REPORT NO.
UCB/SESM-84/04

STRUCTURAL ENGINEERING AND
STRUCTURAL MECHANICS

THE LANCZOS ALGORITHM FOR SOLUTION OF LARGE GENERALIZED EIGENPROBLEMS

by

BAHRAM NOUR-OMID

MAY 1984

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

The Lanczos Algorithm for Solution of Large Generalized Eigenproblems

by

B. Nour-Omid

This work was supported in part by the Office of Naval Research under contract N00014-76-C-0013.

TABLE OF CONTENTS

1. Introduction

The Lanczos algorithm was first proposed in 1950. Lanczos intended his algorithm to be used with a symmetric matrix for computing a few of the extreme eigenvalues and their corresponding eigenvectors. However, the algorithm was then adopted as a method for reducing a symmetric matrix to tridiagonal form. Lanczos himself observed that roundoff error has a significant effect on the algorithm and expensive modifications became necessary to overcome this defect. By 1955 the Householder method replaced the Lanczos algorithm as a more efficient method for tridiagonalizing a matrix.

In most engineering applications only a few eigenvalues at one end of the spectrum is required. For small problems the Householder-QR can find all the eigenvalues almost as fast as a few. For large problems, the subspace iteration method could take advantage of the fact that only a few eigenvalues are desired and therefore was more efficient both in storage and cost than the Householder-QR method. Recently, the Lanczos algorithm has emerged as a viable alternative to both of these methods. The reasons for the slow acceptance of the method were two fold:

1. The early users of the Lanczos algorithm avoided using matrix factorization as part of their solution procedure. When we are interested in the eigenvalues at the left end of the spectrum of a positive definite system this procedure results in long and therefore expensive Lanczos runs.

2. Computer roundoff has a strong influence on the algorithm in its basic form. This effect appears in the form of severe loss of orthogonality among the Lanczos vectors and so casts doubt on the ability of the Lanczos approximations to converge to the desired values.

These defects have now been overcome and the algorithm can be used effectively to compute the desired eigenpairs.

In this chapter we are concerned with the application of the Lanczos algorithm to the generalized symmetric eigenproblem

$$[\mathbf{K} - \lambda \mathbf{M}]\mathbf{z} = 0 \qquad (1)$$

where **K** and **M** are $n \times n$ real symmetric matrices. We assume for the moment that the eigenvalues of (1) are real. In a later section we state the conditions on **K** and **M** that must hold to ensure real eigenvalues. The matrices in equation (1) are obtained by some form of discretization, such as finite element, finite difference, lumping, etc., of the continuum problem under consideration. For example in vibration analysis **K** and **M** are the stiffness and mass matrices respectively; then the eigenvalues of (1) are the squares of the frequencies and the eigenvectors are the modes of free vibration.

In the following sections we will derive the generalized Lanczos algorithm for the solution of the eigenproblem (1). The interrelation between the Lanczos method and vector iteration methods is established. We then look at the effect of round off on the algorithm and the resulting loss of orthogonality. We consider two possible modifications to the algorithm that can maintain a desired level of orthogonality among the Lanczos vectors. Finally, we give a detailed description of the algorithm we prefer together with a listing of the computer subprograms.

2. Spectral Transformation

We turn now to an important misconception concerning the Lanczos algorithm. It is usually presented as a way of computing eigenpairs, $(\lambda, \mathbf{z})$, of the standard eigenproblem

$$[\mathbf{A} - \lambda \mathbf{I}]\mathbf{z} = \mathbf{0} \tag{2}$$

The Lanczos method is so powerful that one can work directly with **A** to evaluate eigenvalues at both ends of the spectrum of (2) without solving any system of equations. Only products of **A** with a sequence of vectors need be computed. This virtue blinded certain users of the Lanczos method to the great advantages to be gained by a shift and invert procedure. Of course there are cases when factoring of **A** is not possible (e.g. when the matrix is only known implicitly) or not desirable (e.g. when **A** has a given sparsity structure that can be destroyed when factored).

However, we are more interested in the generalized eigenproblem (1). For this problem some form of inversion or factoring of a matrix, either explicitly or implicitly, is required. It is interesting to note that when the structure of **M** is the same as that of **K**, as in the case of a consistent mass matrix, then the cost of the power iteration method is exactly the same as that for inverse iteration method.

There are two different procedures for transforming the generalized eigenproblem to the standard form:

(i) Factor $\mathbf{M}$ into $\mathbf{M} = \mathbf{C}\mathbf{C}^T$. This is the Choleski factorization of $\mathbf{M}$. Then

$$(\mathbf{K} - \lambda\mathbf{M}) = \mathbf{C}(\mathbf{C}^{-1}\mathbf{K}\mathbf{C}^{-T} - \lambda\mathbf{I})\mathbf{C}^T \tag{3}$$

and the eigenproblem of (1) reduces to the standard form $(\mathbf{A}_1 - \lambda\mathbf{I})\hat{\mathbf{z}} = \mathbf{0}$ where $\mathbf{A}_1 = \mathbf{C}^{-1}\mathbf{K}\mathbf{C}^{-T}$ and $\hat{\mathbf{z}} = \mathbf{C}^T\mathbf{z}$. The eigenvalues of the reduced problem are the same as those of the generalized problem. Note that if $\mathbf{M}$ is positive semi-definite then its Choleski factors are singular and this transformation can not be performed.

(ii) Using a similar procedure, equation (1) can be transformed into

$$(\mathbf{A}_2 - \mu\mathbf{I})\hat{\mathbf{z}} = \mathbf{0} \tag{4}$$

where $\mathbf{A}_2 = \mathbf{C}^T\mathbf{K}^{-1}\mathbf{C}$ and $\mu = \frac{1}{\lambda}$. Note in this case both the eigenvalues and the eigenvectors of $\mathbf{A}_2$ are different form those of (1).

A more general form of the second transformation is to first perform a linear transformation or shift of the origin, $[\mathbf{K}_\sigma - (\lambda-\sigma)\mathbf{M}]\mathbf{z} = 0$, where $\mathbf{K}_\sigma = \mathbf{K} - \sigma\mathbf{M}$, and then perform (ii). This result in a standard eigenproblem with $\mathbf{A}_\sigma = \mathbf{C}^T\mathbf{K}_\sigma^{-1}\mathbf{C}$. The spectrum of $\mathbf{A}_\sigma$ is related to the original spectrum through

$$\nu = \frac{1}{\lambda-\sigma} \tag{5}$$

where ν is the eigenvalue of $\mathbf{A}_\sigma$ [2].

The second reduction procedure requires two triangular factorizations; one for $\mathbf{M}$ and one for $\mathbf{K}$. It is possible to avoid the factorization of $\mathbf{M}$ by working with

$$[\mathbf{K}_\sigma^{-1}\mathbf{M} - \nu\mathbf{I}]\mathbf{z} = \mathbf{0} \tag{6}$$

Although the matrix of this transformation is not symmetric, it is self-adjoint with respect to the inertial inner product defined by

$$(\mathbf{u},\mathbf{v})_\mathbf{M} = \mathbf{v}^T\mathbf{M}\mathbf{u} \tag{7}$$

which is shown in the following steps;

$$\begin{aligned}(\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{u}\ ,\ \mathbf{v})_{\mathbf{M}} &= \mathbf{v}^T\mathbf{M}\,\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{u} \\ &= (\mathbf{u}\ ,\ \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{v})_{\mathbf{M}}\end{aligned}$$

This unsymmetric form is particularly advantageous since it has the same eigenvector as the original problem [7]. The algorithm that is derived later in this chapter employs the transformation of equation (6).

3. Conditions for Real Eigenvalues

Contrary to common belief, the fact that **K** and **M** are symmetric is not a sufficient condition to ensure real eigenvalues for (1). This can be illustrated using the following simple example.

Example:

Let $\mathbf{K} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ and $\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Then the eigenvalues of (1) with these matrices are $\frac{1}{2}(-1 \pm i\sqrt{3})$, where $i^2 = -1$.

The eigenvalues of (1) are all real if some linear combination of **K** and **M** is positive definite; that is, $\rho\mathbf{K} + \tau\mathbf{M}$ is positive definite for some choice of ρ and τ. This is both a necessary and sufficient condition. Fortunately, in many problems encountered in structural mechanics either **K** or **M** or both are positive definite. However, in certain finite element implementations, such as lumping of a consistent mass matrix, together with Lagrange parameter formulation of various constraints in the problem, can result in indefinite **M** and **K** and complex eigenvalues for the pair. Special attention must be paid to these problems.

4. The Rayleigh-Ritz Approximation

Consider a given set of vectors, $\mathbf{X}_m = [\mathbf{x}_1\ , \mathbf{x}_2\ \cdots\ , \mathbf{x}_m]$, with $m << n$. We refer to these as trial vectors. We proceed to obtain an approximation to some of the eigenvectors of (1) by taking a linear combination of the trial vectors $\mathbf{X}_m$. Let $\mathbf{y} = \mathbf{X}_m\,\mathbf{s} = \sum_{i=1}^{m}\mathbf{x}_i s_i$ be the desired approximation. The i-th component of $\mathbf{s}$ is the coefficient of $\mathbf{x}_i$ in this representation of $\mathbf{y}$. We denote the corresponding approximation to the eigenvalues of (1) by θ. The residual, $\mathbf{r}$, associated with the approximating pair $(\theta, \mathbf{y})$ is given by

$$\mathbf{r} = \mathbf{K}\mathbf{y} - \theta\mathbf{M}\mathbf{y}$$

The Rayleigh-Ritz method requires the residual vector, $\mathbf{r}$, be orthogonal to each of the trial vectors; that is

$$\mathbf{X}_m^T\mathbf{r} = \mathbf{X}_m^T\mathbf{K}\mathbf{y} - \theta\mathbf{X}_m^T\mathbf{M}\mathbf{y} = 0$$

Substituting the representation of $\mathbf{y}$ in terms of the trial vectors, in the above orthogonality condition for $\mathbf{r}$, we obtain the reduced eigenproblem

$$(\mathbf{K}_m - \theta\mathbf{M}_m)\mathbf{s} = 0$$

where $\mathbf{K}_m = \mathbf{X}_m^T\mathbf{K}\mathbf{X}_m$ and $\mathbf{M}_m = \mathbf{X}_m^T\mathbf{M}\mathbf{X}_m$. The eigenvector of the reduced eigenproblem, $\mathbf{s}^{(k)}$, will determine the approximation, $\mathbf{y}^{(k)}$ to the eigenvector of (1) and its eigenvalue will approximate those of (1). We refer to $\mathbf{y}^{(k)}$ as the Ritz vector and to its corresponding $\theta^{(k)}$ as the Ritz value, for each $k = 1, \cdots, m$.

5. Derivation of The Lanczos Algorithm

The Lanczos method can be thought of as a means of constructing an orthogonal set of vectors, known as Lanczos vectors, for use in the Rayleigh-Ritz procedure. The algorithm is closely related to the inverse iteration and power methods for calculating a single eigenpair. Given a pair of matrices $\mathbf{K}_\sigma = \mathbf{K} - \sigma\mathbf{M}$ and $\mathbf{M}$, and a starting vector $\mathbf{r}$ these basic methods generate a sequence of vectors, $[\mathbf{r}, \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{r}, (\mathbf{K}_\sigma^{-1}\mathbf{M})^2\mathbf{r}, \ldots, (\mathbf{K}_\sigma^{-1}\mathbf{M})^j\mathbf{r}]$, during j iterations. These vectors are referred to as the Krylov sequence; the sequence converges to the eigenvector corresponding to the eigenvalue, λ, of (1) closest to the shift σ.

The basic difference between the Lanczos method and the other two is that the information contained in each successive vector of the Krylov sequence is used to obtain the best approximation to the wanted eigenvectors, instead of using only the last vector in the sequence. To be more specific, the Lanczos algorithm is equivalent to obtaining the Rayleigh-Ritz approximation using the vectors in the Krylov sequence as the trial vectors but it is much more efficient. This involves supplementing the Krylov sequence with the Gram-Schmidt orthogonalization process at each step. The result is a set of **M**-orthonormal vectors (the Lanczos vectors) that is used in the Rayleigh-Ritz procedure to reduce the dimension of the eigenproblem. We show below that orthogonalization is required only with respect to two preceding vectors; a fact recognized by Lanczos. The Rayleigh-Ritz procedure with the **M**-orthonormal bases for the Krylov subspace leads to a standard eigenproblem with a tridiagonal matrix.

To derive the Lanczos algorithm it will be assumed for the moment that the first j Lanczos vectors, $(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j)$ have been found, and the construction of the $j+1$ vector will be described. The resulting vectors all satisfy the condition $\mathbf{q}_i^T\mathbf{M}\mathbf{q}_j = \delta_{ij}$ where δ_{ij} is the Kronecker delta; that is, the vectors are orthonormal with respect to the mass matrix. To calculate $\mathbf{q}_{j+1}$, we must orthogonalize $\mathbf{v}_j = (\mathbf{K}_\sigma^{-1}\mathbf{M})^j\mathbf{r})$ against the j Lanczos vectors computed so far. From the definition of $\mathbf{v}_j$ we obtain $\mathbf{v}_j = \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{v}_{j-1}$. Now, $\mathbf{v}_{j-1}$ is the vector that is **M**-orthonormalized against the first $j-1$ Lanczos vectors to obtain $\mathbf{q}_j$. Therefore

$$\mathbf{v}_{j-1} = \sum_{i=1}^{j} \upsilon_i \mathbf{q}_i$$

where υ_i is the component of $\mathbf{v}_{j-1}$ along $\mathbf{q}_j$. This result is used to eliminate $\mathbf{v}_{j-1}$ in the above recursive relation for $\mathbf{v}_j$. We then get

$$\begin{aligned}\mathbf{v}_j &= \sum_{i=1}^{j} \upsilon_i \mathbf{K}_\sigma \mathbf{M}\mathbf{q}_i \\ &= \upsilon_j \mathbf{K}_\sigma \mathbf{M}\mathbf{q}_j + \sum_{i=1}^{j-1} \upsilon_i \mathbf{K}_\sigma \mathbf{M}\mathbf{q}_i\end{aligned}$$

Observe that each vector, $\mathbf{K}_\sigma\mathbf{M}\mathbf{q}_i$, in the above summation can be written as a linear combination of the first $i+1$ Lanczos vectors. Therefore the sum can be written as a linear combination of the first j Lanczos vectors. Consequently

$$\mathbf{v}_j = \upsilon_j \mathbf{K}_\sigma \mathbf{M}\mathbf{q}_j + \sum_{i=1}^{j} \overline{\upsilon}_i \mathbf{q}_i$$

The **M**-orthogonalization of $\mathbf{v}_j$ against the preceding j Lanczos vectors will purge the component of $\mathbf{v}_j$ along each of $\mathbf{q}_i$ and therefore the final result will be uneffected by the sum in the last equation. Therefore the next Lanczos vector, $\mathbf{q}_{j+1}$, will be obtained by first computing a preliminary vector $\bar{\mathbf{r}}_j$ from the previous vector, $\mathbf{q}_j$.

$$\bar{\mathbf{r}}_j = \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j \tag{8}$$

and **M**-orthonormalizing it against all the previous Lanczos vectors. Now, in general it may be assumed that this preliminary vector contains components from each of the preceding vectors. Thus,

$$\bar{\mathbf{r}}_j = \mathbf{r}_j + \alpha_j\mathbf{q}_j + \beta_j\mathbf{q}_{j-1} + \gamma_j\mathbf{q}_{j-2} + \cdots \tag{9}$$

where $\mathbf{r}_j$ is the "pure" component of $\bar{\mathbf{r}}_j$ orthogonal to all previous Lanczos vectors, and α_j, β_j, γ_j, $\cdots$ are the amplitudes of the previous vectors contained in $\bar{\mathbf{r}}_j$. These amplitude coefficients are evaluated from the orthonormality of the Lanczos vectors. Thus, if both sides of Eq. (7) are multiplied by $\mathbf{q}_j^T\mathbf{M}$, the result is

$$\mathbf{q}_j^T\mathbf{M}\bar{\mathbf{r}}_j = \mathbf{q}_j^T\mathbf{M}\mathbf{r}_j + \alpha_j\mathbf{q}_j^T\mathbf{M}\mathbf{q}_j + \beta_j\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-1} + \gamma_j\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-2} + \cdots \tag{10}$$

Here the first term on the right hand side vanishes by <u>definition</u>, and all terms beyond the second vanish similarly due to **M**-orthogonality. The normalizing definition applied to the second term then reduces Eq. (10) to an expression for the amplitude of $\mathbf{q}_j$ along $\bar{\mathbf{r}}_j$:

$$\alpha_j = \mathbf{q}_j^T\mathbf{M}\bar{\mathbf{r}}_j \tag{11}$$

The amplitude of $\mathbf{q}_{j-1}$ contained in $\bar{\mathbf{r}}_j$ may be found similarly by multiplying Eq. (9) by $\mathbf{q}_{j-1}^T\mathbf{M}$. In this case all terms except the third vanish by orthogonality, and the coefficient of β_j is unity, so $\beta_j = \mathbf{q}_{j-1}^T\mathbf{M}\bar{\mathbf{r}}_j$. But, using Eq. (8) to eliminate $\bar{\mathbf{r}}_j$ this gives $\beta_j = \mathbf{q}_{j-1}^T\mathbf{M}\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j$ and applying the transpose of Eq. (8) to the $\mathbf{q}_{j-1}^T$ vector gives

$$\beta_j = \bar{\mathbf{r}}_{j-1}^T\mathbf{M}\mathbf{q}_j \tag{12}$$

Finally, expanding $\bar{\mathbf{r}}_{j-1}$ in terms of its pure component, $\mathbf{r}_{j-1}$, and the preceding Lanczos vectors, as in Eq. (9), the transpose of Eq. (12) becomes

$$\beta_j = \mathbf{q}_j^T\mathbf{M}\mathbf{r}_{j-1} + \alpha_{j-1}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-1} + \beta_{j-1}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-2} + \gamma_{j-1}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-3} + \cdots \tag{13}$$

It is evident that all terms except the first vanish on the right hand side. Now $\mathbf{q}_j$ is the vector obtained by normalizing $\mathbf{r}_{j-1}$ i.e.

$$\mathbf{q}_j = \frac{1}{\|\mathbf{r}_{j-1}\|_M}\mathbf{r}_{j-1} \tag{14}$$

where $\|\mathbf{r}_{j-1}\|_{\mathbf{M}} = (\mathbf{r}_{j-1}^T\mathbf{M}\mathbf{r}_{j-1})^{1/2}$. Using this expression for $\mathbf{q}_j$ in Eq. (13), we obtain β_j as $\beta_j = \frac{1}{\|\mathbf{r}_{j-1}\|_{\mathbf{M}}}\mathbf{r}_{j-1}^T\mathbf{M}\mathbf{r}_{j-1}$ or

$$\beta_j^2 = \mathbf{r}_{j-1}^T\mathbf{M}\mathbf{r}_{j-1} \tag{15}$$

This is an alternate expression, to (12), for evaluating β_j. In [17] Scott established that computing β_j using the expression in Eq. (15) is preferable to that given in (12) and numerical experiments also confirm his results. Using the value of β_j obtained via Eq. (15) ensures that the Lanczos vectors are properly normalized even if orthogonality $\mathbf{q}_{j+1}^T\mathbf{M}\mathbf{q}_{j-1}$ is not exactly zero.

Continuing in the same way, the amplitude of $\mathbf{q}_{j-2}$ contained in $\bar{\mathbf{r}}_j$ is found to be

$$\gamma_j = \mathbf{q}_{j-2}^T\mathbf{M}\bar{\mathbf{r}}_j \tag{16}$$

Following the procedure used to derive Eq. (13), this leads to

$$\gamma_j = \mathbf{q}_j^T\mathbf{M}\mathbf{r}_{j-2} + \alpha_{j-2}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-2} + \beta_{j-2}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-3} + \gamma_{j-2}\mathbf{q}_j^T\mathbf{M}\mathbf{q}_{j-4} + \cdots \tag{17}$$

But, using the normalizing relationship equivalent to Eq. (14), $\mathbf{r}_{j-2} = \beta_{j-1}\mathbf{q}_{j-1}$. Hence, when this is substituted into Eq. (17) all terms on the right hand side vanish, with the result that $\gamma_j = 0$. A corresponding procedure could be used to demonstrate that all further terms in the expansion for $\bar{\mathbf{r}}_j$, Eq. (7), vanish; in other words, the orthogonalization procedure used in generating each Lanczos vector need be applied only to the previous two vectors.

In summary, the Lanczos algorithm may be expressed by the following sequence of equations:

$$\left\{\begin{array}{ll} (a) & \mathbf{q}_j = \frac{1}{\beta_j}\mathbf{r}_{j-1} \\ (b) & \bar{\mathbf{r}}_j = \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j \\ (c) & \alpha_j = \mathbf{q}_j^T\mathbf{M}\bar{\mathbf{r}}_j \\ (d) & \mathbf{r}_j = \bar{\mathbf{r}}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1} \\ (e) & \beta_{j+1} = (\mathbf{r}_j^T\mathbf{M}\mathbf{r}_j)^{1/2} \end{array}\right. \tag{18}$$

The above process may be started from a given vector, $\mathbf{r}_0$, with $\mathbf{q}_0 = 0$ and $\beta_1 = (\mathbf{r}_0^T\mathbf{M}\mathbf{r}_0)^{1/2}$. At a typical step, j, the Lanczos algorithm computes $\mathbf{q}_j$, α_j, and β_{j+1} in order. In addition to the storage needs of $\mathbf{K}_\sigma$ and $\mathbf{M}$, the algorithm requires storage for 5 vectors of length n; one for each of the vectors, $\mathbf{q}_{j-1}$, $\mathbf{q}_j$, $\mathbf{M}\mathbf{q}_j$, $\bar{\mathbf{r}}_j$ and $\mathbf{r}_j$. The total cost for one step of the algorithm involves a multiply with $\mathbf{M}$, the solution of a system of equations with $\mathbf{K}_\sigma$ as the coefficient matrix, two inner products and three products of a scalar by a vector.

6. Reduction to Tridiagonal Form

Using the results of the previous section (9) can be rewritten as the three term relation

$$\mathbf{r}_j = \beta_{j+1}\,\mathbf{q}_{j+1} = \mathbf{K}_\sigma^{-1}\,\mathbf{M}\,\mathbf{q}_j - \mathbf{q}_j\,\alpha_j - \mathbf{q}_{j-1}\,\beta_j \tag{19}$$

where $\alpha_j = \mathbf{q}_j^T\,\mathbf{M}\,\mathbf{K}_\sigma^{-1}\,\mathbf{M}\,\mathbf{q}_j$ and $\mathbf{r}_j$ is normalized with respect to the mass matrix to obtain $\mathbf{q}_{j+1}$ with normalizing factor $\beta_{j+1} = (\mathbf{r}_j^T\mathbf{M}\mathbf{r}_j)^{\frac{1}{2}}$. After m Lanczos steps all the terms obtained from equation (19) can be arranged in a global matrix form

$$\left[\quad \mathbf{K}_\sigma^{-1}\,\mathbf{M} \quad\right]\left[\quad \mathbf{Q}_m \quad\right] - \left[\quad \mathbf{Q}_m \quad\right]\left[\,\mathbf{T}_m\,\right] = \left[\quad 0 \quad \middle|\, \mathbf{r}_m \right] = \mathbf{r}_m\,\mathbf{e}_m^T \tag{20}$$

Here $\mathbf{e}_m^T = (0,0,\cdots,0,1)$, $\mathbf{Q}_m$ is an $n \times m$ matrix with columns $\mathbf{q}_i$, $i = 1,2,\cdots m$, and $\mathbf{T}_m$ is a tridiagonal matrix of the form

$$\mathbf{T}_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot & \beta_m \\ & & & & & \beta_m & \alpha_m \end{bmatrix} \tag{21}$$

The orthogonality property of the Lanczos vectors, $\mathbf{Q}_m^T \mathbf{M} \mathbf{Q}_m = \mathbf{I}_m$, can be used in equation (20) to obtain

$$\mathbf{Q}_m^T \mathbf{M} \mathbf{K}_\sigma^{-1} \mathbf{M} \mathbf{Q}_m = \mathbf{T}_m \tag{22}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix.

Choosing the set of Lanczos vectors, $\mathbf{Q}_m$, for the trial vectors, the Rayleigh-Ritz procedure can be used to obtain the best approximation to the eigenvectors of (6). The approximating Ritz vectors will then be of the form

$$\mathbf{y}_i^{(m)} = \mathbf{Q}_m \mathbf{s}_i^{(m)} \qquad , i = 1, \cdots, m \tag{23}$$

When a residual vector, $[\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)}]$, associated with the pair $(\theta_i^{(m)}, \mathbf{y}_i^{(m)})$ is **M**-orthogonal to the set of Lanczos vectors, then $(\theta_i^{(m)}, \mathbf{y}_i^{(m)})$ is a Ritz pair. Accordingly

$$\mathbf{Q}_m^T\mathbf{M}[\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)}] = 0$$

Using the relation between the Ritz vector, $\mathbf{y}_i^{(m)}$ and the the Lanczos vectors, given in (23), together with the orthonormality condition of $\mathbf{Q}_j$, and the tridiagonal properties of the Lanczos vectors, Eq. (22), the above equation reduces to the tridiagonal eigenproblem

$$\mathbf{T}_m \, \mathbf{s}_i^{(m)} - \theta_i^{(m)}\mathbf{s}_i^{(m)} = 0 \tag{24}$$

Thus $(\theta_i^{(m)}, \mathbf{s}_i^{(m)})$ is an eigenpair of the tridiagonal matrix, $\mathbf{T}_m$. As the total number of the Lanczos vectors increase, i.e. as we take more Lanczos steps, the size of the tridiagonal matrix increases and the eigenvalues of $\mathbf{T}_m$ converge to the eigenvalues of the transformed problem (6), $\frac{1}{\lambda_i - \sigma}$. When $m = n$, the order of **K**, then $\theta_i^{(n)} = \frac{1}{\lambda_i - \sigma}$ for all i but we hope to stop long before $m = n$. The steps described by equation (18) is repeated until the all Ritz pairs $(\theta_i^{(m)}, \mathbf{y}_i^{(m)})$ have sufficiently converged to the desired eigenpairs.

Example:

$$\mathbf{K} = \frac{1}{5}\begin{bmatrix} 39 & -9 & 21 & -11 \\ -9 & 39 & -11 & 21 \\ 21 & -11 & 39 & -9 \\ -11 & 21 & -9 & 39 \end{bmatrix}$$

For this example we assume the mass matrix is identity. Then the eigenvalue matrix for this problem is $\Lambda = diag[\, 1/5 \,,\, 1/4 \,,\, 1/2 \,,\, 1 \,]$. The Lanczos algorithm presented here requires the solution of a linear system of equations with $\mathbf{K}$. We therefore give $\mathbf{K}^{-1}$ explicitly for convenience. Accordingly

$$\mathbf{K}^{-1} = \frac{1}{2}\begin{bmatrix} 6 & 0 & -3 & 1 \\ 0 & 6 & 1 & -3 \\ -3 & 1 & 6 & 0 \\ 1 & -3 & 0 & 6 \end{bmatrix}$$

Choosing a starting vector $\mathbf{r}_0 = [\, 1 \,,\, 0 \,,\, 0 \,,\, 0 \,]^T$ then

step 1:

$$\beta_1 = \|\mathbf{r}_0\| = 1 \,, \quad \mathbf{q}_1 = \frac{\mathbf{r}_0}{\beta_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{K}^{-1}\mathbf{q}_1 = \frac{1}{2}\begin{bmatrix} 6 \\ 0 \\ -3 \\ 1 \end{bmatrix},$$

$$\alpha_1 = \mathbf{q}_1^T\mathbf{K}^{-1}\mathbf{q}_1 = 3 \,, \quad \mathbf{r}_1 = \mathbf{K}^{-1}\mathbf{q}_1 - \alpha_1\mathbf{q}_1 = \frac{1}{2}\begin{bmatrix} 0 \\ 0 \\ -3 \\ 1 \end{bmatrix}, \quad \mathbf{T}_1 = [\, 3 \,]$$

step 2:

$$\beta_2 = \|\mathbf{r}_1\| = \frac{\sqrt{10}}{2} \,, \quad \mathbf{q}_2 = \frac{\mathbf{r}_1}{\beta_2} = \frac{1}{\sqrt{10}}\begin{bmatrix} 0 \\ 0 \\ -3 \\ 1 \end{bmatrix}, \quad \mathbf{K}^{-1}\mathbf{q}_2 = \frac{1}{\sqrt{10}}\begin{bmatrix} 5 \\ -3 \\ -9 \\ 3 \end{bmatrix},$$

$$\alpha_2 = \mathbf{q}_2^T\mathbf{K}^{-1}\mathbf{q}_2 = 3 \,, \quad \mathbf{r}_2 = \mathbf{K}^{-1}\mathbf{q}_2 - \alpha_2\mathbf{q}_2 - \beta_2\mathbf{q}_1 = \frac{1}{\sqrt{10}}\begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \end{bmatrix},$$

$$\mathbf{T}_2 = \begin{bmatrix} 3 & \sqrt{10}/2 \\ \sqrt{10}/2 & 3 \end{bmatrix}$$

step 3:

$$\beta_3 = \|\mathbf{r}_2\| = \frac{3}{\sqrt{10}}\,, \quad \mathbf{q}_3 = \frac{\mathbf{r}_2}{\beta_3} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{K}^{-1}\mathbf{q}_3 = \frac{1}{2}\begin{bmatrix} 0 \\ -6 \\ -1 \\ 3 \end{bmatrix},$$

$$\alpha_3 = \mathbf{q}_3^T\mathbf{K}^{-1}\mathbf{q}_3 = 3\,, \quad \mathbf{r}_3 = \mathbf{K}^{-1}\mathbf{q}_3 - \alpha_3\mathbf{q}_3 - \beta_3\mathbf{q}_2 = \frac{1}{5}\begin{bmatrix} 0 \\ 0 \\ 2 \\ 6 \end{bmatrix},$$

$$\mathbf{T}_3 = \begin{bmatrix} 3 & \sqrt{10}/2 & \\ \sqrt{10}/2 & 3 & 3/\sqrt{10} \\ & 3/\sqrt{10} & 3 \end{bmatrix}$$

step 4:

$$\beta_4 = \|\mathbf{r}_3\| = \frac{4}{\sqrt{10}}\,, \quad \mathbf{q}_4 = \frac{\mathbf{r}_3}{\beta_4} = \frac{1}{\sqrt{10}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{bmatrix}, \quad \mathbf{K}^{-1}\mathbf{q}_4 = \frac{1}{\sqrt{10}}\begin{bmatrix} 0 \\ -8 \\ 6 \\ 18 \end{bmatrix},$$

$$\alpha_4 = \mathbf{q}_4^T\mathbf{K}^{-1}\mathbf{q}_4 = 3\,, \quad \mathbf{r}_4 = \mathbf{K}^{-1}\mathbf{q}_4 - \alpha_4\mathbf{q}_4 - \beta_4\mathbf{q}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\mathbf{T}_4 = \begin{bmatrix} 3 & \sqrt{10}/2 & & \\ \sqrt{10}/2 & 3 & 3/\sqrt{10} & \\ & 3/\sqrt{10} & 3 & 4/\sqrt{10} \\ & & 4/\sqrt{10} & 3 \end{bmatrix}$$

The eigenvalues of the tridiagonal matrix converge to the inverses of the eigenvalues of **K** in this example. This can be demonstrated by computing the eigenvalues of the tridiagonal at each step as shown in table 1.

j	Eigenvalues of $\mathbf{T}_j$
1	3.0000
2	1.4189, 4.5811
3	1.1561, 3.0000, 4.8439
4	1.0000, 2.0000, 4.0000, 5.0000

Table 1. Convergence of the Ritz values for the small 4×4 example.

In figure 1. we plot the Ritz values for a larger example. Notice the convergence at both end of the spectrum.

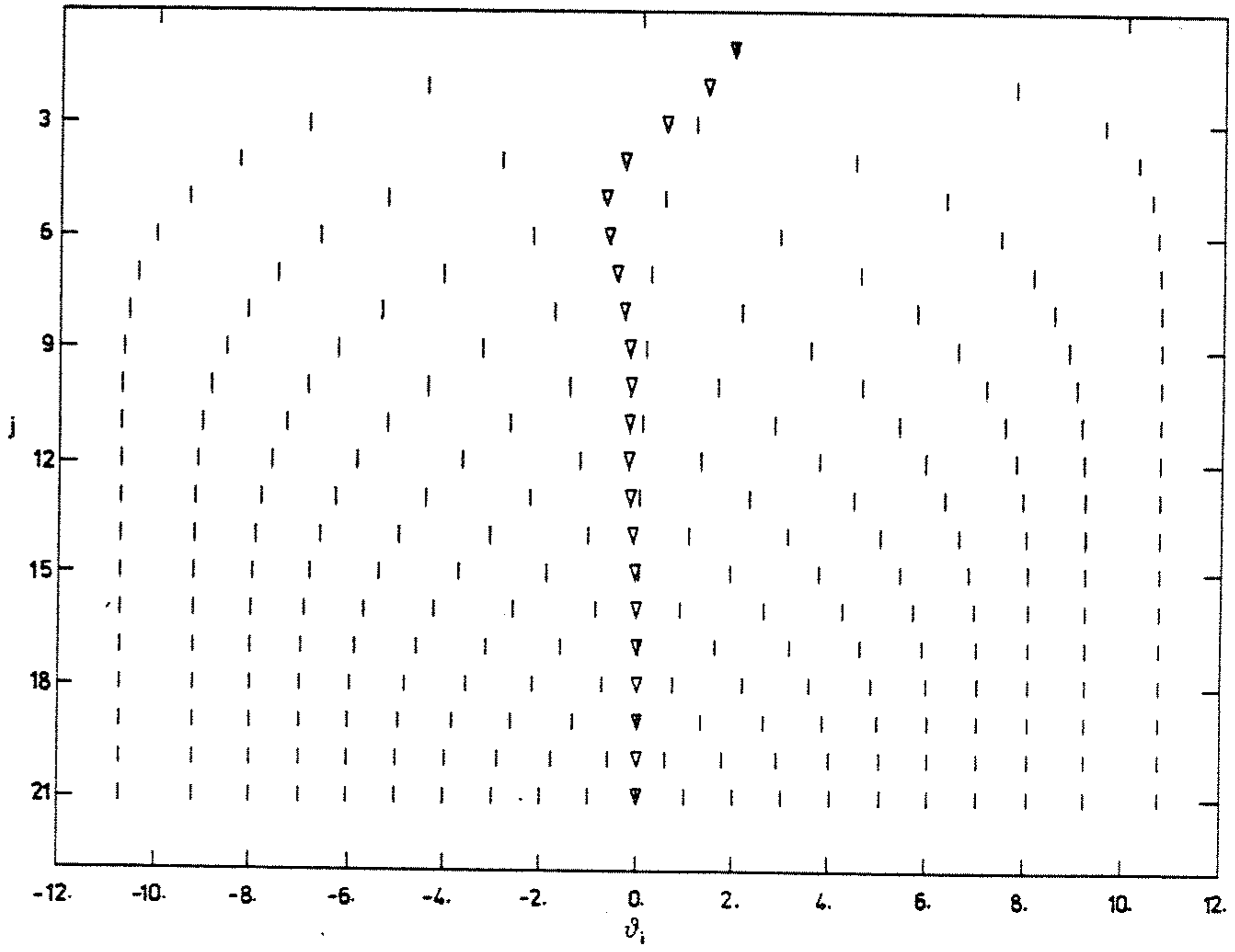

Figure 1. A typical pattern for the progress of the eigenvalues of the tridiagonal matrix, $\mathbf{T}_j$. ∇ indicates the diagonals of $\mathbf{T}_j$.

7. Convergence Criterion for Eigenvalues

The eigenvalues, $\theta_i^{(j)}$, $i=1,...,j$, of the tridiagonal $\mathbf{T}_j$ are the Rayleigh-Ritz approximations to eigenvalues of (6). As j increases these Ritz values get closer to the eigenvalues they are approximating. Often users wait until the change in the computed quantities are within a specified tolerance; that is

$$\frac{|\theta^{(j-1)} - \theta^{(j)}|}{|\theta^{(j)}|} \leq tol \tag{25}$$

Suppose one is performing inverse iteration with a specified shift, σ, in an interval of interest $[\sigma_1 , \sigma_2]$; that is

$$\sigma = \gamma\sigma_1 + (1-\gamma)\sigma_2$$

where γ is a positive number less than one. Further, assume that the problem has eigenvalues, λ_a and λ_b, outside this interval with $\lambda_a < \sigma_1$ and $\lambda_b > \sigma_2$. Then the corresponding eigenvalues of the transformed problem, Eq. (6), are $\nu_a = \dfrac{1}{\lambda_a - \sigma}$ and $\nu_b = \dfrac{1}{\lambda_b - \sigma}$, with $\nu_a < 0$ and $\nu_b > 0$. Now, let us perform a step of inverse iteration method with an unfortunate starting vector, $\mathbf{x}^{(1)} = \mathbf{z}_a\sin\psi + \mathbf{z}_b\cos\psi$. Here, $\mathbf{z}_a$ and $\mathbf{z}_b$ are eigenvectors corresponding to λ_a and λ_b respectively. We assume that the eigenvectors are normalized with respect to the mass matrix. Then $\mathbf{x}^{(1)}$ will also be normalized. The Ritz value due to $\mathbf{x}^{(1)}$ is

$$\begin{aligned}\theta^{(1)} &= \mathbf{x}^{(1)T}\mathbf{K}\mathbf{x}^{(1)} \\ &= \lambda_a\sin^2\psi + \lambda_b\cos^2\psi\end{aligned}$$

After one step of inverse iteration the improved vector is

$$\mathbf{x}^{(2)} = \frac{1}{[\nu_a^2\sin^2\psi + \nu_b^2\cos^2\psi]^{\frac{1}{2}}}(\nu_a\mathbf{z}_a\sin\psi + \nu_b\mathbf{z}_b\cos\psi)$$

and the Ritz value associated with $\mathbf{x}^{(2)}$ is

$$\theta^{(2)} = \frac{1}{[\nu_a^2\sin^2\psi + \nu_b^2\cos^2\psi]^{\frac{1}{2}}}(\nu_a^2\lambda_a\sin^2\psi + \nu_b^2\lambda_b\cos^2\psi)$$

Now if $\lambda_a = \sigma - \tau$ and $\lambda_b = \sigma + \tau$ then $\nu_a = -\dfrac{1}{\tau}$ and $\nu_b = \dfrac{1}{\tau}$ then

$$\theta^{(1)} = \theta^{(2)} = (\sigma - \tau)\sin^2\psi + (\sigma + \tau)\cos^2\psi = \sigma + \tau\cos 2\psi$$

These θ's, referred to as "ghost" eigenvalues in [16], satisfies the condition of Eq. (25), and therefore will be accepted as an eigenvalue by test (25). Further, an appropriate choice for ψ can result in a Ritz value that is in the interval of interest. This behavior, although rare, has been observed in certain implementations of subspace iteration method [16] and reflects a serious difficulty. We refer to this as misconvergence [14] and as we observed it can not be detected by a natural criterion such as (25). A rigorous inexpensive termination criterion can be obtained using the residual error bound given bellow.

Consider a pair $(\theta, \mathbf{y})$ which are approximations to an eigenpair $(\nu, \mathbf{z})$ of Eq (6), where ν is the closest eigenvalue of (6) to θ. For simplicity we consider only normalized Ritz vectors, $\|\mathbf{y}\|_M = 1$ where $\|\mathbf{y}\|_M = (\mathbf{y}^T\mathbf{M}\mathbf{y})^{\frac{1}{2}}$ is the inertial norm of $\mathbf{y}$. The residual vector associated with this approximation is given by

$$\mathbf{r} = \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y} - \theta\mathbf{y} \tag{26}$$

The norm of the residual vector can then be used to assess the accuracy of θ through the inequality

$$|\nu - \theta| \leq \|\mathbf{r}\|_M \tag{27}$$

For the proof and an in depth study see [12]. This result allows us to predict the accuracy of any candidate eigenpair of a matrix.

We use the residual vector associated with the Ritz pair $(\theta_i^{(m)}, \mathbf{y}_i^{(m)})$ given by Eqs. (23) and (24) to check for their convergence. The tridiagonal property of the Lanczos algorithm, Eq. (20), greatly simplifies the computation of the residual norm. Postmultiplying Eq. (20) by $\mathbf{s}_i^{(m)}$ leads to

$$\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{Q}_m\mathbf{s}_i^{(m)} - \mathbf{Q}_m\mathbf{T}_m\mathbf{s}_i^{(m)} = \mathbf{r}_m\mathbf{e}_m^T\mathbf{s}_i^{(m)} \tag{28}$$

Since $\mathbf{s}_i^{(m)}$ is an eigenvector of $\mathbf{T}_m$ (see Eq. 23), we can replace $\mathbf{T}_m\mathbf{s}_i^{(m)}$ by $\theta_i^{(m)}\mathbf{s}_i^{(m)}$. Further, by definition (Eq. 24), $\mathbf{Q}_m\mathbf{s}_i^{(m)}$ is simply the Ritz vector $\mathbf{y}_i^m$ and so Eq. (28) will reduce to

$$\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)} = \mathbf{r}_m\mathbf{e}_m^T\mathbf{s}_i^{(m)} \tag{29}$$

Taking norms

$$\begin{aligned} \| \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)} \|_M &= \| \mathbf{r}_m \mathbf{e}_m^T \mathbf{s}_i^{(m)} \|_M \\ &= \| \mathbf{r}_m \|_M \, | \mathbf{e}_m^T \mathbf{s}_i^{(m)} | \\ &= \beta_{j+1} | \varsigma_i | \end{aligned} \tag{30}$$

where $\varsigma_i = \mathbf{e}_m^T \mathbf{s}_i^{(m)}$ is the bottom element of $\mathbf{s}_i^{(m)}$, the normalized eigenvector of $\mathbf{T}_m$. β_{j+1} is a scalar quantity that is computed in the course of the Lanczos process. The bottom element of the eigenvector of $\mathbf{T}_m$ can be obtained at little cost, once the associated eigenvalue is found. Therefore, the residual norm associated with a Ritz pair can be computed as $\rho_{ji} = \beta_{j+1} | \varsigma_i |$ and the Ritz value is considered converged once $\rho_{ji} < tol$. There is no need to compute $\mathbf{y}_i^{(m)}$ until it has converged.

Multiple Eigenvalues

In exact arithmetic the simple Lanczos algorithm can not compute a second copy of a repeated eigenvalue. Suppose λ is an eigenvalue with multiplicity two. Then there is no unique eigenvector associated with λ. In fact one can obtain two vectors, $\mathbf{z}_1$ and $\mathbf{z}_2$, such that any linear combination of these vectors is also an eigenvector of λ. Moreover, it is possible to obtain a linear combination of $\mathbf{z}_1$ and $\mathbf{z}_2$, $\bar{\mathbf{z}} = \xi_1 \mathbf{z}_1 + \xi_2 \mathbf{z}_2$, that is orthogonal to $\mathbf{q}_1$. Then the eigenvector $\bar{\mathbf{z}}$ will be orthogonal to all the subsequent Lanczos vectors. Therefore only one copy of λ will be found.

Fortunately, the above argument only holds in exact arithmetic. In finite precision, roundoff comes to rescue. If $\mathbf{q}_1$ is perfectly orthogonal to an eigenvector, $\bar{\mathbf{z}}$, then because of roundoff error $\mathbf{q}_2$ will have a small component along $\bar{\mathbf{z}}$. This component, although tiny, will eventually grow such that $\bar{\mathbf{z}}$ can be represented by a linear combination of the computed Lanczos vectors. However, the second copy of λ will converge some steps after the first has converged.

8. Loss of Orthogonality

In a previous section we derived the equations governing the Lanczos algorithm. These relations are only satisfied by quantities obtained in exact arithmetic. In finite precision, however, each computation introduces a small error and therefore each computed quantity will differ from its exact counterpart. Our objective here is to describe the effect of roundoff error on the Lanczos process. For this purpose we need to introduce an important quantity that measures the accuracy of the arithmetic. Let ϵ be the smallest number in the computer such that $1 + \epsilon > 1$. It is known as the unit roundoff error.

Although the tridiagonal relation, Eq. (20), is preserved to within roundoff, the **M**-orthogonality property of the Lanczos vectors completely breaks down after a certain number of steps depending on ϵ and the distribution of the eigenvalues of Eq. (6). The Lanczos vectors, which are orthogonal in exact arithmetic, not only loose their orthogonality, but may even become linearly dependent. Initially it was believed that loss of orthogonality is due to cancellations that occurs each time $\mathbf{r}_j$ is evaluated using Eq. (19). This step is simply **M**-orthogonalization of $\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j$ against $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$. Comparing the final vector resulting from this computation to the starting vector, one can obtain a measure of the cancellation that occurs in this step; that is the ratio

$$\chi_j = \frac{\|\mathbf{r}_j\|_{\mathbf{M}}^2}{\|\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j\|_{\mathbf{M}}} = \frac{\beta_{j+1}^2}{\beta_j^2 + \alpha_j^2 + \beta_{j-1}^2} \tag{31}$$

indicates how much cancellation has occurred. χ_j is the sine of the angle between the vector $\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j$ and the plane containing the vectors $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$. When this χ_j is zero it indicates that in the j-th Lanczos step we are orthogonalizing a vector that is already in this plane against $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$ and therefore complete cancellation occurs. In practice χ_j rarely drops bellow $\frac{1}{16}$. When χ_j drops to say $\frac{1}{100}$ it indicates that $\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j$ is nearly parallel to this plane and therefore at the end of this Lanczos step the computed $\mathbf{r}_j$ may not be orthogonal to the plane containing $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$ to working accuracy. In such cases $\mathbf{r}_j$ should be orthogonalized against $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$ a second time. For a long time it was believed that loss of orthogonality was solely due to this cancellation. Indeed if this were the case then one would have no option other than a complete reorthogonalization at each step. However, as we soon shall see a completely different mechanism is at work.

Suppose for the moment the algorithm were carried out in exact arithmetics for j steps, except that at some step $k < j$ a small error was introduced into the computation of $\mathbf{q}_k$. The first k-1 Lanczos vectors will be perfectly **M**-orthonormal, but they will not be orthogonal to all the vectors computed after the k-th step. The error introduced at step k will be amplified in the subsequent steps to such an extent that linear independency may also be lost.

Hence, the loss of orthogonality can be viewed as the subsequent amplification of the error introduced after each computation. To analyze the way in which orthogonality deteriorates we let $\mathbf{Q}_m$ denote the computed Lanczos vectors and define the following matrix

$$\mathbf{H}_m = \mathbf{Q}_m^T\mathbf{M}\mathbf{Q}_m \tag{32}$$

where the i,j component of $\mathbf{H}_m$ is $\eta_{i,j} = \mathbf{q}_i^T\mathbf{M}\mathbf{q}_j$. In exact arithmetic $\mathbf{H}_j$ is the identity matrix. The off-diagonals of $\mathbf{H}_j$ will depend on ϵ, the unit roundoff error. Further, Eq. (19) will be satisfied by the computed quantities only to within roundoff error. Now, multiply Eq. (19) by $\mathbf{q}_i^T\mathbf{M}$ and use the above definition to get the approximate relation

$$\mathbf{q}_i^T\mathbf{M}\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j \approx \alpha_j\eta_{i,j} + \beta_j\eta_{i,j-1} + \beta_{j+1}\eta_{i,j+1} \tag{33}$$

A similar equation can be obtained when the above procedure is repeated for the i-th Lanczos step.

$$\mathbf{q}_j^T\mathbf{M}\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_i \approx \alpha_i\eta_{j,i} + \beta_i\eta_{j,i-1} + \beta_{i+1}\eta_{j,i+1} \tag{34}$$

By symmetry of $\mathbf{M}\mathbf{K}_\sigma^{-1}\mathbf{M}$, the terms on the left hand side of Eqs. (33) and (34) are equal and therefore can be eliminated by subtraction, resulting in the relation

$$\beta_{j+1}\eta_{j+1,i} \approx \beta_{i+1}\eta_{j,i+1} + (\alpha_i - \alpha_j)\eta_{j,i} + \beta_i\eta_{j,i-1} - \beta_j\eta_{j,i-1} \tag{35}$$

This recursion holds for $j \geq 2$ and $1 \leq i \leq j$-1 and starts by assuming that the diagonal of $\mathbf{H}_m$ is the Identity, $\eta_{j,j} = 1$ for all $j \geq 1$, and the first off-diagonal of $\mathbf{H}_m$ is at roundoff level, $\eta_{j,j-1} = \epsilon$ for $j \geq 2$. The above relation provides a means of estimating the elements of a column of $\mathbf{H}_m$ from the elements of $\mathbf{T}_m$ and the elements in the previous columns of $\mathbf{H}_m$. This recursion can be restated in the vector form

$$\beta_{j+1}\mathbf{h}_{j+1} \approx \mathbf{T}_{j-1}\mathbf{h}_j - \alpha_j\mathbf{h}_j - \beta_j\mathbf{h}_{j-1} \tag{36}$$

where $\mathbf{h}_{j-1}$, $\mathbf{h}_j$ and $\mathbf{h}_{j+1}$ are vectors of length j-1 containing the top j-1 elements of the j-1, j, and $j+$1-th columns of $\mathbf{H}_m$. Here, the bottom element of $\mathbf{h}_{j-1}$ is zero. Then, all the terms in $\mathbf{h}_j$ depend on ϵ. Taking norms we can show

$$\begin{aligned}\beta_{j+1}\|\mathbf{h}_{j+1}\| &\le (\|\mathbf{T}_{j-1}\| + |\alpha_j|)\|\mathbf{h}_j\| + \beta_j\|\mathbf{h}_{j-1}\| \\ &\le 2\|\mathbf{T}_j\|\max(\|\mathbf{h}_j\|\ ,\ \|\mathbf{h}_{j-1}\|)\end{aligned} \tag{37}$$

This result shows that the level of orthogonality can grow by at most a factor of $2\|\mathbf{T}_j\|/\beta_{j+1}$ after each step. A drop in the value of β_{j+1} can result in a sudden loss of orthogonality but this rarely occurs.

An alternative characterization of the pattern in which orthogonality is lost was presented by Paige [9,10,11]. Instead, of examing the vector $\mathbf{h}_{j+1} = \mathbf{Q}_j^T\mathbf{M}\mathbf{q}_{j+1}$ he looked at a linear combination of the components in this vector. To be more specific he examined the inner product between each Ritz vector $\mathbf{y}_i^{(j)}$ given by Eq. (23) and $\mathbf{q}_{j+1}$. That is

$$\begin{aligned}\mathbf{y}_i^{(j)\,T}\mathbf{M}\mathbf{q}_{j+1} &= \mathbf{s}_i^{(j)\,T}\mathbf{Q}_j^T\mathbf{M}\mathbf{q}_{j+1} \\ &= \mathbf{s}_i^{(j)\,T}\mathbf{h}_{j+1}\end{aligned} \tag{38}$$

In exact arithmetic this value is zero. However, in his work Paige showed that

$$\mathbf{y}_i^{(j)\,T}\mathbf{M}\mathbf{q}_{j+1} = \frac{\gamma_{ji}\,\epsilon\,\|\mathbf{T}_j\|}{\beta_{j+1}|\varsigma_i|} \tag{39}$$

Recall that ς_i is the bottom element of $\mathbf{s}_i^{(j)}$, the i-th eigenvector of $\mathbf{T}_j$. γ_{ji} is a scalar quantity usually close to unity. We omit the derivation of this result and refer the interested reader to [12]. Note that Paige's result also shows that a sudden drop in β_{j+1} can result in a severe loss of orthogonality. Moreover, recall the quantity in the denominator is also a measure of the convergence of the Ritz value $\theta_i^{(j)}$ (see Eq. 30). The only way the left side of Eq. (39) can rise up to values like 0.1 is for ρ_{ji} ($= \beta_{j+1}|\varsigma_i|$) to drop down to $10\gamma_{ji}\epsilon\|\mathbf{T}_j\|$, so

Loss of orthogonality $\Longrightarrow$ Convergence of a Ritz value

When only a single Ritz value converges then $\mathbf{q}_{j+1}$ loses orthogonality by tilting towards the converged Ritz vector which in turn is a linear combination of the previous Lanczos vectors. When more than one Ritz vector converges simultaneously the picture is more complicated. In this case, $\mathbf{q}_{j+1}$ tilts toward a linear combination of these vectors.

Return of Banished Ritz Vectors

In theory, if two successive Lanczos vectors, $\mathbf{q}_{j-1}$ and $\mathbf{q}_j$, are orthogonal to an eigenvector,

$\mathbf{z}$, then all the subsequent Lanczos vectors will also be orthogonal to $\mathbf{z}$. In practice, however, a converged Ritz vector, $\mathbf{y}_i$, will not remain orthogonal to all the subsequent Lanczos vectors. This is a consequence of the same mechanism by which multiple eigenvalues are found. Roundoff errors will add to each Lanczos vector a small component of $\mathbf{y}_i$ which will grow eventually to such a magnitude that orthogonality to $\mathbf{y}_i$ is lost. This phenomenon can also be observed in the inverse iteration method; that is, orthogonalizing the starting vector against the first eigenvector, does not guarantee convergence of the iteration vectors to the next eigenvector.

The state of orthogonality between a converged Ritz vector, $\mathbf{y}_i$, and the current Lanczos vector, $\mathbf{q}_j$ can be measured by the component of $\mathbf{y}_i$ along $\mathbf{q}_j$. We define $\tau_j = \mathbf{y}_i^T\mathbf{M}\mathbf{q}_j$. Then multiplying Eq. (19) by $\mathbf{y}_i^T\mathbf{M}$ and considering the effect of roundoff on (19) yields

$$\beta_{j+1}\mathbf{y}_i^T\mathbf{M}\mathbf{q}_{j+1} \approx \mathbf{y}_i^T\mathbf{M}\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j - \alpha_j\mathbf{y}_i^T\mathbf{M}\mathbf{q}_j - \beta_j\mathbf{y}_i^T\mathbf{M}\mathbf{q}_{j-1}$$

Using the fact that $\mathbf{y}_i$ is a converged Ritz value; that is $\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{y}_i = \theta_i\mathbf{y}_i$, together with the definition of τ_j we obtain

$$\tau_{j+1} \approx \frac{(\theta_i - \alpha_j)\tau_j - \beta_j\tau_{j-1}}{\beta_{j+1}} \tag{40}$$

This recurrence can be updated for each converged Ritz value, θ_i. The magnitude of τ_j can be used as an indicator for loss of orthogonality against a converged Ritz vector otherwise referred to as the return of banished Ritz vector.

9. Restoring Orthogonality

In this section we look at a number of preventive measures that one can adopt to maintain a certain level of orthogonality. Lanczos [5] was aware of the effects of roundoff on the algorithm when he presented his work. He proposed that the newly computed Lanczos vector, $\mathbf{q}_{j+1}$, be explicitly orthogonalized against all the preceding vectors at the end of each step j. We will refer to this technique as "full reorthogonalization" method. This scheme is also adopted in [3,20]. With this procedure the Lanczos vectors will meet the stringent requirement

$$|\mathbf{q}_i^T\mathbf{M}\mathbf{q}_j| < n\epsilon \tag{41}$$

Although this scheme increases the overall cost of an eigenvalue computation, for short Lanczos runs (when the number of Lanczos steps is less than the half bandwidth of $\mathbf{K}$) the increase in cost is small compared to the cost of solving a system of equations with $\mathbf{K}_\sigma$. For longer runs the cost of a full reorthogonalization step will begin to dominate the cost of a Lanczos step although vector computers will delay this effect.

The orthogonality condition of Eq. (41) can be replaced by a more relaxed condition

$$|\mathbf{q}_i^T \mathbf{M} \mathbf{q}_j| < \sqrt{n\epsilon} \tag{42}$$

We refer to this as the "Semi-Orthogonality" condition, and to procedures that adopt the weaker condition as "Selective Orthogonalization" methods. Imposing the more relaxed condition can result in considerable reduction in the number of operations in the reorthogonalization step and semi-orthogonality is sufficient to make $\mathbf{T}_j$ exact to working precision.

We Consider two different reorthogonalization schemes that adopt the more relaxed condition of Eq. (42).

A. *Orthogonalization against Ritz vectors:* This procedure is a consequence of the result of Paige [9,10,11]. As soon as a Ritz value converges, its corresponding Ritz vector is computed and the component of this vector along $\mathbf{q}_{j+1}$ is purged. Further, for each converged Ritz value the three term recurrence for τ_j is updated and whenever τ_j becomes greater than $\sqrt{\epsilon}$ in absolute value, it signals that the component of the corresponding eigenvector has grown too much. So the new Lanczos vector is orthogonalized against this known eigenvector [13,16].

B. *Orthogonalization against previous Lanczos vectors:* This schemes can be based on either of the techniques given in [4,18]. In [18] the vector $\mathbf{h}_{j+1}$ is updated using Eq. (36) and the magnitude of its elements is monitored. Whenever the i-th element of $\mathbf{h}_{j+1}$ is greater than $\sqrt{\epsilon}$ then semi-orthogonality is lost between $\mathbf{q}_{j+1}$ and $\mathbf{q}_i$. At this step the appropriate Lanczos vectors are brought in from secondary store and their components along $\mathbf{q}_{j+1}$ are removed. This scheme is also adopted in [8].

Both schemes indicate loss of orthogonality at about the same step. The first method performs orthogonalization against fewer vectors and therefore costs less. However, alone it has some shortcomings. We mentioned earlier that the quantity γ_{ji} in Eq. (39) is close to one. This is only true in the early stages of a Lanczos run. So long as $|\gamma_{ji}| < 1$ scheme A maintains semi-orthogonality, but as soon as γ_{ji} becomes larger than one semi-orthogonality may be lost. For long runs this characteristic appears in the form of a gradual loss of semi-orthogonality. For this reason we do not use scheme A on its own. However, for short Lanczos runs where only a very few (less than 10) eigenpairs are wanted scheme A is very effective.

We use a combination of the above two schemes. Whenever possible we orthogonalize against a Ritz vector (scheme A) because of lower costs. We updates $\mathbf{h}_{j+1}$ using (36) and monitors its elements. If any of the elements of $\mathbf{h}_{j+1}$ is greater than $\sqrt{\epsilon}$ then semi-orthogonality may have been lost. This is the only procedure we use to determine loss of semi-orthogonality which can occur in two possible ways:

(i) Convergence of a Ritz value,

(ii) Growth of components of computed eigenvector along the Lanczos vector.

We take different actions for each of these. If the observed orthogonality loss is due to (i), then we perform a step of scheme B. The Lanczos vectors are brought in from secondary store and the newly computed Lanczos vector, $\mathbf{r}_j$, is orthogonalized against them. Further, to minimize data transfers from secondary storage we also compute some of the converged Ritz vectors at the same time. Therefore, the computation of newly converged Ritz vectors are delayed until the Lanczos vectors are brought back for reorthogonalization. We should note that when (i) occurs both methods A and B would require recalling the Lanczos vectors from secondary store, but for different reasons; the first for computing Ritz vectors and the later for reorthogonalization. We perform both. Our method would clearly require more operations than scheme A but has the following two advantages:

(a) An eigenvector is computed few steps after the convergence of its eigenvalue and therefore has more than half correct digits.

(b) The gradual loss of orthogonality mentioned above will not occur with the combined procedure.

However, if loss of orthogonality is due to (ii), then we can use scheme A; that is orthogonalize against a computed Ritz vector. For this reason the τ recurrence, Eq. (40) is also monitored to establish which Ritz vector, $\mathbf{y}_k$, has contaminated $\mathbf{q}_{j+1}$.

The orthogonalization of $\mathbf{q}_{j+1}$ against $\mathbf{y}_k$ alters the state of orthogonality among the Lanczos vectors. The modified $\overline{\mathbf{q}}_{j+1}$ is given by

$$\overline{\mathbf{q}}_{j+1} = \mathbf{q}_{j+1} - \xi_k \mathbf{y}_k$$

where $\xi_k = \mathbf{y}_k^T \mathbf{M} \mathbf{q}_{j+1}$. Let $\overline{\mathbf{h}}_{j+1} = \mathbf{Q}_j^T \mathbf{M} \overline{\mathbf{q}}_{j+1}$. Then

$$\begin{aligned} \overline{\mathbf{h}}_{j+1} &= \mathbf{Q}_j^T \mathbf{M}(\mathbf{q}_{j+1} - \xi_k \mathbf{y}_k) \\ &= \mathbf{h}_{j+1} - \xi_k \mathbf{s}_k \end{aligned} \tag{43}$$

An approximation to ξ_k can be obtained via

$$\xi_k = \mathbf{y}_k^T \mathbf{M} \mathbf{q}_{j+1} = \mathbf{s}_k^T \mathbf{Q}_j^T \mathbf{M} \mathbf{q}_{j+1} \approx \mathbf{s}_k^T \mathbf{h}_{j+1}$$

If the approximate result for ξ_k is used then (43) reduces to the orthogonalization of $\mathbf{h}_{j+1}$ against $\mathbf{s}_k$. This simple step is used to modify $\mathbf{h}_{j+1}$ to reflect the changes caused by orthogonalizing $\mathbf{q}_{j+1}$ against a Ritz vector. The flowchart in figure 4 gives a global view of our implementation of selective orthogonalization algorithm.

When it becomes necessary to restore semi-orthogonality, the computed vector, $\mathbf{q}_{j+1}$, must be orthogonalized against some linear combination of the previous Lanczos vectors. $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$ remain unchanged at the end of such step. But, at the next step, $\mathbf{q}_j$ appears again in the computation of $\mathbf{q}_{j+2}$. If no action is taken then $\mathbf{q}_{j+2}$ will be contaminated by $\mathbf{q}_j$ and the reorthogonalization efforts of the previous step would be wasted. Therefore a second reorthogonalization step must be performed. If the Lanczos vectors or the converged Ritz vectors reside in secondary store then they must be retrieved in two successive steps.

Alternatively, at the same time the orthogonality state of $\mathbf{q}_{j+1}$ is being restored, one can also perform similar modifications on $\mathbf{q}_j$. Then at the end of the next step, no reorthogonalization of $\mathbf{q}_{j+2}$ will be necessary. The number of operations for this scheme is the same as that of the scheme above, but vectors are retrieved only once and therefore the I/O overhead is halved.

10. LANSEL Package

In this section we describe all the ingredients that go into the LANSEL eigenpackage and provide sufficeint detail for installing LANSEL into a finite element program. The steps in each subroutine are described in sequence followed by the listing. Higher level routines are described first.

User Supplied Subroutines

The only interface between the algorithm and the eigenproblem is through two user supplied subroutines. In general, the structure of the matrices, **K** and **M**, vary greatly from problem to problem. No single subroutine can be designed to take advantage of the special structure of **K** and **M** for all problems. Furthermore, the charcteristics of a given equation solver depend strongly on the computing enviroment. Only the user is aware of the properties of his matrices and the computer system that is being used. Therfore, the job of solving the linear system of equations $\mathbf{K}_\sigma \overline{\mathbf{r}}_j = \mathbf{p}_j$ and multiplication of a vector by the mass matrix, $\overline{\mathbf{p}}_j = \mathbf{M}\overline{\mathbf{r}}_j$ is relegated to the user. This has the added advantage that a shifting strategy may be implemented without modifying any part of the routines presented here. The two routines that connect **K** and **M** with our program must take the form

1) SUBROUTINE OPK (X , Y , N). This solves the linear system of equations $\mathbf{K}_\sigma \mathbf{x} = \mathbf{y}$. N is the length of the vectors X and Y.

2) SUBROUTINE OPM (X , Y , N). This forms the matrix-vector product, $\mathbf{x} = \mathbf{M}\mathbf{y}$. N is the length of the vectors X and Y.

A third subroutine must also be provided for the management of the Lanczos vectors. From time to time the program calls the subroutine STORE. The subroutine statement for STORE is

SUBROUTINE STORE (V , N , J , ISW)

When ISW = 1 the routine must store the column vector V of length N and identifier J in the available storage. When ISW = 2 a vector V with identifier J that is in secondary store must be fetched. This way the user can take full advantage of any data management system that is available.

LANSEL also makes use of the LINPACK subroutines [1] called the BLAs (Basic Linear Algebra). These are routines for performing basic vector operations such as additions of two vectors. In many computer center there are carefully written assembly language implementations of these subroutines wich are much faster. We give a list of BLAs that are used from LINPACK in table 2 below.

BLA	Description
DATX	Computes the product of a scalar, a, by a vector, $\mathbf{x}$; $\mathbf{y} := a\mathbf{x}$.
DAXPY	Computes the product of a scalar, a, by a vector, $\mathbf{x}$, adding the result to a vector, $\mathbf{y}$; $\mathbf{y} := a\mathbf{x} + \mathbf{y}$.
DSCAL	Scales the elements of a vector, $\mathbf{y}$, by a scalar factor, a; $\mathbf{y} := a\mathbf{y}$.
DZERO	Resets the elements of a vector, $\mathbf{y}$, to zero ; $\mathbf{y} := 0$.
DDOT	Computes the Eucleadian inner product of two vectors, $\mathbf{x}$ and $\mathbf{y}$; $'dot' := \mathbf{x}^T\mathbf{y}$.
IDAMAX	Finds the index of the element of a vector with maximum absolute value; $i := arg(\max_{i=1,n} \mid y_i \mid)$.

Table 2. List of LINPACK subroutines used by LANSEL.

Subroutine LANDRV

LANDRV is the driver for the main subroutine LANSEL. Table 3 bellow gives a brief description of the list of parameters for LANDRV.

Name	Type	Dimension	Description
N	I	scalar	The dimension of the eigenproblem. **K** and **M** are N×N matrices.
LANMAX	I	scalar	Upper limit to the number of Lanczos steps. LANMAX must not exceed N.
MAXPRS	I	scalar	Upper limit to the number of wanted eigenpairs. MAXPRS must not exceed LANMAX.
ENDL	F	scalar	Left end of the interval containing the wanted eigenvalues.
ENDR	F	scalar	Right end of the interval containing the wanted eigenvalues.
NW	I	scalar	Length of the work array W. NW must be at least 6×N + 2×MAXPRS + 8×LANMAX. A larger NW can result in faster computation time. NW need not be greater than 6×N + 2×MAXPRS + LANMAX×(6 + LANMAX).
W	F	NW	Work array of length NW. The first N words of W hold a user supplied starting vector.
IW	I	MAXPRS	Work array of length MAXPRS.
EIG	F	MAXPRS	EIG will hold the converged Ritz values on return. The Ritz values are in the order they were computed.
Y	F	MAXPRS×N	Y will hold the converged Ritz vectors on return. Y is dimensioned as a two dimensional array Y(N,MAXPRS) and the I-th column of Y holds the Ritz vector corresponding to EIG(I).
NEIG	I	scalar	The number of computed eigenpairs.
IERR	I	scalar	IERR is an error flag. A succesful exectution is indicated when IERR = 0. For a list of error flags see table 4.

Table 3. Description of the parameters for LANDRV.

LANDRV acts as an interface between the user and LANSEL. It performs the following tasks before calling LANSEL:

(i) Checks the control parameters for possible error.

(ii) Allocates the working memeory for LANSEL.

(iii) If user does not supply a starting vector in W then LANDRV supplies a random vector.

(iv) Performs the first step of the Lanczos algorithm by calling STPONE.

If an error is detected IERR is reset and retured as soon as the checking of the control parameters is complete. Each bit of the integer IERR is used as an error flag. IERR is set to zero when enetring LANDRV. The I-th bit of IERR can then be reset to 1 using the the command IERR = IERR + 2**I. For example the third bit is set to 1 by adding 8 to IERR. Table 4 gives a description of possible errors indicated by IERR.

BIT	Indicates
1	$N < 0$
2	LANMAX < 0
3	ENDR $<$ ENDL
4	MAXPRS < 0
5	MAXPRS $>$ LANMAX
6	LANMAX $>$ N
7	NW too small

Table 4. List of errors indicated by IERR.

In addition, IERR = −1 indiactes that the maximum number of Lanczos steps in LANSEL has been reached.

```
      SUBROUTINE LANDRV(N,LANMAX,MAXPRS,ENDL,ENDR,EIG,Y,W,IW,NW,NEIG,
     1                    IERR)
C
C               ***************************************
C               *                                     *
C               *         LANCZOS ALORITHM WITH       *
C               *      SELECTIVE ORTHOGONALIZATION    *
C               *            L  A  N  S  O            *
C               *                                     *
C               ***************************************
C
C.... INPUTS
C.... N        DIMENSION OF THE EIGENPROBLEM
C.... LANMAX   UPPER LIMIT TO THE NUMBER OF LANCZOS STEPS
C.... MAXPRS   UPPER LIMIT TO THE NUMBER OF WANTED EIGENPAIRS
C.... ENDL     LEFT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C.... ENDR     RIGHT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C.... NW       LENGTH OF THE WORK ARRAY W
C.... W        WORK ARRAY OF LENGTH NW
C
C.... OUTPUTS
C.... IW       WORK ARRAY OF LENGTH MAXPRS
C.... EIG      ARRAY OF LENGTH MAXPRS TO HOLD THE CONVERGED RITZ VALUES
C.... Y        ARRAY OF LENGTH MAXPRS*N TO HOLD THE CONVERGED RITZ VECTORS
C.... IERR     ERROR FLAG
C
C.... SUBROUTINES: DDOT,STPONE,LANSEL,OPM,RANDOM
C
C.... COMMON RDATA
C.... RNM      NORM OF THE RISIDUAL VECTOR IN R(1)
C.... RNM2     SQUARE OF RNM
C.... SPREAD   WIDTH OF THE INTERVAL CONTAINING THE EIGENVALUES
C.... TOL      TOLARANCE FOR CONVERGENCE OF THE EIGENVALUES
C.... EPS      COMPUTER PRECISION
C.... EPS1     EPS*SQRT(N)
C
C.... COMMON IDATA
C.... EIGL     INNER MOST EIGENVALUE CONVERGED FROM LEFT END OF SPECTRUM
C.... EIGR     INNER MOST EIGENVALUE CONVERGED FROM RIGHT END OF SPECTRUM
C.... NEIG     TOTAL NUMBER OF CONVERGED EIGENVALUES
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TOL,EPS,EPS1
      COMMON /IDATA/EIGL,EIGR,NEIG
      DIMENSION NQ(5),Y(1),EIG(1),W(1),IW(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... CHECK INPUT DATA
C
      IERR = 0
      MT = 6*N + 2*MAXPRS + 8*LANMAX
      IF ( N .LE. 0 )              IERR = IERR + 1
      IF ( LANMAX .LE. 0 )         IERR = IERR + 2
      IF ( ENDR .LE. ENDL )        IERR = IERR + 4
      IF ( MAXPRS .LE. 0 )         IERR = IERR + 8
      IF ( MAXPRS .GT. LANMAX )    IERR = IERR + 16
      IF ( LANMAX .GT. N )         IERR = IERR + 32
      IF ( MT .GT. NW )            IERR = IERR + 64
      IF (IERR .GT. 0 ) RETURN
C
C.... COMPUTE THE MACHINE PRECISION
      EPS = ONE
      DO 10 I = 1,64
         IF (ONE + EPS .GT. ONE) EPS = EPS*0.5
10    CONTINUE
      EPS = EPS + EPS
C
C.... SET POINTERS
      M1 = MAXPRS + 6*N
      M2 = MAXPRS + M1
      M3 = MAXPRS + M2
      M4 = LANMAX + M3
      M5 = LANMAX + M4
      M6 = LANMAX + M5
      M7 = LANMAX + M6
      M8 = LANMAX + M7
      M9 = LANMAX + M8
      NS = NW - MT + 2*LANMAX
      NQ(1) = N + 1
      DO 20 I = 2,5
         NQ(I) = NQ(I-1) + N
20    CONTINUE
C
C.... CHECK FOR STARTING VECTOR
      RNM2 = ZERO
      DO 30 I = 1,N
         RNM2 = RNM2 + DABS(W(I))
30    CONTINUE
      IF (RNM2 .EQ. ZERO) THEN
C
C....    GET RANDOM VECTOR
         IRAND = N + LANMAX + MAXPRS + NW
         DO 40 I = 1,N
            W(I) = RANDOM(IRAND+I)
40       CONTINUE
      END IF
      CALL OPM(W,W(NQ(3)),N)
      RNM2 = DDOT(N,W,1,W(NQ(3)),1)
      CALL STPONE(N,W(M3),W(M4),W(M5),W(M6),W,NQ)
      CALL LANSEL(N,LANMAX,MAXPRS,NS,ENDL,ENDR,W,W(M3),W(M4),W(M5),
     1            W(M6),EIG,W(M1),W(M2),W(M7),W(M8),IW,Y,W(M9),NQ,IERR)
      RETURN
      END
```

Subroutine STPONE

This routine performs the initialization of ALF, BET, ALPH and BET2 as well as the first step of the Lanczos algorithm. See section 5 for notation. Since $\mathbf{q}_{j-1} = 0$ at the first step the algorithm reduces to an orthogonalization of $\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{r}$ against $\mathbf{q}_1$. $\mathbf{q}_1$ is obtained by normalizing $\mathbf{r}$ and is stored in the array R starting at location NQ(1). $\mathbf{M}\mathbf{r}$, stored in starting loaction NQ(3) of R is then normalized to get $\mathbf{M}\mathbf{q}_1$ which is put in R starting form NQ(4). NQ(2) and NQ(5) are the loaction of $\mathbf{q}_{j-1}$ and a temporary vector in R respectively.

```
      SUBROUTINE STPONE(N,ALF,BET,ALPH,BET2,R,NQ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TOL,EPS,EPS1
      DIMENSION NQ(1),R(1)
      DATA ONE,ZERO/0.0D0,0.0D0/
C
C.... THIS ROUTINE PERFORMS THE FIRST STEP OF THE LANCZOS ALGORITHM.
C.... N       DIMENTION OF THE EIGENPROBLEM
C.... ALF     THE NEW DIAGONAL ELEMENTS OF T
C.... BET     THE NEW OFF-DIAGONAL ELEMENTS OF T
C.... ALPH    THE NEW DIAGONAL ELEMENTS OF THE DEFLATED T. SEE ANALZT.
C.... BET2    THE NEW OFF-DIAGONAL SQUARED ELEMENTS OF THE DEFLATED T
C.... R       AN ARRAY CONTAINING [R(J),Q(J),Q(J-1),P(J),MR(J)]
C.... NQ(5)   LOCATION POINTERS FOR THE ARRAY R
C
C.... SUBROUTINES : DATX,DAXPY,DDOT,OPM,OPK
C
      EPS1 = EPS*DSQRT(DFLOAT(N))
      RNM = DSQRT(RNM2)
      BET2 = ZERO
      BET = RNM
      T = ONE/RNM
      CALL DATX(N,T,R,1,R(NQ(1)),1)
      CALL DATX(N,T,R(NQ(3)),1,R(NQ(4)),1)
      CALL OPK(R(NQ(4)),R,N)
      ALF = DDOT(N,R,1,R(NQ(4)),1)
      ALPH = ALF
      T = - ALF
      CALL DAXPY(N,T,R(NQ(1)),1,R,1)
      CALL OPM(R,R(NQ(3)),N)
      RNM2 = DDOT(N,R,1,R(NQ(3)),1)
      RETURN
      END
```

Subroutine LANSEL

LANSEL, LANczos algorithm with SELective orthogonalization, performs the main steps of the procedure described in the previous sections. The flowchart of figure 2 gives a global structure of LANSEL. After some initalizations the algorithm checks if any orthogonalization is necessary by calling PURGE. Then it calls LANSIM to perform a step of simple Lanczos. The orthogonality estimates and the eigenvalues of the tridiagonal matrix are updated by calling ORTBND and ANALZT respectively. When enough eigenvalues have been computed loop 10 is terminated and finally the wanted eigenvectors are computed using RITVEC.

LANSEL terminates the iteration if any of the following conditions are satisfied:

(a) The norm of $\mathbf{r}_j$ becomes small.

(b) The number of Lanczos steps reaches LANMAX, the maximum allowed. In this case IERR is set to -1.

(c) The number of computed eigenvalues exceeds the maximum number requested, MAXPRS.

(d) An eigenvalue is found outside the interval [ENDL, ENDR] at each end.

If [ENDL, ENDR] is empty then LANSEL will continue until an eigenvalue has converged outside each end of [ENDL, ENDR]. If the first MAXPRS eigenvalues are wanted, whatever they may be then set ENDL $= -\infty$, and ENDR $= +\infty$.

Structure of R: The first vector starting at location 1 is $\mathbf{r}_j$ when entering the subroutine. The remaining 5 vectors are stored at locations NQ(1) through NQ(5). This eliminates the need for moving the content of one array to another for swaping. Only the pointers of the arrays are changed. The vectors $\mathbf{q}_j$, $\mathbf{q}_{j-1}$, $\mathbf{Mr}_j$ and $\mathbf{Mq}_j$ are held in R starting from locations NQ(1) through NQ(4) respectively. The space in R starting form NQ(5) is used for a working vector in other parts of the program.

```
      SUBROUTINE LANSEL(N,LANMAX,MAXPRS,NS,ENDL,ENDR,R,ALF,BET,ALPH,
     1                  BET2,EIG,TAU,OLDTAU,ETA,OLDETA,INFO,Y,S,NQ,
     2                  NEIG,IERR)
C
C.... INPUTS
C.... N        DIMENSION OF THE EIGENPROBLEM
C.... LANMAX   UPPER LIMIT TO THE NUMBER OF LANCZOS STEPS
C.... MAXPRS   UPPER LIMIT TO THE NUMBER OF WANTED EIGENPAIRS
C.... NS       LENGTH OF THE ARRAY S
C.... ENDL     LEFT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C.... ENDR     RIGHT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C
C.... WORK SPACE
C.... R        HOLDS 6 VECTORS OF LENGTH N. SEE ABOVE FOR DETAILS.
C.... NQ(5)    CONTAINS THE POINTERS TO THE BEGINING OF EACH VECTOR IN R.
C.... ALF      ARRAY OF LENGTH LANMAX TO HOLD DIAGONAL OF THE TRIDIAGONAL T
C.... BET      ARRAY OF LENGTH LANMAX TO HOLD OFF-DIAGONAL OF T
C.... ALPH     DIAGONAL OF THE DEFLATED TRIDIAGONAL
C.... BET2     SQUARE OF THE OFF-DIAGONALS OF THE DEFLATED TRIDIAGONAL
C.... TAU      ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J
C.... OLDTAU   ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J-1
C.... ETA      ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J
C.... OLDETA   ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J-1
C.... INFO     INFORMATION ABOUT EIGENVECTORS OF T
C.... S        ARRAY FOR THE EIGENVECTORS OF THE TRIDIAGONAL
C
C.... OUTPUTS
C.... EIG      ARRAY OF LENGTH MAXPRS TO HOLD THE CONVERGED RITZ VALUES
C.... Y        ARRAY OF LENGTH MAXPRS*N TO HOLD THE CONVERGED RITZ VECTORS
C.... NEIG     NUMBER OF COMPUTED EIGENPAIRS
C.... IERR     ERROR FLAG
C
C.... SUBROUTINES: PURGE,LANSIM,ORTHBND,ANALZT,RITVEC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TOL,EPS,EPS1
      COMMON /IDATA/EIGL,EIGR,NEIG
      DIMENSION R(1),Y(N,1),EIG(1),TAU(1),OLDTAU(1),ALF(1),BET(1),S(1)
      DIMENSION NQ(1),ALPH(1),BET2(1),ETA(1),OLDETA(1),INFO(1)
      LOGICAL ENOUGH
      REPS = DSQRT(EPS)
      JJ = 1
      ETA(1) = EPS1
      EIGL = ENDL - (ENDR - ENDL)
      EIGR = ENDR + (ENDR - ENDL)
      NEIG = 0
C
C.... LANCZOS LOOP
C
      DO 10 J = 2,LANMAX
         NBUF = NS/J
         JJ   = JJ + 1
         RNM  = DSQRT(RNM2)
C
C....    RESTORE THE SEMI-ORTHOGONALITY STATE WHEN NEEDED
         CALL PURGE(R,R(NQ(1)),R(NQ(3)),R(NQ(4)),R(NQ(5)),Y,ALF,BET,S,
     1              EIG,ETA,OLDETA,TAU,OLDTAU,INFO,N,J-1,NBUF)
C
         IF (RNM .LT. REPS*SPREAD) GO TO 20
C
C....    TAKE A LANCZOS STEP
         CALL LANSIM(R,ALF(J),BET(J),ALPH(JJ),BET2(JJ),RNM,RNM2,NQ,
     1               N,J)
C
C....    UPDATE THE ORTHOGONALITY BOUNDS
         CALL ORTBND(ALF,BET,J,EPS1,ETA,OLDETA,TAU,OLDTAU,EIG,INFO,
     1               RNM,NEIG,N)
C
C....    UPDATE THE RITZ VALUES
         ENOUGH = NEIG .GE. MAXPRS .OR.
     1            ( EIGL .GT. ENDL .AND. EIGR .LT. ENDR)
         IF ( ENOUGH ) THEN
```

```
            GO TO 20
         ELSE
            CALL ANALZT(JJ,ALPH,BET2,EIG,TAU,OLDTAU,INFO)
         END IF
C
10    CONTINUE
      IERR = -1
C
C.... COMPUTE THE REMAINING RITZ VECTORS
20    DO 40 I=1,NEIG
         M = 1
         DO 30 K = 1,NEIG
            M =  MIN0(IABS(INFO(K)),M)
30       CONTINUE
         IF ( M .EQ. 0 ) THEN
            CALL RITVEC(R,R(NQ(1)),R(NQ(3)),R(NQ(4)),R(NQ(5)),Y,ALF,
     1                  BET,EIG,S,INFO,N,J,NEIG,NBUF,.TRUE.)
         ELSE
            GO TO 50
         END IF
40    CONTINUE
50    CONTINUE
      RETURN
      END
```

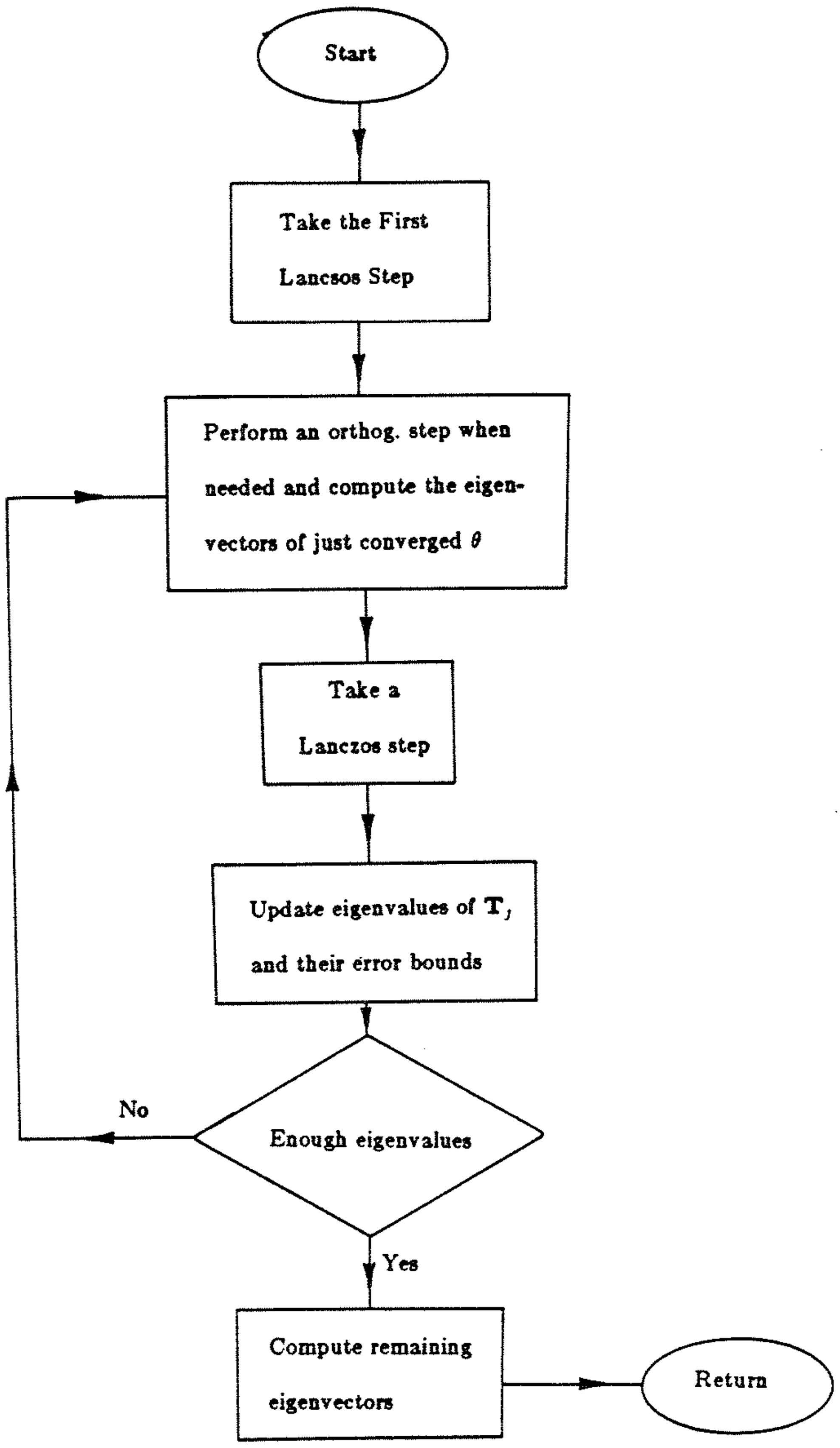
Start
Take the First Lancsos Step
Perform an orthog. step when needed and compute the eigen-vectors of just converged θ
Take a Lanczos step
Update eigenvalues of $\mathbf{T}_j$ and their error bounds
Enough eigenvalues
No
Yes
Compute remaining eigenvectors
Return

Figure 2. A global picture of the Lanczos method with selective orthogonalization indicating the way the individual modules are used.

Subroutine LANSIM

This routine performs a single step of the simple Lanczos process. The flowchart in figure 3 gives a global view of the algorithm. All the vectors required in this routine are stored in array R.

```
      SUBROUTINE LANSIM(R,ALF,BET,ALPH,BET2,RNM,RNM2,NQ,N,J)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION NQ(1),R(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... THIS ROUTINE PERFORMS A SINGLE STEP OF THE LANCZOS ALGORITHM.
C.... IT PERFORMS A STEP OF LOCAL REORTHOGONALIZATION IF NEEDED.
C.... SEE LANSEL FOR DEFINITIONS OF THE PARAMETERS.
C.... J        CURRENT LANCZOS STEP
C
C.... SUBROUTINES : DATX,DAXPY,DDOT,OPM,OPK,STORE
C
C.... SWAP Q(J) AND Q(J-1)
      NTMP    = NQ(2)
      NQ(2)   = NQ(1)
      NQ(1)   = NTMP
C
C.... Q = R/BETA
      T = ONE/RNM
      CALL DATX(N,T,R,1,R(NQ(1)),1)
C
C.... P = PBAR/BETA
      CALL DATX(N,T,R(NQ(3)),1,R(NQ(4)),1)
C
C.... R = ( K INVERSE )*Q
      CALL OPK(R(NQ(4)),R,N)
C
C.... R = R - Q(J-1)*BETA
      T = -RNM
      CALL DAXPY(N,T,R(NQ(2)),1,R,1)
C
C.... STORE Q(J-1)
      CALL STORE(R(NQ(2)),N,J-1,1)
      BET = RNM
      BET2 = RNM2
      ALF = ZERO
C
C.... ALF = ( R TRANSPOSE ) * P
      DALF = DDOT(N,R,1,R(NQ(4)),1)
C
C.... LOACL REORTHOGONALIZATION WHEN WARRANTED
      DO 10 I=1,2
         CALL DAXPY(N,-DALF,R(NQ(1)),1,R,1)
         ALF = ALF + DALF
         CALL OPM(R,R(NQ(3)),N)
         RNM2 = DDOT(N,R,1,R(NQ(3)),1)
         ALPH = ALF
         IF (RNM2*256.0D0 .GT. (ALF**2 + BET2) .OR. I .EQ. 2) RETURN
         DALF = DDOT(N,R(NQ(1)),1,R(NQ(3)),1)
         DBET = DDOT(N,R(NQ(2)),1,R(NQ(3)),1)
         CALL DAXPY(N,-DBET,R(NQ(2)),1,R,1)
         BET = BET + DBET
         BET2 = BET**2
10    CONTINUE
      END
```

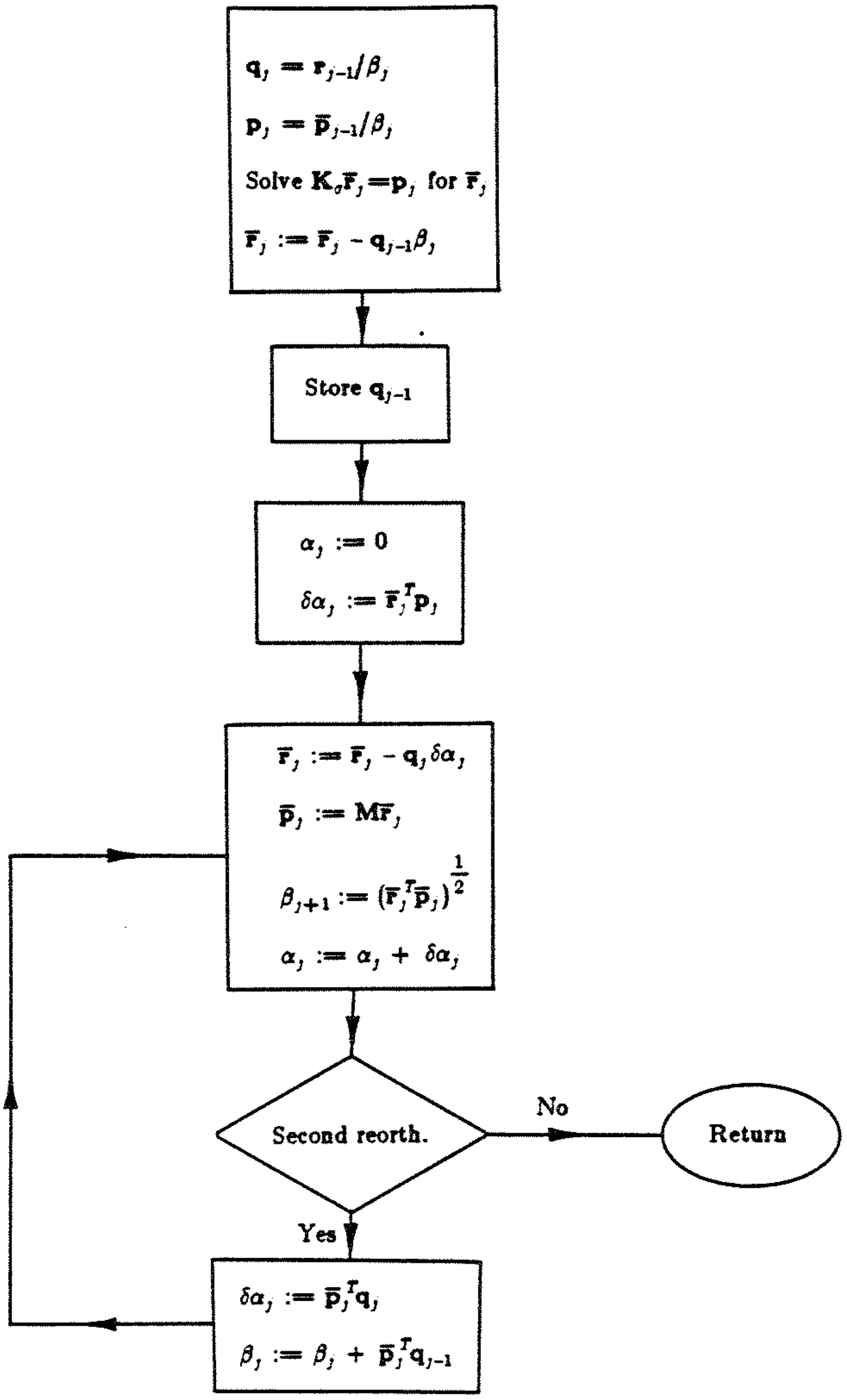
$q_j = r_{j-1}/\beta_j$
$p_j = \bar{p}_{j-1}/\beta_j$
Solve $K_\sigma \bar{r}_j = p_j$ for $\bar{r}_j$
$\bar{r}_j := \bar{r}_j - q_{j-1}\beta_j$
Store q_{j-1}
$\alpha_j := 0$
$\delta\alpha_j := \bar{r}_j^T p_j$
$\bar{r}_j := \bar{r}_j - q_j\delta\alpha_j$
$\bar{p}_j := M\bar{r}_j$
$\beta_{j+1} := (\bar{r}_j^T \bar{p}_j)^{\frac{1}{2}}$
$\alpha_j := \alpha_j + \delta\alpha_j$
Second reorth.
No
Return
Yes
$\delta\alpha_j := \bar{p}_j^T q_j$
$\beta_j := \beta_j + \bar{p}_j^T q_{j-1}$

Figure 3. Flowchart description of the j-th step of the simple Lanczos algorithm.

Subroutine ORTBND

The subroutine ORTBND updates the orthogonality bounds, $\mathbf{w}_j$, of Eq. 35, and the τ recurance of Eq. 40 in section 7. These quantities are later examined in subroutine PURGE for possible loss of semi-orthogonality. LANSEL is the only routine that calls ORTBND.

```
      SUBROUTINE ORTBND (ALF,BET,J,EPS1,ETA,OLDETA,TAU,OLDTAU,EIG,INFO,
     1                   RNM,NEIG,N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ALF(1),BET(1),ETA(1),OLDETA(1),TAU(1),OLDTAU(1),
     1          EIG(1),INFO(1)
      DATA ONE/1.0D0/
C
C.... INPUTS
C.... FOR [ALF,BET,ETA,OLDETA,TAU,OLDTAU,N] SEE LANSEL.
C.... J          ORDER OF T
C.... EPS1       ESTIMATED DOT PRODUCT AFTER ORTHOG. = SQRT(N)*EPS
C.... EIG        ARRAY OF CONVERGED EIGENVALUES
C.... INFO       INFORMATION ARRAY FOR EIGENVECTORS OF T. SEE SUBTJ.
C.... RNM        NORM OF THE NEXT RESIDUAL VECTOR
C.... NEIG       NUMBER OF CONVERGED EIGENVALUES
C
C.... UPDATE THE OMEGA AND TAU RECURANCES.
C
      IF ( J .GT. 1 ) THEN
         OLDETA(1) = ( BET(2)*ETA(2) + (ALF(1) - ALF(J))*ETA(1)
     1               - BET(J)*OLDETA(1)) / RNM
         IF ( J .GT. 2 )  THEN
            J1 = J - 1
            DO 100 K=2,J1
               OLDETA(K) = (BET(K+1)*ETA(K+1) + (ALF(K) - ALF(J))*ETA(K)
     1                   + BET(K)*ETA(K-1) - BET(J)*OLDETA(K)) / RNM
100         CONTINUE
         END IF
         DO 200 K=1,J1
              T = OLDETA(K)
              OLDETA(K) = ETA(K)
              ETA(K) = T
200      CONTINUE
      END IF
      ETA(J) = EPS1*BET(2)/RNM
C
C     UPDATE THE TAU RECURANCE.
C
      DO 300 I=1,NEIG
         IF ( INFO(I) .NE. 0 ) THEN
            T = TAU(I)
            TAU(I) = (EIG(I) - ALF(J))*TAU(I) - BET(J)*OLDTAU(I)
            OLDTAU(I) = T
         END IF
300   CONTINUE
      RETURN
      END
```

Subroutine PURGE

PURGE examins first the array ETA, which holds the vector $\mathbf{h}_{j+1}$. See section 7. If the element of ETA with largest absolute value is less than REPS, $\sqrt{\epsilon}$, indicating no loss of semi-orthogonality, then PURGE does nothing and returns. Otherwise, the elements of TAU are examined to determine if loss of orthogonality might be due to the return of a previously banished Ritz vector. If of TAU(I) is greater than REPS in absolute value it indicates loss of semi-orthogonality against the Ritz vector with index I. The corresponding eigenvector of $\mathbf{T}_j$ is computed using GIVENS. ETA and OLDETA are orthogonalized against the computed eigenvector of $\mathbf{T}_j$.

The elements of ETA are examined for a second time. If ETA still holds an element with absolute value greater than REPS, then loss of orthogonality is also due to the convergence of a Ritz value. Then RITVEC is called and the contents of TAU, OLDTAU, ETA and OLDETA are all set to EPS1 $= \sqrt{n}\ \epsilon$. Otherwise, $\mathbf{q}_j$ and $\mathbf{r}_j$, held in Q and R, are orthogonalized against those Ritz vectors in columns of Y with TAU value greater than REPS in absolute value. These elements of TAU and OLDTAU are then set to EPS1.

All orthogonalizations of the Lanczos vectors are performed with respect to the inertial inner product and therfore we require two vectors, QA and RA that hold $\mathbf{Mq}_j$ and $\mathbf{Mr}_j$ respectively. At the end of PURGE, $\mathbf{Mq}_j$ and $\mathbf{Mr}_j$ are recomputed if $\mathbf{q}_j$ and $\mathbf{r}_j$ are modified.

```
      SUBROUTINE PURGE(R,Q,RA,QA,T,Y,ALF,BET,S,EIG,ETA,OLDETA,TAU,
     1                 OLDTAU,INFO,N,J,NBUF)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /IDATA/ EIGL,EIGR,NEIG
      COMMON /RDATA/ RNM,RNM2,SPREAD,TEST,EPS,EPS1
      DIMENSION R(1),Q(1),RA(1),QA(1),T(1),Y(N,1),ALF(1),BET(1),S(J,1),
     1          EIG(1),ETA(1),OLDETA(1),TAU(1),OLDTAU(1),INFO(1)
      LOGICAL ORTHO
C
C.... FOR [ALF,BET,EIG,ETA,OLDETA,TAU,OLDTAU,INFO,N,J] SEE LANSEL.
C.... R         THE RESIDUAL VECTOR TO BECOME THE NEXT LANCZOS VECTOR
C.... Q         THE CURRENT LANCZOS VECTOR
C.... RA        THE PRODUCT OF THE MASS MATRIX AND R
C.... QA        THE PRODUCT OF THE MASS MATRIX AND Q
C.... T         A TEMPORARY VECTOR TO HOLD THE PREVIOUS LANCZOS VECTORS
C.... Y         CONTAINES THE COMPUTED RITZVECTORS
C.... S         VECTOR FOR COMPUTING EIGENVECTORS OF T(J)
C.... NBUF      NUMBER OF VECTORS IN S
C
C.... SUBROUTINES  : DAXPY,DZERO,DDOT,IDAMAX,SUBTJ,GIVENS,OPM
C
C.... THIS ROUTINE EXAMINES ETA, OLDETA, TAU AND OLDTAU TO DECIDE
C.... WHICH FORM OF REORTHOGONALIZATION IF ANY SHOULD BE PERFORMED.
C
      ORTHO = .FALSE.
      REPS = DSQRT(EPS)
      TOL = REPS*SPREAD
      K = IDAMAX(J-2,ETA,1)
      IF (DABS(ETA(K)) .GT. REPS) THEN
         DO 10 I=1,NEIG
            IF (INFO(I) .NE. 0 .AND. DABS(TAU(I)) .GT. REPS) THEN
               CALL DZERO(J,S,1)
               CALL SUBTJ(EIG,INFO,TOL,I,N,NEIG,J,L,K,M)
               CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS,
     1                     S,RESID,RAYCOR)
               ZETA = - DDOT(J,ETA,1,S,1)
               CALL DAXPY(J,ZETA,S,1,ETA,1)
               ZETA = - DDOT(J,OLDETA,1,S,1)
               CALL DAXPY(J,ZETA,S,1,OLDETA,1)
            END IF
10       CONTINUE
C
         K = IDAMAX(J-2,ETA,1)
C
         IF (DABS(ETA(K)) .GT. REPS) THEN
C
C....         GRAM-SCHMIDT NEEDED
            ORTHO = .TRUE.
            CALL RITVEC(R,Q,RA,QA,T,Y,ALF,BET,EIG,S,INFO,N,J,NEIG,
     1                  NBUF,.FALSE.)
            DO 20 I=1,NEIG
               TAU(I) = EPS1
               OLDTAU(I) = EPS1
20          CONTINUE
            DO 30 I=1,J-1
               ETA(I) = EPS1
               OLDETA(I) = EPS1
30          CONTINUE
         ELSE
C
C....         REMOVE COMPONENTS OF A RITZ VECTOR
            ORTHO = .TRUE.
            DO 40 I=1,NEIG
               IF (DABS(TAU(I)) .GT. REPS) THEN
                  TAU(I) = EPS1
                  OLDTAU(I) = EPS1
                  ZETA = DDOT(N,RA,1,Y(1,I),1)
                  CALL DAXPY(N,-ZETA,Y(1,I),1,R,1)
                  ZETA = DDOT(N,QA,1,Y(1,I),1)
                  CALL DAXPY(N,-ZETA,Y(1,I),1,Q,1)
               END IF
40          CONTINUE
         END IF
      END IF
      IF ( ORTHO ) THEN
         CALL OPM(R,RA,N)
      END IF
      RETURN
      END
```

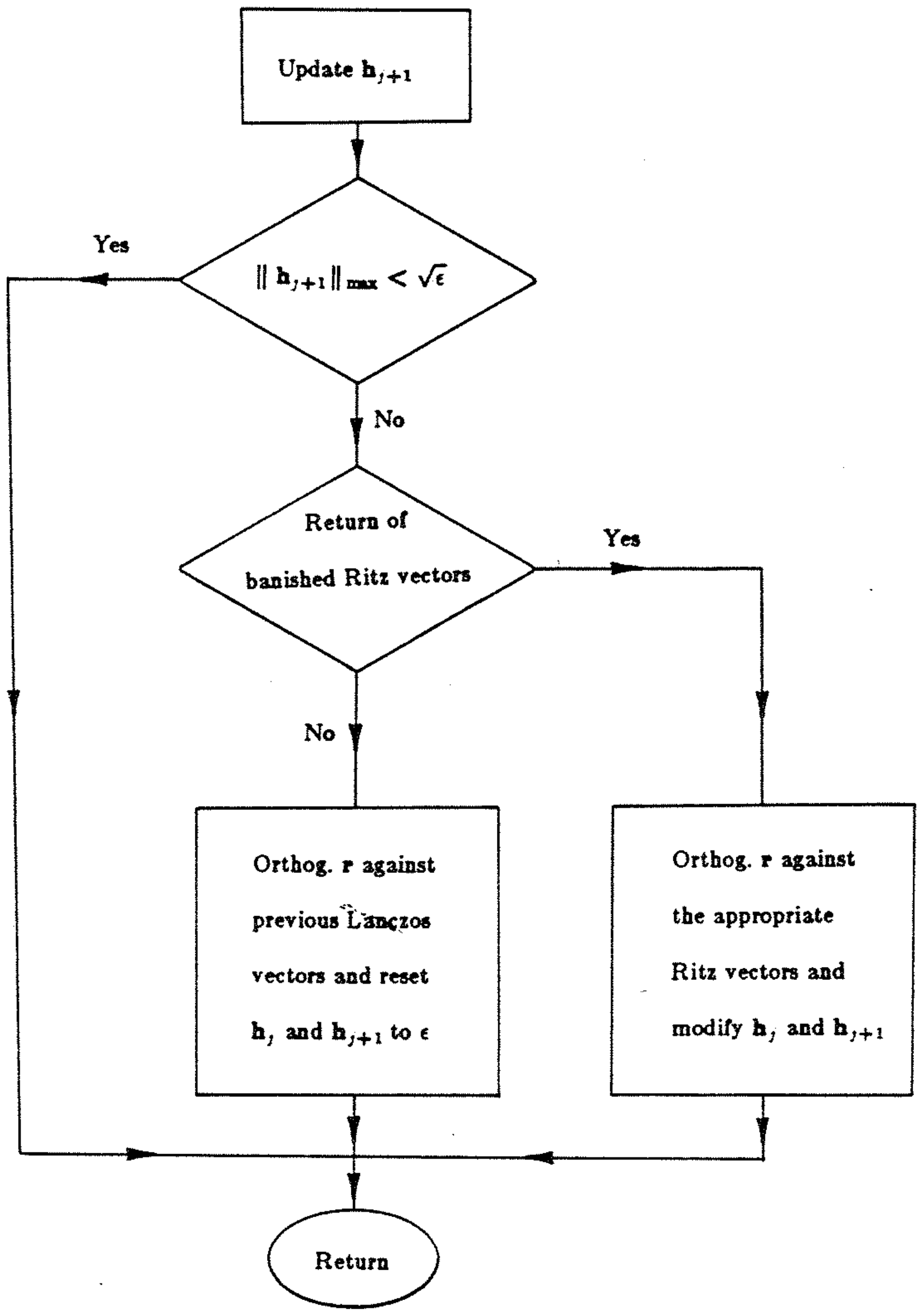
Update $\mathbf{h}_{j+1}$
$\| \mathbf{h}_{j+1} \|_{\max} < \sqrt{\epsilon}$
Yes
No
Return of banished Ritz vectors
Yes
No
Orthog. $\mathbf{r}$ against previous Lanczos vectors and reset $\mathbf{h}_j$ and $\mathbf{h}_{j+1}$ to ϵ
Orthog. $\mathbf{r}$ against the appropriate Ritz vectors and modify $\mathbf{h}_j$ and $\mathbf{h}_{j+1}$
Return

Figure 4. A flowchart for selective orthogonalization method.

Subroutine RITVEC

The purpose of this routine is two fold:

(1) compute the Ritz vector corresponding to a converged Ritz value

(2) perform a full reorthogonalization step.

The first step of the program is to compute some eigenvectors of the tridiagonal $\mathbf{T}_j$. The eigenvectors of $\mathbf{T}_j$ are stored in an array S and the number of computed vectors depends on the available storage indicated by NBUF. GIVENS is used to obtain the wanted eigenvectors. The next step is to recall all the Lanczos vectors from secondary store (by calling STORE) and compute the Ritz vectors using Eq. 23, accumulating the result in columns of Y. The old Lanczos vector that is brought in is stored in the temporary vector T. In the same loop, when EVONLY is false, the two current Lanczos vectors, $\mathbf{q}_j$ and $\mathbf{r}_j$, held in Q and R, are orthogonalized against the previous Lanczos vectors held in T. In this way the number of I/O transfers is minimized. Arrays QA and RA that hold $\mathbf{Mq}_j$ and $\mathbf{Mr}_j$ respectively are also needed by RITVEC to perform the orthogonalization with respect to the inertial inner product.

```
      SUBROUTINE RITVEC(R,Q,RA,QA,T,Y,ALF,BET,EIG,S,INFO,N,J,NEIG,NBUF,
     1                  EVONLY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TEST,EPS,EPS1
      DIMENSION R(1),Q(1),RA(1),QA(1),T(1),Y(N,1),ALF(1),BET(1),S(J,1),
     1          EIG(1),INFO(1)
      LOGICAL EVONLY
C
C.... SUBROUTINES : DAXPY,DZERO,DDOT,IDAMAX,SUBTJ,GIVENS,STORE
C
C.... IF EVONLY = .TRUE. COMPUTES ONLY RITZVECTORS (NO REORTHOG.)
C.... THIS ROUTINE COMPUTES SOME RITZVECTORS AND PERFORMS A
C.... REORTHOGONALIZATION OF THE LANCZOS VECTORS.
C.... FOR [R,Q,RA,QA,T,Y,ALF,BET,EIG,S,INFO,N,J,NEIG,NBUF] SEE
C.... PURGE AND LANSEL.
C
      IBUF = 0
      TOL = DSQRT(EPS)*SPREAD
      RNEPS = RNM*EPS
C
C.... COMPUTE EIGENVECTORS OF T AND PUT IN THE BUFFER.
C
      DO 30 I=NEIG,1,-1
         IF (INFO(I) .GE. 0 .AND. IBUF .LT. NBUF) THEN
            IF (EVONLY .AND. INFO(I) .NE. 0) GO TO 30
            IBUF = IBUF + 1
            CALL SUBTJ(EIG,INFO,TOL,I,N,NEIG,J,L,K,M)
            CALL DZERO(N,Y(1,I),1)
            DO 10 IDUM=1,J
               CALL DZERO(J,S(1,IBUF),1)
               CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS1,
     1                     S(M,IBUF),RESID,RAYCOR)
               IF (RESID .LE. TOL) GO TO 20
```

```
              K = K + 1
              IF (K .GT. J) K = 1
10          CONTINUE
20          EIG(I) = EIG(I) + RAYCOR
            INFO(I) = N*IDAMAX(J,S(1,IBUF),1) + J - 1
         END IF
30    CONTINUE
C
C....  COMPUTE THE RITZ VECTORS AND PERFORM G-S ORTHOGONALIZATION.
C
      DO 50 I=1,J-1
         CALL STORE(T,N,I,2)
         KBUF = 0
         DO 40 K=NEIG,1,-1
            IF (INFO(K) .GE. 0 .AND. KBUF .LT. IBUF) THEN
               KBUF = KBUF + 1
               SI = S(I,KBUF)
               IF (DABS(SI) .GT. EPS) CALL DAXPY(N,SI,T,1,Y(1,K),1)
            END IF
40       CONTINUE
         IF (.NOT.EVONLY) THEN
            ZETOLD = -DDOT(N,QA,1,T,1)
            IF ( DABS(ZETOLD) .GT. EPS ) CALL DAXPY(N,ZETOLD,T,1,Q,1)
            ZETA = -DDOT(N,RA,1,T,1)
            IF ( DABS(ZETA) .GT. RNEPS ) CALL DAXPY(N,ZETA,T,1,R,1)
         END IF
50    CONTINUE
      KBUF = 0
      DO 60 I=NEIG,1,-1
         IF (INFO(I) .GE. 0 .AND. KBUF .LT. IBUF) THEN
            KBUF = KBUF + 1
            IF (DABS(S(J,KBUF)) .LT. EPS1) INFO(I) = -INFO(I)
            CALL DAXPY(N,S(J,KBUF),Q,1,Y(1,I),1)
         END IF
60    CONTINUE
      RETURN
      END
```

Subroutine GIVENS

This routine computes the eigenvector of a tridiagonal with corresponding to an eigenvalue stored in THET. It first assumes a value for the bottom element of the eigenvector and solves for the rest using the Givens recurrance [19] going backward. This phase is terminated once the K-th element of S has been obtained. Then all the terms computed thus far are scaled to make S(K) unity. A similar procedure is performed starting form the top element and running the recurrance in the forward direction. Then the computed vector is normalized and finally the residual and the Rayliegh correction to THETA are computed and returned in RES and COR respectively. The algorithm scales of S to avoid overflow whenever it becomes necessary.

```
      SUBROUTINE GIVENS(K,J,ALF,BET,THET,EPS,S,RES,COR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ALF(1),BET(1),S(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... GIVENS RECURANCE FOR COMPUTING EIGENVECTORS OF A TRIDIAGONAL T.
C
C.... INPUTS
C.... FOR [ALF,BET,EPS] SEE LANSEL.
C.... K        INDEX OF THE RIGHT HAND SIGE E(K)
C.... THET     EIGENVALUE OF T
C
C.... OUTPUTS
C.... S(J)     COMPUTED EIGENVECTOR
C.... RES      NORM OF THE RESIDUAL
C.... COR      RAYLEIGH CORRECTION FOR THET
C
C.... SUBROUTINES : DSCAL
C
C.... BACKWARD RECURRENCE
      EPS2 = ONE/(EPS**2)
      RES = ZERO
      S(J) = EPS
      S(J-1) = -(ALF(J) - THET)*EPS/BET(J)
      SUM2 = EPS**2 + S(J-1)**2
      DO 10 I = J-1,K+1,-1
         S(I-1) = -((ALF(I) - THET)*S(I) + BET(I+1)*S(I+1))/BET(I)
C
C....    SCALE TO AVOID OVERFLOW
         IF ( S(I-1) .GT. EPS2 ) THEN
            F = ONE/S(I-1)
            S(I-1) = ONE
            CALL DSCAL(J-I+1,F,S(I),1)
            SUM2 = (SUM2*F)*F
         END IF
         SUM2 = SUM2 + S(I-1)**2
10    CONTINUE
      F = ONE/S(K)
      S(K) = ONE
      SUM2 = SUM2*F**2
      CALL DSCAL(J-K,F,S(K+1),1)
      IF ( K .LE. 1 ) GO TO 30
C
C.... FORWARD RECURRENCE
         X = ZERO
         S(1) = EPS
         SUM1 = EPS**2
         IF ( K .GT. 2 ) THEN
            S(2) = -((ALF(1) - THET)*S(1))/BET(2)
            SUM1 = SUM1 + S(2)**2
            DO 20 I = 2,K-2
               S(I+1) = -((ALF(I) - THET)*S(I) + BET(I)*S(I-1))/BET(I+1)
               IF ( S(I+1) .GT. EPS2 ) THEN
                  F = ONE/S(I+1)
                  S(I+1) = ONE
                  CALL DSCAL(I,F,S,1)
                  SUM1 = (SUM1*F)*F
               END IF
               SUM1 = SUM1 + S(I+1)**2
20          CONTINUE
            X = BET(K-1)*S(K-2)
         END IF
         X = -(X + (ALF(K-1) - THET)*S(K-1))/BET(K)
C
C....    MATCH X WITH S(K)
         IF ( X .EQ. ZERO ) THEN
            RES = ONE
            RETURN
         END IF
         F = S(K)/X
         CALL DSCAL(K-1,F,S,1)
         SUM2 = SUM2 + SUM1*F**2
         RES = BET(K)*S(K-1)
C
C.... NORMALIZE S
30    F = ONE/DSQRT(SUM2)
      CALL DSCAL(J,F,S,1)
      RES = RES*F + (ALF(K) - THET)*S(K) + BET(K+1)*S(K+1)
      COR = S(K)*RES
      RES = DABS(RES)
      RETURN
      END
```

Subroutine SUBTJ

In theory unreduced tridiagonal matrices do not have equal eigenvalues but in practice they can have two eigenvalues that are so close to one another that they are indistinguishable in finite precision. This creates certain defficulties when computing eigenvectors with close eiegnvalues. The problem is solved by working with a submatrix $\mathbf{T}_{l,m}$ of the tridiagonal $\mathbf{T}_j$. This routine obtains an estimate of the indices l and m that define a submatrix for which EIG(I) is simple. INFO(I) = N×K + J stores the two indices J and K. J is the step at which the eigenvector corresponding to EIG(I) was computed and K is the index of the element of this eigenvector with largest absolute value. When INFO(I) is negative it indicates that the Ritz vector in Y(I) is converged.

```
      SUBROUTINE SUBTJ(EIG,INFO,TOL,I,N,NEIG,J,L,K,M)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION EIG(1),INFO(1)
C
C.... INPUTS
C.... TOL              TOLARANCE FOR FINDING COPIES OF EIG(I)
C.... I                INDEX OF THE EIGENVALUE CONSIDERED
C.... FOR [EIG,N] SEE LANSEL. FOR INFO SEE ABOVE.
C
C.... OUTPUTS
C.... K                INDEX OF THE RIGHT HAND SIDE FOR GIVENS
C.... L                INDEX OF THE LAST ELEMENT OF THE SUBMATRIX
C.... M                INDEX OF THE FIRST ELEMENT OF THE SUBMATRIX
C
C.... THIS ROUTINE SCANS BACK THROUGH THE CONVERGED EIGEN VALUES
C.... FOR COPIES OF EIG(I) TO DETERMINE THE SUBMATRIX T(M,L) AND
C.... THE RIGHT HAND SIDE E(K) FOR THE GIVENS RECURENCE.
C
      L = J
      EIGI = EIG(I)
      DO 100 I1 = I-1,1,-1
         IF (DABS(EIGI - EIG(I1)) .LT. TOL) GO TO 110
100   CONTINUE
110   IF (I1 .LT. 1) THEN
C....    EIG(I) IS DISTINCT
         M = 1
         K = 1
      ELSE
C....    EIG(I) HAS A REPEATED COPY
         M = 1 + IABS(INFO(I1))/N
         K = (3*MOD(IABS(INFO(I1)),N) + J)/4
      END IF
      IF ( INFO(I) .GT. 0 ) K = INFO(I)/N
      DO 200 I2 = I+1,NEIG
         IF (DABS(EIGI - EIG(I2)) .LT. TOL) GO TO 210
200   CONTINUE
210   IF (I2 .LE. NEIG) THEN
         L = IABS(INFO(I2))/N - 1
         IF (INFO(I2) .EQ. 0) L = MOD(IABS(INFO(I)),N)
      END IF
      RETURN
      END
```

Subroutine ANALZT

The goal of this routine is to obtain the smallest possible inerval that contains a single eigenvalue of $\mathbf{T}_j$. It makes full use of the corresponding information that was obtained at the previous step. When J = 2 it simply computes the eigenvalues of the 2×2 tridiagonal matrix. Phase I of the routine updates the data structure (THETA, BJ), that containes those eigenvalues of $\mathbf{T}_{j-1}$ that are about to converge to eigenvalues of the big problem. Their residual bounds go in BJ. Phase II of the routine appends some new items to this data structure. Any converged eigenvalue is removed form THETA and put into EIG and then explicitly deflated form the tridiagonal. This routine is called by LANSEL. For a more detailed description see [15].

```
      SUBROUTINE ANALZT(J,ALF,BET2,EIG,TAU,OLDTAU,INFO)
C
C.... THIS ROUTINE UPDATES SOME EIGENVALUES OF A TRIDIAGONAL T(J) USING
C.... THE EIGENVALUES OF T(J-1).
C
C     J          ORDER OF THE TRIDIAGONAL T.
C     ALF        DIAGONAL OF T.
C     BET2       SQUARES OF THE OFFDIAGONAL TERMS, BET2(1) = 0.0D0
C     THET       EXTERIOR EIGENVALUES OF T, NEARLY CONVERGED
C                RITZVALUES. THET(1)=LEFTMOST, THET(8)=RIGHTMOST.
C     BJ         ERROR BOUND ON THET
C                BJ(I) IS SET TO -1 IF THET(I) DISAPPEARS.
C     NBD        CONTAINES L AND R IN THE TEXT.
C     SPREAD     THET(8) - THET(1)
C     EPS        PRECISION OF ARITHMATIC OPERATIONS
C     IP         IP=1 FOR UPDATING LEFT END, IP=2 FOR THE RIGHTEND.
C     INC        INC=1 FOR UPDATING LEFT END, INC=-1 FOR THE RIGHTEND.
C     IS         STARTING INDEX (EITHER 1 OR 8)
C     NEWRTZ     FALSE UNLESS AN EXTRA RITZ VALUE HAS BEEN INSERTED.
C     START      LEFT BOUND ON EIGENVALUES (INC=1), RIGHT BOUND (INC=-1)
C     PROBE      THE OUTER END OF THE NEXT SUBINTERVAL TO BE UPDATED.
C     INDXOK     TRUE, IF THERE ARE I-INC RITZ VALUES EXTERIOR TO THE
C                NEW THET(I).
C.... FOR [EIG,TAU,OLDTAU,N] SEE LANSEL. FOR [INFO] SEE SUBTJ.
C
C     SUBROUTINES:NEWCOR,DEFLAT,MOVE1,NUMLES
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TOL,EPS,EPS1
      COMMON /ATDATA/THET(8),BJ(8),NBD(2)
      COMMON /IDATA/EIGL,EIGR,NEIG
      DIMENSION ALF(1),BET2(1),EIG(1),TAU(1),OLDTAU(1),INFO(1)
      LOGICAL  INSERT,INDXOK,APPEND
      DATA ONE / 1.0D0 /,ZERO / 0.0D0/
      IF ( J .LE. 1) RETURN
      IF ( J .EQ. 2) THEN
         THET(1) = (ALF(1) + ALF(2) - DSQRT(4.*BET2(2) +
     1             (ALF(1) - ALF(2))**2))/2.
         THET(8) = ALF(1) + ALF(2) - THET(1)
         BJ(1)   = DSQRT(RNM2/(ONE + BET2(2)/(THET(1) - ALF(1))**2))
         BJ(8)   = DSQRT(RNM2/(ONE + BET2(2)/(THET(8) - ALF(1))**2))
         NBD(1)  = 1
         NBD(2)  = 8
         SPREAD  = THET(8) - THET(1)
         REPS    = DSQRT(EPS)
         RETURN
      END IF
      TOL = 2.0*REPS*SPREAD
      W = 64.*TOL
C
C.... BEGIN PHASE 1.
C.... LOOP FOR LEFT END, THEN RIGHT
C
      DO 100 IP = 1,2
         INC = 3 - 2*IP
         IS = 7*IP - 6
         I = IS
         INSERT = .FALSE.
         START = (THET(I) + ALF(J) - INC*DSQRT(BET2(J)*4. +
     1            (ALF(J) - THET(I))**2))/2.
         PROBE = THET(I) - INC*BJ(I)
         INDXOK = NUMLES(ALF,BET2,PROBE,J,INC,EPS) .EQ. 0
         DO 50 IDUMMY =1,8
            IF (I - NBD(IP) .EQ. INC) GO TO 100
C
C....       EXAMINE I TH SUBINTERVAL
C
            IF (INDXOK) THEN
               IF (INSERT) THEN
                  START = THET(I)
                  THET(I) = START + INC*DMIN1(B**2/
     1                           DABS(START - THET(I-INC)) , B)
               ELSE
                  IF (INT(DSIGN(ONE,PROBE-START)) .EQ. INC) START=PROBE
               END IF
C
C....          CHECK FOR DISJOINT SUBINTERVALS
C
               IF (I .EQ. NBD(IP)) THEN
                  PROBE=THET(I)+INC*(THET(NBD(2))-THET(NBD(1)))/(4.0*J)
               ELSE
                  PROBE = THET(I+INC) - INC*BJ(I+INC)
               END IF
               IF (INT(DSIGN(ONE,PROBE - THET(I))) .EQ. INC) THEN
C
C....             CHECK FOR AN EXTRA RITZVALUE
C
                  K = NUMLES(ALF,BET2,PROBE,J,INC,EPS)
                  IF (K .LT. IABS(I - IS + INC)) THEN
C
C....                THET(I) DISAPPEARS
C
                     BJ(I) = -ONE
                  ELSE
C
C....                RECORD INDEXOK FOR NEXT LOOP. USE REFINED BOUNDS.
C
                     IF ( .NOT. INSERT) THEN
                        B = BJ(I)
                        INDXOK = (K .LE. IABS(I - IS + INC))
                        BND = DMIN1(B**2/DABS(PROBE-THET(I)) , B)
                        IF (INDXOK.AND.BND.LT.DABS(THET(I)-START)) THEN
                           START = THET(I) - INC*BND
                        END IF
                     END IF
                  END IF
               END IF
            ELSE
C
C....          PREPARE FOR AN INTRUDING RITZVALUE
C
               IF ((IS.EQ.NBD(IP) .OR. BJ(NBD(IP)-INC).LT.W) .AND.
     1             NBD(2)-NBD(1).GT.1 ) NBD(IP) = NBD(IP) + INC
               CALL MOVE1(THET,I,NBD(IP),-INC,PROBE)
               CALL MOVE1(BJ,I,NBD(IP),-INC,2.0*TOL)
               INSERT = .TRUE.
               INDXOK = .TRUE.
            END IF
C
```

```
             IF ( BJ(I) .GT. TOL ) THEN
C
C....        USE NEWTON ITERATION TO FIND NEW THET(I)
C
                CALL NEWCOR(ALF,BET2,START,THET,BJ,NBD,INC,I,J)
             END IF
C
             IF ( BJ(I) .LT. 0 ) THEN
C
C....           THET(I) DISAPPEARS
C
                CALL MOVE1(THET,NBD(IP),I,INC,ZERO)
                CALL MOVE1(BJ,NBD(IP),I,INC,ZERO)
                NBD(IP) = NBD(IP) - INC
                INSERT = .FALSE.
                INDXOK = .TRUE.
                I = I - INC
             END IF
             I = I + INC
50        CONTINUE
C
C....     END OF PHASE 1
C
100   CONTINUE
C
C.... BEGIN PHASE 2.
C.... APPEND MORE RITZVALUES AND CHECK FOR CONVERGED RITZVALUES.
C
      DO 200 IP = 1,2
         INC = 3 - 2*IP
         IS  = 7*IP - 6
         I = IS
         DO 150 IDUMMY = 1,J
            NREM = J - NBD(1) + NBD(2) - 9
            IF ( (I-NBD(IP))*INC .GT. 0 ) GO TO 200
            APPEND = I .EQ. NBD(IP) .AND. (BJ(I) .LT. W .OR. (J .EQ. 4
     1               .AND. NBD(IP) .EQ. IS)) .AND. NREM .GT. 0
            IF (APPEND) THEN
               START = THET(I)
               PROBE = INC*(THET(NBD(2)) - THET(NBD(1)))/NREM
            END IF
            IF (BJ(I) .LT. TOL) THEN
C
C....          INSERT THET(I) INTO EIG
C
               NEIG = NEIG + 1
               IF (IP .EQ. 1 ) THEN
                  EIGL  = DMAX1(EIGL,THET(I))
                  NEIGL = NEIGL + 1
               ELSE
                  EIGR  = DMIN1(EIGR,THET(I))
                  NEIGR = NEIGR + 1
               END IF
               EIG(NEIG)  = THET(I)
               INFO(NEIG) = 0
C
C....          REMOVING STABLIZED RITZ VALUES
C
               CALL DEFLAT(ALF,BET2,THET(I),RNM2,J)
               CALL MOVE1(THET,NBD(IP),I,INC,ZERO)
               CALL MOVE1(BJ,NBD(IP),I,INC,ZERO)
               NBD(IP) = NBD(IP) - INC
               I = I - INC
            END IF
            IF (APPEND .AND. NBD(2)-NBD(1) .GT. 1) THEN
               T = START + PROBE
               NBD(IP) = NBD(IP) + INC
               IK = IABS(IS - NBD(IP))
               DO 110 IDUM = 1,J
                  IF (NUMLES(ALF,BET2,T,J,INC,EPS) .NE. IK) GO TO 120
                  T = T + PROBE
110            CONTINUE
```

```
120             THET(NBD(IP)) = T
                START = T - PROBE
                NP = NBD(IP)
                CALL NEWCOR(ALF,BET2,START,THET,BJ,NBD,INC,NBD(IP),J)
              END IF
              IF (J.GT.8 .AND. I.EQ.NBD(IP) .AND. I.NE.IS .AND. BJ(I).GT.
     1            BJ(I-INC) .AND. BJ(I-INC).GT.W) NBD(IP) = NBD(IP) - INC
              I = I + INC
150       CONTINUE
200     CONTINUE
C
        DO 300 IP = 1,2
          INC = 3 - 2*IP
          IS  = 7*IP - 6
          IF (NBD(IP) .EQ. IS-INC) THEN
            THET(IS) = THET(NBD(3-IP))
            BJ(IS)   = BJ(NBD(3-IP))
            NBD(IP)  = IS
            NBD(3-IP)= NBD(3-IP) + INC
          END IF
300     CONTINUE
C
        SPREAD = THET(8) - THET(1)
        RETURN
        END
```

Subroutine NEWCOR

The object of this routine is to find an eigenvalue of $\mathbf{T}_j$ in a given interval. It uses a combination of bisection and Newton's method. For a detailed description see [15]. INDEX points to the position in THETA that will evantually hold the computed eigenvalue. On entry THETA(INDEX) contains the eigenvalue of $\mathbf{T}_{j-1}$ that is also one end of the interval and ZETA is the other. ZETA is also the starting for Newton's method. If this interval is too large then the algorithm performs a few steps of bisection to obtain a better estimate, ZETA, to the desired eigenvalue. The eigenvalue counts are obtained by calling NUMLES. Then Newton's iteration is used to compute the eigenvalue to the full computer precision. The Newton correction is computed using a recurrence that is given in [15]. The convergence of newton is guaranteed by deflating the eigenvalues exterior to ZETA implicitly. NEWCOR is called only by ANALZT.

```
      SUBROUTINE NEWCOR(ALF,BET2,ZETA,THET,BJ,NBD,INC,INDX,J)
C
C     COMPUTES EXTRERIOR EIGENVALUES OF A TRIDIAGONAL USING A
C     COMBINATION OF BISECTIONS AND NEWTON METHOD
C
C     ALF        DIAGONAL OF T.
C     BET2       SQUARES OF THE OFFDIAGONAL TERMS, BET2(1) = 0.0D0
C     ZETA       EXTERIOR BOUND FOR EIGENVALUE OF T IN THET(INDEX)
C     THET       EXTERIOR EIGENVALUES OF T, NEARLY CONVERGED
C                RITZVALUES. THET(1)=LEFTMOST, THET(8)=RIGHTMOST.
C     BJ         ERROR BOUND ON THET
C     NBD        CONTAINES LEFTMOST OR RIGHTMOST INDEX OF THETA.
C     SPREAD     THET(8) - THET(1)
C     INC        INC=1 FOR UPDATING LEFT END, INC=-1 FOR THE RIGHTEND.
C     INDX       INDEX OF THET TO BE UPDATED.
C     J          ORDER OF THE TRIDIAGONAL T.
C
C     SUBROUTINES: NUMLES,QLBOT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /RDATA/RNM,RNM2,SPREAD,TOL,EPS,EPS1
      DIMENSION ALF(1),BET2(1),THET(1),BJ(1),NBD(1)
      REPS = DSQRT(EPS)
      IF ( J .EQ. 1 ) THEN
         ZETA = THET(INDX)
         THET(INDX) = ALF(J)
         BJ(INDX) = DSQRT(RNM2)
         RETURN
      END IF
C
C.... PERFORM BISECTION FOR AN IMPROVED ZETA
C
      IS = (9 - 7*INC)/2
      FACT = 32.0*DFLOAT(J)*DLOG(DFLOAT(J))
      WIDTH = (THET(INDX) - ZETA)/2.
      IF ( WIDTH .EQ. 0.0D0 ) WIDTH = BJ(INDX)/2.
      IOLD = IABS(IS - INDX)
      DO 10 IDUMMY = 1,15
         IF (DABS(WIDTH)*FACT .LE. DABS(THET(9-IS)-THET(INDX))) GO TO 20
         ZNEW = ZETA + WIDTH
         INEW = NUMLES(ALF,BET2,ZNEW,J,INC,EPS)
         WIDTH = WIDTH/2.
```

```
            IF ( INEW .EQ. IOLD ) ZETA = ZNEW
10      CONTINUE
20      CONTINUE
C
        DO 50 IDUMMY = 1,30
           U = ALF(1) - ZETA
           TT = 0.1*EPS*BET2(2)
           IF ( U .EQ. 0.0D0 ) U = 0.1*EPS*BET2(2)
           RAT = 1.0D0/U
           SUM = RAT
           DO 30 I = 2,J
              H   = BET2(I)/U
              U = ALF(I) - ZETA - H
              IF ( U .EQ. 0.0D0 ) U = 0.1*EPS*(H + BET2(I))
              RAT = ( 1.0D0 + H*RAT )/U
              SUM = SUM + RAT
30         CONTINUE
           BOT = U*SUM
C
C....      DEFLATION
C
           DO 40 I = IS,INDX-INC,INC
              DEL = ZETA - THET(I)
              IF (DABS(DEL).LT.EPS*DABS(ZETA)) DEL = EPS*DABS(ZETA)
              SUM = SUM + 1.0D0/DEL
40         CONTINUE
C
C....      CHECK FOR CONVERGENCE
C
           ZETA = ZETA + 1.0D0/SUM
           IF (DABS(SUM*SPREAD*REPS) .GT. 1.0D0 ) GO TO 60
50      CONTINUE
        WRITE(6,2000)
60      CONTINUE
        CALL QLBOT(ALF,BET2,ZETA,BOT,J)
        BOT = DSQRT(RNM2*BOT)
C
        ZNEW = THET(INDX)
        THET(INDX) = ZETA
        ZETA = ZNEW
        BJ(INDX) = BOT
        RETURN
2000    FORMAT(' NO CONVERGENCE IN NEWTON AFTER 30 STEPS')
        END
```

Subroutine DEFLAT

The subroutine DEFLAT deflates tridiagonal $\mathbf{T}_j$ using an eigenvalue THETA. One step of the QR algorithm is used to perform the deflation. Each of the last few QR rotations will be almost a row and column interchange. For a more detail description of the algorithm see [15]. ANALZT is the only routine that calls DEFLAT.

```
      SUBROUTINE DEFLAT(ALF,BET2,THET,RNM2,J)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION ALF(1),BET2(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... INPUTS
C.... THET      EIGENVALUE OF T TO BE DEFLATED EXPLICITLY
C.... RNM2      SQUARE OF THE NORM OF THE RESIDUAL VECTOR
C.... J         DIMENSION OF THE TRIDIAGONAL MATRIX
C
C.... OUTPUTS
C.... ALF(J-1) MODIFIED DIAGONALS OF T
C.... BET(J-1) MODIFIED OFF-DIAGONALS OF T
C
C.... THE ROUTINE PERFORMS AN EXPLICIT DEFLATION OF THET.
C
      C = ONE
      S = ZERO
      G = ALF(1) - THET
      P = G**2
      DO 100 I=1,J-1
         B = BET2(I+1)
         R = P + B
         BET2(I) = S*R
         OLDC = C
         C = P/R
         S = B/R
         OLDG = G
         A = ALF(I+1)
         G = C*(A - THET) - S*OLDG
         ALF(I) = OLDG + (A - G)
         IF ( C .EQ. ZERO ) THEN
            P = OLDC*B
         ELSE
            P = G**2/C
         END IF
100   CONTINUE
      BET2(J) = S*P
      ALF(J) =  G + THET
      RNM2 = RNM2*S
      J = J - 1
      RETURN
      END
```

Function NUMLESS

This routine performs the $\mathbf{LDL}^T$ factorization of the tridaigonal matrix $\mathbf{T}_j$ to obtain eigenvalue counts for $\mathbf{T}_j - \varsigma$. By Sylvester inertia theorm [12], $\mathbf{D}$ has the same signature as $\mathbf{T}_j - \varsigma$. $\mathbf{L}$ is not needed and therefore is not computed. The diagonal elements of $\mathbf{D}$ are not preserved but a record of the number of negative or positive terms is kept and returned in NUMLES. NUMLES is called by ANALZT and NEWCOR.

```
      INTEGER FUNCTION NUMLES(ALF,BET2,ZETA,N,INC,EPS)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ALF(1),BET2(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... ROUTINE TO PERFOM THE SPECTRUM SLICING OF A TRIDIAGONAL MATRIX.
C.... IF INC = 1, NUMLES RETURNS THE NUMBER OF EIGENVALUES BELOW ZETA.
C.... IF INC = -1, NUMLES RETURNS THE NUMBER OF EIGENVALUES ABOVE ZETA.
C
C.... INPUTS
C.... ZETA      THE SHIFT TO BE APLIED TO T
C.... N         DIMENSION OF THE TRIDIAGONAL MATRIX
C.... INC       INDEX TO INDICATE ABOVE OR BELOW
C
C.... OUTPUTS
C.... NUMLES    THE NUMBER OF EIGENVALUES ABOVE/BELOW ZETA.
C
      DEL = ONE
      K = 0
      DO 10 J=1,N
         DEL = (ALF(J) - ZETA) - BET2(J)/DEL
         IF ( DEL .EQ. ZERO ) DEL = EPS*BET2(J+1)*INC
         IF ( DEL .LT. ZERO ) K = K + 1
10    CONTINUE
C
      NUMLES = K
      IF ( INC .LT. 0 ) NUMLES = N - K
      RETURN
      END
```

Subroutine QLBOT

This routine computes the bottom element of the eigenvector of tridiagonal $\mathbf{T}_j$ corresponding to the eigenvalue contained in THET. The algorithm performs a step of the QL factorization. The product of the sine of the rotation angles is the desired bottem element and is returned in BOT. NEWCOR is the only routine that calls QLBOT.

```
      SUBROUTINE QLBOT(ALF,BET2,THET,BOT,J)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION ALF(1),BET2(1)
      DATA ONE,ZERO/1.0D0,0.0D0/
C
C.... COMPUTE THE BOTTOM ELEMENT OF THE NORMALIZED EIGENVECTOR OF A
C.... TRIDIAGONAL MATRIX CORRESPONDING TO EIGENVALUE THET.
C
C.... INPUTS
C.... ALF(J)     DIAGONALS OF T
C.... BET2(J)    SQUAR OF THE OFF-DIAGONALS OF T
C.... THET       EIGENVALUE OF T
C.... J          DIMENSION OF THE TRIDIAGONAL MATRIX
C
C.... OUTPUTS
C.... BOT        BOTTOM ELEMENT OF THE NORMALIZED EIGENVECTOR
C
      BOT = ONE
      C = ONE
      S = ZERO
      G = ALF(J) - THET
      P = G**2
      DO 100 I=J-1,1,-1
         B = BET2(I+1)
         R = P + B
         OLDC = C
         C = P/R
         S = B/R
         OLDG = G
         A = ALF(I)
         G = C*(A - THET) - S*OLDG
         IF ( C .EQ. ZERO ) THEN
            P = OLDC*B
         ELSE
            P = G**2/C
         END IF
         BOT = BOT*S
100   CONTINUE
      RETURN
      END
```

Subroutine MOVE1

This routine inserts the value of T into the L-th location of the array Y. Elements of Y are shifted up or down to make room for T. If MINC is positive the shift is downward. MOVE1 is only called by ANALZT for data management.

```
      SUBROUTINE MOVE1(Y,K,L,MINC,T)
C
C.... MOVES THE CONTENT OF Y TO OPEN A SPACE FOR T.
C
C.... Y      THE ARRAY TO TO BE REORGANIZED.
C.... K      THE POSITION IN Y OF THE NEW ELEMENTS T.
C.... L      END OF THE DATA IN Y
C.... MINC   THE INCREAMENT +1 OR -1
C.... T      THE NEW ELEMENT TO BE INSERTED.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION Y(1)
      DO 100 I=L,K-MINC,MINC
         Y(I) = Y(I+MINC)
100   CONTINUE
      Y(K) = T
      RETURN
      END
```

References

[1] J. J. Dongarra, C. B. Moler, J. R. Bunch and G. W. Stewart, *LINPACK Users' Guide,* SIAM, Philadelphia, 1979.

[2] T. Ericsson and A. Ruhe, "The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems," *Math. Comp.* Vol. 35, pp. 1251-1268, 1980.

[3] Golub, G., Underwood, R., and Wilkinson, J. H., "The Lanczos Algorithm for the symmetric $Ax = \lambda Bx$ Problem," *Tech. Rep. STAN-CS-72-720,* Comp. Sci. Dept., Stanford University, 1972.

[4] J. Grcar, *Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method,* Ph.D. Thesis, University of Illinois at Urbana-Champain, 1981.

[5] C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *J. Res. Nat. Bur. Standards,* Vol. 45, pp. 255-281, 1950.

[6] B. Nour-Omid, B. N. Parlett, and R. L. Taylor, "Lanczos versus Subspace Iteration for Solution of Eigenvalue Problems," *Inter. J. Num. Meth. Eng.,* Vol. 19, pp. 859-871, 1983.

[7] B. Nour-Omid and B. N. Parlett, "How to Implement the Spectral Transformation," *Tech. Rep. PAM-224* (Center for Pure and Applied Mathematics, University of California, Berkeley, 1984).

[8] B. Nour-Omid and R. W. Clough, "Dynamic Analysis of Structures Using Lanczos Coordinates," *Earthquake Engineering and Structural Dynamics,* Vol. 12, 1984.

[9] C. C. Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," *J. Inst. Math. Appl.,* Vol. 10, pp. 373-381, 1972.

[10] C. C. Paige, "Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix," *J. Inst. Math. Appl.,* Vol. 18, pp. 341-349, 1976.

[11] C. C. Paige, "Accuracy and Effectiveness of the Lanczos Algorithm for Symmetric Eigenproblems," *Lin. Alg. Appl.,* Vol. 34, pp. 235-258, 1980.

[12] B. N. Parlett, *The Symmetric Eigenvalue Problem,* Prentice-Hall, Englewood Cliffs, 1980.

[13] B. N. Parlett and D. Scott, "The Lanczos Algorithm with Selective Orthogonalization," *Math. Comp.,* Vol. 33, No. 145, pp. 217-238, 1979.

[14] B. N. Parlett, H. D. Simon, and L. M. Stringer, "On Estimating the Largest Eigenvalue with the Lanczos Algorithm," *Math. Comp.,* Vol. 34, No. 157, Jan 1982.

[15] B. N. Parlett and B. Nour-Omid, "The Use of Refined Error Bounds when updating Eigenvalues of Tridiagonals," *Tech. Rep. PAM-175* (Center for Pure and Applied Mathematics, University of California, Berkeley, 1983).

[16] R. L. Taylor, and J. M. Rollins, "Modified Subspace Computation for Eigenvalue and Eigenvectors," *Report No. 78-1,* prepared for PMB Systems Engineering Inc., San Francisco, 1978.

[17] D. S. Scott, "Analysis of the Symmetric Lanczos Process," *Tech. Rep. ERL-M78/40* Electronics Research Lab., University of California, Berkeley, 1978.

[18] Simon, H. D., "The Lanczos Algorithm with Partial Reorthogonalization," *Math Comp.,* Vol. 42, No. 165; pp. 115-142, Jan 1984.

[19] H. D. Wilkinson, *The Algebraic Eigenvalue Problem,* Clarendon, Oxford, 1965.

[20] Wilson, E. L., Yuan, M., and Dickens, J. M., "Dynamic Analysis by Direct Superposition of Ritz Vectors," *Earthquake Engineering and Structural Dynamics,* Vol. 7, pp. 405-411, 1979.