mathrm{sp}\{MR^{(0)},\ldots,(MA)^{k-1}MR^{(0)}\}$.

Since $[(X^{(k)}-X^*)^TAMA(X^{(k)}-X^*)] = R^{(k)T}MR^{(k)}$, the name block minimum residual algorithm is appropriate.

Other algorithms in this family can be derived for $M=I$ by setting $\bar{R}^{(0)}$ equal to a polynomial in SA times $R^{(0)}$, where S is a symmetric matrix which commutes with A. Numerically, such algorithms often do not perform as well as the standard conjugate gradient algorithm because the use of powers of the matrix in inner products can introduce instability if SA is poorly conditioned. Thus the conjugate gradient algorithm is more popular, and in the rest of this section we consider alternate forms of the block conjugate gradient algorithm only. Alternate forms of the other algorithms would in many cases be derived analogously.

Rutishauser [31] developed a three term recurrence relation form of the conjugate gradient algorithm. This form is not as efficient computationally, because it takes more storage and more operations per iteration, but it is interesting theoretically because it clarifies the relation between the conjugate gradient algorithm and other second order algorithms such as the Chebyshev semiiterative or the Richardson second order method. All of these second order algorithms can be written in the form

$$r^{(k+1)} = r^{(k)} + \left(-AMr^{(k)} + \left(r^{(k)} - r^{(k-1)}\right)\eta_{k-1}\right)\omega_{k+1},$$

where η_{k-1} and ω_{k+1} are scalar parameters which vary from method to method, and $\eta_{-1}=0$. The R-matrices in the block conjugate gradient algorithm satisfy a similar relation. To see this, without loss of generality, take $\gamma_k=I$ and, following Reid [28], notice that the definition of $P^{(k)}$ implies

$$AP^{(k)}=AMR^{(k)}+AP^{(k-1)}\beta_{k-1}.$$

Using this and the definition of $R^{(k+1)}$, we get

$$(-R^{(k+1)}+R^{(k)})\alpha_k^{-1}=AP^{(k)}=AMR^{(k)}+AP^{(k-1)}\beta_{k-1}.$$

Substituting for $AP^{(k-1)}$ in this expression, using the definition of $R^{(k)}$, and simplifying yields

$$R^{(k+1)}=R^{(k)}+\left[-AMR^{(k)}+(R^{(k)}-R^{(k-1)})\eta_{k-1}\right]\omega_{k+1},$$

where $\eta_{k-1}=\alpha_{k-1}^{-1}\beta_{k-1}$ and $\omega_{k+1}=\alpha_k$. Thus,

$$\eta_{k-1}=\omega_k^{-1}\left(R^{(k-1)T}MR^{(k-1)}\right)^{-1}R^{(k)T}MR^{(k)}.$$

We eliminate the matrices $P^{(k)}$ from the definition of ω_k through some algebraic manipulation. From the definition of $P^{(k)}$,

$$\begin{aligned}R^{(k)T}MAMR^{(k)}&=\left(P^{(k)}-P^{(k-1)}\beta_{k-1}\right)^T A\left(P^{(k)}-P^{(k-1)}\beta_{k-1}\right)\\&=P^{(k)T}AP^{(k)}+\beta_{k-1}^T P^{(k-1)T}AP^{(k-1)}\beta_{k-1}.\end{aligned}$$

Thus

$$\begin{aligned}\left(R^{(k)T}MR^{(k)}\right)^{-1}R^{(k)T}MAMR^{(k)}&=\alpha_k^{-1}+\left(R^{(k-1)T}MR^{(k-1)}\right)^{-1}P^{(k-1)T}AP^{(k-1)}\beta_{k-1}\\&=\omega_{k+1}^{-1}+\alpha_{k-1}^{-1}\beta_{k-1}=\omega_{k+1}^{-1}+\eta_{k-1}.\end{aligned}$$

This gives us an alternate form of the block conjugate gradient algorithm.

Algorithm B-CG Version 2 (Rutishauser form). Given $X^{(0)}$, define $R^{(0)}=B-AX^{(0)}$, $\eta_{-1}=0$, and for $k=0,1,\dots,$ update the iterates and residuals

$$X^{(k+1)}=X^{(k)}+\left[MR^{(k)}-(X^{(k)}-X^{(k-1)})\eta_{k-1}\right]\omega_{k+1},$$

$$R^{(k+1)}=R^{(k)}+\left[-AMR^{(k)}+(R^{(k)}-R^{(k-1)})\eta_{k-1}\right]\omega_{k+1},$$

where

$$\omega_{k+1}^{-1}=\left(R^{(k)T}MR^{(k)}\right)^{-1}R^{(k)T}MAMR^{(k)}-\eta_{k-1},$$

$$\eta_k=\omega_{k+1}^{-1}\left(R^{(k)T}MR^{(k)}\right)^{-1}R^{(k+1)T}MR^{(k+1)},\qquad k\geqslant 0.$$

The form of this algorithm suggests investigating alternate choices of the parameters η_k and ω_k in order to obtain block forms of other second order methods, but this idea will not be pursued here.

Using this three term recurrence form of the B-CG algorithm, a special form of the algorithm could be derived, as in Reid [29], which is useful for problems of the form

$$A=\begin{pmatrix} A_1 & A_3 \\ A_3^T & A_2 \end{pmatrix},\qquad M^{-1}=\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix},$$

where A_1 and A_2 are square nonsingular matrices. This algorithm uses a special initial guess in order to reduce the computational work to less than half that for the other versions of the algorithms. The algorithms of Chandra [3] and Hageman, Luk, and Young [12] for this problem can be extended similarly.

In some problems, computational savings can be achieved by a change of variables. If, for example, it is expensive to form A times a vector, but there is a matrix M such that forming M and AM times a vector is efficient, an appropriate change of variables is $Y=M^{-1}X$. In terms of these variables, the conjugate gradient algorithm is given as follows:

Algorithm B-CG Version 3 (Change of variables form). Given $Y^{(0)}$, let $R^{(0)}=B-AMY^{(0)}$ and $P^{(0)}=R^{(0)}\gamma_0$, and for $k=0,1,\ldots$, update the transformed variables, the residual, and the directions:

$$Y^{(k+1)}=Y^{(k)}+P^{(k)}\alpha_k,$$

$$R^{(k+1)}=R^{(k)}-AMP^{(k)}\alpha_k,$$

$$P^{(k+1)}=\left(R^{(k+1)}+P^{(k)}\beta_k\right)\gamma_{k+1},$$

where

$$\alpha_k=\left(P^{(k)T}MAMP^{(k)}\right)^{-1}\gamma_k^TR^{(k)T}MR^{(k)},$$

$$\beta_k=\gamma_k^{-1}\left(R^{(k)T}MR^{(k)}\right)^{-1}R^{(k+1)T}MR^{(k+1)}.$$

Upon termination, compute $X^{(k+1)}=MY^{(k+1)}$.

In the case of a single vector, this algorithm and applications are discussed, for example, in [1] and [23].

Other versions of the block conjugate gradient algorithms are discussed in Sec. 5.

4. CONVERGENCE ANALYSIS OF THE BLOCK CONJUGATE GRADIENT ALGORITHM

In this section we establish results on the rate of convergence of the block conjugate gradient algorithm for positive definite matrices. The tools used are certain properties of Chebyshev polynomials and the fact that, of all algorithms which form

$$x_i^{(k)} = x_i^{(0)} + \sum_{j=1}^{s} \mathcal{P}_{kij}(MA)MAe_j^{(0)}, \qquad i=1,2,\dots,s, \tag{12}$$

where $\mathcal{P}_{kij}(\lambda)$ is a polynomial of degree less than or equal to $k-1$ and $e_j^{(0)} = x_j^{(0)} - x_j^*$, the B-CG algorithm is optimal in the sense of minimizing a certain measure of the error.

For simplicity in the presentation, we work with the unscaled algorithm $(M=I)$ first, and then generalize the results. First we summarize the facts we need concerning Chebyshev polynomials.

LEMMA 3. *Let*

$$\phi_k(\lambda) = \mathcal{T}_k\left(\frac{d_2+d_1-2\lambda}{d_2-d_1}\right) \Big/ \mathcal{T}_k\left(\frac{d_2+d_1}{d_2-d_1}\right),$$

where $0<d_1<d_2$ *and* $\mathcal{T}_k(x)=\cos(k\arccos x)$ *for* $-1\leqslant X\leqslant 1$ *is the* kth *Chebyshev polynomial of the first kind. Then*

(a) *For* $0\leqslant\lambda\leqslant d_1$, *we have* $0<\phi_k(\lambda)\leqslant 1$.

(b) *For* $d_1\leqslant\lambda\leqslant d_2$, *we have*

$$|\phi_k(\lambda)|\leqslant 2\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^k,$$

where $\kappa = d_2/d_1$.

Proof. (a): It is well known that the polynomials $\mathcal{T}_k(x)$ and $\mathcal{T}_k'(x)$ have all of their roots within the interval $-1<x<1$, and obviously, $\mathcal{T}_k(1)=1$. Also, $\mathcal{T}_k(x)$ is a positive monotone increasing function for $1\leqslant x<\infty$. Now

$$\phi_k(\lambda) = \mathcal{T}_k(x) \Big/ \mathcal{T}_k\left(\frac{d_2+d_1}{d_2-d_1}\right), \qquad \text{where} \quad x=\frac{d_2+d_1-2\lambda}{d_2-d_1},$$

so $\phi_k(\lambda)$ is monotone decreasing for $x>1$. The range $0\leqslant\lambda\leqslant d_1$ corresponds to $(d_2+d_1)/(d_2-d_1)\geqslant x\geqslant 1$, and $\phi_k(0)=1$. Thus the first conclusion follows.

(b): It is also well known that $|\mathcal{T}_k(x)|\leqslant 1$ for $-1\leqslant x\leqslant 1$. The calculation below is standard. We define t such that $\cosh t=(d_2+d_1)/(d_2-d_1)$. Then

$$\begin{aligned}\mathcal{T}_k\left(\frac{d_2+d_1}{d_2-d_1}\right)&=\cos(k\arccos\cosh t)\\&=\cos(k\arccos\cos it)\\&=\cos kit=\cosh kt=\frac{(e^t)^k+(e^{-t})^k}{2},\end{aligned}$$

where $e^t=(1+\sqrt{\kappa^{-1}})/(1-\sqrt{\kappa^{-1}})$. Therefore

$$\begin{aligned}2\left|\mathcal{T}_k\left(\frac{d_2+d_1}{d_2-d_1}\right)\right|&=\left(\frac{1+\sqrt{\kappa^{-1}}}{1-\sqrt{\kappa^{-1}}}\right)^k+\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^k\\&\geqslant\left(\frac{1+\sqrt{\kappa^{-1}}}{1-\sqrt{\kappa^{-1}}}\right)^k\end{aligned}$$

and the result follows. ∎

Lemma 4. *For $x>1$ and $k\geqslant 1$,*

$$\mathcal{T}_k(x)/\mathcal{T}_{k-1}(x)\leqslant 2x.$$

Proof. This follows directly from the well-known recurrences

$$\mathcal{T}_0(x)=1,\qquad \mathcal{T}_1(x)=x,$$
$$\mathcal{T}_{k+1}(x)=2x\mathcal{T}_k(x)-\mathcal{T}_{k-1}(x),\qquad k\geqslant 1.$$ ∎

Let us establish some notation. We denote the eigensystem of A by the $n\times n$ matrices U and Λ, where

$$AU=U\Lambda,\quad U^TU=I,\qquad \Lambda=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n),$$

and $0<\lambda_1\leqslant\lambda_2\leqslant\cdots\leqslant\lambda_n$. The columns of U are the eigenvectors. We partition the matrix Λ into

$$\Lambda=\begin{pmatrix}\Lambda_1&0\\0&\Lambda_2\end{pmatrix},$$

where $\Lambda_1=\operatorname{diag}(\lambda_1,\dots,\lambda_{s-1})$ and $\Lambda_2=\operatorname{diag}(\lambda_s,\dots,\lambda_n)$. Recall from Theorem 2 that the block conjugate gradient algorithm minimizes $e_m^{(k+1)T}Ae_m^{(k+1)}$ over all choices of polynomials $\mathcal{P}_{kmi}$ in Eq. (12). We denote by $E^{(s)}$ the $n\times(s-1)$ matrix formed by deleting the last column of the matrix E, where $E=[e_1^{(0)},e_2^{(0)},\dots,e_s^{(0)}]$.

The vectors g_m, $m=1,2,\dots,s-1$, defined in the following lemma will play a central role in our derivation of a convergence bound for the block conjugate gradient algorithm. They are the same vectors as those used by Underwood [34] to obtain a bound for the convergence rate for the block Lanczos algorithm for finding the eigenvalues of a matrix.

Lemma 5. *Let F be a matrix defined so that the columns of*

$$UF=U\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}\begin{matrix} (s-1)\times(s-1) \\ (n-s+1)\times(s-1) \end{matrix}$$

are an orthonormal basis for $\operatorname{sp}\{AE^{(s)}\}$, *and suppose that* $\sigma_{\min}(F_1)>0$. *Define*

$$\tilde{R}=U\begin{pmatrix} I \\ F_2F_1^{-1} \end{pmatrix}=(\tilde{r}_1,\dots,\tilde{r}_{s-1}),$$

$$\tilde{F}=F_2F_1^{-1}.$$

Then:

(a) $\|\tilde{F}\|^2=\tan^2\theta$, *where* $\theta=\arccos\sigma_{\min}(F_1)$.

(b) *For each* $m=1,2,\dots,s-1$ *there is a vector* $g_m\in\operatorname{sp}\{AE^{(s)},\dots,A^kE^{(s)}\}$ *such that*

$$g_m=u_m+\sum_{i=s}^{n}\delta_{im}u_i, \tag{13}$$

where

$$\delta_{im}=u_i^T\tilde{r}_m\mathcal{T}_{k-1}\left(\frac{\lambda_n+\lambda_s-2\lambda_i}{\lambda_n-\lambda_s}\right)\Big/\mathcal{T}_{k-1}\left(\frac{\lambda_n+\lambda_s-2\lambda_m}{\lambda_n-\lambda_s}\right),\qquad i=s,s+1,\dots,n.$$

Proof. (a): By definition,

$$F^TU^TUF=F_1^TF_1+F_2^TF_2=I.$$

Therefore, we have

$$F_1^{-T}F_2^TF_2F_1^{-1}=F_1^{-T}F_1^{-1}-I$$

and thus

$$\|F_2F_1^{-1}\|^2=\frac{1}{\sigma^2_{\min}(F_1)}-1=\tan^2\theta.$$

(b): Notice that the columns of $\tilde{R}$ are also in $\mathrm{sp}\{AE^{(s)}\}$, and let

$$g_m=\mathcal{P}_m(A)\tilde{r}_m$$

where $\mathcal{P}_m$ is the polynomial of degree $k-1$ defined by

$$\mathcal{P}_m(\lambda)=\mathcal{T}_{k-1}\left(\frac{\lambda_n+\lambda_s-2\lambda}{\lambda_n-\lambda_s}\right)\Big/\mathcal{T}_{k-1}\left(\frac{\lambda_n+\lambda_s-2\lambda_m}{\lambda_n-\lambda_s}\right).\qquad\blacksquare$$

Notice that the numbers $u_i^T\tilde{r}_m$ are elements in the matrix $\tilde{F}$.

In the following theorem we let κ denote the spread of the eigenvalues λ_s through λ_n:

$$\kappa=\lambda_n/\lambda_s.$$

We now state and prove a special case of the main result of this section.

THEOREM 5. *After k steps of the unscaled block conjugate gradient algorithm, the error in component s is bounded as*

$$e_s^{(k)T}Ae_s^{(k)}\leqslant\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^{2k}c_1.$$

NOTE. This result requires the hypothesis of Lemma 5 that $\sigma_{\min}(F_1)>0$. The result is valid for any component m using the definition $E^{(m)}=[e_1^{(0)},\dots,e_{m-1}^{(0)},e_{m+1}^{(0)},\dots,e_s^{(0)}]$. The precise definition of the constant c_1 is given in the course of proving the theorem. If $s=1$, the theorem reduces to a well-known result [6].

Proof. We define the errors at the kth step by

$$e_s^{(k)}=x_s^{(k)}-x_s^*=e_s^{(0)}+\sum_{j=1}^{s}\mathcal{P}_{kj}^*(A)Ae_j^{(0)},$$

$$e_s^{(0)}=x_s^{(0)}-x_s^*=U\xi,$$

where each $\mathcal{P}_{kj}^*$ is a polynomial of degree $k-1$. The strategy is to show that the bound holds for a particular choice of polynomials $\mathcal{P}_{kj}$ in the definition of $e_s^{(k)}$, and thus, since the block conjugate gradient algorithm chooses the optimal set of polynomials, the bound must hold for $\mathcal{P}_{kj}^*$, too.

We choose the polynomial $\mathcal{P}_{ks}$ to satisfy

$$1+\mathcal{P}_{ks}(\lambda)\lambda=\phi_k(\lambda)=\mathcal{T}_k\left(\frac{\lambda_n+\lambda_s-2\lambda}{\lambda_n-\lambda_s}\right)\Big/\mathcal{T}_k\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right).$$

The other polynomials are chosen to obtain a certain linear combination of the columns of G:

$$\sum_{j=1}^{s-1}\mathcal{P}_{kj}(A)Ae_j^{(0)}=-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j g_j.$$

Then

$$\begin{aligned}e_s^{(k)}&=\left[1+\mathcal{P}_{ks}(A)A\right]e_s^{(0)}+\sum_{j=1}^{s-1}\mathcal{P}_{kj}(A)Ae_j^{(0)}\\&=\left[1+\mathcal{P}_{ks}(A)A\right]U\xi-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j g_j.\end{aligned}$$

Using Eq. (13), we conclude that

$$\begin{aligned}e_s^{(k)}&=\sum_{i=1}^{n}\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]\xi_i u_i-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\left(u_j+\sum_{i=s}^{n}\delta_{ij}u_i\right)\\&=\sum_{i=s}^{n}\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]\xi_i u_i-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\sum_{i=s}^{n}\delta_{ij}u_i\\&=\sum_{i=s}^{n}\left(\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]\xi_i-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\delta_{ij}\right)u_i\end{aligned}$$

and thus

$$\begin{aligned}e_s^{(k)T}Ae_s^{(k)}&=\sum_{i=s}^{n}\left(\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]\xi_i-\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\delta_{ij}\right)^2\lambda_i\\&=\sum_{i=s}^{n}\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]^2\lambda_i\xi_i^2+\sum_{i=s}^{n}\left(\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\delta_{ij}\right)^2\lambda_i\\&\quad-2\sum_{i=s}^{n}\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_i\xi_j\delta_{ij}\lambda_i.\end{aligned}$$

We now bound each term in this final expression individually. The first term is bounded as

$$\sum_{i=s}^{n}\left[1+\mathcal{P}_{ks}(\lambda_i)\lambda_i\right]^2\lambda_i\xi_i^2=\sum_{i=s}^{n}\left[\frac{\mathcal{T}_k\left(\dfrac{\lambda_n+\lambda_s-2\lambda_i}{\lambda_n-\lambda_s}\right)}{\mathcal{T}_k\left(\dfrac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)}\right]^2\lambda_i\xi_i^2$$

$$\leqslant \xi_{\mathrm{II}}^T\Lambda_2\xi_{\mathrm{II}}\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right),$$

where we define

$$\xi_{\mathrm{I}}=\begin{bmatrix}\xi_1\\ \vdots\\ \xi_{s-1}\end{bmatrix},\qquad \xi_{\mathrm{II}}=\begin{bmatrix}\xi_s\\ \vdots\\ \xi_n\end{bmatrix}.$$

We have, for the second term,

$$\sum_{i=s}^{n}\left[\sum_{j=1}^{s-1}\left[1+\mathcal{P}_{ks}(\lambda_j)\lambda_j\right]\xi_j\delta_{ij}\right]^2\lambda_i$$

$$=\sum_{i=s}^{n}\left[\sum_{j=1}^{s-1}\frac{\mathcal{T}_k\left(\dfrac{\lambda_n+\lambda_s-2\lambda_j}{\lambda_n-\lambda_s}\right)}{\mathcal{T}_k\left(\dfrac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)}\,\frac{\mathcal{T}_{k-1}\left(\dfrac{\lambda_n+\lambda_s-2\lambda_i}{\lambda_n-\lambda_s}\right)}{\mathcal{T}_{k-1}\left(\dfrac{\lambda_n+\lambda_s-2\lambda_j}{\lambda_n-\lambda_s}\right)}\xi_j u_i^T\tilde{r}_j\right]^2\lambda_i$$

$$\leqslant 4\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)^2\sum_{i=s}^{n}\left[\sum_{j=1}^{s-1}\xi_j u_i^T\tilde{r}_j\right]^2\lambda_i\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)$$

$$=4\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)^2\xi_{\mathrm{I}}^T\tilde{F}^T\Lambda_2\tilde{F}\xi_{\mathrm{I}}\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)$$

$$\leqslant 4\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)^2\|\tilde{F}^T\tilde{F}\|\,\|\Lambda_2\|\,\|\xi_{\mathrm{I}}\|^2\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right).$$

Finally, the third term is bounded in absolute value by

$$2\left|\sum_{i=s}^{n}\sum_{j=1}^{s-1}\xi_i\xi_j u_i^T\tilde{r}_j\lambda_i\right|^2 \mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)$$

$$\leqslant 4\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)|\xi_{\mathrm{II}}^T\Lambda_2\tilde{F}\xi_{\mathrm{I}}|\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)$$

$$\leqslant 4\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)\|\xi_{\mathrm{I}}\|\,\|\Lambda_2\xi_{\mathrm{II}}\|\,\|\tilde{F}\|\Big/\mathcal{T}_k^2\left(\frac{\lambda_n+\lambda_s}{\lambda_n-\lambda_s}\right)$$

Adding these three bounds and using Lemma 3(b), the result follows. ■

We now discuss the generalization of this result to the scaled version of the algorithm.

THEOREM 5. *Let*

$$U^T(M^{1/2}AM^{1/2})U=\Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\dots\lambda_n)$$

where $U^TU=I$ *and* $0<\lambda_1\leqslant\lambda_2\leqslant\cdots\leqslant\lambda_n$. *Let*

$$\kappa=\lambda_n/\lambda_s.$$

For $m=1,2,\dots,s$ *let*

$$E^{(m)}=\left[e_1^{(0)},\dots,e_{m-1}^{(0)},e_{m+1}^{(0)},\dots,e_s^{(0)}\right].$$

Suppose that $\sigma_{\min}(F_1^{(m)})>0$, *where* $UF^{(m)}$ *is an orthonormal basis for the* $\mathrm{sp}\{AE^{(m)}\}$ *and*

$$F^{(m)}=\begin{pmatrix}F_1^{(m)}\\F_2^{(m)}\end{pmatrix}\begin{matrix}(s-1)\times(s-1)\\(n-s+1)\times(s-1)\end{matrix}$$

Then

$$e_m^{(k)T}Ae_m^{(k)}\leqslant\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^{2k}c_1,$$

where c_1 is a constant depending on m but independent of k, given by

$$c_1 \leqslant 4\xi_{II}^T\Lambda_2\xi_{II} + 16\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)^2 \tan^2\theta_m \|\Lambda_2\| \|\xi_I\|^2$$

$$+ 16\left(\frac{\lambda_n+\lambda_s-2\lambda_1}{\lambda_n-\lambda_s}\right)\|\xi_I\| \|\Lambda_2\xi_{II}\| \tan\theta_m$$

where $\theta_m = \arccos\sigma_{min}(F_1^{(m)})$.

Proof. This result is most easily established by noting that the scaled conjugate gradient algorithm can be derived by writing the unscaled algorithm for the equation

$$M^{1/2}AM^{1/2}Y = M^{1/2}B$$

and then transforming back to the original variables, replacing Y, R, and P by $M^{-1/2}X$, $M^{1/2}R$, and $M^{-1/2}P$ respectively. Making the same substitutions in the previous theorem yields the desired conclusion. ∎

The same kind of result holds for distinguished sets of eigenvalues other than λ_s through λ_n. Suppose $\lambda_{\max}$ and $\lambda_{\min}$ are chosen so that the interval $0 \leqslant \lambda \leqslant \lambda_{\max}+\lambda_{\min}$ contains all of the eigenvalues and there are $n-s+1$ eigenvalues in the interval $\lambda_{\min} \leqslant \lambda \leqslant \lambda_{\max}$. Then, defining

$$\phi_k = \mathcal{T}_k\left(\frac{\lambda_{\max}+\lambda_{\min}-2\lambda}{\lambda_{\max}-\lambda_{\min}}\right) \Big/ \mathcal{T}_k\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right),$$

we note that as λ ranges between 0 and $\lambda_{\max}+\lambda_{\min}$, $x = (\lambda_{\max}+\lambda_{\min}-2\lambda)/(\lambda_{\max}-\lambda_{\min})$ ranges between $(\lambda_{\max}+\lambda_{\min})/(\lambda_{\max}-\lambda_{\min})$ and $-(\lambda_{\max}+\lambda_{\min})/(\lambda_{\max}-\lambda_{\min})$, so $|\phi_k(\lambda)| \leqslant 1$. Thus a similar result holds with this partitioning of eigenvalues.

The convergence rate for the block conjugate gradient algorithm depends on the distribution of eigenvalues of MA and the choice of the initial matrix $X^{(0)}$. The first factor is of primary importance; for fast convergence, M and s can be chosen so that the matrix MA has a narrow cluster of $n-s+1$ eigenvalues. In addition, it is helpful if $X^{(0)}$ is chosen so that the space $\mathrm{sp}\{MR^{(0)},\ldots,(MA)^kMR^{(0)}\}$ is rich in the solution vectors for small k. In any case, the block method cannot be slower than the standard conjugate gradient method, which obeys the bound

$$e^{(k)T}Ae^{(k)} \leqslant 4\left(\frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}}\right)^{2k} e^{(0)T}Ae^{(0)}$$

with $\kappa = \lambda_n/\lambda_1$.

5. THE BLOCK LANCZOS ALGORITHM AND METHODS FOR SYMMETRIC INDEFINITE MATRICES

In this section we discuss an interpretation of the B-CG algorithm in terms of a similarity transformation of the matrix A. This approach yields further insight and alternate computational algorithms. For simplicity, we work with the unscaled version of the algorithm, generalizing to preconditioning at the end of the section. Our starting point is the following algorithm, applicable to any symmetric matrix.

ALGORITHM BLOCK LANCZOS (Golub and Underwood [10, 34], Cullum and Donath [5]). Choose an arbitrary $n\times s$ matrix B, choose ν_1 so that $Z^{(1)}=B\nu_1$ satisfies $Z^{(1)T}Z^{(1)}=I$, and let $Z^{(0)}=0$. Iterate for $k=1,2,\dots$:

$$Z^{(k+1)}=\left(AZ^{(k)}-Z^{(k)}\rho_k-Z^{(k-1)}\nu_k^{-T}\right)\nu_{k+1},$$

where $\rho_k=Z^{(k)T}AZ^{(k)}$ and ν_{k+1} is chosen so that $Z^{(k+1)T}Z^{(k+1)}=I$.

It is easy to show that the matrices $Z^{(k)}$ produced by the algorithm are mutually orthogonal: i.e.,

$$Z^{(i)T}Z^{(j)}=0 \qquad \text{for} \quad i\neq j,$$

and, if the algorithm does not break down, after $k\leq\bar{k}=\lceil n/s\rceil$ steps we have a decomposition of the matrix A as

$$\begin{aligned} &A\left(Z^{(1)},Z^{(2)},\dots,Z^{(k)}\right)\\ &\quad=\left(Z^{(1)},Z^{(2)},\dots,Z^{(k)}\right)\begin{bmatrix} \rho_1 & \nu_2^{-T} & & & \\ \nu_2^{-1} & \rho_2 & \nu_3^{-T} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \nu_k^{-T} \\ & & & \nu_k^{-1} & \rho_k \end{bmatrix}\\ &\qquad+\left(0,0,\dots,0,Z^{(k+1)}\nu_{k+1}^{-1}\right),\end{aligned}$$

where $Z^{(\bar{k}+1)}=0$. We abbreviate this equation as

$$A\mathfrak{Z}^{(k)}=\mathfrak{Z}^{(k)}T^{(k)}+\hat{Z}^{(k)}.$$

This algorithm uses a three term recurrence relation, as does Version 2 of B-CG, and both algorithms yield a tridiagonal decomposition. The major difference is that the Z's are orthonormalized while the R's are not.

Ruhe [30] has observed that the block Lanczos vectors can be generated in such a way that T is not only a block tridiagonal matrix but also a band matrix with $2s+1$ nonzero diagonals. This is done by using the natural choice of ν_{k+1}, a right triangular matrix generated by either the QR algorithm or a modified Gram-Schmidt procedure. He also proposes generating the Z-columns one by one and reorthogonalizing against several previous Z-blocks to slow the growth of roundoff error. With this implementation, the stability of the process can be monitored quite easily.

The columns of $\mathfrak{Z}^{(\bar{k})}$ are an orthonormal basis for the space of n-vectors, so we can express the solution to our linear system as

$$X^{*}=\mathfrak{Z}^{(\bar{k})}\hat{\psi}^{(\bar{k})},$$

where $\hat{\psi}$ is an n-vector, and the system itself becomes

$$A\mathfrak{Z}^{(\bar{k})}\hat{\psi}^{(\bar{k})}=B=Z^{(1)}\nu_1^{-1},$$

or, multiplying by $\mathfrak{Z}^{(\bar{k})T}$,

$$T^{(\bar{k})}\hat{\psi}^{(\bar{k})}=\mathfrak{Z}^{(\bar{k})T}B=\begin{bmatrix}\nu_1^{-1}\\0\\\vdots\\0\end{bmatrix}.$$

In this form we cannot conveniently accumulate iterates $X^{(k)}$ as the Lanczos matrices are generated, but by a further change of variables we can. Let us define a QR decomposition of $T^{(k)}$,

$$T^{(k)}\mathfrak{Q}^{(k)T}=L^{(k)},$$

where $\mathcal{V}^{(k)T}\mathcal{V}^{(k)} = I$ and

$$L^{(k)} = \begin{bmatrix} L_{1,1} & & & & & \\ L_{2,1} & L_{2,2} & & & & \\ L_{3,1} & L_{3,2} & L_{3,3} & & & \\ & \ddots & \ddots & \ddots & & \\ & & L_{k-1,k-3} & L_{k-1,k-2} & L_{k-1,k-1} & \\ & & & L_{k,k-2} & L_{k,k-1} & \bar{L}_{k,k} \end{bmatrix}.$$

Here L_{ij} is a matrix of size $s \times s$. Introducing new variables

$$W^{(k)} = \mathcal{Z}^{(k)}\mathcal{V}^{(k)T}, \qquad \psi^{(k)} = \mathcal{V}^{(k)}\hat{\psi}^{(k)},$$

our linear system becomes

$$L^{(\bar{k})}\psi^{(\bar{k})} = \begin{bmatrix} \nu_1^{-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$X^{(\bar{k})} = \mathcal{Z}^{(\bar{k})}\mathcal{V}^{(\bar{k})T}\psi^{(\bar{k})} = W^{(\bar{k})}\psi^{(\bar{k})}.$$

On examining the details of these formulas, we discover that the factors of $\mathcal{V}$ and the blocks of columns of W can be formed and discarded one by one. This development is entirely parallel to that of Paige and Saunders for the single-vector case, and a detailed derivation of the algorithm for $s = 1$ is given in [27]. Note that for $j \leqslant k$,

$$\begin{aligned} Z^{(j)T}(AX^{(k)} - B) &= Z^{(j)T}\left(A\mathcal{Z}^{(k)}\hat{\psi}^{(k)} - B\right) \\ &= \left(T^{(k)}\hat{\psi}^{(k)} - \mathcal{Z}^{(k)T}B\right)_{j\text{th block}} = 0, \end{aligned}$$

and since the span of the columns of $\mathcal{Z}^{(k)}$ is the same as that of the Krylov sequence $\{B, AB, \ldots, A^{k-1}B\}$, $X^{(k)}$ must be the same as the iterate from the B-CG algorithm. Thus we have a new alternate form of this algorithm.

Algorithm B-CG Version 4 (Paige and Saunders form). Given B and $X^{(0)}$, let $\hat{R}^{(0)} = B - AX^{(0)}$, $\hat{X}^{(0)} = X^{(0)}$, $Z^{(0)} = 0$, $Z^{(1)} = \hat{R}^{(0)}\nu_1$, where $Z^{(1)T}Z^{(1)} =$

I, $\overline{W}^{(1)}=Z^{(1)}$, $\rho_0=0$, $\rho_1=Z^{(1)T}AZ^{(1)}$, $\bar{L}_{1,1}=\rho_1$, and $V_1=I_{2s\times 2s}$. For $k=1,2,\ldots$, update the Lanczos vectors

$$Z^{(k+1)}=\left(AZ^{(k)}-Z^{(k)}\rho_k-Z^{(k-1)}\nu_k^{-T}\right)\nu_{k+1},$$

where ν_{k+1} is chosen so that $Z^{(k+1)T}Z^{(k+1)}=I$. Update the factorization; set

$$\rho_{k+1}=Z^{(k+1)T}AZ^{(k+1)},$$

$$\left(L_{k+1,k-1},\bar{L}_{k+1,k}\right)=\left(0,\nu_{k+1}^{-1}\right)V_k^T,$$

compute V_{k+1} and $L_{k,k}$ so that $V^{(k+1)T}V^{(k+1)}=I_{2s\times 2s}$ and

$$\left(\bar{L}_{k,k},\nu_{k+1}^{-T}\right)V_{k+1}^T=\left(L_{k,k},0\right),$$

and let

$$\left(L_{k+1,k},\bar{L}_{k+1,k+1}\right)=\left(\bar{L}_{k+1,k},\rho_{k+1}\right)V_{k+1}^T,$$

$$\left(W^{(k)},\overline{W}^{(k+1)}\right)=\left(\overline{W}^{(k)},Z^{(k+1)}\right)V_{k+1}^T.$$

Update the iterates; compute ψ_k to satisfy

$$L_{k,k-2}\psi_{k-2}+L_{k,k-1}\psi_{k-1}+L_{k,k}\psi_k=0$$

or, if $k=1$,

$$L_{1,1}\psi_1=\nu_1^{-1},$$

and set

$$\hat{X}^{(k)}=\hat{X}^{(k-1)}+W^{(k)}\psi_k.$$

Upon termination, determine $\bar{\psi}_{k+1}$ and $X^{(k+1)}$ from

$$L_{k+1,k-1}\psi_{k-1}+L_{k+1,k}\psi_k+\bar{L}_{k+1,k+1}\bar{\psi}_{k+1}=0,$$

$$X^{(k+1)}=\hat{X}^{(k)}+\overline{W}^{(k+1)}\bar{\psi}_{k+1}.$$

One major advantage of Version 4 over the forms previously discussed is its behavior on indefinite matrices. All of the versions produce least squares solutions if the matrix A is semidefinite, but the first three may be unstable if A is indefinite, since they rely implicitly on an LU decomposition of the matrix T. Because the QR factorization is stable for indefinite matrices, Version 4 can be used for any symmetric matrix as long as the Z matrices retain rull rank.

As one final variation on the theme, we extend to block form an algorithm due to Fridman [9] and discovered independently by Fletcher [7]. Starting from $X^{(0)}=0$, instead of minimizing $\mathrm{tr}[(X-X^*)^TA(X-X^*)]$ over all matrices in the kth Krylov space as the B-CG algorithm does, we minimize the norm of the error $\mathrm{tr}[(X-X^*)^T(X-X^*)]$ over the space $\{AR^{(0)},\dots, A^{(k)}R^{(0)}\}$. Using the Z columns as a convenient basis, we can express our iterates as

$$X^{(k+1)}=\sum_{j=1}^{k+1} Z^{(j)}\varepsilon_j,$$

where $\mathrm{sp}\{Z^{(1)}\}=\mathrm{sp}\{AR^{(0)}\}$ and

$$\varepsilon_j=Z^{(j)T}X^*.$$

In this form, the ε's are not computable, but notice that

$$X^*-X^{(k)}=\sum_{j=k+1}^{n} Z^{(j)}\varepsilon_j$$

and thus the columns of this error matrix are orthogonal to all $Z^{(i)}$ for $i\leqslant k$. Using this, the fact that $Z^{(k+1)T}X^{(k)}=0$, the definition of the Lanczos vectors, and the orthogonality of the columns of Z, we see that

$$\begin{aligned}\varepsilon_{k+1}&=Z^{(k+1)T}X^*=Z^{(k+1)T}(X^*-X^{(k)})\\&=\nu_{k+1}^T\left(AZ^{(k)}-Z^{(k)}\rho_k-Z^{(k-1)}\nu_k^{-T}\right)^T(X^*-X^{(k)})\\&=\nu_{k+1}^TZ^{(k)T}A(X^*-X^{(k)})\\&=\nu_{k+1}^TZ^{(k)T}R^{(k)},\end{aligned}$$

and this gives us a minimum error (ME) algorithm.

ALGORITHM B-ME. Given B and $X^{(0)}$, let $R^{(0)}=B-AX^{(0)}$ and $Z^{(1)}=AR^{(0)}\nu_1$, where $Z^{(1)T}Z^{(1)}=I$. Define $Z^{(0)}=0$. For $k=0,1,\dots$

$$X^{(k+1)}=X^{(k)}+Z^{(k+1)}\varepsilon_{k+1},$$

$$R^{(k+1)}=R^{(k)}-AZ^{(k+1)}\varepsilon_{k+1},$$

where

$$\varepsilon_{k+1}=\nu_{k+1}^{T}Z^{(k)T}R^{(k)},$$

and let

$$Z^{(k+2)}=\left(AZ^{(k+1)}-Z^{(k+1)}\rho_{k+1}-Z^{(k)}\nu_{k+1}^{-T}\right)\nu_{k+2},$$

where $\rho_{k+1}=Z^{(k+1)T}AZ^{(k+1)}$ and ν_{k+2} is chosen so that $Z^{(k+2)T}Z^{(k+2)}=I$.

This algorithm also is suitable for symmetric indefinite systems, but Paige and Saunders have reported that Version 4 of the CG algorithm is less sensitive to roundoff than the single vector form of the B-ME algorithm. A version of the B-ME algorithm analogous to the single vector implementation suggested by Hestenes [13] could be derived from the B-BCG algorithm.

To derive the scaled versions for the algorithms in this section, we use the variable transformation employed in the proof of Theorem 6. Each algorithm is applied to the problem

$$M^{1/2}AM^{1/2}Y=M^{1/2}B.$$

The resulting formulas are then rewritten in terms of the original residuals and variables, replacing Z, R, W, and Y in the formulas by $M^{1/2}Z$, $M^{1/2}R$, $M^{-1/2}W$, and $M^{-1/2}X$ respectively. The final algorithms require the matrix M but not $M^{1/2}$ or $M^{-1/2}$.

6. REMARKS AND CONCLUSIONS

We have presented several block iterative algorithms for solving systems of linear equations. Storage and operations counts for some typical implementations of these algorithms and some single vector algorithms are given in Table 1. The leading terms are of order ns plus s^2 for storage, and order ns^2 plus s^3 plus the multiplication times for A and M with a vector for operations per iteration. The amount of storage can be reduced at the cost of increasing the operations counts somewhat.

Many questions relating to the rate of convergence of the B-CG algorithms remain open. It should be possible, for example, to obtain results

TABLE 1

STORAGE, MATRIX OPERATIONS, AND OPERATIONS COUNTS FOR VARIOUS ALGORITHMS

Algorithm	Storage: n-vectors	Storage: Total	Matrix operations per iteration: n-vector multiplications by A	Matrix operations per iteration: n-vector multiplications by M
BCG ($s=1, M=I$)	$x,r,\tilde{r},p,\tilde{p},Ap\ (A^T\tilde{p})$	$6n$	2	—
CG Version 1 ($s=1, M=I$)	x,r,p,Ap	$4n$	1	—
Preconditioned CG Version 1 ($s=1$)	$x,r,p,Ap\ (MR)$	$4n$	1	1
B-BCG	$X,R,\bar{R},P,\bar{P},AP\ (MR,A^TP,M^TR)$	$6ns$	$2s$	$2s$
B-CG Version 1	$X,R,P,AP\ (MR)$	$4ns$	$1s$	$1s$
B-CG Version 2	$X,\Delta X,R,\Delta R,MR,AMR$	$6ns$	$1s$	$1s$
B-CG Version 3	Y,R,P,AMP,MR (and MP)	$5ns$	$1s$	$2s$
B-MR	$X,R,P,AP\ (MR),MAP(AMR)$	$5ns$	$2s$	$2s$
B-ME	X,R,AZ,Z,Z,MZ	$6ns$	$1s$	$1s$

Overhead per Iteration

Algorithm	n-vector inner products	Multiply $n\times s$ and $s\times s$ matrices	n-vector additions	Factor $s\times s$ matrix	Orthogonalize s n-vectors	Maximum number of iterations
BCG ($s=1, M=I$)	2	4	4	—	—	n
CG Version 1 ($s=1, M=I$)	2	3	3	—	—	n
Preconditioned CG Version 1 ($s=1$)	2	3	3	—	—	n
B-BCG	$2s^2$	4	$4s$	2	(2)	$\lceil n/s \rceil$
B-CG Version 1	s^2+s	3	$3s$	2	(1)	$\lceil n/s \rceil$
B-CG Version 2	s^2+s	4	$4s$	2	—	$\lceil n/s \rceil$
B-CG Version 3	s^2+s	3	$3s$	2	(1)	$\lceil n/s \rceil$
B-MR	s^2+s	3	$3s$	2	(1)	$\lceil n/s \rceil$
B-ME	$1.5s^2+0.5s$	4	$4s$	—	1	$\lceil n/s \rceil$

analogous to those of Axelsson [1] and Greenbaum [11] for particular eigenvalue distributions.

Many computational questions also remain open. Only extensive computational experience and roundoff error analysis will determine how the algorithms should be implemented in order to balance reliability and efficiency. Limited computational experience thus far indicates that the linear independence of the direction vectors should be monitored, and orthonormalization of them seems essential in order to avoid underflow and overflow.

As expected, using the block conjugate gradient algorithm can result in savings for solving multiple systems, for solving systems in which several eigenvalues (large or small) are widely separated from the others, and for problems in which the matrix is stored on a secondary storage device and thus is expensive to access. For example, on a matrix of dimension $n=100$ with eigenvalues 1, 1.5, 2,..., 49.5, 50, and 400, the block CG algorithm with two random B-vectors took 19 iterations to reduce the residual norms by a factor of 10^{-4}, whereas the conjugate gradient algorithm took a total of 45 iterations for the two problems, and the partial (s-step) conjugate gradient method described by Luenberger [21, p. 187], designed for such eigenvalue distributions, took 207. Matrices with separated low eigenvalues show the same trends: a similar experiment with three vectors and a matrix of dimension 200 with eigenvalues 1,2 and 400,401,...,596,597 required 5 iterations of the block conjugate gradient algorithm, 25 iterations of the conjugate gradient algorithm, and 39 iterations of the partial conjugate gradient algorithm. Thus, in both cases the block algorithm took the smallest number of matrix multiplications and also the smallest number of accesses to the matrix.

Richard R. Underwood will present further results on computation and implementation of the block conjugate gradient algorithm in a later report.

I am grateful to Richard Underwood for providing notes from a talk he presented on the block conjugate gradient algorithm and for his very helpful comments on a draft of the manuscript. I am also grateful to Olof Widlund and G. W. Stewart for suggesting that it should be possible to strengthen an earlier version of the result given in Theorem 5.

REFERENCES

1 O. Axelsson, On preconditioning and convergence acceleration in sparse matrix computations, Report 74-10, CERN, Geneva, 1974.

2 O. Axelsson, Solution of linear systems of equations: iterative methods, in *Sparse Matrix Techniques* (V. A. Barker, Ed.), Springer, New York, 1977, pp. 1–11.

3 R. Chandra, Conjugate gradient methods for partial differential equations, Computer Science Department Report 129, Yale Univ., 1978.

4 P. Concus, G. H. Golub, and D. P. O'Leary, A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, Eds.), Academic, New York, 1976, pp. 309–332.

5 J. Cullum and W. E. Donath, A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices, *Proc. of 1974 IEEE Conf. on Decision and Control*, Phoenix.

6 J. W. Daniel, The conjugate gradient method for linear and nonlinear operator equations, Ph.D. thesis, Stanford Univ.; *SIAM J. Numer. Anal.* 4:10–26 (1967).

7 R. Fletcher, Conjugate gradient methods for indefinite systems, in *Numerical Analysis Dundee 1975* (G. A. Watson, Ed.), Springer, New York, 1976, pp. 73–89.

8 L. Fox, H. D. Huskey, and J. H. Wilkinson, Notes on the solution of algebraic linear simultaneous equations, *Quart. J. Mech. Appl. Math.* 1:149–173 (1948).

9 V. M. Fridman, The method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetric matrix, *U.S.S.R. Computational Math. and Math. Phys.* 2:362–363 (1963).

10 G. H. Golub and R. R. Underwood, The block Lanczos method for computing eigenvalues, in *Mathematical Software III* (J. R. Rice, Ed.), Academic, New York, 1977.

11 A. Greenbaum, Comparison of splittings used with the conjugate gradient algorithm, *Numer. Math.* 33:181–194 (1979).

12 L. Hageman, F. Luk, and D. Young, On the acceleration of iterative methods, Report CNA 129, Univ. of Texas, Austin, 1977.

13 M. R. Hestenes, The conjugate gradient method for solving linear systems, *Proc. Symp. Appl. Math.* 6:83–102 (1956).

14 M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards* 49:409–436 (1952).

15 W. Kahan and B. N. Parlett, How far should you go with the Lanczos process, in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, Eds.), Academic, New York, 1976, pp. 131–144.

16 S. Kaniel, Estimates for some computational techniques in linear algebra, *Math. Comp.* 20:369–378 (1966).

17 W. Karush, An iterative method for finding characteristic vectors of a symmetric matrix, *Pacific J. Math.* 1:233–248 (1951).

18 C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bur. Standards* 45:255–282 (1950).

19 C. Lanczos, Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Standards* 49:33–53 (1952).

20 J. G. Lewis, Algorithms for sparse matrix eigenvalue problems, Computer Science Dept. Report 595, Stanford Univ., 1977.

21 D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass., 1973.

22 J. A. Meijerink and H. A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comp.* 31:148–162 (1977).

23 D. P. O'Leary and O. Widlund, Capacitance matrix methods for the Helmholtz equation on general three dimensional regions, *Math. Comp.* 33:849–880 (1979).

24 C. C. Paige, The computation of eigenvalues and eigenvectors of very large sparse matrices, Ph.D. thesis, Institute of Computer Science, London Univ., 1971.

25 C. C. Paige, Computational variants of the Lanczos method for the eigenproblem, *J. Inst. Math. Appl.* 10:373–381 (1972).

26 C. C. Paige, Practical use of the symmetric Lanczos process with re-orthogonalization, *BIT* 10:183–195 (1970).

27 C. C. Paige and M. A. Saunders, Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.* 12:617–629 (1975).

28 J. K. Reid, On the method of conjugate gradients for the solution of large sparse systems of linear equations, in *Large Sparse Sets of Linear Equations* (J. K. Reid, Ed.), Academic, New York, 1971, pp. 231–254.

29 J. K. Reid, The use of conjugate gradients for systems of linear equations possessing "Property A", *SIAM J. Numer. Anal.* 9:325–332 (1972).

30 A. Ruhe, Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, Mathematics Dept. Report, Univ. of Calif., San Diego, 1978.

31 H. Rutishauser, Theory of gradient methods, in *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems* (M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, Eds.), Birkhäuser, Stuttgart, 1959, pp. 24–49.

32 G. W. Stewart, Conjugate direction methods for solving systems of linear equations, *Numer. Math.* 21:285–297 (1973).

33 G. W. Stewart, The convergence of the method of conjugate gradients at isolated extreme points of the spectrum, *Numer. Math.* 24:85–93 (1975).

34 R. Underwood, An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems, Computer Science Dept. Rept. 496, Stanford Univ., 1975.

Received 25 September 1978