JOURNAL OF COMPUTATIONAL PHYSICS 53, 82–89 (1984)

A Lanczos-Type Algorithm for the Generalized Eigenvalue Problem $Ax = \lambda Bx$

T. Z. KALAMBOUKIS

Department of Mathematics, University of Crete, Iraclion, Crete, Greece

Received January 4, 1983

This paper describes a generalization of the Lanczos algorithm to make it applicable directly to the generalized matrix eigenvalue problem $Ax = \lambda Bx$, where A, B are real symmetric matrices of high order ($n \times n$) and B is positive definite. The algorithm is based on the construction of a sequence of B-orthonormal vectors b_i; if $U_p = (b_1, ..., b_p)$ then the eigenvalues of the $p \times p$ matrix $U_p^t A U_p$ ($p \ll n$) approximate some of the extreme eigenvalues of the original problem, $Ax = \lambda Bx$.

1. INTRODUCTION

Frequently problems in quantum mechanics, for example, configuration interaction calculations of atomic or molecular correlation energies, or numerical solution of vibrational problems using the method of finite elements, lead to the computation of a small number of eigenvalues and associate eigenvectors, usually the extreme ones, of the problem

$$Ax = \lambda Bx \tag{1.1}$$

where A and B are real symmetric and sparse matrices of high order and B is positive definite. In these cases classical diagonalization methods [8] are impractical.

As it is known Lanczos' algorithm cannot be applied to (1.1) unless matrix B can be factored, $B = LL^t$ [4]. Then the usual Lanczos algorithm can be applied straightforwardly to the implicitly defined matrix $L^{-1}AL^{-t}$. In cases where matrix B cannot be factored easily, B being of very high order, we can again apply Lanczos' scheme to the implicitly defined matrix $B^{-1}A$ [4]. However, in this case we have to solve a linear system for matrix B in every iteration, which is not convenient. Of interest is the iterative method proposed by Nesbet [3] for finding the extreme eigensolution based on the minimization of the Rayleigh quotient which was later improved by Shavitt *et al.* [6] for the calculation of several of the extreme eigensolutions. Also a method developed by Davidson [1] has been applied to the generalized eigenvalue problem in [2] for symmetric matrices and recently in [5] for unsymmetric matrices with real eigenvalues.

In what follows we describe an explicitly orthogonalized Lanczos-type method for

0021-9991/84 $3.00

constructing a sequence of B-orthonormal vectors and approximate the eigenvalues of the original problem (1.1) from the eigenvalues of the restriction of A into the subspace spanned by those vectors. That is, if U_p is the $n \times p$ matrix whose columns are the vectors derived in the course of the process, such that $U_p^t B U_p = I$, then the eigenvalues of the $p \times p$ matrix $A_p = U_p^t A U_p$ approximate some of the extreme eigenvalues of (1.1). The method developed here can be used to approximate both the largest and smallest eigenvalues and requires no inversion avoiding any complications of an ill-conditioned B. Although the process should theoretically produce the exact solution in n steps it is well known that a satisfactory accuracy is often achieved for values of p much less than n. The only drawback is that the vectors b_i have to be kept somewhere in an auxiliary storage device for the formation of the new vector in every iteration and the elements of A_p. Because A_p is a full matrix its dimension p cannot be chosen as large as to provide a desired accuracy. The only way to overcome this difficulty is to use the algorithm in an iterative manner restarting after a certain number of iterations. The method works with the original matrices unchanged so their sparsity is taken into account to full extent. Hence no rounding errors are introduced by any transformation. Rounding errors due to the recursive computation can be eliminated, if necessary, by restarting the process.

Good approximations to the eigenvectors can be obtained by using the vectors b_i, $i = 1,..., p$, which are held in backing store.

In the following we shall use the B-norm for a vector x, defined by $\|x\|_B = (x, Bx)^{1/2}$.

In Section 2 we give a description of the algorithm and in Section 3 some numerical results to demonstrate the behaviour of the method in practice.

2. Description of the Algorithm

Given a real symmetric matrix A of order $n \times n$, a positive definite symmetric matrix B of the same order and a system of orthonormal vectors $U_p = (b_1,..., b_p)$ $(n \times p)$, the projection method on the subspace spanned by U_p [2] is approximating an eigensolution λ, x of the problem (1.1) by $\lambda^{(p)}$, $x^{(p)}$ such that $x^{(p)} \in \mathrm{Span}(U_p)$ and $Ax^{(p)} - \lambda^{(p)} B x^{(p)}$ is orthogonal to all b_j, $j = 1,..., p$. The eigenvector $x^{(p)}$ is then writen $x^{(p)} = U_p y^{(p)}$, where $\lambda^{(p)}$, $y^{(p)}$ are the eigenelements of the generalized eigenvalue problem

$$(A_p - \lambda^{(p)} B_p)\, y^{(p)} = 0 \tag{2.1}$$

where the matrices A_p, B_p of order $p \times p$ are defined by $A_p = U_p^t A U_p$ and $B_p = U_p^t B U_p$. The solutions $\lambda^{(p)}$, $y^{(p)}$ of (2.1) are usually referred to as the Ritz values and Ritz vectors of (1.1). However, in applications it is more convenient if B_p reduces to the unit matrix, in other words, if the vectors b_i, $i = 1,..., p$, are B-orthonormal. In this case the projection method on U_p where b_i are now B-orthonormal aims to approximate λ, x by a pair $\lambda^{(p)}$, $x^{(p)}$ such that $x^{(p)} \in \mathrm{Span}(U_p)$

and $(A-\lambda^{(p)}I)\,x^{(p)}$ is B-orthogonal to b_i, $i=1,\ldots,p$. Once again $x^{(p)}=U_p y^{(p)}$, where $\lambda^{(p)}$, $y^{(p)}$ are now the eigenelements of the single $p\times p$ eigenvalue problem

$$(A_p-\lambda^{(p)}I)\,y^{(p)}=0.$$

The above description can be stated in an algorithm as follows.

Algorithm I. Start with a vector $b_1\neq 0$ such that $b_1^tBb_1=1$. For $j=1,2,\ldots$ do:

(a) Solve the $j\times j$ eigenproblem $A_j y^{(j)}=\lambda^{(j)}y^{(j)}$.

(b) Form the new vector as the residual

$$q_{j+1}=AU_j y^{(j)}-\lambda^{(j)}U_j y^{(j)}. \tag{2.2}$$

(c) B-orthogonalize q_{j+1} to the previous vectors.

(d) Normalize to B-norm one, $b_{j+1}=q_{j+1}/\|q_{j+1}\|_B$.

The B-orthogonalization step of the new vector to the previous ones is achieved by the generalized Gram-Schmidt orthogonalization process

$$b_{j+1}\leftarrow q_{j+1}-\sum_{i=1}^{j}(b_i^tBq_{j+1})\,b_i. \tag{2.3}$$

However, as we shall show, this algorithm can be considerably simplified by forming the new vector in such a way that q_{j+1} can be calculated independently of $\lambda^{(j)}$, $y^{(j)}$.

In Algorithm I for $j=1$ starting with a vector b_1 ($b_1^tBb_1=1$), the vector q_2 is given from step (b) by

$$q_2=Ab_1-a_{11}b_1,\qquad \text{where } a_{11}=b_1^tAb_1$$

and B-orthogonalizing q_2 with respect to b_1 we obtain

$$q_2\leftarrow q_2-(b_1^tBq_2)\,b_1=Ab_1-h_{11}b_1\qquad \text{with}\quad h_{11}=b_1^tBAb_1.$$

Normalizing now q_2 to B-norm one we have

$$h_{21}b_2=Ab_1-h_{11}b_1,\qquad \text{and}\quad h_{21}=b_2^tBAb_1.$$

For $j=2$, $q_3=AU_2y^{(2)}-\lambda^{(2)}U_2y^{(2)}$. B-orthogonalizing from (2.3) q_3 with respect to b_1, b_2 and using the fact that $A_2y^{(2)}=\lambda^{(2)}y^{(2)}$ we obtain

$$q_3=(Ab_2-h_{12}b_1-h_{22}b_2)\,y_2^{(2)}$$

where $y_2^{(2)}$ is the second component of the vector $y^{(2)}$. Finally, B-normalizing we have that

$$h_{32}b_3=Ab_2-h_{12}b_1-h_{22}b_2\qquad \text{with}\quad h_{ij}=b_i^tBAb_j.$$

In the same way by an induction argument it is easy to show that

$$h_{j+1j}b_{j+1} = Ab_j - \sum_{i=1}^{j} h_{ij}b_i. \tag{2.4}$$

So the new vectors b_{j+1} from (2.4) theoretically are identical to those given in Algorithm I. From (2.4) we can easily verify that b_{j+1} is B-orthogonal to all previous vectors b_i, $i = 1,..., j$, thus the B-orthogonalization step should not be necessary. However, loss of significant figures in the subtraction step of (2.4) means that the new vector after a few iterations is not longer B-orthogonal to the previous ones so a re-B-orthogonalization process would be necessary in practice. Another advantageous feature of writing the algorithm in this way is that we do not have to solve the small $j \times j$ eigenproblem (2.2) in every step which occupies a considerable amount of computer time as j increases. We solve the small eigenproblem after a certain number of iterations as a test for the termination of the algorithm. Summarising, the algorithm can be described as follows.

Algorithm II. Initialization: start with $b_1 \neq 0$ such that $b_1^t B b_1 = 1$, calculate $a_{11} = b_1^t A b_1$. Iteration: for $j = 1, 2,...$ do:

(a) Form $w = Ab_j - \sum_{i=1}^{j} h_{ij}b_i$, $h_{ij} = b_i^t BAb_j$.

(b) Re-B-orthogonalize w to b_i, $i = 1,..., j$.

(c) Normalize, $b_{j+1} = w/\|w\|_B$.

(d) Form $a_{i,j+1} = b_i^t A b_{j+1}$ for $i = 1,..., j+1$.

Relation (2.4) when applied for $j = 1,..., p$ can be written in matrix form as

$$AU_p = U_p H + h_{p+1p}b_{p+1}e_p^t \tag{2.5}$$

where H is a $p \times p$ matrix of upper Hessenberg form with elements $h_{ij} = b_i^t BAb_j$ and e_p is the unit vector with one in the pth position and zero elsewhere. In actual computation due to cancellation the vectors b_i lose their B-orthogonality and in (2.4) an error vector $0(\varepsilon)$ must be added. For this particular behaviour we suggest re-B-orthogonalization.

From (2.4) and step (d) of Algorithm II, we also see that the calculation of the new vector requires that all the previous vectors have to be kept in auxiliary store and fetched into fast computer memory when needed. As j increases the process of computing the b_i's becomes more expensive and memory demanding so the number p of steps is limited. However, it is possible to use the algorithm in an iterative manner. After having approximated a solution $x^{(p)}$ for a certain p we may find that the accuracy is unsatisfactory. In such a case we can use $x^{(p)}$ as starting vector and restart the algorithm. This process can be repeated as many times as needed to ensure a satisfactory accuracy. However, in physical problems the matrix–vector multiplications usually dominate in cost all the other operations in each step so it may be advantageous to choose p as large as possible. So at the end of Algorithm II

a test can be incorporated which can be used to determine whether the process must be stopped or restarted, that is: If $\lambda^{(p)}$, $x^{(p)}$ are sufficiently accurate then stop or else take $b_1 = x^{(p)}$ and restart from the beginning. The process will stop when the B-norm of the residual is less than a tolerance, that is,

$$\|r\|_B = \|AU_p y^{(p)} - \lambda^{(p)} U_p y^{(p)}\|_B = \left\| \left(Ab_p - \sum_{i=1}^{p} h_{ip} b_i \right) y_p^{(p)} \right\|_B$$

$$= \|h_{p+1p} b_{p+1} y_p^{(p)}\|_B = |h_{p+1p}| \cdot |y_p^{(p)}|.$$

From the last relation we see that we can terminate the process when the last component of vector $y^{(p)}$ is absolutely less than a tolerance even if h_{p+1p} is of moderate size.

We shall show here the assertion made in the introduction that the eigenvalues of A_p approximate those of (A, B). The proof is based on the Courant–Fischer theorem and is given in the following theorem.

THEOREM. *If* $\lambda_j(A_p) = \lambda_j'$ *and* $\lambda_j(A, B) = \lambda_j$ *then* $\lambda_j' \simeq \lambda_j$ *for* $j = 1,..., p$, *where* $A_p = U_p^t A U_p$ *and* $U_p^t B U_p = I$.

Before we proceed to the proof of the theorem we give the Courant–Fischer theorem for the generalized case which states that:

For every $j = 1,..., n$ and for any linear subspace $U \subseteq R^n$ with $\dim(U) = n + 1 - j$ if the eigenvalues of (A, B) are ordered as $\lambda_1 \geqslant \cdots \geqslant \lambda_n$ then

$$\lambda_j = \min \max_{0 \neq x \in U} \frac{x^t A x}{x^t B x} \tag{2.6}$$

where $U = \{x \in \mathbb{R}^n : (x, u_i)_B = 0,\ i = 1,..., j-1\},\ u_i \neq 0$. Relation (2.6) takes its minimum value only if the u_i's span an eigenspace of (1.1) corresponding to the eigenvalues $\lambda_1,..., \lambda_{j-1}$.

Proof of the Theorem. We shall show that

$$\lambda_j' = \min \max_{0 \neq x \in U} \frac{x^t A x}{x^t B x}$$

where U is as above. Applying Courant–Fischer theorem to A_p, for a fixed p, we have

$$\lambda_j' = \min \max_{0 \neq y \in W} \frac{y^t A_p y}{y^t y}$$

with $W = \{y \in \mathbb{R}^p : (y, q_i) = 0, i = 1,..., j-1\},\ q_i \neq 0$. Then

$$\lambda_j' = \min \max_{0 \neq y \in W} \frac{(U_p y)^t A (U_p y)}{(U_p y)^t B (U_p y)} = \min \max_{0 \neq x \in U} \frac{x^t A x}{x^t B x}.$$

The last equation holds if we let $u_i = U_p q_i$ which completes the proof.

As we have mentioned, if U_p was an eigenspace then $\lambda_j' = \lambda_j$ but in practice U_p is very far from such a space so λ_j' only approximate λ_j or may even be different from them.

3. Numerical Results and Conclusions

In order to reduce the amount of input–output operations the trial vectors b_i and also the Bb_i's are kept in core and only the matrix resides in external storage. These matrices are not declared explicitly but they are represented in a matrix–vector multiplication form. We thus reduce the storage and minimize the number of operations since zeros are not taken into account.

Storage requirements are that only four vectors $n \times 1$ need to be used in core memory plus the necessary storage for the small eigenproblem. The storage of the A_p symmetric matrix requires $p(p+1)/2$ locations, stored in a one-dimensional array in its upper triangular form. If α and β are the average number of nonzero elements per row of the matrices A, B and p the iteration number, the number of multiplications–divisions is $n(\alpha+\beta) + 4np + 2n$.

In our examples Jacobi's method has been used for the diagonalization of the small $p \times p$ eigenproblem A_p.

When the matrices A, B commute then formula (2.4) can be simplified in a Lanczos-type three-term recurrence formula as

$$h_{p+1p} b_{p+1} = A b_p - h_{pp} b_p - h_{p-1p} b_{p-1} \tag{3.1}$$

with h_{ij} as before.

All the experiments described here have been performed on a CDC CYBER 72 machine using a mantissa of 48 bits. We shall give two examples from differential equations which, although they are of relatively small order, illustrate well the principle of the algorithm. However, numerical examples tested for large matrices with elements randomly chosen show the same behaviour.

Consider the problem of a vibrating string of length one fixed at the ends. This problem in certain cases leads to the ordinary differential equation

$$y'' + Ay = 0 \qquad \text{with} \quad y(0) = 0, \quad y(1) = 0.$$

To find the smallest eigenvalue λ_1 we have the variational problem of minimizing the expression, called Rayleigh quotient, that is,

$$R = \int_0^1 y'(x)^2 \, dx \Big/ \int_0^1 y(x)^2 \, dx = \min \tag{3.2}$$

for all functions y. For intermediate eigenvalues, say, the jth-largest eigenvalue, the problem will be to minimize the max of (3.2) for those functions y which are subjected to the condition of being orthogonal to the $(j-1)$ first eigenfunctions.

TABLE I

Lowest-Lying Eigenvalues for Example 2, Order of Matrices 77×77

k	Exact	A_{30} (linear splines)
1	3.564024	3.6021696
2	6.853892	6.9868646
3	10.96623	11.489541
4	12.33701	12.885911
5	14.25610	14.874238
6	19.73921	20.793830

Approximating the two integrals in (3.2) using linear or quadratic spline functions we obtain a generalized eigenvalue problem. In the following we shall attempt to solve this problem for different kinds of approximations of the function y. In all our examples we start the algorithm using a vector with equal components.

EXAMPLE 1. We shall approximate the integrals in (3.2) using linear spline functions with distances of the mesh points $h = 1/41$ [7]. We have 40 interior points so we obtain a 40×40 system. The smallest eigenvalue of A_{20} was 9.874434 with $\|r\|_B = 0.3 \times 10^{-15}$. The exact eigenvalue is $\pi^2 = 9.8696044$. The difference is due to the linear spline functions used to approximate y. In the case that quadratic spline functions were used the calculated value of the smallest eigenvalue of A_{15} was 9.8696032 with $\|r\|_B = 0.3 \times 10^{-6}$ and $|y_{15}^{(15)}| = 0.1 \times 10^{-5}$.

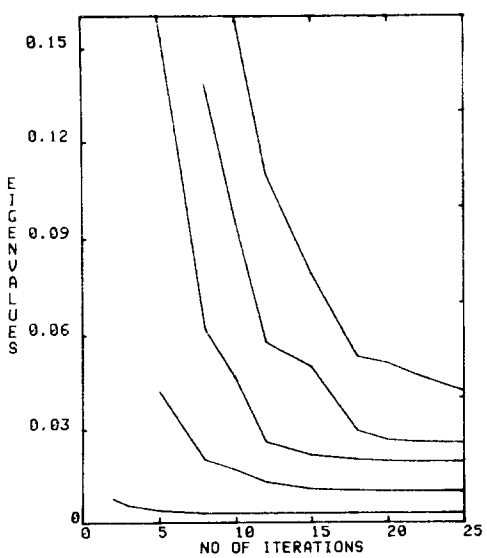

FIG. 1. Convergence of the lowest-lying eigenvalues for the vibrating membrane problem, order of matrices 551×551.

EXAMPLE 2. In this example the same problem has been examined in two dimensions (vibrating membrane problem). Using a square grid with side length $h = 0.25$ and $h = 0.1$ we obtain two systems of order 77×77 and 551×551, respectively. For the 77×77 system the smallest eigenvalue has been found after only 15 iterations with $\|r\|_B = 0.4 \times 10^{-8}$ and $|y_{15}^{(15)}| = 0.4 \times 10^{-8}$. To find more eigenvalues we continued the iterations up to 30. In Table I we give a listing of the smallest eigenvalues of A_{30} together with the exact ones. Linear spline functions were used to approximate y. The difference in these results is mainly due to the spline fitting of the original differential equation. In Fig. 1 we give the convergence of the lowest eigenvalues for the 551×551 system.

The convergence properties of the algorithm have been studied by carrying out a series of test examples using different matrices. The dominant eigenvalues are not the only ones which converge rapidly. Actually best accuracy is first obtained for those eigenvalues which lie in the outer part of the spectrum. This generalizes the well-known property of the Lanczos method for symmetric matrices which states that the best accuracy is first achieved for the largest eigenvalues as well as the smallest.

REFERENCES

1. E. R. DAVIDSON, *J. Comput. Phys.* **17** (1975), 87.
2. T. Z. KALAMBOUKIS, *J. Phys. A* **13** (1980), 57.
3. R. K. NESBET, *J. Chem. Phys.* **43** (1965), 311.
4. B. N. PARLETT, "The Symmetric Eigenvalue Problem," Prentice–Hall, Englewood Cliffs, N. J., 1980.
5. S. RETTRUP, *J. Comput. Phys.* **45** (1982), 100.
6. I. SHAVITT, C. F. BENDER, A. PIPANO, AND R. P. HOSTENY, *J. Comput. Phys.* **11** (1973), 90.
7. H. R. SCHWARZ, *Comput. Methods Appl. Mech. Engrg.* **3** (1974), 11.
8. J. H. WILKINSON, "The Algebraic Eigenvalue Problem," Oxford Univ. Press (Clarendon), London/New York, 1965.