RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Recent Advances in Lanczos-based Iterative Methods for Nonsymmetric Linear Systems

Roland W. Freund, Gene H. Golub,

and Noël M. Nachtigal

RIACS Technical Report 92.02

January 1992

To appear in
Algorithmic Trends for Computational Fluid Dynamics in the 90's

Recent Advances in Lanczos-based Iterative Methods for Nonsymmetric Linear Systems

Roland W. Freund, Gene H. Golub,
and Noël M. Nachtigal

The Research Institute for Advanced Computer Science is operated by
Universities Space Research Association (USRA),
The American City Building, Suite 311, Columbia, MD 21044, (301)730–2656.

Work reported herein was supported in part by Cooperative Agreement NCC 2–387 between NASA and USRA.

Recent Advances in Lanczos-based Iterative Methods for Nonsymmetric Linear Systems

Roland W. Freund*
RIACS, Mail Stop T041-5
NASA Ames Research Center
Moffett Field, CA 94035
E-mail: na.freund@na-net.ornl.gov

Gene H. Golub†
Computer Science Department
Stanford University
Stanford, CA 94305
E-mail: golub@sccm.stanford.edu

Noël M. Nachtigal*
RIACS, Mail Stop T041-5
NASA Ames Research Center
Moffett Field, CA 94035
E-mail: na.nachtigal@na-net.ornl.gov

Abstract. In recent years, there has been a true revival of the nonsymmetric Lanczos method. On the one hand, the possible breakdowns in the classical algorithm are now better understood, and so-called look-ahead variants of the Lanczos process have been developed, which remedy this problem. On the other hand, various new Lanczos-based iterative schemes for solving nonsymmetric linear systems have been proposed. This paper gives a survey of some of these recent developments.

1 Introduction

Many numerical computations involve the solution of large nonsingular systems of linear equations

$$Ax = b. \tag{1.1}$$

For example, such systems arise from finite difference or finite element approximations to partial differential equations (PDEs), as intermediate steps in computing the solution of nonlinear problems, or as subproblems in large-scale linear and nonlinear programming. Typically, the coefficient

*The work of these authors was supported by Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association (USRA).

†The work of this author was supported in part by the National Science Foundation under Grant NSF CCR-8821078.

matrix A of (1.1) is sparse and highly structured. A natural way to exploit the sparsity of A in the solution process is to use iterative techniques, which involve A only in the form of matrix-vector products. Most iterative schemes of this type fall into the category of *Krylov subspace methods*: they produce approximations x_n to $A^{-1}b$ of the form

$$x_n \in x_0 + K_n(r_0, A), \quad n = 1, 2, \ldots . \tag{1.2}$$

Here x_0 is any initial guess for $A^{-1}b$, $r_0 := b - Ax_0$ is the corresponding residual vector, and

$$K_n(r_0, A) := \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\} \tag{1.3}$$

is the nth *Krylov subspace* generated by r_0 and A.

The most powerful iterative method of this type is the *conjugate gradient algorithm* (CG) due to Hestenes and Stiefel [33], which is a scheme for linear systems (1.1) with Hermitian positive definite A. Although CG was introduced as early as 1952, its true potential was not appreciated until the 1970s. In 1971, Reid [45] revived interest in the method when he demonstrated its usefulness for solving linear systems arising from self-adjoint elliptic PDEs. Moreover, it was realized (see, e.g., [7]) that the performance of CG can be enhanced by combining it with preconditioning, and efficient preconditioners, such as the incomplete Cholesky factorization [40], were developed.

Thereafter, the success of CG triggered an extensive search for CG-type Krylov subspace methods for non-Hermitian linear systems, and a number of such algorithms have been proposed; we refer the reader to [1, 51, 48, 47, 17] and the references given there. Among the many properties of CG, the following two are the most important ones: its nth iterate is defined by a minimization property over $K_n(r_0, A)$, and the algorithm is based on three-term vector recurrences. Ideally, a CG-like method for non-Hermitian matrices would have features similar to these two. It would produce iterates x_n in (1.2) that:

(i) are characterized by a minimization property over $K_n(r_0, A)$, such as the minimal residual property

$$\|b - Ax_n\| = \min_{x \in x_0 + K_n(r_0, A)} \|b - Ax\|, \quad x_n \in x_0 + K_n(r_0, A);$$

(ii) can be computed with little work per iteration and low overall storage requirements.

Unfortunately, it turns out that, for general non-Hermitian matrices, one cannot fulfill (i) and (ii) simultaneously. This result is due to Faber and Manteuffel [10, 11] who have shown that, except for a few anomalies, CG-type algorithms with (i) and (ii) exist only for matrices of the special form

$$A = e^{i\theta}(T + \sigma I), \quad \text{where} \quad T = T^H, \quad \theta \in \mathbf{R}, \quad \sigma \in \mathbf{C}, \tag{1.4}$$

(see also Voevodin [55] and Joubert and Young [35]). Note that the class (1.4) consists of just the shifted and rotated Hermitian matrices. We remark that the important subclass of real nonsymmetric matrices

$$A = I - S, \quad \text{where} \quad S = -S^T \quad \text{is real}, \tag{1.5}$$

is contained in (1.4), with $e^{i\theta} = i$, $\sigma = -i$, and $T = iS$. Concus and Golub [6] and Widlund [56] were the first to devise a CG-type algorithm for the family (1.5).

Most of the non-Hermitian Krylov subspace methods that have been proposed satisfy either (i) or (ii). Until recently, the emphasis was on requirement (i), and numerous algorithms with

iterates characterized by (i) or a similar condition have been developed, starting with Vinsome's Orthomin [54]. The most widely used method in this class is the *generalized minimal residual algorithm* (GMRES) due to Saad and Schultz [49]. Of course, none of these methods fulfills (ii), and indeed, for all these algorithms work per iteration and overall storage requirements grow linearly with the iteration number n. Consequently, in practice one cannot afford to run the full version of these algorithms, and it is necessary to use restarts. For difficult problems, this often results in very slow convergence.

The second category of CG-like non-Hermitian Krylov subspace methods consists of schemes that satisfy (ii), but not (i). The archetype in this class is the classical *biconjugate gradient algorithm* (BCG), which was proposed by Lanczos [38] already in 1952 and later revived by Fletcher [12] in 1976. Since no minimization condition of type (i) holds for BCG, the algorithm can exhibit—and typically does—a rather irregular convergence behavior with wild oscillations in the residual norm. Even worse, breakdowns in the form of division by 0 may be encountered during the iteration process. In finite precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns may occur, leading to numerical instabilities in subsequent iterations.

The BCG method is intimately connected with the *nonsymmetric Lanczos process* [37] for tridiagonalizing square matrices. In particular, the Lanczos algorithm in its original form is also susceptible to breakdowns and potential numerical instabilities. In recent years, there has been a true revival of the nonsymmetric Lanczos process. On the one hand, the possible breakdowns in the classical algorithm are now better understood, and so-called *look-ahead* variants of the Lanczos process have been developed, which remedy this problem. On the other hand, various new Lanczos-based Krylov subspace methods for solving general non-Hermitian linear systems have been proposed. Here we review some of these recent developments.

The remainder of the paper is organized as follows. In Section 2, we focus on the nonsymmetric Lanczos process; in particular, we sketch a look-ahead variant of the method and briefly discuss related work. We then turn to Lanczos-based Krylov subspace algorithms for non-Hermitian linear systems. First, in Section 3, we consider the recently proposed *quasi-minimal residual method* (QMR) and outline two implementations. In addition to matrix-vector products with the coefficient matrix A of (1.1), BCG and QMR also require multiplications with its transpose A^T. This is a disadvantage for certain applications where A^T is not readily available. It is possible to devise Lanczos-based methods that do not involve A^T, and in Section 4, we survey some of these so-called *transpose-free* schemes. In Section 5, we make some concluding remarks.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, M^T and M^H denote the transpose and conjugate transpose of a matrix M, respectively. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm. The notation

$$\mathcal{P}_n = \{\phi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \mid \sigma_0, \ldots, \sigma_n \in \mathbf{C}\}$$

is used for the set of all complex polynomials of degree at most n. Finally, A is always assumed to be a square matrix of order N.

2 The Nonsymmetric Lanczos Process

In this section, we consider the nonsymmetric Lanczos process. Here the matrix A is not required to be nonsingular.

2.1 A Look-Ahead Lanczos Algorithm

The Lanczos method in its original form as proposed by Lanczos [37] can break down prematurely. Taylor [52] and Parlett, Taylor, and Liu [44]—with their look-ahead Lanczos algorithm—were the first to devise a variant of the classical process that skips over possible breakdowns. We use the term look-ahead Lanczos method in a broader sense to denote any extension of the standard algorithm that circumvents breakdowns. In this section, we sketch an implementation of a look-ahead Lanczos algorithm that was recently developed by Freund, Gutknecht, and Nachtigal [18].

Given two nonzero starting vectors $v_1 \in \mathbf{C}^N$ and $w_1 \in \mathbf{C}^N$, the look-ahead Lanczos process generates two sequences of vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ such that, for $n = 1, 2, \ldots$,

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \ldots, v_n\} &= K_n(v_1, A), \\ \operatorname{span}\{w_1, w_2, \ldots, w_n\} &= K_n(w_1, A^T). \end{aligned} \tag{2.1}$$

Here, $K_n(v_1, A)$ and $K_n(w_1, A^T)$ denote the nth Krylov subspace of $\mathbf{C}^N$ generated by v_1 and A, and w_1 and A^T, respectively (cf. (1.3)). Moreover, the Lanczos vectors are constructed so that the block biorthogonality relation

$$(W^{(j)})^T V^{(k)} = \begin{cases} D^{(k)} & \text{if } j = k, \\ 0 & \text{if } j \neq k, \end{cases} \quad j, k = 1, \ldots, l, \tag{2.2}$$

holds. Here, the matrices $V^{(k)}$ and $W^{(k)}$ contain the Lanczos vectors built during the kth look-ahead step. More precisely,

$$\begin{aligned} V^{(k)} &= [\, v_{n_k} \quad v_{n_k+1} \quad \cdots \quad v_{n_{k+1}-1} \,], \\ W^{(k)} &= [\, w_{n_k} \quad w_{n_k+1} \quad \cdots \quad w_{n_{k+1}-1} \,], \end{aligned} \qquad k = 1, \ldots, l-1,$$

and

$$\begin{aligned} V^{(l)} &= [\, v_{n_l} \quad v_{n_l+1} \quad \cdots \quad v_n \,], \\ W^{(l)} &= [\, w_{n_l} \quad w_{n_l+1} \quad \cdots \quad w_n \,], \end{aligned}$$

where

$$1 = n_1 < n_2 < \cdots < n_k < \cdots < n_l \leq n < n_{l+1}.$$

The first vectors v_{n_k} and w_{n_k} in each block are called *regular*, and any remaining vectors are called *inner*. Note that $l = l(n)$ denotes the index of the last constructed regular vector. Furthermore, in (2.2), the blocks $D^{(k)}$ are nonsingular for $k = 1, \ldots, l-1$, and $D^{(l)}$ is nonsingular if $n = n_{l+1} - 1$.

With these preliminaries, the look-ahead Lanczos algorithm can be sketched as follows.

Algorithm 2.1 (Sketch of the look-ahead Lanczos process)

0) *Choose nonzero vectors* v_1, $w_1 \in \mathbf{C}^N$.

Set $V^{(1)} = v_1$, $W^{(1)} = w_1$, $D^{(1)} = (W^{(1)})^T V^{(1)}$.

Set $n_1 = 1$, $l = 1$, $v_0 = w_0 = 0$, $V_0 = W_0 = \emptyset$.

For $n = 1, 2, \ldots$, *do* :

1) *Decide whether to construct* v_{n+1} *and* w_{n+1} *as regular or inner vectors and go to* 2) *or* 3), *respectively.*

2) (*Regular step.*) *Compute*

$$\begin{aligned} \mu_n &= (D^{(l)})^{-1}(W^{(l)})^T A v_n, \\ \nu_n &= (D^{(l-1)})^{-1}(W^{(l-1)})^T A v_n, \\ v_{n+1} &= A v_n - V^{(l)}\mu_n - V^{(l-1)}\nu_n, \\ w_{n+1} &= A^T w_n - W^{(l)}\mu_n - W^{(l-1)}\nu_n. \end{aligned} \tag{2.3}$$

Set $n_{l+1} = n+1$, $l = l+1$, $V^{(l)} = W^{(l)} = \emptyset$, *and go to* 4).

3) (*Inner step.*) *Compute*

$$\begin{aligned} \nu_n &= (D^{(l-1)})^{-1}(W^{(l-1)})^T A v_n, \\ v_{n+1} &= A v_n - \zeta_n v_n - \eta_n v_{n-1} - V^{(l-1)}\nu_n, \\ w_{n+1} &= A^T w_n - \zeta_n w_n - \eta_n w_{n-1} - W^{(l-1)}\nu_n. \end{aligned} \tag{2.4}$$

4) *If* $v_{n+1} = 0$ *or* $w_{n+1} = 0$, *stop. Otherwise, set*

$$V^{(l)} = \left[\begin{array}{cc} V^{(l)} & v_{n+1} \end{array}\right], \quad W^{(l)} = \left[\begin{array}{cc} W^{(l)} & w_{n+1} \end{array}\right],$$

$$D^{(l)} = (W^{(l)})^T V^{(l)}.$$

In [18], it is shown how one can implement Algorithm 2.1 so that only two inner products are computed at every step, for either μ_n and ν_n in (2.3), or for ν_n in (2.4). The crucial part of Algorithm 2.1 is the look-ahead strategy used in step 1). As described in [18], the decision in 1) is based on three checks. For a regular step, it is necessary that $D^{(l)}$ be nonsingular. Therefore, one of the checks monitors the size of smallest singular value of $D^{(l)}$. The other two checks attempt to ensure the linear independence of the Lanczos vectors. The algorithm monitors the size of the components μ_n and ν_n along the two previous blocks $V^{(l)}$ and $V^{(l-1)}$, respectively $W^{(l)}$ and $W^{(l-1)}$, in (2.3), and performs a regular step only if these terms do not dominate the components Av_n and $A^T w_n$ in the new Krylov spaces. Complete details of the implementation of the look-ahead Lanczos Algorithm 2.1 are given in [18].

We note that, in (2.4), ζ_n and η_n are arbitrary inner recurrence coefficients, with $\zeta_{n_k} = 0$. One possibility is to choose the Chebyshev iteration [25, 39] parameters for ζ_n and η_n. However, since the length of look-ahead steps is usually small, the choice of the inner recurrence coefficients is not crucial; in our experience, $\zeta_n = 1$ and, if $n \neq n_k$, $\eta_n = 1$, works satisfactorily. Indeed, with the look-ahead strategy proposed in [18], the algorithm performs mostly regular steps, and typically, only a few look-ahead steps of length bigger than 1 occur. In our experiments, the longest look-ahead step we encountered was of length 4.

For later use, we remark that the recurrences in (2.3) and (2.4) can be written compactly in matrix form. For example, for the right Lanczos vectors v_n, we have

$$AV_n = V_{n+1}H_n, \tag{2.5}$$

where

$$V_n := [\, v_1 \quad v_2 \quad \cdots \quad v_n \,],$$

and

$$H_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_l \\ \vdots & & \ddots & \gamma_l & \alpha_l \\ 0 & \cdots & \cdots & 0 & \gamma_{l+1} \end{bmatrix} \in \mathbb{C}^{(n+1)\times n}$$

is a block tridiagonal matrix. The diagonal blocks α_k are square unreduced upper Hessenberg matrices, whose size is equal to the number of vectors in the corresponding block $V^{(k)}$. The matrices γ_k have only one nonzero element, in their upper right corner, and thus H_n is an upper Hessenberg matrix, with full rank

$$\operatorname{rank} H_n = n. \tag{2.6}$$

If only regular steps 2) are performed, then the Algorithm 2.1 reduces to the classical Lanczos process. In this case, the blocks $V^{(k)}$ and $W^{(k)}$ consist of just the single vector v_k and w_k, respectively, and the orthogonality relations (2.2) now read:

$$w_j^T v_k = \begin{cases} \delta_k \neq 0 & \text{if } j = k, \\ 0 & \text{if } j \neq k, \end{cases} \qquad j,k = 1,\ldots,n. \tag{2.7}$$

Moreover, H_n is just a scalar tridiagonal matrix. The condition $\delta_k \neq 0$ in (2.7) is crucial, since each step of the classical Lanczos algorithm involves a division by δ_k. The point is that one cannot guarantee $\delta_k \neq 0$, and in fact, when $\delta_k = 0$ with $v_k \neq 0$ and $w_k \neq 0$, the algorithm breaks down. Note that $\delta_k \approx 0$ signals a near-breakdown of the procedure.

Algorithm 2.1 will handle exact and near-breakdowns in the classical Lanczos process, except for the special event of an incurable breakdown [52]. These are situations when the look-ahead procedure would build an infinite block, without ever finding a nonsingular $D^{(l)}$. Taylor [52] has shown in his Mismatch Theorem that in case of an incurable breakdown, one can still recover eigenvalue information. For linear systems, an incurable breakdown would require restarting the procedure with a different choice of starting vectors. Fortunately, in practice round-off errors will make an incurable breakdown highly unlikely.

Finally, we remark that, for the important class of p-cyclic matrices A, exact breakdowns in the Lanczos process occur in a regular pattern. In this case, as was shown by Freund, Golub, and Hochbruck [16], look-ahead steps are absolutely necessary if one wants to exploit the p-cyclic structure. For details of a look-ahead Lanczos algorithm for p-cyclic matrices, we refer the reader to [16].

2.2 Historical Remarks and Related Work

The problem of breakdowns in the classical Lanczos algorithm has been known from the beginning. Although a rare event in practice, the possibility of breakdowns has certainly brought the method into discredit and has prevented many people from actually using the algorithm. On the other hand, as was demonstrated by Cullum and Willoughby [8], the Lanczos process—even without look-ahead—is a powerful tool for sparse matrix computation.

The Lanczos method has intimate connections with many other areas of Mathematics, such as formally orthogonal polynomials (FOPs), Padé approximation, Hankel matrices, control theory, and

coding theory. The problem of breakdowns has a corresponding formulation in all of these areas, and remedies for breakdowns in these different settings have been known for quite some time. For example, the breakdown in the Lanczos process is equivalent to a breakdown of the generic three-term recurrence relation for FOPs, and it is well known how to overcome such breakdowns by modifying the recursions for FOPs (see [26, 9, 31] and the references given there). In the context of the partial realization problem in control theory, remedies for breakdowns were given in [36, 27]. The Lanczos process is also closely related to fast algorithms for the factorization of Hankel matrices, and again it was known how to overcome possible breakdowns of these algorithms (see [32, 22] and the references therein). However, in all these cases, only the problem of exact breakdowns was addressed.

The look-ahead Lanczos algorithm of Taylor [52] and Parlett, Taylor, and Liu [44] was the first procedure that remedies both exact and near-breakdowns. We point out that their implementation is different from Algorithm 2.1. In particular, it always requires more work per step than Algorithm 2.1, and it does not reduce to the classical Lanczos process in the absence of look-ahead steps. Furthermore, in [52, 44], details are given only for the case of look-ahead steps of size 2, and their algorithm does not generalize easily to blocks of more than two vectors.

In recent years, there has been a revival of the nonsymmetric Lanczos algorithm, and since 1990, in addition to the papers we have already cited in this section, there are several others dealing with various aspects of the Lanczos process. We refer the reader to [2, 3, 4, 22, 29, 34, 43] and the references given therein.

3 The Quasi-Minimal Residual Approach

We now return to linear systems (1.1). From now on, it is always assumed that the matrix A is nonsingular. In this section, we describe the QMR method. The procedure was first proposed by Freund [13, 15] for the case of complex symmetric matrices $A = A^T$, and then extended by Freund and Nachtigal [19] for the case of general non-Hermitian matrices.

3.1 The Standard QMR Algorithm

Recall that the nth iterate of any Krylov subspace method is of the form (1.2). If now we choose

$$v_1 = r_0 \tag{3.1}$$

in Algorithm 2.1, then, by (2.1), the Lanczos vectors $v_1, \ldots, v_n$ span $\mathcal{K}_n(r_0, A)$; hence we can write

$$x_n = x_0 + V_n z_n,$$

for some $z_n \in \mathbf{C}^n$. Together with (3.1) and (2.5), this gives the corresponding residual vector

$$r_n = r_0 - AV_n z_n = V_{n+1}(e_1 - H_n z_n), \tag{3.2}$$

where e_1 denotes the first unit vector in $\mathbf{R}^{n+1}$. As V_{n+1} is not unitary, it is not possible to minimize the Euclidean norm of the residual without expending $\mathcal{O}(Nn^2)$ work and $\mathcal{O}(Nn)$ storage. Instead, one minimizes just some weighted Euclidean norm of the coefficient vector in (3.2). More precisely, let

$$\Omega_n = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, \ldots, n+1, \tag{3.3}$$

be a weighting matrix. Then $z_n \in \mathbf{C}^n$ is chosen as the solution of the least squares problem

$$\|\omega_1 e_1 - \Omega_n H_n z_n\| = \min_{z \in \mathbf{C}^n} \|\omega_1 e_1 - \Omega_n H_n z\|. \tag{3.4}$$

Note that, in view of (2.6) and (3.3), the problem (3.4) always has a unique solution. Usually, the weights in (3.4) are chosen as $\omega_j \equiv \|v_j\|$, which means that all components in

$$r_n = \left(V_{n+1}\Omega_n^{-1}\right)(\omega_1 e_1 - \Omega_n H_n z_n)$$

are treated equally.

The least-squares problem (3.4) can be solved by standard techniques based on a QR decomposition of $\Omega_n H_n$. One computes a unitary matrix $Q_n \in \mathbf{C}^{(n+1)\times(n+1)}$ and an upper triangular matrix $R_n \in \mathbf{C}^{n\times n}$ such that

$$Q_n \Omega_n H_n = \begin{bmatrix} R_n \\ 0 \end{bmatrix}, \tag{3.5}$$

and then obtains z_n from

$$z_n = R_n^{-1} t_n, \quad t_n = \omega_1 [\, I_n \quad 0 \,] Q_n e_1, \tag{3.6}$$

which gives

$$x_n = x_0 + V_n R_n^{-1} t_n. \tag{3.7}$$

This gives the following QMR algorithm.

Algorithm 3.1 (QMR algorithm)

0) *Choose* $x_0 \in \mathbf{C}^N$ *and set* $v_1 = r_0 = b - Ax_0$.
Choose $w_1 \in \mathbf{C}^N$ *with* $w_1 \neq 0$.
For $n = 1, 2, \ldots,$ *do* :

1) *Perform the nth iteration of the look-ahead Lanczos Algorithm 2.1.*
This yields matrices V_n, V_{n+1}, *and* H_n *which satisfy* (2.5).

2) *Update the QR factorization* (3.5) *of* H_n *and the vector* t_n *in* (3.6).

3) *Compute* x_n *from* (3.7). *If* x_n *has converged, stop.*

We note that Q_n in (3.5) is just a product of n Givens rotations, and thus the vector t_n is easily updated in step 2). Also, as H_n is block tridiagonal, R_n also has a block structure that is used in step 3) to update x_n using only short recurrences. For complete details, see [19].

The quasi-minimization (3.4) is strong enough to obtain convergence results for QMR. One can derive error bounds for QMR that are comparable to those for GMRES. Also, it is possible to relate the norms of the QMR and GMRES residual vectors. This is in contrast to BCG and methods derived from BCG, for which no such convergence results are known. Finally, if desired, one can recover BCG iterates from the QMR Algorithm 3.1, at the expense of only one additional SAXPY per step. For these and other properties of QMR, we refer the reader to [19, 41].

Algorithm 3.1 is only one possible implementation of the QMR method. Instead of using three-term recurrences as in the underlying look-ahead Lanczos Algorithm 2.1, the basis vectors $\{v_n\}$ and $\{w_n\}$ can also be generated by coupled two-term recurrences. Empirical observations indicate that,

in finite precision arithmetic, the latter approach is more robust than the former. Details of such an implementation of the QMR method based on coupled two-term recurrences with look-ahead are presented in [20].

FORTRAN 77 implementations of the QMR Algorithm 3.1 and of the look-ahead Lanczos Algorithm 2.1 are available electronically from netlib.[1]

3.2 BCG and an Implementation of QMR without Look-Ahead

We now look at BCG in more detail. The BCG algorithm attempts to generate iterates x_n^{BCG} that are characterized by the Galerkin condition

$$x_n^{\mathrm{BCG}} \in x_0 + K_n(r_0, A) \quad \text{and} \quad w^T(b - Ax_n^{\mathrm{BCG}}) = 0 \quad \text{for all} \quad w \in K_n(w_1, A^T). \tag{3.8}$$

Unfortunately, such iterates need not exist for every n, and this is one source of possible breakdowns in BCG.

As noted already, BCG is closely related to the classical Lanczos process. More precisely, the BCG residual vectors are just scalar multiples of the right Lanczos vectors:

$$r_n = b - Ax_n^{\mathrm{BCG}} = \theta_n v_{n+1}, \quad \theta_n \in \mathbf{C}, \quad \theta_n \neq 0. \tag{3.9}$$

In addition to r_n, the BCG algorithm also involves a second sequence of vectors $\tilde{r}_n \in K_{n+1}(\tilde{r}_0, A^T)$. Here $\tilde{r}_0 \in \mathbf{C}^N$ is an arbitrary nonzero starting vector; usually one sets $\tilde{r}_0 = r_0$ or chooses $\tilde{r}_0$ as a vector with random coefficients. The vectors $\tilde{r}_n$ are connected with the left vectors generated by the classical Lanczos process:

$$\tilde{r}_n = \tilde{\theta}_n w_{n+1}, \quad \tilde{\theta}_n \in \mathbf{C}, \quad \tilde{\theta}_n \neq 0. \tag{3.10}$$

From (3.9) and (3.10), we have

$$\tilde{r}_{n-1}^T r_{n-1} = \tilde{\theta}_{n-1}\theta_{n-1} w_n^T v_n. \tag{3.11}$$

Recall from (2.7) that the classical Lanczos process breaks down if $w_n^T v_n = 0$ with $v_n \neq 0$ and $w_n \neq 0$. In view of (3.11), this is equivalent to

$$\tilde{r}_{n-1}^T r_{n-1} = 0, \quad r_{n-1} \neq 0, \quad \tilde{r}_{n-1} \neq 0. \tag{3.12}$$

As Algorithm 3.2 below shows, BCG also breaks down if (3.12) occurs. In addition to (3.12), there is a second source of breakdowns in BCG, namely

$$\tilde{q}_{n-1}^T A q_{n-1} = 0, \quad q_{n-1} \neq 0, \quad \tilde{q}_{n-1} \neq 0. \tag{3.13}$$

Here q_{n-1} and $\tilde{q}_{n-1}$ are the vectors generated by Algorithm 3.2 below. It can be shown (see, e.g., [46]) that a breakdown of the kind (3.13) occurs if, and only if, no Galerkin iterate x_n^{BCG} with (3.8) exists.

Unlike the BCG iterates, the QMR iterates are always well defined by (2.6). In particular, breakdowns of the kind (3.13) can be excluded in the QMR Algorithm 3.1. We stress that this

[1] To obtain the codes, one needs to send a message consisting of the single line "send lalqmr from misc" to *netlib@ornl.gov*.

remains true even if, in the QMR Algorithm 3.1, one uses the classical Lanczos process in step 1). Of course, the use of the look-ahead Lanczos Algorithm 2.1 avoids breakdowns of the first kind (3.12), except for incurable breakdowns.

As already noted, existing BCG iterates can be easily obtained from quantities generated by the QMR Algorithm 2.1. Therefore, QMR can also be viewed as a stable implementation of BCG. It is also possible to reverse the roles of the two algorithms and to get QMR iterates directly from the BCG algorithm. Such an implementation of QMR without look-ahead was derived by Freund and Szeto in [21], and is as follows.

Algorithm 3.2 (QMR without look-ahead from BCG)

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $x_0^{\mathrm{QMR}} = x_0^{\mathrm{BCG}} = x_0$.

Set $q_0 = r_0 = b - Ax_0$, $\hat{p}_0 = 0$, $\tau_0 = \omega_1\|r_0\|$, $\vartheta_0 = 0$.

Choose $\tilde{r}_0 \in \mathbb{C}^N$, $\tilde{r}_0 \neq 0$, *and set* $\tilde{q}_0 = \tilde{r}_0$, $\rho_0 = \tilde{r}_0^T r_0$.

For $n = 1, 2, \ldots$, *do :*

1) *Set* $\sigma_{n-1} = \tilde{q}_{n-1}^T A q_{n-1}$.

If $\sigma_{n-1} = 0$, *stop. Otherwise, compute*

$$\begin{aligned} \alpha_{n-1} &= \rho_{n-1}/\sigma_{n-1}, \\ r_n &= r_{n-1} - \alpha_{n-1} A q_{n-1}, \\ \tilde{r}_n &= \tilde{r}_{n-1} - \alpha_{n-1} A^T \tilde{q}_{n-1}. \end{aligned}$$

If BCG *iterates are desired, set*

$$x_n^{\mathrm{BCG}} = x_{n-1}^{\mathrm{BCG}} + \alpha_{n-1} q_{n-1}.$$

2) *Compute*

$$\begin{aligned} \vartheta_n &= \frac{\omega_n\|r_n\|}{\tau_{n-1}}, \quad c_n = \frac{1}{\sqrt{1+\vartheta_n^2}}, \quad \tau_n = \tau_{n-1}\vartheta_n c_n, \\ \hat{p}_n &= c_n^2 \vartheta_n^2 \hat{p}_{n-1} + c_n^2 \alpha_{n-1} q_{n-1}, \\ x_n^{\mathrm{QMR}} &= x_{n-1}^{\mathrm{QMR}} + \hat{p}_n. \end{aligned}$$

3) *If* $\rho_{n-1} = 0$, *stop. Otherwise, compute*

$$\begin{aligned} \rho_n &= \tilde{r}_n^T r_n, \quad \beta_n = \rho_n/\rho_{n-1}, \\ q_n &= r_n + \beta_n q_{n-1}, \\ \tilde{q}_n &= \tilde{r}_n + \beta_n \tilde{q}_{n-1}. \end{aligned}$$

We remark that, exact for the additional update in step 2), this algorithm is just the classical BCG. Of course, unlike the QMR Algorithm 3.1, the implementation of QMR in Algorithm 3.2 can breakdown due to (3.12) and (3.13).

Algorithm 3.2 is only one of several possible implementations of the BCG approach; see [34, 28] for an overview of the different BCG variants. As in the nonsymmetric Lanczos process, exact and near-breakdowns in the BCG methods can be avoided by incorporating look-ahead procedures. Such look-ahead BCG algorithms have been proposed by Joubert [34] and Gutknecht [29].

4 Transpose-Free Methods

Krylov subspace methods such as BCG and QMR, which are based directly on the Lanczos process, involve matrix-vector products with A and A^T. This is a disadvantage for certain applications, where A^T is not readily available. It is possible to devise Lanczos-based Krylov subspace methods that do not involve the transpose of A. In this section, we give an overview of such transpose-free schemes.

First, we consider the QMR algorithm. As pointed out by Freund and Zha [23], in principle it is always possible to eliminate A^T altogether, by choosing the starting vector w_1 suitably. This observation is based on the fact that any square matrix is similar to its transpose. In particular, there always exists a nonsingular matrix P such that

$$A^T P = PA. \tag{4.1}$$

Now suppose that in the QMR Algorithm 3.1 we choose the special starting vector $w_1 = Pv_1$. Then, with (4.1), one readily verifies that the vectors generated by look-ahead Lanczos Algorithm 2.1 satisfy

$$w_n = Pv_n \quad \text{for all} \quad n. \tag{4.2}$$

Hence, instead of updating the left Lanczos vectors $\{w_n\}$ by means of the recursions in (2.3) or (2.4), they can be computed directly from (4.2). The resulting QMR algorithm no longer involves the transpose of A; in exchange, it requires one matrix-vector multiplication with P in each iteration step. Therefore, this approach is only viable for special classes of matrices A, for which one can find a matrix P satisfying (4.1) easily, and for which, at the same time, matrix-vector products with P can be computed cheaply. The most trivial case are real or complex symmetric matrices $A = A^T$, which fulfill (4.1) with $P = I$. Another simple case are Toeplitz matrices A, i.e., matrices whose entries are constant along each diagonal. Toeplitz matrices satisfy (4.1) with $P = J$, where

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

is the $N \times N$ antidiagonal identity matrix. Finally, the condition (4.1) is also fulfilled for matrices of the form

$$A = TM^{-1}, \quad P = M^{-1},$$

where T and M are real symmetric matrices and M is nonsingular. Matrices of this type arise when real symmetric linear systems $Tx = b$ are preconditioned by M. The resulting QMR algorithm for the solution of preconditioned symmetric linear system has the same work and storage requirements as preconditioned SYMMLQ or MINRES [42]. However, the QMR approach is more general, in that it can be combined with any nonsingular symmetric preconditioner M, while SYMMLQ and MINRES require M to be positive definite (see, e.g., [24]). For strongly indefinite matrices T, the use of indefinite preconditioners M typically leads to considerably faster convergence; see [23] for numerical examples.

Next, we turn to transpose-free variants of the BCG method. Sonneveld [50] with his CGS algorithm was the first to devise a transpose-free BCG-type scheme. Note that, in the BCG

Algorithm 3.2, the matrix A^T appears merely in the update formulas for the vectors $\tilde{r}_n$ and $\tilde{q}_n$. On the other hand, these vectors are then used only for the computation of the vector products $\rho_n = \tilde{r}_n^T r_n$ and $\sigma_n = \tilde{q}_n^T A q_n$. Sonneveld observed that, by rewriting these products, the transpose can be eliminated from the formulas, while at the same time one obtains iterates

$$x_{2n} \in x_0 + \mathcal{K}_{2n}(r_0, A), \quad n = 1, 2, \ldots, \tag{4.3}$$

that are contained in a Krylov subspace of twice the dimension, as compared to BCG. First, we consider ρ_n. From Algorithm 3.2 it is obvious that

$$r_n = \psi_n(A) r_0 \quad \text{and} \quad \tilde{r}_n = \psi_n(A^T) \tilde{r}_0, \tag{4.4}$$

where ψ_n is the nth residual polynomials of the BCG process. With (4.4), one obtains the identity

$$\rho_n = \tilde{r}_0^T \left(\psi_n(A)\right)^2 r_0, \tag{4.5}$$

which shows that ρ_n can be computed without using A^T. Similarly,

$$q_n = \varphi_n(A) r_0 \quad \text{and} \quad \tilde{q}_n = \varphi_n(A^T) \tilde{r}_0,$$

for some polynomial $\varphi_n \in \mathcal{P}_n$, and hence

$$\sigma_n = \tilde{r}_0^T A \left(\varphi_n(A)\right)^2 r_0. \tag{4.6}$$

By rewriting the vector recursions in Algorithm 3.2 in terms of ψ_n and φ_n and by squaring the resulting polynomial relations, Sonneveld showed that the vectors in (4.5) and (4.6) can be updated by means of short recursions. Furthermore, the actual iterates (4.3) generated by CGS are characterized by

$$r_{2n}^{\mathrm{CGS}} = b - A x_{2n} = \left(\psi_n^{\mathrm{BCG}}(A)\right)^2 r_0. \tag{4.7}$$

Hence the CGS residual polynomials $\psi_{2n}^{\mathrm{CGS}} = \left(\psi_n^{\mathrm{BCG}}\right)^2$ are just the squared BCG polynomials. As pointed out earlier, BCG typically exhibits a rather erratic convergence behavior. As is clear from (4.7), these effects are magnified in CGS, and CGS typically accelerates convergence as well as divergence of BCG. Moreover, there are cases for which CGS diverges, while BCG still converges.

For this reason, more smoothly converging variants of CGS have been sought. Van der Vorst [53] was the first to propose such a method. His Bi-CGSTAB again produces iterates of the form (4.3), but instead of squaring the BCG polynomials as in (4.7), the residual vector is now of the form

$$r_{2n} = \psi_n^{\mathrm{BCG}}(A) \chi_n(A) r_0.$$

Here $\chi_n \in \mathcal{P}_n$, with $\chi_n(0) = 1$, is a polynomial that is updated from step to step by adding a new linear factor:

$$\chi_n(\lambda) \equiv (1 - \eta_n \lambda) \chi_{n-1}(\lambda). \tag{4.8}$$

The free parameter η_n in (4.8) is determined by a local steepest descent step, i.e., η_n is the optimal solution of

$$\min_{\eta \in \mathbb{C}} \|(I - \eta A) \chi_{n-1}(A) \psi_n^{\mathrm{BCG}}(A) r_0\|.$$

Due to the steepest descent steps, Bi-CGSTAB typically has much smoother convergence behavior than BCG or CGS. However, the norms of the Bi-CGSTAB residuals may still oscillate considerably for difficult problems. Finally, Gutknecht [30] has noted that, for real A, the polynomials χ_n will always have real roots only, even if A has complex eigenvalues. He proposed a variant of Bi-CGSTAB with polynomials (4.8) that are updated by quadratic factors in each step and thus can have complex roots in general.

In the CGS algorithm, the iterates (4.3) are updated by means of a formula of the form

$$x_{2n}^{\mathrm{CGS}} = x_{2(n-1)}^{\mathrm{CGS}} + \alpha_{n-1}(y_{2n-1} + y_{2n}). \tag{4.9}$$

Here the vectors $y_1, y_2, \ldots, y_{2n}$ satisfy

$$\mathrm{span}\{y_1, y_2, \ldots, y_m\} = \mathcal{K}_m(r_0, A), \quad m = 1, 2, \ldots, 2n.$$

In other words, in each iteration of the CGS algorithm two search directions y_{2n-1} and y_{2n} are available, while the actual iterate is updated by the one-dimensional step (4.9) only. Based on this observation, Freund [14] has proposed a variant of CGS that makes use of all available search directions. More precisely, instead of one iterate x_{2n}^{CGS} per step it produces two iterates x_{2n-1} and x_{2n} of the form

$$x_m = x_0 + [\, y_1 \quad y_2 \quad \cdots \quad y_m \,] z_m, \quad z_m \in \mathbb{C}^m. \tag{4.10}$$

Furthermore, the free parameter vector z_m in (4.10) can be chosen such that the iterates satisfy a quasi-minimal residual condition, similar to the quasi-minimization property of the QMR Algorithm 3.4. For this reason, the resulting scheme is called transpose-free quasi-minimal residual algorithm (TFQMR). For details, we refer the reader to [14], where the following implementation of TFQMR is derived.

Algorithm 4.1 (TFQMR algorithm)

0) *Choose* $x_0 \in \mathbb{C}^N$.

Set $w_1 = y_1 = r_0 = b - Ax_0$, $v_0 = Ay_1$, $d_0 = 0$.

Set $\tau_0 = \|r_0\|$, $\vartheta_0 = 0$, $\eta_0 = 0$.

Choose $\tilde{r}_0 \in \mathbb{C}^N$, $\tilde{r}_0 \neq 0$, *and set* $\rho_0 = \tilde{r}_0^T r_0$.

For $n = 1, 2, \ldots,$ *do* :

1) *Compute*

$$\begin{aligned} \sigma_{n-1} &= \tilde{r}_0^T v_{n-1}, \\ \alpha_{n-1} &= \rho_{n-1}/\sigma_{n-1}, \\ y_{2n} &= y_{2n-1} - \alpha_{n-1} v_{n-1}. \end{aligned}$$

2) *For* $m = 2n-1, 2n$ *do* :

Compute

$$\begin{aligned} w_{m+1} &= w_m - \alpha_{n-1} A y_m, \\ \vartheta_m &= \|w_{m+1}\|/\tau_{m-1}, \ c_m = 1/\sqrt{1+\vartheta_m^2}, \\ \tau_m &= \tau_{m-1}\vartheta_m c_m, \ \eta_m = c_m^2 \alpha_{n-1}, \\ d_m &= y_m + (\vartheta_{m-1}^2 \eta_{m-1}/\alpha_{n-1}) d_{m-1}, \\ x_m &= x_{m-1} + \eta_m d_m. \end{aligned}$$

If x_m has converged, stop.

3) *Compute*

$$\begin{aligned} \rho_n &= \tilde{r}_0^T w_{2n+1}, \\ \beta_n &= \rho_n / \rho_{n-1}, \\ y_{2n+1} &= w_{2n+1} + \beta_n y_{2n}, \\ v_n &= A y_{2n+1} + \beta_n (A y_{2n} + \beta_n v_{n-1}). \end{aligned}$$

We would like to point out that the iterates generated by the QMR Algorithm 3.1 and the TFQMR Algorithm 4.1 are different in general.

Another transpose-free QMR method was proposed by Chan, de Pillis, and Van der Vorst [5]. Their scheme is mathematically equivalent to the QMR Algorithm 3.1, when the latter is based on the classical Lanczos process without look-ahead. The method first uses a transpose-free squared version of the Lanczos algorithm (see, e.g., Gutknecht [28]) to generate the scalar tridiagonal Lanczos matrix H_n. The right Lanczos vectors v_n are then computed by running the standard Lanczos recurrence, and finally the QMR iterates are obtained as in Algorithm 3.1. Freund and Szeto [21] have derived yet another transpose-free QMR scheme, which is modeled after CGS and is based on squaring the residual polynomials of the standard QMR Algorithm 3.1. However, the algorithm given in [5] and the squared QMR approach both require three matrix-vector products with A at each iteration, and hence they are more expensive than CGS, Bi-CGSTAB, or TFQMR, which involve only two such products per step.

Finally, we remark that none of the transpose-free methods considered in this section, except for Freund and Zha's simplified QMR algorithm based on (4.1), addresses the problem of breakdowns. Indeed, in exact arithmetic, all these schemes break down every time a breakdown occurs in the BCG Algorithm 3.2. Practical look-ahead techniques for avoiding exact and near-breakdowns in these transpose-free methods still have to be developed.

5 Concluding Remarks

In this paper, we have covered only some of the recent advances in iterative methods for non-Hermitian linear systems. A more extensive survey of recent developments in this field can be found in [17].

The introduction of CGS in the 1980s spurred renewed interest in the nonsymmetric Lanczos algorithm, with most of the effort directed towards obtaining a method with better convergence properties than BCG or CGS. Several BCG-based algorithms were proposed, such as Bi-CGSTAB, introduced by Van der Vorst [53]. The quasi-minimal residual technique was introduced by Freund [13, 15] in the context of complex symmetric systems, then later coupled with a new variant of the look-ahead Lanczos approach to obtain a general non-Hermitian QMR algorithm [19]. Finally, several transpose-free algorithms based on QMR have been introduced recently, which trade the multiplication by A^T for one or more multiplications by A. However, their convergence properties are not well understood, and none of these algorithms have been combined with look-ahead techniques yet. In general, it seems that the transpose-free methods have more numerical problems than the corresponding methods that use A^T, and more research is needed into studying their behavior.

Finally, even though the field of iterative methods has made great progress in the last few years, it is still in its infancy, especially with regard to the packaged software available. Whereas there are

well-established robust general-purpose solvers based on direct methods, the same cannot be said about solvers based on iterative methods. There are no established iterative packages of the same robustness and wide acceptance as, for example, the LINPACK library, and as a result many of the scientists who use iterative methods write their own specialized solvers. We feel that this situation needs to change, and we would like to encourage researchers to provide code for their methods.

References

[1] O. Axelsson, Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations, *Linear Algebra Appl.* **29** (1980), pp. 1–16.

[2] D.L. Boley, S. Elhay, G.H. Golub, and M.H. Gutknecht, Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights, *Numer. Algorithms* 1 (1991), pp. 21–43.

[3] D.L. Boley and G.H. Golub, The nonsymmetric Lanczos algorithm and controllability, *Systems Control Lett.* **16** (1991), pp. 97–105.

[4] C. Brezinski, M. Redivo Zaglia, and H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, Technical Report ANO–239, Université des Sciences et Techniques de Lille Flandres-Artois, Villeneuve d'Ascq, France, 1991.

[5] T.F. Chan, L. de Pillis, and H.A. Van der Vorst, A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems, Technical Report CAM 91-17, Department of Mathematics, University of California, Los Angeles, CA, 1991.

[6] P. Concus and G.H. Golub, A generalized conjugate gradient method for nonsymmetric systems of linear equations, in *Computing Methods in Applied Sciences and Engineering* (R. Glowinski and J.L. Lions, eds.), Lecture Notes in Economics and Mathematical Systems 134, Springer, Berlin, 1976, pp. 56–65.

[7] 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 Press, New York, 1976, pp. 309–332.

[8] J. Cullum and R.A. Willoughby, A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices, in *Large Scale Eigenvalue Problems* (J. Cullum and R.A. Willoughby, eds.), North-Holland, Amsterdam, 1986, pp. 193–240.

[9] A. Draux, *Polynômes Orthogonaux Formels - Applications*, Lecture Notes in Mathematics 974, Springer, Berlin, 1983.

[10] V. Faber and T. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, *SIAM J. Numer. Anal.* **21** (1984), pp. 352–362.

[11] V. Faber and T. Manteuffel, Orthogonal error methods, *SIAM J. Numer. Anal.* **24** (1987), pp. 170–187.

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

[13] R.W. Freund, Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices, RIACS Technical Report 89.54, NASA Ames Research Center, Moffett Field, CA, 1989.

[14] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, RIACS Technical Report 91.18, NASA Ames Research Center, Moffett Field, CA, 1991.

[15] R.W. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, *SIAM J. Sci. Stat. Comput.* **13** (1992), to appear.

[16] R.W. Freund, G.H. Golub, and M. Hochbruck, Krylov subspace methods for non-Hermitian p-cyclic matrices, Technical Report, RIACS, NASA Ames Research Center, Moffett Field, CA, 1992.

[17] R.W. Freund, G.H. Golub, and N.M. Nachtigal, Iterative solution of linear systems, RIACS Technical Report 91.21, NASA Ames Research Center, Moffett Field, 1991, to appear in Acta Numerica.

[18] R.W. Freund, M.H. Gutknecht and N.M. Nachtigal (1991b), An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Technical Report 91.09, RIACS, NASA Ames Research Center, Moffett Field, CA, 1991, *SIAM J. Sci. Stat. Comput.*, to appear.

[19] R.W. Freund and N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, *Numer. Math.* **60** (1991), pp. 315–339.

[20] R.W. Freund, N.M. Nachtigal, and T. Szeto, An implementation of the QMR method based on coupled two-term recurrences, Technical Report, RIACS, NASA Ames Research Center, Moffett Field, CA, 1992.

[21] R.W. Freund and T. Szeto, A quasi-minimal residual squared algorithm for non-Hermitian linear systems, Technical Report 91.26, RIACS, NASA Ames Research Center, Moffett Field, CA, 1991.

[22] R.W. Freund and H. Zha, A look-ahead algorithm for the solution of general Hankel systems, Technical Report 91.24, RIACS, NASA Ames Research Center, Moffett Field, CA, 1991.

[23] R.W. Freund and H. Zha, Simplifications of the nonsymmetric Lanczos process and a new algorithm for solving indefinite symmetric linear systems, Technical Report, RIACS, NASA Ames Research Center, Moffett Field, CA, 1992.

[24] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders, Preconditioners for indefinite systems arising in optimization, Technical Report SOL 90-8, Stanford University, Stanford, CA, 1990.

[25] G.H. Golub and R.S. Varga, Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods, *Numer. Math.* **3** (1961), pp. 147–168.

[26] W.B. Gragg, Matrix interpretations and applications of the continued fraction algorithm, *Rocky Mountain J. Math.* **4** (1974), pp. 213–225.

[27] W.B. Gragg and A. Lindquist, On the partial realization problem, *Linear Algebra Appl.* **50** (1983), pp. 277–319.

[28] M.H. Gutknecht, The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the QD algorithm, in *Proceedings of the Copper Mountain Conference on Iterative Methods*, 1990.

[29] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, IPS Research Report No. 90–16, ETH Zürich, Switzerland, 1990.

[30] M.H. Gutknecht, Variants of BiCGStab for matrices with complex spectrum, IPS Research Report No. 91–14, ETH Zürich, Switzerland, 1991.

[31] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, *SIAM J. Matrix Anal. Appl.* **13** (1992), to appear.

[32] G. Heinig and K. Rost, Algebraic methods for Toeplitz-like matrices and operators, Birkhäuser, Basel, 1984.

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

[34] W.D. Joubert, Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations, Ph.D. Dissertation, University of Texas, Austin, TX, 1990.

[35] W.D. Joubert and D.M. Young, Necessary and sufficient conditions for the simplification of generalized conjugate-gradient algorithms, *Linear Algebra Appl.* **88/89** (1987), pp. 449–485.

[36] S.-Y. Kung, Multivariable and multidimensional systems: analysis and design, Ph.D. Dissertation, Stanford University, Stanford, CA, 1977.

[37] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Natl. Bur. Stand.* **45** (1950), pp. 255–282.

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

[39] T.A. Manteuffel, The Tchebychev iteration for nonsymmetric linear systems, *Numer. Math.* **28** (1977), pp. 307–327.

[40] J.A. Meijerink and H.A. van der Vorst, An iterative solution for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comp.* **31**, 1977, pp. 148–162.

[41] N.M. Nachtigal, A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non-Hermitian linear systems, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[42] C.C. Paige and M.A. Saunders (1975), 'Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.* **12** (1975), pp. 617–629.

[43] P.N. Parlett, Reduction to tridiagonal form and minimal realizations, *SIAM J. Matrix Anal. Appl.* **13** (1992), to appear.

[44] B.N. Parlett, D.R. Taylor, and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comp.* **44** (1985), pp. 105–124.

[45] 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 Press, New York, 1971, pp. 231–253.

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

[47] Y. Saad, Krylov subspace methods on supercomputers, *SIAM J. Sci. Stat. Comput.* **10** (1989), pp. 1200–1232.

[48] Y. Saad and M.H. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, *Math. Comp.* **44** (1985), pp. 417–424.

[49] Y. Saad and M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **7** (1986), pp. 856–869.

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

[51] J. Stoer, Solution of large linear systems of equations by conjugate gradient type methods, in *Mathematical Programming - The State of the Art* (A. Bachem, M. Grötschel and B. Korte, eds.), Springer, Berlin, 1983, pp. 540–565.

[52] D.R. Taylor, Analysis of the look ahead Lanczos algorithm, Ph.D. Dissertation, University of California, Berkeley, CA, 1982.

[53] H.A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **13** (1992), to appear.

[54] P.K.W. Vinsome, Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, in *Proceedings of the Fourth Symposium on Reservoir Simulation*, Society of Petroleum Engineers of AIME, Los Angeles, CA, 1976.

[55] V.V. Voevodin, The problem of a non-selfadjoint generalization of the conjugate gradient method has been closed, *USSR Comput. Math. and Math. Phys.* **23** (1983), pp. 143–144.

[56] O. Widlund, A Lanczos method for a class of nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.* **15** (1978), pp. 801–812.

RIACS
Mail Stop 230-5
NASA Ames Research Center
Moffett Field, CA 94035