HOW FAR SHOULD YOU GO WITH THE LANCZOS PROCESS?

by

W. Kahan and B.N. Parlett

Memorandum No. UCB/ERL M78/48

18 July 1978

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720
Abstract

The Lanczos algorithm can be used to approximate both the largest and smallest eigenvalues of a symmetric matrix whose order is so large that similarity transformations are not feasible. The algorithm builds up a tridiagonal matrix row by row and the key question is when to stop. An analysis leads to a stopping criterion which is inspired by a useful error bound on the computed eigenvalues.

1. INTRODUCTION

The Lanczos algorithm came back into prominence about five years ago [7,8] as the most promising way to compute a few of the eigenvectors of very large symmetric matrices. To be specific we think of computing the p smallest (or largest) eigenvalues of the n×n symmetric matrix A together with the associated eigenvectors. Typical values are p=3, n=1000.

The algorithm must be provided with a starting vector q₁ and then it builds up, one column per step, two auxiliary matrices. After j steps it will have produced j×j symmetric tridiagonal matrix Tₖ and an n×(j+1) matrix Qₖ₊₁ = (q₁,q₂,...,qₖ₊₁). Let θ₁,...,θₚ denote the p extreme eigenvalues of Tₖ.

What we want is that the {θₖ} should be good approximations to the wanted eigenvalues of A (α₁,...,αₚ, say) and we ask the following questions. Will the θₖ inevitably improve as j increases? How much or rather how little work is needed to compute an a posteriori bound on the errors in the θₖ?

We still do not know how best to use the Lanczos process. The surprising fact is that the θₖ are sometimes correct to 3 or 4 decimal figures even when j is as small as √n. What makes the method interesting is that such good fortune cannot be guaranteed. It depends on q₁, on the spread of the αᵢ, and on the precision of the arithmetic operations.

†The authors are pleased to acknowledge partial support from Office of Naval Research Contract N00014-69-A-0200-1017.
It is a pleasure to acknowledge the excellent pioneering work of Paige in his doctoral thesis [8] and the studies by Golub [2] and others [1] on the block version of the method. Sparsity of A plays a simple but crucial role here. It permits the computation of Av, for any n-vector v, in only wn basic operations, where w is the average number of nonzero elements per row. Of more importance sparsity discourages the use of explicit similarity transformations which invariably destroy sparsity even when they preserve bandwidth. The only knowledge of A which the Lanczos algorithm demands is how A acts on selected vectors, an attractive property for sparse problems.

To get round some of the difficulties posed by the gaps in our understanding it has been proposed that the Lanczos algorithm be used iteratively as follows. Start with \( q_1^{(1)} \) and run the algorithm for \( j = j^{(1)} \) steps, compute the p best eigenvector approximations available from \( T_j^{(1)} \) and \( Q_j^{(1)} \) and take a weighted combination of them as a new starting vector \( q_1^{(2)} \), run Lanczos for \( j^{(2)} \) steps, compute a new \( q_1^{(3)} \), and so on. The process can be continued until the \( \{q_j\} \) computed at the end of each run converge to the desired number of figures.

Is this a good idea? How should the \( j^{(1)} \) be chosen? We do not have definitive answers to the questions raised here but we do present a computable a posteriori error bound, an estimate for loss of orthogonality among the columns of \( Q_j \), a tentative stopping criterion, and a useful way of analyzing the algorithm.

Standard Householder matrix conventions will be followed except that \( M^* \) denotes the conjugate transpose of M and \( I \) denotes the identity matrix.

2. ERROR BOUNDS FOR THE EXACT LANCZOS ALGORITHM

When executed in exact arithmetic the matrix

\[ Q_j \equiv (q_1, \ldots, q_j), \]

which is generated column by column, is orthonormal:

\[ Q_j^* Q_j = I, \]

and is part of an orthogonal matrix Q which reduces A to tridiagonal form. The algorithm, or rather its \( j \)-th step, is completely specified by the single matrix relation
where \( e^*_j = (0, \ldots, 0, 1) \) has \( j \) elements and \( \| r_j \|^2 = r_j^* r_j = \beta_j^2 \).

The residual vector

\[
\begin{align*}
r_j &= \beta_j e_{j+1} = (A Q_j - Q_j T_j) e_j,
\end{align*}
\]

is always uniquely determined by \( A, Q_j, \) and \( T_j \). An essential characteristic of the Lanczos process is that the residual matrix \( AQ_j - Q_j T_j \) has all its substance concentrated in its final column. An attractive feature of the process, not obvious from (2), is that only the two most recent columns of \( Q_k \) need be kept in fast memory at the \( k \)-th step, so \( q_{k-2} \) can be put out to secondary storage. Sometimes the early \( q \) vectors are discarded but this makes it very difficult to estimate the accuracy of our approximations. Equation (2) is a nice compact way of remembering the algorithm.

Suppose now that the Lanczos process is halted at the end of the \( j \)-th step. The approximations to the desired eigenvectors and values of \( A \) are made as follows. The \( p \) extreme eigenvalues of \( T_j \) are computed along with their normalized eigenvectors. We suppress the dependence on \( j \) and write

\[
\begin{align*}
T_j c_i &= c_i \theta_i, \quad i = 1, \ldots, p, \quad \text{i.e. } T_j C = \Theta C
\end{align*}
\]

where

\[
C = (c_1, \ldots, c_p), \quad \Theta = \text{diag}(\theta_1, \ldots, \theta_p).
\]

Then we compute the Ritz vectors \( v_i = Q_j c_i, \quad i = 1, \ldots, p \). When \( Q_j \) is orthonormal these Ritz vectors are the best approximate eigenvectors, in the sense of residuals, that can be made from linear combinations of the columns of \( Q_j \) and the \( \theta_i \) are the best approximations to the corresponding eigenvalues. There is a considerable body of knowledge concerning the accuracy of the exact Ritz approximations. The Kaniel-Paige theory [3] gives a priori error bounds on \( \theta_i \) and \( v_i \) which tell how accuracy increases with \( j \). Moreover, there are refined a posteriori error bounds which can be computed when the algorithm is terminated. However, even with exact calculations there is no inexpensive, feasible criterion which can tell us the best \( j \) at which to stop.
Let us look at some simple a posteriori bounds. They employ the spectral matrix norm \( \|B\| = \max \|Bv\|/\|v\|, v \neq 0 \).

**THEOREM 1** (Kahan, 1967). Let \( H \) be any \( p \times p \) symmetric matrix with eigenvalues \( \theta_i \) and let \( S \) be \( n \times p \) and orthonormal. Then there are \( p \) eigenvalues \( \alpha_i \) of \( A \) such that, for \( i = 1, \ldots, p \),

\[
|\theta_i - \alpha_i| < \|AS-SH\|.
\]

We do not know which of \( A \)'s eigenvalues are the \( \alpha_i \).

The proof is based on the Weyl/Wielandt monotonicity theorem and is given in [3].

We are not interested in all of \( T_j \)'s eigenvalues, only \( p \) of them. In fact we have

\[
T_jC = 0.
\]

Now apply Theorem 1 with \( S = Q_jC \) to obtain the following result.

**COROLLARY.** Let \( AQ_jQ_j^T = r_je_j^T \) and \( T_jC = 0 \) where \( Q_j \) and \( C \) are orthonormal and \( \|r_j\| = \beta_j \). Then there are \( p \) eigenvalues \( \alpha_i \) of \( A \) such that for \( i = 1, \ldots, p \),

\[
|\theta_i - \alpha_i| < \beta_j\|e_j^TC\|.
\]

Thus when the last elements of some normalized eigenvectors of \( T_j \) are small good accuracy is obtained for their eigenvalues even when \( \beta_j \) is not small. Unfortunately we have no easy way of guaranteeing which eigenvalues of \( A \) are being approximated. This is an intrinsic limitation of the Lanczos method and will not be discussed in detail here.

The best way we know of testing which of \( A \)'s eigenvalues \( \{\alpha_k\} \) are being approximated by the \( \theta_i \) \( (i = 1, \ldots, p) \) is to perform a triangular factorization of \( A-\theta_iI \) into \( LDL^T \) and count the number \( v_i \) of negative elements of \( D \). By Sylvester's Inertia theorem there are \( v_i \) of the \( \alpha_i \)'s less than \( \theta_i \).

3. **PRACTICAL ERROR BOUNDS AND TERMINATION CRITERIA**

From now on we let \( Q_j, T_j, \) etc. stand for the quantities stored in the computer. Because of roundoff error \( Q_j \) will not be orthonormal and the residual matrix will not have all its substance in the last column. Fortunately Theorem 1 can
be generalized.

**THEOREM 2 (Kahan, 1967).** Let \( H \) be any \( p \times p \) matrix with eigenvalues \( \theta_1 \), let \( S \) be any \( n \times p \) matrix of full rank \( p \); then there are \( p \) eigenvalues \( \alpha_{1}^{*} \) of \( A \) such that, for \( i = 1, \ldots, p \),

\[
|\theta_{i} - \alpha_{i}^{*}| \leq \sqrt{2} \| \text{S} - \text{SH} \| \| (S*S)^{-1/2} \|.
\]

Note that \( \| (S*S)^{-1/2} \| \) is the reciprocal of \( S \)'s smallest singular value \( \sigma_{1}(S) \equiv \sqrt{\lambda_{1}(S*S)} \). A proof may be found in [3]. The factor \( \sqrt{2} \) is believed to be superfluous.

In order to use Theorem 2 we observe that \( Q_{j} \) and \( T_{j} \) actually satisfy

\[
AQ_{j} - Q_{j}T_{j} = F_{j} + r_{j}e_{j}^{*}
\]

where \( F_{j} \) accounts for roundoff error.

**COROLLARY.** Let \( T_{j}C = C0 \); then there are \( p \) eigenvalues \( \alpha_{1}^{*} \) of \( A \) such that, for \( i = 1, \ldots, p \),

\[
|\theta_{i} - \alpha_{i}^{*}| \leq \sqrt{2}(\|F_{j}\| \|C\| + \beta_{j} \|e_{j}^{*}C\|)/\sigma_{1}(Q_{j}C).
\]

**PROOF.** Apply Theorem 2 with \( S = Q_{j}C \).

In order to use the corollary an upper bound is needed on \( \|F_{j}\| \) and a lower bound on the singular values of \( Q_{j}C \). With the best of current techniques \( C \) will be very close to orthonormal. In the sense of quadratic forms,

\[
0 \leq C^{*}C \leq 1
\]

and hence, by the Cauchy interlace inequalities

\[\sigma_{1}(Q_{j}) \leq \sigma_{1}(Q_{j}C) \]

Error analyses in [5] and [8] show that \( \|F_{j}\| \) is small, like roundoff in \( \|A\| \), and so the term \( \|F_{j}\| \|C\| \) is completely dominated by \( \beta_{j} \|e_{j}^{*}C\| \) for all realistic values of \( j \). Thus in practice the bound in Corollary 1 is degraded by the factor \( 1/\sigma_{1} \).

Corollary 2 assures us that it is worthwhile to continue the Lanczos algorithm even after orthogonality among the \( q_{1} \) has been lost provided that they are still linearly independent. Experience suggests that good approximation to
internal eigenvalues can be obtained by continuing the algorithm indefinitely. However after \( \sigma_1(Q_j) = 0 \), i.e. when linear independence is lost, the user is faced with the identification problem, namely to say which of the \( \Theta_i \) do not approximate any of the \( \alpha_i \)'s and so are spurious. This problem can often be solved in particular applications but it is very troublesome to devise a procedure which will make this identification for the general case.

In order to escape this difficulty we would like to stop immediately \( \sigma_1(Q_j) < 1/2 \). However it is out of the question to compute this number at each step, or even update it. Instead we develop a computable lower bound on \( \sigma_1(Q_j) \) and stop a Lanczos run as soon as it vanishes. If the approximations are not satisfactory we can form a new starting vector from the Ritz vectors and start another run.

Since \( C \) is not computed at each step there is no simple way of bounding \( \sigma_1(Q_j^C) \) rather than \( \sigma_1(Q_j) \).

4. MONITORING LOSS OF ORTHOGONALITY

Because of finite precision arithmetic \( Q_j \) will not be orthonormal. Let us write

\[
\|1 - Q_j^*Q_j\| \leq \kappa_j
\]

and find some specific expressions for \( \kappa_j \). Note that

\[
\|Q_j\|^2 = \|Q_j^*Q_j\| = \|1 - (1 - Q_j^*Q_j)\| \leq 1 + \kappa_j,
\]

and, in the sense of quadratic forms

\[
1 - \kappa_j \leq Q_j^*Q_j \leq 1 + \kappa_j.
\]

Hence, while \( \kappa_j < 1 \),

\[
\sqrt{1 - \kappa_j} \leq \sqrt{\lambda_1(Q_j^*Q_j)} = \sigma_1(Q_j) \leq \sigma_1(Q_j^C).
\]

Our problem is thus reduced to finding a computable bound \( \kappa_j \). This can be accomplished in a variety of ways. Here is one of the simplest. Observe that

\[
1 - Q_j^*Q_{j+1} = \begin{bmatrix}
1 - Q_j^*Q_j & -Q_j^*q_{j+1}^t \\
-q_{j+1}^*Q_j & 1 - \|q_{j+1}\|^2
\end{bmatrix}.
\]

Let

\[
\|Q_j^*q_{j+1}\| \leq \zeta_j.
\]

Then, by considering the quadratic forms we see that
Here we have used the fact that, by definition,
\[ \|I - Q_j^{+}Q_{j+1}\| = |1 - q_1^2| \leq \kappa_1 \]
but, actually, \( \kappa_1 \) is the bound on the error in normalizing any vector to the given precision, including \( q_{j+1} \), and is taken as known.

Any computable bound \( \zeta_j \) will yield a corresponding definition of \( \kappa_j \), namely
\[ \kappa_{j+1} \equiv \left\| \begin{pmatrix} \kappa_j & \zeta_j \\ \zeta_j & \kappa_1 \end{pmatrix} \right\| = \frac{1}{2} \left( \kappa_j + \kappa_1 + \sqrt{(\kappa_j - \kappa_1)^2 + 4\zeta_j^2} \right). \]

Since \( \kappa_1 \) is known we focus our attention on \( \zeta_j \) and \( q_j^+q_{j+1} \).

5. AN EXPRESSION FOR \( Q_j^+r_j \)

We write
\[ T_j = \begin{bmatrix} \alpha_1 & \beta_1 & \circ & \circ \\ \beta_1 & \alpha_2 & \beta_2 & \circ \\
\circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \beta_{j-1} \end{bmatrix} \]
and let \( F_j = (f_1, \ldots, f_j) \). Our bound \( \zeta_j \) comes from a useful expression for \( Q_j^+r_j \).

**LEMMA 1.** Let \( Q_j \) and \( T_j \) satisfy (H): \( AQ_j - Q_jT_j = F_j + r_je_j^\perp \).
Then
\[ Q_j^+r_j = [(1-Q_j^+Q_j)T_j - (1-e_j^\perp e_j^\perp)T_j(1-Q_j^+C_j)]e_j + F_j^+q_j + (q_j^+Aq_j - \alpha_j)e_j - Q_j^+f_j. \]

**PROOF.**

\[ Q_j^+r_j = Q_j^+(AQ_j - Q_jT_j - F_j)e_j, \text{ using (H),} \]
\[ = [(AQ_j^+)Q_j - Q_j^+Q_jT_j]e_j - Q_j^+f_j, \text{ using } A^* = A, \]
SPARSE MATRIX COMPUTATIONS

\[ = [T^*_j Q^*_j + F^*_j Q^*_j + e^*_j Q^*_j - C^*_j Q^*_j T^*_j] e^*_j - Q^*_j f_j, \]
\[ = [-T^*_j (1 - Q^*_j) + (1 - Q^*_j) T^*_j] e^*_j + F^*_j q_j + e^*_j r^*_j Q^*_j e_j - Q^*_j f_j. \]

From the first line above

\[ e^*_j Q^*_j f_j = q^*_j A q_j - e^*_j Q^*_j T^*_j e_j = q^*_j A q_j - \alpha_j + e^*_j (1 - Q^*_j) T^*_j e_j. \]

On transposing and rearranging the desired expression is obtained. \( \square \)

In practice

\[ q_{j+1} = r_j / \beta_j + g_j, \]

where \( g_j \) accounts for roundoff error in the division by \( \beta_j \) and is always insignificant. In any case

\[ Q^*_j q_{j+1} = Q^*_j r_j / \beta_j + Q^*_j g_j. \]

We observe that \( \|Q^*_j q_{j+1}\| \) will not be small, like roundoff in \( 1 \) whenever, because of past errors, \( \|Q^*_j r_j\| / \|Q^*_j g_j\| \). In exact arithmetic \( Q^*_j r_j = 0 \) and this property fails in practice to the extent that there is cancellation in forming \( r_j \); the resulting large relative error in the small vector \( r_j \) becomes significant when \( r_j \) is divided by \( \beta_j \) to produce \( q_{j+1} \). As Paige points out in [7] this happening must be seen in perspective. We want cancellation to occur in the formation of \( r_j \) because cancellation is the harbinger of convergence of the \( \xi_j \) to an \( \alpha \).

6. A COMPUTABLE BOUND ON \( Q^*_j q_{j+1} \)

The expression for \( Q^*_j r_j \) given in Lemma 1 is best split into two parts

\[ b_j \equiv [(1 - Q^*_j Q^*_j) T^*_j - (1 - e^*_j e^*_j) T^*_j (1 - Q^*_j Q^*_j)] e^*_j, \]
\[ d_j \equiv F^*_j q_j + (Q^*_j Q^*_j - \alpha_j) e^*_j - Q^*_j f_j. \]

It turns out that our bounds on \( \|b_j\| \) overwhelm those on \( \|d_j\| \) as soon as \( j > 4 \). To bound \( \|d_j\| \) a detailed error analysis of the Lanczos algorithm is needed. There is little incentive to present this because the resulting bound makes an insignificant contribution to \( \xi_j \). We shall simply quote the results in [5] which are quite similar to those in [7].

The results are stated in terms of
SPARSE MATRIX COMPUTATIONS

\[ \|A\|_E \equiv \left( \sum_{\mu,\nu} |a_{\mu\nu}|^2 \right)^{1/2} = \sqrt{\text{trace}(A^*A)} \]

which arises in the general bound on the error in computing \( Av \). However \( \|A\|_E \) is just a convenient bound on the more realistic but less accessible quantity \( \|\tilde{A}\| \) where \( \tilde{A} = (|a_{1k}|) \). For large sparse matrices \( \|\tilde{A}\| \) is usually much smaller than \( \|A\|_E \).

The following estimates are crude but adequate for our purposes: for \( i \leq j \),

\[ \|f_i\| + \beta_i \|g_i\| < \kappa_1 \|A\|_E, \]

\[ |q_i \tilde{A} q_i - \alpha_i| < \kappa_1 \|A\|_E, \]

\[ \|Q_i\|^2 < 1 + \kappa_i, \]

\[ \|P_j\|_E < \sqrt{j} \max \|f_i\|, \ i < j, \]

\[ \kappa_1 < 2(n+6)\epsilon, \]

where \( \epsilon \) is the precision of the arithmetic facilities.

Finally we quote

**Lemma 2.** \[ \|d_j\| \leq (\sqrt{j} + 3 + \kappa_j) \|A\|_E. \]

Turning to \( \|b_j\| \) we find

**Lemma 3.** If \( \|Q_j^t q_{j+1}\| \leq \kappa_1, \ i < j, \) then

\[ \|b_j\|^2 \leq \left( \|T_{j-1} - \alpha_j\| \|\tilde{q}_{j-1}\| + \beta_{j-1} (\kappa_j + 2\kappa_1) \right)^2 + (3j+1)\kappa_1 \|A\|_E^2 \]

**Proof.** Partition the terms in \( b_j \) and observe

\[ (1-Q_j^t q_j)T_j e_j = \begin{bmatrix} -Q_j^t q_j \\ 1-\|q_j\|^2 \end{bmatrix} \alpha_j + \begin{bmatrix} -Q_j^t q_j \\ 1-\|q_j\|^2 \end{bmatrix} \beta_{j-1}, \]

\[ T_j (1-Q_j^t q_j) e_j = \begin{bmatrix} -T_j^{-1} Q_j^t q_j + e_j \beta_{j-1} (1-\|q_j\|^2) \\ \alpha_j (1-\|q_j\|^2) - \beta_{j-1} q_j \end{bmatrix}. \]
SPARSE MATRIX COMPUTATIONS

The factor \((1-e_i e_j^*)\) simply annihilates the bottom element. Recall that \(|1-\|q_i\|^2| < \kappa_1\), by definition of \(\kappa_1\). Hence the top part of \(b_j\) is

\[
(T_{j-1}-\alpha_j)Q_j^*q_j - \beta_j-1\left[\begin{array}{c}
Q_j^*q_j - 2q_{j-1}^*q_{j-1} \\
The factor (1-e_i e_j^*) simply annihilates the bottom element. Recall that \(|1-\|q_i\|^2| < \kappa_1\), by definition of \(\kappa_1\). Hence the top part of \(b_j\) is

\[
(T_{j-1}-\alpha_j)Q_j^*q_j - \beta_j-1\left[\begin{array}{c}
Q_j^*q_j - 2q_{j-1}^*q_{j-1} \\
0
\end{array}\right] + \beta_j-1\|q_j\|^2 - \|q_{j-1}\|^2
\]

and the bottom element is

\[
(1-\|q_j\|^2)\alpha_j - \beta_j-1 q_j^* q_j-1.
\]

By definition of the basic Lanczos steps

\[
q_{j+1} = Aq_j - \alpha_j q_j - \beta_j-1 q_{j-1} - f_j + \beta_j g_j
\]

then, subtracting and adding \(\alpha_j\),

\[
|q_j^* q_{j+1}^* \beta_j| \leq |q_j^* q_{j+1} - \alpha_j| + |1-\|q_j\|^2| \cdot |\alpha_j| + |\beta_j-1 q_j^* q_{j-1}|
\]

Now suppose that \(|q_j^* q_{j+1}^* \beta_j| < \psi_j\), then the error bounds quoted above Lemma 2 yield

\[
\psi_j \leq 3\kappa_1 \|A\| \psi_j-1,
\]

Straightforward application of the triangle inequality gives the desired bound. □

Adding these results together we define

\[
\omega_j = (\|T_{j-1}-\alpha_j\|_{\text{op}} \tau_{j-1} + \beta_j-1(\tau_{j-2} + 2\kappa_1)) + (3j+1)^2 \kappa_1 \|A\|_{\text{op}} + (3j+3+\kappa_j) \|A\|_{\text{op}},
\]

Then, by induction,

\[
\text{Lemma 4. If } \tau_{-1} = \tau_0 = 0, \|Q_j^* q_{j+1}\| \leq \tau_j, i < j \text{ then }
\]

\[
\|Q_j^* q_{j+1}\| \leq \tau_j.
\]

PROOF. Apply Lemmas 2 and 3 and the error bounds to

\[
Q_j^* q_{j+1} = Q_j^* q_{j+1} / \beta_j + \tau_j e_j.
\]

Let us summarize the procedure. During the \( j \)-th step of a Lanczos run we compute 
\[
\alpha_j, r_j, \beta_j, q_{j+1},
\]
then 
\[
\omega_j, z_j, \text{ and } \kappa_{j+1}.
\]
While \( \kappa_j < 1 \) the algorithm proceeds to step \((j+1)\).

The extra work in updating \( \kappa_j \) is very slight and is dominated by the computation of \( \| (T_{j-1} - \alpha_j) \|. \) This may be bounded as follows.

\[
\| (T_{j-1} - \alpha_j) \| \leq \max_{i<j} (\beta_{i-1} + |\alpha_i - \alpha_j| + \beta_i),
\]
\[
< \bar{\beta} + \max_{k<j} |\alpha_k - \alpha_j|,
\]
\[
< \bar{\beta} + \max(|\alpha - \alpha_j|, |\bar{\alpha} - \alpha_j|),
\]

where 
\[
\underline{\alpha} = \min \alpha_i, \quad \bar{\alpha} = \max \alpha_i, \quad \underline{\beta} = \max (\beta_{i-1} + \beta_i).
\]

Note that the third inequality involves no searching over \( j \) elements provided that \( \alpha, \bar{\alpha}, \) and \( \bar{\beta} \) are updated at each step; \( \alpha = \min(\alpha, \alpha_j), \bar{\alpha} = \max(\bar{\alpha}, \alpha_j), \bar{\beta} = \max(\bar{\beta}, \beta_{j-2} + \beta_{j-1}) \). The second inequality costs \((j-1)\) comparisons and is tighter. The first inequality is \( \| (T_{j-1} - \alpha_j) \|_{\infty}, \) except for the presence of \( \beta_{j-1}, \) and this can never exceed \( \| (T_{j-1} - \alpha_j) \| \) by more than a factor of \( \sqrt{2} \).

A more complicated procedure for monitoring loss of orthogonality is described in [5]. \( 1 - Q_j^* Q_j \) is majorized by a \( j \times j \) matrix \( W_j \), called the scoreboard. The triangular factorization of \( 1 - W \) is updated at each step and the Lanczos run continues until a nonpositive diagonal element appears. The cost is approximately \( j^2/2 \) arithmetic operations and, more seriously, \( j^2 \) storage locations.

The quantity \( \kappa_j \) grows exponentially and its use will certainly terminate Lanczos runs prematurely. This is not necessarily inefficient for the following reason. The Lanczos algorithm yields monotonically improving approximations. It thus can be used for calculations to low accuracy as well as high. It is important not to give the user unwanted figures when the extra cost is significant. If Lanczos is run until \( |q_j^* q_{j+1}| > 0.1 \) (say) then \( \theta_1 \) may already have converged to too many figures.

In practice \( \| A \|_E \) may not be readily available and the
estimate \(\|T_{j-1} - \alpha_j\| + |\alpha_j|\) is used in its place. Sometimes the bound on \(\|d_j\|\) is omitted for simplicity.

7. BEHAVIOR OF THE SPECTRUM OF \(T_j\)

In Figure 1 we show the lowest five eigenvalues of \(T_j\), for \(j = 10, 20, \ldots, 60\), in a Lanczos run on a matrix \(A\) with eigenvalues \(\alpha_i = i, i = 1, 2, \ldots, 253\). Also shown is \(\sigma_j(Q_j)\) and \(\sqrt{1 - \kappa_j}\). The starting vector \(q_1\) was chosen to be rich in the first four eigenvectors.

Three phases may be distinguished in the run. Note that \(p = 4\).

**Early Phase:** \(\theta_i\) is not an accurate approximation to \(\alpha_i\) for \(i = 1, \ldots, p\).

**Middle Phase:** One or more of the \(\theta_i\) have converged to working accuracy and there is still a one-one correspondence between \(\theta_i\) and \(\alpha_i\) for \(i = 1, \ldots, p\).

**Late Phase:** The one-one correspondence has been lost. Among the \(\theta_i\) occur multiple approximations to various \(\alpha_i\) as well as spurious values close to no \(\alpha\).

If \(q_1\) had been chosen to be rich in eigenvectors 1, 3, and 4 the picture would have been more complicated. If \(p\) is
too large then there may be multiple approximations to $\alpha_1$ before any $\theta$ has converged to $\alpha_0$.

Corollary 2 in Section 2 implies that the late phase cannot begin until linear independence is lost to working accuracy. Paige's error analyses [7] show that Phase 1 ends before orthogonality is lost to working accuracy.

It is usual for an outer eigenvalue to converge before an inner one but this cannot be taken for granted. The outcome depends on $q_1$.

We recommend that Lanczos algorithms be tailored to the problem in hand. The urge to write a universal Lanczos program should be resisted, at least until the process is better understood.

REFERENCES


The Lanczos algorithm can be used to approximate both the largest and smallest eigenvalues of a symmetric matrix whose order is so large that similarity transformations are not feasible. The algorithm builds up a tridiagonal matrix row by row and the key question is when to stop. An analysis leads to a stopping criterion which is inspired by a useful error bound on the computed eigenvalues.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HOLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
<tr>
<td>eigenvalues</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eigenvectors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lanczos' algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sparse symmetric matrices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stopping criteria</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>