MATHEMATICS OF COMPUTATION, VOLUME 35, NUMBER 152
OCTOBER 1980, PAGES 1251–1268

The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems

By Thomas Ericsson and Axel Ruhe

Abstract. A new algorithm is developed which computes a specified number of eigenvalues in any part of the spectrum of a generalized symmetric matrix eigenvalue problem. It uses a linear system routine (factorization and solution) as a tool for applying the Lanczos algorithm to a shifted and inverted problem. The algorithm determines a sequence of shifts and checks that all eigenvalues get computed in the intervals between them.

It is shown that for each shift several eigenvectors will converge after very few steps of the Lanczos algorithm, and the most effective combination of shifts and Lanczos runs is determined for different sizes and sparsity properties of the matrices. For large problems the operation counts are about five times smaller than for traditional subspace iteration methods.

Tests on a numerical example, arising from a finite element computation of a nuclear power piping system, are reported, and it is shown how the performance predicted bears out in a practical situation.

1. Introduction and Summary. In the present contribution we set out to obtain numerical solutions, λ and x, to the generalized symmetric eigenproblem,

$$Ku = \lambda Mu, \tag{1.1}$$

where K and M are symmetric matrices, and M is positive semidefinite. Our main practical interest will be in finite element computations; then K stands for the stiffness matrix, M for the mass matrix, and λ and u will give approximations to the modes of vibration of the structure. Our discussion will, however, not be restricted to such computations but will cover any problem formulated as (1.1) and satisfying the restriction

$$N(K) \cap N(M) = \{0\},$$

where $N(\ )$ denotes nullspace.

Our goal is to find all the eigenvalues λ in a given interval (α, β), most often in the lower end of the spectrum.

The matrices K and M are so large and sparse that it is not practical to perform similarity transformations, so the methods from [12], [15] cannot be used. On the other hand, we assume that a routine for solving linear systems is available (in the case

Received October 2, 1979; revised December 5, 1979.

1980 *Mathematics Subject Classification.* Primary 65F15; Secondary 15A18, 65N25, 65N30, 70J10, 73K25.

0025-5718/80/0000-0166/$05.50

of FEM computations that we can solve a static problem). Such a routine solves

$$(K - \mu M)x = y \tag{1.2}$$

for x, with y and μ given. It is divided into two parts; one *factorization*,

$$K - \mu M = LDL^T, \tag{1.3}$$

finding (possibly block-) triangular L and (block-) diagonal D and one *solution*,

$$x = L^{-T}D^{-1}L^{-1}y, \tag{1.4}$$

yielding x. As a by-product, we get a count on the number of eigenvalues $\lambda < \mu$, in the simplest case as the number of negative elements of D.

We also assume that M can be factored as

$$M = WW^T, \tag{1.5}$$

where W may be rectangular but is assumed to have linearly independent columns.

We will now apply an iterative algorithm, the Lanczos method, to this inverted and shifted problem. We compute approximations to ν_i, the eigenvalues of

$$(A - \mu I)^{-1} = W^T(K - \mu M)^{-1}W, \tag{1.6}$$

which are related to the original eigenvalues λ_k by

$$\nu_i = 1/(\lambda_k - \mu); \tag{1.7}$$

see Figure 1.1. This is the spectral transformation that maps eigenvalues λ_k close to the shift μ onto eigenvalues ν_i of large absolute values. The Lanczos algorithm applied to (1.6) will converge first to those extreme ν_i, and we will obtain approximations to all λ_k in the interval by searching it through with a sequence of shifts $\mu_0, \mu_1, \ldots, \mu_s$. We choose the shifts sequentially, starting from the lower end of the interval, using the eigenvalue count obtained from (1.3) as a check.

The use of the spectral transformation (1.7) is quite natural when dealing with iterative algorithms. The sectioning algorithm of Jensen [6] finds a set of shifts and uses inverse iteration to find the eigenvalues in their neighborhoods. In the engineering literature, spectral transformation is most often combined with simultaneous iteration as in Bathe and Wilson [3], where the single shift $\mu = 0$ is used, and in some works of Jennings et al. [1], where a sequence of shifts is used.

Hitherto the Lanczos algorithm has been used on (1.1) without inversion and has proved very successful as the works by Paige [8] and Parlett and Scott [10] show. Even though it is very natural to use it in conjunction with spectral transformation, as outlined in [11], it has only been used so with one shift at the end of the spectrum as in Underwood [14, Example 7].

After introducing some of our notation, we continue, in Section 2, by giving a general outline of the algorithm we have used. In Section 3 we give criteria for acceptance of a computed eigenvalue and eigenvector approximation. Properties of the spectral transformation have to be taken into account, when adapting the standard criteria (see [9]), to find a bound on the angle between the computed and true

eigenvectors as well as the difference between the eigenvalues. In Section 4 we put practical aspects on the algorithm and use operation counts to determine how many shifts we shall use, choosing between many shifts with few Lanczos steps each and few shifts with many Lanczos steps each. The remarkable fact is that we obtain operation counts that are several times better than those for inverse iteration algorithms when large problems are treated. We conclude the description in Section 5 by discussing the choice of shifts and starting vectors and the mechanism to check that approximations to all the relevant eigenvalues are obtained.

We discuss a practical example arising from a finite element calculation of a piping system in a nuclear power reactor. We demonstrate that the performance predicted in the earlier sections really bears out in practice when we compare our program to a widely distributed FEM package based on the procedures in [3].

We have chosen not to discuss two important aspects of our algorithm. One is the influence of rounding errors. It has happened to us that the eigenvalues have been obtained with a much better accuracy than predicted by a general perturbation theory applied to a standard backward error analysis of (1.1). What limits the accuracy is the precision of the factorization (1.3) and the solution (1.4), and it is believed that the rounding errors are correlated in a way that avoids the worst possible perturbations. See the discussion by Argyris et al. [2] or Strang and Fix [13] on finite element computations. We have also chosen to use only available (possibly unstable) codes, for the factorization and solution, and to postpone the introduction of a safer strategy based on the methods of Bunch and Kaufman [4] to a later occasion.

We believe that our notation agrees with standard practice in numerical analysis. Unless otherwise stated, all matrices are denoted by capital latin letters, A, B, K, M etc., and have order $n \times n$. We then let A_j stand for the matrix obtained by taking the j first columns of A, and $a_1, \ldots, a_j$ stand for those columns. The eigenvalues of a matrix A are denoted by $\lambda(A)$ and ordered from below

$$\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n.$$

Exceptions are the unit matrix I, with columns $e_1, e_2, \ldots, e_n$, and the tridiagonal matrix

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & 0 & \cdots & 0 \\ 0 & \beta_2 & \alpha_3 & \beta_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & & \beta_{j-1} & \alpha_j \end{bmatrix}.$$

We will use the Euclidean vector norm,

$$\|x\|_2 := (x^T x)^{1/2},$$

and we define the angle between two vectors by

$$\sphericalangle(x, y) := \arccos |x^T y| / (\|x\|_2 \, \|y\|_2).$$

For any set S, we let $|S|$ stand for its cardinality, i.e., number of members.

2. **The Algorithm.** In this section we formulate the algorithm we have used in an informal algorithmic language. Some of the steps will in themselves involve non-trivial computations and choices. We devote the rest of this section and the following ones to a discussion of these.

From now on we will often discuss the problem (1.1) using the reformulation as a standard eigenvalue problem,

$$Ax = \lambda x, \quad A = W^{-1}KW^{-T}, \quad x = W^T u, \tag{2.1}$$

understanding that we never form A explicitly but use the factorizations (1.3) and (1.5), of K and M, to solve shifted systems as (1.6).

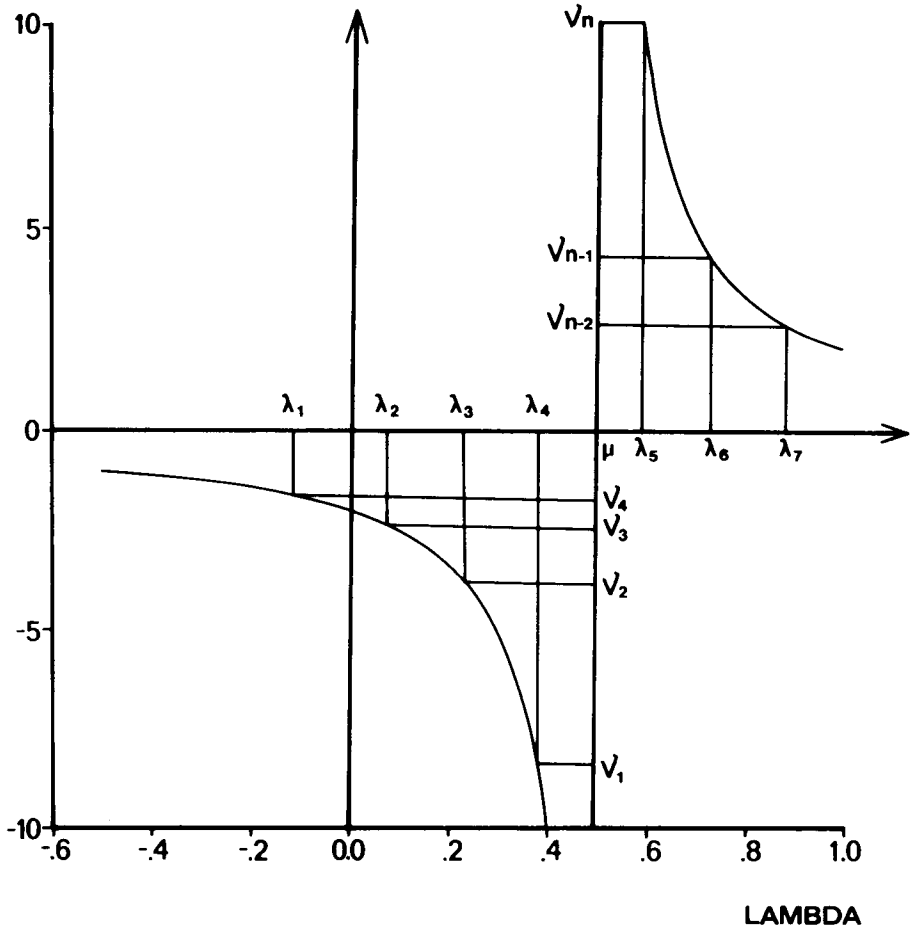

FIGURE 1.1

Original eigenvalues $\lambda(A)$ *versus transformed eigenvalues* $\nu = \lambda((A - \mu I)^{-1})$

The algorithm is built up around the spectral transformation (1.7) computing the eigenvalues,

$$\nu_i = \lambda_i((A - \mu I)^{-1}), \tag{2.2}$$

for different shifts μ. As seen from Figure 1.1, the smallest $\nu_1, \nu_2, \ldots$ correspond to those eigenvalues $\lambda(A)$ that are below μ, $\lambda_r, \lambda_{r-1}, \ldots$, while the largest ν_n, $\nu_{n-1}, \ldots$ correspond to those above, $\lambda_{r+1}, \lambda_{r+2}, \ldots$. We also see that the extreme ν_1, ν_n are very well separated from the rest, and that those λ's far from μ are lumped together around zero in the transformed spectrum. We will use the Lanczos algorithm to determine the extreme eigenvalues $\nu_1, \nu_2, \ldots$ and $\ldots, \nu_{n-1}, \nu_n$, and to obtain approximations to the eigenvalues of (1.1) around μ.

The outermost level of our algorithm deals with determining a sequence of shifts μ_s and records which eigenvalues converge in different subintervals.

ALGORITHM A. Determine all eigenvalues λ_k in interval $[\alpha, \beta]$.

$S(\gamma, \delta)$ set of eigenvalues in (γ, δ).

$C(\gamma, \delta)$ set of converged eigenvalues in (γ, δ).

1. Start with $\mu_0 = \alpha$, $C = \varnothing$.
2. *Repeat* (for $s = 0, 1, \ldots$).
   1. Factorize $K - \mu_s M$, giving eigenvalue count $|S(\alpha, \mu_s)|$.
   2. Draw random starting vector r_0, orthogonalizing to converged vectors if necessary.
   3. Run Lanczos, with $(A - \mu_s I)^{-1}$ starting at r_0, giving $\nu_1, \ldots, \nu_r$ as converged eigenvalues.
   4. Add $\mu_s + \nu_i^{-1}$ to C, $i = 1, \ldots, r$.
   5. Determine a new shift μ_{s+1}, or restart with a new vector at 2.2.

   until $|C(\alpha, \beta)| = |S(\alpha, \beta)|$.
3. Transform the eigenvectors to original coordinates.

Let us now comment on some of the steps in more detail:

Step 2.1. The factorization (1.3),

$$K - \mu_s M = LDL^T, \tag{2.3}$$

is required to give the inertia of $K - \mu_s M$, that is the number of eigenvalues below μ_s.

We have used symmetric Gaussian elimination without pivoting, making D diagonal and L lower triangular with unit diagonal. The count is then simply the number of negative diagonal elements in D. Such a routine is what traditionally has been used by practitioners; see [3]. The algorithm is safe numerically, provided that the elements in the factorization do not grow uncontrollably. Such growth occurs if μ is close to an eigenvalue of (K, M) or any of its leading minors, and the general tactic is to move μ away in case any of those $n(n + 1)/2$ trouble spots are encountered. The reliability of such a tactic and the motivation for including a safer strategy, based on indefinite factorization [4], need further study and are not discussed in this contribution.

Step 2.2. A random direction is generated by taking n normally distributed random numbers. Rectangular numbers avoid directions close to the coordinate directions.

Step 2.3. The Lanczos algorithm needs a routine for multiplying a vector by the matrix whose eigenvalues are to be determined. In this case the relation (1.6) is used, together with (1.4). Note that if $\operatorname{rank}(M) = m < n$, then the matrix A will also have order $m \times m$.

The reader is referred to any good work about the Lanczos method for a detailed description of the computations in this step; see [8], [10], [9] or [11]. At this level it is sufficient to know that Lanczos delivers the extreme eigenvalues of $(A - \mu I)^{-1}$ first and then approximations to those further inside the spectrum. When to determine convergence will be discussed in Section 3. Section 4 is devoted to a discussion of how many steps of the Lanczos algorithm should be executed before leaving Step 2.3 of our algorithm.

Step 2.5. The choice of the next shift μ_{s+1} depends on the success of the current step. A discussion of the different cases that can occur is given in Section 5.

Step 3. When M has full rank, we simply take (2.1)

$$u = W^{-T}x,$$

but otherwise we have to compute

$$u = (K - \mu M)^{-1} Wx,$$

for any suitable shift μ, e.g., using the last factorization (2.3) available.

3. Accuracy of the Computed Results. In this section we will take a careful look at the spectral transformation (2.2) and the Lanczos algorithm to see how to determine whether an eigenvalue and eigenvector approximation is good enough to label it converged.

The Lanczos algorithm will compute, after j steps, an $n \times j$ matrix Q_j of orthonormal columns and a $j \times j$ tridiagonal matrix T_j, satisfying

$$(A - \mu I)^{-1} Q_j - Q_j T_j = \beta_j q_{j+1} e_j^T, \tag{3.1}$$

up to rounding errors. The eigenvalues ν_i of T_j will approximate those of $(A - \mu I)^{-1}$, from the inside of the spectrum, so that $\mu + \nu_i^{-1}$ will give under (over) estimates of the eigenvalues $\lambda_k(A)$ to the left (right) of μ. Generally the approximations to the extreme eigenvalues of T_j are the best ones, and they will correspond to the eigenvalues of A closest to μ.

Computing the eigendecomposition of T,

$$T = SDS^T, \quad D = \operatorname{diag}\{\nu_i\}, \tag{3.2}$$

(drop j from now on) we can assess the accuracy of its eigenvalues as eigenvalues of $(A - \mu I)^{-1}$ by noting that with

$$y_i = Qs_i \tag{3.3}$$

we can get the residual

$$(A - \mu I)^{-1} y_i - y_i \nu_i = (A - \mu I)^{-1} Q s_i - Q s_i \nu_i$$

$$= ((A - \mu I)^{-1} Q - QT) s_i = \beta_j q_{j+1} e_j^T s_i \tag{3.4}$$

$$= q_{j+1} \beta_j s_{ji} = q_{j+1} \beta_{ji}.$$

The computable quantity β_{ji}, the product of the last nondiagonal element, and the last element of the normalized eigenvector of T, will be the basic quantity in assessing the accuracy. From (3.4) we get

$$|(\lambda_k - \mu)^{-1} - \nu_i| \leqslant |\beta_{ji}|, \tag{3.5}$$

for an appropriate choice k, and using the gap,

$$\gamma_k = \min_{l \neq k} |(\lambda_l - \mu)^{-1} - (\lambda_k - \mu)^{-1}|,$$

we can bound the angle between the eigenvector x_k and y_i by

$$\sin \sphericalangle (x_k, y_i) \leqslant |\beta_{ji}| / \gamma_k, \tag{3.6}$$

and improve the eigenvalue bound,

$$|(\lambda_k - \mu)^{-1} - \nu_i| \leqslant \beta_{ji}^2 / \gamma_k. \tag{3.7}$$

Let us now turn to the original problem and see how well $\mu + \nu_i^{-1}$ approximates λ_k. We can reformulate (3.5) and make use of the fact that $|\nu_i| \leqslant |\lambda_k - \mu|^{-1}$ (approximation from the inside of spectrum) to get

$$|\lambda_k - (\mu + \nu_i^{-1})| = |\nu_i^{-1}(\lambda_k - \mu)((\lambda_k - \mu)^{-1} - \nu_i)| \leqslant |\beta_{ji}| / \nu_i^2, \tag{3.8}$$

showing that only a moderate accuracy β_{ji} is necessary to give good approximations to those λ_k that correspond to large ν, that is those that are close to the shift μ. Note, however, that (3.4) implies $|\beta_{ji}| \leqslant \max(-\nu_1, \nu_n) \cdot |s_{ji}|$, making the error essentially proportional to s_{ji}/ν_i.

The eigenvector bound (3.6), on the other hand, cannot be improved in a corresponding manner. Moreover, even though y_i is a good eigenvector approximation for $(A - \mu I)^{-1}$, it is not so for A. That can be seen by looking at its Rayleigh quotient which is not close to $\mu + \nu_i^{-1}$. Instead we should use the modified vector,

$$z_i = y_i + (\beta_{ji}/\nu_i) q_{j+1}, \tag{3.9}$$

obtained by adding a little of the next Lanczos vector. The reason for this is that, by (3.4),

$$z_i = y_i + \{(A - \mu I)^{-1} y_i - y_i \nu_i\}/\nu_i = \nu_i^{-1} (A - \mu I)^{-1} y_i,$$

and now we see that the Rayleigh quotient is (drop i for convenience)

$$\rho(z) = z^T A z / z^T z = \mu + z^T (A - \mu I) z / z^T z.$$

To express this in terms of ν and β, we first note that y_i and q_{j+1} are two orthonormal vectors so that $z^T z = 1 + \beta^2/\nu^2$, by (3.9), giving us the value of the denom-

inator. To evaluate the numerator, we first express z in terms of y and then use the fact that y is a Ritz approximative eigenvector to $(A - \mu I)^{-1}$, corresponding to the eigenvalue approximation ν. We get

$$\begin{aligned}\rho(z) &= \mu + z^T(A - \mu I)z/z^Tz \\ &= \mu + \nu^{-2}y^T(A - \mu I)^{-1}y/(1 + \beta^2/\nu^2) \\ &= \mu + \nu^{-1}(1 + \beta^2/\nu^2)^{-1},\end{aligned}$$

differing from $\mu + \nu^{-1}$ only by second order quantities, and the residual will be using (3.4);

$$Az - z\rho(z) = \frac{\beta}{\nu^2}\left(\frac{\beta}{\nu}y - q\right)/(1 + \beta^2/\nu^2),$$

$$\|Az - z\rho(z)\|/\|z\| = \frac{|\beta|}{\nu^2}/(1 + \beta^2/\nu^2).$$

With z_i we can now get a bound for the error angle of the eigenvector,

$$\sin \sphericalangle (x_k, z_i) \leqslant |\beta_{ji}|/[\nu_i^2\delta_k(1 + \beta_{ji}^2/\nu_i^2)], \tag{3.10}$$

where δ_k is the gap of λ_k in the original problem,

$$\delta_k = \min\{\lambda_{k+1} - \lambda_k, \lambda_k - \lambda_{k-1}\}. \tag{3.11}$$

The last factor in the denominator of (3.10) is close to, and larger than, unity, and so we realize that $|\beta_{ji}|/\nu_i^2$ also is the relevant quantity for bounding the error in the eigenvector approximation. We can, further, get a bound similar to (3.7) for the eigenvalue,

$$|\lambda_k - \rho(z_i)| \leqslant (\beta_{ji}/\nu_i^2)^2/[\delta_k(1 + \beta_{ji}^2/\nu_i^2)^2]. \tag{3.12}$$

If we want to use (3.10) to determine when to stop the computations and label the eigenvalue as converged, we need a computable estimate for δ_k (3.11) and an assurance that (3.10) always will drop below a certain tolerance if only we perform sufficiently many iterations. Such an assurance can be granted only for separated eigenvalues, and it is therefore necessary to replace δ_k by a gap between such eigenvalues, with the silent understanding that problems with closer ones will only get a subspace basis accurately determined.

One way of estimating the gap is to study the eigenvalues just above and below the shift μ. For these we see that $\lambda_{k+1} - \lambda_k \leqslant \nu_n^{-1} - \nu_1^{-1}$, giving a slight overestimate of δ_k. In the program, our strongest concern is to avoid demanding so much accuracy as to cause an infinite loop, so there we use the less stringent scaling invariant stopping criterion

$$|\beta_{ji}/\nu_i| < \text{tolerance}. \tag{3.13}$$

One complication arising from using z_i, instead of y_i as eigenvector approximation, is that different z_i's are not exactly orthogonal to each other. However, the departure from orthogonality is only a second order effect since

$$z_i^T z_l = (y_i + \beta_{ji}/\nu_i q_{j+1})^T (y_l + \beta_{jl}/\nu_l q_{j+1}) = (\beta_{ji}/\nu_i)(\beta_{jl}/\nu_l),$$

so that the z vectors corresponding to different eigenvalues converged during the same Lanczos run will be nearly orthogonal. The orthogonality is comparable to that between two z vectors converged during different Lanczos runs.

4. How Many Eigenvalues for Each Shift? The efficiency of our algorithm is intimately dependent on how we choose to place the shifts μ_s and how many steps j we run the Lanczos algorithm each time. The considerations will be of a software engineering nature, trying to minimize the expenditure in computer time and storage, to reach the goal of computing all the required results. The choice is between using many shifts μ_s, necessitating many factorizations of the matrix (1.3), but for each shift only demanding few Lanczos steps j, or using few shifts, saving the factorizations, but needing more time and space for longer Lanczos runs.

We will first estimate how many eigenvalues, r say, will converge if we perform j Lanczos steps. Then we will evaluate the cost of performing these j steps, on different assumptions on the sparsity pattern of the matrices, and the reorthogonalization strategy in Lanczos. Finally we minimize the cost for each eigenvalue by making a choice on how far to go in each Lanczos run, either in terms of r or j. We also give comparable operation counts for traditional algorithms based on inverse iteration and subspace iteration.

In order to estimate r, the number of eigenvalues converged after j Lanczos steps, we have made a calculation based on the Kaniel-Paige-Saad error bounds; see [9, Chapter 12]. We assume that the eigenvalues $\lambda(A)$ are approximately linearly distributed. This gives a special distribution to $\lambda[(A - \mu I)^{-1}]$, if we assume that μ is in the middle between two of the $\lambda(A)$'s. We calculate the amplification factor for the component of x_i, the eigenvector, after different numbers of Lanczos steps j. When that amplification factor, corrected for the scaling by multiplying by ν_k^2, exceeds 10^6, we declare that the corresponding eigenvalue has converged. Since the eigenvalue distribution of $(A - \mu I)^{-1}$ will be essentially symmetric, the eigenvectors will converge in pairs. Inspecting Table 4.1, where the amplification factors are recorded, we see that ten steps are needed for the first pair to converge and that then one pair converges each five steps. We therefore assume that

$$j = 5 + 2.5r, \tag{4.1}$$

where r is the number of eigenvectors converged. This is of course only a qualified quess, but we have also found that our practical test runs behaved like this in essentials.

When evaluating the cost of performing j Lanczos steps, we assume that the expenditure is proportional to the number of arithmetic operations. (One operation is one multiplication plus one addition.) This disregards the expenditure for storage administration and assumes that reading and writing from external storage can be overlapped by other operations. Such an assumption is not entirely unfounded since external storage will be used only sequentially when reading and writing Lanczos vectors Q (3.1), (3.3).

TABLE 4.1

Amplification factors for eigenvectors from Kaniel-Paige-Saad bounds

Step	Converged	Eigenvalue pair r/2									
j	r	1	2	3	4	5	6	7	8	9	10
5	0	1.5E2									
10	2	6.3E6	5.7E2	2.2E1	3.2E0						
15	4	5.0E12	1.6E6	3.4E3	1.2E2	1.8E1	4.2E0				
20	6	1.8E19	2.3E10	3.4E6	1.8E4	6.2E2	7.9E1	1.8E1	5.3E0	2.0E0	
25	8	2.4E26	1.3E15	1.4E10	1.2E7	9.7E4	3.1E3	2.8E2	6.8E1	2.0E1	6.7E0
30	10	8.6E33	1.9E20	1.5E14	2.3E10	5.0E7	5.2E5	1.7E4	1.3E3	2.2E2	6.4E1
35	14	7.3E41	6.5E25	4.1E18	1.1E14	6.3E10	2.2E8	2.7E6	8.5E4	5.9E3	7.8E2
40	16	1.2E50	4.5E31	2.2E23	1.1E18	1.7E14	2.1E11	1.0E9	1.5E7	4.6E5	2.8E4

Thus a vector inner product takes n operations, as well as adding a multiple of one vector to another $x := x + y\alpha$. The bulk of the operations will be of this kind except for the factorization and solution of the linear systems. To keep the discussion general, assume that

$$F \cdot n = \text{operation count for factorization (1.3)},$$
$$S \cdot n = \text{operation count for solution (1.4)}.$$

One Lanczos run now consists of one factorization. Then one solution and $5n$ operations are needed in each step of the Lanczos recursion (3.1) to compute q_{j+1}, α_j, and β_j. We also need to compute the eigenvalues and some components of the eigenvectors of T_j (3.2), but that takes a negligible cost for large n and small j. To insure the linear independence of the basis vectors, Q_j, we need to reorthogonalize the vectors, and the choice is between performing full [7], or selective reorthogonalization [10]. Finally we need to compute the eigenvector approximations (3.3) which is an expensive operation for large j. The operation counts are summarized in Table 4.2.

The operation counts for selective orthogonalization are intimately dependent on the convergence behavior and are made on the following assumptions; see Table 4.2. Each time one pair of eigenvalues converges far enough to necessitate reorthogonalization, the pair that did converge last time has reached full accuracy. We thus compute four Ritz vectors every five iterations; see (4.1). We also need to orthogonalize two basis vectors q_i against all converged vectors, since the amplification factors, recorded in Table 4.1, indicate that it will take about five steps for a new copy of an eigenvector, that has already converged, to grow up after a reorthogonalization.

Summing up, the work per eigenvalue will be

$$W = n/r\{\underbrace{F}_{\text{Factorization}} + \underbrace{j \cdot S}_{\text{Solution}} + \underbrace{5j}_{\text{Lanczos}} + \underbrace{r(j + r + 12) - 20\}}_{\text{Eigenvectors + Orthogonalization}}, \tag{4.2}$$

provided that we use selective orthogonalization. Insert $j = 2.5r + 5$ (4.1), and get

$$W = n\{(F + 5S + 5)r^{-1} + (2.5S + 29.5) + 3.5r\},$$

which is minimized for

$$r = ((F + 5S + 5)/3.5)^{1/2}, \tag{4.3}$$

giving

$$W = n\{2(3.5(F + 5S + 5))^{1/2} + 2.5S + 29.5\}. \tag{4.4}$$

The corresponding figures for full reorthogonalization can be obtained analogously.

To get some idea of how these operation counts turn out, let us discuss two special cases that we believe to be typical. They are:

(a) Band matrix, half width m, factorization $0.5nm^2$, solution $2nm$ operations.

(b) Nested dissection scheme of a 2-dimensional elliptic equation, factorization $10n^{3/2}$, solution $10n \log n$.

The reader is referred to George [5] for a discussion justifying these figures. Case (a) also covers such schemes as profile or envelope storage; m is used only to get a value to insert in the operation counts and can be termed equivalent mean (half) bandwidth. A third case, using a fast Poisson solver, would also be of interest, but so far we have not seen any such algorithm that works on indefinite matrices.

We record the values of r and W/n in Table 4.3. For the band case we list both the figures (4.3) and (4.4) obtained for selective orthogonalization and full reorthogonalization. In that case the last term in (4.2) is replaced by $j(j + r + 1)$, which makes the optimal r a good deal smaller and the optimal work about 20% larger. Full reorthogonalization is safer and simpler to program, and from these figures it is indicated that it is not significantly more time consuming than selective orthogonalization.

As a comparison we list also the operation count for solving one system (factorization + solution) as well as for computing the eigenvalues using inverse iteration. The first inverse iteration acts on one vector at a time, needing four factorizations for each eigenvalue, as assumed for the algorithm DETERMINANT SEARCH of [3]. The second inverse iteration is SUBSPACE ITERATION of [3], using $q = 58$ vectors to find $p = 50$ eigenvalues. Smaller p, q give even larger operation count for large bandwidths m.

We see that the counts for our algorithm are significantly better than those for any of the inverse iterations, for all bandwidths recorded. The improvement is around a factor five for the larger bandwidths.

The figures for nested dissection with selective orthogonalization are given in the last columns of Table 4.3. It is not a coincidence that the data for nested dissection for $n = m^2$ are at the same line as those for band m. Those data correspond in the case when we deal with a 2-dimensional elliptic problem. For matrices with narrow bandwidth, $m << n^{1/2}$, a band solution scheme is the most natural. Nested dissection will only be of advantage for relatively large problems. The crossover point is at about $n = 1000$ precisely as in the linear systems case [5].

TABLE 4.2

Operation counts for different steps of algorithm

Step		Operation count	
		Band	Dissection
Factorization	(1.3)	$0.5nm^2$	$10n^{3/2}$
Solution	(1.4)	$2nmj$	$10jn \log n$
Lanczos	(3.1)	$5nj$	
Eigensystem of T	(3.2)	Crj^2	
		Full	Selective
Reorthogonalization		$nj(j+1)$	$nr(j+r+12) - 20$
Eigenvectors of A	(3.3)	nrj	

Assumed history of convergence of eigenvectors and selective orthogonalization

steps j	# converged vectors r	# eigenvectors / # basisvectors		innerprod + subtr q_j, q_{j+1} / # eigenvectors
10	2	2·10	+	2·2·2
15	4	4·15	+	2·2·4
20	6	4·20	+	2·2·6
j		4·j	+	4· r
Total	r	$4\frac{r}{2}\frac{1}{2}(10+j+2+r) - 20$		

TABLE 4.3

Optimal number of eigenvalues for each shift r and work per eigenvalue W/n. Band elimination bandwidth 2m +1 and nested dissection of elliptic problem of order n

	Band storage				Inverse iteration			Nested dissection		
	Full reorth.		Selective orth.		F + S	Single vector	Subspace iteration	selective orthog.		
m	r	W/n	r	W/n	W/n	W/n	W/n	n	r	W/n
5	3	135	4	85	22	296	1524	25	7	165
10	4	181	6	126	70	576	1667	100	9	213
15	6	225	8	166	142	956	1811	225	11	242
20	7	271	10	205	240	1436	1956	400	12	263
40	11	455	18	359	880	4356	2545	1600	14	318
60	16	638	26	513	1920	8876	3150	3600	17	353
80	21	822	33	666	3360	14296	3771	6400	18	381
100	26	1005	41	819	5200	22716	4408	10000	20	403

5. Choice of Shift and Starting Vectors. The strategy for choosing shifts μ_s and the tactic of finding a good starting vector for each shift are governed mainly by heuristics. We found that it is most economic if each shift is used to find, on the average, r eigenvalues, r determined by the assumptions on the density of the matrix made in the previous section. Moreover, we want to be sure to have found all eigenvalues in the interval (μ_{s-1}, μ_s).

The first shift μ_0 is always chosen at the left end α of the interval, most often zero. We then run Lanczos until $r/2$ eigenvalues are found or until j has reached a limit $j_{\max}$. From now on μ_0 will be treated like any other shift and enters the decisions as the old shift. We will now choose a sequence of new shifts, walking through the spectrum from left to right and making sure that no eigenvalues will be passed without being calculated. The main tool for this is the eigenvalue count, produced as a by-product from each factorization (1.3).

To choose the new shift μ_s we use the information gathered when running the old one, μ_{s-1}. If some eigenvalues have converged to the right of μ_{s-1}, say λ_{k+1}, $\ldots$, λ_{p_s}, we choose

$$\mu_s = \mu_{s-1} + 2(\lambda_{p_s} - \mu_{s-1}), \tag{5.1}$$

so that the last eigenvalue λ_{p_s} is half way between μ_{s-1} and μ_s. The next iteration is started by factorizing $(K - \mu_s M)$, and this yields a count on the number of eigenvalues to the left of μ_s. In the unfortunate case when the eigenvalues cluster between λ_{p_s} and μ_s, that is when $|S(\mu_{s-1}, \mu_s)| >> r$, we have to move back and choose a new μ_s.

When choosing a starting vector, we have a choice between using some combination of the old Lanczos vectors or taking a random direction. We have found that, when several eigenvalues converge for each shift, it is advantageous to choose the starting vector at random. We avoid convergence towards eigenvectors in (μ_{s-1}, μ_s) which have already converged in the following way: Before starting we orthogonalize the starting vector r_0 against all converged vectors in (μ_{s-1}, μ_s). Now rounding errors may grow fast (see Table 4.1), so that, nevertheless, we get a new copy of an already converged vector. Such a copy, however, reveals itself by being orthogonal to the starting vector. We add a test against that, discarding vectors below μ_s for which s_{1i} is close to zero; see (3.2), (3.3).

If no eigenvalues to the right of μ_{s-1} did converge during the old iteration, but T had some positive eigenvalues, we choose the λ corresponding to the largest one as the new shift, making $\mu_s = \mu_{s-1} + \nu_j^{-1}$. If no eigenvalues to the right of μ_{s-1} did converge, and T does not even have any positive eigenvalues, we have either exhausted the spectrum, or the next eigenvalue is far indeed to the right of μ_{s-1}, and we choose to stop.

Now we have got the new shift μ_s together with a starting vector, and are assigned to compute $|S(\mu_{s-1}, \mu_s)| - |C(\mu_{s-1}, \mu_s)|$ eigenvalues in the interval (μ_{s-1}, μ_s), together with whatever eigenvalues we can find to the right of μ_s. We run Lanczos until r eigenvalues have converged, or $j_{\max}$ steps, whichever occurs first, and examine the results. In nearly all cases we have got precisely $|S(\mu_{s-1}, \mu_s)|$ eigenvalues, fulfilling

the assignment at the first shot, but it may happen that we have too few. If no eigenvalues at all have converged to the left of μ_s, a most unlikely event, we restart with a new shift to the left of μ_s, $\mu_s = \mu_{s-1} + \nu_1^{-1}$. If eigenvalues have converged, the cause of missing eigenvalues is either a multiple (more than double) eigenvalue in (μ_{s-1}, μ_s) or an unfortunate choice of starting vector. Both things are cured, if we start anew, using the same factorization and a new random starting direction, orthogonalized against all eigenvectors that have been found to eigenvalues in (μ_{s-1}, μ_s). This makes the missing eigenvalues stand out as more and more dominating negative eigenvalues of $(A - \mu_s I)^{-1}$. This restart may have to be repeated several times; in the presence of a five fold eigenvalue we need three restarts.

When all eigenvalues in (μ_{s-1}, μ_s) are found, μ_s becomes the old shift, and the positions of those eigenvalues converged to its right determine how to continue. Finally we reach the right end of the interval or have computed the number of eigenvalues that were asked for initially.

6. A Practical Example. In this section we will report the results of several test runs on a small but yet realistic test example. All the computations have been performed on a Honeywell 6000 computer at the ASEA company, Västerås, Sweden. Single precision has been used, so rounding errors leave about eight correct decimal digits. The timings given are for an in core routine; we wait to use external storage until the results have to be written out.

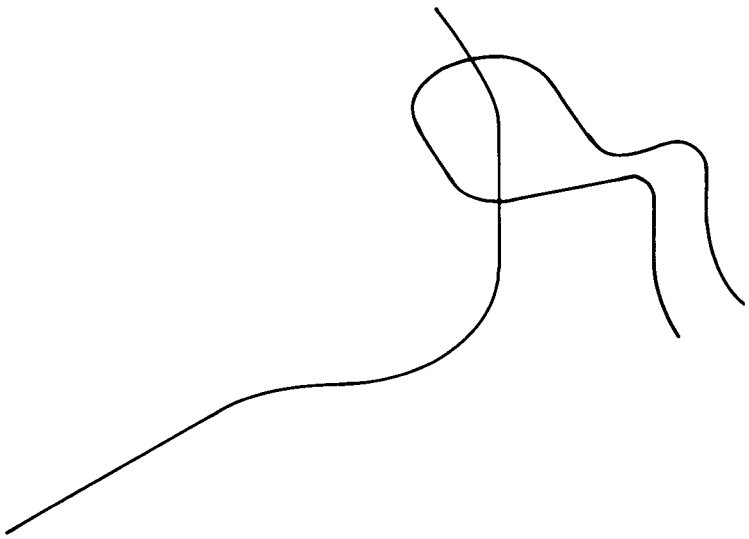

FIGURE 6.1

Perspective drawing of piping system for which test results are reported

The program is a straightforward FORTRAN implementation of our algorithm, and we have used the routine from [3] to factorize and solve linear systems (1.3), (1.4).

The program for the Lanczos algorithm uses complete reorthogonalization of the q_j vectors.

The matrix is a finite element approximation to a piping system which is displayed in Figure 6.1. The order n is 384. The stiffness matrix K (1.1) needs 4512 words when an envelope storage scheme is used, while the mass matrix M is made diagonal. The condition numbers are,

$$K(A) \simeq 10^9, \qquad K(M) \simeq 10^4,$$

so the general perturbation theory for symmetric matrices does not promise to give any accuracy at all in the smallest eigenvalues when single-precision computations are used. In spite of this, we consistently got several correct figures in all eigenvalues we tried to get.

We are interested in getting the fifty smallest eigenvalues, which are in the interval (0, 0.02) whereas $\lambda_{\max} \simeq 2 \cdot 10^4$. We have plotted those in (0, 0.002) in Figure 6.2, and we see that they are distributed fairly evenly with a few small clusters.

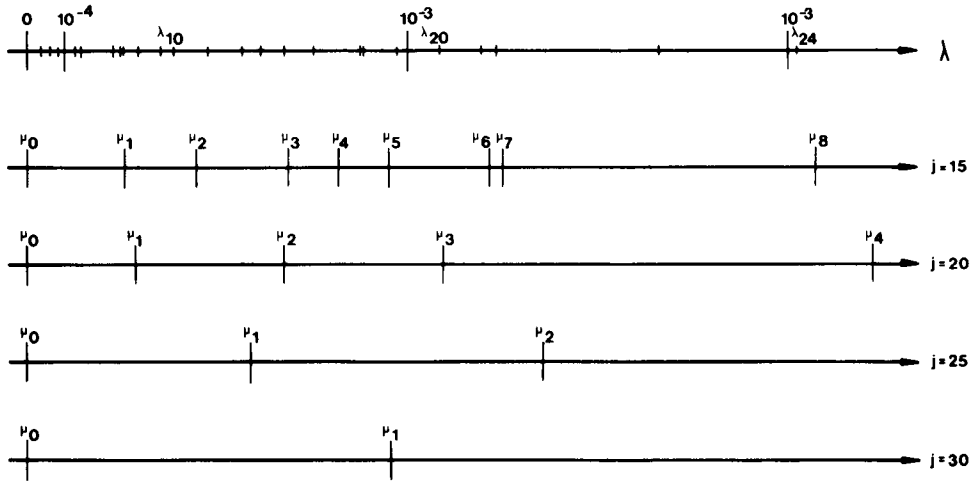

FIGURE 6.2

Eigenvalues λ_k and shifts μ_s of test problem

Each Lanczos run (Step 2.3 of the algorithm) used a predetermined number of j steps, and then convergence was tested with the tolerance (3.13) set to 10^{-5}. We tried $j = 5(5)30$ and 45. For $j \geqslant 15$ we got one or more eigenvalues converging each time, and we summarize the results in Table 6.1.

TABLE 6.1

Summary of results of test runs

Steps j	Total number of shifts μ_s	Total number of eigenvalues λ_k	Total time	Time per eigenvalue
10	3	3	9.4	3.13
15	18	47	87.2	1.86
20	10	47	72.3	1.54
25	8	49	79.0	1.61
30	5	46	67.2	1.46
45	3	52	83.3	1.60

In the lower part of Figure 6.2, we show how the shifts μ_s were chosen and when the corresponding eigenvalues converged. It is noticeable that the clustering was not as severe as to activate the special routines for treating clustered eigenvalues and that the eigenvalues always were computed in order, i.e., converging when the closest shift was used.

It is interesting to see if the assumption (4.1) on the number of eigenvalues converging for different numbers of steps j is realistic. We plot the values measure in Figure 6.3 and see that, for small j, (4.1) was too optimistic. This is due to the fact that in those cases the clustering of the eigenvalues have a detrimental effect on the placement of the shifts. Possibly a more sophisticated rule than (5.1) should be used. For larger j, on the other hand, the actual behavior was better than predicted, since then the clusterings were averaged out. The improvement over (4.1) is also indicated by the theoretical bounds in Table 4.1.

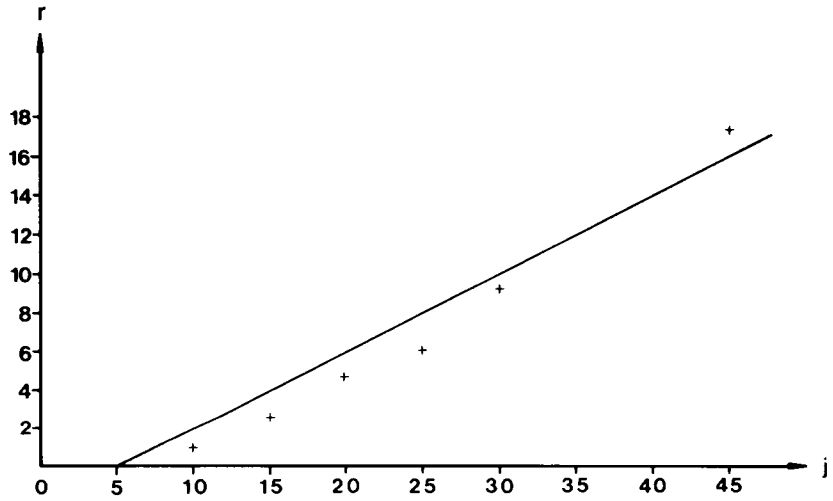

FIGURE 6.3

Number of converged eigenvectors r versus Lanczos step j, in test example. Predicted relation drawn as straight line

We have also tried to verify the validity of the reasoning in Section 4, concerning the choice of r to minimize work. We plot the time per eigenvalue spent in different parts of the program in Figure 6.4, for the j values we have tested. We see how the time spent with factorization and solution decreases with j, while the time for computing vectors and reorthogonalization grows, precisely as predicted. The effect is not as strong as expected, since the bandwidth is not very large ($m \approx 15$), which makes the factorization a very small part of the work, and the faster convergence for larger j makes the orthogonalization part grow slower than expected.

We have run the problem with the ADINA package, using the determinant search and inverse iteration routine of [3]. When computing forty-five eigenvalues, it needed seven seconds per eigenvalue. This is in line with the figures in Table 4.3, comparing the work between our algorithm and inverse iteration, and supports our comparison

made there. The eigenvalues computed by the two methods agreed to at least four digits of accuracy.

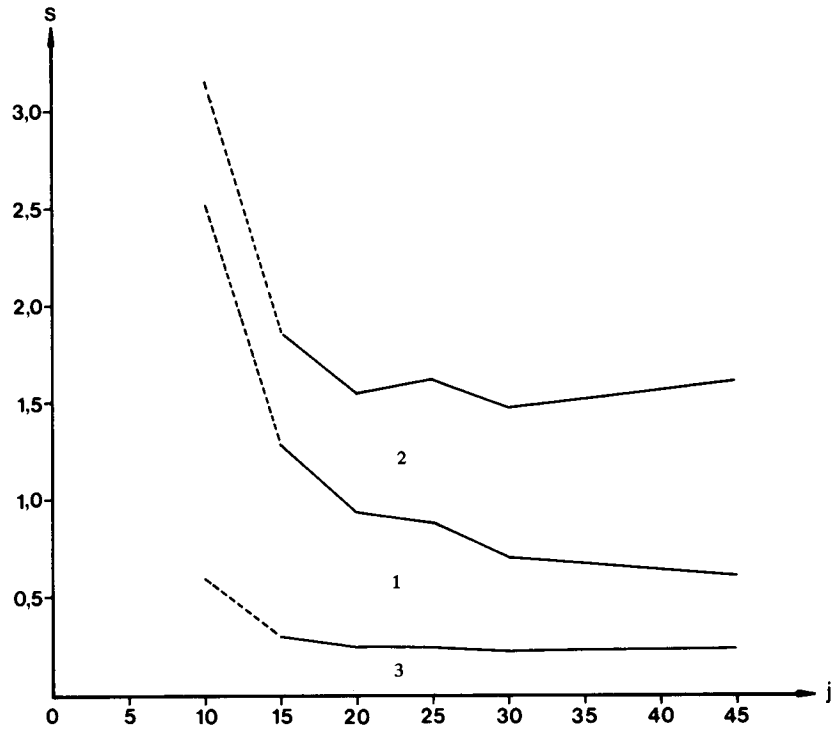

FIGURE 6.4

Time per eigenvalue in seconds as a function of number of Lanczos steps j.
Time spent in different parts of the program:

1. *Factorization and solution.*
2. *Computing eigenvectors and reorthogonalization.*
3. *The rest.*

Acknowledgement. This work would not have come about without the eager interest and enthusiastic support of Dr. Christer Gustafsson at ASEA ATOM, Västerås.

Department of Numerical Analysis
Institute of Information Processing
University of Umeå
S-90187 Umeå, Sweden

1. T. J. A. AGAR & A. JENNINGS, *Hybrid Sturm Sequence and Simultaneous Iteration Methods,* Internat. Sympos. Appl. of Computer Methods in Engineering, Univ. of Southern California, Los Angeles, Calif., 1977, pp. 405–412.

2. J. H. ARGYRIS, T.L. JOHNSEN, R. A. ROSANOFF & J. R. ROY, "On numerical error in the finite element method," *Comput. Methods Appl. Mech. Engrg.*, v. 7, 1976, pp. 261–282.

3. K. J. BATHE & E. L. WILSON, *Numerical Methods in Finite Element Analysis,* Prentice-Hall, Englewood Cliffs, N.J., 1976.

4. J. R. BUNCH & L. KAUFMAN, "Some stable methods for calculating inertia and solving symmetric linear systems," *Math. Comp.*, v. 31, 1977, pp. 163–179.

5. J. A. GEORGE, "Solution of linear systems of equations: Direct methods for finite element problems," *Sparse Matrix Techniques* (V. A. Barker, Ed.), Lecture Notes in Math., Vol. 572, Springer-Verlag, Berlin and New York, 1977, pp. 52–101.

6. P. S. JENSEN, "The solution of large symmetric eigenproblems by sectioning," *SIAM J. Numer. Anal.*, v. 9, 1972, pp. 534–545.

7. C. C. PAIGE, "Practical use of the symmetric Lanczos process with re-orthogonalization," *BIT*, v. 10, 1970, pp. 183–195.

8. C. C. PAIGE, "Computational variants of the Lanczos method for the eigenproblem," *J. Inst. Math. Appl.*, v. 10, 1972, pp. 373–381.

9. B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980.

10. B. N. PARLETT & D. S. SCOTT, "The Lanczos algorithm with selective orthogonalization," *Math. Comp.*, v. 33, 1979, pp. 217–238.

11. A. RUHE, "Computation of eigenvalues and eigenvectors," in *Sparse Matrix Techniques*, (V. A. Barker, Ed.), Lecture Notes in Math., Vol. 572, Springer-Verlag, Berlin and New York, 1977, pp. 130–184.

12. B. T. SMITH ET AL., *Matrix Eigensystem Routines EISPACK Guide*, Lecture Notes in Comput. Sci., Vol. 6 and Vol. 51, Springer-Verlag, Berlin and New York, 1974, 1977.

13. G. STRANG & G. J. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

14. R. UNDERWOOD, *An Iterative Block Lanczos Method for the Solution of Large Sparse Symmetric Eigenproblems*, Tech. Report STAN-CS-75-496, Stanford University, Stanford, Calif., 1975.

15. J. H. WILKINSON & C. REINSCH (Eds.), *Handbook for Automatic Computation*, Vol. II, *Linear Algebra*, Springer-Verlag, Berlin and New York, 1971.