J. Inst. Maths Applics (1972) **10,** 373–381

Computational Variants of the Lanczos Method for the Eigenproblem

C. C. PAIGE *

London University, Institute of Computer Science,
44 Gordon Square, London WC1H OPD

[Received 28 October 1971 and in revised form 27 January 1972]

The Lanczos process is theoretically ideal for finding several extreme eigenvalues of large sparse symmetric matrices, but because of loss of orthogonality in practice it has largely been ignored, or used with re-orthogonalization. Here it is shown that it can still be an extremely efficient and accurate algorithm if used in an iterative manner. An error analysis is given showing the most accurate algorithms, and the conclusions are supported by computational results.

1. Introduction

ALTHOUGH the problem of computing eigensystems of symmetric matrices that fit easily into the fast store of a computer has been satisfactorily solved (see Wilkinson, 1965; Wilkinson & Reinsch, 1971), the same cannot be said for very large matrices. If a large matrix is sparse then the advantages of methods which only use the matrix in matrix-vector multiplications are obvious as the matrix $\mathbf{A}$ need not be stored explicitly, all that is needed is a procedure for computing $\mathbf{Av}$ from a given vector $\mathbf{v}$. At present the most popular of such methods is that of simultaneous iterations (see for example Rutishauser's algorithm in Wilkinson & Reinsch, 1971), which computes k extreme eigenvalues and their eigenvectors of an $n \times n$ matrix $\mathbf{A}$, usually $1 < k \ll n$. Unfortunately this algorithm requires at least $k+1$ vectors of length n throughout the computation, and usually more, with consequent problems of time and storage. Compared to this the far more simple method suggested by Lanczos (1950) for reducing a symmetric matrix to tri-diagonal form appears initially more attractive as it is very fast and only space for two vectors of length n need be used at each step if only the eigenvalues are required; storage problems increase if some eigenvectors are also needed, but sometimes none and often only one or two are required.

The Lanczos process is as follows (see for example Wilkinson, 1965: 388). Choose a non-zero vector $\mathbf{v}_1$ and take $\beta_1 = 0$, then with arbitrary normalizing factors γ_{j+1} form for $j = 1, 2, \ldots$

$$\alpha_j = \mathbf{v}_j^T\mathbf{A}\mathbf{v}_j/\mathbf{v}_j^T\mathbf{v}_j, \qquad \gamma_{j+1}\mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1},$$
$$\beta_{j+1} = \mathbf{v}_j^T\mathbf{A}\mathbf{v}_{j+1}/\mathbf{v}_j^T\mathbf{v}_j = \gamma_{j+1}\mathbf{v}_{j+1}^T\mathbf{v}_{j+1}/\mathbf{v}_j^T\mathbf{v}_j.$$

This ensures that $\mathbf{v}_1, \mathbf{v}_2, \ldots$ are orthogonal so that for some value of $j \leqslant n$, say s, $\gamma_{s+1}\mathbf{v}_{s+1} = 0$. Thus if $\mathbf{V} \equiv [\mathbf{v}_1, \ldots, \mathbf{v}_s]$ it follows that $\mathbf{AV} = \mathbf{VT}$ where $\mathbf{T}$ is tri-diagonal and every eigenvalue of $\mathbf{T}$ is also an eigenvalue of $\mathbf{A}$. If $s < n$ it is possible to continue the process to find the remaining eigenvalues of $\mathbf{A}$, but this is not of interest here.

A very significant feature of the algorithm is that several of the extreme eigenvalues

* Present address: School of Computer Science, McGill University, Montreal 101, Quebec, Canada.

of $\mathbf{T}_k$, the leading k by k part of $\mathbf{T}$, are likely to be very good approximations to some of the extreme eigenvalues of $\mathbf{A}$ even when $k \ll s$. Lanczos was well aware of this and Kaniel (1966) has given some theoretical results on the rate of convergence. These results suggest an iterative use of the Lanczos process for finding several extreme eigenvalues of large sparse symmetric matrices to some required accuracy. Good approximations to the corresponding eigenvectors are also available in this way using $\mathbf{v}_1, \ldots, \mathbf{v}_k$, which may be held in some large backing store, or may be recomputed when needed.

Unfortunately in an actual computation of the Lanczos process, cancellation in the vector subtraction step leads to loss of orthogonality in the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$ and the process usually never terminates. This practical behaviour led Lanczos to suggest re-orthogonalization, and to the widespread disregard of the method in its natural state. Nevertheless it is possible to obtain accurate eigenvalues and eigenvectors by using the method in an iterative manner using no re-orthogonalization at all, *even in the face of total loss of orthogonality*. This was realized by the author in the work for his doctoral thesis, which was then largely devoted to explaining this unexpected result and finding the most accurate algorithms. The excellent properties of the method can be partly understood when it is realized that partial cancellation corresponds to partial convergence of the eigenvalues of $\mathbf{T}_k$, that is, loss of orthogonality and convergence go hand in hand. This has been shown by Paige (1971), where for certain algorithms it is also shown that eigenvalues of $\mathbf{T}_k$ must necessarily converge as k increases, and that when they do they only differ from the eigenvalues of $\mathbf{A}$ by amounts depending on the computer precision. It is also shown that accurate eigenvectors can be computed by these algorithms. Unfortunately the proofs are extremely long and complicated and the proven rate of convergence does not begin to approach the remarkably fast rate found in practice. For these reasons the proofs will not be published until the above inadequacies have been remedied.

Although others have suspected that the Lanczos process could still be useful, no-one seemed to be aware that the most popular algorithm does not necessarily converge, and that even when it does, it gives poor results for close eigenvalues. It is the purpose of this paper to show why this is so, and to give other algorithms which do not suffer from such shortcomings. The error analyses will be supported by computational results exhibiting the significant differences in the various algorithms.

2. Possible Algorithms

It will be assumed that a floating point computer with precision ε is used. It can easily be shown that any normalization has a negligible effect on the rounding errors produced with the Lanczos process, although some normalization is usually necessary to avoid numbers that are too large or too small for the computer. For simplicity then, the γ_j will be chosen non-negative to give $\|\mathbf{v}_j\| = 1$ (this notation will always represent the 2-norm). Four important algorithms can now be described as follows. Choose $\mathbf{v}_1$ such that $\mathbf{v}_1^T\mathbf{v}_1 = 1$ and form $\mathbf{u}_1 = \mathbf{A}\mathbf{v}_1$, then for $j = 1, 2, \ldots$

$$\begin{cases} \alpha_j = \mathbf{v}_j^T\mathbf{A}\mathbf{v}_j & (1) \\ \quad \text{or} & \\ \alpha_j = \mathbf{v}_j^T\mathbf{u}_j, & (2) \end{cases}$$

these being essentially different when $j > 1$, since $\mathbf{u}_{j+1}$ is defined by (8);

$$\mathbf{w}_j = \mathbf{u}_j - \alpha_j \mathbf{v}_j \tag{3}$$

$$\gamma_{j+1} = +\sqrt{(\mathbf{w}_j^T \mathbf{w}_j)} \tag{4}$$

$$\mathbf{v}_{j+1} = \mathbf{w}_j / \gamma_{j+1} \tag{5}$$

$$\begin{cases} \beta_{j+1} = \mathbf{v}_j^T \mathbf{A} \mathbf{v}_{j+1} & (6) \\ \quad \text{or} \\ \beta_{j+1} = \gamma_{j+1} & (7) \end{cases}$$

$$\mathbf{u}_{j+1} = \mathbf{A}\mathbf{v}_{j+1} - \beta_{j+1}\mathbf{v}_j. \tag{8}$$

The choices here are between (1) and (2) and between (6) and (7). Let A(1, 6) denote the algorithm using equations (1) and (6). Similarly there are algorithms A(1, 7), A(2, 6), and A(2, 7). Now A(1, 6) and A(2, 6) use the natural and the modified Schmidt orthogonalization coefficients respectively, and if for these $\mathbf{v}_j$ is orthogonal to $\mathbf{v}_{j-1}$, then $\mathbf{v}_{j+1}$ will theoretically be orthogonal to both these vectors. But for A(1, 7) and A(2, 7) this orthogonality depends on the orthogonality of $\mathbf{v}_j$ to $\mathbf{v}_{j-2}$ as well as $\mathbf{v}_{j-1}$, which is known in practice to be poor; thus the previous two algorithms are understandably more popular despite requiring small amounts of extra computation. Note that A(1, 7) and A(2, 7) can be implemented using only two vectors, but A(2, 7) can also be used with a fast double-length accumulation of inner-products routine, while A(1, 7) requires a third vector to make this possible; these two are the best algorithms if only speed and storage are considered.

In the analysis the results given by Wilkinson (1965, Ch. 3) will be assumed, and for simplicity the n by n symmetric matrix $\mathbf{A}$ will be such that $\|\mathbf{A}\| = 1$. The equivalents of equations (1) to (8) for the computed values when rounding errors are present are then

$$\begin{cases} \alpha_j = \mathbf{v}_j^T \mathbf{A} \mathbf{v}_j + O(\varepsilon) & (1R) \\ \quad \text{or} \\ \alpha_j = \mathbf{v}_j^T \mathbf{u}_j + O(\varepsilon) & (2R) \end{cases}$$

$$\mathbf{w}_j = \mathbf{u}_j - \alpha_j \mathbf{v}_j + O(\varepsilon) \tag{3R}$$

$$\gamma_{j+1} = [1 + O(\varepsilon)] \cdot \|\mathbf{w}_j\| \tag{4R}$$

$$\mathbf{v}_{j+1} = \mathbf{w}_j / \gamma_{j+1} + O(\varepsilon) \tag{5R}$$

$$\begin{cases} \beta_{j+1} = \mathbf{v}_j^T \mathbf{A} \mathbf{v}_{j+1} + O(\varepsilon) & (6R) \\ \quad \text{or} \\ \beta_{j+1} = \gamma_{j+1} & (7R) \end{cases}$$

$$\mathbf{u}_{j+1} = \mathbf{A}\mathbf{v}_{j+1} - \beta_{j+1}\mathbf{v}_j + O(\varepsilon). \tag{8R}$$

Note that factors such as n have been omitted from the error terms for simplicity, so that these results hold for ordinary as well as double-length accumulation of inner-products. Anyone wishing to check these results closely should do so for (4)–(8), then (1)–(3), showing on the way that $\mathbf{v}_{j+1}^T \mathbf{v}_{j+1} = 1 + O(\varepsilon)$, and that for (1R) and (6R)

$$|\alpha_j|, \quad |\beta_{j+1}| < 1 + O(\varepsilon)$$

while for (2R)

$$|\alpha_j| \leqslant 1+|\beta_j|+O(\varepsilon),$$

and certainly by a very simple induction, for (7R)

$$\beta_{j+1} \leqslant 2j+O(\varepsilon),$$

whether (1R) or (2R) is used. In fact by carrying out a very long and tedious induction proof it is possible to show that both $|\alpha_j|$ and $|\beta_j|$ are less than $1+O(\varepsilon)$ for all four methods (see Paige, 1971), but such a refinement is not necessary here.

Loss of orthogonality can occur as follows. If $\mathbf{w}_j$ in (3R) is small as a result of cancellation then γ_{j+1} will be small in (4R) and the $O(\varepsilon)$ error in $\mathbf{w}_j$ will be greatly magnified in (5R), causing $\mathbf{v}_{j+1}$ to be very different from the desired value. It is clear that this loss of orthogonality is unavoidable in any of the algorithms, but nevertheless important relations involving $\mathbf{v}_j^T\mathbf{v}_{j+1}$ still hold. First if (1R) is used it follows from (3R), (5R), and (8R) that

$$\gamma_{j+1}\mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j-\alpha_j\mathbf{v}_j-\beta_j\mathbf{v}_{j-1}+O(\varepsilon) \tag{9}$$

$$\therefore \quad \gamma_{j+1}\mathbf{v}_j^T\mathbf{v}_{j+1} = -\beta_j\mathbf{v}_{j-1}^T\mathbf{v}_j+O(\varepsilon)$$

$$= \sum_{r=2}^{j} \frac{\beta_j}{\gamma_j}\cdot\frac{\beta_{j-1}}{\gamma_{j-1}}\cdots\frac{\beta_r}{\gamma_r}\cdot O(\varepsilon)+O(\varepsilon). \tag{10}$$

In fact the more precise analysis given by Paige (1971) shows that each $O(\varepsilon)$ in (10) is bounded in modulus by $2(n+4)\varepsilon\|\mathbf{A}\|$ for ordinary accumulation of inner-products. If instead double length accumulation is used, n may effectively be replaced by 1.

Next if (2R) is used then from (3R) and (5R)

$$\gamma_{j+1}\mathbf{v}_{j+1} = \mathbf{u}_j-\alpha_j\mathbf{v}_j+O(\varepsilon) \tag{11}$$

$$\therefore \quad \gamma_{j+1}\mathbf{v}_j^T\mathbf{v}_{j+1} = O(\varepsilon) \tag{12}$$

and here $O(\varepsilon)$ has as good a bound as each $O(\varepsilon)$ in (10). Clearly (12) holds for both A(2, 6) and A(2, 7), and since $\beta_j = \gamma_j$ in A(1, 7), (10) has the same form as (12) for this algorithm too, thus if in this step there has been very little cancellation so that $\gamma_{j+1} = O(1)$, it follows that the orthogonality of $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ is quite satisfactory, *regardless of any previous cancellation*. This does not follow for A(1, 6), for although β_j and γ_j are theoretically equal they can be orders of magnitude different if (6R) is used, leading to unnecessary loss of orthogonality in (10). To understand this, suppose that everything up to and including $\mathbf{A}\mathbf{v}_{j-1}$, α_{j-1}, β_{j-1} is known accurately, and then rounding errors occur in the subtractions (8R) or (3R), but not in the normalization. If $\bar{\gamma}_j$, $\bar{\mathbf{v}}_j$ are the computed values and γ_j, $\mathbf{v}_j$ the true values, then

$$\bar{\gamma}_j\bar{\mathbf{v}}_j = \mathbf{A}\mathbf{v}_{j-1}-\alpha_{j-1}\mathbf{v}_{j-1}-\beta_{j-1}\mathbf{v}_{j-2}+O(\varepsilon) = \gamma_j\mathbf{v}_j+O(\varepsilon). \tag{13}$$

Now even if no errors occur in (6)

$$\bar{\beta}_j = \mathbf{v}_{j-1}^T\mathbf{A}\bar{\mathbf{v}}_j$$

$$\therefore \quad \bar{\gamma}_j\bar{\beta}_j = \gamma_j\beta_j+O(\varepsilon) = \gamma_j^2+O(\varepsilon) \tag{14}$$

and since $\bar{\gamma}_j^2 = \gamma_j^2+\gamma_jO(\varepsilon)+O(\varepsilon^2)$ it follows that

$$\frac{\bar{\beta}_j}{\bar{\gamma}_j} = \frac{1+O(\varepsilon)/\gamma_j^2}{1+O(\varepsilon)/\gamma_j+O(\varepsilon^2)/\gamma_j^2}\cdot \tag{15}$$

Now if $\gamma_j < O(\varepsilon)$, this would give a satisfactory conclusion to the algorithm, so it can be seen that the denominator here will always be $O(1)$ when it matters. However it is only necessary for γ_j^2 to be much less then $O(\varepsilon)$ and the numerator could be far in excess of 1, and so no respectable *a priori* bound on orthogonality can follow from (10).

Note that the factor (15) appears in all subsequent expressions (10) for orthogonality, and so orthogonality having once been lost in this sense is unlikely to be regained. This is observed in practice, for example if a computer has a precision of $\varepsilon = 2^{-t}$ and a cancellation of more than $t/2$ binary figures occurs with A(1, 6), then orthogonality between consecutive vectors is usually lost and stays lost, the eigenvalues which have not already converged then do not converge properly, up to $t/2$ of their last figures wandering about aimlessly with each further step.

In applying the Lanczos algorithm it is often found that a cancellation of $t/2$ or more figures never occurs, and in such cases it might be hoped that A(1, 6) would perform satisfactorily. Such is not the case, as there is a further significant fault of A(1, 6) that is also shared by A(2, 6). The situation represented in (14) is a result of (6), and so applies to both the above algorithms. The matrix **T** in this case is not likely to be symmetric, but has the same eigenvalues as the symmetric matrix with the same diagonal but next-to-diagonal elements $(\bar{\gamma}_j\bar{\beta}_j)^{\frac{1}{2}}$. Theoretically these should be γ_j, but from (14) if $\gamma_j^2 > O(\varepsilon)$

$$(\bar{\gamma}_j\bar{\beta}_j)^{\frac{1}{2}} = \gamma_j[1+O(\varepsilon)/\gamma_j^2] = \gamma_j+O(\varepsilon)/\gamma_j,$$

while if $\gamma_j^2 \leqslant O(\varepsilon)$ the right hand side becomes $O(\varepsilon^{\frac{1}{2}})$. Thus if there are s figures of cancellation, i.e. $\gamma_j = O(2^{-s})$, an extra s figures are lost in the next-to-diagonal elements of the equivalent tri-diagonal matrix, up to a maximum of about $t/2$. This has no deleterious effect on well separated eigenvalues of **T** (see Wilkinson, 1965: 312), but close eigenvalues are likely to be in error by up to $O(\varepsilon^{\frac{1}{2}})$ if either A(1, 6) or A(2, 6) is used. This conclusion is supported by the computational results at the end of this paper where it is seen that close eigenvalues of **A** are given inaccurately by A(1, 6) and A(2, 6).

Thus A(1, 6), which appears to be the most popular algorithm in the literature, turns out to have two very significant deficiencies: occasional failure to converge, and poor separation of close eigenvalues. The modified Schmidt orthogonalization equivalent of this, A(2, 6), is not so bad but still suffers the second of these deficiencies. It now only remains to show that A(1, 7) and A(2, 7) do not suffer from this second loss of accuracy. This follows immediately, for if (7R) is used on the case represented by (13) then $\bar{\gamma}_j\|\bar{\mathbf{v}}_j\| = \gamma_j+O(\varepsilon)$, and even with errors in the normalization it follows that $\|\bar{\mathbf{v}}_j\| = 1+O(\varepsilon)$, so that the next-to-diagonal elements of **T** (which is automatically symmetric with these two algorithms) are in error by at most $O(\varepsilon)$.

This analysis has then shown that not only are A(1, 7) and A(2, 7) the more efficient algorithms, but they also do not suffer the deficiencies outlined above that seriously limit the accuracy of A(1, 6) and A(2, 6).

There remains one other possible algorithm, this is obtained by using (1) to compute α_j and then taking $\beta_j = \mathbf{v}_{j-1}^T(\mathbf{A}\mathbf{v}_j-\alpha_j\mathbf{v}_j)$ etc. A swift analysis shows that (10), (13), (14), and so (15) hold in this case, thus this algorithm has all the faults of A(1, 6) including the extra computation and storage, and can be discarded.

3. Computational Results

The following computations were carried out on the University of London Atlas computer which has $\varepsilon = 8^{-12} \doteq 1{\cdot}5 \times 10^{-11}$; ordinary accumulation of inner-products was used. The eigenvalues were found using the *tql1* procedure (Wilkinson & Reinsch, 1971), the possible error of each eigenvalue then being bounded by $s\varepsilon\|\mathbf{T}\|.O(1)$ if s iterations were needed.

TABLE 1

The eigenvalues computed from 20 steps of each of four variants of the Lanczos method applied to the Rosser matrix

True roots	A(1, 6)	A(1, 7)	A(2, 6)	A(2, 7)
−1020·0490184	−1020·0490190	−1020·0490188	−1020·0490189	−1020·0490189
	−1020·0490189	−1020·0490187	−1020·0490189	−1020·0490189
	−1020·0490187	−1020·0490186	−1020·0490187	−1020·0490187
	−1020·0490183	−1020·0490184	−1020·0490186	−1020·0490186
		−987·2137098		
0·0	−0·0000007	−0·0000002	−0·0000002	−0·0000004
	−0·0000005	−0·0000001	0·0000001	−0·0000003
	−0·0000002	0·0000001	0·0000005	−0·0000002
0·0980486	0·0980485	0·0980484	0·0980477	0·0980483
	0·0980487	0·0980485	0·0980482	0·0980483
	0·0980490	0·0980488	0·0980485	0·0980484
1000·0	999·9999890	999·9999998	1000·0000000	1000·0000000
twice	999·9999965	999·9999999	1000·0000006	1000·0000000
	1000·0000000	1000·0000000	1000·0000020	1000·0000000
				1000·0152578
1019·9019514	1019·9006454	1019·9019513	1019·9019034	1019·9019514
	1019·9014508	1019·9019514	1019·9019334	1019·9019514
	1019·9186942		1019·9329014	
1020·0	1019·9996229	1019·9999998	1019·9999628	1020·0000000
	1020·0006577	1020·0000000	1020·0000199	1020·0000000
1020·0490184	1020·0491866	1020·0490184	1020·0490238	1020·0490184
	1020·0504332	1020·0490184	1020·0490953	1020·0490185

In order to exhibit the differences in performance of the four analysed algorithms they were all applied to the 8×8 Rosser matrix (see Westlake, 1968: 150) using an initial vector with equal elements. This matrix has interestingly grouped eigenvalues as can be seen in the first column of Table 1. In order to exhibit an extraordinary property of the Lanczos process 20 steps of each algorithm were carried out and the eigenvalues tabulated. The process does not curtail, instead eigenvalues continue to converge so that $\mathbf{T}$ can have several nearly equal eigenvalues corresponding to one *simple* eigenvalue of $\mathbf{A}$. Table 1 shows the definite superiority of A(1, 7) and A(2, 7)

over the other two which lose up to five more decimal figures of accuracy for close roots, A(1, 6) being the worst as was expected from the error analysis. The accuracy for both A(1, 7) and A(2, 7) was quite remarkable, being within $5 \times 10^{-10}\|\mathbf{A}\|$ for the converged eigenvalues; this is about the same size as the bound given for *tql*1. The superiority of these two algorithms was even more marked for further iterations, even well separated eigenvalues being found distinctly more accurately by them.

The algorithms A(1, 7) and A(2, 7) are thus seen to perform very well for the Rosser matrix and there is nothing to choose between them. In order to examine their suitability for very large sparse matrices they were both applied to the matrix **A** of order 1000 derived from the usual five-point approximation with unit step-length to the Laplace operator for a class of functions that vanish on the boundary of a rectangular region of sides 51 and 21. The initial vector contained equal components of all eigenvectors and 600 steps of the algorithms were carried out. Again ordinary accumulation of inner-products was used.

The results are too numerous to present in a table and a verbal description for A(1, 7) will suffice; A(2, 7) exhibited occasional minor differences but again the two algorithms were almost identical for accuracy.

92 of the lowest eigenvalues of $\mathbf{T}_{600}$ agreed with the corresponding eigenvalues of **A** in at least the ninth decimal place; these true eigenvalues were those from the lowest 0·0261316900 up to 0·7718718678, and included 58 different eigenvalues. From the smallest upwards the numbers of each of these were

5, 5, 4, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2

with the remaining 38 appearing once only. The eigenvalues of **A** from 7·2281281322 to the largest one 7·9738683100 were also given accurately in at least the ninth decimal place, and numbering from the largest down the numbers of each different eigenvalue were the same as for the lower group. Outside these two groups the accuracy fell off quite quickly, so that about 370 of the remaining 416 eigenvalues of $\mathbf{T}_{600}$ approximated eigenvalues of **A** to only two or three decimal places.

The absolute errors in the 184 converged eigenvalues of $\mathbf{T}_{600}$ were less than 3×10^{-10}. This is a really remarkable result, as it means that these eigenvalues of $\mathbf{T}_{600}$ represent those of **A** more accurately than the eigenvalues of $\mathbf{T}_{600}$ can be computed using *tql*1 with the same precision. *tql*1 had to be used with increased precision to show the true accuracy of the algorithm A(1, 7).

Finally to give a rough idea of the speed of the process, both A(1, 7) and A(2, 7) took about the same time to reduce the 1000 by 1000 Laplace matrix to $\mathbf{T}_{600}$ as it took *tql*1 to find all the eigenvalues of $\mathbf{T}_{600}$ using the same precision, i.e. about six minutes. Both these Lanczos algorithms were extravagantly programmed with no consideration for speed. The speed of convergence of the Lanczos process can be gauged from the fact that if the eigenvalues of **A** are ordered $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_{1000}$ then after 100 steps λ_1 was given accurately to 9 decimal places, λ_2 to 7, λ_3 to 5, λ_4 to 3, λ_{997} to 3, λ_{998} to 5, λ_{999} to 7, and λ_{1000} to 9 decimal places.

4. Eigenvectors and Eigenvalue Multiplicities

Some eigenvectors of **A** can be found from the algorithm A(1, 7) or A(2, 7) by computing the corresponding orthonormal eigenvectors $\mathbf{q}_i$ such that $\mathbf{T}_k\mathbf{q}_i = \lambda_i\mathbf{q}_i$,

then reproducing $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ in order to form the required approximate eigenvectors $\mathbf{z}_i \equiv \mathbf{V}\mathbf{q}_i$ of $\mathbf{A}$. These eigenvectors may then be used to obtain a satisfactory indication of the multiplicities of some of the eigenvalues of $\mathbf{A}$ as follows.

Suppose a group of s very close eigenvalues $\lambda_{i+1}, \ldots, \lambda_{i+s}$ have converged and are well separated from the remaining eigenvalues of $\mathbf{T}_k$, then these must approximate an eigenvalue, or a group of very close eigenvalues, of $\mathbf{A}$; suppose there are r close eigenvalues of $\mathbf{A}$ in this group and these are well separated from the remainder, the corresponding eigenvectors then form an r dimensional invariant subspace of $\mathbf{A}$. Note that as is usually the case it will be impossible to distinguish multiple eigenvalues of $\mathbf{A}$ from a group of very close eigenvalues. The approximate eigenvectors $\mathbf{z}_{i+1}, \ldots, \mathbf{z}_{i+s}$ can be computed, normalized, and checked for accuracy by forming $f_j = \mathbf{A}\mathbf{z}_j - \lambda_j \mathbf{z}_j$. If these $f_{i+1}, \ldots, f_{i+s}$ are small the $\mathbf{z}_{i+1}, \ldots, \mathbf{z}_{i+s}$ must lie approximately in the above invariant subspace of $\mathbf{A}$. The rank of $\mathbf{Z} \equiv [\mathbf{z}_{i+1}, \ldots, \mathbf{z}_{i+s}]$ then gives a good indication of the dimension r of this invariant subspace of $\mathbf{A}$. For suppose $\mathbf{Z}$ has p negligible singular values with the remaining $s-p$ not very small, then $r \geqslant s-p$. If $p = 0$ this is all that can be concluded, however if $p > 0$ then the probability is very high that $r = s-p$, the certainty increasing with increase in p. This uncertainty is common whenever partial eigensystems only are computed. Thus the tendency of the Lanczos method to produce more solutions than in fact exist can be used to advantage here to check the multiplicity of solutions.

In practice it has been found that normalizing the $\mathbf{z}_j$ and computing $\mathbf{Z}^T\mathbf{Z}$ has led to the correct results on multiplicity, even though some information is lost in this approach. When $\lambda_{i+1}, \ldots, \lambda_{i+s}$ correspond to a simple eigenvalue of $\mathbf{A}$, every element of $\mathbf{Z}^T\mathbf{Z}$ has been found to have a modulus of unity. Otherwise the correct information has been found by taking p to be the number of computed eigenvalues of $\mathbf{Z}^T\mathbf{Z}$ that are not very much greater than the computer precision.

5. Conclusion

To sum up, this paper provides very convincing arguments for using the Lanczos process in the form of algorithms A(1, 7) and A(2, 7) in an iterative manner to find several extreme eigenvalues of very large sparse symmetric matrices, both accurately and quickly. The algorithms are very easily programmed, being so simple and repetitive in nature, and seem easily amenable to parallel processing for even greater speed. A satisfactory criterion for convergence of λ_i and $\mathbf{z}_i = \mathbf{V}\mathbf{q}_i$, where $\mathbf{T}_k\mathbf{q}_i = \lambda_i\mathbf{q}_i$, can be found by computing the last element q_{ki} of $\mathbf{q}_i$, then

$$\|\mathbf{A}\mathbf{z}_i - \lambda_i \mathbf{z}_i\| \leqslant \beta_{k+1}|q_{ki}| + O(\varepsilon).$$

Note that $q_{k1}, \ldots, q_{kk}$ can easily be computed while finding the eigenvalues of $\mathbf{T}_k$ *without having to compute the full vectors* $\mathbf{q}_1, \ldots, \mathbf{q}_k$.

Although the error analysis here has shown which are the best algorithms, it has not shown that these necessarily give accurate results, nor has it shown that the remarkably fast convergence found in practice will necessarily occur. The excellent accuracy of the successful algorithms has been proven by Paige (1971), and it is hoped that this, with a proof of the rapid convergence, will be published shortly. It would also be useful to devise a means of obtaining the multiplicities of eigenvalues of $\mathbf{A}$ without having to compute the corresponding eigenvectors, though this may not be possible.

I would like to thank Dr. M. J. M. Bernal for reading the script and making some useful comments.

References

KANIEL, S. 1966 *Maths Comput.* **20,** 369–378.

LANCZOS, C. 1950 *J. Res. natn. Bur. Stand.* **45,** 255–282.

PAIGE, C. C. 1971 *The computation of eigenvalues and eigenvectors of very large sparse matrices.* University of London Ph.D. thesis.

WESTLAKE, JOAN R. 1968 *A handbook of numerical matrix inversion and solution of linear equations.* New York: John Wiley.

WILKINSON, J. H. 1965 *The algebraic eigenvalue problem.* Oxford: Clarendon Press.

WILKINSON, J. H. & REINSCH, C. (Eds) 1971 *Handbook for automatic computation,* **2.** Berlin: Springer Verlag.