SU 326 P30-41

AN ITERATIVE BLOCK LANCZOS METHOD FOR THE SOLUTION OF LARGE SPARSE SYMMETRIC EIGENPROBLEMS

by

Richard Underwood

STAN-CS-75-496
MAY 1975

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Patent Cl 9/16/75

AN ITERATIVE BLOCK LANCZOS METHOD FOR THE SOLUTION OF LARGE SPARSE SYMMETRIC EIGENPROBLEMS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Richard Ray Underwood

May 1975

ACKNOWLEDGEMENTS

I am deeply indebted to my research adviser, Professor Gene H. Golub, for his support and friendship throughout the years of my graduate study and for originally suggesting the topic of this dissertation.

Special thanks are due to Professor Cleve B. Moler for his patience and considerable expenditure of time and energy on my behalf.

I am also indebted to two of my teachers at Stanford, Professor John C. Herriot and the late Professor George E. Forsythe for their interest and support, both financial and academic.

These acknowledgements would not be complete without mentioning those who afforded me the priceless gift of their friendship. Included in this group are Mike and Fran Malcolm, Mike and Prue Saunders, Alan and Doris George, Aurora Morgana, Margaret Wright, John Palmer, John and Fran Lewis, Richard and Erin Brent, Linda Kaufman, David and Pauline Saunders, John Bolstad, George Ramos, Dee Larson, and Jean Cook.

I wish to thank Phyllis Winkler for typing the manuscript. Without her fast and expert work over a very short period of time, all would have come to nought.

Last, but certainly not least, to my wife, Mary, whose love and friendship made it possible for me to finally finish my work on this thesis, I give a very special thanks that words are incapable of expressing. To Mary, I dedicate my thesis.

This work was supported by the U.S. Atomic Energy Commission under grant AT-(04-3)-326-PA30 and the National Science Foundation under grants GJ35135X and GJ29988X.

TABLE OF CONTENTS

Chapter			Page
I.	INTRODUCTION.		1
	1.1	Historical Background and Survey of Literature. . .	2
	1.2	The Accuracy of Computed Eigenvalues and Eigenvectors.	4
	1.3	Outline of Thesis and Summary of Results.	6
II.	THEORETICAL DEVELOPMENT		9
	2.1	Notation, Definitions, and Basic Results.	9
	2.2	Restricted Operators.	14
	2.3	The Basic Idea.	17
	2.4	Lanczos' Algorithm for Symmetric Matrices	21
	2.5	Extending the Basic Idea.	29
	2.6	The Error in the Least Eigenvalues of A Restricted to $L(s,X,A)$	31
	2.7	A Block Lanczos Algorithm	44
	2.8	Iterating to Improve Accuracy	60
	2.9	Restricting A to a Space Orthogonal to Computed Eigenvectors	62
	2.10	A Complete Iterative Block Lanczos Algorithm. . . .	68
III.	IMPLEMENTATION.		72
	3.1	Reorthogonalization in the Block Lanczos Method . .	72
	3.2	Estimating Accuracy and Convergence Criteria. . . .	75
	3.3	Choice of Block Size.	81
	3.4	Program and Storage Organization.	86
IV.	NUMERICAL EXAMPLES.		94
APPENDIX: A FORTRAN PROGRAM.			110
BIBLIOGRAPHY.			132

LIST OF TABLES

	Page
Table 3.3.1.	82
Table 3.3.2.	82
Table 3.3.3.	83
Table 4.1.	98
Table 4.2.	99
Table 4.3.	102
Table 4.4.	108
Table 4.5.	109

1. INTRODUCTION

In this dissertation, we will be concerned with the development, implementation and application of an algorithm to solve the following problem:

> Compute accurate approximations to the r least eigenvalues of a large, sparse symmetric matrix A where r is much less than n, the order of A.

Problems of this type often arise in mechanics where A represents a discrete differential operator, the order of A is one thousand or more, fewer than 5% of its elements are non-zero, and r is only a small fraction of the value of n.

The more common algorithms for solving general symmetric eigenproblems such as the Householder, QR, bisection, and inverse iteration methods, can generally not be applied to the above problem because they would require excessive amounts of storage or computer time. In contrast to these methods, our algorithm does not transform the matrix A in any way, and therefore any special structure that A may possess is preserved. Rather, the only way in which A is used is in computing the product Ay given a vector y, and if A is sparse, even though of large order, this matrix multiplication can usually be carried out efficiently.

Our method is organized about a Block Lanczos algorithm which is an extension and generalization of a method originally proposed by Lanczos. In the next section, we will review the historical background of the Lanczos method. In Section 1.2 we will make some general remarks concerning the accuracy of computed eigenvalues and eigenvectors, and in Section 1.3, we will outline our thesis and summarize our results.

1.1 Historical Background and Survey of Literature

In 1950, Lanczos [13] described an algorithm which could be used to compute some or all of the eigenvalues and eigenvectors of a symmetric matrix A. Although not a method for computing eigenvalues and eigenvectors per se, it could be used to compute the minimum polynomial p of A with respect to a vector x (cf. §2.4) and a sequence of vectors $(x_i)_{i=1}^{m}$ where $1 \leq m \leq n$ and n is the order of A. Some or all of the eigenvalues of A could be found by computing the roots of p and Lanczos showed how the x_i could be combined to form eigenvectors once the eigenvalues had been found. Although very attractive at first glance, Lanczos' method presented some unforseen difficulties (cf. §2.4) when implemented and applied, and with the development of the Givens and bisection methods and then the Householder and QR methods, it was soon set aside as a method of general application.

In recent years, however, interest in Lanczos' method has increased due to its consideration as a means of computing a few of the extreme eigenvalues and eigenvectors of large, sparse, symmetric matrices. From a modern viewpoint, Lanczos' method is a way of obtaining from A a symmetric tridiagonal matrix T_m, say, where T_m is of order $m \leq n$. The eigenvalues of T_m are also eigenvalues of A and the eigenvectors of T_m can be used to compute eigenvectors of A. Let T_s stand for the s-by-s leading principal submatrix of T_m, $s \leq m$. T_s can be computed by carrying out s steps of the Lanczos method and stopping short of its normal completion point. In 1966, Kaniel [11] published a paper containing results which implied that a few of the eigenvalues of T_s at either end of its spectrum will usually be very

accurate approximations to the corresponding eigenvalues of A for relatively small values of s. (Lanczos was also aware of this phenomenon. See [13], p. 270.) Kaniel also gave bounds on the errors in the eigenvalues of T_s as approximations to the eigenvalues of A and showed that for the extreme eigenvalues, they decrease rapidly as s increases. Kaniel's work suggested that for the relatively small cost of computing the s-by-s matrix T_s and its eigenvalues and eigenvectors, one could obtain accurate approximations to some of the eigenvalues and eigenvectors of A.

During the application of the Lanczos method, a sequence of vectors $(x_i)_{i=1}^{s}$ is computed which, although orthogonal in exact arithmetic, in practice with finite precision arithmetic, lose orthogonality very rapidly. In order to be sure of the stability of the method, these vectors must be reorthogonalized with respect to all previously computed vectors as they are generated. Were it not for this shortcoming, Lanczos' method would be an attractive approach in general for the solution of the eigenproblem. Motivated by Kaniel's work, Paige [17] carried out a detailed study of Lanczos' method and found that useful results could be computed even if reorthogonalization is not carried out. The advantage of this approach is that the entire sequence of vectors $(x_i)_{i=1}^{s}$ need not be kept around at all times, resulting in a considerable savings in both storage and time. A drawback is that, unless this method is carefully applied, the computed results may indicate that A has multiple roots even though this may not be the case. This same phenomenon was reported by Godunov and Prokopov [6] who applied the Lanczos method in the same way as Paige to the solution of the eigenproblem of an elliptic differential operator.

Aside from the Lanczos method, one of the principal methods of solving the eigenproblem for large sparse symmetric matrices is the power method [23]. The method known as simultaneous iteration [19,20] is based on the power method but iterates simultaneously with several vectors by means of which improved rates of convergence are achieved. In 1973, Golub suggested to this author that a similar improvement might be realized for the Lanczos method if it too were extended so as to work simultaneously with several vectors. This thesis is concerned with the development and application of a method based on a Block Lanczos algorithm following the suggestion of Golub.

Cullum and Donath [4] have also developed and applied a Block Lanczos algorithm but their use and implementation of the method differs from ours. Kahan and Parlett [10] have recently given an error analysis of Lanczos' method which is based on Kahan's work with a Block Lanczos method dating back to the late 1950's.

The papers mentioned previously deal primarily with the use of the Lanczos method as an iterative algorithm in a fashion suggested by Kaniel's paper. For more general discussions of Lanczos' method see Wilkinson [23], Golub [8], Golub, Underwood, and Wilkinson [7], and Paige [15,16].

1.2 The Accuracy of Computed Eigenvalues and Eigenvectors

If A is a symmetric matrix of order n, then the eigenvalues λ_i and eigenvectors q_i satisfy

$$Aq_i - \lambda_i q_i = \theta \quad , \quad i = 1,\ldots,n \quad ,$$

where θ is the zero vector and the q_i are orthonormal. Generally

we can only compute values μ_i and vectors x_i with $\|x_i\| = 1$ such that

$$Ax_i - \mu_i x_i = \rho_i \quad , \quad i = 1,\ldots,n \quad , \tag{1.3.1}$$

where

$$\|\rho_i\|_2 = \epsilon_i \quad . \tag{1.3.2}$$

and $\|\cdot\|_2$ denotes the spectral norm. By Weinstein's inequality [21], we can be sure that there is an eigenvalue λ of A such that

$$|\lambda - \mu_i| \leq \epsilon_i \quad . \tag{1.3.3}$$

However, we can <u>not</u> be sure that the computed vector x_i is close to an eigenvector of A, and this is an inherent limitation in our computations. The most that we can say is that x_i is close to the subspace spanned by the eigenvectors corresponding to the eigenvalues which are near to λ. If λ is a single or multiple eigenvalue which is isolated from the other eigenvalues, then x_i will be close to an eigenvector. If λ is one of a cluster of very close but distinct eigenvalues, then x_i may not be close to an eigenvector even if the corresponding ϵ_i in (1.3.3) is very small.

<u>Example</u>. Let

$$A = \begin{bmatrix} 1 & 10^{-10} \\ 10^{-10} & 1 \end{bmatrix} \quad , \quad \mu = 1 \text{ , and } \quad x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad .$$

It follows that

$$Ax - \mu x = \begin{bmatrix} 0 \\ 10^{-10} \end{bmatrix}$$

so that

$$\epsilon_i = 10^{-10} \quad .$$

We can conclude that there is an eigenvalue λ of A such that

$$|\lambda - \mu| \leq 10^{-10} \quad .$$

(In fact, the eigenvalues of A are $1+10^{-10}$ and $1-10^{-10}$.) However, the eigenvectors of A are

$$q_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad q_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

and x is close to neither vector.

Hence, throughout this thesis, statements to the effect that we will compute accurate approximations to the eigenvectors of a matrix are made with this limitation in mind. Our goal will be to find scalars μ_i and vectors x_i which satisfy (1.3.2) with ϵ_i relatively small. How close these computed scalars and vectors are to the actual eigenvalues and eigenvectors of A will depend on the spectrum of A and the magnitudes of the ϵ_i.

Note: It is often possible to compute a posteriori bounds on the errors in computed values and vectors which are much smaller than those indicated here. See, for example, Wilkinson [23], Paige [16], Stewart [22], Davis and Kahan [5], and Ortega [14].

1.3 Outline of Thesis and Summary of Results

In Chapter 2, we will present a theoretical development of our algorithm. We will review the notion of a restricted operator and show that the extreme eigenvalues and vectors of a matrix A restricted to a particular subspace will be accurate approximations to the corresponding

eigenvalues and eigenvectors of A. We will review Lanczos' method and see how it can be used to compute the eigenvalues and eigenvectors of the above restricted operator. We then generalize these notions to work with several vectors simultaneously. In particular, we will extend Kaniel's basic result on the rate of convergence of the least eigenvalue computed using the Lanczos method to the least eigenvalue of A. We will also develop a Block Lanczos algorithm which is an extension of Lanczos' original algorithm. We will then construct a new algorithm which utilizes our Block Lanczos algorithm to compute a specified number of the least eigenvalues and corresponding eigenvectors of a symmetric matrix to a given accuracy.

In Chapter 3, we consider the practical aspects of implementing the algorithm developed in Chapter 2. The number of vectors we choose to iterate with at each application of the Block Lanczos algorithm affects the number of operations required to compute a given number of vectors. In Chapter 3, we will consider some of the problems associated with the choice of block size and suggest some strategies based on our theoretical knowledge of the algorithm and our computational experience.

An important issue relating to the use of the Lanczos method is whether reorthogonalization is carried out. In our current application, we do reorthogonalize the vectors generated by our Block Lanczos algorithm. In Chapter 3, we discuss this issue and indicate why we have decided on this course.

Also in Chapter 3, we consider various aspects of the program implementing our method. We discuss program and data organization, how to estimate the accuracy of computed results in the context of the Lanczos method, the effects of round-off errors, and give operation counts.

Finally, in Chapter 4, we present the results of numerical experiments on a number of problems comparing our method with the method of simultaneous iteration. We will see that in most cases our method is superior to the latter method in terms of the amount of work required to compute a given number of vectors to a specified accuracy.

2. THEORETICAL DEVELOPMENT

In this chapter we will be concerned with the development of an algorithm to solve the following problem:

Given a symmetric matrix A of order n with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $q_1, q_2, \ldots, q_n$, and given an integer r greater than zero and less than or equal to n, compute accurate approximations to λ_i and q_i for $i = 1, \ldots, r$.

We will define the notion of a restricted operator in Section 2.2 and show in Section 2.3 that the least eigenvalue of A restricted to the subspace spanned by the set of vectors $(x, Ax, \ldots, A^{s-1}x)$ where x is a vector and s is an integer less than n, will usually be a very accurate approximation to the least eigenvalue of A itself. In Section 2.4 we will show how Lanczos' method can be used to compute the eigenvalues and eigenvectors of the restricted operator described above. In Sections 2.5 and 2.6, we will extend this basic idea by replacing the vector x with a matrix X. The basis of our algorithm will be a Block Lanczos method which is an extension of an algorithm originally proposed by Lanczos. In Section 2.7 we will develop a Block Lanczos algorithm and show how it can be used to compute the eigenvalues and eigenvectors of A restricted to a space similar to the space suggested above. Finally, in Sections 2.8, 2.9, and 2.10, we will integrate our Block Lanczos method into a complete algorithm for solving the above problem.

2.1 Notation, Definitions, and Basic Results

In this section we will give the notation and basic definitions and lemmas which will be used elsewhere in this chapter.

If $x_1, x_2, \ldots, x_m$ are m vectors of order n, then

$$X = (x_1, x_2, \ldots, x_m)$$

will mean that X is the n-by-n matrix whose j-th column is x_j. Similarly, if $X_1, X_2, \ldots, X_m$ are n-by-p matrices, then

$$X = (X_1, X_2, \ldots, X_m),$$

will mean that X is an n-by-$p \times m$ matrix whose first p columns are X_1, whose second p columns are X_2, etc.

If $X = (x_1, x_2, \ldots, x_m)$, then

$$Sp(x_1, x_2, \ldots, x_m) \quad \text{or} \quad Sp(X)$$

will denote the subspace spanned by the columns of X.

If $\lambda_1, \lambda_2, \ldots, \lambda_m$ are scalars, then

$$diag(\lambda_1, \lambda_2, \ldots, \lambda_m)$$

will stand for the diagonal matrix of order m whose j-th diagonal element is λ_j.

Let p be a polynomial of degree m. Let c_j be the coefficient of λ^j in the expansion of $p(\lambda)$ in powers of λ. That is,

$$p(\lambda) = c_0 + c_1\lambda + \ldots + c_m\lambda^m .$$

For any matrix A, $p(A)$ is a matrix defined as follows:

$$p(A) = c_0 I + c_1 A + \ldots + c_m A^m .$$

Note that if x is a vector, then

$$p(A)x = c_0 x + c_1 Ax + \ldots + c_m A^m x .$$

Furthermore, if $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$, then

$$p(\Lambda) = diag(p(\lambda_1), p(\lambda_2), \ldots, p(\lambda_n)) .$$

Let A be a symmetric matrix of order n with eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_n$ and orthonormal eigenvectors $q_1,q_2,\ldots,q_n$, and let

$$Q = (q_1,\ldots,q_n)$$

and

$$\Lambda = \operatorname{diag}(\lambda_1,\ldots,\lambda_n) \quad .$$

By definition,

$$AQ = Q\Lambda \quad .$$

If p is the polynomial defined above, then it can be shown that

$$p(A)Q = Qp(\Lambda) \quad .$$

Lemma 2.1.1. Let A be a symmetric matrix of order n with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$; then

$$\lambda_k = \min_{\{y_1,\ldots,y_{n-k}\}} \max_{\substack{y \neq \theta \\ y^t y_i = 0}} \frac{y^t A y}{y^t y}$$

where the minimum is taken over all subsets of $n-k$ vectors $\{y_1,\ldots,y_{n-k}\}$ and the maximum over all vectors y such that $y \neq \theta$ and $y^t y_i = 0$, $i = 1,\ldots,n-k$. Similarly, we have

$$\lambda_{n-k+1} = \max_{\{y_1,\ldots,y_{n-k}\}} \min_{\substack{y \neq \theta \\ y^t y_i = 0}} \frac{y^t A y}{y^t y} \quad .$$

Proof. This is the Courant-Fischer theorem. For a discussion and proof, see Wilkinson [23], pp. 98-101.

For our purposes, we restate this result as follows.

Lemma 2.1.2. Let A be a symmetric matrix of order n with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$; then

$$\lambda_k = \min_{E_k} \max_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y}$$

where the minimum is taken over all subspaces E_k of R_n of dimension at least k and the maximum over all non-zero vectors y in E_k. Similarly,

$$\lambda_{n-k+1} = \max_{E_k} \min_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y} \quad .$$

Proof. This is a direct consequence of the previous theorem.

Let S be a subspace of R_n of dimension m. The projection matrix for S, denoted by P_S, is defined to be that matrix such that for any vector $x \in R_n$, $y = P_S x \in S$ and $y^t(x-y) = 0$.

Intuitively $P_S x$ is the vector in S which is closest to x if the vector norm $\|\cdot\|_2$ is used to measure distance. Note that for any $x \in S$, $P_S x = x$.

If Q is an orthonormal matrix whose columns form a basis for S, then

$$P_S = QQ^t \quad .$$

The projection operator onto the space orthgonal to S, denoted by $P_S^{\perp}$, is given by

$$P_S^{\perp} = I - P_S = I - QQ^t \quad .$$

If x is a vector, the Euclidean norm $\|x\|_2$ of x is defined as follows:

$$\|x\|_2 = (x^t x)^{\frac{1}{2}} \quad .$$

We will usually omit the subscript and write simply $\|x\|$.

If A is a matrix, then $\|A\|_2$ or $\|A\|$ denotes the spectral norm of A induced by the Euclidean norm. That is,

$$\|A\|_2 = \max_{y \neq \theta} \frac{\|Ay\|}{\|y\|} \quad .$$

It is easy to show that

$$\|A\|_2 = \lambda_{max}^{\frac{1}{2}}(A^t A)$$

where $\lambda_{max}(A^t A)$ is the largest eigenvalue of $A^t A$.

The Frobenius norm $\|A\|_F$ of a matrix A of order n is defined as follows:

$$\|A\|_F = \left(\sum_{i=1}^{n} a_i^t a_i \right)^{\frac{1}{2}}$$

where a_i is the i-th column of A .

The singular values of a matrix A are the square roots of the eigenvalues of $A^t A$. That is,

$$\sigma_i = \lambda_i^{\frac{1}{2}}(A^t A) \quad , \quad i = 1, \ldots, n \quad ,$$

where λ_i is an eigenvalue of $A^t A$. Note then that

$$\|A\|_2 = \sigma_{max}(A)$$

where $\sigma_{max}(A)$ is the largest singular value of A .

If A is a symmetric matrix and

$$\|Ax - \mu x\| = \epsilon$$

where μ is a scalar and x is a vector with $\|x\| = 1$, then Weinstein's inequality [23] states that there is an eigenvalue λ of A such that

$$|\lambda - \mu| \leq \epsilon \quad .$$

2.2 Restricted Operators

Let A be a symmetric matrix of order n which maps the real n-dimensional Euclidean vector space R_n into R_n. Let S be an m-dimensional subspace of R_n where $m \leq n$.

Definition 2.2.1. The restriction of A to S, denoted by $\bar{A}$, is a linear operator (matrix) which maps S onto S as follows: For any vector $x \epsilon S$,

$$\bar{A}x = P_S A x$$

where P_S is the projection matrix onto S.

Let Q be an n-by-m orthonormal matrix whose columns are a basis for S; then

$$P_S = QQ^t$$

and for any $x \epsilon S$,

$$\begin{aligned} \bar{A}x &= P_S A x \\ &= P_S A P_S x \quad \text{since } x = P_S x \\ &= (QQ^t)A(QQ^t)x \\ &= QBv \end{aligned}$$

where

$$B = Q^t A Q$$

and

$$v = Q^t x \quad .$$

B is a symmetric matrix of order m and is essentially the matrix representation of $\bar{A}$. Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_m$ be the eigenvalues of B with eigenvectors $v_1, v_2, \ldots, v_m$. Let

$$\bar{q}_i = Q v_i \quad , \quad i = 1,2,\ldots,m \quad .$$

It follows that μ_i and $\bar{q}_i$ are an eigenvalue and eigenvector, respectively, of $\bar{A}$ for $i = 1,2,\ldots,m$. This can be seen as follows:

$$\bar{A}\bar{q}_i = QBv_i \quad \text{since} \quad v_i = Q^t \bar{q}_i \quad ,$$

$$= \mu_i Q v_i \quad \text{since} \quad Bv_i = \mu_i v_i \quad ,$$

$$= \mu_i \bar{q}_i \quad .$$

Thus,

$$\bar{A}\bar{q}_i = \mu_i \bar{q}_i \quad , \quad i = 1,2,\ldots,m \quad .$$

$\bar{A}$ therefore has m eigenvalues and eigenvectors which can be computed using Q and B . It can also be shown that if S is an invariant subspace of A , then the eigenvalues and eigenvectors of $\bar{A}$ will also be eigenvalues and eigenvectors of A .

By Lemma 2.1.2, we have

$$\mu_k = \min_{E_k} \max_{\substack{y \neq 0 \\ y \in E_k}} \frac{y^t B y}{y^t y}$$

for $k = 1,2,\ldots,m$, where the minimum is taken over all subspaces of R_m

is dimension at least k. By observing that

$$\frac{y^t B y}{y^t y} = \frac{y^t Q^t A Q y}{y^t Q^t Q y} = \frac{(Qy)^t A (Qy)}{(Qy)^t (Qy)}$$

and that for any subspace E_k of R_m, the set of vectors

$$\{z \mid z = Qy,\ y \in E_k\}$$

is a subspace of S of the same dimension, we have the following result.

Lemma 2.2.1. For $k = 1,2,\ldots,m$,

$$\mu_k = \min_{E_k} \max_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y} \tag{2.2.1}$$

where the minimum is taken over all subspaces E_k of dimension at least k of S and the maximum over all non-zero vectors y in E_k. Similarly, we have for $k = 1,2,\ldots,m$,

$$\mu_{m-k+1} = \max_{E_k} \min_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y} . \tag{2.2.2}$$

Proof. This result is a straightforward application of Lemma 2.1.2.

In Equation 2.2.1, the minimum is achieved when

$$E_k = Sp(q_1,\ldots,q_k) ,$$

where q_i is the i-th eigenvector of A, and the maximum in Equation 2.2.2 when

$$E_k = Sp(q_{m+k-1},\ldots,q_m) .$$

Combining this observation with Lemma 2.1.3 gives us the following result.

Lemma 2.2.2. For $k = 1,2,\ldots,m$,

$$\lambda_k \leq \mu_k \leq \lambda_{n-m+k} \quad .$$

A simple consequence of Lemma 2.2.1 is the following.

Lemma 2.2.3. Let μ_1 be the least eigenvalue of A restricted to a subspace S ; then

$$\mu_1 = \min_{\substack{y \neq \theta \\ y \in S}} \frac{y^t A y}{y^t y}$$

where the minimum is taken over all non-zero vectors y in S .

2.3 The Basic Idea

Let A be a symmetric matrix of order n and let x be a given vector.

Definition 2.3.1. The Krylov sequence of x with respect to A is the sequence of vectors

$$x, Ax, A^2x, \ldots \quad .$$

For any s greater than zero, we will denote by $K(s,x,A)$ the subspace spanned by the first s elements of the above sequence. That is,

$$K(s,x,A) = Sp(x,Ax,\ldots,A^{s-1}x) \quad .$$

Kaniel [11] showed that if we consider $\bar{A}$, the restriction of A to $K(s,x,A)$ for a relatively small value of s , then a few of the

least (and greatest) eigenvalues of $\bar{A}$ will usually be good approximations to the eigenvalues of A and showed that they decrease rapidly as s increases.

The phenomenon described in the last paragraph is the basic idea behind our algorithm. In the next section we will describe and discuss Lanczos' method and show how it can be used to compute the eigenvalues and eigenvectors of $\bar{A}$. We will see that for the relatively small cost of computing the eigenvalues and eigenvectors of $\bar{A}$, we often obtain remarkably accurate approximations to some of the eigenvalues and eigenvalues of A. From the standpoint of large sparse matrices, this approach will prove to be particularly effective since no transformation of A itself is required.

Before proceeding, however, the basic result of Kaniel concerning the least eigenvalue of $\bar{A}$ will be stated and its proof reviewed to provide some intuitive background for these ideas.

Note: Some of Kaniel's results were incorrect as stated in his paper. Paige [17] redeveloped this theory, correcting the errors in the process. It is essentially Paige's result which is stated here.

Theorem 2.3.1. Let A be a symmetric matrix of order n, and let x be a vector such that $\|x\| = 1$. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of A with corresponding orthonormal eigenvectors $q_1, \ldots, q_n$. Let s be an integer greater than zero and less than n. Suppose that $\lambda_1 < \lambda_2$ and

$$b_1 \equiv q_1^t x = \cos \theta \neq 0 ,$$

where θ is the angle between q and x; then μ_1, the least eigenvalue of $\bar{A}$, the restriction of A to the subspace $K(s,x,A)$ satisfies

$$\lambda_1 \le \mu_1 \le \lambda_1 + \epsilon_1^2$$

where

$$\epsilon_1^2 = \frac{(\lambda_n - \lambda_1)\tan^2\theta}{T_{s-1}^2\left(\frac{1+\gamma}{1-\gamma}\right)} \quad ,$$

T_{s-1} is the $(s-1)$-st Chebyshev polynomial of the first kind, and

$$\gamma = \frac{(\lambda_2 - \lambda_1)}{(\lambda_n - \lambda_1)} \quad .$$

Example. Suppose $n = 500$, $\lambda_1 = 0.0$, $\lambda_2 = 0.10$, $\lambda_{500} = 1.00$, $s = 20$, and x is such that $b_1 = 0.04$. We then have

$$\tan^2\theta = (1 - b_1^2)/b_1^2 = 624.0 \quad ;$$

$$\gamma = .10/1.00 = .10 \quad ,$$

$$\frac{1+\gamma}{1-\gamma} \doteq 1.222 \quad , \text{ and}$$

$$T_{19}\left(\frac{1+\gamma}{1-\gamma}\right) \doteq 1.27 \times 10^5 \quad .$$

Thus,

$$\epsilon_1^2 = \frac{1.00 \times 624}{(1.27 \times 10^5)^2} = 4 \times 10^{-8} \quad ,$$

and

$$\lambda_1 \le \mu_1 \le \lambda_1 + .0000004$$

implying that μ_1 is accurate to at least seven significant digits. The above bound is an overestimate and if we computed μ_1 (using the Lanczos method, say,) it would actually be far more accurate than the bound indicates.

Indication of proof. We will only outline the proof here. For the details, refer to Paige [17], pp. 44-51.

We know by Lemmas 2.2.2 and 2.2.3 that

$$\lambda_1 \leq \mu_1 \leq g^t Ag / g^t g$$

for any non-zero g in $K(s,x,A)$. Our strategy is to pick a vector g in $K(s,x,A)$ for which

$$g^t Ag / g^t g \leq \lambda_1 + \epsilon_1^2 \quad ,$$

where ϵ_1^2 is as given in the statement of the theorem. Once we have established this result, the theorem is proved.

Choose g as follows: Let c be a polynomial such that

$$c(\lambda) = T_{s-1}(z)$$

where T_{s-1} is the $(s-1)$-st Chebyshev polynomial of the first kind and for any λ,

$$z = 1 - 2\frac{(\lambda - \lambda_2)}{(\lambda_n - \lambda_2)} \quad .$$

Note that by the properties of the Chebyshev polynomials,

$$|c(\lambda_i)| \leq 1 \quad \text{for} \quad i = 2,3,\ldots,n \text{ , and}$$

$$c(\lambda_1) = T_{s-1}\left(\frac{1+\gamma}{1-\gamma}\right) > 1$$

where γ is as defined in the theorem. We now let

$$g = c(A)x \quad .$$

Since c is of degree $s-1$, g is a linear combination of the vectors $x, Ax, A^2x, \ldots, A^{s-1}x$ and thus is contained in $K(s,x,A)$. Furthermore, if we let $b \equiv Q^t x$, then $x = Qb$ and

$$
\begin{aligned}
g &= c(A)x \\
&= c(A)Qb \\
&= Qc(\Lambda)b \quad \text{where } \Lambda = \operatorname{diag}(\lambda_1,\ldots,\lambda_n) \ , \\
&= b_1 c(\lambda_1)q_1 + b_2 c(\lambda_2)q_2 + \ldots + b_n c(\lambda_n)q_n
\end{aligned}
$$

where b_i is the i-th component of b. Note that in comparison to x, the component of q_1 corresponding to λ_1 in g has been amplified while the components of the other eigenvectors have been decreased. If we now form the Rayleigh quotient g^tAg/g^tg, we will find after some algebraic manipulation that

$$g^tAg/g^tg \leq \lambda_1 + \epsilon_1^2$$

which establishes the theorem.

2.4 Lanczos' Algorithm for Symmetric Matrices

Let A be a symmetric matrix of order n and let x be a vector. Let m be the first value for which the vectors

$$x, Ax, A^2x, \ldots, A^mx$$

are dependent. Since each of these vectors is of order n, it must be the case that $m \leq n$. Furthermore, since m is the first value for which the above vectors are dependent, A^mx must be a linear combination of the vectors $x, Ax, \ldots, A^{m-1}x$. That is,

$$A^mx = c_0x + c_1Ax + \ldots + c_{m-1}A^{m-1}x \tag{2.4.1}$$

for some scalar values $c_0, c_1, \ldots, c_{m-1}$. Denote by $P_{x;A}$ the polynomial

$$P_{x;A}(\lambda) = \lambda^m - c_{m-1}\lambda^{m-1} - \ldots - c_1\lambda - c_0 \ .$$

Note that by Equation (2.4.1),

$$P_{x;A}(A)x = A^m x - c_{m-1}A^{m-1}x - \ldots - c_1 Ax - c_0 x = \theta \quad .$$

Definition 2.4.1. $P_{x;A}$ is the minimum polynomial of x with respect to A.

It can be shown that the zeroes of $P_{x;A}$ are eigenvalues of A.

In 1950, Lanczos published a paper [13] on computing solutions to the eigenproblem which contained a description of an algorithm for computing $P_{x;A}$. His approach, although very attractive at first glance, presented some numerical problems in implementation and application (cf. Section 3.1) and with the development of the Givens and Householder methods [23], was soon set aside as a method of general application.

In recent years, however, some researchers, notably C. Paige of McGill University and G. Golub of Stanford University, have proposed that Lanczos' method be used as means of computing solutions to the symmetric eigenproblem when the matrix is of large order and sparse for the following reasons: (1) Many methods such as Householder's method and the QR method, carry out similarity transformations of the matrix. Such transformations generally destroy sparse structure. By contrast, Lanczos' method does not transform the matrix and, therefore, any sparse structure can be preserved throughout the application of the algorithm. In particular, the only way in which the matrix A is used in Lanczos' method is in computing the product Ay given a vector y, and if A is sparse, even though of large order, this multiplication can generally be accomplished efficiently. (2) Although originally intended to be

used to compute the minimum polynomial of a vector, Lanczos' method can be used to achieve other ends. As we will soon see, it can be used as means of computing the eigenvalues and eigenvectors of A restricted to the space spanned by $\{x, Ax, A^2x, \ldots, A^{s-1}x\}$ for some s less than n. As we saw in the previous section, the least eigenvalue of this restricted operator will generally be an accurate approximation to the least eigenvalue of A itself.

We will now review Lanczos' method and some of its properties. Later on we will extend Lanczos' method and the ideas of the previous section to work with a matrix of vectors X instead of a single vector x. This generalization will afford us certain advantages computationally over the single vector approach.

The results stated here will be given without proof. For a more complete discussion of Lanczos' method, refer to the following sources: Wilkinson [23], Golub [8], Golub, Underwood and Wilkinson [7], and Paige [17,18].

Lanczos' method can take many different forms depending on the application, but for present purposes, it is as follows:

Let A be a symmetric matrix of order n. Let x be a vector of unit length $(\|x\| = 1)$.

Compute sequences of scalars $(\alpha_i)_{i=1}^{m}$ and $(\beta_i)_{i=2}^{m}$, and a sequence of <u>orthonormal</u> vectors $(x_i)_{i=1}^{m}$ as follows:

Step 1. Let $x_1 = x$ and $i = 1$.

Step 2. Compute $y_i = Ax_i$, $\alpha_i = x_i^t y_i$ $(= x_i^t A x_i)$, and z_{i+1} where

$$z_{i+1} = \begin{cases} y_1 - \alpha_1 x_1 & \text{if } i = 1, \text{ or} \\ y_i - \alpha_i x_i - \beta_i x_{i-1} & \text{if } i > 1. \end{cases}$$

Step 3. Compute $\beta_{i+1} = \|z_{i+1}\|$.

Step 4. If $\beta_{i+1} = 0$, then stop.

Step 5. Compute $x_{j+1} \quad z_{i+1}/\beta_{i+1}$.

Step 6. Increase the value of i by one and go to Step 2.

This algorithm will stop for some value of $i \leq n$. Let m be the final value of i .

As we will see, Lanczos' method is not a method for computing eigenvalues and eigenvectors per se. Rather it is a way of transforming the eigenproblem into a problem in a different form and it must be combined with an algorithm to solve the second problem to produce a complete method for computing eigenvalues and eigenvectors. For example, Lanczos used the sequences $(\alpha_i)_{i=1}^m$ and $(\beta_i)_{i=1}^m$ to form $P_{x,A}$, the minimum polynomial of x with respect to A . Computing the zeroes of $P_{x,A}$ yielded eigenvalues of A , and once the eigenvalues had been found, Lanczos showed how the x_i could be combined to form eigenvectors. The more modern viewpoint is that Lanczos' method is a way of transforming a general symmetric matrix into a symmetric tridiagonal matrix T . The eigenproblem for T can then be solved in a variety of ways, e.g. the QR method or a bisection method based on Sturm sequences [23], and the resulting solution can be used to find the eigenvalues and eigenvectors of A .

Furthermore, there is a practical difficulty with Lanczos' method as described above. Although the sequence of vectors $x_1,\ldots,x_s$ generated by the above algorithm in exact arithmetic will be orthonormal, in practice they will generally lose orthogonality after a few steps of

the algorithm have been carried out. The main cause of this phenomenon is the loss of accuracy caused by cancellation when z_{i+1} is computed in Step 2. Since for many applications of the method, this is a serious source of error, it is usually modified so that after Step 5, x_{i+1} is reorthogonalized with respect to $x_1, x_2, \ldots, x_i$ and then renormalized. Since reorthogonalizing x_{i+1} is such a time consuming operation, it was this shortcoming that originally caused many to disregard Lanczos' method. From the standpoint of the way we intend to use Lanczos' method, i.e., as a means of computing a few of the least eigenvalues and eigenvectors of a large, sparse symmetric matrix, it is still a relatively efficient method even if a reorthogonalization step is included.
C. Paige has suggested [17] that reorthogonalization is unnecessary if Lanczos' method is used as we intend to use it. He argues that, rather than being a liability, loss of orthogonality is actually a blessing in disguise since it is indicative of convergence of some of the eigenvalues of the restricted operator to eigenvalues of the matrix A. We will discuss this issue further in Section 3.1. For the time being, we will ignore this aspect of the algorithm and deal with its theoretical properties. For this purpose, the above description of the algorithm is adequate.

To begin with, observe that the α_i, β_i, and x_i satisfy the following equations:

$$\beta_2 x_2 = z_2 = Ax_1 - \alpha_1 x_1 ,$$

$$\beta_3 x_3 = z_3 = Ax_2 - \alpha_2 x_2 - \beta_2 x_1 ,$$

$$\vdots$$

$$\beta_m x_m = z_m = Ax_{m-1} - \alpha_{m-1} x_{m-1} - \beta_{m-1} x_{m-2} ,$$

$$0 = Ax_m - \alpha_m x_m - \beta_m x_{m-1} .$$

We can rewrite these equations as follows:

$$\begin{aligned} Ax_1 &= \alpha_1 x_1 + \beta_2 x_2 \\ Ax_2 &= \beta_2 x_1 + \alpha_2 x_2 + \beta_3 x_3 \\ &\vdots \\ Ax_k &= \beta_k x_{k-1} + \alpha_k x_k + z_{k+1} \end{aligned} \qquad (2.4.2)$$

for any k between one and m where $z_{m+1} = 0$. Define

$$X_k \equiv (x_1, x_2, \ldots, x_k)$$

and

$$M_k \equiv \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \beta_k & \alpha_k \end{bmatrix}$$

for $k = 1,2,\ldots,m$. (M_k is a symmetric tridiagonal matrix with $\alpha_1,\ldots,\alpha_k$ along its diagonal and $\beta_2,\ldots,\beta_k$ along its off-diagonals.) Using this notation we can write Equations (2.4.2) as

$$AX_k = X_k M_k + (0,0,\ldots,z_{k+1}) \qquad (2.4.4)$$

where the last matrix is n-by-k with zeroes in its first k-1 columns and z_{k+1} in its last column. In particular, for $k = m$,

$$AX_m = X_m M_m .$$

From this equation we see that X_m spans an invariant subspace of A. Therefore, the eigenvalues of M_m are eigenvalues of A and if v is an eigenvector of M_m, then $X_m v$ is an eigenvector of A (cf. Section 2.2.)

Also, by Equation (2.4.3), we have

$$X_k^t A X_k = M_k$$

since $z_{k+1}^t X_k = 0$. Referring to Section 2.2, we conclude that M_k is the representation of the matrix A restricted to the space spanned by the columns of X_k. Furthermore, we can show that x_k is a linear combination of the vectors $x, Ax, A^2x, \ldots, A^{k-1}x$, for $k = 1,2,\ldots,m$. Therefore, for $k = 1,2,\ldots,m$, the columns of X_k form an orthonormal basis for the space $K(k,x,A)$ spanned by the vectors $x, Ax, A^2x, \ldots, A^{k-1}x$, and M_k is the representation of A restricted to $K(k,x,A)$. The work of Kaniel and Paige suggest that for relatively small values of k, the least (and greatest) eigenvalues of M_k will usually be very good approximations to the least (and greatest) eigenvalues of A. Computational experience verifies this idea. See, for example, Paige [17] and Godunov and Prokopov [6].

This suggests that instead of carrying out the algorithm until $\beta_{n+1} = 0$ as described before, we stop after a fixed number of steps, say s steps, and use the resulting matrices M_s and X_s to compute

accurate approximations to some of the eigenvalues and eigenvectors of A. For this use, Lanczos' method can be described as follows.

Let A be a symmetric matrix of order n and let x be a vector such that $\|x\| = 1$. Let s be an integer greater than one and less than or equal to n. Compute sequences $(\alpha_i)_{i=1}^{s}$, $(\beta_i)_{i=2}^{s}$, and $(x_i)_{i=1}^{s}$ as follows:

Step 1. Let $x_1 = x$ and $i = 1$.

Step 2a. Compute $y_i = Ax_i$ and $\alpha_i = x_i^t y_i$.

Step 2b. If $i = s$, stop.

Step 2c. Compute z_{i+1} as before.

Step 3. Compute $\beta_{i+1} = \|z_{i+1}\|$.

Step 4. If $\beta_{i+1} = 0$, stop.

Step 5. Compute $x_{i+1} = z_{i+1}/\beta_{i+1}$.

Step 6. Increase i by one and go to Step 2.

If the final value of i is less than s, we decrease s to this value.

Note: If μ and v are an eigenvalue and eigenvector, respectively, of M_k and if we define

$$\bar{q} = M_k v ,$$

then Equation (2.4.3) implies that

$$A\bar{q} - \mu\bar{q} = v_k z_{k+1} \qquad (2.4.4)$$

where v_k is the k-th component of v. For the extreme eigenvalues of M_k, the corresponding v_k's are often extremely small regardless

of the magnitude of z_{k+1} which serves to explain partly why the extreme eigenvalues of M_k are often very good approximations to the eigenvalues of A. Equation (2.4.4) can in fact be used to estimate the error in eigenvalues and eigenvectors computed using the Lanczos method. We will develop a similar formula for our Block Lanczos algorithm.

While an efficient and viable algorithm for computing eigenvalues and eigenvectors could be built around Lanczos' method as described above, preliminary experiments by this author indicated, however, that some advantages could be gained by extending the ideas of the last two sections to work with a matrix of vectors X instead of a single vector x as above. In particular, these experiments indicated that less work overall was required if we iterated with a block of vectors rather than a single vector. Furthermore, with the standard Lanczos method, at most one eigenvalue and vector corresponding to a multiple eigenvalue can be computed at a time. This shortcoming is overcome partly or wholely by working with several vectors simultaneously.

For this reason, we will move on at this point to the development of a Block Lanczos method.

2.5 Extending the Basic Idea

Let A be a symmetric matrix of order n and let x be a vector of unit length. In the last two sections we saw that the least eigenvalues and eigenvectors of $\bar{A}$, the restriction of A to the space spanned by the vectors $(x, Ax, \ldots, A^{s-1}x)$ where s is an integer value such that $1 \leq s \leq n$, were usually good approximations to the least eigenvalues

and eigenvectors of A. We also saw how the Lanczos method could be used to compute the eigenvalues and eigenvectors of $\bar{A}$. In this and the next two sections we will extend these ideas so that instead of working with a single vector x, we will work with an orthonormal matrix X. This generalization will allow us to compute several eigenvalues and eigenvectors simultaneously and will lead us to an algorithm for computing solutions to the symmetric eigenproblem which will require a fewer number of operations overall when compared with an algorithm based on a single vector approach. With this extended approach, we will also be able to compute multiple eigenvalues and eigenvectors at the same time.

In the remainder of this section, we will outline this idea and establish basic definitions and notation.

Let A be defined as above and let p and s be integer values such that $s \geq 1$, $p \geq 1$, and $1 \leq p \times s \leq n$. Let X be an n-by-p orthonormal matrix.

Definition 2.5.1. Let $K(s,X,A)$ be the space spanned by the $p \times s$ columns of the matrices $X, AX, \ldots, A^{s-1}X$.

If the set of vectors comprised of the columns of the matrices $X, AX, \ldots, A^{s-1}X$ is independent, then the dimension of $K(s,X,A)$ will be $p \times s$. Otherwise, it will be less than $p \times s$.

We now redefine $\bar{A}$.

Definition 2.5.2. Let $\bar{A}$ denote the restriction of A to a subspace $L(s,X,A)$ of dimension $p \times s$ containing $K(s,X,A)$.

$L(s,X,A)$ will be determined by means of a Block Lanczos algorithm to be described in Section 2.7. For the moment it is important to know only that $L(s,X,A)$ contains the columns of the matrices $X, AX, \ldots, A^{s-1}X$.

We now proceed as before. Let χ_s be an n-by-pxs orthonormal matrix whose columns form a basis for $L(s,X,A)$. Let

$$\mathcal{M}_s = \chi_s^t A \chi_s \quad .$$

$\mathcal{M}_s$ is the matrix representation of $\bar{A}$. Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{pxs}$ and $y_1, y_2, \ldots, y_{pxs}$ be the eigenvalues and eigenvectors respectively of $\mathcal{M}_s$. Let

$$\bar{q}_i = \chi_s y_i$$

for $i = 1,2,\ldots,pxs$. It follows that (cf. Section 2.2) μ_i and $\bar{q}_i$ are an eigenvalue and eigenvector respectively of $\bar{A}$ for $i = 1,2,\ldots,pxs$.

In the next section we will show that the p least eigenvalues of $\bar{A}$ will usually be accurate approximations to the p least eigenvalues of A and give bounds on the errors. In Section 2.7 we will describe a Block Lanczos algorithm which can be used to compute $\mathcal{M}_s$ and χ_s. The results of this and the next two sections indicate that the least eigenvalues of $\mathcal{M}_s$ will be accurate approximations to the eigenvalues of A. We will base our algorithm on this idea.

2.6 The Error in the Least Eigenvalues of A Restricted to L(s,X,A)

Let A be a symmetric matrix of order n. Let p and s be integer values such that $s \geq 1$, $p \geq 1$, and $1 \leq pxs \leq n$. Let X be an n-by-p orthonormal matrix.

Let $\bar{A}$ be the matrix A restricted to a space $L(s,X,A)$ of dimension pxs containing $K(s,X,A)$.

In this section we will give bounds on the errors in the p least eigenvalues of $\bar{A}$ as approximations to the p least eigenvalues of A. The bounds will be stated as a theorem and derived in the course of a proof of the theorem.

First, however, we will establish some lemmas which will be used in the proof of the theorem.

Lemma 2.6.1. Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{pxs}$ be the eigenvalues of $\bar{A}$; then

$$\mu_k \leq \max_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y}$$

where E_k is any k-dimensional subspace of $L(s,X,A)$ and the maximum is taken over all non-null vectors y in E_k.

Proof. This lemma is a direct consequence of Lemma 2.2.1.

Lemma 2.6.2. Let E_k be a subspace of $L(s,X,A)$ of dimension k. Let G_k be an n-by-k matrix whose columns form a basis for E_k; then

$$\max_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y} = \lambda_k'$$

where λ_k' is the largest eigenvalue of the generalized eigenproblem

$$(G_k^t A G_k)y = \lambda(G_k^t G_k)y \quad .$$

Proof. In general it can be shown that if C and F are symmetric matrices of order k and F is positive definite, then

$$\max_{\substack{z \neq \theta \\ z \in R_k}} \frac{z^t C z}{z^t F z}$$

is equal to the largest eigenvalue of the generalized eigenproblem

$$Cz = \lambda Fz \quad .$$

Observe now that any vector y in E_k can be written

$$y = G_k z \quad ,$$

where z is in R_k. Therefore,

$$\max_{\substack{y \neq \theta \\ y \in E_k}} \frac{y^t A y}{y^t y} = \max_{\substack{z \neq \theta \\ z \in R_k}} \frac{z^t G_k^t A G_k z}{z^t G_k^t G_k z}$$

and the lemma follows directly from this equation.

Lemma 2.6.3. Let $\nu_1 \leq \nu_2 \leq \cdots \leq \nu_k$ be the eigenvalues of the generalized eigenproblem

$$Cz = \nu Fz \tag{2.6.1}$$

where C and F are symmetric matrices of order k and F is positive definite. Then $\nu_1-\sigma, \nu_2-\sigma, \ldots, \nu_k-\sigma$ are the eigenvalues of

$$(C - \sigma F)z = \nu Fz \tag{2.6.2}$$

for any real σ.

Proof. Subtract σFz from both sides of Equation (2.6.1) and we have

$$(C - \sigma F)z = (\nu - \sigma)Fz \quad .$$

Thus, if ν is an eigenvalue of Equation (2.6.1), then $\nu-\sigma$ is an eigenvalue of Equation (2.6.2).

Lemma 2.6.4. Let C and F be symmetric matrices of order n. Suppose C is negative semi-definite and F is positive definite. Let E be a symmetric matrix. The largest eigenvalue λ' of

$$(C+E)z = \lambda Fz \tag{2.6.3}$$

satisfies

$$\lambda' \leq \|F^{-1}\| \cdot \|E\| \quad .$$

Proof. Let S be the Cholesky factor of F [23]. The eigenvalues of the generalized eigenproblem (2.6.3) are the same as the eigenvalues of the standard problem

$$S^{-1}(C+E)S^{-t}w = \lambda w \quad , \tag{2.6.4}$$

where

$$SS^t = F \quad \text{and} \quad w = S^t z \quad .$$

Note that

$$S^{-1}(C+E)S^{-t} = S^{-1}CS^{-t} + S^{-1}ES^{-t} \quad .$$

Since C is negative semi-definite, $S^{-1}CS^{-t}$ must also be negative semi-definite and all of its eigenvalues must be less than or equal to zero. By Weinstein's inequality, the eigenvalues of Equation (2.6.4) can differ from those of $S^{-1}CS^{-t}$ by quantities which are bounded by $\|S^{-1}ES^{-t}\|$. Thus the largest eigenvalue of Equation (2.6.3) must satisfy

$$\lambda' \leq \|S^{-1}ES^{-t}\| \quad .$$

By Lemma 2.6.5,

$$\|S^{-1}ES^{-t}\| \leq \|S^{-1}S^{-t}\| \cdot \|E\| \quad .$$

Since S is the Cholesky factor of F,

$$\|S^{-1}S^{-t}\| = \|F^{-1}\| \quad ,$$

and the lemma is proved.

Lemma 2.6.5. For any matrices C and F,

$$\|F^tCF\| \leq \|F^tF\| \cdot \|C\| \quad .$$

Proof. By the general properties of matrix norms,

$$\|F^tCF\| \leq \|F^t\| \, \|C\| \cdot \|F\| \quad .$$

For the spectral norm, we also have

$$\|F\| = \|F^t\| = \|F^tF\|^{1/2}$$

and the lemma follows from the last two equations.

Note: Lemma 2.6.5 was also established by Crawford [3], but the proof given here is different.

Lemma 2.6.6. If C is an n-by-n symmetric matrix and C_k is the leading k-by-k principal submatrix of C, then

$$\|C_k\| \leq \|C\|$$

for $k = 1,2,\ldots,n$.

Proof. For any symmetric matrix F, say, $\|F\| = \max_i|\lambda_i(F)|$. The lemma follows from the fact that the eigenvalues of C_k must lie within the interval containing the eigenvalues of C [ ].

Lemma 2.6.7. If $D = \text{diag}(d_1,d_2,\ldots,d_n)$, then

$$\|D\| = \max_i|d_i| \quad .$$

Proof.

$$\|D\| = \max_i|\lambda_i(D)| = \max_i|d_i|$$

Lemma 2.6.8. Let E be a positive semi-definite symmetric matrix; then

$$\|(I+E)^{-1}\| \leq 1 \quad .$$

Proof. This lemma follows from the observation that all the eigenvalues of $(I+E)$ are greater than or equal to one, and therefore, all the eigenvalues of $(I+E)^{-1}$ are greater than zero and less than or equal to one.

Lemma 2.6.9. Let W be an n-by-p orthonormal matrix. Let

$$W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$$

where W_1 and W_2 are composed of the first p and last $n-p$ rows of W, respectively. Let σ_{min} be the least singular value of W_1. If $\sigma_{min} > 0$, then

$$\|W_1^{-t} W_2^t W_2 W_1^{-1}\| = \frac{1}{\sigma_{min}^2} - 1 \quad .$$

Proof. Since W is orthonormal,

$$W_1^t W_1 + W_2^t W_2 = I \quad .$$

Since $\sigma_{min} > 0$, W_1^{-1} exists. Therefore, after multiplying the last equation by W_1^{-t} on the left and by W_1^{-1} on the right, and then rearranging terms, we have

$$W_1^{-t} W_2^t W_2 W_1^{-1} = W_1^{-t} W_1^{-1} - I \quad .$$

Since the largest eigenvalue of $(W_1^{-t}W_1^{-1}) = (W_1W_1^t)^{-1}$ is $1/\sigma_{min}^2$,

$$\|W_1^{-t}W_1^{-1} - I\| = \frac{1}{\sigma_{min}^2} - 1 \quad ,$$

and the lemma follows from the last two equations.

We will now state and prove the theorem giving the bounds on the errors.

Theorem 2.6.1. Let A be a symmetric matrix of order n with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and orthonormal eigenvectors $q_1, q_2, \ldots, q_n$. Let p and s be integer values such that $p > 0$, $s > 0$, and $1 \leq p \times s \leq n$. Assume that $\lambda_p < \lambda_{p+1}$. Let X be an n-by-p orthonormal matrix, and $\bar{A}$, the restriction of A to a subspace $L(s,X,A)$ of dimension $p \times s$ containing $K(s,X,A)$. Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{p \times s}$ be the eigenvalues of $\bar{A}$. Define

$$Q \equiv (q_1, q_2, \ldots, q_n)$$

and

$$W \equiv \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = Q^t X \quad ,$$

where W_1 and W_2 are composed of first p and last $n-p$ rows of W, respectively. Let σ_{min} be the smallest singular value of W_1.

If $\sigma_{min} > 0$, then for $k = 1,2,\ldots,p$,

$$\lambda_k \leq \mu_k \leq \lambda_k + \epsilon_k^2 \quad ,$$

where

$$\epsilon_k^2 = (\lambda_n - \lambda_k) \frac{\tan^2 \theta}{T_{s-1}^2\left(\frac{1+\gamma_k}{1-\gamma_k}\right)} \quad ,$$

$$\theta = \arccos \sigma_{min} \quad ,$$

$$\gamma_k = (\lambda_k - \lambda_{p+1})/(\lambda_k - \lambda_n) \quad , \text{ and}$$

$$T_{s-1} = (s-1)\text{-st Chebyshev polynomial of the first kind.}$$

Proof. We will show that there are p vectors $g_1, g_2, \ldots, g_p$ in $L(s,X,A)$ such that if $E_k = Sp(g_1, \ldots, g_k)$, then

$$\lambda_k' \leq \max_{\substack{y \in E_k \\ y \neq 0}} \frac{y^t A y}{y^t y} \leq \lambda_k + \epsilon_k^2 \quad . \tag{2.6.5}$$

By Lemma 2.2.2 and Lemma 2.6.1,

$$\lambda_k \leq \mu_k \leq \lambda_k' \quad . \tag{2.6.6}$$

Combining (2.6.5) and (2.6.6) will complete the proof of our theorem.

Let P be the polynomial such that

$$P(\lambda) = T_{s-1}(z)$$

where

$$z = 1 - 2 \frac{(\lambda_{p+1} - \lambda)}{(\lambda_{p+1} - \lambda_n)} \quad ,$$

and T_{s-1} is the $(s-1)$-st Chebyshev polynomial of the first kind. Note that, by the properties of Chebyshev polynomials,

$$|P(\lambda_i)| \leq 1 \tag{2.6.7}$$

for $i = p+1,\ldots,n$, and

$$P(\lambda_1) \geq P(\lambda_2) \geq \cdots \geq P(\lambda_p) > 1 \quad . \qquad (2.c.3)$$

Let $c_0, c_1, \ldots, c_{s-1}$ be the coefficients in the expansion of $P(\lambda)$ in powers of λ . That is,

$$P(\lambda) = c_0 + c_1\lambda + \ldots + c_{s-1}\lambda^{s-1} \quad .$$

Let

$$H = P(A)\,X = (c_0 I + c_1 A + \ldots + c_{s-1}A^{s-1})\,X \quad .$$

Note that the columns of H are linear combinations of the columns of $X, AX, \ldots, A^{s-1}X$ and hence are in $L(s,X,A)$. Since Q is the matrix of eigenvectors of A ,

$$P(A)\,Q = Q\,P(\Lambda)$$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

From the definition of W ,

$$X = QW \quad .$$

Thus,

$$\begin{aligned} H &= P(A)X \\ &= P(A)QW \\ &= QP(\Lambda)W \quad . \end{aligned}$$

Now let $\Lambda_1 = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ and $\Lambda_2 = \mathrm{diag}(\lambda_{p+1}, \ldots, \lambda_n)$. Thus,

$$P(\Lambda) = \begin{bmatrix} P(\Lambda_1) & \\ & P(\Lambda_2) \end{bmatrix} ,$$

$$P(\Lambda)W = \begin{bmatrix} P(\Lambda_1)W_1 \\ P(\Lambda_2)W_2 \end{bmatrix} ,$$

and

$$H = Q\begin{bmatrix} P(\Lambda_1)W_1 \\ P(\Lambda_2)W_2 \end{bmatrix} \quad . \tag{2.6.9}$$

Define

$$G = (g_1, g_2, \ldots, g_p) = HW_1^{-1}P(\Lambda_1)^{-1} \quad .$$

Note that $g_k \in L(s,X,A)$, $k = 1,2,\ldots,p$, since it is a linear combination of the columns of H . By Equation (2.6.9)

$$G = Q\begin{bmatrix} I \\ \Delta \end{bmatrix}$$

where

$$\Delta = (\delta_1, \delta_2, \ldots, \delta_p) = P(\Lambda_2)W_2W_1^{-1}P(\Lambda_1)^{-1} \quad .$$

Now let

$$G_k = (g_1, g_2, \ldots, g_k) \quad ,$$

$$\Delta_k = (\delta_1, \delta_2, \ldots, \delta_k) \quad , \text{ and}$$

$$E_k = Sp(G_k) \quad .$$

We now want to bound

$$\lambda_k' = \max_{\substack{y \in E_k \\ y \neq \theta}} \frac{y^tAy}{y^ty} \quad .$$

By Lemma 2.6.2, λ_k' is the largest eigenvalue of the generalized eigenproblem

$$(G_k^tAG_k)y = \lambda(G_k^tG_k)y \quad .$$

Now,

$$G_k^t A G_k = D_k + \Delta_k^t \Lambda_2 \Delta_k \quad ,$$

$$G_k^t G_k = I + \Delta_k^t \Delta_k$$

$$D_k = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k) \quad .$$

Thus, λ_k' is the largest eigenvalue of

$$(D_k + \Delta_k^t \Lambda_2 \Delta_k)y = \lambda(I + \Delta_k^t \Delta_k)y \quad .$$

By Lemma 2.6.3, the largest eigenvalue of

$$(D_k + \Delta_k^t \Lambda_2 \Delta_k - \lambda_k(I + \Delta_k^t \Delta_k))y = \lambda(I + \Delta_k^t \Delta_k)y$$

is $\lambda_k' - \lambda_k$. Observe that

$$(D_k + \Delta_k^t \Lambda_2 \Delta_k - \lambda_k(I + \Delta_k^t \Delta_k)) = ((D_k - \lambda_k I) + \Delta_k^t(\Lambda_2 - \lambda_k I)\Delta_k)$$

and that

$$(D_k - \lambda_k I) = \begin{bmatrix} \lambda_1 - \lambda_k & & & \\ & \ddots & & \\ & & & \lambda_k - \lambda_k \end{bmatrix}$$

is negative semidefinite. By Lemma 2.6.4,

$$\lambda_k' - \lambda_k \leq \|(I + \Delta_k^t \Delta_k)^{-1}\| \cdot \|\Delta_k^t(\Lambda_2 - \lambda_k I)\Delta_k\| \quad . \tag{2.6.10}$$

By Lemma 2.6.8,

$$\|(I + \Delta_k^t \Delta_k)^{-1}\| \leq 1 \quad . \tag{2.6.11}$$

By Lemmas 2.6.5 and 2.6.7,

$$\|\Delta_k^t(\Lambda_2 - \lambda_k I)\Delta_k\| \leq \|\Delta_k^t \Delta_k\| \cdot \|(\Lambda_2 - \lambda_k I)\| \leq (\lambda_n - \lambda_k)\|\Delta_k^t \Delta_k\| \quad . \tag{2.6.12}$$

Note that

$$\Delta_k = P(\Lambda_2)(W_2W_1^{-1})_k P(D_k)^{-1}$$

where $(W_2W_1^{-1})_k$ stands for the first k columns of $W_2W_1^{-1}$. Thus

$$\|\Delta_k^t \Delta_k\| = \|P(D_k)^{-1}(W_2W_1^{-1})_k^t P(\Lambda_2)^t P(\Lambda_2)(W_2W_1^{-1})_k P(D_k)^{-1}\| \quad .$$

Three applications of Lemma 2.6.5 give us

$$\|\Delta_k^t \Delta_k\| = \|P(D_k)^{-t}P(D_k)^{-1}\| \cdot \|(W_2W_1^{-1})_k^t (W_2W_1^{-1})_k\| \cdot \|P(\Lambda_2)^t P(\Lambda_2)\| \quad . \quad (2.6.13)$$

By Lemma 2.6.7 and Equations (2.6.7) and (2.6.8)

$$\|P(D_k)^{-t}P(D_k)^{-1}\| \leq \frac{1}{P(\lambda_k)^2} \quad (2.6.14)$$

and

$$\|P(\Lambda_2)^t P(\Lambda_2)\| \leq 1 \quad . \quad (2.6.15)$$

By Lemma 2.6.6,

$$\|(W_2W_1^{-1})_k^t (W_2W_1^{-1})_k\| \leq \|(W_2W_1^{-1})^t (W_2W_1^{-1})\| = \|W_1^{-t}W_2^t W_2 W_1^{-t}\| \quad .$$

Since X is orthonormal, W is orthonormal and therefore, by Lemma 2.6.9,

$$\|W_1^{-t}W_2^t W_2 W_1^{-1}\| = \frac{1}{\sigma_{min}^2} - 1 \quad . \quad (2.6.16)$$

By the definition of θ,

$$\cos\theta = \sigma_{min}$$

so that

$$\frac{1}{\sigma_{min}^2} - 1 = \frac{1-\sigma_{min}^2}{\sigma_{min}^2} = \frac{\sin^2\theta}{\cos^2\theta} = \tan^2\theta \quad . \quad (2.6.17)$$

By combining Equations (2.6.10) through (2.6.17), we get

$$\lambda_k' \leq \lambda_k + \frac{(\lambda_n - \lambda_k)}{P^2(\lambda_k)} \tan^2 \theta \quad .$$

Finally,

$$P(\lambda_k) = T_{s-1}\left(1-2 \frac{(\lambda_{p+1} - \lambda_k)}{(\lambda_{p+1} - \lambda_n)} \right)$$

and

$$1 - 2 \frac{(\lambda_{p+1} - \lambda_k)}{(\lambda_{p+1} - \lambda_n)} = \frac{\lambda_{p+1} - \lambda_n - 2\lambda_{p+1} + 2\lambda_k}{\lambda_{p+1} - \lambda_n}$$

$$= \frac{(\lambda_k - \lambda_n) + (\lambda_k - \lambda_{p+1})}{(\lambda_k - \lambda_n) - (\lambda_k - \lambda_{p+1})}$$

$$= \frac{1+\gamma_k}{1-\gamma_k} \qquad \text{where} \quad \gamma_k = \frac{\lambda_k - \lambda_{p+1}}{\lambda_k - \lambda_n} \quad .$$

Therefore,

$$\lambda_k' \leq \lambda_k + \frac{(\lambda_n - \lambda_k)}{T_{s-1}^2\left(\frac{1+\gamma_k}{1-\lambda_k}\right)} \tan^2 \theta \quad .$$

and the proof of the theorem is complete.

Example. Suppose A of order 1000 is such that $\lambda_1 = 0.0$, $\lambda_2 = 0.1$, $\lambda_3 = 0.5$ and $\lambda_{1000} = 1.0$. Suppose $s = 10$ and X is such that $\sigma_{min} = .04$; then

$$\gamma_1 = \frac{0.0 - 0.5}{0.0 - 1.0} = \frac{1}{2} \quad ,$$

$$\gamma_2 = \frac{0.1 - 0.5}{0.1 - 1.0} = \frac{4}{9}$$

$\theta = \arccos(.04)$, and

$\tan^2\theta = .924$.

We have

$$T_9\left(\frac{1+\gamma_1}{1-\gamma_1}\right) = T_9(3) = 3.9 \times 10^6 \quad , \quad \text{and}$$

$$T_9\left(\frac{1+\gamma_2}{1-\gamma_2}\right) = T_9(2.6) = .98 \times 10^6 \quad .$$

Thus, μ_1 and μ_2 satisfy,

$$\lambda_1 \leq \mu_1 \leq \lambda_1 + \frac{1.9}{1.5 \times 10^{13}} \cdot 3.9 \times 10^5 = \lambda_1 + 2.6 \times 10^{-8}$$

and

$$\lambda_2 \leq \mu_2 \leq \lambda_2 + \frac{.9}{.96 \times 10^{12}} \cdot 3.9 \times 10^5 = \lambda_2 + 3.6 \times 10^{-7} \quad .$$

The theorem and the example suggest that μ_i will tend to converge to λ_i more rapidly than μ_{i+1} to λ_{i+1}, for $i = 1,2,\ldots,p-1$. This does, in fact, occur in practice.

In the next section we will develop a Block Lanczos algorithm which can be used to compute the eigenvalues $\mu_1,\ldots,\mu_{p\times s}$ of A restricted to $L(s,X,A)$.

2.7 A Block Lanczos Algorithm

In this section we will develop an algorithm which is an extension of Lanczos' original algorithm presented in Section 2.4. Rather than start with a single vector x, we will begin with a block of vectors X and generate sequences of matrices (M_i), (R_i), and (X_i) which play

roles similar to those played by the sequences (α_i), (β_i), and (x_i), respectively, in Lanczos' method.

Lanczos developed his method as follows:

Let A be a symmetric matrix of order n. Given a vector x such that $\|x\| = 1$, compute Ax and choose α_1 such that $\|z_2\|$ is minimized where $z_2 = Ax - \alpha_1 x$. It can be shown that $\alpha_1 = x^t Ax$ and that with this choice for α_1, z_2 is orthogonal to x. Note that if $z_2 = \theta$, then x would be an eigenvector and α_1, an eigenvalue. Define $x_2 = z_2 / \|z_2\|$ and $x_1 = x$. At the j-th step, we have vectors $x_1, x_2, \ldots, x_j$ and we choose α_j and $\gamma_{1j}, \gamma_{2j}, \ldots, \gamma_{j-1,j}$ such that $\|z_{j+1}\|$ is minimized where $z_{j+1} = Ax_j - \alpha_j x_j - \gamma_{j-1,j} x_{j-1} - \cdots - \gamma_{1,j} x_1$. Lanczos showed that, in fact, $\|z_{j+1}\|$ need only be minimized with respect to α_j and $\gamma_{j-1,j}$ $(\gamma_{i,j} = 0$ for $i < j-1)$ and that with these optimal choices for α_j and $\gamma_{j-1,j}$, z_{j+1} is orthogonal to $x_1, x_2, \ldots, x_j$. If $z_{j+1} \neq \theta$, we let $x_{j+1} = z_{j+1} / \|z_{j+1}\|$. For some value m less than or equal to n, z_{m+1} will be equal to θ and the sequences $(\alpha_k)_{k=1}^m$, $(\gamma_{k-1,k})_{k=2}^m$, and $(x_k)_{k=1}^m$ can be used to compute some or all of the eigenvalues and eigenvectors of A.

Note: If we let $\beta_k = \gamma_{k-1,k}$, then the sequences generated by the above procedure are the same as those computed by the algorithm in Section 2.4.

We also saw in Section 2.4 that if we stopped the algorithm after computing α_s in Step 5, then useful information could be obtained from the sequences $(\alpha_k)_{k=1}^s$, $(\beta_k)_{k=2}^s$ and $(x_k)_{k=1}^s$. Namely, by observing

that the vectors $x_1,\ldots,x_c$ were an orthonormal basis for the space $K(c,x,A) = Sp(x,Ax,\ldots,A^{c-1}x)$ and that the $(\alpha_k)_{k=1}^{c}$ and $(\beta_k)_{k=1}^{c-1}$ could be used to obtain the matrix representation of the restriction of A to $K(c,x,A)$, we saw that accurate approximations to the least and greatest eigenvalues of A could be computed.

We will develop a Block Lanczos algorithm with a similar application in mind. Our goal is an algorithm which, starting from an n-by-p orthonormal matrix X, computes a sequence of mutually orthogonal n-by-p orthonormal matrices $X_1,X_2,\ldots,X_s$ and sequences of p-by-p matrices $M_1,M_2,\ldots,M_s$ and $R_2,R_3,\ldots,R_s$ such that the columns of

$$X_s = (X_1,X_2,\ldots,X_s) \tag{2.7.1}$$

form an orthonormal basis for a space $L(s,X,A)$ which contains the columns of the matrices $X,AX,\ldots,A^{s-1}X$, and

$$T_s = \begin{bmatrix} M_1 & R_2^t & & & \\ R_2 & M_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & M_{s-1} & R_s^t \\ & & & R_s & M_s \end{bmatrix} \tag{2.7.2}$$

is the matrix representation of $\bar{A}$, now defined to be the operator A restricted to $L(s,X,A)$.

In order to reduce somewhat the complexity of the development, we restrict slightly the range of values the parameters in the problem may assume. In particular, we assume that the number p of columns in X

and the number of steps s satisfy

$$1 \leq p \times s \leq n \quad . \tag{2.7.3}$$

This restriction implies, for example, that if we start with an n-by-p matrix X where $1 \leq p \leq n$, then we can carry through the Block Lanczos method at most $\bar{s}$ steps where $\bar{s}$ satisfies

$$1 \leq \bar{s} \leq n/p \quad .$$

From the standpoint of the problems to which this method will usually be applied, i.e., problems of very large order n for which, because of limited storage, $p \ll n$ and $p \times s \ll n$, Equation (2.7.3) does not represent a real restriction.

We will follow a path similar to that followed by Lanczos in developing his algorithm. To begin, let A be a symmetric matrix of order n and let X be an n-by-p orthonormal matrix X. Let

$$X_1 = X \quad ,$$

compute

$$Y_1 = AX_1 \quad ,$$

and let Z_2 be the result of projecting Y_1 onto the subspace orthogonal to X_1. That is,

$$\begin{aligned} Z_2 &= (I - X_1 X_1^t) A X_1 \\ &= AX_1 - X_1 M_1 \end{aligned} \tag{2.7.4}$$

where

$$M_1 = X_1^t A X_1 \quad .$$

By definition, $X_1^t Z_2 = 0$. Strictly speaking, choosing Z_2 in this manner does not follow Lanczos' development. It can be shown, however,

that choosing Z_2 as in Equation (2.7.4) minimizes $\|Z_2\|$ with respect to all possible choices of M_1 .

Z_2 is an n-by-p matrix. Assume for the moment that $Z_2 \neq 0$, and let ρ_2 be the rank of Z_2 . Since $Z_2 \neq 0$, $\rho_2 > 0$, and by definition, $\rho_2 \leq p$. Factor Z_2 into the product of an n-by-p orthonormal matrix X_2 and a p-by-p matrix R_2 . That is,

$$Z_2 = X_2 R_2$$

where

$$X_2^t X_2 = I \quad .$$

If $\rho_2 = p$, then X_2 is orthogonal to X_1 since Z_2 is orthogonal to X_1 . If $\rho_2 < p$, then this may not necessarily be the case, so we add the orthogonality condition as an additional criterion for choosing X_2 . In any event,

$$X_2^t X_1 = 0 \quad . \tag{2.7.5}$$

Note that X_2 and R_2 can be computed using a Gram-Schmidt method or a QR factorization method based on Householder transformations. If $\rho_2 < p$, then $(p - \rho_2)$ columns of X_2 will not be determined by either of these methods. However, both methods can be programmed in such a way that the additional $p - \rho_2$ columns can be chosen so that X_2 is orthonormal and Equation (2.7.5) is satisfied.

Thus, at the end of the first step, starting from $X_1 = X$, we have computed matrices M_1 , R_2 and X_2 such that

$$X_2 R_2 = Z_2 = AX_1 - X_1 M_1$$

where

$$M_1 = X_1^t A X_1$$

and X_2 is orthonormal and orthogonal to X_1 .

Assume now that we are at the beginning of the j-th step where $j < s$, and that we have a sequence of mutually orthonormal matrices $X_1,\ldots,X_j$ and sequences of matrices $M_1,M_2,\ldots,M_{j-1}$ and $R_2,R_3,\ldots,R_j$ such that

$$\begin{aligned} X_2R_2 &= Z_2 = AX_1 - X_1M_1 \\ X_3R_3 &= Z_3 = AX_2 - X_2M_2 - X_1R_2^t \\ &\vdots \\ X_jR_j &= Z_j = AX_{j-1} - X_{j-1}M_{j-1} - X_{j-2}R_{j-1}^t \end{aligned} \qquad (2.7.6)$$

where

$$M_i = X_i^t A X_i \quad , \quad i = 1,\ldots,j-1 \ .$$

Compute $Y_j = AX_j$ and let Z_{j+1} be the result of projecting Y_j onto the space $\overset{\perp}{Sp}(X_1,\ldots,X_j)$ orthogonal to that spanned by $X_1,X_2,\ldots,X_j$. Since the projector onto $\overset{\perp}{Sp}(X_1,\ldots,X_j)$ is

$$P_{\overset{\perp}{Sp}(X_1,X_2,\ldots,X_j)} = (I - X_jX_j^t - X_{j-1}X_{j-1}^t - \cdots - X_1X_1^t) \ ,$$

we have

$$\begin{aligned} Z_{j+1} &= (I - X_jX_j^t - X_{j-1}X_{j-1}^t - \cdots - X_1X_1^t)AX_j \\ &= AX_j - X_jM_j - X_{j-1}N_{j-1,j} - \cdots - X_1N_{1,j} \end{aligned}$$

where

$$M_j = X_j^t A X_j$$

and

$$N_{i,j} = X_i^t A X_j$$

for $i = 1,2,\ldots,j-1$. However, for $i < j-1$, we have

$$N_{i,j} = 0$$

since

$$AX_i = X_{i-1}R_i^t + X_iM_i + X_{i+1}R_{i+1}$$

by Equation (2.7.6) and

$$X_j^t X_k = 0$$

for $k = i-1$, $k = i$, or $k = i+1$ if $i < j-1$. Thus,

$$Z_{j+1} = AX_j - X_jM_j - X_{j-1}N_{j-1,j} \quad .$$

Note that by Equation (2.7.6),

$$AX_{j-1} = X_{j-2}R_{j-1}^t + X_{j-1}M_{j-1} + X_jR_j$$

so that

$$N_{j-1,j}^t = X_j^t AX_{j-1} = R_j \quad .$$

Thus.

$$Z_{j+1} = AX_j - X_jM_j - X_{j-1}R_j^t \quad .$$

Note also that in computing Z_{j+1} , we need only project AX_j onto the space orthogonal to X_{j-1} and X_j , and Z_{j+1} will automatically be orthogonal to $X_1,\ldots,X_{j-2}$.

Z_{j+1} is an n-by-p matrix. Assume for the moment that $Z_{j+1} \neq 0$ and let ρ_{j+1} be the rank of Z_{j+1} . As we did for Z_2 , factor Z_{j+1} into the product of an n-by-p orthonormal matrix X_{j+1} and a p-by-p matrix R_{j+1} . That is,

$$Z_{j+1} = X_{j+1}R_{j+1} \quad ,$$

where $X_{j+1}^t X_{j+1} = I$. If $\rho_{j+1} < p$, then X_{j+1} is required in addition

to be orthogonal to $X_1, X_2, \ldots, X_j$.

Thus, during the j-th set, we have computed matrices M_j , X_{j+1} , and R_{j+1} such that

$$X_{j+1}R_{j+1} = Z_j = AX_j - X_jM_j - X_{j-1}R_j^t$$

where

$$M_j = X_j^t A X_j$$

and X_{j+1} is orthogonal to $X_1, X_2, \ldots, X_j$.

Assume now that we are at the beginning of Step 5 and that we have s n-by-p mutually orthonormal matrices $X_1, X_2, \ldots, X_s$ and sequences of matrices $M_1, M_2, \ldots, M_{s-1}$ and $R_2, R_3, \ldots, R_s$ such that

$$X_2R_2 = Z_2 = AX_1 - X_1M_1$$

$$X_3R_3 = Z_3 = AX_2 - X_2M_2 - X_1R_2^t$$

$$\vdots$$

$$X_sR_s = Z_s = AX_{s-1} - X_{s-1}M_{s-1} - X_{s-2}R_{s-1}^t \quad .$$

As before, we now compute $Y_s = AX_s$ and let Z_{s+1} be the result of projecting Y_s onto the space orthogonal to that spanned by $X_1, X_2, \ldots, X_s$. However, as we saw above for the j-th step, AX_s need only be projected onto the space orthogonal to X_s and X_{s-1} and it will be automatically orthogonal to $X_1, X_2, \ldots, X_{s-2}$. That is,

$$Z_{s+1} = (I - X_sX_s^t - X_{s-1}X_{s-1}^t)AX_s$$

$$= AX_s - X_sM_s - X_{s-1}N_{s-1,s}$$

where

$$M_s = X_s^t A X_s$$

and

$$N_{s-1,s} = X_{s-1}^t A X_s \quad .$$

Also, as before,

$$N_{s-1,s} = R_s^t \quad .$$

At this point, stop and consider the results of our efforts. We have computed sequences of matrices $(X_1,X_2,\ldots,X_s)$, $(M_1,M_2,\ldots,M_s)$, and $(R_2,R_3,\ldots,R_s)$ such that the X_i are mutually orthonormal and

$$\begin{aligned}
X_2R_2 &= Z_2 = AX_1 - X_1M_1 \\
X_3R_3 &= Z_3 = AX_2 - X_2M_2 - X_1R_2^t \\
&\vdots \\
X_{j+1}R_{j+1} &= Z_{j+1} = AX_j - X_jM_j - X_{j-1}R_j^t \\
&\vdots \\
X_sR_s &= Z_s = AX_{s-1} - X_{s-1}M_{s-1} - X_{s-2}R_{s-1}^t \\
Z_{s+1} &= AX_s - X_sM_s - X_{s-1}R_s^t \quad .
\end{aligned} \tag{2.7.7}$$

Let

$$X_k = (X_1,X_2,\ldots,X_k) \tag{2.7.8}$$

and

$$\mathcal{M}_k = \begin{bmatrix}
M_1 & R_2^t & 0 & . & . & 0 \\
R_2 & M_2 & . & . & . & . \\
0 & . & . & . & . & 0 \\
. & . & . & . & M_{k-1} & R_k^t \\
0 & . & . & 0 & R_k & M_k
\end{bmatrix} \tag{2.7.9}$$

for $k = 1,2,\ldots,s$. That is, χ_k is an n-by-pXk orthonormal matrix formed from the sequence $(X_1,X_2,\ldots,X_k)$ and $\mathcal{M}_k$ is a symmetric block tridiagonal matrix of order p·k formed from the sequences $(M_1,\ldots,M_2)$ and $(R_2,\ldots,R_s)$. We will now show that

$$\mathcal{M}_s = \chi_s^t A \chi_s \tag{2.7.10}$$

and that

$$L(s,X,A) = Sp(\chi_s) \tag{2.7.11}$$

contains

$$K(s,X,A) = Sp(X,AX,\ldots,A^{s-1}X) \quad . \tag{2.7.12}$$

We will then give a precise description of an algorithm for computing $(X_1,\ldots,X_s)$, $(M_1,\ldots,M_s)$, and $(R_2,\ldots,R_s)$. This algorithm will be based directly on the preceding development and in light of Equations (2.7.9), (2.7.10), and (2.7.11), is our goal in this section

Observe that Equations (2.7.7) can be rewritten

$$\begin{aligned}
AX_1 &= X_1M_1 + X_2R_2 \\
AX_2 &= X_1R_2^t + X_2M_2 + X_3R_3 \\
&\vdots \\
AX_j &= X_{j-1}R_j^t + X_jM_j + X_{j+1}R_{j+1} \\
&\vdots \\
AX_s &= X_{s-1}R_s^t + X_sM_s + Z_{s+1} \quad .
\end{aligned} \tag{2.7.13}$$

In matrix notation, these equations can be written

$$AX_s = \chi_s \mathcal{M}_s + Z_{s+1} \tag{2.7.14}$$

where

$$Z_{s+1} = (\theta,\theta,\ldots,\theta,Z_{s+1})$$

is an n-by-ps matrix all of whose columns are zero except the last p which are the columns of Z_{s+1}. By definition, Z_{s+1} is orthogonal to $X_1,\ldots,X_s$. Therefore,

$$X_s^t Z_s = \Theta \quad ,$$

and since

$$X_s^t X_s = I \quad ,$$

we have by Equation (2.7.14),

$$X_s^t A X_s = \mathcal{M}_s \quad .$$

Thus, $\mathcal{M}_s$ is the representation of $\bar{A}$, the restriction of A to the space $L(s,X,A)$.

We also have the following result.

Theorem 2.7.1. $L(s,X,A)$, the space spanned by the columns of $(X_1,X_2,\ldots,X_s)$, contains the columns of the matrices $X,AX,\ldots,A^{s-1}X$.

Proof. We will show inductively that

$$(X,AX,\ldots,A^{k-1}X) = (X_1,X_2,\ldots,X_k)U_k \qquad (2.7.15)$$

for $k = 1,2,\ldots,s$, where U_k is matrix of order pk. This will imply that each column of the matrices $X,AX,\ldots,A^{k-1}X$ is a linear combination of the columns of $(X_1,\ldots,X_k)$ and, therefore, that each column is contained in $L(s,X,A)$.

Clearly, Equation (2.7.15) holds for $k = 1$ since

$$X = X_1$$

and $U_1 = I$.

Assume that Equation (2.7.15) holds for some $k < s$. Multiplying both sides of Equation (2.7.15) by A gives us

$$(AX, A^2X, \ldots, A^kX) = (AX_1, AX_2, \ldots, AX_k)U_k \qquad (2.7.16)$$

from which we can conclude that

$$(X, AX, \ldots, A^kX) = (X_1, AX_1, \ldots, AX_k)\begin{pmatrix} I & \theta \\ \theta & U_k \end{pmatrix} . \qquad (2.7.17)$$

By Equations (2.7.13) with s replaced by k,

$$(X_1, AX_1, \ldots, AX_k) = (X_1, X_2, \ldots, X_{k+1}) \cdot \begin{pmatrix} I & M_1 & R_2^t & & & & \\ & R_2 & M_2 & R_3^t & & & \\ & & R_3 & M_3 & \ddots & & \\ & & & R_4 & \ddots & R_k^t & \\ & & & & \ddots & M_k & \\ & & & & & R_{k+1} & \end{pmatrix} \qquad (2.7.18)$$

Let V_k denote the last matrix in Equation (2.7.18). By combining Equations (2.7.17) and (2.7.18), we have

$$(X, AX, \ldots, A^kX) = (X_1, X_2, \ldots, X_{k+1})U_{k+1}$$

where

$$U_{k+1} = V_k \cdot \begin{pmatrix} I & \theta \\ \theta & U_k \end{pmatrix} .$$

This completes the inductive step and the proof of the theorem.

Before describing our algorithm, one point needs to be cleared up. Namely, our assumption that $Z_{j+1} \neq \theta$ for $j = 1, 2, \ldots, s-1$. Suppose that we intend to carry through s steps of our algorithm, computing

sequences $(M_i)_{i=1}^{s}$, $(R_i)_{i=2}^{s}$, and $(X_i)_{i=1}^{s}$, but for some value $j < s$, $Z_{j+1} = \Theta$. In this circumstance, we replace the value s by j and use the matrices $\mathcal{M}_j$ and X_j to compute exact eigenvalues and eigenvectors of A. This can be seen by considering Equation (2.7.14). If we replace s by j, the equation is still valid, and since $Z_{j+1} = \Theta$,

$$AX_j = X_j \mathcal{M}_j \quad .$$

Thus, the eigenvalues of $\mathcal{M}_j$ are eigenvalues of A and its eigenvectors can be used to compute eigenvectors of A (cf. Section 2.2).

It would also be possible to continue computing if $Z_{j+1} = \Theta$ for $j < s$ by simply choosing X_{j+1} such that

$$X_i^t X_{j+1} = \Theta \quad , \quad i \leq j \quad ,$$

and letting $R_{j+1} = \Theta$. An algorithm incorporating this idea is of little interest considering the applications we have in mind. Furthermore, it is extremely unlikely in practice that $Z_{j+1} = \Theta$ for any j even if exact arithmetic operations are assumed.

We now describe a Block Lanczos algorithm which can be used to compute the sequences $(M_i)_{i=1}^{s}$, $(R_i)_{i=2}^{s}$, and $(X_i)_{i=1}^{s}$:

Let A be a symmetric matrix of order n. Let p and s be integer values such that $p \geq 1$ and

$$1 \leq ps \leq n \quad .$$

Let X be an n-by-p orthonormal matrix.

Step 1. Let $X_1 = X$ and $i = 1$.

Step 2. Compute $Y_i = AX_i$ and $M_i = X_i^t Y_i$.

Step 3. If $i = s$, stop.

Step 4. Compute Z_{i+1} where

$$Z_{i+1} = \begin{cases} AX_1 - X_1M_1 & \text{if } i = 1 \\ AX_i - X_iM_i - X_{i-1}R_i^t & \text{if } i > 1 \end{cases}.$$

Step 5. If $Z_{i+1} = \theta$, set $s = i$.

Step 6. Compute X_{i+1} and R_{i+1} such that

$$Z_{i+1} = X_{i+1}R_{i+1}$$

and X_{i+1} is orthonormal. If the rank of Z_{i+1} is less than p, we require X_{i+1} to be orthonormal to X_j, $j \leq i$.

Step 7. Increase the value of i by one and go to Step 2.

The only time s will be different from its original value is if $Z_{i+1} = \theta$ for some $i < s$. As noted before, this is an extremely unlikely circumstance.

As the development preceding the above description suggests, the matrices $(X_i)_{i=1}^{s}$ computed using this algorithm will be mutually orthonormal and if

$$X_s = (X_1, X_2, \ldots, X_s)$$

and

$$\mathcal{M}_s = \begin{pmatrix} M_1 & R_2^t & & & \\ R_2 & M_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & M_{s-1} & R_s^t \\ & & & R_s & M_s \end{pmatrix}$$

then

$$\mathcal{M}_s = X_s^t A X_s$$

and

$$L(s,X,A) = Sp(X_1, X_2, \ldots, X_s)$$

contains the space

$$K(s,X,A) = Sp(X, AX, \ldots, A^{k-1}X) \quad .$$

If we compute the eigenvalues of $\mathcal{M}_s$, then the results of the previous section indicate that the p least eigenvalues of $\mathcal{M}_s$ will usually be accurate approximations to the p least eigenvalues of A. Computational experience has shown this to be the case.

In the case of the standard Lanczos algorithm, we saw that the sequence of vectors (x_i) generated by the method would lose orthogonality unless the vectors z_i were reorthogonalized with respect to all previously computed x_j, $j < i$. The same problem arises in the Block Lanczos method. Namely, the sequence of matrices X_i, although theoretically orthogonal with respect to each other, will in practice lose orthogonality unless the matrices Z_i are reorthogonalized with respect to all matrices X_i, $i < j$. The reorthogonalization can be combined with the computation of X_{i+1} and R_{i+1} in Step 6.

As with the standard Lanczos method, there is a question as to whether reorthogonalization is actually needed. That is, loss of orthogonality implies convergence of some of the eigenvalues of the restricted operator to those of the original matrix. Continuing the computation beyond the point that orthogonality is lost, however, will result in eigenvalues being computed more than once even if they are not multiple. Thus, for a reliable algorithm, we must either reorthogonalize or develop a criterion for determining when orthogonality is lost. We have chosen the former path which, although more time consuming, is more straightforward than the latter. We will discuss this point further in Section 3.1.

Note: If μ and ν are an eigenvalue and eigenvector, respectively, of $\mathcal{M}_s$, then μ and

$$\bar{q} = X_s \nu$$

are eigenvalue and eigenvector of $\bar{A}$, the restriction of A to $L(s,X,A) = Sp(X_s)$. By Equation (2.7.14), we have

$$A\bar{q} - \mu\bar{q} = Z_{s+1}\nu^{(s)}$$

where $\nu^{(s)}$ denotes the vector composed of the last p components of ν. This equation implies by Weinstein's inequality that there is an eigenvalue λ of A such that

$$|\lambda - \mu| \leq \|Z_{s+1}\nu^{(s)}\| \leq \|Z_{s+1}\| \cdot \|\nu^{(s)}\| \quad .$$

The eigenvectors corresponding to the extreme eigenvalues of $\mathcal{M}_s$ are often such that $\|\nu^{(s)}\|$ is very small. This seems to explain partly why the eigenvalues of $\mathcal{M}_s$ are often good approximations to the

eigenvalues of A. Note that if we computed X_{s+1} and R_{s+1} such that

$$Z_{s+1} = X_{s+1}R_{s+1} ,$$

then we have

$$\|Z_{s+1}v^{(s)}\| = \|R_{s+1}v^{(s)}\|$$

since the spectral norm is unitarily invariant. Thus,

$$|\lambda - \mu| \leq \|R_{s+1}v^{(s)}\| .$$

2.8 Iterating to Improve Accuracy

Let A be a symmetric matrix of order n and let X be an n-by-p orthonormal matrix. Let s be an integer greater than or equal to 1 and suppose that p and s satisfy

$$1 \leq p*s \leq n .$$

Let $\mathcal{M}_s$ be the representation of $\bar{A}$, the restriction of A to the space spanned by the columns of $X_s = (X_1, X_2, \ldots, X_s)$ which contains the space $Sp(X, AX, A^2X, \ldots, A^{s-1}X)$ where $\mathcal{M}_s$ and X_s have been computed using the Block Lanczos method of Section 2.7. Finally, let $\mu_1, \mu_2, \ldots, \mu_{ps}$ and $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_{ps}$ be the eigenvalues and eigenvectors, respectively, of $\bar{A}$ computed using $\mathcal{M}_s$ and X_s.

Theorem 2.6.1 suggests that the first p eigenvalues and eigenvectors of $\bar{A}$ will usually be accurate approximations to the corresponding eigenvalues of A. However, the expression bounding the errors in the eigenvalues contains a term $\tan\theta$ where θ is essentially the angle between $Sp(X)$ and $Sp(Q_1)$ where Q_1 is the orthonormal matrix

comprised of the first p eigenvectors of A. If we let $\bar{Q}_1 = (\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_p)$ and $\bar{\theta}$ be the angle between $Sp(\bar{Q}_1)$ and $Sp(Q_1)$, then we might reasonably expect that $\bar{\theta} < \theta$ and therefore that $\tan \bar{\theta} < \tan \theta$ since the vectors in $\bar{Q}_1$ will usually be more accurate approximations to eigenvectors of A than X. Thus, we might reasonably expect to compute more accurate approximations by re-applying the Block Lanczos method to $\bar{Q}_1$.

This discussion suggests the following algorithm for computing approximations to the p least eigenvalues and eigenvectors of A to a specified accuracy:

Let A, p, s, and X be as defined above.

Step 1. Using the Block Lanczos method, compute χ_s and $\mathcal{M}_s$, the representation of A restricted to the space spanned by X_s which contains the space $Sp(X, AX, \ldots, A^{s-1}X)$.

Step 2. Compute the eigenvalues μ_i and eigenvectors y_i of $\mathcal{M}_s$. Compute $\bar{q}_i = X_s y_i$, $i = 1, 2, \ldots, p$.

Step 3. Estimate the accuracy of μ_i and $\bar{q}_i$ as approximations to the p least eigenvalues and eigenvectors of A. If they are all sufficiently accurate, stop.

Step 4. Let $X = (\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_p)$ and go to Step 1.

We will discuss how to estimate the accuracy of computed results in Chapter 3.

The above algorithm contains most of the essential features of our final method. We will modify it however for the following reasons: (1) The block size p will usually be different from the number of

eigenvalues we are attempting to compute. In fact, we will want to vary the block size as the computation proceeds. (2) As we saw in Section 2.6, the errors in the computed eigenvalues decrease at varying rates. Hence, some of the eigenvalues, usually those least in value, will converge sooner than others. For this reason, we will want to continue computing after we have accepted and stored some eigenvalue and eigenvector approximations, without recomputing the same eigenvalues and eigenvectors. Some modification of our current method is thus required since it will always tend to compute the least eigenvalues. In the next section, we will see how to do this.

2.9 Restricting A to a Space Orthogonal to Computed Eigenvectors

Let A be a symmetric matrix. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of A with eigenvectors $q_1, q_2, \ldots, q_n$. Let $\bar{\lambda}_1 \leq \bar{\lambda}_2 \leq \ldots \leq \bar{\lambda}_m$ and $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_m$ be approximate eigenvalues and eigenvectors, respectively, of A in the sense that

$$A\bar{q}_i - \bar{\lambda}_i \bar{q}_i = \rho_i \quad , \quad i = 1,2,\ldots,m \quad ; \tag{2.9.1}$$

and the $\bar{q}_i$ are orthonormal where $\|\rho_i\| = \epsilon_i \ll 1$ and m is some integer greater than zero and less than n.

How can we use the algorithm of the previous section to compute approximations to eigenvalues and eigenvectors different from those we have already computed without recomputing these latter values and vectors? For instance, if the $\bar{\lambda}_i$ and $\bar{q}_i$ were computed by means of this method, then they will most likely correspond to the m least eigenvalues and eigenvectors of A. Re-applying the method to A without taking these already computed approximations into account in

some fashion would result in our recomputing the same eigenvalues and eigenvectors.

The answer to the above question is to apply our method to an operator which is different from A but, nonetheless, related to A. In particular, we apply our method to $\hat{A}$, the restriction of A to the space <u>orthogonal</u> to $Sp(\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m)$. $\hat{A}$ has $n-m$ eigenvalues and eigenvectors which are approximations to the eigenvalues and vectors of A different from those already approximated. To see this, let $\bar{q}_{m+1},\bar{q}_{m+2},\ldots,\bar{q}_n$ be an orthonormal basis for the space (of dimension $n-m$) orthogonal to $Sp(\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m)$. Let

$$\bar{Q}_1 = (\bar{q}_1,\ldots,\bar{q}_m) ,$$

$$\bar{Q}_2 = (\bar{q}_{m+1},\ldots,\bar{q}_n) ,$$

$$M_1 = \bar{Q}_1^t A \bar{Q}_1 , \quad \text{and}$$

$$M_2 = \bar{Q}_2^t A \bar{Q}_2 .$$

Let $\bar{\lambda}_{m+1},\bar{\lambda}_{m+2},\ldots,\bar{\lambda}_n$ denote the $n-m$ eigenvalues of M_2. Note that M_2 is the representation of $\hat{A}$ and, hence its eigenvalues are the eigenvalues of $\hat{A}$.

We now want to show that if each ϵ_i defined in Equation (2.9.1) is small, then $\bar{\lambda}_{m+1},\ldots,\bar{\lambda}_n$ will be accurate approximations to the remaining eigenvalues of A. Note that if all the $\epsilon_i = 0$, then the eigenvalues of $\hat{A}$ would also be eigenvalues of A.

Define

$$\bar{Q} = (\bar{Q}_1,\bar{Q}_2)$$

and

$$B = \bar{Q}^t A \bar{Q} = \begin{bmatrix} \bar{Q}_1^t A \bar{Q}_1 & \bar{Q}_1^t A \bar{Q}_2 \\ \bar{Q}_2^t A \bar{Q}_1 & \bar{Q}_2^t A \bar{Q}_2 \end{bmatrix} .$$

Note that B is similar to A. Let

$$C = \begin{bmatrix} \bar{Q}_1^t A \bar{Q}_1 & \Theta \\ \Theta & \bar{Q}_2^t A \bar{Q}_2 \end{bmatrix} = \begin{bmatrix} M_1 & \Theta \\ \Theta & M_2 \end{bmatrix}$$

and

$$\Delta = \begin{bmatrix} \Theta & \bar{Q}_1^t A \bar{Q}_2 \\ \bar{Q}_2^t A \bar{Q}_1 & \Theta \end{bmatrix} .$$

Thus,

$$B = C + \Delta .$$

Since B is similar to A, the eigenvalues of B are the same as the eigenvalues of A. By the theory of perturbations for symmetric matrices, (see, for example, Wilkinson [23], Chapter 3), the eigenvalues of C differ from those of B (and hence A) by amounts that are bounded by $\|\Delta\|$. This quantity can in turn be bounded as follows: It can be shown that

$$\|\Delta\| = \|\bar{Q}_2^t A \bar{Q}_1\| .$$

Let

$$R = [\rho_1, \rho_2, \cdots, \rho_m] ,$$

where ρ_k is the residual vector defined in Equation (2.9.1). By the definition of ρ_k,

$$A\bar{Q}_1 - \bar{Q}_1\bar{\Lambda}_1 = R \qquad (2.9.2)$$

where $\bar{\Lambda}_1 = \text{diag}(\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_m)$. Since

$$\bar{Q}_2^t \bar{Q}_1 = \Theta \quad ,$$

we have

$$\bar{Q}_2^t A \bar{Q}_1 = \bar{Q}_1^t R \quad ,$$

and therefore,

$$\|\Delta\| = \|\bar{Q}_2^t A \bar{Q}_1\| = \|\bar{Q}_1^t R\| \quad .$$

Since the spectral norm is invariant with respect to orthogonal transformations, and $\bar{Q}_1$ is orthogonal,

$$\|\Delta\| = \|\bar{Q}_1^t R\| = \|R\| \quad .$$

If all the $\epsilon_i = \|\rho_i\|$ are small, then $\|\Delta\|$ will be small also. For example,

$$\|R\| \leq (\epsilon_1^2 + \epsilon_2^2 + \ldots + \epsilon_m^2)^{1/2} \quad .$$

If $m = 9$ and $\epsilon_i \leq 10^{-10}$, $i = 1,2,\ldots,9$, then

$$\|\Delta\| \leq 3 \cdot 10^{-10} \quad ,$$

and the eigenvalues of C differ from those of B , and hence A , by quantities which are less in modulus than $3 \cdot 10^{-10}$.

The set of eigenvalues of C is the union of the sets of eigenvalues of M_1 and M_2 . From Equation (2.9.2) we can conclude that

$$M_1 = \bar{\Lambda}_1 + \bar{Q}_1^t R \quad .$$

Thus, $\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_m$ differ from the eigenvalues of M_1 by amounts

bounded by $\|R\|$. Furthermore, the eigenvalues of M_2 are the same as the eigenvalues of $\hat{A}$. Taken together, $(\bar{\lambda}_i\ ,\ i = 1,\ldots,m)$ and the $(n-m)$ eigenvalues of $\hat{A}$ approximate the entire spectrum of A . Thus, for example, if $\bar{\lambda}_1,\ldots,\bar{\lambda}_m$ approximate the m least eigenvalues $\lambda_1,\ldots,\lambda_m$ of A , then the p least eigenvalues of $\hat{A}$ will approximate $\lambda_{m+1},\lambda_{m+2},\ldots,\lambda_{m+p}$ with errors bounded by $\|R\|$.

The Block Lanczos algorithm of Section 2.6 can be applied directly to $\hat{A}$ to compute approximations to its least eigenvalues. The initial orthonormal matrix X must lie in the domain of $\hat{A}$. That is, X must lie in the space orthogonal to the vectors $\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m$. Note also that

$$\hat{A}y = (I - \bar{Q}_1\bar{Q}_1^t)Ay$$

so that to multiply by $\hat{A}$, we first multiply by A and then project the result onto the space orthogonal to $\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m$. Furthermore, referring to Section 2.7, we add the extra requirement that X_j be computed so that it is orthogonal to $\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m$ in the event that Z_j is of less than full rank. Note that this will automatically be the case if Z_j is of full rank.

The algorithm of the previous section, when modified to take into account previously computed eigenvalue approximations, can be described as follows:

Let A be a symmetric matrix of order n . Let $\bar{q}_1,\bar{q}_2,\ldots,\bar{q}_m$ be orthonormal vectors with $m < n$. Let p and s be integer values such that $p \geq 1$, $s \geq 1$ and

$$1 \leq ps \leq n-m \quad .$$

Let X be an n-by-p orthonormal matrix which satisfies

$$X^t \bar{Q}_1 = 0 \qquad (2.9.3)$$

where $\bar{Q}_1 = (\bar{q}_1, \ldots, \bar{q}_p)$.

Compute approximations to the p least eigenvalues of $\hat{A}$, the restriction of A to the space orthogonal to $Sp(\bar{q}_1, \ldots, \bar{q}_m)$, as follows:

Step 1. Using the Block Lanczos algorithm, compute X_s and $\mathcal{M}_s$, the representation of $\hat{A}$ restricted to the space spanned by X_s which contains the vectors $(X, \hat{A}X, \ldots, \hat{A}^{s-1}X)$.

Step 2. Compute the eigenvalues μ_i and eigenvectors y_i of $\mathcal{M}_s$. Compute $\bar{q}_{m+i} = \mathcal{M}_s y_i$ for $i = 1, \ldots, p$.

Step 3. Estimate the accuracy of μ_i and $\bar{q}_i$ as approximations to an eigenvalue and eigenvector, respectively, of A . If they are all sufficiently accurate, stop.

Step 4. Let $X = (\bar{q}_{m+1}, \bar{q}_{m+2}, \ldots, \bar{q}_{m+p})$ and go to Step 1.

Note that, by earlier comments, each column in X_s will be orthogonal to Q_1 and, hence, each $\bar{q}_{m+i}$ will be orthogonal to $\bar{Q}_1$, $i = 1, \ldots, p$. Therefore, each time Step 1 is executed, the matrix represented by X will satisfy Equation (2.9.3).

If $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_m$ are accurate approximations to some of the eigenvectors of A in the sense of Equation (2.9.1), then our discussion suggests that the above algorithm will compute approximations to eigenvalues and eigenvectors of A different from those already computed.

In the next section, we will integrate this method into a complete algorithm which will also allow us to vary the block size p .

2.10. A Complete Iterative Block Lanczos Algorithm

Let A be a symmetric matrix of order n with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and orthonormal eigenvectors $q_1, q_2, \ldots, q_n$. Let r be an integer greater than zero and less than or equal to n. In this section we will outline an algorithm to solve the following problem: Compute accurate approximations $\bar{\lambda}_i$ and $\bar{q}_i$ to λ_i and q_i for $i = 1, \ldots, r$. Our algorithm will incorporate the idea of the previous section and have as its basis the Block Lanczos method. Basically, the plan of the algorithm is as follows: Compute approximations to the least eigenvalues and eigenvectors of A. When some of them, say m, are sufficiently accurate, compute approximations to the least eigenvalues and eigenvectors of $\hat{A}$, the restriction of A to the space orthogonal to those vectors already computed.

Our method can be described as follows:

Step 1. Let $m = 0$. Pick values for p and s such that $p \geq 1$, $s \geq 2$, and $1 \leq p \times s \leq n$. Choose an n-by-p orthonormal matrix X.

Step 2. Starting with X, apply the Block Lanczos method to $\hat{A}$, the restriction of A to the space orthogonal to $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_m$. (If $m = 0$, then $\hat{A} = A$.) Let $\mathcal{M}_s$ and X_s be the matrices computed by the Block Lanczos method.

Step 3. Compute the eigenvalues μ_i and eigenvectors y_i of $\mathcal{M}_s$, $i = 1, \ldots, p \times s$. Compute $\bar{q}_{m+i} = X_s y_i$, $i = 1, 2, \ldots, p$.

Step 4. Estimate the accuracy of $\bar{\lambda}_{m+i} = \mu_i$ and $\bar{q}_{m+i}$ as approximations to an eigenvalue and eigenvector, respectively, of A for $i = 1,2,\ldots,p$. Suppose the first k of these approximations are accepted. (If none is accepted, $k = 0$.)

Step 5. Choose new values for p and s such that $p \geq 1$, $s \geq 2$, and $1 \leq p \times s \leq n-(m+k)$. Let

$$X = (q_{m+k+1}, q_{m+k+2}, \ldots, q_{m+k+p}) .$$

Step 6. Increase the value of m by k. If $m < r$, go to Step 2. Otherwise, stop.

An unfortunate choice for the initial X can cause this algorithm to fail. For instance, if none of the columns of X contains components corresponding to any of the eigenvectors for one of the initial eigenvalues λ_i, $1 \leq i \leq r$, then the above algorithm will fail to compute λ_i. In practice, however, such a circumstance is unlikely to occur so we overlook this possibility and accept the above algorithm as a solution to the problem posed at the start of this section.

Note that in Step 5, if $k = 0$, then X is chosen as in the last section. That is, X is chosen to be the eigenvectors corresponding to the p least eigenvalues of $\hat{A}$. Otherwise, if $k > 0$, then X is chosen to be the next p eigenvectors following the k that were accepted.

Also, each time Step 2 is executed after the first, X will be such that

$$X^t A X = \operatorname{diag}(\bar{\lambda}_{m+1}, \bar{\lambda}_{m+2}, \ldots, \bar{\lambda}_{m+p})$$

where $\bar{\lambda}_{m+1}, \bar{\lambda}_{m+2}, \ldots, \bar{\lambda}_{m+p}$ are the eigenvalues of $\hat{A}$ computed during the previous step. This can be seen as follows:

We have

$$\mathcal{M}_s = X_s^t A X_s \quad .$$

The X we chose in Step 5 satisfies

$$X = X_s Y$$

where $Y = (y_{m+1}, y_{m+2}, \ldots, y_{m+p})$. By definition,

$$Y^t \mathcal{M}_s Y = \operatorname{diag}(\bar{\lambda}_{m+1}, \bar{\lambda}_{m+2}, \ldots, \bar{\lambda}_{m+p}) \quad .$$

Therefore,

$$\begin{aligned} X^t A X &= Y^t X_s^t A X_s Y \\ &= Y^t \mathcal{M}_s Y \\ &= \operatorname{diag}(\bar{\lambda}_{m+1}, \bar{\lambda}_{m+2}, \ldots, \bar{\lambda}_{m+p}) \quad . \end{aligned}$$

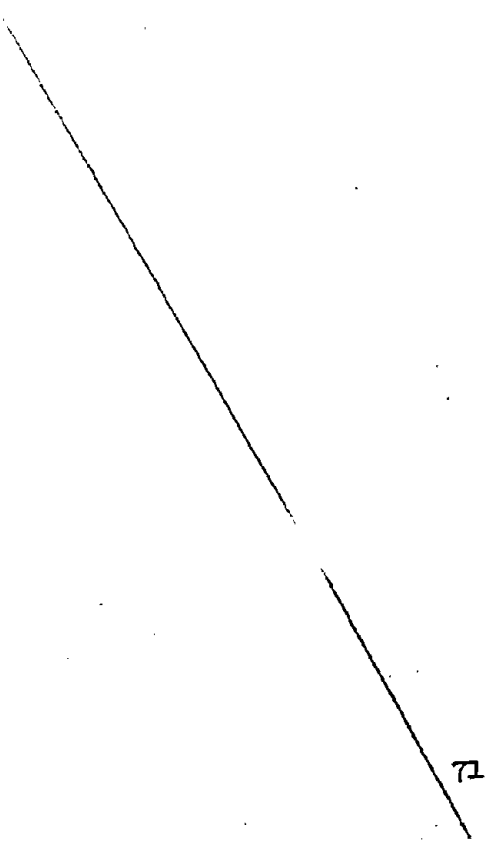
Since advantage can be taken of this property, the initial X we choose in Step 1 will, in practice, also be chosen so that $X^t A X$ is a diagonal matrix.

Note, in addition, that each time Step 2 is executed, X will be orthogonal to all previously computed vectors. This allows us to use the Block Lanczos algorithm to compute the eigenvalues and eigenvectors of $\hat{A}$.

Finally, the range of values for p and s will in practice be restricted somewhat more than indicated in Steps 1 and 5. In our implementation, we require basically $p \times s$ vectors in which to carry out the Block Lanczos method. In practice, we will generally have

much fewer than n vectors for this purpose and the values of p and s must be chosen with this limitation in mind.

In the next chapter, we will consider the problems associated with the implementation of the above algorithm. In particular, we will consider strategies for choosing values for p and s. We will also consider the problem of estimating the accuracy of computed eigenvalues and eigenvectors.



3. IMPLEMENTATION

In this chapter we will consider the problems associated with implementing and applying the algorithm developed in the previous chapter.

In Section 3.1, we will discuss the need for reorthogonalizing the sequence of matrices (X_i) computed in the Block Lanczos method. In our use of this method, we do reorthogonalize these matrices and in this section we will discuss our reasons for taking this path.

Estimating the accuracy of computed eigenvalues and eigenvectors will be the subject matter of Section 3.2. We will see how information on the accuracy of computed results can be obtained in the context of our method and how it can be used to stop the program when a specified accuracy has been obtained.

In Section 3.3, we will examine the problem of choosing a block size for the Block Lanczos method. By considering some examples, we will see that this is not a simple problem. We will then suggest some guidelines by which an informed choice might be made.

Finally, in Section 3.4 we will consider certain practical matters such as program and storage organization, storage requirements and operation counts.

3.1 Reorthogonalization in the Block Lanczos Method

Recall that in the Block Lanczos method we compute a sequence of matrices $X_1, X_2, \ldots, X_s$ which theoretically form a basis for the space $Sp(X, AX, \ldots, A^{s-1}X)$ where A is a symmetric matrix of order n and X is an n-by-p orthonormal matrix, where p and s are integers

such that $p \geq 1$, $s \geq 1$, and $ps \leq n$. While theoretically the sequence of matrices (X_i) is orthonormal, in practice they depart from orthonormality after a few steps of the method. From the standpoint of the standard Lanczos method $(p = 1)$, this loss of orthogonality was a serious shortcoming. To remedy this situation, an orthogonalization step was added to the algorithm whereby each vector x_i is reorthogonalized with respect to all previously computed x_j, $j < i$. Paige [17] however found that useful results could be obtained even if a reorthogonalization step is not included since loss of orthogonality implies convergence of some of the eigenvalues of the tridiagonal matrix to those of the original matrix A. The major drawback with Paige's approach is that eigenvalues of A will often appear more than once when the eigenvalues of the tridiagonal matrix are computed. The reason for this is that once orthogonality is lost, the method essentially restarts and recomputes eigenvalues it has already computed. Thus, the validity of results computed using the Lanczos method without reorthogonalization is questionable.

The same problem arises with the Block Lanczos method. That is, if we apply the method without reorthogonalizing the X_i then accurate results can be computed but their validity is questionable in the same sense as before -- we can not determine which of the eigenvalues we compute are real and which are images.

Adding reorthogonalization to the Block Lanczos method stabilizes it and we can be sure of the results we compute, but the cost of this insurance is considerable. (The stability of the method with reorthogonalization is not something we have proved, but is an observation

based on our computational experience.) Not only does reorthogonalization add a large number of operations to the method, but necessitates the presence of each matrix X_i in memory during each step of the Block Lanczos method. If reorthogonalization could be safely eliminated, then not only would there be a considerable reduction in the amount of computation, but at most two elements of the sequence (X_i) would need to be present in memory at any time allowing the others to be stored on magnetic disk or tape.

However, even with reorthogonalization, our early experiments indicated that the Block Lanczos method could compete effectively in terms of reliability, efficiency, and storage requirements with the method of simultaneous iteration, previously the most effective method in general for the solution of large sparse eigenproblems. For this reason, we chose originally to remedy the above problem with the Block Lanczos method by adding reorthogonalization.

Since this time, Professor W. Kahan of the University of California at Berkeley has related some of his results and conclusions obtained from experiments using a Block Lanczos method in the late 1950's, which have never been published. He concluded that a Block Lanczos method could be applied in an iterative fashion (as we have used it) without reorthogonalization as long as the sequence of matrices (X_i) retained "healthy independence". He also discovered a way of determining when independence is lost and used this test as a means of stopping the method. The work of Cullum [4] appears to reflect Kahan's ideas and approach. The reader is urged to consult Cullum's work for more details on this alternate approach.

3.2 Estimating Accuracy and Convergence Criteria

Given a symmetric matrix A, the goal of our algorithm is to compute scalars μ_i and vectors x_i (where we assume $\|x_i\| = 1$) such that

$$Ax_i - \mu_i x_i = \rho_i \tag{3.2.1}$$

and

$$\|\rho_i\| = \epsilon_i < \tau \tag{3.2.2}$$

where τ is some tolerance (to be determined). In this section we are interested in determining when the eigenvalue and vector approximations computed using the iterative Block Lanczos method satisfy Equation (3.2.2).

Let X be an n-by-p orthonormal matrix. Suppose that

$$X^t AX = M = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p) \tag{3.2.3}$$

where

$$x_j^t Ax_j = \mu_j \tag{3.2.4}$$

and x_j is the j-th column of X. If X is used to start the Block Lanczos method (refer to Section 2.7 for notation), then $X_1 = X$ and

$$Z_2 = AX_1 - X_1 M_1 \tag{3.2.5}$$

and the j-th column of Z_2 is

$$z_j = Ax_j - \mu_j x_j = \rho_j \quad . \tag{3.2.6}$$

Thus, the Block Lanczos method immediately provides estimates of the form (3.2.1) for the columns of X and the Rayleigh quotients μ_j defined by Equation (3.2.4) provided X satisfies (3.2.3). However, for the method we have developed, the X used to start the Block Lanczos method each time will satisfy (3.2.3) where the μ_j are the eigenvalues computed during the previous step (cf. Section 2.10).

Note: Since (cf. Section 2.7)

$$Z_2 = X_2 R_2$$

and X_2 is orthonormal, we also have

$$\epsilon_j = \|\rho_j\| = \|z_j\| = \|r_j\|$$

where r_j is the j-th column of R_2. In our program, however, we have found it easier to use Z_2 than R_2 to compute the values of ϵ_j.

The tolerance τ we use is basically relative error for eigenvalues greater than one in modulus and absolute error for eigenvalues less than or equal to one in modulus. That is,

$$\tau = \begin{cases} |\mu_i| \times \text{eps} & \text{if } |\mu_i| > 1 \text{ , and} \\ \text{eps} & \text{if } |\mu_i| \leq 1 \text{ ,} \end{cases} \qquad (3.2.7)$$

where μ_i is the eigenvalue approximation corresponding to ϵ_i and eps is some specified precision.

However, in determining the accuracy of computed results and establishing a stopping criterion, the error in previously computed eigenvalues and eigenvectors must also be taken into account. Recall (cf. Section 2.9) that if we have already computed m eigenvalue and eigenvector approximations $\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_m$ and $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_m$, then additional approximations may be obtained by computing the eigenvalues and eigenvectors of $\hat{A}$, the restriction of A to the space orthogonal to $Sp(\bar{q}_1, \ldots, \bar{q}_m)$. However, the eigenvalues and eigenvectors of $\hat{A}$ will differ from those of A by amounts that depend on the magnitudes of $\bar{\epsilon}_1, \bar{\epsilon}_2, \ldots, \bar{\epsilon}_m$ where

$$\bar{\epsilon}_i = \|\bar{\rho}_i\|$$

and

$$\bar{\epsilon}_i = A\bar{q}_i - \bar{\lambda}_i\bar{q}_i$$

for $i = 1,2,\ldots,m$. For instance at one extreme, if the $\bar{\epsilon}_i$ are large and the $\bar{q}_i$ bear no relationship to the eigenvectors of A, then we would not expect the eigenvalues and eigenvectors of $\hat{A}$ to be useful approximations to those of A. At the other extreme, if all the $\bar{\epsilon}_i$ are zero, then each $\bar{q}_i$ is an eigenvector of A and the eigenvalues and eigenvectors of $\hat{A}$ will also be eigenvalues and eigenvectors of A.

When applying our algorithm, the eigenvalues and eigenvectors we are computing are converging to those of $\hat{A}$ and not A and, therefore, if we compute the ϵ_i in Equation (3.2.2) with respect to A and not $\hat{A}$, then there is a lower limit to the values of the ϵ_i beyond which we can not reasonably expect them to descend.

Our program takes this error into account as follows:

If

$$\bar{\tau}_m = (\bar{\epsilon}_1^2 + \ldots + \bar{\epsilon}_m^2)^{1/2} \qquad (3.2.8)$$

then we accept a value μ and a vector x if their corresponding ϵ_i satisfies

$$\epsilon_i < \tau + \bar{\tau}_m \qquad (3.2.9)$$

where τ is given by Equation (3.2.7). (We will add one more term to τ later on to account for round-off error in computing ϵ_i.) One way of looking at this criterion is that it is a way of estimating when the computed eigenvalue μ has converged to an eigenvalue $\hat{\mu}$ of $\hat{A}$. Recall from Section 2.9 that there is an eigenvalue λ of A such that

$$|\lambda - \hat{\mu}| \leq (\bar{\epsilon}_1^2 + \ldots + \bar{\epsilon}_m^2)^{1/2} = \tau_m \quad .$$

If we let

$$\bar{Q}_1 = (\bar{q}_1,\ldots,\bar{q}_m) \quad ,$$

then

$$\begin{aligned}\bar{\rho} &= \hat{A}x - \mu x \\ &= (I - \bar{Q}_1\bar{Q}_1^t)(Ax - \mu x) \qquad \text{by definition of } \hat{A} \ , \\ &= (I - \bar{Q}_1\bar{Q}_1^t)\rho \qquad \text{where } \rho \equiv Ax - \mu x \\ &= \rho - \bar{Q}_1\bar{Q}_1^t\rho \quad .\end{aligned}$$

Thus.

$$\|\bar{\rho}\| \geq \|\rho\| - \|\bar{Q}_1\bar{Q}_1^t\rho\| \quad .$$

It can be shown that

$$\|\bar{Q}_1\bar{Q}_1^t\rho\| \leq \bar{\tau}_m \quad .$$

Thus. as long as

$$\tau_i \geq \tau + \bar{\tau}_m \quad ,$$

then

$$\|\bar{\rho}\| \geq \tau$$

and we can improve on the accuracy of μ relative to τ . Basically, once (3.2.9) has been satisfied we have reached the obtainable accuracy allowed by the errors in $\bar{\lambda}_i$ and $\bar{q}_i$, $i = 1,\ldots,m$.

Note: The eigenvalues μ_j computed by our program are Rayleigh quotients computed using the vector x_j . (We assume that $\|x_j\| = 1$.) That is,

$$\mu_j = x_j^t Ax_j \quad .$$

It is well known that the Rayleigh quotient is often twice as accurate

an approximation to an eigenvalue of A as the vector used to form it. That is, if

$$Ax_j - \mu_j x_j = \rho_j \quad ,$$

and

$$\|\rho_j\| = \epsilon_j \quad ,$$

then there is an eigenvalue λ of A such that

$$|\lambda - \mu_j| \leq \epsilon_j^2/\delta$$

where δ is the minimum separation between μ_j and the remaining eigenvalues of A [23, p. 188]. Thus, for well separated eigenvalues, the error in μ_j will be approximately ϵ^2 rather than ϵ.

Furthermore, the error in the eigenvalues of $\hat{A}$ as approximations to the eigenvalues of A will often be far less than the bound suggested by Equation (3.2.8). The results of Paige [16] in fact suggest that the error for well separated eigenvalues will be proportional to $(\epsilon_1^2 + \epsilon_2^2 + \ldots + \epsilon_m^2)$ rather than the square root of this term.

Therefore, we expect the computed eigenvalues to be twice as accurate as the norms ϵ_i of the residuals ρ_i indicate. Computational experience verifies that this is usually the case.

The error in the computed value of ϵ_i arising from round-off errors will generally be unimportant in estimating the accuracy of computed results and determining when convergence has occurred. That is, while ϵ_i may be somewhat imprecise in terms of number of significant digits, its order of magnitude will usually be correct and this is what is important in determining convergence. The exception to this statement occurs when almost complete cancellation takes place when ρ_i is computed. This will

happen when the quantity eps in Equation (5.2.7) is chosen close to the limits of the precision obtainable on the machine the program is being executed on. To remedy this situation, we modify τ as follows: Accept μ_i and x_i when the norm r_i of their residual satisfies

$$r_i \leq \tau + \bar{\tau}_i$$

where τ is now defined to be

$$\tau = \begin{cases} |\mu_i|(\text{eps} + 10n \times \text{macheps}) & \text{if } |\mu_i| > 1 \\ (\text{eps} + 10n \times \text{macheps}) & \text{if } |\mu_i| \leq 1 \end{cases} .$$

where n is the order of A and macheps is the smallest positive floating point value such that 1 + macheps > 1 on the machine the program is being executed on. (E.g., macheps = 16^{-13} for double precision computation on I.B.M. System/360 computers.) The form of the new term is suggested by the floating point error analysis of the computation of inner products (see Wilkinson [23], Chapter 3) and actually is a considerable overestimate (by about a factor of 10 which is arbitrarily chosen) of the errors that actually occur. However, as we pointed out above, this new term will usually be insignificant in comparison to eps in most applications.

3.3 Choice of Block Size

One of the problems we have not discussed up to this point is how to choose the block size for the Block Lanczos method during each iteration of the algorithm of Section 2.10. A good choice for the block size can often considerably reduce the number of iterations required to compute a given number of eigenvalues and eigenvectors. The difficulty is that the best information for choosing the block size is accurate knowledge of the spectrum of the problem matrix which is, of course, the information we are trying to determine. Even for the same matrix, however, the best block size will also depend on the number of eigenvalues and eigenvectors we are trying to compute and the number of steps s of the Block Lanczos method we can carry out for a given block size p.

Example. Let $A = -H^{-1}$ where H is the discrete biharmonic operator of order 256 [1]. Suppose we are given $q = 12$ vectors in which to apply the Block Lanczos method. This means (cf. Section 3.4) that at any point in our computation, the number m of eigenvalues we have already computed, the block size p, and the number of steps s for the Block Lanczos method must satisfy $m + p \times s \leq 12$. Suppose also that we are trying to compute $r = 6$ eigenvalues and eigenvectors. If we apply the program of Appendix A to this problem, we arrive at the results given in Table 3.3.1, where an iteration basically involves an execution of steps two through six of the iterative Block Lanczos algorithm of Section 2.10, and the number of matrix multiplies is the number of times the matrix-vector product Ay is computed where y is a given vector.

Table 5.3.1

initial block size	iterations	matrix multiplies
1	10	93
2	7	68
3	7	71
4	8	80
5	8	86

Note: The strategy used in these tests is at each point to choose the block size equal to the previous block size p unless there are fewer than p vectors to be computed in which case it is chosen equal to the number of vectors left to be computed.

If we increase the required number of vectors r to 10, then we have the results of Table 5.3.2.

Table 5.3.2

initial block size	iterations	matrix multiplies
1	55	175
2	20	126
3	22	138
4	20	139
6	19	133

Note that with $q = 12$, the largest block size allowed is $p = 6$. Note also that it is inefficient to choose $p = 5$ since with this block size, only two steps of the Block Lanczos method can be carried out as is the case with $p = 6$.

Finally with q increased to 24 and r equal to 10, we have the results of Table 3.3.3.

Table 3.3.3

initial block size	iterations	matrix multiplies
1	11	219
2	8	158
3	8	156
4	7	140
6	7	128
8	7	139
12	8	166

It would appear from this example that the best block size when $q = 12$ is $p = 2$, and when $q = 24$, it is $p = 6$. This example also demonstrates that there are advantages to be gained by using a Block Lanczos method in comparison to a standard Lanczos method $(p = 1)$.

The point of this example is that it is difficult in advance to predict what the best block size will be. Therefore, rather than attempt to describe specific strategy for choosing the block size, we will establish some guidelines that we can use to make informed decisions in particular problems.

In our program, we are assuming that there is an upper bound on the product pXs imposed by storage limitations. Thus, if we increase the

value of p, the value of s must usually decrease. This generally implies that each individual vector in the block will converge at a slower rate. These slower rates of convergence are compensated for by the fact that we are computing more vectors at once. One conclusion we can draw, however, is that it seldom pays to choose a block size larger than the number of eigenvalues we are trying to compute. For example, if we are interested in computing two eigenvalues, then while it might sometimes prove advantageous to choose $p = 1$, it will scarcely ever pay to choose $p = 3$.

Cullum and Donath [4] choose the block size equal to the number of vectors that remain to be computed and, thus, initially equal to the required number of eigenvalues. There is much to recommend this approach. There is no difficulty in restarting after some eigenvalues have converged since the block size can only decrease. All useful information is retained from one iteration to the next. However, as the above example indicates, choosing the block size in this way does not always lead to the best choice, and it also means that we can only compute a number of vectors less than or equal to one-half of the total number q of vectors available to the Block Lanczos method. While this strategy can be utilized with the program we have included in the Appendix, the program has been designed to compute as many as $q-1$ eigenvalues and eigenvectors. In situations where the value of q is several times that of r, the required number of eigenvalues, and where there is no information about the matrix to indicate otherwise, Cullen's approach is a good strategy.

For problems which are known to have multiple eigenvalues, it is best to choose the block size at least as large as the greatest

multiplicity, or if this is not possible, at least greater than one. For example, the biharmonic operator in the above example is known to have multiple eigenvalues with multiplicities at most two because of symmetries in the physical problem it models. This suggests a block size of at least two. As the results indicate, a block size of one was clearly less efficient than a block size greater than one.

Theorem 2.6.1 gave bounds on the errors in the least eigenvalues computed using the Block Lanczos method. These bounds contained a term $T_{s-1}\left(\frac{1+\gamma_k}{1-\gamma_k}\right)$ where $\gamma_k = (\lambda_k - \lambda_{p+1})/(\lambda_k - \lambda_n)$, p is the block size, s is the number of steps, and λ_k, λ_{p+1}, and λ_n are the k-th, (p+1)-st, and n-th eigenvalues of A, the problem matrix of order n. While our attempts to formulate a precise strategy for choosing the block size using this term as a relative measure of effectiveness were largely unsuccessful, it does yield some qualitative information about how to choose the block size. That is, we try to choose p such that the difference $\lambda_k - \lambda_{p+1}$ is as large as possible and s is not too small. This suggests, for instance, that if several eigenvalues are clustered at one end of the spectrum with a gap between them and the remaining eigenvalues, that we should attempt to choose the block size at least as large as the number of eigenvalues in the cluster.

In conclusion, we suggest that simple strategies chosen along the lines suggested above will usually prove to be completely adequate in most applications.

5.4 Program and Storage Organization

It will be convenient to relate our discussion to the actual Fortran program contained in Appendix A. In particular, we will refer to the various parts of the program through the names of the subroutines.

There are certain auxilliary functions performed in the program which we will deal with first. During each iteration of the iterative Block Lanczos method, it is necessary to solve the eigenproblem for $\mathcal{M}_s$, the matrix of the restricted operator computed by the Block Lanczos method (cf. Section 2.7). This is accomplished via the subroutine EIGEN which simply restores $\mathcal{M}_s$ in such a manner that it is acceptable to the subroutines TRED2 and TQL2. These latter subroutines are designed to solve standard symmetric eigenproblems and are Fortran implementations [21] of Algol 60 procedures of the same name described in [24]. Note that $\mathcal{M}_s$ is also a band symmetric matrix with $2 \times p+1$ diagonals. Although there are special techniques and programs available for the solution of eigenproblems for band symmetric matrices, we found that their use did not conveniently allow us to reduce the amount of necessary time or storage. However, it is relatively unimportant which method is used to solve the eigenproblem for $\mathcal{M}_s$ as long as it is numerically stable.

In the Block Lanczos method we compute matrices Z_j and for each matrix, it is necessary to compute its orthogonal factorization:

$$Z_j = X_j R_j$$

where X_j is orthogonal and R_j is upper triangular. This is accomplished through the subroutine ORTHG which implements a stable variant of the Gram-Schmidt orthogonalization method. ORTHG was derived from the

Algol 60 procedure ORTHOG contained in the program for simultaneous iteration described by Rutishauser [20]. ORTHG has also been designed to carry out the functions of re-orthogonalization of the X_j (cf. Section 3.1) and projection of the X_j onto the space orthogonal to previously computed eigenvectors (cf. Section 2.9). ORTHG is also used to generate the initial matrix X used in the Block Lanczos method.

RANDOM is a subroutine used to fill the columns of an array with a pseudo-random sequence of real values. The resulting matrix of random elements is orthogonalized (using ORTHG) and sent to the subroutine SECTN. SECTN rotates the orthonormal matrix X , say, so that X^tAX is diagonal as follows: X is multiplied on the right by U where U is the orthonormal matrix of eigenvectors of $C = X^tAX$. If X is n-by-p and $d_1,d_2,\ldots,d_p$ are the eigenvalues of C , then

$$(XU)^tA(XU) = X^tU^tAUX = \text{diag}(d_1,d_2,\ldots,d_p) \quad .$$

The rotation of X can be accomplished through the subroutine ROTATE which is also used to compute the eigenvectors of the restricted operators (cf. Section 2.5) in the main subroutine using the matrices X_s and V where X_s and $\mathcal{M}_s$ are computed by the Block Lanczos method and V is composed of some of the eigenvectors of $\mathcal{M}_s$. That is, if v_i is an eigenvector of $\mathcal{M}_s$, then ROTATE is used to compute $\bar{q}_i$ where $\bar{q}_i = \mathcal{M}_s v_i$ for several values of i .

The principal part of the program is contained in the five subroutines called CNVTST, ERR, RCH, BKLANC, and MINVAL. CNVTST and ERR are fairly simple subroutines which implement the ideas of Section 3.2.

ERR is called in BKLANC, the subroutine for the Block Lanczos method, after Z_2 has been computed and before it has been orthogonalized with respect to previously computed eigenvectors and reorthogonalized with respect to X, the matrix used to start the Block Lanczos method. ERR simply computes the lengths ϵ_i of the columns of Z_2 which are the residual vectors for X. CNVTST is called from MINVAL, the main subroutine, and determines which of the ϵ_i satisfy the convergence criterion described in Section 3.3.

The purpose of PCH is to choose a new block size for the Block Lanczos method during the next iteration of the program if another iteration is necessary. The strategy of the specific subroutine contained in Appendix A is fairly simple: The block size p during each step will be the same as it was during the previous step unless fewer than p eigenvalues remain to be computed. In the latter case, p is set equal to the number of eigenvalues left to be computed. PCH also chooses a value for s, the number of steps the Block Lanczos method is carried out. The value of s is chosen so as to maximize the use of the storage available to the Block Lanczos method. If the block size p is such that fewer than two steps of the Block Lanczos method can be carried out because of storage limitations, then p is reduced to the point that s can be assigned a value of two.

The subroutine BKLANC implements the Block Lanczos method of Section 2.7 with reorthogonalization. If p is the block size, s is the number of steps, n is the order of the matrix A whose eigenvalues are being computed, and X is an n-by-p orthonormal matrix, the main purpose of BKLANC is to compute α_s and β_s, the representation of A

restricted to X_s where $Sp(X_s)$ contains $Sp(X,AX,\ldots,A^{s-1}X)$. Recall, also, that $\mathcal{T}_s$ is a $p\times s$-by-$p\times s$ symmetric block tridiagonal matrix with p-by-p matrices $M_1,M_2,\ldots,M_s$ along its diagonal and p-by-p upper-triangular p-by-p matrices $R_2,R_3,\ldots,R_s$ on its first lower diagonal. The matrix X_s is n-by-$p\times s$ so an array T, say, with <u>at least</u> $p\times s$ columns of length n is supplied to BKLANC to store X_s in. However, as we will see, previously computed eigenvector approximations are also stored in T. If m such approximations have been obtained, they are stored in the first m columns of T, and BKLANC stores X_s in columns m+1 through $m+p\times s$ of T. Note that if the actual dimensions of T are n-by-q where q is some integer value greater than one, that at any point in the execution of the program, m, p, and s must satisfy

$$m+p\times s \leq q \quad .$$

PCH chooses values for p and s with this restriction in mind. The initial n-by-p matrix X is stored in columns m+1 through m+p of T.

The computation performed by BKLANC is comprised of s major steps. During the j-th step, M_j, R_{j+1}, and X_{j+1} are computed except that during the s-th step, only M_s is computed. The matrix X is assumed to be such that $M_1 = X^tAX$ is diagonal. Advantage is taken of this in the first step. R_{j+1} and X_{j+1} are obtained by first forming Z_{j+1} (cf. Section 2.7) and storing it in T in the same location that X_{j+1} will occupy. ORTHG is then executed which orthogonalizes Z_{j+1} with respect to <u>all</u> previous vectors stored in T and decomposes the result into X_{j+1} and R_{j+1}. The

orthogonalization with respect to previous vectors in T accomplishes two ends: (1) Orthogonalization with respect to the first m columns implies that we are applying the Block Lanczos method to $\hat{A}$, the restriction of A to the space orthogonal to previously computed eigenvector approximations (cf. Section 2.9); (2) Orthogonalization with respect to the remaining columns of T accomplishes the re-orthogonalization of Z_{j+1} with respect to X_i, $i < j$ (cf. Section 3.1). M_s is stored in rows and columns m+1 through m + pXs of an array C, say. Since M_s is band symmetric, only its lower p+1 diagonals are stored in C.

Finally, the subroutine MINVAL is the main subroutine which combines the functions of the above subroutines into an implementation of the iterative Block Lanczos method of Section 2.10. An n-by-q array T is supplied to MINVAL which is used by BKLANC as described above and also to store the eigenvector approximations as they are computed. A variable r is used to count the number of computed values and vectors and when its value exceeds r, the required number of eigenvalues and vectors being sought, the program stops.

The initial size of the block size p is supplied to the program. In the preliminary phase of the program, the number of steps s is selected and the initial orthonormal matrix X is computed and rotated so that X^tAX is diagonal. The main part of the subroutine is a sequence of statements which carries out steps two through six of the algorithm described in Section 2.10. The main difference between the program and the description of Section 2.10 is that the computation of the eigenvectors $\bar{q}_i$ is put off until the end of the loop. This is because

both the eigenvectors which have converged and those that will be used to form the orthonormal matrix X for the next iteration must be computed simultaneously. Since the new block size is computed at the end of the iteration, the computation of the eigenvectors must be delayed until this point.

Information on the matrix A is passed into MINVAL through the name of a subroutine with three arguments. When the subroutine is called, one of the arguments will be an array containing a vector v , say. The subroutine computes the product A x v and stores it in a second array parameter. This is the only way the matrix A is referenced in the entire program.

The storage requirements of the program are as follows:

1. An n-by-q array T . T is used in BKLANC and also to store the computed eigenvectors. This array is supplied to MINVAL by the program which calls it. The value of q should be as large as possible, but, in any event, it should be at least one greater than r , the required number of eigenvalues and eigenvectors.

2. An array D of length at least q elements for storing the computed eigenvalues. This array is also supplied externally.

3. An array C with at least q^2 elements. C is used to store $\mathcal{M}_s$ in BKLANC and also to store the eigenvectors of $\mathcal{M}_s$ in EIGEN.

4. An array E with at least q elements for storing the norms of the residuals in ERR.

In addition to the above storage, the program also includes two arrays in the subroutine EIGEN with at least q elements for use with TRED2 and TQL2. Also, two arrays of length n are provided which are used with the subroutine for computing the matrix-vector product Ay where y is a vector. All these arrays were provided to make the program more flexible and usable and are to a certain extent, optional.

By far the bulk of the computation is performed by the subroutines BKLANC, EIGEN, and ROTATE so we will confine our operation count analysis to these three subroutines. The counts given below are for either additive and multiplicative operations and are for <u>one</u> step of the iterative Block Lanczos method. The terms n, m, p, s are the order of A, the number of previously computed vectors, the block size and the number of steps for the Block Lanczos method, respectively.

1. BKLANC:

   Computation of M_i's: $\frac{np(p+1)(s-1)}{2}$

   Computation of Z_i's: $np + \frac{n(3p^2+p(s-1)}{2}$

   Computation of X_i's and R_i's: $2nmps + np^2s^2 + nps$

2. EIGEN (using TRED2 and TQL2): approx. $2(ps)^3$

3. ROTATE: np^2s.

In addition, there are ps matrix-vector products computed in BKLANC. This computation is performed externally and depends on the matrix A. Depending on the problem, it may completely overshadow the rest of the computation or it may be insignificant in comparison.

Example. Suppose $n = 1000$, $p = 5$, $s = 4$, and $m = 3$. We then have the following counts:

1. BKLANC:

   Computation of M_i's: 45,000;

   Computation of Z_i's: 120,000;

   Computation of X's and R's: 540,000;

2. EIGEN: 24,000.

3. ROTATE: 100,000.

In addition, there are 25 matrix-vector products involving A which is of order 1000.

From this example, we see that the bulk of the computation takes place in BKLANC, and in particular, in the computation of the X_i's and R_i's which involves the orthogonalization of the Z_i's. This example is fairly representative of the situation in general.

The above operation counts don't really say anything about the overall running time of the program since this depends on how fast the computed eigenvalues and vectors converge to those of A. The rates of convergence in turn depend on the spectrum of A. In the next chapter, we will consider some specific examples and compare our algorithm with the method of simultaneous iteration.

4. NUMERICAL EXAMPLES

In this chapter, we will consider the results of applying the iterative Block Lanczos algorithm to a number of examples. We will also compare some of our results with those obtained using the method of simultaneous iteration described by Rutishauser [20].

For the purposes of testing our method and the method of simultaneous iteration, diagonal matrices are sufficiently general and particularly convenient. That they are sufficiently general arises from the fact that neither of the above methods transforms the matrix A whose eigenvalues are being computed or in any other way attempts to take into account the structure of A. Rather, the only way A is referenced is through a subroutine which computes $v = Au$ where u is a given vector. If

$$A = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

where n is the order of A, then

$$v_i = \lambda_i u_i$$

where u_i and v_i are the i-th components of u and v, respectively. This can be easily programmed and a large number of different examples can be quickly generated whose exact eigenvalues and eigenvectors are known. More important than knowing the spectrum is the fact that we can specify the spectrum of A and therefore can study the behavior of our method from the standpoint of test examples whose spectrums vary according to the separations and multiplicities of their eigenvalues. All but one of our examples will be chosen from this class of problems.

It is somewhat difficult to compare the iterative Block Lanczos method to the method of simultaneous iteration since the computations they perform are different. One measure is the total time each requires to solve a particular problem but this standard is rather crude and uninformative. There are two areas, however, in which the computations performed by the two methods coincide -- the computation of matrix-vector products Ay where y is a vector, and the orthogonalization of the columns of a matrix which involves computing a large number of vector inner products. As we saw (cf. Section 3.4) a major part of the computation time in the iterative Block Lanczos method is spent in these two areas and the same is true of simultaneous iteration. Thus, for the purposes of comparing the two methods, we will report the following:

1. The computed eigenvalues μ_i.
2. The magnitudes ϵ_i of the residual vectors $Ax_i - \mu_i x_i$ where x_i is the eigenvector corresponding to μ_i.
3. The number of matrix-vector products computed.
4. The number of vector inner products computed in the orthogonalization routines.
5. The total execution time for the entire program.

Additionally, for the iterative Block Lanczos method, we will report the number of iterations required which is also the number of times the Block Lanczos method per se is carried out.

A listing of our iterative Block Lanczos program is contained in Appendix A. The output statements used to print out program statistics have been deleted from this listing.

Our program for simultaneous iteration is a Fortran translation of the Algol 60 procedure ritzit described by Rutishauser [19,20]. This procedure is actually a combination of simultaneous iteration and a Chebyshev type iteration. The biggest difficulty in using this program as a standard for comparison is that it often overshoots its goal. That is, it either computes results far more accurately than desired (taking more time in the process) or it computes more eigenvalues and eigenvectors than asked for. With our method, it is far easier to control both the precision and number of results computed. Thus, our plan has been to perform a computation with the simultaneous iteration program and then attempt to match its results in some sense using our iterative Block Lanczos program.

Note: While our version of simultaneous iteration is a nearly literal translation of the Algol 60 procedure, there are some minor differences between the two. The differences arise from our attempts to rectify some errors in the published version of the Algol 60 procedure, and to clarify the structure of the program which was very complicated at the start. No essential change was made in the algorithm, however, which would compromise its efficiency.

We now proceed to the examples. In each example, we will specify values for r, the required number of eigenvalues and eigenvectors to be computed, and q, the number of columns of length n in an array X which is used in both the iterative Block Lanczos method and the method of simultaneous iteration. For both methods, the value of q must be greater than that of r. In addition, for our method, we will give values for eps, the approximate precision desired in our computed

results, and p, the block size to be used in the Block Lanczos method. The strategy used for choosing the block size is as described in Section 3.4. That is, the block size is chosen to be the least of the two following values: (1) The initial block size p; and, (2) the number of eigenvalues left to be computed. This strategy is implemented by the program listed in Appendix A.

In Examples 1, 2, and 3, we will compare our method with the method of simultaneous iteration when both are applied to problems with characteristic types of spectra. In Examples 1 and 2, we will consider problems for which our method is more effective. Example 3, in contrast, favors simultaneous iteration. In Examples 4 and 5, we will consider the behavior of our method on matrices with multiple eigenvalues. Example 6 involves a matrix with very close eigenvalues. Finally, in Example 7, we will consider the results of applying both programs to the problem of computing the least eigenvalues and eigenvectors of the discrete biharmonic operator.

Example 1.

A is a (diagonal) matrix of order 454 with eigenvalues $\lambda_1 = -10.00$, $\lambda_2 = -9.99$, $\lambda_3 = -9.98$, and $\lambda_i = -9.00 + .02 \times (i-4)$ for $i = 4,5,\ldots,454$. With $q = 15$, $r = 3$, $p = 3$, and $eps = 10^{-8}$, the iterative Block Lanczos program computed

$$\mu_1 = -9.99\ 999\ 999\ 999\ 996 \quad , \quad \epsilon_1 = 1.73 \times 10^{-8} \quad ,$$

$$\mu_2 = -9.98\ 999\ 999\ 999\ 994 \quad , \quad \epsilon_2 = 2.85 \times 10^{-8} \quad ,$$

$$\mu_3 = -9.97\ 999\ 999\ 999\ 991 \quad , \quad \epsilon_3 = 2.11 \times 10^{-8} \quad .$$

Note that $|\epsilon_i/\mu_i| < 10^{-8}$ in each instance. Eleven iterations were required for this computation.

In contrast, the program for simultaneous iteration computed

$$\mu_1 = -9.99\ 999\ 999\ 999\ 995\ , \quad \epsilon_1 = 3.09 \times 10^{-6}\ ,$$

$$\mu_2 = -9.98\ 999\ 999\ 999\ 999\ , \quad \epsilon_2 = 1.09 \times 10^{-6}\ ,$$

$$\mu_3 = -9.97\ 999\ 999\ 999\ 992\ , \quad \epsilon_3 = 2.21 \times 10^{-6}\ .$$

Note that the values of the ϵ_i are greater in this case.

Table 4.1 summarizes the comparative statistics for the two programs.

Table 4.1

	matrix-vector products	vector inner products	time (sec.)	relative precision
Block Lanczos	165	1265	45.95	approx. 10^{-8}
Sim. Iter.	750	1560	69.03	approx. 10^{-6}

Times, unless otherwise indicated, will be total execution times for programs compiled using the University of Waterloo Watfiv Fortran compiler and executed on an I.B.M. System 370/168 computer.

This example is typical of those problems in which the iterative Block Lanczos algorithm can be used to best advantage. In particular, problems in which the eigenvalues to be computed are separated from the remaining eigenvalues by a relatively large gap.

Note in each case that the eigenvalues are about twice as accurate as the ϵ_i indicate.

Example 2.

The matrix in this example is the same as in Example 1 except that the gaps between the eigenvalues have been decreased by a factor of 10. That is, $\lambda_1 = -10.00$, $\lambda_2 = -9.999$, $\lambda_3 = -9.998$, $\lambda_i = -9.900 + (i-4)\times.002$, $i = 4,5,\ldots,454$. As before, $r = 3$ and $q = 15$.

With $p = 3$ and $eps = 10^{-8}$, the iterative Block Lanczos program computed in 10 iterations the following results:

$$\mu_1 = -9.99\ 999\ 999\ 999\ 996 \quad , \quad \epsilon_1 = 8.96 \times 10^{-8} \quad ,$$

$$\mu_2 = -9.99\ 000\ 000\ 000\ 002 \quad , \quad \epsilon_2 = 1.97 \times 10^{-8} \quad ,$$

$$\mu_3 = -9.99\ 799\ 999\ 999\ 999 \quad , \quad \epsilon_3 = 1.39 \times 10^{-8} \quad .$$

The simultaneous iteration program computed

$$\mu_1 = -9.99\ 999\ 999\ 947\ 815 \quad , \quad \epsilon_1 = 7.91 \times 10^{-6} \quad ,$$

$$\mu_2 = -9.99\ 899\ 999\ 997\ 082 \quad , \quad \epsilon_2 = 1.92 \times 10^{-6} \quad ,$$

$$\mu_3 = -9.99\ 799\ 999\ 920\ 753 \quad , \quad \epsilon_3 = 9.61 \times 10^{-6} \quad .$$

In each case, the errors in the computed eigenvectors were approximately the size of the ϵ_i's.

This example produced the results given in Table 4.2.

Table 4.2

	matrix-vector products	vector inner products	time (sec.)	relative precision
Block Lanczos	149	1140	41.80	10^{-8}
Sim. Iter.	1785	1800	88.94	10^{-6}

The behavior of the iterative Block Lanczos program was virtually unaffected by the reduction in the spread of the eigenvalues. This example serves to illustrate the point that the rates of convergence of the approximations depend on the gaps between the eigenvalues relative to the spread of the eigenvalues. This is suggested by Theorem 2.6.1 in which the bounds on the errors in the μ_i depended on the eigenvalues through the quantities γ_i where

$$\gamma_i = \frac{(\lambda_i - \lambda_{p+1})}{(\lambda_i - \lambda_n)} \quad .$$

Decreasing the gaps between all the eigenvalues by a constant factor does not affect the value of γ_i .

The simultaneous iteration program, however, suffered by this change since the results it computes converge at rates that depend on the ratios λ_{p+1}/λ_i which increase when the eigenvalues were brought closer together.

Example 3.

A is a (diagonal) matrix of order 101 with eigenvalues equally spaced in the interval [-1.0 , 0.0] . That is, $\lambda_i = -(101 - i)/100$, $i = 1,2,\ldots,101$. In this example, we have $r = 6$ and $q = 10$. In addition, we choose $p = 2$ and $eps = 10^{-5}$ for the Block Lanczos method. The iterative Block Lanczos program then computed six eigenvalues:

$$\mu_1 = -.999\ 999\ 999\ 7\ldots \quad , \quad \epsilon_1 = .925 \times 10^{-5} \ ,$$

$$\mu_2 = -.989\ 999\ 998\ 3\ldots \quad , \quad \epsilon_2 = 1.494 \times 10^{-5} \ ,$$

$$\mu_3 = -.980\ 000\ 000\ 3\ldots \quad , \quad \epsilon_3 = 1.29 \times 10^{-5} \ ,$$

$$\mu_4 = -.970\ 000\ 000\ 7\ldots \quad , \quad \epsilon_4 = 1.53 \times 10^{-5} \ ,$$

$$\mu_5 = -.959\ 999\ 999\ 6\ldots \quad , \quad \epsilon_5 = 2.07 \times 10^{-5} \ ,$$

$$\mu_6 = -.949\ 999\ 998\ 7\ldots \quad , \quad \epsilon_6 = 1.75 \times 10^{-5} \ .$$

Note that the residuals exceeded 10^{-5} in the last five eigenvalues. This was because of the allowance made for the error in the first eigenvalue and eigenvector.

The simultaneous iteration program computed seven (even though only six were asked for):

$$\mu_1 = -1.000\ 000\ 000\ 000 \quad , \quad \epsilon_1 = 2.13 \times 10^{-9} \ ,$$

$$\mu_2 = -.989\ 999\ 999\ 999 \quad , \quad \epsilon_2 = 2.56 \times 10^{-9} \ ,$$

$$\mu_3 = -.979\ 999\ 999\ 999 \quad , \quad \epsilon_3 = 6.75 \times 10^{-9} \ ,$$

$$\mu_4 = -.969\ 999\ 999\ 994 \quad , \quad \epsilon_4 = 2.21 \times 10^{-8} \ ,$$

$$\mu_5 = -.959\ 999\ 999\ 466 \quad , \quad \epsilon_5 = 1.91 \times 10^{-7} \ ,$$

$$\mu_6 = -.949\ 999\ 999\ 993 \quad , \quad \epsilon_6 = 2.02 \times 10^{-8} \ ,$$

$$\mu_7 = -.939\ 999\ 999\ 959 \quad , \quad \epsilon_7 = 4.54 \times 10^{-8} \ .$$

The comparative statistics for the two programs are given in Table 4.3.

Table 4.3

	matrix-vector products	vector inner-products	time (sec.)	precision
Block Lanczos	350	1974	17.96	approx. 10^{-5}
Sim. Iter.	625	795	9.13	approx. 10^{-7} or less

This example is typical of the type of problem for which simultaneous iteration performs better than the iterative Block Lanczos method. That is, problems for which the spectrum is fairly dense with little or no distance between those eigenvalues being sought and the remaining eigenvalues, and for which r is a large fraction of q.

Example 4.

A is a (diagonal) matrix of order 180 with eigenvalues $\lambda_1 = \lambda_2 = 0.0$, $\lambda_3 = \lambda_4 = 0.1$, and $\lambda_i = .25 + (i-5)\times.01$, $i = 5,6,\ldots,180$. Thus, $\lambda_{180} = 2.00$. No comparison with simultaneous iteration was made as the simultaneous iteration program computes the eigenvalues of greatest modulus which are different in this case from the least eigenvalues.

For this example we chose $q = 10$, $r = 4$, $eps = 10^{-4}$ and tried $p = 1$, 2, 3, and 4. For each of the four values of p, the iterative Block Lanczos method computed four eigenvalues with residuals on the order of 10^{-4}. For instance for $p = 2$, we computed

$$\mu_1 = 5.38 \times 10^{-9} \quad , \quad \mu_1 = .583 \times 10^{-4} \quad ,$$

$$\mu_2 = 1.57 \times 10^{-9} \quad , \quad \mu_2 = .644 \times 10^{-4} \quad ,$$

$$\mu_3 = .100\ 000\ 000\ 9\ldots \quad , \quad \mu_3 = 1.517 \times 10^{-4} \quad ,$$

$$\mu_4 = .999\ 999\ 997\ 9\ldots \quad , \quad \mu_4 = .628 \times 10^{-4} \quad .$$

The results for the other three cases are similar. The largest residual in any case was approximately 1.59×10^{-4}. Note that for this case, the error is absolute and not relative error. The eigenvectors for μ_1 and μ_2 were primarily linear combinations of e_1 and e_2 and the errors in the remaining components were in all cases approximately the size of the residual or less. Similarly for μ_3 and μ_4.

The comparative statistics for the four values of p are as follows:

	iterations	matrix-vector products	vector inner products	time (sec.)
$p = 1$	20	158	997	13.19
$p = 2$	15	125	725	11.15
$p = 3$	17	140	699	12.50
$p = 4$	33	317	1330	29.53

Note that there was a definite improvement from $p = 1$ to $p = 2$. Multiple eigenvalues tend to slow down the standard Lanczos algorithm ($p = 1$) since the eigenvalues of the restricted operator will converge to only one of a set of multiple roots. With $q = 10$, effective use of all the working storage could not be made with $p = 3$ or $p = 4$ (since neither divides 10 evenly). However, with the program listed

in Appendix A, a block size of 5 was chosen after the first iteration with $p = 4$. Even though this change made better use of storage, more time was required here than in the other three cases.

Example 5.

A is a (diagonal) matrix of order 300 with eigenvalues $\lambda_1 = 0.0$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, $\lambda_4 = 0.1$, $\lambda_i = 1 - 3/(i-1)$ for $i = 5,6,\ldots,300$. For this example, we choose $r = 3$, $q = 12$, and $eps = 10^{-3}$. We tried $p = 1, 2, 3$. Of the three values, the fastest execution was achieved for $p = 3$. In four iterations, the following results were computed:

$$\mu_1 = .000\ 000\ 002\ 922\ 313 \quad , \quad \epsilon_1 = .327 \times 10^{-3} \quad ,$$

$$\mu_2 = .100\ 000\ 000\ 000\ 004 \quad , \quad \epsilon_2 = .149 \times 10^{-5} \quad ,$$

$$\mu_3 = .100\ 000\ 000\ 090\ 571 \quad , \quad \epsilon_3 = .110 \times 10^{-3} \quad .$$

The statistics for this computation are as follows:

matrix-vector products:	36
vector inner products:	288
time (secs.):	7.86

The eigenvectors for μ_2 and μ_3 were primarily combinations of e_2 and e_3, the unit vectors with ones in the second and third positions, respectively. The errors in these vectors are again proportional to the sizes of the residuals ϵ_i.

The fact that for $p = 3$, $\lambda_p = \lambda_{p+1}$ shows that Theorem 2.6.1 on the rates of convergence does not adequately explain all situations.

In fact, λ_2 converged much more rapidly than λ_1, leading us to conjecture that the rate of convergence for λ_2 involves $\lambda_5 = .25$ rather than $\lambda_{p+1} = \lambda_4 = .1$. We have, however, found no way of establishing this conjecture.

Example 6.

A is the same matrix as in Example 5 except that $\lambda_2 = .0999999$, $\lambda_3 = .1000000$, and $\lambda_4 = .1000001$. In this example, $r = 4$, $q = 12$, and $eps = 10^{-3}$. Our iterative Block Lanczos program computed

$$\mu_1 = .000\ 000\ 057 \quad , \quad \epsilon_1 = 9.12 \times 10^{-4} \quad ,$$

$$\mu_2 = .099\ 999\ 926 \quad , \quad \epsilon_2 = 1.93 \times 10^{-4} \quad ,$$

$$\mu_3 = .100\ 000\ 008 \quad , \quad \epsilon_3 = 9.47 \times 10^{-4} \quad ,$$

$$\mu_4 = .100\ 000\ 078 \quad , \quad \epsilon_4 = 1.42 \times 10^{-6} \quad .$$

Note that in the case of μ_4, the error is approximately of order ϵ_4 rather than ϵ_4^2. Furthermore, the eigenvectors for μ_2, μ_3 and μ_4 each contained significant components of the eigenvectors corresponding to λ_2, λ_3, and λ_4. Basically, the program regards λ_2, λ_3, and λ_4 as multiple roots and any combination of their eigenvectors as an eigenvector also. Each computed eigenvector of μ_2, μ_3, and μ_4 was very close, therefore, to the space spanned by the eigenvectors corresponding to λ_2, λ_3, and λ_4.

These results are not indicative of a defect in our algorithm but represent inherent limits in the accuracy obtainable for eigenvectors corresponding to very close but distinct roots (cf. Section 1.2).

The statistics in this case are as follows:

1. matrix-vector products: 54
2. vector inner products: 408
3. time (secs.): 11.29
4. iterations: 6 .

Example 7.

The natural modes of vibration of a square clamped elastic plate can be solved by the following partial differential equation for ω and λ :

$$\frac{\partial^4 \omega}{\partial x^2} + \frac{2\partial^4 \omega}{\partial^2 x \partial^2 y} + \frac{\partial^4 \omega}{\partial y^4} = \lambda \omega(x,y) \qquad (4.1)$$

in the interior of the plate with

$$\omega = 0 = \text{normal derivative of } \omega$$

on the boundary. If we attempt to compute a discrete approximation to the solution of the above equation at the points of a mesh superimposed on the plate, then we are led to a symmetric eigenproblem

$$\frac{1}{h^4} Hx = \lambda x$$

where h is the mesh width and H is a symmetric block pentadiagonal matrix derived using a 13-point finite difference approximation to the differential operator in Equation 4.1 [1]. Rather than compute the eigenvalues of H, we will compute the eigenvalues of $A = -H^{-1}$. Note that if $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of H, then

$\nu_1 \le \nu_2 \le \cdots \le \nu_n$ where $\nu_i = -1/\lambda_i$ are the eigenvalues of A. Thus, if we compute the least eigenvalues of A, their negative reciprocals will be the least eigenvalues of H with the same eigenvectors. To compute $y = Ax$ given x, we solve

$$Hz = x \tag{4.2}$$

and set

$$y = -z \quad .$$

To solve Equation (4.2), we use a program provided by Dr. Fred Dorr of the Los Alamos Scientific Laboratory. This program is based on an idea of Buzbee and Dorr [2] and computes a solution to Equation (4.2) by a direct method.

For a unit square and a mesh width of $h = 1/33$, the order of A is 32^2 or 1024. For this problem we choose $q = 16$, $r = 12$. With $p = 3$ and $eps = 10^{-4}$, our program computed the results given in Table 4.4 where for each eigenvalue ν of A, the corresponding frequency f of vibration of the plate satisfies $f = 1/\sqrt{-h^4\nu}$.

Table 4.4

i	eigenvalue of A	residual ϵ_i	frequency of vibration
1	-923.91655	0.65×10^{-3}	35.82709
2	-223.74996	0.21×10^{-1}	72.80252
3	-223.74996	0.25×10^{-3}	72.80252
4	-193.22429	0.96×10^{-4}	107.18577
5	-70.42548	1.03×10^{-3}	129.76846
6	-69.75160	1.82×10^{-4}	130.41065
7	-44.77957	1.29×10^{-3}	162.73760
8	-44.77956	3.06×10^{-3}	162.73760
9	-27.90840	1.42×10^{-2}	206.13913
10	-27.90839	2.05×10^{-2}	206.13917
11	-25.29523	1.87×10^{-2}	216.52528
12	-21.06754	3.83×10^{-2}	237.25804

Because of the allowance made for error in previously computed eigenvalues and vectors, the relative error in the last four eigenvalues exceeds 10^{-4} and for the last eigenvalue is approximately 2×10^{-3} .

The frequencies computed from the eigenvalues of A correspond closely to those reported by Bauer and Reiss in [1]. Note that eigenvalues 2 and 3 , 7 and 8 , and 9 and 10 are multiple with multiplicity 2 . Rough graphs of the eigenvectors verify that they describe fundamental modes of vibration very similar to those reported in [1].

The simultaneous iteration program could not complete its computation in the two minutes of time allotted to it. Thus, we increased the mesh width to 1/25 which lowered the order of A to 576 . The program then computed the 13 least eigenvalues of A to relative precisions ranging from approximately 10^{-14} to 10^{-4} . Both the frequencies and fundamental modes of vibration described by the eigenvectors computed here are what one would expect based on the results reported in [1].

Table 4.5 summarizes the results of the two programs.

Table 4.5

	Order of A	matrix-vector products	vector inner products	time (sec.)	no. of eigen-values	rel. precision
Block Lanczos	1024	145	1233	85.52	12	approx. 10^{-6} to 10^{-3}
Sim. Iter.	576	223	1752	69.92	13	approx. 10^{-14} to 10^{-4}

A total of 19 iterations were required for the iterative Block Lanczos method.

As we pointed out at the start, it is difficult to use the simultaneous iteration program as a standard of comparison since it is hard to control the number and precision of the results it computes. However, we would say that our program did a better job of completing its assigned task of computing a specified number of eigenvalues to a specified precision.

APPENDIX: A FORTRAN PROGRAM

This Appendix contains a listing of the program for the iterative Block Lanczos method. See Section 3.4 for a discussion of the program. A sample driver program is included with a test problem.

```
      SUBROUTINE MINVAL (N,Q,PINIT,R,MMAX,EPS,OP,M,D,X,IECODE)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,PINIT,R,MMAX
      REAL*8 EPS
      EXTERNAL OP
      DIMENSION D(Q),X(N,Q)
      INTEGER IECODE
C
C     THIS SUBROUTINE IS THE MAIN SUBROUTINE
C     IMPLEMENTING THE ITERATIVE BLOCK LANCZOS METHOD
C     FOR COMPUTING THE EIGENVALUES AND EIGENVECTORS OF
C     SYMMETRIC MATRICES.
C
C     DESCRIPTION OF PARAMETERS:
C
C     N:      INTEGER VARIABLE.  THE ORDER OF THE SYMMETRIC
C             MATRIX A WHOSE EIGENVALUES AND EIGENVECTORS ARE
C             BEING COMPUTED.  THE VALUE OF N SHOULD BE LESS THAN
C             OR EQUAL TO 1296 AND GREATER THAN OR EQUAL TO 2.
C
C     Q:      INTEGER VARIABLE.  THE NUMBER OF VECTORS OF LENGTH
C             N CONTAINED IN THE ARRAY X.  THE VALUE OF Q SHOULD
C             BE LESS THAN OR EQUAL TO 25, AT LEAST ONE
C             GREATER THAN THE VALUE OF R AND LESS THAN OR
C             OR EQUAL TO N.
C
C     PINIT:  INTEGER VARIABLE.  THE INITIAL BLOCK SIZE TO BE
C             USED IN THE BLOCK LANCZOS METHOD.  IF PINIT IS
C             NEGATIVE, THEN -PINIT IS USED FOR THE BLOCK SIZE
C             AND COLUMNS M+L, . . . , M+(-PINIT) OF THE ARRAY X
C             ARE ASSUMMED TO BE INITIALIZED TO THE INITIAL
C             MATRIX USED TO START THE BLOCK LANCZOS METHOD.  IF
C             THE SUBROUTINE TERMINATES WITH A VALUE OF M LESS
C             THAN R, THEN PINIT IS ASSIGNED A VALUE -P WHERE P
C             IS THE FINAL BLOCK SIZE CHOSEN.  IN THIS
C             CIRCUMSTANCE, COLUMNS M+1, ... , M+P WILL CONTAIN
C             THE MOST RECENT SET OF EIGENVECTOR APPROXIMATIONS
C             WHICH CAN BE USED TO RESTART THE SUBROUTINE IF
C             DESIRED.
C
C     R:      INTEGER VARIABLE.  THE NUMBER OF EIGENVALUES AND
C             AND EIGENVECTORS BEING COMPUTED.  THAT IS, MINVAL
C             ATTEMPTS TO COMPUTE ACCURATE APPROXIMATIONS TO THE
C             R LEAST EIGENVALUES AND EIGENVECTORS OF THE MATRIX
C             A.  THE VALUE OF R SHOULD BE GREATER THAN
C             ZERO AND LESS THAN Q.
C
C     MMAX:   INTEGER VARIABLE.  THE MAXIMUM NUMBER OF MATRIX-
C             VECTOR PRODUCTS A*X WHERE X IS A VECTOR THAT ARE
C             ALLOWED DURING ONE CALL OF THIS SUBROUTINE TO
C             COMPLETE ITS TASK OF COMPUTING R EIGENVALUES AND
C             EIGENVECTORS.  UNLESS THE PROBLEM INDICATES
C             OTHERWISE, MMAX SHOULD BE GIVEN A VERY LARGE VALUE.
C
C     EPS:    REAL*8 VARIABLE.  INITIALLY, EPS SHOULD CONTAIN A
C             VALUE INDICATING THE RELATIVE PRECISION TO WHICH
C             MINVAL WILL ATTEMPT TO COMPUTE THE EIGENVALUES AND
C             EIGENVECTORS OF A.  FOR EIGENVALUES LESS IN MODULUS
C             THAN 1, EPS WILL BE AN ABSOLUTE TOLERANCE.  BECAUSE
C             OF THE WAY THIS METHOD WORKS, IT MAY HAPPEN THAT
C             THE LATER EIGENVALUES CANNOT BE COMPUTED TO THE
C             SAME RELATIVE PRECISION AS THOSE LESS IN VALUE.
C
C     OP:     SUBROUTINE NAME.  THE ACTUAL ARGUMENT CORRESPONDING
C             TO OP SHOULD BE THE NAME OF A SUBROUTINE USED TO
C             DEFINE THE MATRIX A.  THIS SUBROUTINE SHOULD HAVE
C             THREE ARGUMENTS N, U, AND V, SAY, WHERE N IS AN
C             INTEGER VARIABLE GIVING THE ORDER OF A AND U AND V
C             ARE TWO ONE-DIMENSIONAL ARRAYS OF LENGTH N.  IF W
```

```
C           DENOTES THE VECTOR OF ORDER N STORED IN U, THEN THE
C           STATEMENT
C
C                                CALL OP (N,U,V)
C
C           SHOULD RESULT IN THE VECTOR A*W BEING COMPUTED AND
C           STORED IN V.  THE CONTENTS OF U CAN BE MODIFIED BY
C           THIS CALL.
C
C     M:    INTEGER VARIABLE.  M GIVES THE NUMBER OF
C           EIGENVALUES AND EIGENVECTORS ALREADY COMPUTED.
C           THUS, INITIALLY, M SHOULD BE ZERO.  IF M IS GREATER
C           THAN ZERO, THEN COLUMNS ONE THROUGH M OF THE ARRAY
C           X ARE ASSUMED TO CONTAIN THE COMPUTED
C           APPROXIMATIONS TO THE M LEAST EIGENVALUES AND
C           EIGENVECTORS OF A.  AT EXIT, M CONTAINS A
C           VALUE EQUAL TO THE TOTAL NUMBER OF EIGENVALUES AND
C           EIGENVECTORS COMPUTED INCLUDING ANY ALREADY
C           COMPUTED WHEN MINVAL WAS ENTERED.  THUS, AT EXIT,
C           THE FIRST M ELEMENTS OF D AND THE FIRST M COLUMNS
C           OF X WILL CONTAIN APPROXIMATIONS TO THE M LEAST
C           EIGENVALUES OF A.
C
C     D:    REAL*8 ARRAY.  D CONTAINS THE COMPUTED EIGENVALUES.
C           D SHOULD BE A ONE DIMENSIONAL ARRAY WITH AT LEAST Q
C           ELEMENTS.
C
C     X:    REAL*8 ARRAY.  X CONTAINS THE COMPUTED
C           EIGENVECTORS.  X SHOULD BE AN ARRAY CONTAINING AT
C           LEAST N*Q ELEMENTS.  X IS USED NOT ONLY TO STORE
C           THE EIGENVECTORS COMPUTED BY MINVAL, BUT ALSO AS
C           WORKING STORAGE FOR THE BLOCK LANCZOS METHOD.  AT
C           EXIT, THE FIRST N*M ELEMENTS OF X CONTAIN THE
C           EIGENVECTOR APPROXIMATIONS-- THE FIRST VECTOR IN
C           THE FIRST N ELEMENTS, THE SECOND IN THE SECOND N
C           ELEMENTS, ETC.
C
C     IECODE: INTEGER VARIABLE.  THE VALUE OF IECODE INDICATES
C           WHETHER MINVAL TERMINATED SUCCESSFULLY, AND IF NOT,
C           THE REASON WHY.
C
C                IECODE=0 :  SUCCESSFUL TERMINATION.
C                IECODE=1 :  THE VALUE OF N IS LESS THAN TWO.
C                IECODE=2 :  THE VALUE OF N EXCEEDS 1296.
C                IECODE=3 :  THE VALUE OF R IS LESS THAN ONE.
C                IECODE=4 :  THE VALUE OF Q IS LESS THAN OR
C                            EQUAL TO R.
C                IECODE=5 :  THE VALUE OF Q IS GREATER THAN 25.
C                IECODE=6 :  THE VALUE OF Q EXCEEDS N.
C                IECODE=7 :  THE VALUE OF MMAX WAS EXCEEDED
C                            BEFORE R EIGENVALUES AND
C                            EIGENVECTORS WERE COMPUTED.
C
C     NOTE THAT THE SUBROUTINE HAS BEEN DESIGNED TO ALLOW INITIAL
C     APPROXIMATIONS TO THE EIGENVECTORS CORRESPONDING TO THE LEAST
C     EIGENVALUES TO BE UTILIZED IF THEY ARE KNOWN (BY STORING THEM
C     IN X AND ASSIGNING PINIT MINUS THE VALUE OF THEIR NUMBER.)
C     FURTHERMORE, IT HAS ALSO BEEN DESIGNED TO ALLOW RESTARTING
C     IF IT STOPS WITH IECODE=7.  THUS, THE USER OF THIS PROGRAM CAN
C     RESTART IT AFTER EXAMINING ANY PARTIAL RESULTS WITHOUT LOSS
C     OF PREVIOUS WORK.
C
      DIMENSION E(25),C(25,25)
      DIMENSION U(1296),V(1296)
      INTEGER P,S,PS
C
C
```

```
C     CHECK THAT THE INITIAL VALUES OF THE SUBROUTINE
C     PARAMETERS ARE IN RANGE.
C
      IF (N.LT.2) GO TO 901
      IF (N.GT.1296) GO TO 902
      IF (R.LT.1) GO TO 903
      IF (Q.LE.R) GO TO 904
      IF (Q.GT.25) GO TO 905
      IF (Q.GT.N) GO TO 906
C
C     CHOOSE INITIAL VALUES FOR THE BLOCK SIZE P, THE NUMBER
C     OF STEPS THE BLOCK LANCZOS METHOD IS CARRIED OUT, AND
C     CHOOSE AN INITIAL N-BY-P ORTHONORMAL MATRIX X1 USED TO
C     START THE BLOCK LANCZOS METHOD.
C
      P=PINIT
      IF (P.LT.0) P=-P
      S=(Q-M)/P
      IF (S.GT.2) GO TO 100
      S=2
      P=Q/2
C
  100 IF (PINIT.LT.0) GO TO 150
      DO 120 K=1,P
  120 CALL RANDOM(N,Q,K,X)
C
  150 IF (M.GT.0) GO TO 200
C
      CALL ORTHG(N,Q,M,P,C,X)
C
C     ROTATE THE INITIAL N-BY-P MATRIX X1 SO THAT
C
C           X1'*A*X1 = DIAG(D1,D2, ... , DP)
C
C     WHERE DI IS STORED IN D(I), I=1, ... , P.
C
      CALL SECTN(N,Q,M,P,OP,X,C,D,U,V)
C
      ERRC=0.0D0
  200 ITER=0
      IMM=0
C
C     THE MAIN BODY OF THE SUBROUTINE STARTS HERE.  IMM
C     COUNTS THE NUMBER OF MATRIX-VECTOR PRODUCTS COMPUTED
C     WHICH IS THE NUMBER OF TIMES THE SUBROUTINE NAMED BY
C     OP IS CALLED.  ERRC MEASURES THE ACCUMULATED ERROR IN
C     THE EIGENVALUES AND EIGENVECTORS.
C
  300 IF (M.GE.R) GO TO 900
      IF (IMM.GT.MMAX) GO TO 907
      ITER=ITER+1
      PS=P*S
C
C     BKLANC CARRIES OUT THE BLOCK LANCZOS METHOD AND
C     STORES THE RESULTING BLOCK TRIDIAGONAL MATRIX MS IN C
C     AND THE N-BY-PS ORTHONORMAL MATRIX XS IN X.  THE
C     INITIAL N-BY-P ORTHONORMAL MATRIX IS ASSUMED TO
C     BE STORED IN COLUMNS M+1 THROUGH M+PS OF X.  THE
C     RESIDUALS FOR THESE VECTORS AND THE EIGENVALUES
C     APPROXIMATIONS IN D ARE COMPUTED AND STORED IN E.
C
      CALL BKLANC (N,Q,M,P,S,OP,D,C,X,E,U,V)
```

```
C
C       EIGEN SOLVES THE EIGENPROBLEM FOR MS, STORING THE
C       EIGENVALUES IN ELEMENTS M+1 THROUGH M+PS OF D
C       AND THE EIGENVECTORS IN THE FIRST P*S ROWS AND
C       COLUMNS OF C (OVERWRITING MS, POSSIBLY.)
C
        CALL EIGEN(Q,M,P,PS,C,D)
C
C       CNVTST DETERMINES HOW MANY OF THE EIGENVALUES AND
C       EIGENVECTORS HAVE CONVERGED USING THE ERROR ESTIMATES
C       STORED IN E.  THE NUMBER THAT HAVE CONVERGED IS STORED
C       IN NCONV.  IF NCONV=0, THEN NONE HAVE CONVERGED.
C
        CALL CNVTST(N,Q,M,P,ERRC,EPS,D,E,NCONV)
C
C       PCH CHOOSES NEW VALUES FOR P AND S, THE BLOCK SIZE AND
C       THE NUMBER OF STEPS FOR THE BLOCK LANCZOS SUBPROGRAM,
C       RESPECTIVELY.
C
        CALL PCH(N,Q,M,R,NCONV,P,S)
C
C       ROTATE COMPUTES THE EIGENVECTORS OF THE RESTRICTED
C       OF THE RESTRICTED MATRIX USING XS STORED IN X AND
C       THE EIGENVECTORS OF MS STORED IN C.  THESE VECTORS
C       SERVE BOTH AS EIGENVECTOR APPROXIMATIONS AND TO
C       FORM THE MATRIX USED TO START THE BLOCK LANCZOS
C       METHOD IN THE NEXT ITERATION.
C
        CALL ROTATE(N,Q,M,PS,NCONV+P,C,X)
C
        M=M+NCONV
        IMM=IMM+P*S
        GO TO 300
C
C       THIS IS THE END OF THE MAIN BODY OF THE SUBROUTINE.  NOW
C       SET THE VALUE OF IECODE AND EXIT.
C
  900 IECODE=0
        RETURN
  901 IECODE=1
        RETURN
  902 IECODE=2
        RETURN
  903 IECODE=3
        RETURN
  904 IECODE=4
        RETURN
  905 IECODE=5
        RETURN
  906 IECODE=6
        RETURN
  907 IECODE=7
        PINIT=-P
        RETURN
C
        END
```

```
      SUBROUTINE BKLANC (N,Q,M,P,S,OP,D,C,X,E,U,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,P,S
      DIMENSION D(Q),C(Q,Q),X(N,Q)
      DIMENSION E(Q),U(N),V(N)
C
C     THIS SUBROUTINE IMPLEMENTS THE BLOCK LANCZOS
C     METHOD WITH REORTHOGONALIZATION.  BKLANC COMPUTES
C     A BLOCK TRIDIAGONAL MATRIX MS WHICH IT STORES IN
C     ROWS AND COLUMNS M+1 THROUGH M+P*S OF THE ARRAY C,
C     AND AN ORTHONORMAL MATRIX XS WHICH IT STORES IN
C     COLUMNS M+1 THROUGH M+P*S OF THE N-BY-Q ARRAY X.
C     MS IS A SYMMETRIC MATRIX WITH P-BY-P
C     SYMMETRIC MATRICES M(1), ... , M(S) ON ITS
C     DIAGONAL AND P-BY-P UPPER TRIANGULAR MATRICES
C     R(2), ... , R(S) ALONG ITS LOWER DIAGONAL.  SINCE
C     MS IS SYMMETRIC AND BANDED, ONLY ITS LOWER
C     TRIANGLE (P+1 DIAGONALS) IS STORED IN C.  XS IS
C     COMPOSED OF S N-BY-P ORTHONORMAL MATRICES X(1),
C     ... , X(S) WHERE X(1) IS GIVEN AND SHOULD BE
C     STORED IN COLUMNS M+1 THROUGH M+P OF X.
C     FURTHERMORE X(1) IS ASSUMED TO SATISFY
C     X(1)'*A*X(1) = DIAG(D(M+1),D(M+2), ... , D(M+P)),
C     AND IF M>0, THEN X(1) IS ASSUMED TO BE ORTHOGONAL
C     TO THE VECTORS STORED IN COLUMNS 1 THROUGH M OF X.
C     OP IS THE NAME OF AN EXTERNAL SUBROUTINE USED TO
C     DEFINE THE MATRIX A.  DURING THE FIRST STEP, THE
C     SUBROUTINE ERR IS CALLED AND THE QUANTITIES EJ ARE
C     COMPUTED WHERE EJ = || A*XlJ - D(M+J)*XlJ ||, XlJ
C     IS THE J-TH COLUMN OF X(1), AND || * || DENOTES
C     THE EUCLIDEAN NORM.  EJ IS STORED IN E(M+J), J=1,
C     2, ... , P.  U AND V ARE AUXILLIARY VECTORS USED
C     BY OP.
C
      MP1=M+1
      MPPS=M+P*S
C
      DO 90 L=1,S
C
      LL=M+(L-1)*P+1
      LU=M+L*P
C
      DO 70 K=LL,LU
C
      DO 10 I=1,N
   10 U(I)=X(I,K)
      CALL OP(N,U,V)
C
      IF (L.GT.1) GO TO 19
      DO 12 I=K,LU
   12 C(I,K)=0.0D0
      C(K,K)=D(K)
      DO 14 I=1,N
   14 V(I)=V(I)-D(K)*X(I,K)
      GO TO 61
C
   19 DO 30 I=K,LU
      T=0
      DO 20 J=1,N
   20 T=T+V(J)*X(J,I)
   30 C(I,K)=T
C
      IT=K-P
      DO 60 I=1,N
      T=0
      DO 40 J=IT,K
   40 T=T+X(I,J)*C(K,J)
      IF (K.EQ.LU) GO TO 60
      KP1=K+1
      DO 50 J=KP1,LU
```

```
   50 T=T+X(I,J)*C(J,K)
   60 V(I)=V(I)-T
C
   61 IF (L.EQ.S) GO TO 70
      DO 63 I=1,N
   63 X(I,K+P)=V(I)
C
   70 CONTINUE
C
      IF (L.EQ.1) CALL ERR(N,Q,M,P,X,E)
      IF (L.EQ.S) GO TO 90
C
      CALL ORTHG (N,Q,LU,P,C,X)
C
      IL=LU+1
      IT=LU
      DO 80 J=1,P
      IT=IT+1
      DO 80 I=IL,IT
   80 C(I,IT-P)=C(I,IT)
C
   90 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PCH(N,Q,M,R,NCONV,P,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,R,NCONV,P,S
C
C     BASED ON THE VALUES OF N, Q, M, R, AND NCONV, PCH
C     CHOOSES NEW VALUES FOR P AND S, THE BLOCK SIZE AND
C     NUMBER OF STEPS FOR THE BLOCK LANCZOS METHOD.  THE
C     STRATEGY USED HERE IS TO CHOOSE P TO BE THE
C     SMALLER OF THE TWO FOLLOWING VALUES:  1) THE
C     PREVIOUS BLOCK SIZE; AND, 2) THE NUMBER OF VALUES
C     LEFT TO BE COMPUTED.  S IS CHOSEN AS LARGE AS
C     POSSIBLE SUBJECT TO THE CONSTRAINTS IMPOSED BY THE
C     LIMITS OF STORAGE.  IN ANY EVENT, S IS GREATER
C     THAN OR EQUAL TO 2.  N IS THE ORDER OF THE PROBLEM
C     AND Q IS THE NUMBER OF VECTORS AVAILABLE FOR
C     STORING EIGENVECTORS AND APPLYING THE BLOCK
C     LANCZOS METHOD.  M IS THE NUMBER OF EIGENVALUES
C     AND EIGENVECTORS THAT HAVE ALREADY BEEN COMPUTED
C     AND R IS THE REQUIRED NUMBER.  FINALLY, NCONV IS
C     THE NUMBER OF EIGENVALUES AND EIGENVECTORS THAT
C     HAVE CONVERGED IN THE CURRENT ITERATION.
C
      INTEGER PT,ST
C
      MT=M+NCONV
      PT=R-MT
      IF (PT.GT.P) PT=P
      IF (PT.GT.0) GO TO 101
      P=0
      RETURN
C
  101 CONTINUE
      ST=(Q-MT)/PT
      IF (ST.GT.2) GO TO 110
      ST=2
      PT=(Q-MT)/2
C
  110 P=PT
      S=ST
C
      RETURN
      END
```

```
      SUBROUTINE ERR(N,Q,M,P,X,E)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,P
      DIMENSION X(N,Q),E(Q)
C
C     ERR COMPUTES THE EUCLIDEAN LENGTHS OF THE VECTORS
C     STORED IN COLUMNS M+P+1 THROUGH M+P+P OF THE
C     N-BY-Q ARRAY X AND STORES THEM IN ELEMENTS M+1
C     THROUGH M+P OF E.
C
      MP1=M+P+1
      MPP=M+P+P
      DO 200 K=MP1,MPP
      T=0.0D0
      DO 100 I=1,N
  100 T=T+X(I,K)**2
  200 E(K-P)=DSQRT(T)
      RETURN
      END
```

```
      SUBROUTINE CNVTST(N,Q,M,P,ERRC,EPS,D,E,NCONV)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER Q,M,P
      REAL*8 EPS
      DIMENSION D(Q),E(Q)
C
C     CNVTST DETERMINES WHICH OF THE P EIGENVALUES
C     STORED IN ELEMENTS M+1 THROUGH M+P OF D HAVE
C     CONVERGED USING THE TEST DESCRIBED IN SECTION 3.2.
C     ERRC IS A MEASURE OF THE ACCUMULATED ERROR IN THE
C     M PREVIOUSLY COMPUTED EIGENVALUES AND
C     EIGENVECTORS.  ERRC IS UPDATED IF MORE
C     APPROXIMATIONS HAVE CONVERGED.  THE NORMS OF THE
C     RESIDUAL VECTORS ARE STORED IN ELEMENTS M+1
C     THROUGH M+P OF E.  EPS IS THE PRECISION TO WHICH
C     WE ARE COMPUTING THE APPROXIMATIONS.  FINALLY,
C     NCONV IS THE NUMBER THAT HAVE CONVERGED.  IF
C     NCONV=0, THEN NONE HAVE CONVERGED.
C
      REAL*8 MCHEPS / 2.22D-16/
C
      K=0
      DO 100 I=1,P
      T=DABS(D(M+I))
      IF (T.LT.1.0D0) T=1.0D0
      IF (E(M+I).GT.T*(EPS + 10D0*N*MCHEPS)+ERRC) GO TO 200
  100 K=I
  200 NCONV=K
      IF (K.EQ.0) RETURN
C
      T=0.0D0
      DO 300 I=1,K
      T=T+E(M+I)**2
  300 CONTINUE
      ERRC=DSQRT(ERRC**2+T)
      RETURN
      END
```

```
      SUBROUTINE EIGEN(Q,M,P,PS,C,D)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER M,P,Q,PS
      DIMENSION C(Q,Q),D(Q)
C
C     EIGEN SOLVES THE EIGENPROBLEM FOR THE SYMMETRIC
C     MATRIX MS OF ORDER PS STORED IN ROWS AND COLUMNS
C     M+1 THROUGH M+PS OF C.  THE EIGENVALUES OF MS ARE
C     STORED IN ELEMENTS M+1 THROUGH M+PS OF D AND THE
C     EIGENVECTORS ARE STORED IN ROWS AND COLUMNS 1
C     THROUGH PS OF C POSSIBLY OVERWRITTING MS.  EIGEN
C     SIMPLY RE-STORES MS IN A MANNER ACCEPTABLE TO THE
C     SUBROUTINES TRED2 AND TQL2.
C
      DIMENSION DD(25),V(25)
C
      DO 150 I=1,PS
      LIM=I-P
      IF (I.LE.P) LIM=1
      IF (LIM.LE.1) GO TO 130
C
      LM1=LIM-1
      DO 120 J=1,LM1
  120 C(I,J)=0.0D0
C
  130 DO  140 J=LIM,I
  140 C(I,J)=C(I+M,J+M)
C
  150 CONTINUE
C
      CALL TRED2(Q,PS,C,DD,V,C)
      CALL TQL2(Q,PS,DD,V,C,IERR)
C
      DO 160 I=1,PS
  160 D(M+I)=DD(I)
C
      RETURN
      END
```

```
      SUBROUTINE SECTN(N,Q,M,P,OP,X,C,D,U,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,P
      EXTERNAL OP
      DIMENSION X(N,Q),C(Q,Q),D(Q),U(N),V(N)
C
C     SECTN TRANSFORMS THE N-BY-P ORTHONORMAL MATRIX X1,
C     SAY, STORED IN COLUMNS M+1 THROUGH M+P OF THE
C     N-BY-Q ARRAY X SO THAT X1 *A*X1 = DIAG(D1,D2, ...
C      , DP), WHERE   DENOTES TRANSPOSE AND A IS A
C     SYMMETRIC MATRIX OF ORDER N DEFINED BY THE
C     SUBROUTINE OP.  THE VALUES D1, D2, ..., DP ARE
C     STORED IN ELEMENTS M+1 THROUGH M+P OF D.  SECTN
C     FORMS THE MATRIX X1 *A*X1 = CP, STORING CP IN THE
C     ARRAY C.  THE VALUES D1, D2, ... , DP AND THE
C     EIGENVECTORS QP OF CP ARE COMPUTED BY EIGEN AND
C     STORED IN D AND C, RESP.  ROTATE THEN CARRIES OUT
C     THE TRANSFORMATION X1<=X1*QP.
C
      ICOL1=M
      DO 300 J=1,P
      ICOL1=ICOL1+1
      DO 100 I=1,N
  100 U(I)=X(I,ICOL1)
      CALL OP(N,U,V)
      ICOL2=M
      DO 300 I=1,J
      ICOL2=ICOL2+1
      T=0.0D0
      DO 200 K=1,N
  200 T=T+V(K)*X(K,ICOL2)
  300 C(ICOL1,ICOL2)=T
C
      CALL EIGEN(Q,M,P,P,C,D)
C
      CALL ROTATE(N,Q,M,P,P,C,X)
C
      RETURN
      END
```

```
      SUBROUTINE ROTATE (N,Q,M,PS,L,C,X)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,PS,L
      DIMENSION C(Q,Q),X(N,Q)
C
C     ROTATE COMPUTES THE FIRST L COLUMNS OF THE MATRIX
C     XS*QS WHERE XS IS AN N-BY-PS ORTHONORMAL MATRIX
C     STORED IN COLUMNS M+1 THROUGH M+PS OF THE N-BY-Q
C     ARRAY X AND QS IS A PS-BY-PS ORTHONORMAL MATRIX
C     STORED IN ROWS AND COLUMNS 1 THROUGH PS OF THE
C     ARRAY C.  THE RESULT IS STORED IN COLUMNS M+1
C     THROUGH M+L OF X OVERWRITTING PART OF XS.
C
      DIMENSION V(25)
C
      DO 300 I=1,N
      DO 200 K=1,L
      T=0
      DO 100 J=1,PS
  100 T=T+X(I,M+J)*C(J,K)
  200 V(K)=T
      DO 300 K=1,L
  300 X(I,M+K)=V(K)
C
      RETURN
      END
```

```
      SUBROUTINE ORTHG(N,Q,F,P,B,X)
      IMPLICIT REAL *8 (A-H,O-Z)
      INTEGER N,Q,F,P
      DIMENSION B(Q,Q),X(N,Q)
C
C     ORTHG REORTHOGONALIZES THE N-BY-P MATRIX Z STORED
C     IN COLUMNS F+1 THROUGH F+P OF THE N-BY-Q ARRAY X
C     WITH RESPECT TO THE VECTORS STORED IN THE FIRST F
C     COLUMNS OF X AND THEN DECOMPOSES THE RESULTING
C     MATRIX INTO THE PRODUCT OF AN N-BY-P ORTHONORMAL
C     MATRIX XORTH, SAY, AND A P-BY-P UPPER TRIANGULAR
C     MATRIX R.  XORTH IS STORED OVER Z AND THE UPPER
C     TRIANGLE OF R IS STORED IN ROWS AND COLUMNS F+1
C     THROUGH F+P OF THE Q-BY-Q ARRAY B.  A STABLE
C     VARIANT OF THE GRAM-SCHMIDT ORTHOGONALIZATION
C     METHOD IS UTILIZED.  THIS SUBROUTINE IS BASED
C     DIRECTLY ON THE ALGOL 60 PROCEDURE ORTHOG
C     CONTAINED IN THE SIMULTANEOUS ITERATION PROGRAM OF
C     RUTISHAUSER.
C
      INTEGER FP1,FPP
      LOGICAL ORIG
C
      IF (P.EQ.0) RETURN
      FP1=F+1
      FPP=F+P
C
      DO 50 K=FP1,FPP
      ORIG=.TRUE.
      KM1=K-1
C
   10 T=0.0D0
      IF (KM1.LT.1) GO TO 25
      DO 20 I=1,KM1
      S=0.0D0
      DO 15 J=1,N
   15 S=S+X(J,I)*X(J,K)
      IF (ORIG.AND.I.GT.F) B(I,K)=S
      T=T+S*S
      DO 20 J=1,N
   20 X(J,K)=X(J,K)-S*X(J,I)
C
   25 S=0.0D0
      DO 30 J=1,N
   30 S=S+X(J,K)*X(J,K)
      T=T+S
      IF (S.GT.T/100) GO TO 40
      ORIG=.FALSE.
      GO TO 10
   40 S=DSQRT(S)
      B(K,K)=S
      IF (S.NE.0) S=1/S
      DO 50 J=1,N
   50 X(J,K)=S*X(J,K)
C
      RETURN
      END
```

```
      SUBROUTINE RANDOM(N,Q,L,X)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,L
      DIMENSION X(N,Q)
C
C     RANDOM COMPUTES AND STORES A SEQUENCE OF N PSEUDO-RANDOM
C     NUMBERS IN THE L-TH COLUMN OF THE N-BY-Q ARRAY X.  RANDOM
C     GENERATES TWO SEQUENCES OF PSEUDO-RANDOM NUMBERS, FILLING
C     AN ARRAY WITH ONE SEQUENCE AND USING THE SECOND TO ACCESS
C     THE ARRAY IN A RANDOM FASHION.
C
      DIMENSION T(100)
      INTEGER F1/71416/,F2/27183/,FT
      INTEGER A/6821/,C/5327/,X0/5328/,X1
C
      DO 100 I=1,100
      X1=A*X0+C
      IF (X1.GE.10000) X1=X1-10000
      T(I)=X1/9999D0-.5D0
  100 X0=X1
C
      DO 200 I=1,N
C
      FT=F1+F2
      IF (FT.GE.1000000) FT=FT-1000000
      F1=F2
      F2=FT
      K=FT/1D6*100+1
      X(I,L)=T(K)
C
      X1=A*X0+C
      IF (X1.GE.10000) X1=X1-10000
      T(K)=X1/9999D0-.5D0
  200 X0=X1
C
      RETURN
      END
```

```
C
C     ------------------------------------------------------------------
C
      SUBROUTINE TRED2(NM,N,A,D,E,Z)
C
      INTEGER I,J,K,L,N,II,NM,JP1
      REAL*8 A(NM,N),D(N),E(N),Z(NM,N)
      REAL*8 F,G,H,HH,SCALE
      REAL*8 DSQRT,DABS,DSIGN
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE TRED2,
C     NUM. MATH. 11, 181-195(1968) BY MARTIN, REINSCH, AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 212-226(1971).
C
C     THIS SUBROUTINE REDUCES A REAL SYMMETRIC MATRIX TO A
C     SYMMETRIC TRIDIAGONAL MATRIX USING AND ACCUMULATING
C     ORTHOGONAL SIMILARITY TRANSFORMATIONS.
C
C     ON INPUT:
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT;
C
C        N IS THE ORDER OF THE MATRIX;
C
C        A CONTAINS THE REAL SYMMETRIC INPUT MATRIX.  ONLY THE
C          LOWER TRIANGLE OF THE MATRIX NEED BE SUPPLIED.
C
C     ON OUTPUT:
C
C        D CONTAINS THE DIAGONAL ELEMENTS OF THE TRIDIAGONAL MATRIX;
C
C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE TRIDIAGONAL
C          MATRIX IN ITS LAST N-1 POSITIONS.  E(1) IS SET TO ZERO;
C
C        Z CONTAINS THE ORTHOGONAL TRANSFORMATION MATRIX
C          PRODUCED IN THE REDUCTION;
C
C        A AND Z MAY COINCIDE.  IF DISTINCT, A IS UNALTERED.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY
C     ------------------------------------------------------------------
C
      DO 100 I = 1, N
C
         DO 100 J = 1, I
            Z(I,J) = A(I,J)
  100 CONTINUE
C
      IF (N .EQ. 1) GO TO 320
C     :::::::::: FOR I=N STEP -1 UNTIL 2 DO -- ::::::::::
      DO 300 II = 2, N
         I = N + 2 - II
         L = I - 1
         H = 0.0D0
         SCALE = 0.0D0
         IF (L .LT. 2) GO TO 130
```

```
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
      DO 120 K = 1, L
  120 SCALE = SCALE + DABS(Z(I,K))
C
      IF (SCALE .NE. 0.0D0) GO TO 140
  130 E(I) = Z(I,L)
      GO TO 290
C
  140 DO 150 K = 1, L
         Z(I,K) = Z(I,K) / SCALE
         H = H + Z(I,K) * Z(I,K)
  150 CONTINUE
C
      F = Z(I,L)
      G = -DSIGN(DSQRT(H),F)
      E(I) = SCALE * G
      H = H - F * G
      Z(I,L) = F - G
      F = 0.0D0
C
      DO 240 J = 1, L
         Z(J,I) = Z(I,J) / (SCALE * H)
         G = 0.0D0
C     ********** FORM ELEMENT OF A*U **********
         DO 180 K = 1, J
  180    G = G + Z(J,K) * Z(I,K)
C
         JP1 = J + 1
         IF (L .LT. JP1) GO TO 220
C
         DO 200 K = JP1, L
  200    G = G + Z(K,J) * Z(I,K)
C     ********** FORM ELEMENT OF P **********
  220    E(J) = G / H
         F = F + E(J) * Z(I,J)
  240 CONTINUE
C
      HH = F / (H + H)
C     ********** FORM REDUCED A **********
      DO 260 J = 1, L
         F = Z(I,J)
         G = E(J) - HH * F
         E(J) = G
C
         DO 260 K = 1, J
            Z(J,K) = Z(J,K) - F * E(K) - G * Z(I,K)
  260 CONTINUE
C
      DO 280 K = 1, L
  280 Z(I,K) = SCALE * Z(I,K)
C
  290 D(I) = H
  300 CONTINUE
C
  320 D(1) = 0.0D0
      E(1) = 0.0D0
```

```
C     ********** ACCUMULATION OF TRANSFORMATION MATRICES **********
      DO 500 I = 1, N
         L = I - 1
         IF (D(I) .EQ. 0.0D0) GO TO 380
C
         DO 360 J = 1, L
            G = 0.0D0
C
            DO 340 K = 1, L
  340       G = G + Z(I,K) * Z(K,J)
C
            DO 360 K = 1, L
               Z(K,J) = Z(K,J) - G * Z(K,I)
  360    CONTINUE
C
  380    D(I) = Z(I,I)
         Z(I,I) = 1.0D0
         IF (L .LT. 1) GO TO 500
C
         DO 400 J = 1, L
            Z(I,J) = 0.0D0
            Z(J,I) = 0.0D0
  400    CONTINUE
C
  500 CONTINUE
C
      RETURN
C     ********** LAST CARD OF TRED2 **********
      END
```

```
C     ------------------------------------------------------------------
C
      SUBROUTINE TQL2(NM,N,D,E,Z,IERR)
C
      INTEGER I,J,K,L,M,N,II,L1,NM,MML,IERR
      REAL*8 D(N),E(N),Z(NM,N)
      REAL*8 B,C,F,G,H,P,R,S,MACHEP
      REAL*8 DSQRT,DABS,DSIGN
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE TQL2,
C     NUM. MATH. 11, 293-306(1968) BY BOWDLER, MARTIN, REINSCH, AND
C     WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 227-240(1971).
C
C     THIS SUBROUTINE FINDS THE EIGENVALUES AND EIGENVECTORS
C     OF A SYMMETRIC TRIDIAGONAL MATRIX BY THE QL METHOD.
C     THE EIGENVECTORS OF A FULL SYMMETRIC MATRIX CAN ALSO
C     BE FOUND IF  TRED2  HAS BEEN USED TO REDUCE THIS
C     FULL MATRIX TO TRIDIAGONAL FORM.
C
C     ON INPUT:
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT;
C
C        N IS THE ORDER OF THE MATRIX;
C
C        D CONTAINS THE DIAGONAL ELEMENTS OF THE INPUT MATRIX;
C
C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE INPUT MATRIX
C          IN ITS LAST N-1 POSITIONS.  E(1) IS ARBITRARY;
C
C        Z CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE
C          REDUCTION BY  TRED2, IF PERFORMED.  IF THE EIGENVECTORS
C          OF THE TRIDIAGONAL MATRIX ARE DESIRED, Z MUST CONTAIN
C          THE IDENTITY MATRIX.
C
C     ON OUTPUT:
C
C        D CONTAINS THE EIGENVALUES IN ASCENDING ORDER.  IF AN
C          ERROR EXIT IS MADE, THE EIGENVALUES ARE CORRECT BUT
C          UNORDERED FOR INDICES 1,2,...,IERR-1;
C
C        E HAS BEEN DESTROYED;
C
C        Z CONTAINS ORTHONORMAL EIGENVECTORS OF THE SYMMETRIC
C          TRIDIAGONAL (OR FULL) MATRIX.  IF AN ERROR EXIT IS MADE,
C          Z CONTAINS THE EIGENVECTORS ASSOCIATED WITH THE STORED
C          EIGENVALUES;
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE J-TH EIGENVALUE HAS NOT BEEN
C                     DETERMINED AFTER 30 ITERATIONS.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY
C     ------------------------------------------------------------------
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C                MACHEP = 16.0D0**(-13) FOR LONG FORM ARITHMETIC
C                ON S360 **********
      DATA MACHEP/Z3410000000000000/
```

```
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E(I-1) = E(I)
C
      F = 0.0D0
      B = 0.0D0
      E(N) = 0.0D0
C
      DO 240 L = 1, N
         J = 0
         H = MACHEP * (DABS(D(L)) + DABS(E(L)))
         IF (B .LT. H) B = H
C     ::::::::::: LOOK FOR SMALL SUB-DIAGONAL ELEMENT :::::::::::
         DO 110 M = L, N
            IF (DABS(E(M)) .LE. B) GO TO 120
C     ::::::::::: E(N) IS ALWAYS ZERO, SO THERE IS NO EXIT
C                 THROUGH THE BOTTOM OF THE LOOP :::::::::::
  110    CONTINUE
C
  120    IF (M .EQ. L) GO TO 220
  130    IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ::::::::::: FORM SHIFT :::::::::::
         L1 = L + 1
         G = D(L)
         P = (D(L1) - G) / (2.0D0 * E(L))
         R = DSQRT(P*P+1.0D0)
         D(L) = E(L) / (P + DSIGN(R,P))
         H = G - D(L)
C
         DO 140 I = L1, N
  140    D(I) = D(I) - H
C
         F = F + H
C     ::::::::::: QL TRANSFORMATION :::::::::::
         P = D(M)
         C = 1.0D0
         S = 0.0D0
         MML = M - L
C     ::::::::::: FOR I=M-1 STEP -1 UNTIL L DO -- :::::::::::
         DO 200 II = 1, MML
            I = M - II
            G = C * E(I)
            H = C * P
            IF (DABS(P) .LT. DABS(E(I))) GO TO 150
            C = E(I) / P
            R = DSQRT(C*C+1.0D0)
            E(I+1) = S * P * R
            S = C / R
            C = 1.0D0 / R
            GO TO 160
  150       C = P / E(I)
            R = DSQRT(C*C+1.0D0)
            E(I+1) = S * E(I) * R
            S = 1.0D0 / R
            C = C * S
  160       P = C * D(I) - S * G
            D(I+1) = H + S * (C * G + S * D(I))
C     ::::::::::: FORM VECTOR :::::::::::
            DO 180 K = 1, N
               H = Z(K,I+1)
               Z(K,I+1) = S * Z(K,I) + C * H
               Z(K,I) = C * Z(K,I) - S * H
  180       CONTINUE
C
  200    CONTINUE
C
```

```
         E(L) = S * P
         D(L) = C * P
         IF (DABS(E(L)) .GT. B) GO TO 130
  220    D(L) = D(L) + F
  240 CONTINUE
C     ********** ORDER EIGENVALUES AND EIGENVECTORS **********
      DO 300 II = 2, N
         I = II - 1
         K = I
         P = D(I)
C
         DO 260 J = II, N
            IF (D(J) .GE. P) GO TO 260
            K = J
            P = D(J)
  260    CONTINUE
C
         IF (K .EQ. I) GO TO 300
         D(K) = D(I)
         D(I) = P
C
         DO 280 J = 1, N
            P = Z(J,I)
            Z(J,I) = Z(J,K)
            Z(J,K) = P
  280    CONTINUE
C
  300 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
C     ********** LAST CARD OF TQL2 **********
      END
```

```
      SUBROUTINE PRINTX (N,Q,M,K,X)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N,Q,M,K
      DIMENSION X(N,Q)
C
C     PRINT OUT COLUMNS M+1 THROUGH M+K OF THE
C     N-BY-Q ARRAY X.
C
      DO 100 I=1,N
  100 PRINT 1001,(X(I,M+J),J=1,K)
 1001 FORMAT(' =>X/ ',8D12.4)
      RETURN
      END



      SUBROUTINE AX(N,U,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER N
      DIMENSION U(N),V(N)
C
C     AX COMPUTES Y = A*X WHERE A = DIAG(-1,-1/2,-1/3, ... ,-1/N).
C     X IS STORED IN U AND AX STORES Y IN V.
C
      DO 100 I=1,N
  100 V(I)=-1D0/I*U(I)
      RETURN
      END



      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION D(25),X(20000)
      EXTERNAL AX
      INTEGER Q,PINIT,R
C
C     SAMPLE MAIN PROGRAM.  MINVAL IS USED TO COMPUTE
C     THE 4 LEAST EIGENVALUES OF THE MATRIX
C     A = DIAG(-1,-1/2, ... , -1/300) TO AN APPROXIMATE
C     PRECISION OF 10**(-3).  TWELVE VECTORS ARE ALLOWED
C     FOR THE BLOCK LANCZOS METHOD AND AN INITIAL BLOCK
C     SIZE OF 4 IS CHOSEN.
C
      N=300
      Q=12
      PINIT=4
      R=4
      MMAX=5000
      EPS=1D-03
      M=0
C
      CALL MINVAL(N,Q,PINIT,R,MMAX,EPS,AX,M,D,X,IECODE)
C
      PRINT 1001,M,IECODE,(D(I),I=1,M)
 1001 FORMAT(/' =>M,IECODE= ',I4,1X,I4/(' =>E ',5D23.15))
C
      PRINT 1002
 1002 FORMAT(//' EIGENVECTORS ... '//)
      CALL PRINTX(N,Q,0,M,X)

      STOP
      END
```

BIBLIOGRAPHY

1. Bauer, L. and E. L. Reiss, "Block Five Diagonal Matrices and the Fast Numerical Solution of the Biharmonic Equation," pp. 311-326, Math. of Comp., Vol. 26, no. 118, April 1972.

2. Buzbee, B. L. and F. W. Dorr, "The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions," SIAM Journal on Numerical Analysis, to appear.

3. Crawford, C. R., "The Numerical Solution of the Generalized Eigenvalue Problem," Dissertation, University of Michigan, Ann Arbor, Michigan, 1970. Also, pp. 41-44, CACM, Vol. 16, no. 1, January 1973.

4. Cullum, J. 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, May 1974.

5. Davis, C. and W. Kahan, "The Rotation of Eigenvectors by a Perturbation-III," SIAM J. of N.A., Vol. 7, no. 1, March 1970.

6. Godunov, S. K. and G. P. Prokopov, "A method of minimal iterations for evaluating the eigenvalues of an elliptic operator," U.S.S.R. Comp. Math. Math., Vol. 10, no. 5, 1970, pp. 141-153. (Translated by D. E. Brown)

7. Golub, G. H., R. Underwood, and J. H. Wilkinson, "The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ problem," Technical Report #142, Computer Science Department, Stanford University, 1970.

8. Golub, G. H., "Some uses of the Lanczos algorithm in numerical linear algebra," Report #STAN-CS-72-302, Computer Science Department, Stanford University, 1972. Also, pp. 173-184, Topics in Numerical Analysis, Academic Press, London and New York, 1973.

9. Kahan, W., "Inclusion Theorems for Clusters of Eigenvalues of Hermitian Matrices," University of Toronto, February 1967.

10. Kahan, W. and B. Parlett, "An Analysis of Lanczos Algorithms for Symmetric Matrices," Memo #ERL-M467, Electronics Research Lab, College of Engineering, University of California, Berkeley, California, September 1974.

11. Kaniel, S., "Estimates for some computational techniques in linear algebra," Math. of Comp., Vol. 20, no. 95, July 1966, pp. 369-378.

12. Karush, W., "An iterative method for finding characteristic vectors of a symmetric matrix," Pacific J. Math., Vol. 1, no. 2, 1951, pp. 233-248.

13. Lanczos, C., "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," Journal of Research of the National Bureau of Standards, Vol. 45, 1950, pp. 255-282.

14. Ortega, J. M., Numerical Analysis--A Second Course, Academic Press, New York and London, 1972.

15. Paige, C. C., "Error analysis of the symmetric Lanczos process for the eigenproblem," Technical Note #207, Institute of Computer Science, The University of London, 1969.

16. Paige, C. C., "Eigenvalues of Perturbed Hermitian Matrices," pp. 1-10, Lin. Algebra and Appl., Vol. 8, 1974.

17. Paige, C. C., "The computation of eigenvalues and eigenvectors of very large sparse matrices," Ph.D. dissertation, The University of London, 1971.

18. Paige, C. C., "Computational variants of the Lanczos method for the eigenproblem," J. Inst. Maths. Applics., Vol. 10, pp. 373-381, 1972.

19. Rutishauser, H., "Computational aspects of F. L. Bauer's simultaneous iteration method," Numer. Math., Vol. 13, pp. 4-13, 1969.

20. Rutishauser, H., "Simultaneous iteration method for symmetric matrices," Numer. Math., Vol. 16, pp. 205-223, 1970.

21. Smith, B. T., J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, Matrix Eigensystem Routines--EISPACK Guide, Springer-Verlag, Berlin, Heidelberg, New York, 1974.

22. Stewart, G. W., "Error and perturbation bounds for subspaces associated with certain eigenvalue problems," SIAM Review, Vol. 18, no. 4, pp. 727-764, October 1973.

23. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

24. Wilkinson, J. H. and C. Reinsch, Handbook for Automatic Computation, Vol. II, Linear Algebra, Part 2, Springer-Verlag, New York, Heidelberg, Berlin, 1971.