SIAM J. SCI. COMPUT.
Vol. 14, No. 1, pp. 137–158, January 1993

AN IMPLEMENTATION OF THE LOOK-AHEAD LANCZOS ALGORITHM FOR NON-HERMITIAN MATRICES*

ROLAND W. FREUND†‡, MARTIN H. GUTKNECHT§, AND NOËL M. NACHTIGAL†

Abstract. The nonsymmetric Lanczos method can be used to compute eigenvalues of large sparse non-Hermitian matrices or to solve large sparse non-Hermitian linear systems. However, the original Lanczos algorithm is susceptible to possible breakdowns and potential instabilities. An implementation of a look-ahead version of the Lanczos algorithm is presented that, except for the very special situation of an incurable breakdown, overcomes these problems by skipping over those steps in which a breakdown or near-breakdown would occur in the standard process. The proposed algorithm can handle look-ahead steps of any length and requires the same number of matrix–vector products and inner products as the standard Lanczos process without look-ahead.

Key words. Lanczos method, orthogonal polynomials, look-ahead steps, eigenvalue problems, iterative methods, non-Hermitian matrices, sparse linear systems

AMS(MOS) subject classifications. 65F15, 65F10

1. Introduction.

In 1950, Lanczos [20] proposed a method for successive reduction of a given, in general non-Hermitian, $N \times N$ matrix A to tridiagonal form. More precisely, the Lanczos procedure generates a sequence $H^{(n)}$, $n = 1, 2, \dots$, of $n \times n$ tridiagonal matrices which, in a certain sense, approximate A. Furthermore, in exact arithmetic and if no breakdown occurs, the Lanczos method terminates after at most L $(\leq N)$ steps with $H^{(L)}$ a tridiagonal matrix that represents the restriction of A or A^T to an A-invariant or A^T-invariant subspace of $\mathbb{C}^N$, respectively. In particular, all eigenvalues of $H^{(L)}$ are also eigenvalues of A, and, in addition, the method produces basis vectors for the A-invariant or A^T-invariant subspace found.

In the Lanczos process, the matrix A itself is never modified and appears only in the form of matrix–vector products $A \cdot v$ and $A^T \cdot w$. Because of this feature, the method is especially attractive for sparse matrix computations. Indeed, in practice, the Lanczos process is mostly applied to large sparse matrices A, either for computing eigenvalues of A or, in the form of the closely related biconjugate gradient (BCG) algorithm [21], for solving linear systems $Ax = b$. For large A, the finite termination property is of no practical importance and the Lanczos method is used as a purely iterative procedure. Typically, the spectrum of $H^{(n)}$ offers good approximations to some of the eigenvalues of A after already relatively few iterations, i.e., for $n \ll N$. Similarly, BCG, especially if used in conjunction with preconditioning, often converges in relatively few iterations to the solution of $Ax = b$.

Unfortunately, in the standard nonsymmetric Lanczos method a breakdown, more precisely, division by 0, may occur before an invariant subspace is found. In finite-precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns may occur, which lead to numerical instabilities in subsequent iterations. The

*Received by the editors April 8, 1991; accepted for publication (in revised form) January 8, 1992.

†Research Institute for Advanced Computer Science, Mail Stop T041–5, NASA Ames Research Center, Moffett Field, California 94035. The work of these authors was supported in part by the Defense Advanced Research Projects Agency via Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

‡Present Address, AT&T Bell Laboratories, Room 2C-420, 600 Mountain Ave., Murray Hill, New Jersey 07974–0636.

§Interdisciplinary Project Center for Supercomputing, ETH Zürich, ETH-Zentrum, CH–8092 Zürich, Switzerland.

possibility of breakdowns has brought the nonsymmetric Lanczos process into discredit and has certainly prevented many people from using the algorithm on non-Hermitian matrices. The symmetric Lanczos process for Hermitian matrices A is a special case of the general procedure in which the occurrence of breakdowns can be excluded.

On the other hand, it is possible to modify the Lanczos process so that it skips over those iterations in which an exact breakdown would occur in the standard method. The related modified recurrences for formally orthogonal polynomials were mentioned by Gragg [14, pp. 222–223] and by Draux [7]; also, in the context of the partial realization problem, by Kung [19, Chap. IV] and Gragg and Lindquist [15]. However, a complete treatment of the modified Lanczos method and its intimate connection with orthogonal polynomials and Padé approximation was presented only recently, by Gutknecht [16], [17]. Clearly, in finite-precision arithmetic, a viable modified Lanczos process also needs to skip over near-breakdowns. Taylor [27] and Parlett, Taylor, and Liu [25], with their look-ahead Lanczos algorithm, were the first to propose such a practical procedure. However, in [27] and [25], the details of an actual implementation are worked out only for look-ahead steps of length 2. We will use the term *look-ahead Lanczos method* in a broader sense to denote extensions of the standard Lanczos process which skip over breakdowns and near-breakdowns. Finally, note that, in addition to [16] and [17], there are several other recent papers dealing with various aspects of look-ahead Lanczos methods (see [1], [2], [4], [5], [9], [13], [18], and [23]).

The main purpose of this paper is to present a robust implementation of the look-ahead Lanczos method for general complex non-Hermitian matrices. Our intention is to develop an algorithm that can be used as a black box. In particular, the code can handle look-ahead steps of any length and is not restricted to steps of length 2. On many modern computer architectures, the computation of inner products of long vectors is a bottleneck. Therefore, one of our objectives is to minimize the number of inner products in our implementation of the look-ahead Lanczos method. The proposed algorithm requires the same number of inner products as the classical Lanczos process, as opposed to the look-ahead algorithm described in [27] and [25], which always requires additional inner products. In particular, our implementation differs from the one in [27] and [25] even for look-ahead steps of length 2.

The outline of the paper is as follows. In §2, we recall the standard nonsymmetric Lanczos method and its close relationship with orthogonal polynomials. Using this connection, we then describe the basic idea of the look-ahead versions of the Lanczos process. In §3, we present a sketch of our implementation of the algorithm with look-ahead and some of its basic properties. In §4, we discuss in more detail issues related to the look-ahead feature of the algorithm, while in §5 we are concerned with issues related to the implementation of the algorithm. Finally, in §6, we report a few numerical experiments with the algorithm, for both eigenvalue problems and linear systems, and in §7, we make some concluding remarks.

We remark that an extended version of the present paper is available as a technical report [11]. In particular, details omitted here can be found therein. Furthermore, the look-ahead Lanczos process can be used to compute approximate solutions to $Ax = b$, which are defined by a quasi-minimal residual (QMR) property. The resulting QMR algorithm is described in detail in [12] and [13].

For notation, we will adhere to the Householder conventions with only a few exceptions, which we will note. Throughout the paper, all vectors and matrices can be assumed to be complex. As usual, $M^T = [\mu_{ji}]$ and $M^H = [\overline{\mu}_{ji}]$ denote the transpose and the conjugate transpose, respectively, of the matrix $M = [\mu_{ij}]$. The largest and smallest singular values of M are denoted by $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$, respectively. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm and $\|M\| = \sigma_{\max}(M)$ denotes the corresponding matrix norm. The notation

$$K_n(c, B) := \text{span}\,\{c, Bc, \ldots, B^{n-1}c\}$$

is used for the nth Krylov subspace of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and the $N \times N$ matrix B.

$$\mathcal{P}_n := \{\Psi(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_n\lambda^n \mid \gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{C}\}$$

denotes the set of all complex polynomials of degree at most n. Furthermore, A is always assumed to be a possibly complex and in general non-Hermitian $N \times N$ matrix.

Finally, we note that in our formulation of the nonsymmetric Lanczos algorithm and its look-ahead variant, we use A^T rather than A^H. This was a deliberate choice in order to avoid complex conjugation of the scalars in the recurrences; the algorithms can be formulated equally well in either terms (cf. (2.18)).

2. Background.

In this section, we briefly recall the classical nonsymmetric Lanczos method [20] and its close relationship with *formally orthogonal polynomials* (FOPs). Using this connection, we then describe the basic idea of the look-ahead Lanczos algorithm.

Given two nonzero starting vectors $v_1 \in \mathbb{C}^N$ and $w_1 \in \mathbb{C}^N$, the standard nonsymmetric Lanczos method generates two sequences of vectors $\{v_n\}_{n=1}^L$ and $\{w_n\}_{n=1}^L$ such that, for $n = 1, \ldots, L$,

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

and

$$w_i^T v_j = \left\{ \begin{array}{ll} \delta_i \neq 0 & \text{if } i = j, \\ 0 & \text{otherwise}, \end{array} \right\} \quad \text{for all} \quad i, j = 1, \ldots, n. \tag{2.2}$$

The actual construction of the vectors v_n and w_n is based on the three-term recurrences

$$\begin{aligned} v_{n+1} &= Av_n - \alpha_n v_n - \beta_n v_{n-1}, \\ w_{n+1} &= A^T w_n - \alpha_n w_n - \beta_n w_{n-1}, \end{aligned} \tag{2.3}$$

where

$$\alpha_n = \frac{w_n^T A v_n}{\delta_n} \quad \text{and} \quad \beta_n = \frac{w_{n-1}^T A v_n}{\delta_{n-1}} = \frac{\delta_n}{\delta_{n-1}}$$

are chosen to enforce (2.2). For $n = 1$, we set $\beta_1 = 0$ and $v_0 = w_0 = 0$ in (2.3). Letting

$$V^{(n)} := [\, v_1\; v_2\; \cdots\; v_n \,] \quad \text{and} \quad W^{(n)} := [\, w_1\; w_2\; \cdots\; w_n \,] \tag{2.4}$$

denote the matrices whose columns are the first n of the vectors v_j and w_j, respectively, and letting

$$H^{(n)} := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ 1 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & 1 & \alpha_n \end{bmatrix}$$

denote the tridiagonal matrix containing the recurrence coefficients, we can rewrite (2.3) as

$$\begin{aligned} AV^{(n)} &= V^{(n)}H^{(n)} + [\,0 \quad \cdots \quad 0 \quad v_{n+1}\,], \\ A^TW^{(n)} &= W^{(n)}H^{(n)} + [\,0 \quad \cdots \quad 0 \quad w_{n+1}\,]. \end{aligned} \tag{2.5}$$

Moreover, the biorthogonality condition (2.2) reads as

$$(W^{(n)})^TV^{(n)} = D^{(n)} := \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n). \tag{2.6}$$

Let L be the largest integer such that there exist vectors v_n and w_n, $n = 1, \ldots, L$, satisfying (2.1) and (2.2). Note that $L \leq N$ and that, in view of (2.3), L is the smallest integer such that

$$w_{L+1}^Tv_{L+1} = 0. \tag{2.7}$$

Moreover, let

$$L_r = L_r(v_1, A) := \dim K_N(v_1, A) \quad \text{and} \quad L_l = L_l(w_1, A^T) := \dim K_N(w_1, A^T)$$

denote the *grade* of v_1 with respect to A and the *grade* of w_1 with respect to A^T, respectively (cf. [29, p. 37]). There are two essentially different cases for fulfilling the termination condition (2.7). The first case, referred to as *regular termination*, occurs when $v_{L+1} = 0$ or $w_{L+1} = 0$. If $v_{L+1} = 0$, then $L = L_r$ and the *right* Lanczos vectors $v_1, \ldots, v_{L_r}$ span the A-invariant subspace $K_{L_r}(v_1, A)$. Similarly, if $w_{L+1} = 0$, then $L = L_l$ and the *left* Lanczos vectors $w_1, \ldots, w_{L_l}$ span the A^T-invariant subspace $K_{L_l}(w_1, A^T)$. Unfortunately, it can also happen that the termination condition (2.7) is satisfied with $v_{L+1} \neq 0$ and $w_{L+1} \neq 0$. This second case is referred to as *serious breakdown* [29, p. 389]. Note that, in this case,

$$L < L_\star := \min\{L_l, L_r\},$$

and the Lanczos vectors span neither an A-invariant nor an A^T-invariant subspace of $\mathbb{C}^N$.

It is the possibility of serious breakdowns, or, in finite-precision arithmetic, of *near-breakdowns*, i.e.,

$$w_{n+1}^Tv_{n+1} \approx 0, \quad \text{but} \quad w_{n+1} \not\approx 0 \quad \text{and} \quad v_{n+1} \not\approx 0,$$

that has brought the classical nonsymmetric Lanczos algorithm into discredit. However, by means of a look-ahead procedure, it is possible to leap (except in the very special case of an incurable breakdown [27]) over those iterations in which the standard algorithm would break down. Next, using the intimate connection between the Lanczos process and FOPs, we describe the basic idea of the look-ahead Lanczos algorithm.

First, note that

$$\begin{aligned} K_n(v_1, A) &= \{\Psi(A)v_1 \mid \Psi \in \mathcal{P}_{n-1}\}, \\ K_n(w_1, A^T) &= \{\Psi(A^T)w_1 \mid \Psi \in \mathcal{P}_{n-1}\}. \end{aligned} \tag{2.8}$$

In particular, in view of (2.3), for $n = 1, \ldots, L$,

$$v_n = \Psi_{n-1}(A)v_1 \quad \text{and} \quad w_n = \Psi_{n-1}(A^T)w_1, \tag{2.9}$$

where $\Psi_{n-1} \in \mathcal{P}_{n-1}$ is a uniquely defined monic polynomial. Then, introducing the formal inner product

$$(\Phi, \Psi) := \left(\Phi(A^T)w_1\right)^T (\Psi(A)v_1) = w_1^T \Phi(A)\Psi(A)v_1, \tag{2.10}$$

and using (2.1), (2.8), and (2.9), we can rewrite the biorthogonality condition (2.2) in terms of polynomials:

$$(\Psi_{n-1}, \Psi) = 0 \quad \text{for all} \quad \Psi \in \mathcal{P}_{n-2} \tag{2.11}$$

and

$$(\Psi_{n-1}, \Psi_{n-1}) \neq 0. \tag{2.12}$$

Note that, except for the Hermitian case, i.e., $A = A^H$ and $w_1 = \overline{v}_1$, the formal inner product (2.10) is indefinite. Therefore, in the general case, there exist polynomials $\Psi \neq 0$ with "length" $(\Psi, \Psi) = 0$ or even $(\Psi, \Psi) < 0$.

A polynomial $\Psi_{n-1} \in \mathcal{P}_{n-1}$ of exact degree $n-1$ that fulfills (2.11) is called a FOP (with respect to the formal inner product (2.10)); we refer the reader to, e.g., [3], [7], and [16]. Note that the condition (2.11) is empty for $n = 1$, and hence any $\Psi_0 = \gamma_0 \neq 0$ is a FOP of degree 0. From (2.11),

$$\Psi_{n-1}(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_{n-1}\lambda^{n-1}, \quad \text{where} \quad \gamma_{n-1} \neq 0,$$

is a FOP of degree $n-1$ if and only if its coefficients $\gamma_0, \ldots, \gamma_{n-1}$, are a nontrivial solution of the linear system

$$\begin{bmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_{n-2} \\ \mu_1 & \iddots & & \iddots & \vdots \\ \mu_2 & & \iddots & & \vdots \\ \vdots & \iddots & & & \mu_{2n-5} \\ \mu_{n-2} & \cdots & \cdots & \mu_{2n-5} & \mu_{2n-4} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-2} \end{bmatrix} = -\gamma_{n-1} \begin{bmatrix} \mu_{n-1} \\ \mu_n \\ \mu_{n+1} \\ \vdots \\ \mu_{2n-3} \end{bmatrix}. \tag{2.13}$$

Here

$$\mu_j := w_1^T A^j v_1 = (1, \lambda^j), \qquad j = 0, 1, \ldots,$$

are the moments associated with (2.10). A FOP Ψ_{n-1} is called *regular* if it is uniquely determined by (2.11) up to a scalar, and it is said to be *singular* otherwise. We remark that FOPs of degree 0 are always regular. In particular, a regular FOP is unique if it is required to be monic. Moreover, singular FOPs occur if and only if the corresponding linear system (2.13) has a singular coefficient matrix, but is consistent for some $\gamma_{n-1} \neq 0$. If (2.13) is inconsistent when $\gamma_{n-1} \neq 0$, then no FOP Ψ_{n-1} exists. This case is referred to as *deficient*. By relaxing (2.11) slightly, one can define so-called *deficient FOPs* (see [16] for details). Simple examples (see, e.g., [12, §13]) show that the singular and deficient cases do indeed occur. Thus, regular FOPs need not exist for every degree $n-1$. We would like to stress that this phenomenon is due to the indefiniteness of (2.10). For a positive definite inner product $(\cdot, \cdot)$, unique monic formally orthogonal polynomials always exist, up to the degree equal to the grade of v_1 with respect to A. Such a definite inner product is induced in the Hermitian case $A = A^H$ and $w_1 = \overline{v_1}$. In this case, the FOPs are true orthogonal polynomials with respect to a positive weight whose support is a set of points on the real axis (see, e.g., [26]). In addition, they have real coefficients and therefore

$$(\Psi, \Psi) = w_1^T \Psi(A)\Psi(A)v_1 = v_1^H \overline{\Psi}(A^H)\Psi(A)v_1 = \|\Psi(A)v_1\|^2 .$$

Finally, given a regular FOP Ψ_{n-1}, it is easily checked whether a regular FOP of degree n exists. Indeed, using (2.13), one readily obtains the following lemma.

LEMMA 2.1. *Let* Ψ_{n-1} *be a regular* FOP (*with respect to the formal inner product* (2.10)) *of degree* $n-1$. *Then, a regular* FOP *of degree* n *exists if and only if* (2.12) *is satisfied.*

Let us return to the standard nonsymmetric Lanczos process (2.3). Using (2.7), (2.9), (2.10), and Lemma 2.1, we conclude that a serious breakdown occurs if and only if no regular FOP exists for some $L < L_\star$. In this case, the termination index L is the smallest integer L for which there exists no regular FOP of degree L.

On the other hand, there is a maximal subset of indices

$$\{n_1, n_2, \ldots, n_J\} \subseteq \{1, 2, \ldots, L_\star\}, \qquad n_1 := 1 < n_2 < \cdots < n_J \le L_\star, \tag{2.14}$$

such that, for each $j = 1, 2, \ldots, J$, there exists a monic regular FOP $\Psi_{n_j - 1} \in \mathcal{P}_{n_j - 1}$. Note that $n_1 = 1$ since $\Psi_0(\lambda) \equiv 1$ is a monic regular FOP of degree 0. It is well known [7], [15] that three successive regular FOPs $\Psi_{n_{j-1}-1}$, Ψ_{n_j-1}, and $\Psi_{n_{j+1}-1}$ are connected via a three-term recurrence. Consequently, setting, in analogy to (2.9),

$$v_{n_j} = \Psi_{n_j-1}(A)v_1 \quad \text{and} \quad w_{n_j} = \Psi_{n_j-1}(A^T)w_1,$$

we obtain two sequences of vectors $\{v_{n_j}\}_{j=1}^J$ and $\{w_{n_j}\}_{j=1}^J$ which can be computed by means of three-term recurrences. These vectors will be called *regular* vectors, since they correspond to regular FOPs. Note that the starting vectors v_1 and w_1 are always regular. The look-ahead Lanczos procedure is an extension of the classical nonsymmetric Lanczos algorithm; in exact arithmetic, it generates the vectors v_{n_j} and w_{n_j}, $j = 1, \ldots, J$. If $n_J = L_\star$ in (2.14), then these vectors can be complemented to a basis for an A-invariant or A^T-invariant subspace of $\mathbb{C}^N$. An incurable breakdown occurs if and only if $n_J < L_\star$ in (2.14). Finally, note that

$$w_{n_j}^T v = w^T v_{n_j} = 0 \quad \text{for all} \quad v \in K_{n_j-1}(v_1, A),\ w \in K_{n_j-1}(w_1, A^T), \quad j = 1, \ldots, J.$$

The look-ahead procedure we have sketched so far only skips over exact breakdowns. It yields what is called the *nongeneric* Lanczos algorithm in [16]. Of course, in finite-precision arithmetic, the look-ahead Lanczos algorithm also needs to leap over near-breakdowns. Roughly speaking, a robust implementation should attempt to generate only the "well-defined" regular vectors. In practice, then, one aims to generate two sequences of vectors $\{v_{n_{j_k}}\}_{k=1}^K$ and $\{w_{n_{j_k}}\}_{k=1}^K$ where

$$\{n_{j_k}\}_{k=1}^K \subseteq \{n_j\}_{j=1}^J, \qquad j_1 := 1, \tag{2.15}$$

is a suitable subset of (2.14). We set $j_1 = 1$, since v_1 and w_1 are always regular. The problem of how to determine the set (2.15) of indices of the "well-defined" regular vectors will be addressed in detail in §4.

In order to obtain complete bases for the subspaces $K_n(v_1, A)$ and $K_n(w_1, A^T)$, we need to add vectors

$$\begin{gathered} v_n \in K_n(v_1, A) \setminus K_{n-1}(v_1, A) \quad \text{and} \quad w_n \in K_n(w_1, A^T) \setminus K_{n-1}(w_1, A^T), \\ n = n_{j_{k-1}} + 1, \ldots, n_{j_k} - 1, \qquad k = 2, 3, \ldots, K, \end{gathered} \tag{2.16}$$

to the two sequences $\{v_{n_{j_k}}\}_{k=1}^{K}$ and $\{w_{n_{j_k}}\}_{k=1}^{K}$, respectively. Clearly, (2.16) guarantees that (2.1) remains valid for the look-ahead Lanczos algorithm. The vectors in (2.16) are called *inner* vectors. Moreover, for each k, the vectors v_n, $n = n_{j_k}, n_{j_k}+1, \ldots, n_{j_{k+1}}-1$, and correspondingly for w_n, are referred to as the kth *block*. The inner vectors of a block built because of an exact breakdown correspond to singular or deficient FOPs, while the inner vectors of a block built because of a near-breakdown correspond to polynomials that in general are combinations of regular, singular, and deficient FOPs. We will refer to both the regular and the inner vectors v_n and w_n generated by the look-ahead variant as right and left Lanczos vectors, in analogy to the terminology of the standard nonsymmetric Lanczos algorithm.

So far, we have not specified how to actually construct the inner vectors. The point is that the inner vectors can be chosen such that the v_n's and w_n's from blocks corresponding to different indices k are still biorthogonal to each other. More precisely, with $V^{(n)}$ and $W^{(n)}$ defined as in (2.4), we have, in analogy to (2.6),

$$(W^{(n)})^T V^{(n)} = D^{(n)}, \quad n = n_{j_l}-1, \quad l = 2, 3, \ldots, K. \tag{2.17}$$

Here, $D^{(n)}$ is now a nonsingular block diagonal matrix with $l-1$ blocks of respective size $(n_{j_{k+1}}-n_{j_k}) \times (n_{j_{k+1}}-n_{j_k})$, $k = 1, \ldots, l-1$. Similarly, (2.5) holds, for $n = n_{j_l}-1$, $l = 2, 3, \ldots, K$, where $H^{(n)}$ is now a block tridiagonal matrix with diagonal blocks of size $(n_{j_{k+1}}-n_{j_k}) \times (n_{j_{k+1}}-n_{j_k})$, $k = 1, \ldots, l-1$ (cf. (3.4)–(3.5)).

There are two fundamentally different approaches for constructing inner vectors with the property (2.17). In both cases, inner vectors are first generated using a simple three-term recurrence. However, in the first approach, each inner vector in a block is then biorthogonalized against the previous block as soon as it is constructed. This variant will be called the *sequential algorithm*. In the second approach, all the inner vectors in a block are first constructed using the three-term recurrence, and then the entire block is biorthogonalized against the previous block and possibly, depending on the size of the current block, against vectors from blocks further back. This variant will be called the *block algorithm*. The sequential algorithm is more suitable for a serial computer, while the block algorithm is more suitable for a parallel computer. In this paper, we describe only the sequential algorithm and its implementation. A sketch of the block algorithm can be found in [11]. Details of an actual implementation and numerical results will be presented elsewhere.

Finally, two more notes. First, the inner product (2.10) could have been defined as

$$(\Phi, \Psi) := \left(\overline{\Phi}(A^H)\overline{w_1}\right)^H (\Psi(A)v_1) = \overline{w_1}^H \Phi(A)\Psi(A)v_1, \tag{2.18}$$

and the algorithm could be formulated equally well in either terms. Second, in the rest of the paper, we will use the notation $n_k := n_{j_k}$ for the indices of the "well-defined" regular vectors. However, notice that there is no guarantee that the indices n_k generated by the look-ahead Lanczos algorithm in finite-precision arithmetic actually satisfy (2.15).

3. The sequential algorithm.

In this section, we start the discussion of the sequential Lanczos algorithm with look-ahead. We present a sketch of the algorithm and its basic properties, then discuss some aspects related to its practical implementation in the next two sections.

First, we introduce some notation. As in the last section, $n = 1, 2, \ldots$, denote the indices of the Lanczos vectors v_n and w_n. The index $k = 1, 2, \ldots$, is used as a counter for the blocks built by the look-ahead algorithm. Moreover, we always use $l = l(n)$ to denote the index of the block which contains the Lanczos vectors v_n and w_n. Recall that by n_k we denote the indices of the computed regular vectors, which are always the first vectors in each block k. Thus, n_l is the index of the last computed regular vector with index $\leq n$. We have $n_1 = 1$. Capital letters with subscript k denote the matrices containing quantities from block k. For example,

$$V_k := \begin{bmatrix} v_{n_k} & v_{n_k+1} & \cdots & v_{n_{k+1}-1} \end{bmatrix}$$

is the matrix whose columns are the Lanczos vectors from a completed block k. Capital letters with superscripts $^{(n)}$ denote matrices containing quantities from steps 1 through n, as in (2.4). With this notation, the matrix form of the sequential algorithm with look-ahead is similar to (2.5)–(2.6)

$$\begin{aligned} AV^{(n)} &= V^{(n)}H^{(n)} + \begin{bmatrix} 0 & \cdots & 0 & v_{n+1} \end{bmatrix}, \\ A^TW^{(n)} &= W^{(n)}H^{(n)} + \begin{bmatrix} 0 & \cdots & 0 & w_{n+1} \end{bmatrix}, \end{aligned} \tag{3.1}$$

and

$$(W^{(n)})^T V^{(n)} = D^{(n)}. \tag{3.2}$$

Here

$$D^{(n)} = \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_l), \quad \delta_k := W_k^T V_k, \quad k = 1, 2, \ldots, l = l(n), \tag{3.3}$$

is block diagonal, and the blocks $\delta_1, \delta_2, \ldots, \delta_{l-1}$ are nonsingular. If $n = n_{l+1} - 1$, then the lth block, δ_l, in (3.3) is also nonsingular and it is called *complete*. In particular, if δ_l is complete, then $D^{(n)}$ itself is nonsingular, and (3.3) reduces to (2.17). In this case, the next regular vectors $v_{n_{l+1}}$ and $w_{n_{l+1}}$ can be computed and start a new block.

In (3.1),

$$H^{(n)} = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_l \\ 0 & \cdots & 0 & \gamma_l & \alpha_l \end{bmatrix} \tag{3.4}$$

is an $n \times n$ block tridiagonal upper Hessenberg matrix with blocks of the form

$$\alpha_k = \begin{bmatrix} * & \cdots & \cdots & \cdots & * \\ 1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & * \end{bmatrix}, \qquad \gamma_k = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 1 \\ \vdots & \ddots & & & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}, \tag{3.5}$$

while the β_k's are in general full matrices. Note that here we violate the Householder conventions by using small Greek letters to denote quantities which may be matrices. The justification is that in general the algorithm takes regular steps, and hence these quantities are usually scalars. Let $h_k := n_{k+1} - n_k$, $k = 1, 2, \ldots$, be the size of the kth block. For $k < l = l(n)$ the matrices α_k, β_k, and γ_k are of size $h_k \times h_k$, $h_{k-1} \times h_k$, and $h_k \times h_{k-1}$, respectively. In general, however, the lth block need not be complete. Hence, the matrices α_l, β_l, and γ_l corresponding to the current (lth) block are of size $\tilde{h}_l \times \tilde{h}_l$, $h_{l-1} \times \tilde{h}_l$, and $\tilde{h}_l \times h_{l-1}$, respectively, where $\tilde{h}_l := n + 1 - n_l$.

We will assume that the inner vectors in a block are generated using a three-term recursion of the form

$$v_{n+1} = Av_n - \zeta_n v_n - \eta_n v_{n-1}, \quad w_{n+1} = A^T w_n - \zeta_n w_n - \eta_n w_{n-1}, \tag{3.6}$$

where ζ_n and η_n are recursion coefficients and $\eta_{n_k} = 0$, $k = 1, 2, \ldots$. One may choose these coefficients so that they remain the same from one block to the next and change only with respect to their index inside the block, $n - n_k$, or one may choose these coefficients so that they change from one block to the next. For instance, one practical choice for the polynomials in (3.6) are suitably scaled and translated Chebyshev polynomials, so that the inner vectors are generated by the Chebyshev iteration [22]. In this case, the translation parameters could be adjusted using spectral information obtained from previous Lanczos steps. We do not necessarily advocate the use of fancy recursions in (3.6). From our experience, the algorithm we propose builds very small blocks, typically of size 2 or 3. Except for p-cyclic matrices (cf. Example 6.3 in §6) or contrived examples, the largest block we observed in test runs with "real-life" matrices was of size 4. It occurred for a matrix arising in oil-reservoir simulations where out of 1500 steps, the algorithm built 2×2 blocks 49 times, 3×3 blocks 7 times, and one 4×4 block (see [13, Ex. 2]). Hence, the recursion in (3.6) is not overly important, and in our experiments, we have used the recursion coefficients $\zeta_n = 1$ and, if $n \neq n_k$, $\eta_n = 1$. On the other hand, for the block version of the algorithm, where larger blocks are built, more attention needs to be paid to the recursion used. As indicated, details of the block algorithm will be presented elsewhere. Finally, one could consider orthogonalizing (in the Euclidean sense) the right, respectively, left, Lanczos vectors within each block. However, for the blocks we have seen built, such an orthogonalization process did not lead to better numerical properties of the algorithm. Therefore, in view of the additional inner products that need to be computed, orthogonalizing within each block is not justified.

In practice, for reasons of stability, one computes scaled versions of the right and left Lanczos vectors, rather than the "monic" vectors v_n and w_n corresponding to monic FOPs. A proven choice (see [25] and [27]) is to scale the Lanczos vectors to have unit length. We denote by $\hat{v}_n$ and $\hat{w}_n$ the scaled versions defined by

$$\hat{v}_n := v_n / \|v_n\| \quad \text{and} \quad \hat{w}_n := w_n / \|w_n\|,$$

and more generally, we will denote by hat (ˆ) quantities containing or depending on the scaled vectors. For example, setting

$$\hat{H}^{(n)} := \operatorname{diag}(\|v_1\|, \|v_2\|, \ldots, \|v_n\|) H^{(n)} \operatorname{diag}(1/\|v_1\|, 1/\|v_2\|, \ldots, 1/\|v_n\|),$$

$$\hat{H}_e^{(n)} := \begin{bmatrix} \hat{H}^{(n)} \\ \rho_{n+1} e_n^T \end{bmatrix}, \quad \rho_{n+1} := \frac{\|v_{n+1}\|}{\|v_n\|}, \quad e_n := [0 \quad \cdots \quad 0 \quad 1]^T \in \mathbb{R}^n,$$

we can rewrite the first relation in (3.1) in terms of scaled vectors as follows:

$$A\hat{V}^{(n)} = \hat{V}^{(n+1)} \hat{H}_e^{(n)}. \tag{3.7}$$

With this note, we now present a sketch of the sequential Lanczos algorithm with look-ahead.

ALGORITHM 3.1 (SEQUENTIAL LANCZOS ALGORITHM WITH LOOK-AHEAD).

```
(0) Choose $\hat{v}_1$, $\hat{w}_1 \in \mathbb{C}^N$ with $\|\hat{v}_1\| = \|\hat{w}_1\| = 1$;
    Set $\hat{V}_1 = \hat{v}_1$, $\hat{W}_1 = \hat{w}_1$, $\hat{\delta}_1 = \hat{W}_1^T \hat{V}_1$;
    Set $n_1 = 1$, $l = 1$, $\hat{v}_0 = \hat{w}_0 = 0$, $\hat{V}_0 = \hat{W}_0 = \emptyset$, $\rho_1 = \xi_1 = 1$;
For $n = 1, 2, \ldots$ :
(1) Decide whether to construct $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ as regular or inner vectors
    and go to (2) or (3), respectively;
(2) (Regular step.) Compute
```

$$\begin{aligned} \tilde{v}_{n+1} &= A\hat{v}_n - \hat{V}_l\, \hat{\delta}_l^{-1} \hat{W}_l^T A\hat{v}_n - \hat{V}_{l-1}\hat{\delta}_{l-1}^{-1}\hat{W}_{l-1}^T A\hat{v}_n, \\ \tilde{w}_{n+1} &= A^T\hat{w}_n - \hat{W}_l\, \hat{\delta}_l^{-T} \hat{V}_l^T A^T\hat{w}_n - \hat{W}_{l-1}\hat{\delta}_{l-1}^{-T}\hat{V}_{l-1}^T A^T\hat{w}_n, \end{aligned} \tag{3.8}$$

```
    set $n_{l+1} = n + 1$, $l = l + 1$, $\hat{V}_l = \hat{W}_l = \emptyset$, and go to (4);
(3) (Inner step.) Compute
```

$$\begin{aligned} \tilde{v}_{n+1} &= A\hat{v}_n - \zeta_n\hat{v}_n - (\eta_n/\rho_n)\,\hat{v}_{n-1} - \hat{V}_{l-1}\hat{\delta}_{l-1}^{-1}\hat{W}_{l-1}^T A\hat{v}_n, \\ \tilde{w}_{n+1} &= A^T\hat{w}_n - \zeta_n\hat{w}_n - (\eta_n/\xi_n)\,\hat{w}_{n-1} - \hat{W}_{l-1}\hat{\delta}_{l-1}^{-T}\hat{V}_{l-1}^T A^T\hat{w}_n; \end{aligned} \tag{3.9}$$

```
(4) Compute $\rho_{n+1} = \|\tilde{v}_{n+1}\|$ and $\xi_{n+1} = \|\tilde{w}_{n+1}\|$;
    If $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, stop;
    Otherwise, set
```

$$\begin{aligned} \hat{v}_{n+1} &= \tilde{v}_{n+1}/\rho_{n+1}, \quad \hat{w}_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \\ \hat{V}_l &= [\,\hat{V}_l \quad \hat{v}_{n+1}\,], \quad \hat{W}_l = [\,\hat{W}_l \quad \hat{w}_{n+1}\,], \quad \hat{\delta}_l = \hat{W}_l^T\hat{V}_l; \end{aligned} \tag{3.10}$$

```
(5) Add the nonzero elements of the $n$th column to $\hat{H}^{(n)}$
    and set $(\hat{H}_e^{(n)})_{n+1,n} = \rho_{n+1}$.
```

Note that, if $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ are inner vectors, the size of the current incomplete block l is increased by 1; if they are regular vectors, then the lth block is complete and a new block, the $(l+1)$st, is started, with $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ as its first vectors. Finally, we remark that, in view of (3.7), the nonzero elements of the nth column of $\hat{H}^{(n)}$ occur as coefficients in the first recursion of (3.8), respectively, (3.9).

4. Building blocks.

In this section, we discuss the criteria used in step (1) of Algorithm 3.1 to decide whether a pair of Lanczos vectors $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ is built as inner vectors or as regular vectors. We propose three criteria, namely, (4.3), (4.4), and (4.5) below. If all three checks (4.3)–(4.5) are satisfied, then $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ are constructed as regular vectors; otherwise, they are constructed as inner vectors. Let us motivate these three criteria.

First, recall (cf. (3.3)) that for $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ to be built as regular vectors it is necessary that $\hat{\delta}_l$ be nonsingular. Therefore, it is tempting to base the decision "regular versus inner step" solely on checking whether $\hat{\delta}_l$ is close to singular, and to perform a regular step if and only if

$$\sigma_{\min}(\hat{\delta}_{l(n)}) \geq \text{tol}, \tag{4.1}$$

for some suitably chosen tolerance tol. For example, Parlett [23] suggests tol $= \epsilon^{1/4}$ or tol $= \epsilon^{1/3}$, where ϵ denotes the roundoff unit. Then (4.1) would guarantee that complete blocks of computed Lanczos vectors satisfy

$$\sigma_{\min}(\hat{\delta}_k) \geq \text{tol}, \qquad k = 1, 2, \ldots.$$

This, together with (3.3), would imply by [23, Thm. 10.1] that

$$\sigma_{\min}(\hat{V}^{(n)}) \geq \frac{\text{tol}}{\sqrt{n}} \quad \text{and} \quad \sigma_{\min}(\hat{W}^{(n)}) \geq \frac{\text{tol}}{\sqrt{n}}, \quad n = n_k - 1, \quad k = 1, 2, \ldots. \tag{4.2}$$

Since the columns of $\hat{V}^{(n)}$ and $\hat{W}^{(n)}$ are unit vectors, $\sigma_{\min}(\hat{V}^{(n)})$ and $\sigma_{\min}(\hat{W}^{(n)})$ are a measure of the linear independence of these vectors; in particular, (4.2) would ensure that the Lanczos vectors remain linearly independent. However, in the outlined algorithm, the block orthogonality (3.2)–(3.3) is enforced only among two or three successive blocks, and in finite-precision arithmetic, biorthogonality of blocks whose indices are far apart is typically lost. The theorem assumes that (3.2)–(3.3) hold for all indices, and without this, the theorem fails in finite arithmetic. We illustrate this with a simple example.

Example 4.1. In Fig. 4.1, we plot $\sigma_{\min}(\hat{\delta}_{l(n)})$ (dots), $\min_{1\leq k<l(n)} (\sigma_{\min}(\hat{\delta}_k))$ (solid line), and $\sqrt{n}\,\sigma_{\min}(\hat{V}^{(n)})$ (dotted line), as functions of the iteration index $n = 1, 2, \ldots$, for a random 50×50 dense matrix. The theorem predicts that

$$\sqrt{n}\,\sigma_{\min}(\hat{V}^{(n)}) \geq \min_{1\leq k<l(n)} (\sigma_{\min}(\hat{\delta}_k)),$$

which is clearly not the case.

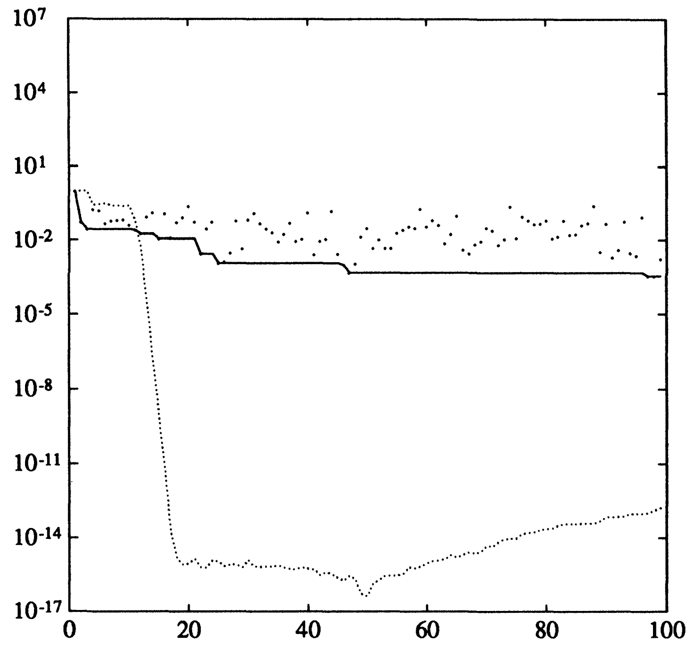

FIG. 4.1. $\sigma_{\min}(\hat{\delta}_{l(n)})$ (*dots*), $\min_{1\leq k<l(n)}(\sigma_{\min}(\hat{\delta}_k))$ (*solid line*), *and* $\sqrt{n}\,\sigma_{\min}$ $(\hat{V}^{(n)})$ (*dotted line*), *plotted versus the iteration index* n.

As this simple example shows, the check (4.1) alone does not ensure that the computed Lanczos vectors are sufficiently linearly independent. In particular, if the look-ahead strategy is based only on criterion (4.1), the algorithm may produce within a block Lanczos vectors that are almost linearly dependent. When this happens, the check (4.1) usually fails in all subsequent iterations and thus the algorithm never completes the current block, i.e., it has generated an *artificial* incurable breakdown.

In addition, numerical experience indicates another problem with (4.1): for values of tol that are "reasonably" larger than machine epsilon, the behavior of the algorithm is very sensitive with respect to the actual value of tol. We also illustrate this with an example.

Example 4.2. We applied the Lanczos algorithm to a nonsymmetric matrix A obtained from discretizing a three-dimensional partial differential equation (cf. Example 6.5 in §6). This example was run on a machine with $\epsilon \approx$ 1.3E-29. In the first case, we set tol $= \epsilon^{1/4} \approx$ 6.0E-08, while in the second case, we set tol $= \epsilon^{1/3} \approx$ 2.3E-10. In Fig. 4.2, we plot $\sigma_{\min}(\hat{\delta}_{l(n)})$ versus the iteration index n for the two runs, the dotted line for $\epsilon^{1/4}$ and the solid line for $\epsilon^{1/3}$. In the first case, the algorithm starts building a block that it never closes, and the singular values clearly become smaller and smaller. Yet if tol is only slightly smaller, as in the second case, the algorithm runs to completion, in this case solving the linear system to the desired accuracy, and thus indicating that the block built in the first case was not a true, but an artificial incurable breakdown.

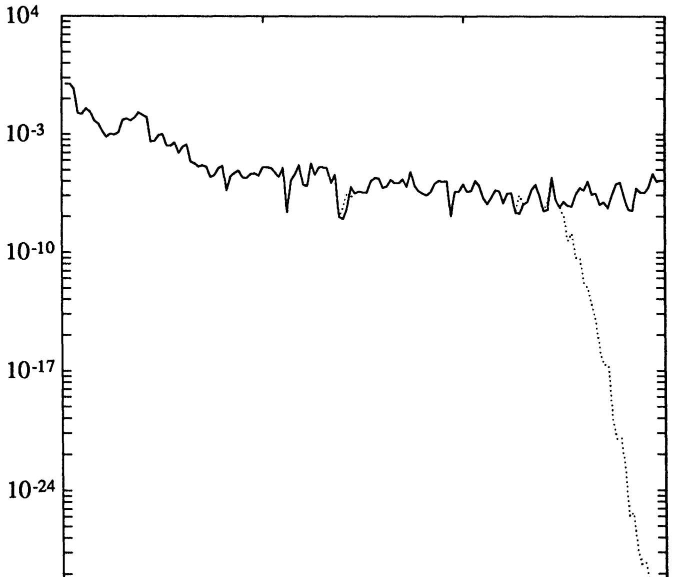

FIG. 4.2. $\epsilon^{1/4}$ *(dotted line) and* $\epsilon^{1/3}$ *(solid line), plotted versus the iteration index* n.

We note that the sensitivity of look-ahead procedures to the choice of tolerances, such as tol in (4.1), was also observed in [5]. However, no remedy for this phenomenon is given in [5]. Furthermore, we remark that the problem of generating almost linearly dependent vectors is not specific to the Lanczos biorthogonalization process. Indeed, similar effects can also occur in true orthogonalization methods (cf. [28]).

Examples 4.1 and 4.2 clearly show that the decision "regular versus inner step" cannot be based on (4.1) alone. Instead, we propose to relax the check (4.1), so that it merely ensures that $\hat{\delta}_{l(n)}$ is numerically nonsingular, and to add the checks (4.4)–(4.5) below, which guarantee that the computed Lanczos vectors remain sufficiently linearly independent. Hence, instead of (4.1), we check for

$$\sigma_{\min}(\hat{\delta}_{l(n)}) \geq \epsilon, \tag{4.3}$$

where ϵ denotes the roundoff unit.

Our numerical experiments have shown that typically the algorithm starts to generate Lanczos vectors which are almost linearly dependent, once a regular vector $\hat{v}_{n+1}$ was computed whose component $A\hat{v}_n \in K_{n+1}(v_1, A)$ is dominated by its component in the previous Krylov space $K_n(v_1, A)$ (and similarly for $\hat{w}_{n+1}$). In order to avoid the construction of such regular vectors, we check the l_1-norm of the coefficients for $\hat{V}_{l-1}$ and $\hat{V}_l$ in (3.8); $\hat{v}_{n+1}$ can be computed as a regular vector only if

$$\sum_{j=n_{l-1}}^{n_l-1} \left|(\hat{\delta}_{l-1}^{-1}\hat{W}_{l-1}^T A\hat{v}_n)_j\right| \leq n(A) \quad \text{and} \quad \sum_{j=n_l}^{n} \left|(\hat{\delta}_l^{-1}\hat{W}_l^T A\hat{v}_n)_j\right| \leq n(A). \tag{4.4}$$

Here $n(A)$ is a factor depending on the norm of A; we will indicate later how this factor is computed. Similarly, we check the l_1-norm of the coefficients for $\hat{W}_{l-1}$ and $\hat{W}_l$ in (3.8); $\hat{w}_{n+1}$ can be computed as a regular vector only if

$$\sum_{j=n_{l-1}}^{n_l-1} \left|(\hat{\delta}_{l-1}^{-T}\hat{V}_{l-1}^T A^T\hat{w}_n)_j\right| \leq n(A) \quad \text{and} \quad \sum_{j=n_l}^{n} \left|(\hat{\delta}_l^{-T}\hat{V}_l^T A^T\hat{w}_n)_j\right| \leq n(A). \tag{4.5}$$

The pair $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ is built as regular vectors only if the checks (4.3)–(4.5) hold true.

We need to indicate how $n(A)$ is chosen in (4.4)–(4.5). Numerical experience with matrices whose norm is known indicates that setting $n(A) = \|A\|$ is too strict and can result in artificial incurable breakdowns. A better setting seems to be $n(A) = 10 \cdot \|A\|$, but even this is dependent on the matrix. In any case, in practice one does not know $\|A\|$, and there is also the issue of a maximal block size, determined by limits on available storage. To solve the problems of estimating the norms and a suitable factor $n(A)$, as well as cope with limited storage and yet allow the algorithm to proceed as far as possible, we propose the following procedure. Suppose we are given an initial value for $n(A)$, based either on an estimate from the user (for example, $n(A)$ from a previous run with the matrix A), or by setting

$$n(A) = \max\left\{\|A\hat{v}_1\|, \|A^T\hat{w}_1\|\right\}.$$

Note that here A denotes the matrix actually used in generating the Lanczos vectors, thus including the case when we are solving a preconditioned linear system. We then update $n(A)$ dynamically, as follows. In each block, whenever an inner vector is built because one of the checks (4.4)–(4.5) is not satisfied, the algorithm keeps track of the size of the terms that have caused one or more of (4.4)–(4.5) to be false. If the block closes naturally, then this information is not needed. If, however, the algorithm is about to run out of storage, then $n(A)$ is replaced with the smallest value which has caused an inner vector to be built. The updated value of $n(A)$ is guaranteed to pass the checks (4.4)–(4.5) at least once, and hence the block is guaranteed to close. This also frees up the storage that was used by the previous block, thus ensuring that the algorithm can proceed.

5. Implementation details.

We now turn to a few implementation details. In particular, we wish to show how one can implement the sequential algorithm with the same number of inner products per step as the classical Lanczos algorithm. For a regular step, one needs to compute $\hat{\delta}_l$, $\hat{W}_l^T A\hat{v}_n$, and $\hat{W}_{l-1}^T A\hat{v}_n$ in (3.8). For an inner step, one needs to compute $\hat{W}_{l-1}^T A\hat{v}_n$ in (3.9) and to update $\hat{\delta}_l$ in (3.10). We will show that for a block of size h_l, only $2h_l$ inner products are required: $2h_l - 1$ will be required to compute $\hat{\delta}_l$, and one inner product will be required to compute $\hat{W}_l^T A\hat{v}_n$. We will obtain $\hat{W}_{l-1}^T A\hat{v}_n$ without performing any inner products. To simplify the derivations, we will use the "monic" vectors v_n and w_n. All quantities involving the scaled vectors $\hat{v}_n$ and $\hat{w}_n$ can be obtained from the corresponding quantities involving v_n and w_n simply by scaling. Finally, we remark that, using a similar argument as in (5.1) below, one easily verifies that

$$W_l^T A v_n = V_l^T A^T w_n \quad \text{and} \quad W_{l-1}^T A v_n = V_{l-1}^T A^T w_n.$$

Therefore, the coefficients $\hat{\delta}_l^{-T} \hat{V}_l^T A^T \hat{w}_n$ and $\hat{\delta}_{l-1}^{-T} \hat{V}_{l-1}^T A^T \hat{w}_n$, which occur in the recursions for the left Lanczos vectors in (3.8) or (3.9), can be generated from $\hat{\delta}_l^{-1} \hat{W}_l^T A\hat{v}_n$ and $\hat{\delta}_{l-1}^{-1} \hat{W}_{l-1}^T A\hat{v}_n$, without computing any additional inner products.

First consider δ_l. Using (2.9) and the fact that polynomials in A commute, we deduce that

$$w_i^T v_j = w_1^T \Psi_i(A)\Psi_j(A)v_1 = w_1^T \Psi_j(A)\Psi_i(A)v_1 = w_j^T v_i. \tag{5.1}$$

This shows that the matrix δ_l is symmetric, and hence we only need to compute its upper triangle.

We will now show that once the diagonal and first superdiagonal of δ_l have been computed by inner products, the remaining upper triangle can be computed by recurrences. Let w_i and v_j be two vectors from the current block. Using (3.6) and the fact that the inner vectors from block l are orthogonal to the vectors from the previous block, we have

$$\begin{aligned} w_i^T v_j &= w_i^T (A v_{j-1} - \zeta_{j-1} v_{j-1} - \eta_{j-1} v_{j-2}) \\ &= (A^T w_i)^T v_{j-1} - \zeta_{j-1} w_i^T v_{j-1} - \eta_{j-1} w_i^T v_{j-2} \\ &= (w_{i+1} + \zeta_i w_i + \eta_i w_{i-1})^T v_{j-1} - \zeta_{j-1} w_i^T v_{j-1} - \eta_{j-1} w_i^T v_{j-2} \\ &= w_{i+1}^T v_{j-1} + \zeta_i w_i^T v_{j-1} + \eta_i w_{i-1}^T v_{j-1} - \zeta_{j-1} w_i^T v_{j-1} - \eta_{j-1} w_i^T v_{j-2}. \end{aligned}$$

Thus, $w_i^T v_j$ depends only on elements of δ_l from the previous two columns, and hence, with the exception of the diagonal and the first superdiagonal, can be computed without any additional inner products. Note that the recurrences and the orthogonality used in the above derivation are enforced numerically, and so computing $w_i^T v_j$ by the above recurrence should give the same results—up to roundoff—as computing the inner product directly.

We will now show how to compute $W_l^T A v_n$ with only one additional inner product, while $W_{l-1}^T A v_n$ can be obtained with no additional inner products. Consider $w_i^T A v_n$, for w_i a vector from either the current or the previous block. Then, we have

$$\begin{aligned} w_i^T A v_n &= (A^T w_i)^T v_n = (w_{i+1} + \zeta_i w_i + \eta_i w_{i-1})^T v_n \\ &= w_{i+1}^T v_n + \zeta_i w_i^T v_n + \eta_i w_{i-1}^T v_n. \end{aligned}$$

For $i < n_l - 1$, $W_{l-1}^T v_n = 0$, and hence $w_i^T A v_n = 0$. For $i = n_l - 1$, the above reduces to $w_{n_l-1}^T A v_n = w_{n_l}^T v_n$, which is computed as part of the first row of δ_l. For $n_l \le i < n_{l+1}$, all of the terms needed are available from δ_l. Finally, for the last vector in the current block, $i = n_{l+1} - 1$, we do not have $w_{n_{l+1}}^T v_n$, and hence have to compute it directly, thus requiring another inner product.

6. Numerical examples.

We have performed extensive numerical experiments with our implementation of the look-ahead Lanczos algorithm, both for eigenvalue problems and for the solution of linear systems. In this section, we present a few typical results of these experiments. Further numerical results are reported in [12] and [13].

Approximations to the eigenvalues of A can be obtained from the look-ahead Lanczos algorithm by computing some or all of the eigenvalues of the Lanczos matrix $\hat{H}^{(n)}$, the so-called *Ritz values*. In general, *spurious* approximate eigenvalues, caused by a loss of orthogonality among the Lanczos vectors, can occur among the Ritz values. This phenomenon is not due to the nonsymmetry of the matrix A; indeed, it also appears in the symmetric Lanczos process. We have used the heuristic due to Cullum and Willoughby [6] to identify and eliminate spurious Ritz values. Although this procedure was originally proposed for the scalar tridiagonal matrices generated by the standard Lanczos process, we also found it to work satisfactorily for the block tridiagonal matrices $\hat{H}^{(n)}$ produced by the look-ahead Lanczos algorithm. The eigenvalues of $\hat{H}^{(n)}$ were always computed using standard EISPACK routines.

For the solution of nonsingular linear systems

$$Ax = b, \tag{6.1}$$

we combine the look-ahead Lanczos algorithm with the QMR approach. More precisely, let $x_0 \in \mathbb{C}^N$ be any initial guess for (6.1) and choose the normalized starting residual vector

$$\hat{v}_1 := r_0/\rho_0, \qquad r_0 := b - Ax_0, \qquad \rho_0 := \|r_0\|,$$

as the first right Lanczos vector in Algorithm 3.1. The QMR method then generates approximate solutions to (6.1) defined by

$$x_n = x_0 + \hat{V}^{(n)} z_n, \qquad n = 1, 2, \ldots, \tag{6.2}$$

where z_n is the solution of the least-squares problem

$$\min_{z \in \mathbb{C}^n} \left\| \rho_0 e_1 - \hat{H}_e^{(n)} z \right\|, \qquad e_1 := [1 \quad 0 \quad \cdots \quad 0]^T \in \mathbb{R}^{(n+1)}. \tag{6.3}$$

We remark that, using (3.7), one easily verifies that the residual vector corresponding to the iterate (6.2) satisfies

$$r_n := b - Ax_n = \hat{V}^{(n+1)} \left(\rho_0 e_1 - \hat{H}_e^{(n)} z_n \right). \tag{6.4}$$

Thus the choice (6.3) of z_n just guarantees that the Euclidean norm of the coefficient vector in the representation (6.4) is minimal. For details and further properties of the QMR method, we refer to [13].

Example 6.1. This example is an eigenvalue problem, taken from [6]. Consider the differential operator

$$\begin{aligned} Lu := &- \frac{\partial}{\partial x}\left(e^{-xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right) \\ &+ 20(x+y)\frac{\partial u}{\partial x} + 20\frac{\partial}{\partial x}\left((x+y)u\right) + \frac{1}{1+x+y}u \end{aligned} \tag{6.5}$$

on the unit square $(0,1) \times (0,1)$. We discretize (6.5) using centered differences on a 29×29 grid with mesh size $h = 1/30$. This leads to a nonsymmetric matrix of order $N = 900$. Unit vectors with random entries were used as starting vectors $\hat{v}_1$, $\hat{w}_1$ for Algorithm 3.1. The look-ahead Lanczos process was run for 320 steps, during which it built seven blocks of size 2. In Fig. 6.1, we plot the Ritz values (marked by "o") generated by the look-ahead Lanczos process after n = 40, 80, 160, 320 steps. We note that after 40 steps, the complex conjugate pair of Ritz values with maximal real part had converged to eigenvalues of A. After 80 steps, 12 Ritz values (all on the right edge of the spectrum) had converged, while after 160 steps the 30 Ritz values (24 on the right edge and 6 on the left edge of the spectrum) had converged to eigenvalues of A. In many cases, the standard and the look-ahead Lanczos procedures give similar results. In particular, for this example, the results obtained from both the standard and the look-ahead Lanczos algorithm match those reported in [6].

Example 6.2. This example is an eigenvalue problem, taken from [24], whose exact eigenvalues are known. Generally, problems of this type arise in modeling concentration waves in reaction and transport interaction of chemical solutions in a tubular reactor. The particular test problem used here corresponds to the so-called Brusselator wave model. This example was run for a matrix A of size $N = 100$. Again, unit vectors with random entries were used as starting vectors $\hat{v}_1$, $\hat{w}_1$. The look-ahead Lanczos algorithm needs $n = 112$ steps to obtain all the eigenvalues of A; it built two blocks of size 2. For this example, we have also run the standard Lanczos process without look-ahead, and computed the Ritz values after n = 100, 112, 120 steps. The denominators $\hat{w}_n^T\hat{w}_n$ were checked to exceed $\sqrt{\epsilon}$ in magnitude. In all three cases, some of the Ritz values obtained from the standard Lanczos process after deleting spurious eigenvalues do not correspond to any of the eigenvalues of A. In particular, the standard Lanczos process does not obtain the smallest eigenvalues of A even after 120 steps, and generates incorrect Ritz values, as shown in the plot. In Fig. 6.2, we plot the Ritz values generated by the look-ahead Lanczos process (marked by "o") and the Ritz values generated by the standard Lanczos process (marked by "+"), both after 120 steps.

Example 6.3. Here we consider a 6-cyclic matrix

$$A = \begin{bmatrix} I_1 & 0 & 0 & 0 & 0 & B_1 \\ B_2 & I_2 & 0 & 0 & 0 & 0 \\ 0 & B_3 & I_3 & 0 & 0 & 0 \\ 0 & 0 & B_4 & I_4 & 0 & 0 \\ 0 & 0 & 0 & B_5 & I_5 & 0 \\ 0 & 0 & 0 & 0 & B_6 & I_6 \end{bmatrix}, \tag{6.6}$$

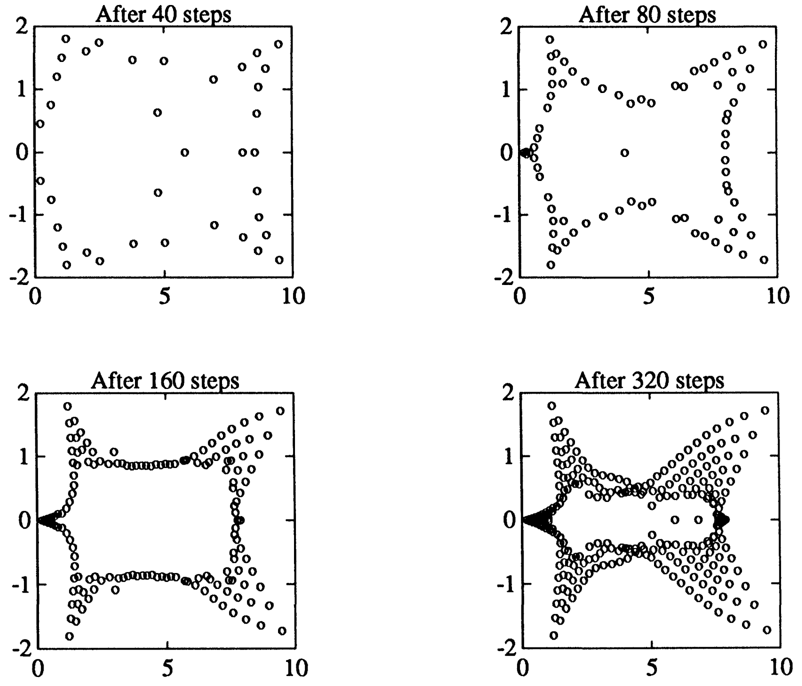

FIG. 6.1. *Ritz values for Example* 6.1, *obtained after* $n = 40, 80, 160, 320$ *steps of the look-ahead Lanczos algorithm.*

where the diagonal blocks I_1, I_2, I_3, I_4, I_5, and I_6 are identity matrices of size 827, 844, 827, 838, 831, and 838, respectively, so that A is a matrix of order $N = 5005$. This matrix arises in Markov chain modeling. For general p-cyclic matrices A of the form (6.6), Freund, Golub, and Hochbruck [10] have shown that work and storage of the look-ahead Lanczos process can be reduced to approximately $1/p$, as compared to arbitrary starting vectors, if v_1 and w_1 have only one nonzero block conforming to the block structure of A. Here, we have chosen

$$\hat{v}_1 = \begin{bmatrix} f_1 \\ 0 \end{bmatrix}, \quad \hat{w}_1 = \begin{bmatrix} g_1 \\ 0 \end{bmatrix},$$

where f_1, $g_1 \in \mathbb{R}^{827}$ have random entries. The look-ahead Lanczos algorithm generates blocks that alternately have sizes 1 and 5, starting with a block of size 1. In Fig. 6.3, we plot the Ritz values (marked by "o") generated by the look-ahead Lanczos process after $n = 40$, 80, 160, 320 steps. The standard Lanczos algorithm without look-ahead generates one Ritz value 1 in the first step, and then breaks down in the second step. Clearly, this example shows that the use of look-ahead is crucial if one wants to exploit the special structure of p-cyclic matrices.

Example 6.4. Here we solve a linear system (6.1) where A is the SHERMAN5 matrix taken from the Harwell–Boeing test collection of sparse matrices [8]. The matrix is of dimension $N = 3312$ and has 20793 nonzero elements. The right-hand side b in (6.1), as well as the first left Lanczos vector $\hat{w}_1$ were generated as different

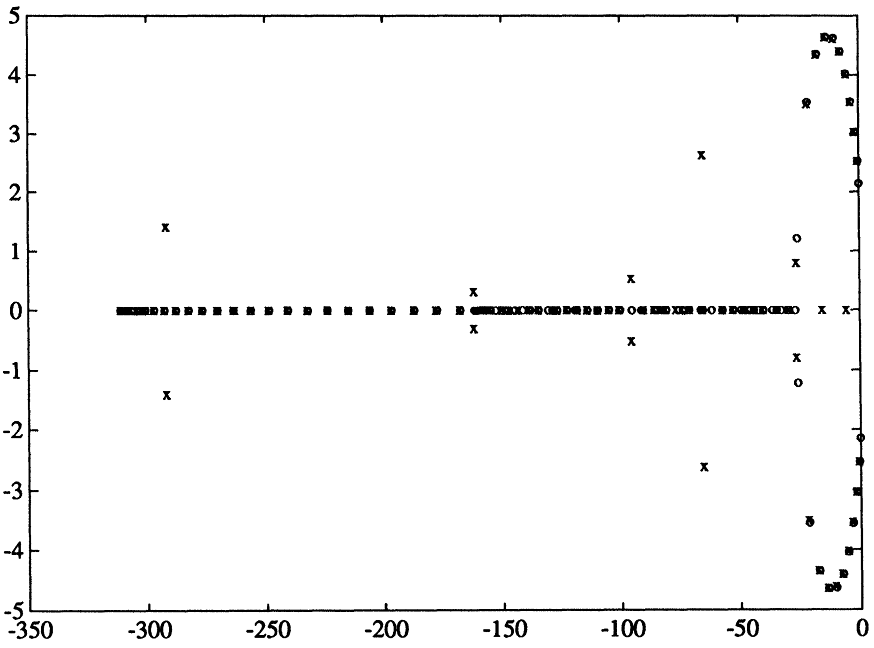

FIG. 6.2. *Ritz values (marked by "o," respectively "+") for Example 6.2, obtained from the look-ahead, respectively standard, Lanczos algorithm.*

unit vectors with random entries, and we set $x_0 = 0$ and $\hat{v}_1 = r_0 = b$. The QMR method takes $n = 1652$ steps to reduce the norm of the initial residual by a factor of 10^{-6}; see Fig. 6.4, where the residual norm $\|r_n\|$ is plotted versus n (solid line). The underlying look-ahead Lanczos algorithm built 34 blocks of size 2 and 7 blocks of size 3. We would like to stress that, for this example, look-ahead is crucial. Indeed, if look-ahead is turned off, then QMR based on the standard Lanczos algorithm does not converge. The corresponding stagnating residual norms (dashed line) are also depicted in Fig. 6.4.

Example 6.5. Here we consider the partial differential equation

$$Lu = f \quad \text{on} \quad (0,1) \times (0,1) \times (0,1), \tag{6.7}$$

where

$$Lu = -\frac{\partial}{\partial x}\left(e^{xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(e^{xy}\frac{\partial u}{\partial z}\right) + \beta(x+y+z)\frac{\partial u}{\partial x} + \left(\gamma + \frac{1}{1+x+y+z}\right)u,$$

with Dirichlet boundary conditions $u = 0$. The right-hand side f is chosen such that

$$u = (1-x)(1-y)(1-z)\left(1-e^{-x}\right)\left(1-e^{-y}\right)\left(1-e^{-z}\right)$$

is the exact solution of (6.7). We set the parameters in (6.7) to $\beta = 30$ and $\gamma = -250$, and then we discretize (6.7) using centered differences on a uniform $15 \times 15 \times 15$

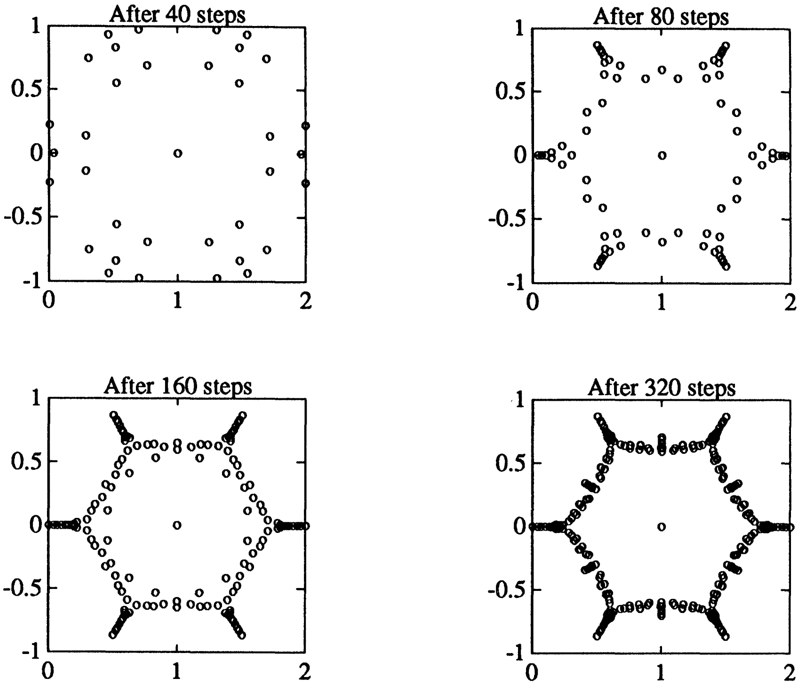

FIG. 6.3. *Ritz values for Example* 6.3, *obtained after* $n = 40, 80, 160, 320$ *steps of the look-ahead Lanczos algorithm.*

grid with mesh size $h = 1/16$. This leads to a linear system (6.1) with coefficient matrix A of order $N = 3375$ and 22275 nonzero elements. The QMR iteration was started with $x_0 = 0$. For the first pair of Lanczos vectors, we have chosen $\hat{w}_1 = \hat{v}_1 = b/\|b\|$ in Algorithm 3.1. The QMR approach takes $n = 149$ steps to reduce the norm of the initial residual by a factor of 10^{-6}; see Fig. 6.5, where the relative norm $\|r_n\| \,/\, \|r_0\|$ is plotted versus n (solid line). For this run, the underlying look-ahead Lanczos algorithm built three blocks of size 2. Next, we note that the matrix A is just the one used in Example 4.2. Recall that the look-ahead Lanczos algorithm based on the check (4.1) with tolerance tol$= \epsilon^{1/4} \approx$ 6.0E-08 encountered an artificial incurable breakdown. We also ran QMR based on this version of the look-ahead Lanczos algorithm, and the resulting convergence curve is shown as the dotted line in Fig. 6.5. Notice that, due to the artificial incurable breakdown, QMR does not converge in this case (cf. Fig. 4.2). Finally, we remark that QMR based on the standard Lanczos algorithm without look-ahead also converges for this example and gives a curve similar to the solid line in Fig. 6.5.

7. Conclusion.

We have proposed an implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. Our implementation can handle look-ahead steps of any length. Also, the proposed algorithm requires the same number of inner products as the standard Lanczos process without look-ahead. It was our intention to develop a robust algorithm which can be used in a black box.

FORTRAN 77 codes of our implementation of the look-ahead Lanczos algorithm

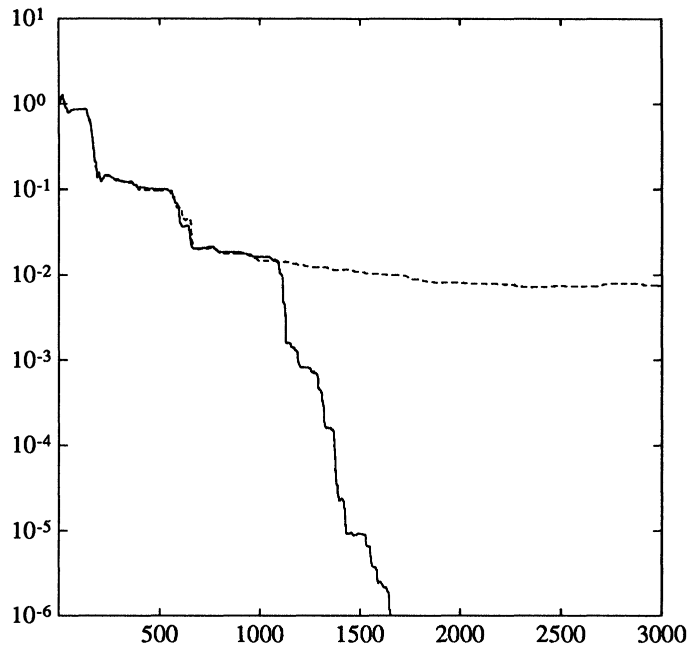

FIG. 6.4. *Residual norm* $\|r_n\|$ *plotted versus* n *for Example* 6.4.

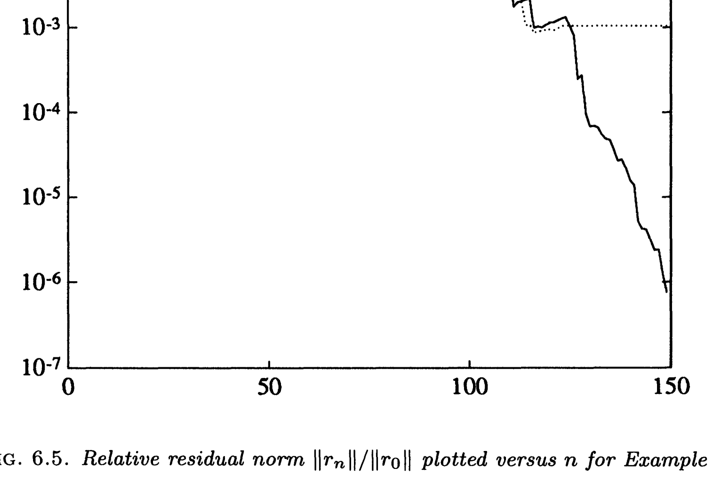

FIG. 6.5. *Relative residual norm* $\|r_n\|/\|r_0\|$ *plotted versus* n *for Example* 6.5.

and the QMR method are available electronically from the authors (na.freund@na-net.ornl.gov or na.nachtigal@na-net.ornl.gov).

Acknowledgments. Part of this work was done while the first and third authors were visiting the Interdisciplinary Project Center for Supercomputing at ETH Zürich, and we thank Martin Gutknecht for his warm hospitality. The authors are grateful to Marlis Hochbruck for her help with Example 6.3 and her careful reading of the manuscript.

REFERENCES

[1] 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.

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

[3] C. Brezinski, *Padé-Type Approximation and General Orthogonal Polynomials*, Birkhäuser, Basel, 1980.

[4] C. Brezinski, M. Redivo Zaglia, and H. Sadok, *A breakdown-free Lanczos type algorithm for solving linear systems*, Tech. Rep. ANO-239, Université des Sciences et Techniques de Lille Flandres-Artois, France, Jan. 1991.

[5] ———, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Algorithms, 1 (1991), pp. 261–284.

[6] 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.

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

[8] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[9] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[10] R. W. Freund, G. H. Golub, and M. Hochbruck, *Krylov subspace methods for non-Hermitian p-cyclic matrices*, Tech. Rep., RIACS, NASA Ames Research Center, Moffett Field, CA, in preparation.

[11] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part* I, Tech. Rep. 90.45, RIACS, NASA Ames Research Center, Moffett Field, CA, November 1990.

[12] R. W. Freund and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non- Hermitian matrices, Part* II, Tech. Rep. 90.46, RIACS, NASA Ames Research Center, Moffett Field, CA, November 1990.

[13] ———, QMR: *A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991) pp. 315–339.

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

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

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

[17] ———, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part* II, IPS Research Report No. 90–16, Zürich, Switzerland, September 1990.

[18] W. Joubert, *Lanczos methods for the solution of nonsymmetric systems of linear equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 926–943.

[19] S.-Y. Kung, *Multivariable and multidimensional systems: analysis and design*, Ph.D. thesis, Dept. of Electrical Engineering, Stanford University, Stanford, June 1977.

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

[21] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[22] T. A. MANTEUFFEL, *The Tchebychev iteration for nonsymmetric linear system*, Numer. Math. 28 (1977), pp. 307–327.

[23] B. N. PARLETT, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[24] B. N. PARLETT AND Y. SAAD, *Complex shift and invert strategies for real matrices*, Linear Algebra Appl., 88/89 (1987), pp. 575–593.

[25] 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.

[26] E. L. STIEFEL, *Kernel polynomials in linear algebra and their numerical applications*, U.S. National Bureau of Standards, Applied Mathematics Series, 49 (1958), pp. 1–22.

[27] D. R. TAYLOR, *Analysis of the look ahead Lanczos algorithm*, Ph.D. thesis, Dept. of Mathematics, University of California, Berkeley, November 1982.

[28] H. F. WALKER, *Implementation of the* GMRES *method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.

[29] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.