THE LANCZOS ALGORITHM WITH IMPLICIT DEFLATION

by

B. N. Parlett and D. S. Scott

Memorandum No. UCB/ERL M77/70

2 December 1977

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

THE LANCZOS ALGORITHM WITH IMPLICIT DEFLATION*

by

B.N. Parlett† and D.S. Scott††

Abstract

The simple Lanczos process is very effective for finding a few extreme eigenvalues of a large symmetric matrix along with the associated eigenvectors. Unfortunately the process computes redundant copies of the outermost eigenvectors and has to be used with some skill. In this paper it is shown how a modification called implicit deflation stifles the formation of duplicate eigenvectors without increasing the cost of a Lanczos step significantly. The degree of linear independence among the Lanczos vectors is controlled without the costly process of reorthogonalization.

Key Words. Eigenvalues, Eigenvectors, Symmetric Matrices, Lanczos Method.

---

*Research supported by Office of Naval Research Contract N00014-76-C-0013.

†Department of Mathematics and Computer Science Division, Department of Electrical Engineering and Computer Sciences, and the Electronics Research Laboratory, University of California, Berkeley, CA 94720.

††Department of Mathematics, University of California, Berkeley, CA 94720.

CONTENTS

1. Introduction 3
2. Notational Conventions 6
3. Lanczos in Exact Arithmetic 7
4. Orthogonality versus Convergence 12
5. Implicit Deflation 16
6. When to Pause 19
7. Monitoring the Return of Banished Ritz Vectors 22
8. Flowchart I 23
9. Running Out of Storage 25
10. Multiple Eigenvalues 26
11. Can Low Accuracy be Achieved Safely? 27
12. Flowchart II 28
References 29

1. INTRODUCTION

The Lanczos method is well suited to the task of computing a few (p) eigenvalues and eigenvectors of a large $(n \times n)$ symmetric matrix A. The wanted eigenvalues may be at either, or both, ends of the spectrum. Typical values are $p = 4$ and $n = 1000$; in a typical application A will be positive definite and the smallest eigenvalues will correspond to the natural frequencies which can be excited in some structure after it is perturbed away from equilibrium.

It seems appropriate to give a brief review of the history of the method. Simple processes, like the Power Method, require, in principle, an infinite number of matrix-vector products to converge to an eigenvector. On the other hand, the method of Minimized Iterations, which Lanczos announced in 1950, expands each eigenvector in a convergent series with at most n terms.† However Lanczos' method was promptly switched to a different channel. It was used as a process for computing a tridiagonal matrix T orthogonally congruent to A; $T = Q^*AQ$, $Q = (q_1, q_2, \ldots, q_n)$, $Q^* = Q^{-1}$.

Despite its theoretical attractions the Lanczos process was soon displaced by the Givens (1954) and Householder (1958) methods which employ explicit similarity transformations on A. To compete in accuracy the Lanczos process has to be supplemented with the explicit orthogonalization of the Lanczos vectors $\{q_i\}$ which, in exact arithmetic, would be orthogonal automatically.

In 1970 C. Paige showed that the simple Lanczos procedure, without orthogonalization, was very effective for finding a few of the extreme eigenvalues and their matching eigenvectors.

---

†Convergence is usually very quick for eigenvectors belonging to extreme eigenvalues.

In part this is because the only way A enters the Lanczos algorithm is through a subprogram which computes Ax for any given vector x. The user is free to exploit sparseness and compact storage of A in the coding of this subprogram. Equally important is the fact that the algorithm need not go the whole way. It builds up $Q_j = (q_1,\ldots,q_j)$ and $T_j = Q_j^* A Q_j$ by step j and can often be stopped at values of j as small as $2\sqrt{n}$. Paige showed [Paige 1976] that loss of orthogonality among the Lanczos vectors $\{q_1,q_2,\ldots\}$ was a necessary and sufficient condition, in finite precision arithmetic, for convergence of at least one of T_j's eigenvalues to one of A's eigenvalues.

This left the Lanczos algorithm as a very powerful tool in the hands of an experienced user. However, it did not provide a black box program which could be used "off the shelf" in the same way as eigenvalue programs for small matrices. There are several rather technical reasons for this. For one thing suitable criteria for accepting good approximations, rejecting spurious approximations, or stopping were all rather elusive. Left to itself a simple Lanczos program will run forever, doggedly finding more and more copies of the outer eigenvalues for each new inner eigenvalue it discovers. This uncertainty about the amount of storage which is needed prompted the suggestion, by Golub and others, that the Lanczos method be used iteratively. That is, after k steps the best approximation to an eigenvector is computed and it, or some modification of it, is used as a new starting vector. With this approach the old difficulties take on new forms: how to choose k and how to select the new starting vector.

Another variation which has been used with success is the block form of the Lanczos method. Each step becomes more costly but fewer are needed and this seemed to be the only way to find small clusters of close

or multiple eigenvalues. However the user has to make the difficult choice of the block size.

The remainder of this article describes an inexpensive modification of the simple algorithm (we call it Lanczos with occasional implicit deflation) which permits the simple Lanczos process to be run as originally envisaged. Moreover,

1. There are no redundant copies of eigenvectors.
2. A posteriori error bounds and estimates cost almost nothing and are used in order to stop the program as soon as possible.
3. Multiple eigenvalues, and their eigenvectors, are found naturally, thanks to roundoff error.

Not surprisingly the idea of deflating Ritz vectors did not come out of the blue. Cullum and Donath [Cullum & Donath 1974] found it necessary to remove converged Ritz vectors from their blocks, Lewis [Lewis 1977] found that some deflation helped in a difficult calculation of interior eigenvalues, and Underwood [Underwood 1975] removed such vectors from his blocks when restarting the iterative version of Lanczos. However we do not regard deflation as an aid in adversity but as a tool for producing orthogonal Ritz vectors and thus our deflation is independent of convergence and may occur beforehand, especially when the user wants high accuracy.

2. NOTATIONAL CONVENTIONS

Integers -- i, j, k, ℓ, m, n, p

Scalars -- small Greek letters $\alpha, \beta, \ldots$

(Column) Vectors -- small roman letters $x, y, \ldots$ (except for the integers)

Matrices -- capital roman letters

Identity matrix -- $I = (e_1, e_2, \ldots, e_n)$

Diagonal matrices -- capital Greek letters

SYMMETRIC (non diagonal) MATRICES -- SYMMETRIC LETTERS A, H, M, U, V, W, X

Tridiagonal matrices --

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & & & \bigcirc \\ \beta_1 & \alpha_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ \bigcirc & & \cdot & \alpha_{j-1} & \beta_{j-1} \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix}$$

All vectors are n-dimensional unless the contrary is stated. All square matrices are $n \times n$ unless the contrary is stated.

$A - \xi$ is written for $A - \xi I$.

Span $(b_1, \ldots, b_j)$ denotes the subspace generated by $b_1, \ldots, b_j$.

x^* is the transpose of x.

$\|x\| = \sqrt{x^* x}$, the Euclidean norm.

$\lambda_i[M]$ -- the ith eigenvalue of M (from the left).

Eigenvalue Ordering: $\begin{cases} \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n , \\ \lambda_{-n} \leq \cdots \leq \lambda_{-2} \leq \lambda_{-1} \end{cases}$

$\|M\| = \max_i |\lambda_i[M]| = \max \|Mv\|/\|v\|, \quad v \neq 0.$

(j) -- a formula in the current section

(k.j) -- a formula in section k

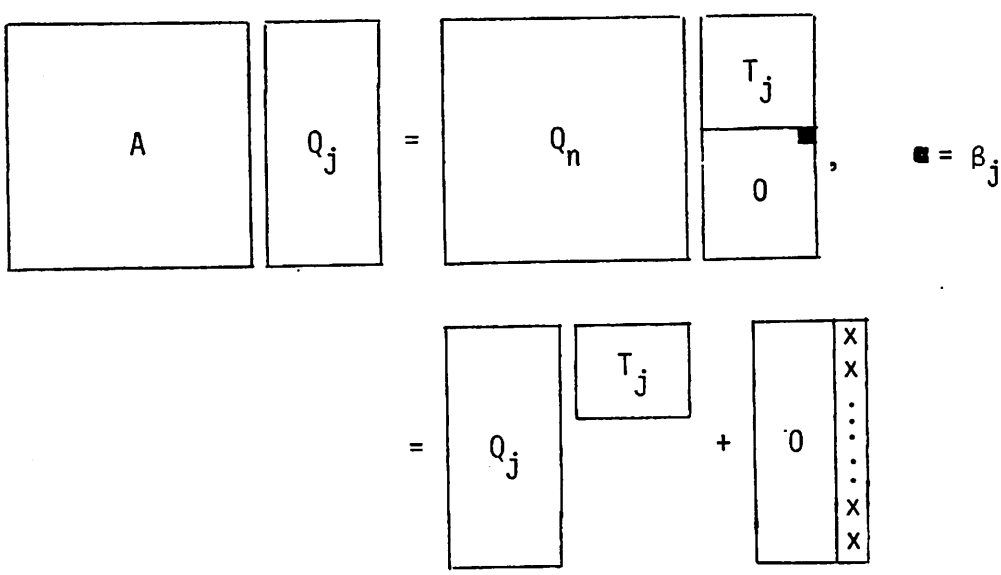
3. LANCZOS IN EXACT ARITHMETIC

A can be reduced to tridiagonal form T_n in many different ways. Let

$$Q_n^* A Q_n = T_n \tag{1}$$

be one such reduction, where $\underset{\sim}{Q}_n = (q_1,\ldots,q_n)$ is orthogonal. If the off-diagonal elements β_i, $i = 1,\ldots,n-1$ of T_n are positive then, in fact T_n and Q_n are completely determined by q_1 or by q_n. Let us write (1) in the form $AQ_n = Q_n T_n$ and see what it says about the $n \times j$ submatrix $Q_j = (q_1,\ldots,q_j)$, $j < n$.

(2)



The last column on the right is $r_j \equiv q_{j+1}\beta_j$. Now (2) can be written compactly as

$$\boxed{AQ_j = Q_j T_j + r_j e_j^* \ , \quad j = 1,\ldots,n} \tag{3}$$

where $e_j^* = (0,\ldots,0,1)$ has j elements and $\beta_n = 0$. From the orthogonality of Q_n follows

$$Q_j^* Q_j = 1 \tag{4}$$

whereas $Q_jQ_j^*$ is an orthogonal projector onto span Q_j.

Note that if β_j $(= \|\underset{\sim}{r}_j\|) = 0$ then span Q_j is an invariant subspace (hurrah!) and T_j is the restriction of A to it. In genuine applications $\beta_j = 0$ never happens (even for $j > n$, in finite precision)!

The Lanczos algorithm builds up Q_j and T_j one column per step. Some important relations follow from (3) and (4) and are independent of the specific implementation of the algorithm.

<u>Orthogonality</u>. Since r_j is a multiple of q_{j+1} it must be orthogonal to all previous q_i, $i = 1,\ldots,j$. In fact this property can be deduced from (3) and (4) without invoking (1).

> LEMMA 1. Let Q_j be any matrix satisfying (3) and (4). Then $Q_{j-1}^*r_j = 0$ and, if $\alpha_j = q_j^*Aq_j$ then $Q_j^*r_j = 0$ too.

A proof is given in [Kahan & Parlett 1974].

The Lanczos algorithm proceeds from Q_j to Q_{j+1} by forcing $q_j^*r_j = 0$ via the choice of α_j and then normalizing r_j to get β_j and q_{j+1}. What could be simpler? Note that $Q_{j-1}^*r_j$ is not forced to vanish because, <u>in exact arithmetic</u>, the lemma guarantees it. From (3)

$$(5) \qquad r_j = r_je_j^*e_j = (AQ_j - Q_jT_j)e_j$$
$$= Aq_j - q_{j-1}\beta_{j-1} - q_j\alpha_j \ .$$

Observe that $q_1,\ldots,q_{j-2}$ are not needed for the computation of α_j, r_j, β_j, q_{j+1} (i.e. the jth step) and so may be put out to a secondary storage medium. This is a very attractive feature of the method.

Suppose that the Lanczos algorithm pauses at the jth step and makes a subsidiary computation of some, or all, of the eigenvalues and eigenvectors of T_j. Let

$$T_j = S_j \Theta_j S_j^* \tag{6}$$

where $S_j = (s_1,\ldots,s_j)$ is $j \times j$ and orthogonal and $\Theta_j = \mathrm{diag}(\theta_1,\ldots,\theta_j)$ has the eigenvalues of T_j. Since $T_j = Q_j^* A Q_j$ the values θ_i and the vectors $y_i = Q_j s_i$ are the (optimal) Rayleigh Ritz approximations to A from the information on hand, i.e. from $\mathrm{span}(Q_j)$. Note that

$$\|AQ_j - Q_j T_j\| = \|r_j e_j^*\| = \beta_j \ . \tag{7}$$

How good are these optimal approximations?

THEOREM 1. There are j eigenvalues of A, call them $\lambda_{1'},\ldots,\lambda_{j'}$, such that $|\lambda_{i'} - \theta_i| \le \beta_j$, $i = 1,\ldots,j$.

A proof is given in [Kahan 1967]. See also [Kahan & Parlett 1976]. The bound covers all the θ_i and does not discriminate between them. We can do better. Let s_{ji} denote the bottom (j^{th}) element of T_j's eigenvector s_i.

THEOREM 2. To each i there is a corresponding eigenvalue of A, call it $\lambda_{i'}$, such that

$$|\lambda_{i'} - \theta_i| \le \beta_j |s_{ji}| \equiv \beta_{ji} \ , \quad i = 1,\ldots,j \ .$$

A proof is given in [Kahan & Parlett 1976]. The quantity $\beta_{ji} = \|(A-\theta_i)\underset{\sim}{y}_i\|$, the ith residual.

> THEOREM 3. Let $Az_{i'} = z_{i'}\lambda_{i'}$, let ψ_i be the angle between z_i' and its Ritz vector $y_i \equiv Q_j s_i$, and let the gap $\gamma_i \equiv \min_{k\neq i'} |\lambda_k-\theta_i|$. Then, for $i = 1,\ldots,j$,
>
> $$|\lambda_{i'}-\theta_i| \leq \beta_{ji}^2/\gamma_i ,$$
>
> $$\tan \psi_i \leq \beta_{ji}/\gamma_i .$$

Proofs are given in [Davis & Kahan 1970].

In principle γ_i is unknown and these bounds are not computable. However $\delta_i = \min_{k\neq i} |\theta_k \pm \beta_{jk}-\theta_i|$ can be used in place of γ_i to give an estimate.

The following result, proved in [Kahan 1967], shows that the previous bounds fail gracefully when Q_j is not orthonormal. Specifically the bounds must be multiplied by $\sqrt{2}/\sigma_1$ where σ_1^2 is the smallest eigenvalue of $Q_j^*Q_j$.

> THEOREM 4. Given A $n\times n$, Q $n\times m$, H $m\times m$ with $\lambda_i[H] = \theta_i$, $i = 1,\ldots,m$. There are m of A's eigenvalues, call them $\lambda_{i'}$, so that for $i = 1,\ldots,m$,
>
> $$|\lambda_{i'}-\theta_i| \leq \sqrt{2}\|AQ-QH\|/\sigma_1(Q)$$
>
> where $\sigma_1^2 = \lambda_1[Q^*Q]$.

The Kaniel-Paige theory [Kaniel 1966] shows that it is the extreme (leftmost and rightmost) eigenvalues which are most likely to be approximated

by some of the θ_i. Moreover the rapidity of convergence, as j increases, depends on the (unknown) gaps between A's eigenvalues. Cases have occurred in which an unusual distribution of eigenvalues coupled with special q_1's have caused interior eigenvalues to come out first. See [Cline, Golub & Platzman 1976].

In principle, then, the Lanczos algorithm should be continued, with periodic pauses, until, and only until, adequate approximations to the wanted eigenvalues and eigenvectors are in hand. This sometimes happens for values of j as small as $2\sqrt{n}$; another attraction of the method.

In practice things are not this simple. With finite precision computation convergence goes hand in hand with loss of linear independence among the q_i and so the error bounds cease to be valid by the time the first of the θ_i converges.

Before leaving the context of exact arithmetic we want to emphasize the value of the bounds β_{ji}. They show why the absence of small β_j does not impede convergence of <u>some</u> of the θ_i to eigenvalues and the computable numbers s_{ji} show <u>which</u> of the θ_i are converging. There are extensions of Theorem 3 which allow a bunch of close θ's and their y's to be treated simultaneously, the gap then becomes the distance of the cluster from eigenvalues not associated with the cluster.

4. ORTHOGONALITY VERSUS CONVERGENCE

The use of finite precision arithmetic provokes significant departures from the exact version of the Lanczos algorithm described above. In order to examine these effects we turn our backs on the quantities which would be produced by use of exact arithmetic and make a standard change of notation. The symbols Q_j, T_j, α_j, β_j from now on denote the computed quantities stored in the computer under these names. We shall not try to compare them with their Platonic counterparts but instead we will seek the (more complicated) relations which do hold between the objects on hand.

The fundamental equation (3.3) becomes

$$AQ_j = Q_jT_j + r_je_j^* - \bar{F}_j \tag{1}$$

where $\bar{F}_j$ accounts for the round off effects. Paige has shown that if the algorithm is implemented correctly, $\bar{F}_j$ is harmless, satisfying an inequality of the form $\|\bar{F}_j\| \leq \phi(n)\varepsilon\|A\|$ for some almost linear function ϕ [Paige 1972, 1976]. The orthogonality relation (3.4) fails and in its place we write

$$\|1-Q_j^*Q_j\| \leq \kappa_j \ . \tag{2}$$

In the next section we give an expression for κ_j but here we focus on the more special and more important issue of orthogonality loss among the vectors $y_i = Q_js_i$, $i = 1,\ldots,j$, which we continue to call Ritz vectors despite the fact that the optimality with which they approximate eigenvectors of A departs hand in hand with Q_j's orthogonality.

In [Paige 1971] can be found the following remarkable results in which the bottom elements s_{ji} $(= e_j^*s_i)$ of T_j's eigenvectors s_i appear again.

THEOREM 5. Consider the jth step of the simple Lanczos algorithm and drop the index j on which all the quantities depend. The computed approximate eigenpairs (θ_i, y_i), $i = 1,\ldots,j$ satisfy

$$y_i^* y_k = [g_{ii}(s_{jk}/s_{ji}) - g_{kk}(s_{ji}/s_{jk}) + f_{ik}]/(\theta_i - \theta_k) \ , \quad i \neq k \ ,$$

where G and F are round off matrices; $\|G\| \doteqdot \|F\| \doteqdot \varepsilon\sqrt{n}\|T\|$, where ε is the relative precision of the arithmetic. Moreover

$$y_i^* q_{j+1} = g_{ii}/\beta_{ji} = g_{ii}/(\beta_j |s_{ji}|) \ , \quad i = 1,\ldots,j \ .$$

The bottom elements of the s_i appear in a special way. With any good program S will be orthonormal (to working accuracy) so that $\sum_{i=1}^{j} s_{ji}^2 = 1$. If

$$(3) \qquad |s_{jk}| \doteqdot |s_{ji}| \doteqdot j^{-1/2} \ , \quad |\theta_i - \theta_k| > \|T\|/100$$

then the error bounds (Theorems 3 and 4) on θ_i and θ_k indicate that they are poor eigenvalue approximations while Theorem 5 shows that y_i and y_k are orthogonal to working accuracy. Conversely, if $|s_{ji}| < 10^{-3}$, say, then θ_i (if isolated) is a good eigenvalue approximation, y_i is good too, and y_i will <u>not</u> be orthogonal to any <u>unconverged</u> y_k (indicated by $s_{jk} \doteqdot j^{-1/2}$). Since S is orthogonal to working accuracy it is Q_j which must have lost orthonormality. The better the approximations θ_i and y_i the greater the departure of Q_j from orthogonality.

A further analysis [Paige 1971] shows that

$$\frac{1}{2} \leq \|y_i\| \leq 2$$

provided that the θ's are not too close. What this means in practice is that Ritz vectors y_i cannot shrink alarmingly unless there are two or more θ's approximating a <u>single</u> eigenvalue λ. Our deflation forestalls this calamity.

As the Lanczos algorithm proceeds, as j increases, the pattern of orthogonality loss among the Lanczos vectors q_i is widespread and has no obvious structure. On the other hand the unconverged Ritz vectors remain mutually orthogonal (to working accuracy) <u>except</u> for strong components in the directions of those Ritz vectors which have converged. The orthogonality of the Lanczos vectors $\{q_i\}$ is also destroyed solely by the addition of significant components of the converged Ritz vectors.

Example of Loss of Orthogonality

$n = 6$

$A = \mathrm{diag}(0., .00025, .0005, .00075, .001, 10.)$

$q_1 = 6^{-1/2}(1.,1.,1.,1.,1.,1.)^T$

Unit round off $\doteqdot 10^{-14}$

Simple Lanczos was run for six steps. $Y_6 = Q_6 S_6$.

$$Q_6^* Q_6$$

$$\begin{bmatrix}
.10E{+}01 & .75E{-}14 & -.30E{-}10 & .25E{-}06 & .97E{-}02 & .41E{+}00 \\
.75E{-}14 & .10E{+}01 & .33E{-}10 & .55E{-}06 & .22E{-}01 & .91E{+}00 \\
-.30E{-}10 & .33E{-}10 & .10E{+}01 & -.97E{-}10 & .19E{-}05 & .79E{-}04 \\
.25E{-}06 & .55E{-}06 & -.97E{-}10 & .10E{+}01 & .11E{-}09 & .23E{-}08 \\
.97E{-}02 & .22E{-}01 & .19E{-}05 & .11E{-}09 & .10E{+}01 & -.12E{-}12 \\
.41E{+}00 & .91E{+}00 & .79E{-}04 & .23E{-}08 & -.12E{-}12 & .10E{+}01
\end{bmatrix}$$

$$\begin{array}{c|cccccc}
\theta_i \longrightarrow & .62E{-}05 & .32E{-}03 & .68E{-}03 & .99E{-}03 & .10E{+}02 & .10E{+}02 \\
\downarrow & & & Y_6^* Y_6 & & & \\
\hline
.62E{-}05 & .10E{+}01 & .53E{-}10 & .18E{-}10 & .16E{-}13 & -.12E{-}12 & -.41E{-}08 \\
.32E{-}03 & .53E{-}10 & .10E{+}01 & -.39E{-}14 & -.18E{-}10 & -.93E{-}13 & -.98E{-}08 \\
.68E{-}03 & .18E{-}10 & .39E{-}14 & .10E{+}01 & -.53E{-}10 & -.78E{-}13 & -.98E{-}08 \\
.99E{-}03 & .16E{-}13 & -.18E{-}10 & -.53E{-}10 & .10E{+}01 & -.13E{-}12 & -.41E{-}08 \\
.10E{+}02 & -.12E{-}12 & -.93E{-}13 & -.78E{-}13 & -.13E{-}12 & .10E{+}01 & .10E{+}01 \\
.10E{+}02 & -.41E{-}08 & -.98E{-}08 & -.98E{-}08 & -.41E{-}08 & .10E{+}01 & .10E{+}01
\end{array}$$

Note that the general loss of orthogonality seen in $Q_6^* Q_6$ is represented in $Y_6^* Y_6$ as the second copy of the eigenvector associated with the eigenvalue 10.

5. IMPLICIT DEFLATION

One way to restore orthogonality to Q_j is to use the modified Gram-Schmidt process in order to force q_{j+1} to be orthogonal to all previous q's. Besides the ever increasing expense in arithmetic operations this reorthogonalization process requires the presence of all the q_i at each step. Paige's result suggests that linear independence of the q's can be maintained by merely orthogonalizing the q's against those few Ritz vectors which have nearly converged. Now converged Ritz vectors are eigenvectors and so the orthogonalized q's are precisely what would be obtained from the Lanczos algorithm using A after deflation of the known eigenvectors. Hence the name of our algorithm.

The modification of the simple Lanczos process is as follows. At each pause T_j is diagonalized and the bounds on the as-yet-uncomputed Ritz vectors are inspected. Those Ritz vectors with error bounds less than some tolerance are declared good, are computed, and then stored in the fast memory. From that point until the next pause all future q's are kept orthogonal to these directions.

It might appear to be necessary to orthogonalize only q_{j+1} and q_{j+2} against these good y's. It follows from (3.5) that all subsequent q's would remain orthogonal to them. In finite precision however the error vector in each computation of Aq_k will bring back small multiples of <u>all</u> A's eigenvectors. Fortunately it is not necessary to orthogonalize r_j against the y's at every step as Section 7 reveals.

The purpose of implicit deflation is to prevent the computation of many unwanted copies of the well separated outer eigenvectors. This reduces the number of Lanczos steps required to compute the wanted eigenvalues and eigenvectors and so keeps the number of calls on the large matrix A as

low as possible. The algorithm must compute and store good Ritz vectors even if some of them are not wanted by the user. For example, if the three eigenvectors at the left end of the spectrum are wanted the algorithm may well have computed three or more eigenvectors at the right end as well, should they happen to be better separated from the rest of the spectrum than are the ones we want.

Example of Implicit Deflation

$n = 6$

$A = \mathrm{diag}(0., .00025, .0005, .00075, .001, 10.)$

$q_1 = 6^{-1/2}(1.,1.,1.,1.,1.,1.)^T$

Unit Round off $\doteqdot 10^{-14}$

Simple Lanczos was run for six steps.

$$Q_6^* Q_6$$

$$\begin{bmatrix} .10E{+}01 & .75E{-}14 & -.30E{-}10 & .25E{-}06 & .97E{-}02 & .41E{+}00 \\ .75E{-}14 & .10E{+}01 & .33E{-}10 & .55E{-}06 & .22E{-}01 & .91E{+}00 \\ -.30E{-}10 & .33E{-}10 & .10E{+}01 & -.97E{-}10 & .19E{-}05 & .79E{-}04 \\ .25E{-}06 & .55E{-}06 & -.97E{-}10 & .10E{+}01 & .11E{-}09 & .23E{-}08 \\ .97E{-}02 & .22E{-}01 & .19E{-}05 & .11E{-}09 & .10E{+}01 & -.12E{-}12 \\ .41E{+}00 & .91E{+}00 & .79E{-}04 & .23E{-}08 & -.12E{-}12 & .10E{+}01 \end{bmatrix}$$

The Lanczos algorithm with implicit deflation was run for six steps. It paused after four steps and computed a good Ritz vector for the eigenvalue 10. It then took two more steps orthogonalizing against this vector.

$$Q_6^* Q_6 \text{ for Implicit Deflation}$$

$$\begin{bmatrix} .10E{+}01 & .75E{-}14 & -.30E{-}10 & .25E{-}06 & -.11E{-}09 & .92E{-}10 \\ .75E{-}14 & .10E{+}01 & .33E{-}10 & .55E{-}06 & .51E{-}10 & -.36E{-}10 \\ -.30E{-}10 & .33E{-}10 & .10E{+}01 & -.97E{-}10 & -.44E{-}10 & -.37E{-}07 \\ .25E{-}06 & .55E{-}06 & -.97E{-}10 & .10E{+}01 & .24E{-}07 & -.64E{-}03 \\ -.11E{-}09 & .51E{-}10 & -.44E{-}10 & .24E{-}07 & .10E{+}01 & .10E{-}13 \\ .92E{-}10 & -.36E{-}10 & -.37E{-}07 & -.64E{-}03 & .10E{-}13 & .10E{+}01 \end{bmatrix}$$

Note that the leading 4×4 principal minor is the same in both matrices. Note that robust linear independence has been maintained by the implicit deflation scheme.

6. WHEN TO PAUSE

There are four possible strategies for deciding when to pause. The simplest (and cheapest) is to pause every m steps where m is some constant, possibly depending on $n = \dim(A)$ but independent of all other characteristics of A. Such a plan is completely insensitive to the loss of orthogonality of Q_j and is unsatisfactory in practice.

Paige and others have suggested keeping q_1 in fast store and computing $q_1^* q_j$ as a measure of the loss of orthogonality. This is not cheap since it requires the storage of an n-vector as well as the computation of a vector inner product at each step. This scheme usually works very well. However this estimate is a lower bound rather than an upper bound on $\|I-Q_j^* Q_j\|$. Therefore, on occasion, the pause may come too late and disastrous failures of this kind are possible in practice. Furthermore it is not clear how to apply this scheme, after the first pause, for deciding when to pause again.

Kahan and Parlett [1974, 1976] have described two different schemes for bounding $\|I-Q_j^* Q_j\|$, the scoreboard and the kappa bound. Since the program uses the kappa bound it will be described in more detail.

Some error is made in normalizing any n-vector. Thus

$$(1) \qquad |1 - \|q_i\|^2| \le \kappa_1 \quad \text{for all } i ,$$

where κ_1 depends on the arithmetic unit, the square root routine and other details of the program. Also let

$$(2) \qquad \|Q_i^* q_{i+1}\| \le \zeta_i \quad \text{for all } i .$$

More attention will be given to ζ_i below. For $i = 1,2,\ldots$ we define κ_{i+1} by

(3) $$\kappa_{i+1} \equiv \left\| \begin{bmatrix} \kappa_i & \zeta_i \\ \zeta_i & \kappa_1 \end{bmatrix} \right\| = [\kappa_i + \kappa_1 + \sqrt{(\kappa_i - \kappa_1)^2 + 4\zeta_i^2}]/2 \ .$$

(4) $$\boxed{\text{LEMMA 2. If } \|1 - Q_j^* Q_j\| \le \kappa_j \text{ then } \|1 - Q_{j+1}^* Q_{j+1}\| \le \kappa_{j+1}.}$$

<u>Proof</u>.

$$\|1 - Q_{j+1}^* Q_{j+1}\| \le \left\| \begin{bmatrix} \|1 - Q_j^* Q_j\| & \|-Q_j^* q_{j+1}\| \\ \|-Q_j^* q_{j+1}\| & \|1 - q_{j+1}^* q_{j+1}\| \end{bmatrix} \right\| \qquad \square$$

The vector q_{j+1} is obtained by dividing r_j of (4.1) by β_j. Hence

(5) $$\|Q_j^* q_{j+1}\| \le \|Q_j^* r_j\|/\beta_j + \|Q_j^* g_j\| \ , \quad \|g_j\| < \epsilon \ .$$

In order to update κ_j we must first update ζ_j. A bound for $\|Q_j^* r_j\|$ depends on an interesting relation holding among the quantities satisfying (4.1), namely

(6) $$AQ_j = Q_j T_j + r_j e_j^* - \bar{F}_j \ ,$$

where

(7) $$\boxed{\begin{aligned} \text{LEMMA 3. } Q_j^* r_j = {} & [(1 - Q_j^* Q_j) T_j - (1 - e_j e_j^*) T_j (1 - Q_j^* Q_j)] e_j \\ & - \bar{F}_j^* q_j + (q_j^* A q_j - \alpha_j) e_j + Q_j^* \bar{f}_j \ . \end{aligned}}$$

The proof is given in [Kahan & Parlett 1974]. Please note that the result shows exactly how all the steps in the algorithm contribute to the loss of orthogonality. In exact arithmetic (Lemma 1) only the term $(q_j^* A q_j - \alpha_j) e_j$ appeared and this was made zero by the definition of α_j. None of the terms in the expression need be small relative to the divisor β_j and this is how orthogonality leaks away. The specific algorithm for

ζ_j is relegated to the category of fine detail. The fact is that ζ_j can be computed with about 10 arithmetic operations, independent of j and n.

With κ_j in hand it is possible to implement the natural policy to <u>run Lanczos until the</u> q's <u>begin to lose orthogonality</u>, then pause, do a Rayleigh-Ritz approximation and see what the situation is. The right moment to pause is the first j at which a Ritz vector is good (that is, worth deflating against). It is not crucial to find the right j, but it is important to pause before a spurious second copy of an eigenvalue appears. The program pauses as soon as

$$\kappa_j \geq \text{TOLKAP} \ . \tag{8}$$

The current value of TOLKAP is .9.

After the Rayleigh-Ritz approximation is made there is enough information to estimate $\|1-Q_j^* Q_j\|$ quite accurately. This allows the κ recurrence to be reset accurately so that κ can be used to determine when to make the next pause. This is another reason why the program uses the κ bound.

7. MONITORING THE RETURN OF BANISHED RITZ VECTORS

Let y be a good normalized Ritz vector and let τ_j be a bound on $|y^*q_j|$, the unwanted component of y in q_j. There is a simple three term recurrence governing the τ's. We have

(1) $\quad Ay = \theta y + r \quad$ (r is not to be confused with r_j) .

The quantities computed in the jth step of the Lanczos algorithm satisfy

(2) $$q_{j+1}\beta_j = Aq_j - q_j\alpha_j - q_{j-1}\beta_{j-1} + f_j$$

where f_j accounts for the round off and $\|f_j\| \leq \nu\epsilon\|A\|$ for some constant ν which depends on A but not on j. Hence

(3) $$y^*q_{j+1}\beta_j = y^*Aq_j - y^*q_j\alpha_j - y^*q_{j-1}\beta_{j-1} + y^*f_j \ .$$

Because $|y^*q_j| \leq \tau_j$, (1) and (2) yield

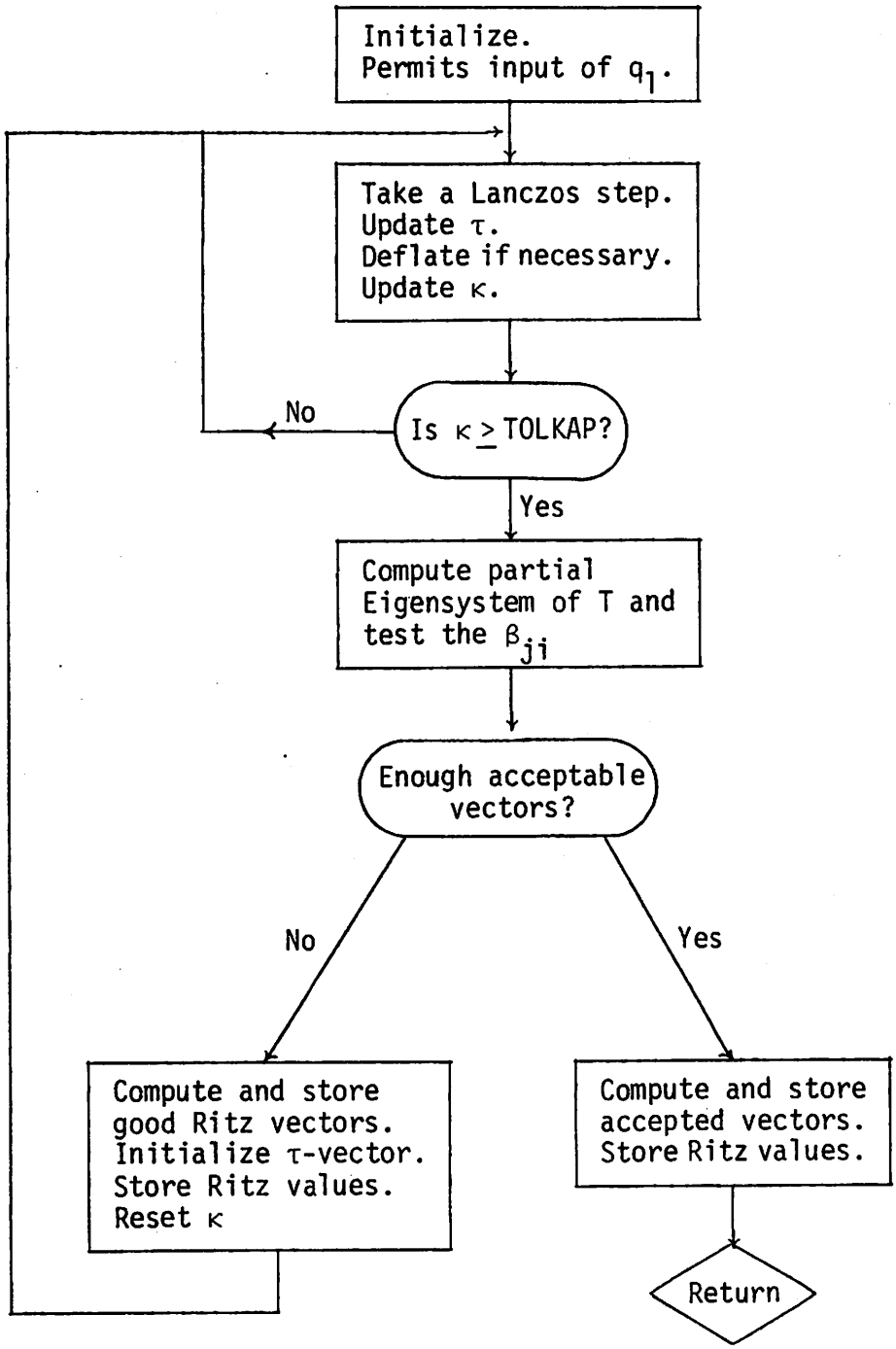
(4) $$|y^*q_{j+1}| \leq [|\theta-\alpha_j|\tau_j + \beta_{j-1}\tau_{j-1} + \|r\| + \nu\epsilon\|A\|]/\beta_j \equiv \tau_{j+1} \ .$$

Since ν and $\|A\|$ are not readily available, the program sets $\nu = 1$ and uses a bound on $\|T_j\|$ instead. Each time that a pair of q's are explicitly orthogonalized against y the corresponding τ's are set to $\varepsilon\|T_j\|$. Then the recurrence is updated by (4) at each step and tested. As soon as τ_j again exceeds the tolerance y is explicitly deflated out of q_j and q_{j+1}. The tolerance is not critical ($1/n$ seems to be an appropriate value).

Along with each computed Ritz vector is stored the associated eigenvalue θ_i, the residual norm estimate β_{ji}, and cells for the current and previous τ-values. The cost of updating this information is negligible. Thus τ may be thought of as a two rowed array of length equal to the number of good Ritz vectors.

8. FLOWCHART I

Lanczos with implicit deflation, ample storage and no multiple eigenvalues.



Notes on Flowchart I

Note 1. κ monitors loss of orthogonality and is described in Section 5.

Note 2. τ monitors the components of the current Lanczos vector in the directions of the good Ritz vectors. See Section 7.

Note 3. A good Ritz vector need not be one which is wanted. It may not be quite accurate enough to be accepted yet or it may belong to the wrong end of the spectrum.

Note 4. Good Ritz vectors will be recomputed at each pause.

9. RUNNING OUT OF STORAGE

Information is always lost when the Lanczos process is restarted. Since our algorithm maintains robust linear independence among the q-vectors at a modest cost, the usual reason for restarting is not present. However since the available storage may be quite limited on some computer systems, the program must be capable of restarting when necessary and as much information as possible should be retained.

Every time that storage has been exhausted, the program calculates and permanently stores all acceptable wanted R-vectors and all good vectors not among those desired. It also stores the corresponding Ritz values and sorts all the permanently stored Ritz vectors by increasing Ritz value.

There remains the question of what the starting vector should be. The new q_1 is currently taken to be a linear combination of some of the Ritz vectors which are not acceptable. The one with smallest residual β_{ji} is always used as well as any others which have converged to an accuracy of $\sqrt{TOL}$. The weights for the linear combination are the reciprocals of the β_{ji}. Other choices for the restart vector could be made. Before restarting q_1 is orthogonalized against all the permanent vectors and the τ-vector (see section 7) is initialized.

10. MULTIPLE EIGENVALUES

Since the Lanczos algorithm only examines the subspace spanned by the vectors $(q_1, Aq_1, A^2q_1, \ldots, A^jq_1)$, it is unable to detect any eigenvector which is orthogonal to q_1. In particular, it is incapable of finding multiple eigenvalues. If V is the eigenspace of a multiple eigenvalue λ, then the Lanczos algorithm will find only the single eigenvector in the direction of the projection of q_1 onto V.

Despite this, the program finds multiple eigenvalues quite naturally. Rounding errors introduce components in all directions. After one eigendirection of a multiple eigenvalue has been found the components in orthogonal directions will persist after deflation. These components will grow as the algorithm continues until a second eigenvector, orthogonal to the first, has been found.

Since multiple eigenvalues are found sequentially instead of simultaneously, a more sophisticated termination criterion is needed. For example, if A has a double eigenvalue at zero, a simple eigenvalue at 1, and the rest of the spectrum larger than 2, then the program will find an eigenvector of 0 and the eigenvector of 1 at about the same time. Therefore if the program finds enough acceptable vectors it must decide whether to start over again to test for undisclosed multiplicities. Currently the program makes a test run if, at the last pause, more than one acceptable eigenvalue is found, or if the only one found is in the convex hull of the rest of the acceptable eigenvalues found so far. This strategy is rather conservative and will often make test runs which are unnecessary. However with this criterion multiple eigenvalues will always be correctly unearthed.

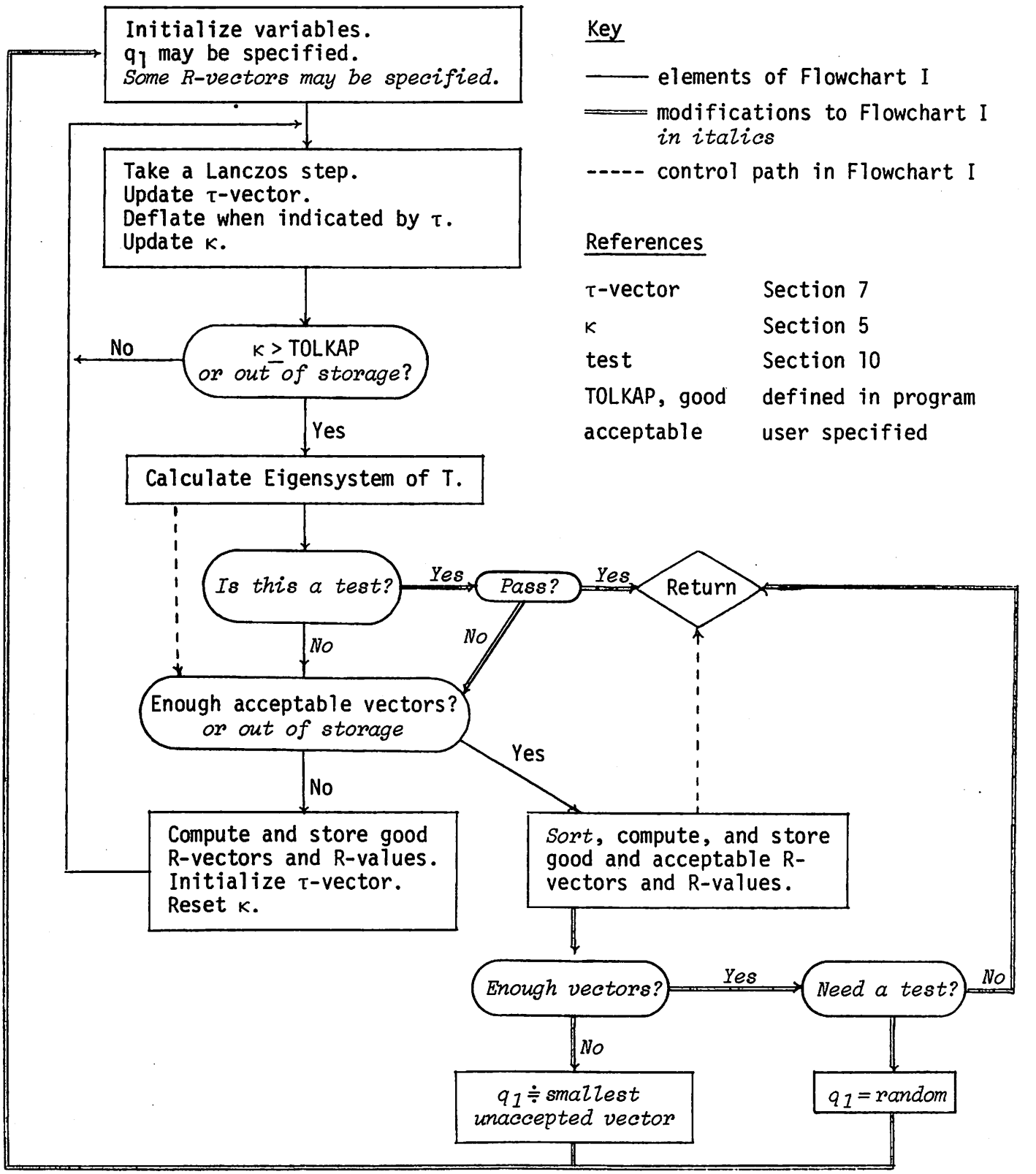
11. CAN LOW ACCURACY BE ACHIEVED SAFELY?

Yes.

The user specifies the desired accuracy with the input parameter TOL. The only time this parameter is used is in determining which of the desired vectors should be made permanent when the process is restarted. A simple perturbation argument shows that any eigenvalue found after a restart is perturbed by no more than the maximum of the norms of the residuals of the permanent vectors. Consequently eigenvalues found on a second pass will be of the same order of accuracy as those found earlier.

12. FLOWCHART II

Motifications of Flowchart I to cope with limited storage and multiple eigenvalues.



REFERENCES

A.K. Cline, G.H. Golub, G.W. Platzman, "Calculation of Normal Modes of Oceans Using a Lanczos Method," in Sparse Matrix Computations (ed. J. Bunch and D. Rose), Academic Press, New York, 1976.

J. Cullum and W.E. Donath, "A Block Generalization of the Symmetric s-step Lanczos Algorithm," Report #RC 4845 (#21570), IBM Thomas J. Watson Research Center, Yorktown Heights, New York (1974).

C. Davis and W. Kahan, "The Rotation of Eigenvectors by a Perturbation--III," SIAM Journal of Numerical Analysis 7, 1 (March 1970).

G.H. Golub, R. Underwood, J.H. Wilkinson, "The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem," Technical Report STAN-CS-72-270, Computer Science Department, Stanford University (

W. Kahan, "Inclusion Theorems for Clusters of Eigenvalues of Hermitian Matrices," Computer Science Report, University of Toronto, Canada (1967).

W. Kahan and B. Parlett, "An Analysis of Lanczos Algorithms for Symmetric Matrices," Electronics Research Memorandum ERL-M467, University of California (September 1974).

W. Kahan and B. Parlett, "How Far Should You Go with the Lancsoc Algorithm?", in Sparse Matrix Computations (eds. J. Bunch and D. Rose), Academic Press, New York, 1976.

S. Kaniel, "Estimates for Some Computational Techniques in Linear Algebra," Mathematics of Computation 20, 95 (July 1966), 369-378.

C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," J. Res. Nat. Bur. Stand. 45 (1950), 255-282.

J. Lewis, "Algorithms for Sparse Matrix Eigenvalue Problems," Technical Report STAN-CS-77-595, Computer Science Department, Stanford University (1977).

C.C. Paige, "The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices," Ph.D. Thesis, University of London (1971).

C.C. Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," J. Inst. Math. Applics. 10 (1972), 373-381.

C.C. Paige, "Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix," J. Inst. Math. Applics. 18 (1976), 341-349.

R. Underwood, "An Iterative Block Lanczos Method for the Solution of Large Sparse Symmetric Eigenproblems," Ph.D. Thesis, Stanford University, STAN-CS-75-496 (1975).