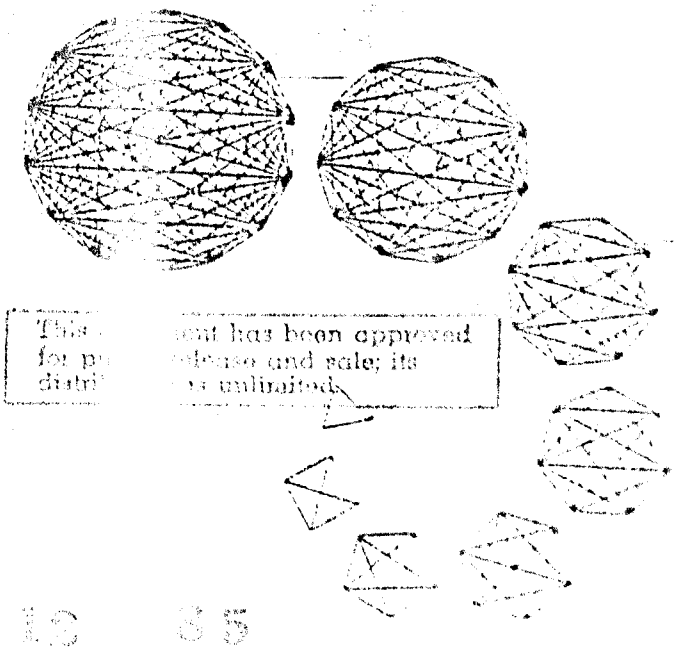
CENTER FOR PURE AND APPLIED MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY

PAM-175

# The Use of Refined Error Bound when Updating Eigenvalues of Tridiagonals

B. N. Parlett
B. Nour-Omid





## Legal Notice

# The Use of Refined Error Bound when Updating Eigenvalues of Tridiagonals

B. N. Parlett†

B. Nour-Omid‡



† Department of Mathematics, and the Computer Science Division of the Department of Electrical Engineering and Computer Science, University of California at Berkeley

‡ Center for Pure and Applied Mathematics, University of California at Berkeley

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

## ABSTRACT

The Lanczos algorithm is used to compute some eigenvalues of a given symmetric matrix of large order. At each step of the Lanczos algorithm it is valuable to know which eigenvalues of the associated tridiagonal matrix have stablized at eigenvalues of the given symmetric matrix. We present a robust algorithm which is fast ($20j$ to $40j$ operation at $j$-th Lanczos step), uses about 30 words of extra storage, and has a fairly short program (approximately 200 executable statements).


## TABLE OF CONTENTS

## 1. Introduction

Let $\mathbf{T}_n$ be a symmetric tridiagonal matrix of order $n$ and let $\mathbf{T}_j$ denote its leading principal submatrix of order $j$. It is a curious and useful fact that if the eigenvalues of $\mathbf{T}_j$ are regarded as functions of $j$ then, as $j$ increases, some of these eigenvalues tend to stagnate at comparatively low values of $j$. Typically if $n = 400$ the largest eigenvalue of $\mathbf{T}_j$ will remain unchanged, to 15 decimal digits, for all $j$ from say 21 to 400. The next largest might settle down at $j = 26$, and so on. We call $j$ the step number.

This article presents a new, efficient algorithm for updating a certain data structure associated with $\mathbf{T}_j$ as $j$ increases. It includes a few eigenvalues of $\mathbf{T}_j$ at each end of the spectrum and with each eigenvalue $\vartheta_i^{(j)}$ is associated an error bound $\beta_{ji}$. The number of eigenvalues at each end is variable and depends on the $\beta_{ji}$ in a complicated way. Roughly speaking the goal is to include all the outermost $\vartheta_i^{(j)}$ whose bounds $\beta_{ji}$ indicate that they are likely to stagnate in the next two or three steps. The algorithm is designed to monitor as few eigenvalues as possible consistent with the mandate to detect the precise value of $j$ at which $\vartheta_i^{(j)}$ stagnates to working precision.

Our algorithm is called ANALYZE T and was developed to be part of the inner loop of the Lanczos algorithm. Its job is to provide information so that the algorithm can be terminated at the first possible step at which all wanted eigenvalues and eigenvectors (of the operator given to Lanczos) are determined to the required accuracy. It is desirable that ANALYZE T increases the cost of a Lanczos step by a modest amount. It requires between $10j$ and $100j$ arithmetic operations at each step depending on whether no eigenvalues are stagnating or several. The important property is that the cost is **linear** in $j$. The profiles of Lanczos runs in Section 10 give the flavor of how the algorithm behaves.

We have spent considerable time in trying to make ANALYZE T short and

intelligible as well as robust and efficient.

Although it is not intended as a tool for computing all the eigenvalues of $T_n$ ANALYZE T can do that job. On several examples (see Section 10) with $n$ = 20 to 40 it took about 2.3 times as long as the EISPACK QR program TQL1. However half of that time is spent at step $n$ in determining those eigenvalues in the middle of the spectrum which had not settled down at all. Our program uses no more storage than TQL1.

We say little about the Lanczos algorithm itself and confine it to Section 3. For those readers who want to skip the Lanczos material, we have put the important notations and definitions in the very next section. Section 4 presents the theorems on which the program is based. This is followed in Section 5 by some very helpful pictures of what usually happens and what can happen to the spectrum as $j$ increases. Then comes ANALYZE T (§ 6), FIND THETA (§ 7), evaluation of $s(j)^2$ (§ 8), deflation (§ 9), and numerical results (§ 10).

## 2. Notation and Terminology

Upper case Roman letters stand for matrices. Symmetric letters stand for symmetric matrices (**A**, **H**, **M**, **T**, **U**, **V**, **W**, **X**, **Y**). Lower case Roman letters stand for column vectors; lower case Greek letters stand for real numbers. The $j \times j$ identity matrix, $\mathbf{I}_j$, is frequently omitted if the context precludes ambiguity. The symbol := indicates either a definition or an assignment in a program.

The Lanczos vectors $\mathbf{q}_1, \mathbf{q}_2, \cdots$ are the columns of a matrix $\mathbf{Q}_j := (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j)$ which would be orthonormal if exact arithmetic were used. The tridiagonal matrices produced by the Lanczos algorithm are written

$$\mathbf{T}_j := \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & & \\ & & & & \beta_j \\ & & & \beta_j & \alpha_j \end{bmatrix} .$$

and $\beta_i > 0$; $i = 2, 3, \ldots, j$. We use $\beta_1$ for other purposes.

Let $\chi_j(\zeta) := det[\mathbf{T}_j - \zeta]$. For reasons given in Section 3 the eigenvalues of $\mathbf{T}_j$ are called *Ritz values*. We use *two* different indexing schemes for the Ritz values and choose which ever is most appropriate. The hypothesis $\beta_j > 0$ compels the Ritz values to be distinct, but some may be equal to working accuracy even though no $\beta_i$ is small. The Ritz values at step $j$ are ordered by

$$\vartheta_1^{(j)} < \vartheta_2^{(j)} < \cdots < \vartheta_{j-1}^{(j)} < \vartheta_j^{(j)},$$

and/or by

$$\vartheta_j^{(j)} < \vartheta_{j+1}^{(j)} < \cdots < \vartheta_2^{(j)} < \vartheta_1^{(j)}.$$

Whenever possible we drop the superscript j. The normalized eigenvector of $\mathbf{T}_j$ associated with $\vartheta_i^{(j)}$ is $\mathbf{s}_i$, thus

$$(\mathbf{T}_j - \vartheta_i^{(j)}\mathbf{I}_j)\mathbf{s}_i = 0$$

and $\| \mathbf{s}_i \| = 1$. We use the Euclidean vector norm exclusively. The $k^{\text{th}}$ element of

$\mathbf{s}_i$ is $s_i(k)$. The *Ritz vector* associated with $\vartheta_i^{(j)}$ is $\mathbf{y}_i^{(j)} := \mathbf{Q}_j \mathbf{s}_i$. The *error bound* associated with $\vartheta_i^{(j)}$ is

$$\beta_{ji} := \beta_{j+1} | s_i(j) |.$$

These important quantities are introduced in Section 3. In some places our symbols $\mathbf{Q}_j$, $\mathbf{T}_j$ denote quantities stored in the computer. The relative precision, or roundoff unit, of the arithmetic processor is $\varepsilon$. It is the largest machine number such that the instruction $1 + \varepsilon$ yields 1.

There is an important threshold connected with the technique of selective orthogonalization. A Ritz value $\vartheta_i^{(j)}$ is *good* (respectively *bad*) according as $\beta_{ji} \leq \sqrt{\varepsilon}\, \|\mathbf{A}\|$ (respectively >). A good Ritz value is one that has usually stabilized to almost full working precision at an eigenvalue $\lambda$ of $\mathbf{A}$. Among the good Ritz values there may be some *threshold* values, namely those which have changed from bad to good at this particular step. Selective orthogonalization (described in [Parlett, 1980], p 275-284) needs to know the indices of the threshold values. **Our algorithm watches those bad Ritz values which are expected to become good within the next few steps.**

## 3. The Lanczos Algorithm and Its Implementation

What follows is a brief outline of material in Chapters 13 and 15 in [Parlett, 1980]. It is directed to the reader who is not familiar with the algorithm and its use with the spectral transformation.

Suppose that we possess a program which, when given any $n$-vector $\mathbf{x}$, returns the $n$-vector $\mathbf{Ax}$. Thus $\mathbf{A}$ need not be an explicit $n \times n$ matrix. One important instance of such an $\mathbf{A}$ comes from the spectral transformation of the generalized eigenvalue problem, [Ericsson and Ruhe, 1980] . The Problem

$$(\mathbf{W} - \lambda \mathbf{M})\mathbf{z} = 0$$

is changed into

$$(\mathbf{M}^{\frac{1}{2}}(\mathbf{W} - \sigma \mathbf{M})^{-1}\mathbf{M}^{\frac{1}{2}} - \nu \mathbf{I})\mathbf{x} = 0,$$

where $\nu = 1/(\lambda - \sigma)$ and $\mathbf{x} = \mathbf{M}^{\frac{1}{2}}\mathbf{z}$. Here $\mathbf{A} := \mathbf{M}^{\frac{1}{2}}(\mathbf{W} - \sigma \mathbf{M})^{-1}\mathbf{M}^{\frac{1}{2}}$ and $\mathbf{Ax}$ is found in three steps: form $\mathbf{w} := \mathbf{M}^{\frac{1}{2}}\mathbf{x}$, solve $(\mathbf{W} - \sigma \mathbf{M})\mathbf{v} = \mathbf{w}$ for $\mathbf{v}$, and form $\mathbf{u} := \mathbf{M}^{\frac{1}{2}}\mathbf{v}$. The cost of factoring $\mathbf{W} - \sigma \mathbf{M}$ is a critical factor in the computation. (A variation on this reduction of $\mathbf{W} - \lambda \mathbf{M}$ that uses $(\mathbf{W} - \sigma \mathbf{M})^{-1}\mathbf{M}$ is discussed in [Nour-Omid and Parlett, 1983]).

In the dynamic analysis of structures $\mathbf{W}$ is the positive definite stiffness matrix, $\mathbf{M}$ is the positive semidefinite mass matrix and $\sigma$ is a suitable shift parameter. Sometimes all the $\lambda$'s in a given interval are required. In that case $\sigma$ is chosen inside the interval and, in terms of $\nu$, the task is to compute all eigenvalues of $\mathbf{A}$ *outside* an interval containing the origin. This is the situation in which the Lanczos algorithm works best because Lanczos cannot help generating approximations to eigenvectors belonging to **both** ends of $\mathbf{A}$'s spectrum whether they are wanted or not.

During the $j^{\text{th}}$ step the **exact** Lanczos process computes in order, $\alpha_j$, $\beta_{j+1}$, $\mathbf{q}_{j+1}$ to satisfy

$$\mathbf{q}_{j+1}^{t}\mathbf{q}_j = 0, \quad \mathbf{q}_{j+1}^{t}\mathbf{q}_{j-1} = 0, \quad \|\mathbf{q}_{j+1}\| = 1$$

and

$$\mathbf{q}_{j+1}\beta_{j+1} = \mathbf{A}\mathbf{q}_j - \mathbf{q}_j\alpha_j - \mathbf{q}_{j-1}\beta_j .$$

It turns out that $\mathbf{q}_{j+1}^t\mathbf{q}_i = 0$ for $i < j-1$.

The relationship governing the practical algorithm is

$$\left[\begin{array}{c} \mathbf{A} \end{array}\right]\left[\begin{array}{c} \mathbf{Q}_j \end{array}\right] - \left[\begin{array}{c} \mathbf{Q}_j \end{array}\right]\left[\begin{array}{c} \mathbf{T}_j \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \end{array}\right] + \mathbf{F}_j = \mathbf{q}_{j+1}\beta_{j+1}\mathbf{e}_j^t + \mathbf{F}_j$$

where $\mathbf{F}_j$ accounts for roundoff errors and $\| \mathbf{f}_i \| \approx \sqrt{n}\,\varepsilon \| \mathbf{A} \|$, independent of $j$. Seldom is $\beta_{j+1}$ small and the algorithm should be halted if $\beta_{j+1} < \sqrt{\varepsilon}\,\| \mathbf{A} \|$. However a Lanczos run is normally halted for other reasons. Let $(\vartheta_i, \mathbf{s}_i)$ be a typical eigenpair of $\mathbf{T}_j$; $(\mathbf{T}_j - \vartheta_i)\mathbf{s}_i = 0$, $\| \mathbf{s}_i \| = 1$. The Ritz vector for $\vartheta_i$ is $\mathbf{y}_i := \mathbf{Q}_j\mathbf{s}_i$, and on multiplying the governing equation given above by $\mathbf{s}_i$ we find

$$\mathbf{A}\mathbf{y}_i - \mathbf{y}_i\vartheta_i = \mathbf{q}_{j+1}\beta_{j+1}s_i(j) + \mathbf{F}_j\mathbf{s}_i .$$

Take norms and use the triangle inequality to find

$$\| \mathbf{A}\mathbf{y}_i - \mathbf{y}_i\vartheta_i \| \le \beta_{ji} + \| \mathbf{F}_j \| ,$$

where

$$\beta_{ji} := \beta_{j+1}|s_i(j)| .$$

A well known error bound (see section 4) states that there is an eigenvalue $\nu$ of $\mathbf{A}$ satisfying

$$|\nu - \vartheta_i| \le \| \mathbf{A}\mathbf{y}_i - \mathbf{y}_i\vartheta_i \| / \| \mathbf{y}_i \| .$$

Thus

$$|\nu - \vartheta_i| \le (\beta_{ji} + \| \mathbf{F}_j \|) / \| \mathbf{y}_i \| .$$

Better bounds on $|\nu - \vartheta_i|$ are described in Section 4.

Roundoff errors have a strong effect on the simple algorithm. They cause the Lanczos vectors to become almost linearly dependent. In fact,

$$|\mathbf{y}_i^t\mathbf{q}_{j+1}| = \gamma_i / \beta_{ji},$$

where $\gamma_i$ is a computed roundoff term satisfying

$$|\gamma_i| \leq const \cdot \varepsilon \cdot \|\mathbf{A}\|.$$

This illuminating result is due to C. C. Paige, see [Paige, 1976] or [Parlett, 1980, p.264-268]. Extensive observation suggests that the constant is close to 1. Paige's theorem says that loss of orthogonality implies convergence; i.e. if $\mathbf{y}_i^t \mathbf{q}_{j+1}$ rises to $10^{-8}$ (rather than $10^{-14}$) then

$$\beta_{ji} \leq 100\varepsilon \|\mathbf{A}\|, \text{ so } \vartheta_i \text{ has stabilized.}$$

Various modifications have been proposed to force the Lanczos vectors to be strongly linearly independent. We advocate a combination of selective orthogonalizaton and partial reorthogonalization, [Parlett and Scott, 1979] and [Simon, 1982]. This requires that the algorithm know those $\vartheta_i^{(j)}$ for which

$$\beta_{ji} \approx \sqrt{\varepsilon}\|\mathbf{A}\|.$$

Certain actions must then be taken to maintain semi-orthogonality ($\mathbf{q}_k^t \mathbf{q}_k < \sqrt{\varepsilon}$) among the Lanczos vectors. Semiorthogonality ensures that $\mathbf{T}_j$ is (to working precision) the same as would occur in exact arithmetic.

With or without selective orthogonalization the bounds $\beta_{ji}$ indicate which Ritz values are close to eigenvalues. Paige showed that for given $i$ and each $k > j$, there is a Ritz value $\vartheta_l^{(k)}$, where $l$ depends on $i$ and $k$, such that $\beta_{k,l} \leq \beta_{ji}$. In other words, there is an eigenvalue of $\mathbf{T}_k$ within $\beta_{ji}$ of $\vartheta_i^{(j)}$ for all $k > j$.

The gist of these remarks is that we would like to know those $\vartheta_i^{(j)}$ whose $\beta_{ji}$ are near to the threshold $\sqrt{\varepsilon}\|\mathbf{A}\|$. Before designing a program to compute some Ritz values and their bounds we need insight into how the Ritz values change as $j$ increases. This is the subject of § 5.

**4. Relevant Theorems and Facts**

Soon after Napoleon was defeated at the battle of Waterloo (1815), Cauchy proved the remarkable interlace theorem for eigenvalues of symmetric matrices. In our case, it says

$$\vartheta_1^{(j+1)} < \vartheta_1^{(j)} < \vartheta_2^{(j+1)} < \vartheta_2^{(j)} < \cdots < \vartheta_{\underline{2}}^{(j)} < \vartheta_{\underline{1}}^{(j+1)} < \vartheta_{\underline{1}}^{(j)} < \vartheta_{\underline{1}}^{(j+1)}.$$

The inequalities are strict (in exact arithmetic) because $\beta_i \neq 0$, $i > 1$. However, it often happens that $\vartheta_i^{(j+1)}$ equals $\vartheta_i^{(j)}$ to within working accuracy.

For any symmetric matrix **A** we have the following residual error bounds:

**Theorem** 1: for any $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, and any real $\sigma$ there is an eigenvalue $\lambda$ of **A** satisfying

$$|\lambda - \sigma| \leq \|\mathbf{A}\mathbf{x} - \mathbf{x}\sigma\|$$

**Proof** : If $\sigma = \lambda$ there is nothing to prove. If $\sigma$ is not an eigenvalue then

$$1 = \|\mathbf{x}\| = \|(\mathbf{A}-\sigma)^{-1}(\mathbf{A}-\sigma)\mathbf{x}\| \leq \|(\mathbf{A}-\sigma)^{-1}\|\,\|(\mathbf{A}\mathbf{x}-\sigma\mathbf{x})\|;\ \|(\mathbf{A}-\sigma)^{-1}\| = 1/|\lambda-\sigma|.\ \bullet$$

**Corollary** 1. For each $i$ there is a $k$ such that

$$|\vartheta_k^{(j+1)} - \vartheta_i^{(j)}| \leq \beta_{ji}$$

where $\beta_{ji} = \beta_{j+1}|s_i(j)|$.

**Proof** : Take $\mathbf{A} = \mathbf{T}_{j+1}$, $\mathbf{x} = \begin{pmatrix} \mathbf{s}_i \\ 0 \end{pmatrix}$, $\mathbf{T}_j\,\mathbf{s}_i = \mathbf{s}_i\,\vartheta_i^{(j)}$, $\|\mathbf{s}_i\| = 1$, in the error bound given above. •

Our choice of indices, together with Cauchy's interlace theorem, dictates that $k = i$ or $i + 1$ in Corollary 1. In any case, at the end of analyzing $\mathbf{T}_{j-1}$ there are a number of known intervals

$$I(i) := [\,\vartheta_i^{(j-1)} - \beta_{j-1,i}\,,\ \vartheta_i^{(j-1)} + \beta_{j-1,i}\,]$$

each of which is guaranteed to contain an eigenvalue of $\mathbf{T}_j$. This fact is of most use when the intervals are disjoint. We remark in passing that when the $\mathbf{T}_j$ arise from the Lanczos algorithm used with some form of orthogonalization then, to

within roundoff terms, each $I(i)$ contains an eigenvalue of the big matrix $\mathbf{A}$. This follows from $\|\mathbf{A}\mathbf{y}_i - \mathbf{y}_i\vartheta_i\| = (\beta_{ji} + \|\mathbf{F}_j\|)/\|\mathbf{y}_i\|$, and $\mathbf{y}_i = \mathbf{Q}_j\mathbf{s}_i^{(j)}$.

In most cases we can determine much smaller intervals that contain $\vartheta_i^{(j)}$ than the $I(i)$. The fact that $\vartheta_i^{(j)}$ is a Rayleigh quotient is crucial here. The result is due to [Temple, 1929] and [Kato, 1949]. Our proof is a modification of [Parlett, 1980], p. 222-224.

**Theorem 2:** Suppose that $\vartheta := \rho(\mathbf{y};\mathbf{A})$ is the Rayleigh quotient of a unit vector $\mathbf{y}$. If $[\vartheta, \vartheta+gap]$ contains no eigenvalues of $\mathbf{A}$ then there is an eigenvalue $\lambda$ of $\mathbf{A}$ satisfying

$$0 < \vartheta - \lambda < \|\mathbf{A}\mathbf{y} - \mathbf{y}\vartheta\|^2 / gap .$$

**Proof :** Let $\mathbf{A}\mathbf{z} = \mathbf{z}\lambda$, $\|\mathbf{z}\| = 1$ and decompose $\mathbf{y}$,

$$\mathbf{y} = \mathbf{z}\cos\psi + \mathbf{w}\sin\psi,\ \mathbf{w} \perp \mathbf{z},\ \|\mathbf{w}\| = 1.$$

Then

$$\mathbf{r}(\mathbf{y}) := \mathbf{A}\mathbf{y} - \mathbf{y}\vartheta = \mathbf{z}(\lambda - \vartheta)\cos\psi + (\mathbf{A} - \vartheta)\mathbf{w}\sin\psi$$

and, since $(\mathbf{A} - \vartheta)\mathbf{w} \perp \mathbf{z}$,

$$\|\mathbf{r}(\mathbf{y})\|^2 = (\lambda - \vartheta)^2\cos^2\psi + \|(\mathbf{A} - \vartheta)\mathbf{w}\|^2\sin^2\psi .$$

The key fact is that $\mathbf{r}(\mathbf{y}) \perp \mathbf{y}$, so

$$0 = \mathbf{y}^t\,\mathbf{r}(\mathbf{y}) = (\lambda - \vartheta)\cos^2\psi + \mathbf{w}^t(\mathbf{A} - \vartheta)\mathbf{w}\sin^2\psi .$$

Eliminating $\psi$ from the last two equations yields

$$\|\mathbf{r}(\mathbf{y})\|^2 = \frac{[(\lambda - \vartheta)^2\mathbf{w}^t(\mathbf{A} - \vartheta)\mathbf{w} + \mathbf{w}^t(\mathbf{A} - \vartheta)^2\mathbf{w}(\lambda - \vartheta)]}{\mathbf{w}^t(\mathbf{A} - \lambda)\mathbf{w}}$$

$$= (\vartheta - \lambda)\,\frac{\mathbf{w}^t(\mathbf{A} - \lambda)(\mathbf{A} - \vartheta)\mathbf{w}}{\mathbf{w}^t(\mathbf{A} - \lambda)\mathbf{w}} \qquad (*)$$

Next expand $\mathbf{w} = \sum_i \mathbf{z}_i\,\xi_i$, in terms of $\mathbf{A}$'s eigenvectors $\mathbf{z}_i$, $\mathbf{A}\mathbf{z}_i = \mathbf{z}_i\,\alpha_i$. Thus

$$\mathbf{w}^t(\mathbf{A} - \lambda)(\mathbf{A} - \vartheta)\mathbf{w} = \sum_i (\alpha_i - \lambda)(\alpha_i - \vartheta)\xi_i^2 ,$$

and $\mathbf{w} \perp \mathbf{z}$ implies $\alpha_i \neq \lambda$. Now take $\lambda$ as the largest eigenvalue of $\mathbf{A}$ less than $\vartheta$, then by the gap hypothesis $(\alpha_i - \lambda)$ and $(\alpha_i - \vartheta)$ have the same sign for each $i$.

So

$\mathbf{w}^t(\mathbf{A}-\lambda)(\mathbf{A}-\vartheta)\mathbf{w} \geq \sum_{\alpha_i > \vartheta+gap} (\alpha_i-\lambda)(\alpha_i-\vartheta)\xi_i^2$, neglecting nonnegative terms

$> gap \sum_{\alpha_i > \vartheta+gap} (\alpha_i-\lambda)\xi_i^2$ ,

$\geq gap \sum_{all\ i} (\alpha_i-\lambda)\xi_i^2$, since the new terms are nonpositive,

$= gap\ \mathbf{w}^t(\mathbf{A}-\lambda)\mathbf{w}$. The result follows from (*). ▪

**Corollary** 1. If, for some $i$,

$$\vartheta_i^{(j+1)} < \vartheta_i^{(j)} < \vartheta_i^{(j)} + gap < \vartheta_{i+1}^{(j+1)} < \vartheta_{i+1}^{(j)}$$

then

$$\vartheta_i^{(j)} - \beta_{ji}^2/gap \leq \vartheta_i^{(j+1)}$$

**Proof** : Take $\mathbf{A} = \mathbf{T}_{j+1}$, $\mathbf{y} = \begin{bmatrix} \mathbf{s}_i \\ 0 \end{bmatrix}$, where $\mathbf{T}_j\,\mathbf{s}_i = \mathbf{s}_i\,\vartheta_i^{(j)}$, $\|\mathbf{s}_i\| = 1$, in the previous theorem. Note that $\mathbf{Ay} - \mathbf{y}\vartheta = \mathbf{e}_{j+1}\beta_{j+1}s_i(j)$.

**Corollary 2.** If, for some $i$,

$$\vartheta_{i-1}^{(j)} < \vartheta_{i-1}^{(j+1)} < \vartheta_i^{(j)} - gap < \vartheta_i^{(j)} < \vartheta_i^{(j+1)}$$

then

$$\vartheta_i^{(j+1)} \leq \vartheta_i^{(j)} + \beta_{j(-i)}^2/gap$$

The proof is analagous to the one above.

In practice we shall take $gap = \vartheta_{i+1}^{(j)} - \beta_{j,i+1} - \vartheta_i^{(j)}$ and test that the number of $\vartheta_k^{(j+1)}$ less than $\vartheta_i^{(j)} + gap$ is the same as the number less than $\vartheta_i^{(j)}$. To accomplish this test an eigenvalue counter is needed.

To find the precise index of a Ritz value one uses spectrum slicing; a much better technique than Sturm sequences. Let

$$\mathbf{T}_j - \xi = \mathbf{LDL}^t$$

be the triangular factorization of $\mathbf{T}_j - \xi$, where $\mathbf{D} = diag(\delta_1, \ldots, \delta_j)$. The quan-

tity $\delta_j(\xi)$ is called the "last pivot" function. Note that $\delta_j(\vartheta_i^{(j)}) = 0$ and $\chi_j(\vartheta_i^{(j)}) = 0$, but $\delta_j$ is a rational function, whereas $\chi_j$ is a polynomial (defined in Section 2).

The Sturm sequence technique computes $l$, the number of sign changes in the sequence $\{\chi_0(\xi), \chi_1(\xi), \cdots, \chi_j(\xi)\}$ where

$$\chi_k(\xi) = (\alpha_k - \xi)\chi_{k-1}(\xi) - \beta_k^2 \chi_{k-2}(\xi).$$

In contrast, the spectrum slice computes $l$ as the number of negative values in the set $\{\delta_1(\xi), \cdots, \delta_j(\xi)\}$. Here $\delta_0(\xi) = 1$ and

$$\delta_k(\xi) = \alpha_k - \xi - \beta_k^2 / \delta_{k-1}(\xi).$$

In either case $l$ is the number of Ritz values less than $\xi$. As users know, the $\delta$'s vary less wildly than the $\chi$'s.

Several properties of $s_i$, the eigenvector of $\vartheta_i^{(j)}$ are given in [SEP, Chap. 7]. We select the following

$$\begin{aligned} s_i(j)^2 &= -\chi_{j-1}(\vartheta_i^{(j)}) / \chi'_j(\vartheta_i^{(j)}), \\ s_i(1)^2 &= -\chi_{2,j}(\vartheta_i^{(j)}) / \chi'_j(\vartheta_i^{(j)}), \\ s_i(1)s_i(j) &= \beta_2\beta_3 \cdots \beta_j / \chi'_j(\vartheta_i^{(j)}). \end{aligned}$$

Here $\chi_{2,j}(\xi) := \det[\mathbf{T}_{2,j} - \xi]$ and $\mathbf{T}_{2,j}$ is the submatrix obtained by deleting row and column 1 from $\mathbf{T}_j$. The $-$ sign occurs because $\chi_j$ is not monic.

These formulas show that it is possible to compute an element of a normalized eigenvector without computing the other elements. Please note that there are two distinguished Sturm sequences of polynomials associated with $\chi_j$, namely $\{\chi_j, \chi_{j-1}, \chi_{j-2}, \cdots, \chi_0\}$ and $\{\chi_j, \chi'_j, \chi''_j, \cdots, \chi_j^{(j)}\}$. At a zero $\xi = \vartheta_i^{(j)}$, of $\chi_j$, we must have $\chi_{j-1}(\xi) \neq 0$ and $\chi'_j(\xi) \neq 0$, and it is remarkable that the quotient of these quantities yields the bottom element of $s_i$.

## 5. How Ritz Values Change

We present two figures which show all the Ritz values at each step of a run of the Lanczos algorithm. The first case is typical, the second is not. The figures need little comment but we emphasize some features by giving a table of the steps at which certain Ritz values stabilized to a certain accuracy.

**Figure** 1 Progress of the Ritz values. $\nabla$ indicates $\alpha_j$.

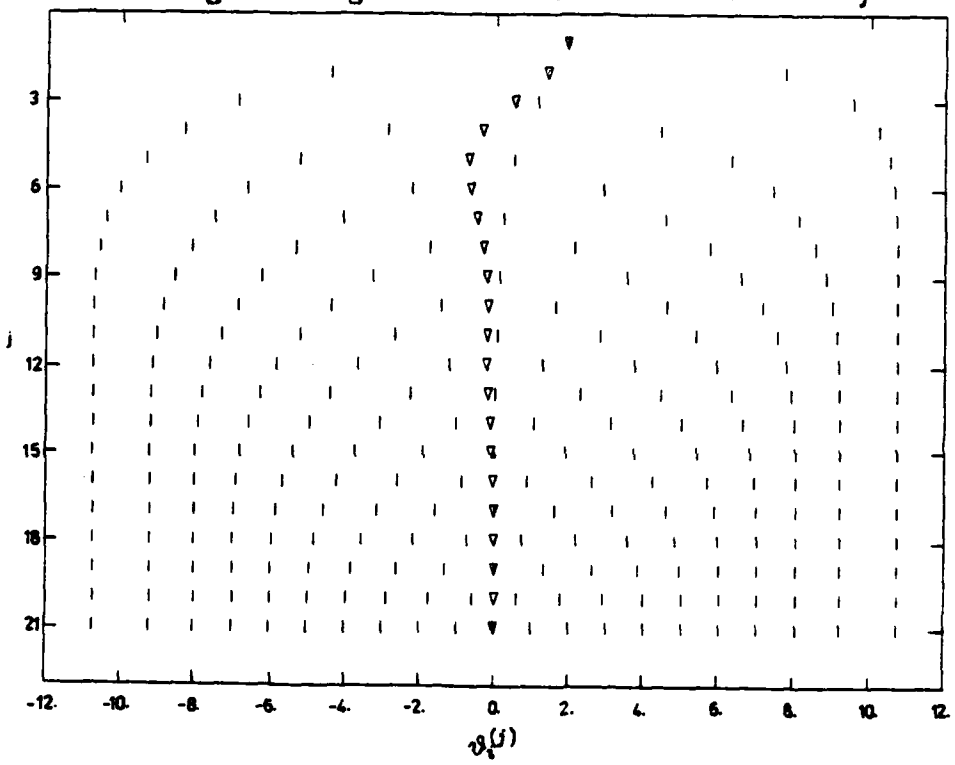


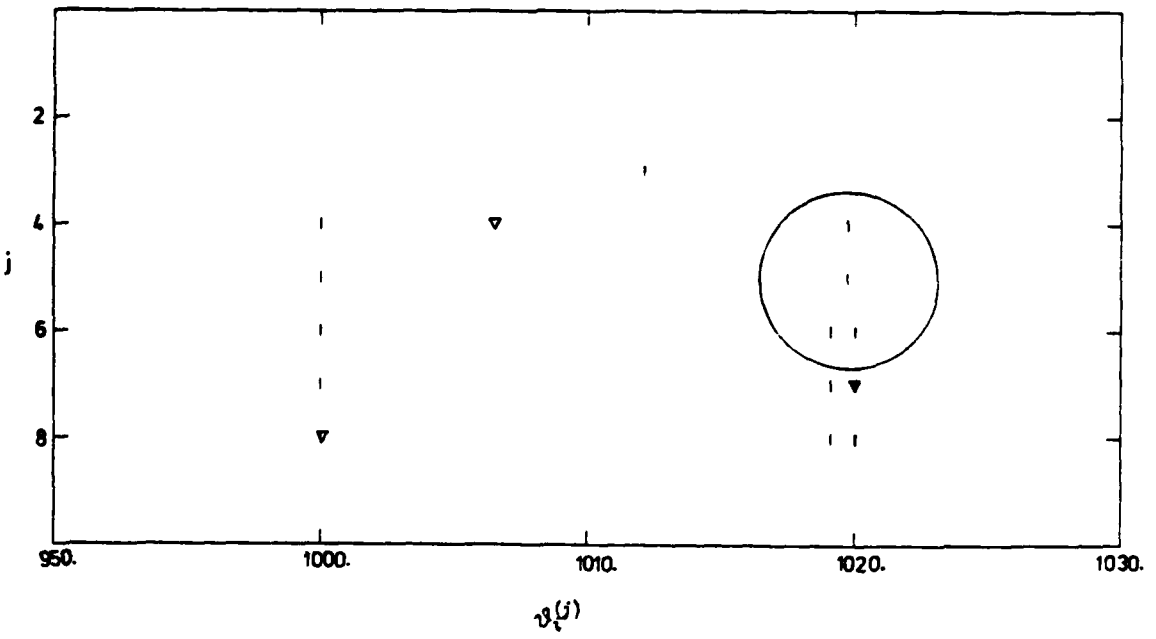

**Table** 1. Eigenvalue with index $i$, converged at step $j$ to 3 decimals.
$\vartheta_1 < \vartheta_2 < \cdots < \vartheta_{-2} < \vartheta_{-1}$

| eigen $i$ | step $j$ |
|---|---|
| -1 | 7 |
| 1 | 9 |
| -2 | 14 |
| 2 | 15 |
| 3 | 17 |
| -3 | 17 |
| 4 | 18 |
| -4 | 18 |
| 5 | 20 |
| -5 | 20 |

**Misconvergence**

The first example suggests that we could avoid computing the $\beta_{ji}$ and simply monitor the $\vartheta_i^{(j)}$. When a Ritz value stops changing, to working accuracy, we can be confident that it is good; i.e. that its bound $\beta_{ji} < \sqrt{\varepsilon}\,\|\,\mathbf{A}\,\|$. Life is not that simple however. A Ritz value can stabilize for several steps and then suddenly change! This phenomenon is called *misconvergence* and it is *not* a bizarre pathology which is never seen by normal human beings. It is not difficult to contrive examples where misconvergence endures for many steps. See [Parlett, Simon and Stringer, 1982] for more details.

**Figure 2.** Misconvergence of $\vartheta$. From Steps 4 to 5.




At step 4 $\vartheta$ stagnated at 1020 near the mean of a pair of close, but distinct eigenvalues and suddenly split into two Ritz values at step 6. The $\beta_{ji}$ values reveal the misconvergence at all times; they are not small during the premature stagnation.

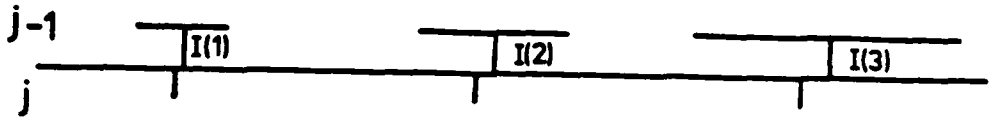
**INDEXING**

We now turn to the indexing problem. At the end of step $j$ we have $\vartheta_i$ and $\beta_{ji}$ for a few values of $i$. Moreover, for some index k the new Ritz value $\vartheta_k^{(j+1)}$ lies in

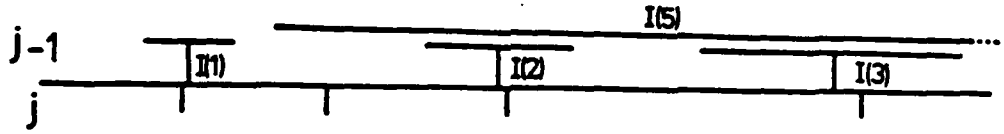
$$I(i):=\left[\vartheta_i^{(j)}-\beta_{ji},\ \vartheta_i^{(j)}+\beta_{ji}\right]$$

Suppose that $I(i)$ is disjoint from $I(l)$, $l \neq i$. Then, by the interlace theorem (in section 5), $k$ is either $i$ or $i+1$. We give three instances.

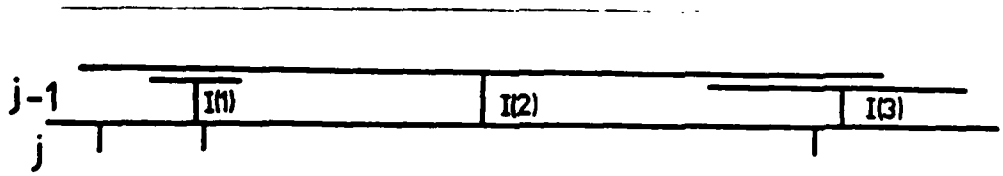
The stem of $\top$ marks the old Ritz value. The bar of $\top$ represents the subinterval $I$.




1. Normal Situation.




2. Apearance of a New Ritz Value.




3. Disappearance of a Ritz Value.

In the second example $\vartheta_2$ seems to have appeared out of the blue. It is essential to recognize this event and to find such $\vartheta_2$. That task is addressed in sections 6 and 7. Here we ask how such an event can occur.

***Observation 1.***

The index of the new Ritz value in $I(i)$ is usually, but not always $i$, for the outer values $i = \pm 1, \pm 2, \pm 3$.

***Observation 2.***

The place (i.e. the value of $i$) where the index of the new Ritz value in $I(i)$ changes from $i$ to $i+1$ (or from $-i$ to $-i-1$) usually, but not always, satisfies

$$\vartheta_{i-1}^{(j)} < \alpha_{j+1} < \vartheta_i^{(j)} \quad or \quad \vartheta_i^{(j)} < \alpha_{j+1} < \vartheta_{i+1}^{(j)}.$$

There is always one more Ritz value at each step of the Lanczos algorithm and we like to say that is is ***seeded*** by the new value of $\alpha$.

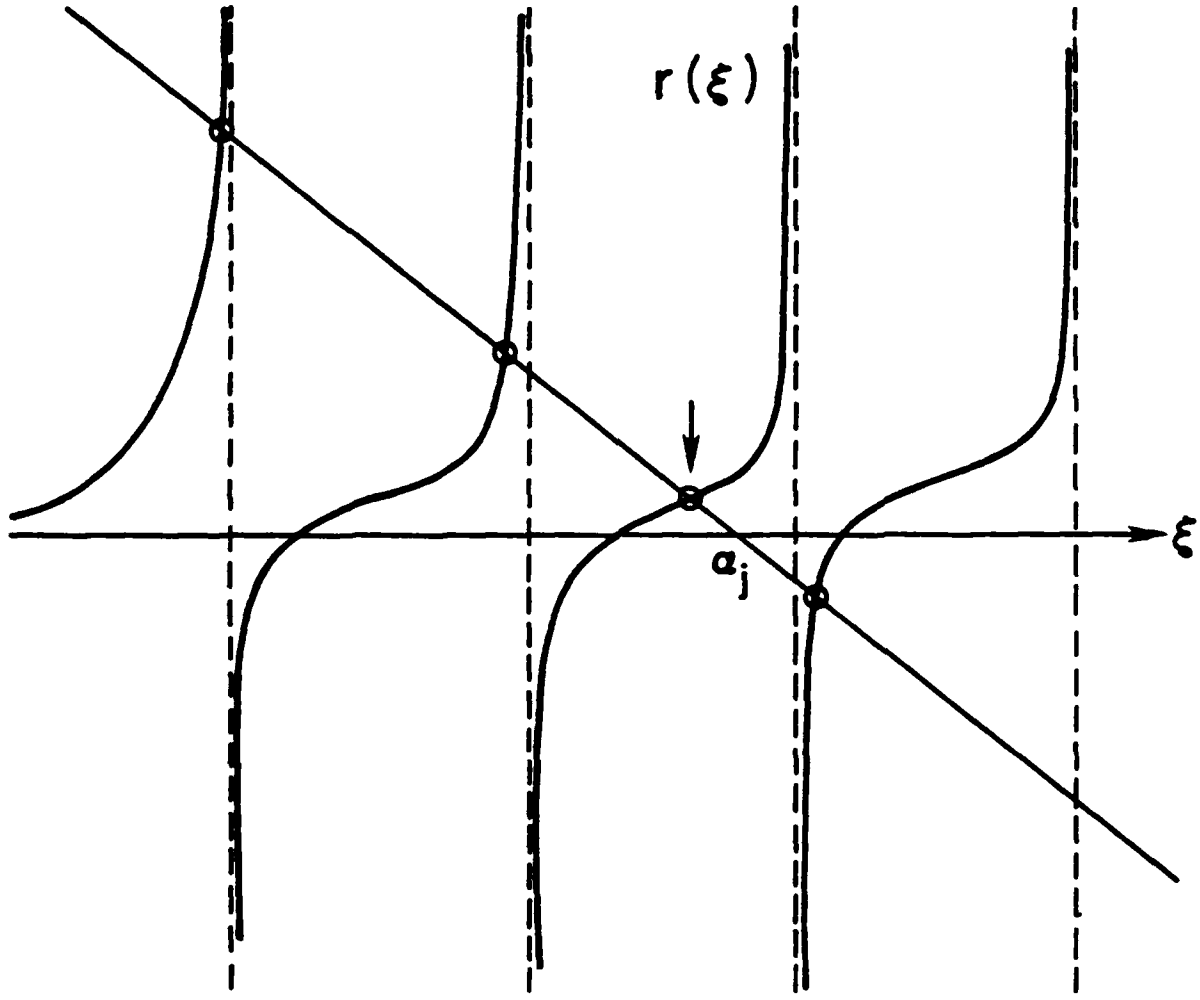
Observation 1 is equivalent to the remark that the new $\alpha$ is usually near the middle of the spectrum well away from the Ritz values being monitored. Observation 2 is best understood by noting that each $\vartheta_i^{(j+1)}$ is a zero of the rational (last pivot) function

$$\delta_{j+1}(\xi) = \alpha_{j+1} - \xi - \sum_{i=1}^{j} \frac{\beta_{ji}^2}{\vartheta_i^{(j)} - \xi} = \alpha_{j+1} - \xi - r(\xi)$$

In the figures below Case 1 occurs more frequently than Case 2. That is the gist of Observation 2.

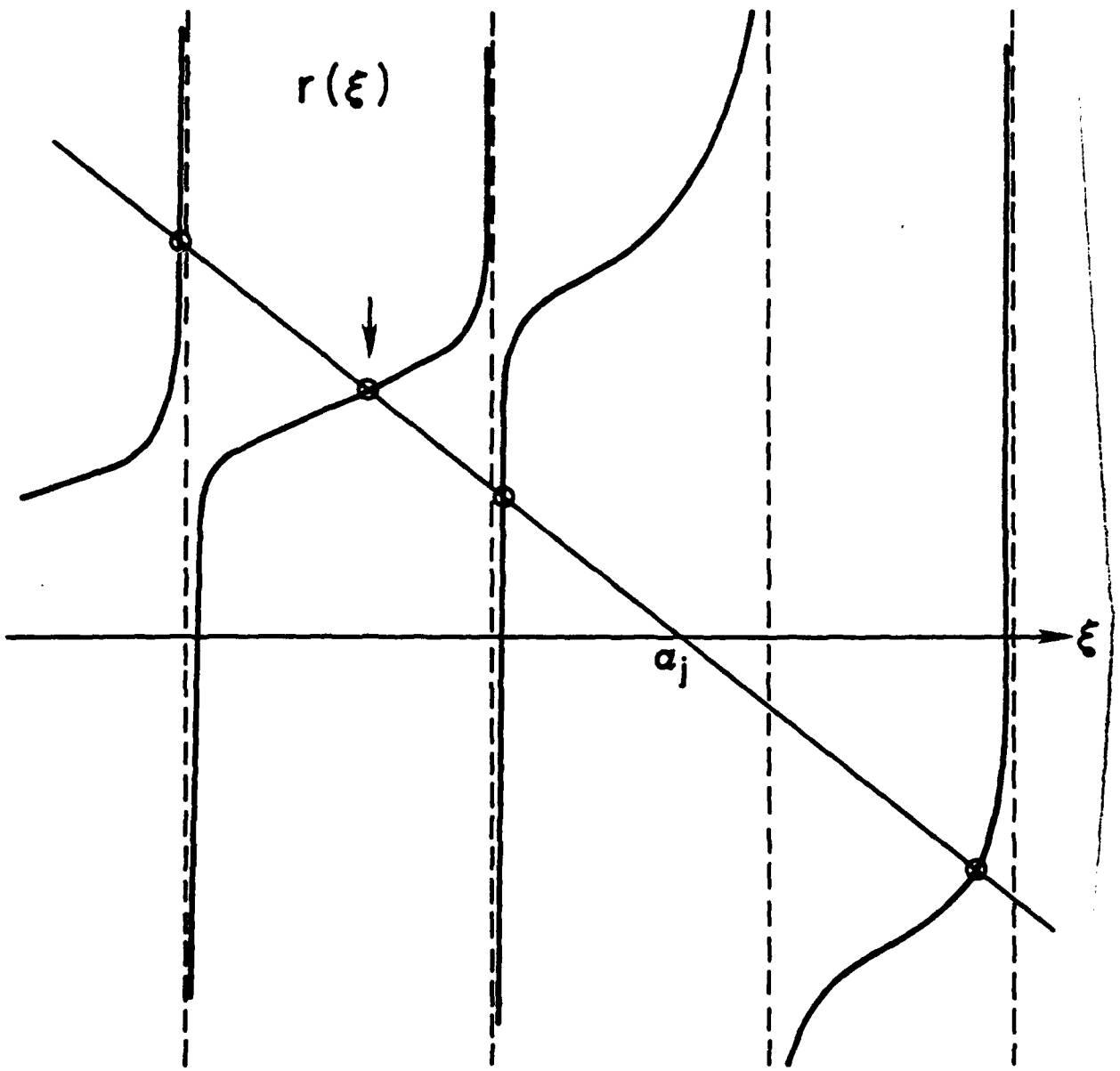
Case 1.

The extra Ritz value at step $j$ is indicated by the vertical arrow.



Case 2.

The extra Ritz value at step $j$ is indicated by the vertical arrow.



## 6. ANALYZE T

At step $j$ the program has knowledge of the intervals $I(1)$, $I(2)$,..., $I(l)$ at the left end of the spectrum of $\mathbf{T}_{j-1}$ and intervals $I(-1)$, $I(-2)$,..., $I(-r)$ at the right end. $I(k)$'s midpoint is $\vartheta_k^{(j-1)}$. The intervals are not necessarily disjoint, but each contains an eigenvalue of $\mathbf{T}_j$. The goal is to obtain corresponding intervals for $\mathbf{T}_j$. The adjustment of the values of $l$ and $r$ is a technical point discussed at the end of this section under the heading Phase II.

There are many ways to achieve this goal. What took a long time was the design of a program which would cope with all situations, would keep arithmetic operations and storage needs low, and yet be fairly short and simple. The previous sections showed that there are three distinct possibilities when an interval $I(i)$ is to be updated: normal, intrusion, and disappearance. The intelligibility of the algorithm depends on the way these cases are distinguished. We found that a particular Boolean variable clarified earlier versions of ANALYZE T. To justify its introduction we describe some simple algorithms that are not quite satisfactory. We will confine ourselves to the left end of the spectrum but the actual program can process either of them.

The updating of each $I(i)$ consists of two distinct phases. First comes the determination of a *small* search interval $S := [start\ ,\ fin]$ which is guaranteed to contain the *next* Ritz value $\vartheta$. Sometimes $S = I(i)$. Note that the index of $\vartheta$ need not be $i$. The interval $S$ is given to a subprogram FINDTHETA that computes the new Ritz value and bound writing them over $\vartheta(i)$ and $bj(i)$. The details are in the next section. The second phase records stabilized Ritz values and moves them out of $\vartheta(i)$ and decides whether to append new Ritz values or to delete the last one. The heart of the algorithm is Phase I.

**Phase I**

A good lower bound on the spectrum of $\mathbf{T}_j$ is essential. Our preference is for an instance of the optimal Lehmann/ Kahan bounds which requires no 'extra' information such as a norm of $\mathbf{T}_j$.

$$\textit{left bound} := \text{ left eigenvalue of } \begin{bmatrix} \vartheta(1) & \beta_j \\ \beta_j & \alpha_j \end{bmatrix}$$

$$:= [(\vartheta(1) + \alpha_j) - \sqrt{(4\beta_j^2 + (\vartheta(1) - \alpha_j)^2)}]/2$$

The algorithms are presented in an informal pseudocode. Pascal conventions are violated when convenient.

Algorithm 1.

```
start := left bound
for i := 1, l do
    fin := ϑ(i)
    call FINDTHETA(start, fin)
    start := fin
```

This algorithm is completely reliable and wins on simplicity. It exploits the Cauchy interlace theorem but ignores all the bounds of section 4 and consequently delivers unnecessarily large search intervals $S$ to FINDTHETA. We reject it simply on the grounds of arithmetic effort.

Let $bj(i) := \beta_{ji}$ and recall that $I(i) = [\vartheta(i) - bj(i)\ ,\ \vartheta(i) + bj(i)]$ contains a Ritz value. Hence a more efficient yet even shorter program than Algorithm 1 is

```
for i := 1, l do
    call FINDTHETA(I(i))
```

Alas, this program will sometimes fail even when the $I(i)$ are disjoint. It will miss an extra Ritz value that may be seeded by an $\alpha_j$ near the left end of the spectrum. For example it will miss $\vartheta_2$ in Case 2 of section 5.

To cope with these situations we employ two subprograms:

NUMLESS($\xi$) is the number of $\vartheta$ less than $\xi$.

MOVE makes room for an extra $\vartheta$ and $bj$ or closes a gap.

The next algorithm removes the bug in the one above.

Algorithm 2.

```
oldϑ := left bound
for i := 1, l do
    if NUMLESS(ϑ(i) − bj(i)) ≥ i then
        MOVE elements i, i+1, ··· , l down
        l := l + 1
        S := [oldϑ , ϑ(i) − bj(i)]
    else
        S := I(i)
    oldϑ := ϑ(i)
    call FINDTHETA(S)
```

Unfortunately Algorithm 2 malfunctions on those rare occasions when $\vartheta_i^{(j-1)}$ disappears at step $j$ as explained in section 5. The precise nature of the failure depends on FINDTHETA but the trouble arises because $I(i+1)$ is contained in $I(i)$ and the subroutine will be asked to find the same Ritz value twice. The remedy is to check for this possibility before processing $I(i+1)$.

Algorithm 3.

```
oldϑ := left bound
for i := 1, l do
    if NUMLESS(ϑ(i) − bj(i)) ≥ i then
        MOVE elements i, i+1, ··· , l down
        l := l + 1
        S := [oldϑ , ϑ(i) − bj(i)]
    else
        S := I(i)
    oldϑ := ϑ(i)
    call FINDTHETA(S)
    if ϑ(i) > ϑ(i+1) − bj(i+1) then
        MOVE elements i+2,...,l up
        l := l − 1
```

The efficiency of this program can be improved significantly for a modest increase in complexity. First the rare occasions when $\vartheta_i^{(j-1)}$ disappears are treated wastefully because FINDTHETA is given a big interval $I(i)$ rather than a small one $I(i+1)$. Much more serious is the failure to use the refined error bounds of section 4 whenever the intervals are disjoint. In fact Algorithm 3 already makes a relevant test but does so one iteration too late. A nice way to preserve the fact that $I(i+1)$ is disjoint from $I(i)$ for use in the next loop is by introducing the Boolean variable ***indexok*** which is true when $I(i)$ contains $\vartheta_i^{(j)}$ and false when it contains $\vartheta_{i+1}^{(j)}$. More precisely, introduce new variables

$$probe := \vartheta(i+1) - bj(i+1)$$

$$indexok := probe > \vartheta(i) \text{ and NUMLESS}(probe) = i$$

It turns out that if ***indexok*** is true at step $i + 1$ then the refined error bounds can be used at step $i$, and conversely. Thus one test serves two purposes.

Algorithm 4

```
oldϑ := left bound
probe := ϑ(1) − bj(1)
indexok := NUMLESS(probe) = 0
for i := 1, l do
    if indexok then
        start := max(oldϑ , probe)
        fin := ϑ(i) + min{bj(i) , bj(i)²/(ϑ(i) − ϑ(i−1))}
        probe := ϑ(i+1) − bj(i+1)
        if probe > ϑ(i) then
            indexok := NUMLESS(probe) ≤ i
            if indexok then start := max{start, ϑ(i) − bj(i)²/(probe − ϑ(i))}
      else
        MOVE elements i, i+1, ..., l down
        l := l + 1 ; indexok := true
        start := oldϑ ; fin := probe
    oldϑ := ϑ(i)
    call FINDTHETA(start , fin)
    if ϑ(i) > probe then
        MOVE elements i+2, ..., l up
        l := l − 1 ; indexok := true
```

Remark: It is only necessary to compute *indexok* explicitly when $probe > \vartheta(i)$. Otherwise it simply remains true. In principle *indexok* can be false at most once for each value of j because there is only one new Ritz value.

Algorithm 4 does not make optimal use of available knowledge. By Cauchy's theorem we have:

```
if indexok has remained true for all i so far then
    new ϑ is in [ϑ(i) - bj(i), ϑ(i)]
  else
    new ϑ is in [ϑ(i), ϑ(i) + bj(i)]
```

In order to combine these bounds with the refined ones it is necessary to keep a

Boolean variable ***newritz*** which remembers whether ***indexok*** has been false. Is the improvement in performance worth an extra Boolean variable? We are not sure, but from a mathematical viewpoint it seemed valid to implement fully the best bounds available to us.

The treatment of disappearing $\vartheta$'s is also wasteful. There is no need to give FINDTHETA any subset of $[\vartheta(i-1) + bj(i-1)\ ,\ \vartheta(i+1) - bj(i+1)]$ when it contains no Ritz values because that condition is easily checked in advance. The most economical treatment of disappearing $\vartheta_i^{(j-1)}$ is to skip the call to FINDTHETA and then close up the gap in the data structure. This requires a flag for this case and we choose to set $bj(i)$ to -1 thus dispensing with an extra variable.

The final version is Algorithm 5 given in Table 2. Of course, it is not a valid program as it stands because end conditions (when $i-1$ is not defined, for example) have not been treated. In addition the code must be written so that it works on the right end of the spectrum as well. These details obscure the issues of concern here. The full program is given in the appendix in Fortran 77.

**Table 2**

Algorithm 5. Phase I: update Ritz values and error bounds

```
old ϑ := left bound ; probe := ϑ(1) − bj(1) ;
indexok := NUMLESS(probe) = 0 ; newritz := false ;
for i := 1, l do
    if indexok then
        if newritz then
            start := ϑ(i)
            fin := start + min{bj(i), bj(i)^2 / (start − ϑ(i−1))}
        else
            start := max{old ϑ, probe}
            fin := ϑ(i)
        probe := ϑ(i+1) − bj(i+1)
        if probe > ϑ(i) then
            k := NUMLESS(probe)
            if k < i then
                bj(i) := -1.0
            else
                if not newritz then
                    indexok := k ≤ i
                    width := min{bj(i), bj(i)^2 / (start − ϑ(i))}
                    if |ϑ(i) − ϑ(i−1)| > bj(i) then
                        start := max{start, ϑ(i) − width}
        else
            MOVE elements i,...,l down
            l := l + 1 ; indexok := true ; newritz := true ;
            start := old ϑ ; fin := probe
    if bj(i) > 0.0 then
        old ϑ := ϑ(i)
        call FINDTHETA(start, fin)
    else
        MOVE elements i+2,...,l up ; i := i − 1 ;
        l := l - 1 ;indexok := true ; newritz := false
```

**Phase II**

At times it is necessary to append more Ritz values to the list or, less frequently, to delete them. The overall goal of ANALYZE T is to monitor as few values as possible consistent with the requirement of catching all extreme Ritz values at the step when their error bounds cross below the threshold $tol$. Recall from section 2 that

$$tol := \sqrt{\varepsilon} \cdot spread = \sqrt{\varepsilon}(\vartheta(-1) - \vartheta(1))$$

The program tries to achieve the goal by monitoring those Ritz values whose bounds lie in the window $[tol\ ,\ w \cdot tol]$ where $w$ is at our disposal. We lack a theory to dictate a proper value. If $w$ is too small ($w = 4$) then it is easy for a Ritz value to skip the window in a single step and so be missed. On the other hand if $w$ is too big ($w = 1024$) then Phase I wastes energy monitoring values long before they stabilize. Fortunately the behavior of the algorithm is not very sensitive to small changes in $\log_2 w$. Currently we take $w = 128$.

Should $\beta_{j+1}$ be unusually small (but greater than $tol$) then a good number of Ritz values may enter the window. This poses no difficulty to Phase II because all the current $\vartheta$ will stabilize and the program will automatically append more values. The reason for taking $w > 4$ is the fear that say $\vartheta(2)$ might overtake $\vartheta(1)$ in their race to stability. If Phase I only monitors $\vartheta(1)$ then it might discover $\vartheta(2)$ several steps after it stabilizes.

Phase II sweeps through the known $\vartheta(i)$ and removes any which have stabilized. Any gap in $\vartheta(\cdot)$ is closed up and $l$ is decreased. It would be simpler to leave these values in place and adjust pointers so that they were not inspected any more. However in our Lanczos program we associate 3 other variables with each computed eigenvalue and we wished to keep ANALYZE T free of this information. Moreover our mechanism isolates $\vartheta(\cdot)$ and $bj(\cdot)$ from the rest of the

Lanczos algorithm. These arrays are of length 8.

Another device which enhances the performance is to deflate stabilized Ritz values from **T**. Analyze T is independent of this feature. The deflation process is discussed in a later section.

Recall that Phase I sometimes inserts Ritz values into $\vartheta(\cdot)$ and frequently these intruding $\vartheta$ have large $bj$ values. The strategy for appending more Ritz values is clear. Phase II will go on appending Ritz values until it finds one *outside* the window or there is no more room. In particular $\vartheta(\cdot)$ holds at least one Ritz value at each end of the spectrum. No interval I is on hand that contains these new values to be appended. However the average gap ( := *avgap*) between the unknown Ritz values is easily computed and NUMLESS is used to check whether $[\vartheta(l)\ ,\ \vartheta(l) + avgap]$ contains the next value. If not the next subinterval of this length is checked and so on. Then FINDTHETA is called and $l$ is increased.

The most complicated expression in Phase II concerns the decision to drop $\vartheta(l)$. The drop is necessary to avoid the waste of carrying two slowly converging values at one end of the spectrum when all the action is happening at the other end. This happens when Lanczos is not used with inverted, shifted operators. All of the following must be satisfied

$$j > 8;\ l > 1;\ bj(l) > bj(l-1) > tol \cdot w$$

Three considerations suggested the separation of Phase II from Phase I. It is desirable to update both ends of the spectrum before appending new Ritz values to either end in order to prevent one end being driven out of $\vartheta(\cdot)$. Secondly it simplifies Phase I. The actual code is careful not to insert items prior to checking that there is room for them. Thirdly, Phase I does not need to know $\beta_{j+1}$. Consequently, if the computer permits it, Phase I can be run in

parallel with the computation of some vector operations in the main loop of the Lanczos algorithm. Phase II must wait until $\beta_{j+1}^2$ has been computed.

**Phase II. Remove and append Ritz values.**

$bj(i)$ containes $s_j(i)^2$, computed in FINDTHETA in Phase I.

```
for i := 1, l do
    bj(i) := sqrt(β²_{j+1} bj(i))
    append := i = l and another Ritz value needed
    if append then
        avgap := (ϑ(r) − ϑ(l))/(j − l − abs(r))
        fin := ϑ(l) + avgap
    if bj(i) < tol then
        insert ϑ(i) in EIG
    if append and enough room then
        while NUMLESS(fin)=l do fin := fin + avgap ;
        start := fin − avgap ; l := l + 1
        call FINDTHETA(start, fin)
        bj(l) := sqrt(β²_{j+1} s_j(l))
    if ϑ(l) not needed then l := l - 1
```

## 7. The Subprogram FINDTHETA

There are several good ways to compute the Ritz value in the given search interval $S$. The simplest is the bisection technique which has the advantage of using the already needed subprogram NUMLESS to evaluate the last pivot function $\delta_j$. Recall that the $\vartheta_i^{(j)}$ are zeros of the rational function $\delta_j$. The error bound is halved at each step although the actual error may be reduced by much more.

The bisection process is much less efficient than rival methods when the approximation is already good to three or four decimal places and that is precisely the situation facing FINDTHETA most, but not all of the time. Recall that for most intervals $I(i)$ the width is less than our window, namely $128 \cdot tol$. When the refined bounds are used we can expect the width of $S$ to be 10 or 100 times smaller than that. Consequently the starting approximation may be accurate to almost half of its digits in the majority of cases. Thus one or two steps of Newton's iteration should suffice independent of the precision $\varepsilon$ of the arithmetic operations.

There will be occasions (intruding and new Ritz values) when $S$ will be large (like the average gap between zeros) and so the Newton iteration must be protected by a bisection facility which chops down large $S$'s. This raises an interesting technical problem that receives little attention in text books. When should the switch from bisection to Newton be made? Our criterion is discussed later in this section.

There is a further attraction in using Newton's method to update the Ritz values. The preferred implementation of the Newton correction yields $s(j)$, the bottom component of the eigenvector, as a byproduct. This pleasant feature is the subject of the next subsection.

**The Newton Correction**

Newton's iteration function for the polynomial $\chi_j$ is

$$N(\xi) := \xi - \frac{\chi_j(\xi)}{\chi'_j(\xi)} = \xi - \frac{1}{(\chi'_j / \chi_j)}$$

and Newton's iteration computes from $\xi_1$ the sequence $\{\xi_i\}$ according to

$$\xi_{i+1} := N(\xi_i).$$

An obvious way to evaluate the correction term $-\chi'_j / \chi_j$ is to use the well-known three term recurrence: $\chi_0 = 1, \chi'_0 = 0, \beta_1^2 = 0,$

$$\text{for } k = 1, 2, \cdots, j \text{ repeat}$$

$$\chi'_k := (\alpha_k - \xi)\chi'_{k-1} - \beta_k^2 \chi'_{k-2} - \chi_{k-1},$$
$$\chi_k := (\alpha_k - \xi)\chi_{k-1} - \beta_k^2 \chi_{k-2}.$$

Unfortunately, this recurrence suffers from severe underflow/overflow problems. Fortunately, there is a more sedate alternative which we now derive. Recall from Section 4 that

$$\chi_j(\xi) = \prod_{i=1}^{j} \delta_i(\xi).$$

So

$$\frac{\chi'_j}{\chi_j} = \frac{d}{d\xi} \ln(\chi_j) = \frac{d}{d\xi} \sum_{i=1}^{j} \ln(\delta_i) = \sum_i \frac{\delta'_i(\xi)}{\delta_i(\xi)}.$$

Recall from Section 4 that

$$\delta_i = \alpha_i - \xi - \beta_i^2 / \delta_{i-1}.$$

Thus

$$\delta'_i = -1 + \left(\frac{\beta_i^2}{\delta_{i-1}}\right)\left(\frac{\delta'_{i-1}}{\delta_{i-1}}\right).$$

or, more conveniently

$$1 \le -\delta'_i = 1 + \left(\frac{\beta_i^2}{\delta_{i-1}}\right)\left(\frac{-\delta'_{i-1}}{\delta_{i-1}}\right).$$

To implement the calculation let

$$h_i := \beta_{i+1}^2 / \delta_i \text{ and } \rho_i := -\delta'_i / \delta_i.$$

The $h_i$ play an important role in the QD algorithm, but we will not pursue that connection here. Using the new notation we can update the ratios $\rho$ by

$$\rho_i = (1 + h_{i-1}\rho_{i-1}) / \delta_i.$$

It may be verified that $h_i\rho_i > 0$ for all i. Finally

$$\frac{-\chi'_j}{\chi_j} = sum_j := \sum_{i=1}^{j} \rho_i.$$

Here is the alternative recurrence (NEWCOR, for Newton Correction).

Set $h \leftarrow \rho \leftarrow sum \leftarrow 0$.

For $i = 1, 2, \cdots, j$ do

- $\delta \leftarrow \alpha_i - \xi - h$
- if $(\delta = 0)$ then $\delta \leftarrow \varepsilon \cdot \beta_{i+1}$
- $\rho \leftarrow (1 + h \cdot \rho) / \delta$
- $h \leftarrow \beta_{i+1}^2 / \delta$
- $sum \leftarrow sum + \rho$

The only operation in which roundoff error is significant is the calculation of $\delta_j$. Digits are lost in successive $\delta_j$ either suddenly at the last step or gradually in the last few. Although the *relative* error in $\delta$, $\rho$, and *sum* increases sharply as $i \to j$ nevertheless the error in $1/sum_j$ is tiny compared with $\xi$. This is a stable computation of the Newton correction. Table 3 shows what happens to $\delta_j$ at, near, and beyond convergence. The product $\delta_j \cdot sum_j$ should be positive (see Section 8). The computed value of $\delta_{13}$ has no correct digits. The example came from the tridiagonal obtained from the same tridiagonal which produced Figure 1.

**Table 3. Last 5 steps of NEWCOR.**

| | $j$ = 9 | | $j$ = 11 | | $j$ = 13 | |
|---|---|---|---|---|---|---|
| | $\xi$ = 10.7461941828997 | | $\xi$ = 10.7461941829034 | | $\xi$ = 10.7461941829034 | |
| $i$ | $\delta_i$ | $sum_i$ | $\delta_i$ | $sum_i$ | $\delta_i$ | $sum_i$ |
| 5 | -.1787e+00 | -.2730e+05 | | | | |
| 6 | -.1512e+00 | -.9970e+06 | | | | |
| 7 | -.1310e+00 | -.4995e+08 | -.1311e+00 | -.4994e+08 | | |
| 8 | -.1143e+00 | -.3318e+10 | -.1157e+00 | -.3278e+10 | | |
| 9 | .2590e-05 | .1103e+17 | -.1036e+00 | -.2726e+12 | -.1036e+00 | -.2726e+12 |
| 10 | | | -.9271e-01 | -.2831e+14 | -.9388e-01 | -.2796e+14 |
| 11 | | | .3963e-01 | .7603e+16 | -.9422e-01 | -.3158e+16 |
| 12 | | | | | .8669e+00 | .3516e+17 |
| 13 | | | | | -.9900e+01 | .3070e+17 |

**When to Switch from Bisection to Newton?**

It is easy to deflate the effect of known zeros from the Newton correction. Consequently there is no loss of generality in considering the calculation of one of the outermost Ritz values. Take $\vartheta_j$ to be specific.

ANALYZE T delivers an interval $S$ guaranteed to contain $\vartheta_j$. Its width provides an initial error bound. The bisection process halves the error bound at each step. In our application one Newton step costs between 2 and 3 times as much as a bisection step for large enough $j$ ($j > 20$). Because of the deflation feature we take 3 as the ratio. It follows that bisection is preferable until Newton reduces the error by a factor of 8 (= $2^3$) at each step.

Convergence is assured in our context. Let the iterates be $\xi_1, \xi_2, \xi_3, \ldots$ As $\xi_m \to \vartheta_j$,

$$\xi_{m+1} - \vartheta_j = \tfrac{1}{2}\frac{\chi''(\vartheta_j)}{\chi'(\vartheta_j)}(\xi_m - \vartheta_j)^2 + O\left[(\xi_m - \vartheta_j)^3\right]$$

A calculation reveals that

$$\chi''(\xi)/\chi'(\xi) = \sum(\xi - \vartheta_i)^{-1} - \frac{\sum(\xi - \vartheta_i)^{-2}}{\sum(\xi - \vartheta_i)^{-1}}$$

where each sum is from 1 to $j$. Let $\zeta \to \vartheta_j$ to find that

$$\tfrac{1}{2}\frac{\chi''(\vartheta_j)}{\chi'(\vartheta_j)} = \sum_{i=1}^{j-1}(\vartheta_j - \vartheta_i)^{-1}.$$

Now drop higher order terms to obtain

$$\frac{\xi_{m+1} - \vartheta_j}{\xi_m - \vartheta_j} \approx (\xi_m - \vartheta_j)\sum_{i=1}^{j-1}(\vartheta_j - \vartheta_i)^{-1}.$$

In some instances the dominant terms in the sum of reciprocals will be available. Let us consider the opposite extreme, when only $spread := \vartheta_j - \vartheta_1$ is known. If the zeros are uniformly spaced then

$$\sum_{i=1}^{j-1}(\vartheta_j - \vartheta_i)^{-1} = (1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{j-1})/gap$$
$$\approx (j-1)[\gamma + \ln(j-1)]/spread\ ,\ \text{as}\ j \to \infty$$

Here $\gamma$ is Euler's constant (=.577...).

If $\vartheta_2, \ldots, \vartheta_{j-1}$ are all bunched at the midpoint then

$$\sum_{i=1}^{j-1}(\vartheta_j - \vartheta_i)^{-1} = (2j-3)/spread\ .$$

In practice, when Lanczos is used with a shifted inverted operator, there is a tendency for many interior $\vartheta$'s to cluster round $0 \in (\vartheta_1, \vartheta_j)$. This situation may be modeled by

$$\sum_{i=1}^{j-1}(\vartheta_j - \vartheta_i)^{-1} \approx \left[\frac{1}{\vartheta_j - \vartheta_{j-1}} + \frac{j-3}{\vartheta_j - 0} + \frac{1}{spread}\right].$$

To use these results $\xi_m - \vartheta_j$ must be replaced by the width of the smallest interval currently known to contain $\vartheta_j$. We use the uniform spacing assumption and test the assertion

$$width \le spread/[8(d-1)\ln(d-1)]$$

where $d = j$ - {no. of known $\vartheta$'s}.

## 8. Evaluation of s(j)$^2$

Let us drop the subscript $i$ and consider a typical Ritz value $\vartheta$ with $\mathbf{T}_j \mathbf{s} = \mathbf{s}\vartheta$, $\|\mathbf{s}\| = 1$. Section 4 reported that $s(j)^2 \chi'_j(\vartheta) = \chi_{j-1}(\vartheta)$, so

$$\begin{aligned} -s(j)^{-2} &= \frac{\chi'_j(\vartheta)}{\chi_{j-1}(\vartheta)} \\ &= \left[\frac{\chi_j}{\chi_{j-1}}\right]' + \left[\frac{\chi_j}{\chi_{j-1}}\right]\left[\frac{\chi'_{j-1}}{\chi_{j-1}}\right] \\ &= \delta'_j(\vartheta), \text{ since } \chi_j(\vartheta) = 0. \end{aligned}$$

Note also that if $\chi_j(\zeta) \neq 0$ then

$$\left[\frac{\chi'_j(\zeta)}{\chi_{j-1}(\zeta)}\right] = \left[\frac{\chi'_j(\zeta)}{\chi_j(\zeta)}\right]\left[\frac{\chi_j(\zeta)}{\chi_{j-1}(\zeta)}\right] = -sum_j(\zeta)\cdot\delta_j(\zeta)$$

where *sum* is evaluated in the recurrence for the Newton correction given in the previous section.

Given below is a list of ways to approximate $s(j)^2$ at little cost.

1. If $\zeta = \vartheta + O(\varepsilon\|\mathbf{A}\|)$ then

$$s(j)^{-2} = -\delta'_j(\zeta) = 1 + h_{j-1}(\zeta)\cdot\rho_{j-1}(\zeta) > 1$$

is available free from the Newton correction. However it is somewhat wasteful to evaluate the recurrence at a point so close to $\vartheta$.

2. If $\zeta = \vartheta + O(\varepsilon\|\mathbf{A}\|)$ then

$$s(j)^{-2} = sum_j(\zeta)\cdot\delta_j(\zeta)$$

is available from the Newton correction. Both $sum_j$ and $\delta_j$ will have high relative error when $\delta_{j-1}(\zeta)$ is tiny, as must happen when $\vartheta$ stagnates. See Table 3.

3. From the three formulae involving $s(1)$ and $s(j)$ given in Section 4 one can derive a fourth one

$$s(j)^{-2} = -\chi'_j(\vartheta)\,\chi_{2,j}(\vartheta)/(\beta_2\cdots\beta_j)^2 .$$

The Newton correction recurrence may be run **backwards** to yield $\chi'_j$ and $\chi_{2,j}$. A little manipulation reveals that

$$-s(j)^{-2} = (\bar{\delta}'_1)\prod_{2}^{j}\left(\frac{\bar{\delta}_i^2}{\beta_i^2}\right)$$

If $\xi$ is the final point at which the backward recurrence was evaluated then

$$\bar{\delta}'_1(\xi) = 1 + \bar{h}_1(\xi)\cdot\bar{\rho}_1(\xi) > 1$$

is available at no cost. The other factor must be formed. A stable way to do this is

$\pi := 1 \; ; h := 0$
for $i := j$ down to 2 do
  $u := \alpha_j - \xi - h$
  $\pi := \pi \cdot u^2 / \beta_i^2$
  if $u = 0$ then $u := \varepsilon(\beta_i^2 + \alpha_i^2)$
  $h := \beta_i^2 / u$

The cost is $2j$ divisions and $2j$ multiplications. If $\xi = \vartheta + O(\varepsilon \|\mathbf{A}\|)$ then this is a more accurate though more expensive procedure than No. 2

**Consequences of using Newton's method**

We deflate stabilized Ritz values from $\mathbf{T}$ and this device forestalls the production of clusters of close Ritz values. Thus $\chi''/\chi'$ varies gently in the neighborhood of each Ritz value and we stop Newton's iteration as soon as the correction $c$ is less than *tol*, confident that one more step would produce a $c = O(\varepsilon \|\mathbf{T}\|)$. We remark in passing that stabilized Ritz values will be refined later in a Lanczos run and so an error in the last few places is of no consequence. This policy presents the challenge of approximating $s(j)^2$ correct to at least one decimal place despite the fact that the Newton recurrence will not have been evaluated at our latest approximation $\zeta$, but at the previous $\xi$.

4. Let $\zeta = \xi + c$. Note that

$$-\delta'_j(x) = 1 + \sum_{i=1}^{j-1} \frac{\beta_{ji}^2}{(\vartheta_i^{(j-1)} - x)^2}$$

The only unbounded term as $x \to \vartheta$ is $\beta_j^2 s(j)^2/(\vartheta - x)^2$, the rest varies slowly. Consequently the correction to be applied to $-\delta'(\xi)$ is

approximately

$$\beta_j^2 s(j)^2\left\{\frac{1}{(\vartheta-\zeta)^2}-\frac{1}{(\vartheta-\xi)^2}\right\}$$
$$=\beta_j^2 s(j)^2\frac{2c(\vartheta-\xi)-c^2}{(\vartheta-\zeta)^2(\vartheta-\xi)^2}$$

5. A more orthodox way to correct $-\delta'_j(\xi)$ is to use Taylor series. However it is preferable to apply this technique to functions which resemble polynomials. So we consider $\varphi(x) := (\vartheta - x)\delta_j(x)$.on the tiny interval $(\vartheta,\xi)$ of interest to us. Let $\vartheta := \vartheta_i^{(j-1)}$. Then, with $\zeta = \xi + c$ we have

$$\varphi'(\zeta) = \varphi'(\xi) + \varphi''(\xi)c + O(c^2)$$

A little calculation reveals that

$$\varphi''(\xi) = (\vartheta-\xi)\delta''_j(\xi) - 2\delta'_j(\xi) = 2 + 2\sum_{\substack{k=1\\k\neq i}}^{j-1}\frac{\beta_{jk}^2(\vartheta_k-\vartheta)}{(\vartheta_k-\xi)^3} \qquad (*)$$

because the dominant parts $2\frac{\beta^2}{(\vartheta-\xi)^2}$ cancel each other. Rearranging terms in the expression for $\varphi'$ yields

$$\delta'_j(\vartheta) = \delta'_j(\xi) + \frac{c}{\vartheta-\xi}\left[(\delta'_j(\xi) + \frac{\delta_j(\xi)}{c}) + \Psi\right]$$

where $\Psi$ is the best available approximation to (*). Although it appears complicated this process does not require $j$ arithmetic operations.

6. Methods based on QR and QL.

Let $\mathbf{T}_j\mathbf{s} = \mathbf{s}\vartheta$. In exact arithmetic the QR transformation with shift $\vartheta$ will deflate $\mathbf{T}_j$. Then $s(j)$ is the cosine of the last rotation angle used in the transformation. The transform may be invoked without bothering to store the elements of the new matrix. The most compact version of the algorithm, given in [SEP,p. 169], allows the computation of $s(j)$ after $3j$ divisions and $3j$ multiplications.

This technique is not reliable in practice because sometimes (for example, when a Ritz value stabilizes) the final rotation is poorly determined by the

initial data. This phenomenon corresponds to the fact that deflation does not always occur in one QR transformation even with an eigenvalue correct to working precision.

Less well known is the fact that $s(j)$ is the product of all the sines used in the QL transform of $\mathbf{T}_j$ with shift $\vartheta$. See [Chen,1983] for instance. This version is very stable and yields $s(j)^2$ after $3j$ divisins and $4j$ multiplications.

7. Givens recurrence.

In our application this much maligned recurrence for computing an eigenvector is very accurate because the $s(1)$ are substantial if not actually maximal components of the $\mathbf{s}$. The recurrence solves the equation

$$(\mathbf{T}_j - \xi\mathbf{I})\mathbf{v} = \mathbf{e}_1\mu, \qquad \|\mathbf{v}\| = 1.$$

If $\xi = \vartheta$ to working accuracy then $\|\mathbf{s} - \mathbf{v}\| = O(\varepsilon)$. There is no need to store the elements of $\mathbf{v}$. The cost is $j$ divisions and $3j$ multiplications. The least attractive feature of Givens for us is that it requires knowledge of the $\beta_i$. All the other techniques utilize $\beta_i^2$, the quantities we actually provide for Analyze T.

In Table 4 a comparison is made of the methods described above on some typical examples.

| $j$ | $\vartheta^{(j)}$ | Bottom Element of Eigen Vector, $s(j)$ | | | | |
|---|---|---|---|---|---|---|
| | | method 1 or 2 | method 3 | QR | QL | Givens |
| 2 | 218.2753378667163 | 0.675e+00 | 0.675e+00 | 0.675e+00 | 0.675e+00 | 0.675e+00 |
| 3 | 233.9326136824040 | 0.435e+00 | 0.435e+00 | 0.435e+00 | 0.435e+00 | 0.435e+00 |
| 4 | 238.2264773067190 | 0.212e+00 | 0.212e+00 | 0.212e+00 | 0.212e+00 | 0.212e+00 |
| 5 | 240.9310281046789 | 0.167e+00 | 0.167e+00 | 0.167e+00 | 0.167e+00 | 0.167e+00 |
| 6 | 242.5758180873734 | 0.165e+00 | 0.165e+00 | 0.165e+00 | 0.165e+00 | 0.165e+00 |
| 7 | 243.3205720927529 | 0.969e-01 | 0.969e-01 | 0.969e-01 | 0.969e-01 | 0.969e-01 |
| 8 | 243.9045963222352 | 0.821e-01 | 0.821e-01 | 0.821e-01 | 0.821e-01 | 0.821e-01 |
| 9 | 244.3596662340472 | 0.910e-01 | 0.910e-01 | 0.910e-01 | 0.910e-01 | 0.910e-01 |
| 10 | 244.5415860111110 | 0.448e-01 | 0.448e-01 | 0.448e-01 | 0.448e-01 | 0.448e-01 |
| 11 | 244.5848222804917 | 0.167e-01 | 0.167e-01 | 0.167e-01 | 0.167e-01 | 0.167e-01 |
| 12 | 244.5850377523666 | 0.103e-02 | 0.103e-02 | 0.103e-02 | 0.103e-02 | 0.103e-02 |
| 13 | 244.5850426917980 | 0.160e-03 | 0.160e-03 | 0.160e-03 | 0.160e-03 | 0.160e-03 |
| 14 | 244.5850427056620 | 0.784e-05 | 0.784e-05 | 0.784e-05 | 0.784e-05 | 0.784e-05 |
| 15 | 244.5850427056857 | 0.317e-06 | 0.317e-06 | 0.318e-06 | 0.317e-06 | 0.317e-06 |
| 16 | 244.5850427056857 | 0.345e-08 | 0.852e-09 | 0.108e-07 | 0.852e-09 | 0.852e-09 |
| 17 | 244.5850427056857 | 0.334e-08 | 0.238e-11 | 0.355e-05 | 0.238e-11 | 0.238e-11 |
| 18 | 244.5850427056857 | 0.334e-08 | 0.792e-14 | 0.107e-02 | 0.792e-14 | 0.792e-14 |
| 19 | 244.5850427056857 | 0.334e-08 | 0.227e-16 | 0.350e+00 | 0.227e-16 | 0.227e-16 |
| 20 | 244.5850427056857 | 0.101e-07 | 0.308e-17 | 0.999e+00 | 0.308e-17 | 0.308e-17 |
| 21 | 244.5850427056857 | 0.421e-08 | 0.862e-18 | 0.526e+00 | 0.862e-18 | 0.862e-18 |
| 22 | 244.5850427056857 | 0.116e-07 | 0.204e-17 | 0.860e+00 | 0.204e-17 | 0.204e-17 |

**Table 4.** Comparison of Different Methods for Computing $s(j)$. Note the change after step 16.

**9. Explicit Deflation**

There is a useful technique which permits some important simplifications in Analyze T at the extra cost of 2 arrays of length lanmax (=the maximum number of Lanczos steps permitted). The simplification is that Analyze T may assume that **T** has no clusters of very close Ritz values. The technique is to remove fully stabilized Ritz values by using the QR algorithm to deflate $\mathbf{T}_j$. The extra arrays are to preserve the **T** of the Lanczos algorithm for computation of the Ritz vectors.

Let $\vartheta$ be a Ritz value which has fully stabilized before step $j$. In other words, the $j^{\text{th}}$ element of $\vartheta$'s normalized eigenvector **s**, satisfies $|s(j)| < \sqrt{\varepsilon}$. At the end of step $j$, apply the QR algorithm with fixed shift $\vartheta$ and consider the situation at step $j+1$. Assume for the moment that only one step of the QR algorithm is needed to cause $\vartheta$ to appear in position $(j,j)$ and to have the $(j,j-1)$ element well below the threshold $\sqrt{\varepsilon}\,\beta_j$. A little notation is needed. Let

$$\mathbf{T}_j - \vartheta I_j = \bar{\mathbf{Q}}\bar{\mathbf{R}}, \qquad \bar{\mathbf{T}}_j := \bar{\mathbf{Q}}^* \mathbf{T}_j \bar{\mathbf{Q}} .$$

Partition $\bar{\mathbf{T}}$ as

$$\bar{\mathbf{T}}_j = \left[\begin{array}{ccc|c} & & & 0 \\ & \bar{\mathbf{T}}_{j-1} & & \vdots \\ & & & 0 \\ & & & \eta \\ \hline 0 & \cdots & \eta & \vartheta \end{array}\right] \qquad |\eta| < \sqrt{\varepsilon}\cdot\beta_j .$$

The success of deflation implies that $\bar{\mathbf{q}}_j := \bar{\mathbf{Q}}\mathbf{e}_j$ satisfies $\mathbf{T}_j\bar{\mathbf{q}}_j = \bar{\mathbf{q}}_j\vartheta + \bar{\mathbf{q}}_{j-1}\eta$. Thus $\sin\angle(\bar{\mathbf{q}}_j, \mathbf{s}) < \eta / gap(\vartheta)$, where $\text{gap}(\vartheta) = min\,|\lambda - \vartheta|$ over eigenvalues $\lambda$ of $\bar{\mathbf{T}}_{j-1}$.

Now consider the effect of the similarity transformation on $\mathbf{T}_{j+1}$:

$$\begin{bmatrix} \bar{\mathbf{Q}} & \vdots & \\ \cdots & \cdots & \\ & \vdots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{T}_j & \vdots & \\ \cdots & \cdots & \beta_{j+1} \\ \beta_{j+1} & \vdots & \alpha_{j+1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{Q}} & \vdots & \\ \cdots & \cdots & \\ & \vdots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} & & \vdots & & \\ & \bar{\mathbf{T}}_{j-1} & \vdots & & \\ & & \vdots & \eta & \beta \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & \eta & \vartheta & \sigma \\ & & \beta & \sigma & \alpha_{j+1} \end{bmatrix}.$$

Let $\varphi$ denote the last rotation angle in the QR sweep, then

$$\sigma = \beta_{j+1}\mathbf{e}_j^*\bar{\mathbf{q}}_j = \beta_{j+1}\cos\varphi \approx \beta_{j+1}|s(j)|,$$
$$\beta = \beta_{j+1}\mathbf{e}_j^*\bar{\mathbf{q}}_{j-1} = \beta_{j+1}\sin\varphi \approx \beta_{j+1}.$$

The last approximation follows from the fact that $\bar{\mathbf{Q}}$ is in upper Hessenberg form. Therefore $\beta^2 + \sigma^2 = \beta_{j+1}^2$ and

$$\beta = \beta_{j+1}(1 - s(j)^2)^{1/2} = \beta_{j+1} \qquad \text{(to working accuracy)}.$$

If the QR transformation were executed in exact arithmetic and if $\vartheta$ were an exact eigenvalue of $\mathbf{T}_j$ then $\eta = 0$ and the magnitude of $\sigma$ could be controlled by choosing the right value of $j$ at which to deflate, namely after $\vartheta$ first stabilizes, but before any second copies of $\vartheta$ appear. When $\eta$ and $\sigma$ are negligible we may simply delete row and column $j$ from the transform of $\mathbf{T}_{j+1}$ and work thereafter with a smaller tridiagonal matrix.

$$\hat{\mathbf{T}}_j := \left[\begin{array}{c|c} \overline{\mathbf{T}}_{j-1} & \begin{matrix} \beta_{j+1} \\ - \end{matrix} \\ \hline \begin{matrix} \multicolumn{2}{c}{\beta_{j+1}} \end{matrix} & \alpha_{j+1} \end{array}\right]$$

The fact that $\alpha_{j+1}$ is unknown at the end of step $j$ is immaterial.

Some information is discarded when $\eta$ and $\sigma$ are neglected, but it is only necessary to preserve the integrity of the Ritz values, not the eigenvectors of $\mathbf{T}_j$. When eigenvectors of the operator **A** are wanted then it is necessary to keep a copy of $\mathbf{T}_{j+1}$ for their computation.

As students of the QR algorithm know, in finite arithmetic it is likely that 2 steps of the QR algorithm will be needed to make $\eta$ negligible. In such cases **Q** is no longer in upper Hessenberg form. Consequently

$$\hat{\mathbf{T}}_j = \left[\begin{array}{c|c} \overline{\mathbf{T}}_{j-1} & \begin{matrix} \delta \\ \gamma \end{matrix} \\ \hline \begin{matrix} \multicolumn{2}{c}{\delta \quad \gamma} \end{matrix} & \alpha_{j+1} \end{array}\right]$$

where $\gamma^2 + \delta^2 = \beta_{j+1}^2$.

Rather than performing 2 QR transformations one can simply deduce the correct rotations in QR from the eigenvector **s** and force the QR transform to use them.

Our subprogram Analyze T can work on $\hat{\mathbf{T}}_j$ happy in the knowledge that $\vartheta$ is not one of its eigenvalues. If at some later step of the Lanczos process a second copy of $\vartheta$ appears then it will be as a simple eigenvalue of **T**

There is an alternative to deflation for protecting NEWCOR from difficult

situations. If a second copy of $\vartheta$ stabilizes at step $k$, then it suffices to compute it as a simple eigenvalue of a submatrix $\mathbf{T}_{m,k}$ of $\mathbf{T}_k$. However, the choice of m is not a trivial matter. It must satisfy $1 < m < j$, but the best choice of $m$ depends on the eigenvector of $\mathbf{T}_j$ belonging to $\vartheta$. It is feasible to try $m = 2$ (i.e. $m$ = multiplicity of $\vartheta$ in $\mathbf{T}_k$) and then increase $m$ if any difficulties arise. More work is needed on this topic. There may be a simple, safe formula for $m$. Until that is discovered, we recommend explicit deflation. The extra storage requirement is for the Lanczos process, not for Analyze T.

**10. Profile of Lanczos Runs**

The tables given below attempt to record the important incidents as the Lanczos algorithm builds up a basis for a Krylov subspace and updates the projection of the given linear operator on that subspace. What is wanted is information on the quality of the Ritz approximations ($\vartheta_i$ , $\mathbf{Q}_j \mathbf{s}_i$) and the cost of obtaining them. The quality is given by the $bj(i)$ and the cost of this information is given below.

At each step the table shows.

1) The number of Ritz values updated (col. 2), and the average cost (col. 3), the maximum cost (col. 4) and the minimum cost (col.5) of updating the Ritz values. A unit of cost is taken to be $\bar{j}$ operation, where $\bar{j}$ is the size of the deflated tridiagonal matrix.

2) The number of Ritz values appended (col. 6) and the average cost of appending these Ritz values (col. 7).

3) The cost of this monitoring as a fraction of the cost of a Lanczos step (col. 8).

4) Column 9 contains a *cumulative* tally of the number of stabilized Ritz values (eigenvalues).

The first profile is obtained from a matrix of size 100 with an average half bandwidth of 23. This matrix arises from a finite element model of a multistory building discretized using truss elements. For this run the cost of Analyze T ranged from 3% to 23% of the cost of a Lanczos step. The second profile is obtained form a larger building frame example ($n$ = 468 and average half bandwidth = 120) that is described in [Nour-Omid, 1983] . The cost of Analyze T as a fraction of cost of a Lanczos step was much less ( ranging form 0.2% to 5%), indicating that for very large examples this cost will be neglegible. This run is

memorable because ten Ritz values stabilized from step 69 to step 70; an unusual occurrence. Nevertheless, the effort to compute all ten values was less than 5% of a Lanczos step. In other words, this was a very cost-effective step in the process. The costs mentioned above include arithemetic operations but exclude fetch and store operations.

Our experience with these profiles is limited but we plan to use them routinely and hope that they will appeal to all who are interested in a detailed comprehension of the Lanczos algorithm.

| | updates | | | | appends | | cost | no. of |
|---|---|---|---|---|---|---|---|---|
| step $j$ | no. of items | average cost | max. cost | min. cost | no. of items | average cost | ratio $\frac{\text{AnalyzeT}}{\text{lanstep}}$ | conv. eigs. |
| 3 | 2 | 15 | 16 | 14 | 0 | 0 | 0.0308 | 0 |
| 4 | 2 | 10 | 12 | 9 | 2 | 19 | 0.0795 | 0 |
| 5 | 4 | 11 | 15 | 6 | 0 | 0 | 0.0753 | 0 |
| 6 | 4 | 10 | 15 | 3 | 0 | 0 | 0.0685 | 1 |
| 7 | 3 | 10 | 13 | 6 | 1 | 17 | 0.0966 | 1 |
| 8 | 4 | 9 | 14 | 3 | 1 | 17 | 0.1089 | 2 |
| 9 | 4 | 10 | 15 | 3 | 0 | 0 | 0.0959 | 2 |
| 10 | 4 | 10 | 16 | 3 | 0 | 0 | 0.0959 | 3 |
| 11 | 3 | 11 | 16 | 9 | 0 | 0 | 0.0904 | 3 |
| 12 | 3 | 10 | 13 | 6 | 0 | 0 | 0.0925 | 3 |
| 13 | 2 | 8 | 11 | 6 | 1 | 15 | 0.1062 | 3 |
| 14 | 3 | 8 | 12 | 3 | 0 | 0 | 0.0904 | 3 |
| 15 | 3 | 8 | 13 | 3 | 0 | 0 | 0.0986 | 3 |
| 16 | 3 | 9 | 16 | 3 | 0 | 0 | 0.1110 | 4 |
| 17 | 2 | 11 | 16 | 6 | 0 | 0 | 0.0979 | 4 |
| 18 | 2 | 11 | 16 | 6 | 1 | 18 | 0.1918 | 4 |
| 19 | 3 | 8 | 13 | 3 | 1 | 15 | 0.1870 | 5 |
| 20 | 3 | 7 | 13 | 3 | 1 | 16 | 0.1901 | 5 |
| 21 | 4 | 7 | 16 | 3 | 0 | 0 | 0.1438 | 6 |
| 22 | 3 | 8 | 15 | 3 | 1 | 15 | 0.2003 | 7 |
| 23 | 3 | 7 | 13 | 3 | 0 | 0 | 0.1151 | 7 |
| 24 | 3 | 8 | 16 | 3 | 1 | 15 | 0.2271 | 7 |
| 25 | 4 | 7 | 13 | 3 | 0 | 0 | 0.1630 | 8 |

Table 5. profile 1 ( $n$ = 100, average half bandwidth = 23)

| step $j$ | updates | | | | appends | | cost ratio $\frac{\text{AnalyzeT}}{\text{lanstep}}$ | no. of conv. eigs. |
|---|---|---|---|---|---|---|---|---|
| | no. of items | average cost | max. cost | min. cost | no. of items | average cost | | |
| 3 | 2 | 18 | 19 | 18 | 0 | 0 | 0.0018 | 0 |
| 4 | 2 | 16 | 17 | 16 | 2 | 19 | 0.0047 | 0 |
| 5 | 4 | 15 | 18 | 13 | 0 | 0 | 0.0051 | 0 |
| 6 | 4 | 15 | 17 | 12 | 0 | 0 | 0.0061 | 0 |
| 7 | 4 | 13 | 16 | 9 | 0 | 0 | 0.0062 | 0 |
| 8 | 4 | 13 | 17 | 9 | 0 | 0 | 0.0070 | 0 |
| 9 | 4 | 13 | 18 | 6 | 0 | 0 | 0.0079 | 0 |
| 10 | 2 | 9 | 12 | 6 | 0 | 0 | 0.0030 | 0 |
| 11 | 2 | 7 | 9 | 6 | 0 | 0 | 0.0026 | 0 |
| 12 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0024 | 0 |
| 13 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0026 | 0 |
| 14 | 2 | 4 | 6 | 3 | 1 | 19 | 0.0064 | 0 |
| 15 | 3 | 6 | 13 | 3 | 1 | 18 | 0.0091 | 0 |
| 16 | 4 | 5 | 12 | 3 | 2 | 17 | 0.0128 | 2 |
| 17 | 4 | 6 | 9 | 3 | 0 | 0 | 0.0061 | 2 |
| 18 | 4 | 6 | 9 | 3 | 0 | 0 | 0.0057 | 4 |
| 19 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0030 | 4 |
| 20 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0032 | 4 |
| 21 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0035 | 4 |
| 22 | 2 | 4 | 6 | 3 | 1 | 15 | 0.0070 | 4 |
| 23 | 3 | 3 | 3 | 3 | 1 | 18 | 0.0087 | 4 |
| 24 | 4 | 3 | 6 | 3 | 2 | 18 | 0.0154 | 5 |
| 25 | 5 | 4 | 9 | 3 | 0 | 0 | 0.0068 | 5 |
| 26 | 5 | 3 | 6 | 3 | 0 | 0 | 0.0048 | 7 |
| 27 | 3 | 4 | 6 | 3 | 1 | 18 | 0.0096 | 8 |
| 28 | 3 | 6 | 9 | 3 | 0 | 0 | 0.0061 | 8 |
| 29 | 3 | 5 | 9 | 3 | 1 | 18 | 0.0117 | 8 |
| 30 | 4 | 5 | 9 | 3 | 0 | 0 | 0.0074 | 8 |
| 31 | 4 | 4 | 9 | 3 | 1 | 19 | 0.0130 | 9 |
| 32 | 4 | 5 | 9 | 3 | 0 | 0 | 0.0074 | 10 |
| 33 | 3 | 6 | 9 | 3 | 0 | 0 | 0.0070 | 10 |
| 34 | 3 | 4 | 6 | 3 | 1 | 15 | 0.0105 | 11 |
| 35 | 3 | 6 | 9 | 3 | 0 | 0 | 0.0073 | 11 |
| 36 | 3 | 5 | 6 | 3 | 0 | 0 | 0.0061 | 12 |
| 37 | 2 | 6 | 6 | 6 | 0 | 0 | 0.0051 | 12 |
| 38 | 2 | 6 | 6 | 6 | 3 | 18 | 0.0290 | 12 |
| 39 | 5 | 4 | 9 | 3 | 2 | 16 | 0.0238 | 12 |
| 40 | 7 | 3 | 6 | 3 | 1 | 16 | 0.0163 | 14 |
| 41 | 6 | 4 | 6 | 3 | 0 | 0 | 0.0102 | 16 |

**Table 6. profile 2 ( $n$ = 468, average half bandwidth = 120)**

| step $j$ | updates | | | | appends | | cost ratio $\frac{\text{AnalyzeT}}{\text{lanstep}}$ | no. of conv. eigs. |
|---|---|---|---|---|---|---|---|---|
| | no. of items | average cost | max. cost | min. cost | no. of items | average cost | | |
| 42 | 4 | 4 | 6 | 3 | 2 | 17 | 0.0211 | 17 |
| 43 | 5 | 4 | 9 | 3 | 0 | 0 | 0.0085 | 18 |
| 44 | 4 | 4 | 9 | 3 | 1 | 15 | 0.0136 | 18 |
| 45 | 5 | 4 | 9 | 3 | 0 | 0 | 0.0091 | 18 |
| 46 | 5 | 4 | 6 | 3 | 0 | 0 | 0.0095 | 18 |
| 47 | 5 | 4 | 6 | 3 | 0 | 0 | 0.0095 | 19 |
| 48 | 4 | 4 | 6 | 3 | 0 | 0 | 0.0076 | 20 |
| 49 | 3 | 4 | 6 | 3 | 1 | 16 | 0.0137 | 20 |
| 50 | 3 | 5 | 6 | 3 | 1 | 18 | 0.0162 | 21 |
| 51 | 4 | 4 | 6 | 3 | 0 | 0 | 0.0081 | 21 |
| 52 | 4 | 3 | 6 | 3 | 0 | 0 | 0.0061 | 22 |
| 53 | 3 | 4 | 6 | 3 | 2 | 15 | 0.0220 | 22 |
| 54 | 5 | 4 | 6 | 3 | 0 | 0 | 0.0105 | 23 |
| 55 | 4 | 4 | 6 | 3 | 0 | 0 | 0.0087 | 23 |
| 56 | 4 | 4 | 6 | 3 | 2 | 17 | 0.0271 | 24 |
| 57 | 5 | 4 | 6 | 3 | 0 | 0 | 0.0112 | 24 |
| 58 | 5 | 3 | 6 | 3 | 4 | 16 | 0.0428 | 26 |
| 59 | 7 | 3 | 6 | 3 | 0 | 0 | 0.0114 | 27 |
| 60 | 6 | 4 | 6 | 3 | 1 | 16 | 0.0217 | 28 |
| 61 | 6 | 4 | 6 | 3 | 1 | 19 | 0.0233 | 29 |
| 62 | 6 | 5 | 13 | 3 | 0 | 0 | 0.0167 | 29 |
| 63 | 6 | 5 | 15 | 3 | 1 | 18 | 0.0252 | 32 |
| 64 | 4 | 6 | 12 | 3 | 0 | 0 | 0.0126 | 33 |
| 65 | 3 | 6 | 9 | 3 | 1 | 18 | 0.0195 | 33 |
| 66 | 4 | 6 | 12 | 3 | 0 | 0 | 0.0134 | 33 |
| 67 | 4 | 6 | 12 | 3 | 4 | 17 | 0.0529 | 33 |
| 68 | 8 | 4 | 12 | 3 | 2 | 17 | 0.0357 | 36 |
| 69 | 7 | 3 | 9 | 3 | 2 | 16 | 0.0233 | 43 |
| 70 | 2 | 9 | 9 | 9 | 5 | 16 | 0.0398 | 46 |
| 71 | 4 | 6 | 9 | 3 | 0 | 0 | 0.0097 | 47 |
| 72 | 3 | 5 | 6 | 3 | 0 | 0 | 0.0063 | 47 |
| 73 | 3 | 5 | 6 | 3 | 0 | 0 | 0.0066 | 47 |
| 74 | 3 | 5 | 6 | 3 | 1 | 12 | 0.0123 | 47 |
| 75 | 4 | 6 | 13 | 3 | 3 | 15 | 0.0315 | 48 |
| 76 | 6 | 6 | 15 | 3 | 0 | 0 | 0.0171 | 48 |
| 77 | 6 | 7 | 15 | 3 | 0 | 0 | 0.0199 | 49 |
| 78 | 5 | 7 | 15 | 3 | 0 | 0 | 0.0166 | 50 |
| 79 | 4 | 8 | 14 | 3 | 0 | 0 | 0.0152 | 51 |
| 80 | 3 | 10 | 16 | 3 | 0 | 0 | 0.0147 | 51 |

**Table 6. profile 2 ( $n$ = 468, average half bandwidth = 120)**

# APPENDIX I. Listing of the Analyze T Routine.

```
      subroutine analzt(j,alf,bet2,that,bj,nbd,spread,eps)
      implicit double precision(a-h,o-z)
      dimension alf(1),bet2(1),that(8),bj(8),nbd(2)
      logical newrtz,indxok
      data one / 1.0d0 /,zero / 0.0d0/
c
c     j           order of the tridiagonal T.
c     alf(.)      diagonal of T.
c     bet2(.)     squares of the offdiagonal terms, bet2(1) = 0.
c     that(.)     exterior eigenvalues of T, nearly converged
c                 Ritzvalues. that(1)=leftmost., that(8)=rightmost.
c     bj(.)       error bound on that(.)
c                 bj(1) is set to -1 if that(1) disappears.
c     nbd(.)      contains l and r in the text.
c     spread      that(8) - that(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),
c                 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
c                 new that(i).
c
      if ( j .le. 1 ) return
      if ( j .eq. 2 ) then
        that(1) = ( alf(1) + alf(2) - dsqrt(4.*bet2(2) +
     1          ( alf(1) - alf(2))**2 ))/2.
        that(8) = alf(1) + alf(2) - that(1)
        bj(1) = one/(one + bet2(2)/(that(1) - alf(1))**2)
        bj(8) = one/(one + bet2(2)/(that(8) - alf(1))**2)
          nbd(1) = 1
          nbd(2) = 8
        spread = that(8) - that(1)
        return
      end if
c
c     begin phase 1.
c     loop for left end, then right
c
      do 4 ip = 1,2
        inc = 3 - 2*ip
        is = 7*ip - 6
        i = is
        newrtz = .false.
        start = (that(i) + alf(j) - inc*dsqrt(bet2(j)*4. +
     1          (alf(j) - that(i))**2))/2.
        probe = that(i) - inc*bj(i)
        indxok = numles(alf,bet2,probe,j,inc,eps) .eq. 0
        do 2 idummy =1,6
          if (i - nbd(ip) .eq. inc) go to 2
c
c         examine i th subinterval
c
          if (indxok) then
            if (newrtz) then
              start = that(i)
              that(i) = start + inc*dmin1(b**2/
     1                  dabs(start - that(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=that(i)+0.25*inc*(that(nbd(2))-that(nbd(1)))/
     1                  (j - nbd(1) + nbd(2) - 8 )
            else
              probe = that(i+inc) - inc*bj(i+inc)
            end if
            if (int(dsign(one,probe - that(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               that(i) disappears
c
                bj(i) = -one
              else
c
c                 record indxok for next loop. use refined bounds.
c
                if ( .not. newrtz) then
                  b = bj(i)
                  indxok = (k .le. iabs(i - is + inc))
                  bnd = dmin1(b**2/dabs(probe-that(i)) , b)
                  if (indxok.and.bnd.lt.dabs(that(i)-start)) then
                    start = that(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(that,i,nbd(ip),-inc,probe)
            call move1(bj,i,nbd(ip),-inc,one)
            newrtz = .true.
            indxok = .true.
          end if
c
          if ( bj(i) .gt. zero ) then
c
c           find new that(i) and bj(i)
c
            call findth(alf,bet2,start,that,bj,nbd,inc,i,j)
          end if
c
          if ( bj(i) .lt. 0 ) then
c
c             that(i) disappears
c
            call move1(that,nbd(ip),i,inc,zero)
            call move1(bj,nbd(ip),i,inc,zero)
            nbd(ip) = nbd(ip) - inc
            newrtz = .false.
            indxok = .true.
            i = i - inc
          end if
          i = i + inc
2       continue
4     continue
      spread = that(8) - that(1)
      return
      end
```

```
      subroutine phase2(j,nsig,alf,bet2,eig,info,that,bj,nbd,rnm2,
     1                  tol,eps)
      implicit double precision(a-h,o-z)
      dimension alf(1),bet2(1),eig(1),info(1),that(6),bj(6),nbd(2)
      logical append
      data one / 1.0d0 /,zero / 0.0d0/
c
c     append more Ritzvalues and check for converged Ritzvalues.
c
c     eig(.)       holds stabilized Ritz values.
c     info(.)      holds information concerning eig(.) for use else where.
c     rnm2         beta(j+1)**2
c     nsig         no. of stabilized Ritzvalues.
c     tol          criterion for stabilization of that(i)
c                  =2*spread*sqrt(eps)
c     w=1.26       window factor
c
      if ( j .le. 1 ) return
      w = 1.26*tol
      do 6 ip = 1,2
        inc = 3 - 2*ip
        is  = 7*ip - 6
        i = is
        do 6 idummy = 1,j
          if ( (i-nbd(ip))*inc .gt. 0 ) go to 6
             bj(i) = dsqrt(rnm2*bj(i))
          nrem = j - nbd(1) + nbd(2) - 9
          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 = that(i)
            probe = inc*(that(nbd(2)) - that(nbd(1)))/(nrem+1)
          end if
          if (bj(i) .lt. tol) then
c
c           remove stablized Ritz values from that into eig
c
            nsig = nsig + 1
            eig(nsig)  = that(i)
            info(nsig) = 0
            call move1(that,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
c
c           find a nonempty subinterval
c
            t = start + probe
            nbd(ip) = nbd(ip) + inc
            ik = iabs(is - nbd(ip))
            do 2 idum = 1,j
              if (mumles(alf,bet2,t,j,inc,eps) .ne. ik) go to 4
              t = t + probe
2           continue
4           that(nbd(ip)) = t
            start = t - probe
            call fmdth(alf,bet2,start,that,bj,nbd,inc,nbd(ip),j)
               bj(i) = dsqrt(rnm2*bj(i))
          end if
c
c         delete end Ritz values if appropriate.
c
          if (j.gt.6 .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
6       continue
6     continue
      return
      end
```

## References

Bunch, James R., Christopher P. Nielsen and Danny C. Sorensen, 1978. Rank-One Modification of the Symmetric Eigenproblem, *Numer. Math.*, **31**:31-48.

Cuppen, J. J. M., 1980. A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, *Numer. Math.*, **36**:177-195.

Ericsson T. and A. Ruhe, 1980. The Spectral Transformation Lanczos Method in the Numerical Solution of Large, Sparse, Generalized, Symmetric Eigenvalue Problems, *Math. Comp.*, **34**:1251-1268.

Kato, T., 1949. On the Upper and Lower Bounds of Eigenvalues, *J. Phys. Soc. Japan*, 334-339.

Paige, C. C., 1976. Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix, *J. Inst. Math. Appl.*, **18**:341-349.

Parlett, B. N., 1980. *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ.

Parlett, B. N. and D. S. Scott, 1979. The Lanczos Algorithm with Selective Orthogonalization, *Math. Comp.*, **33**:217-238.

Parlett, B. N., H. Simon and L. M. Stringer, 1982. On Estimating the Largest Eigenvalue With the Lanczos Algorithm, *Math. Comp.*, **38**:153-165.

Nour-Omid, B. and B. N. Parlett. Reduction of $(\mathbf{K}-\lambda\mathbf{M})\mathbf{z}=\mathbf{0}$ to $(\mathbf{A}-\nu\mathbf{I})\mathbf{x}=\mathbf{0}$, Tech. Report, (in preparation).

Nour-Omid, B., B. N. Parlett and R. L. Taylor, 1983. Lanczos Versus Subspace Iteration for Solution of Eigenvalue Problems, *Int J. num. Meth. Engng.*, **19**:859-871.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER: CPAM-175 | 2. GOVT ACCESSION NO.: AD-A134163 | 3. RECIPIENT'S CATALOG NUMBER: |
| 4. TITLE *(and Subtitle)*: The Use of Refined Error Bound when Updating Eigenvalues of Tridiagonals | | 5. TYPE OF REPORT & PERIOD COVERED: Unclassified |
| | | 6. PERFORMING ORG. REPORT NUMBER: |
| 7. AUTHOR(s): B.N. Parlett & B. Nour-Omid | | 8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0013 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of California Berkeley, CA 94720 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: | | 12. REPORT DATE: September 1983 |
| | | 13. NUMBER OF PAGES: 48 |
| 14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: | | 15. SECURITY CLASS. *(of this report)*: |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: |

16. DISTRIBUTION STATEMENT *(of this Report)*

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Lanczos algorithm is used to compute some eigenvalues of a given symmetric matrix of large order. At each step of the Lanczos algorithm it is valuable to know which eigenvalues of the associated tridiagonal matrix have stabilized at eigenvalues of the given symmetric matrix. We present a robust algorithm which is fast (20j to 40j operation at j-th Lanczos step), uses about 30 words of extra storage, and has a fairly short program (approx~~imately~~ 200 executable statements).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

This report was done with support from the Center for Pure and Applied Mathematics. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Center for Pure and Applied Mathematics or the Department of Mathematics.