SU326P30-16

THE LANCZOS ALGOR ITHM FOR THE SYMMETRIC $Ax=\lambda Bx$ PROBLEM

BY

G. H. GOLUB

R. UNDERWOOD

J. H. WILKINSON

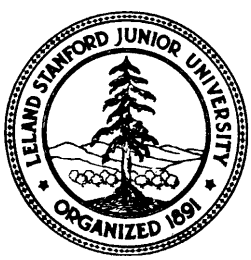
STAN-CS-72-270

MARCH 1972

COMPUTER SC IENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UN IVERS ITY



The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem

by

G. H. Golub*, R. Underwood*, and J. H. Wilkinson**

*Computer Science Department, Stanford University, Stanford, California 94305. The work of these authors was in part supported by the National Science Foundation and the Atomic Energy Commission.

*Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex,' England.

ABSTRACT

The problem of computing the eigensystem of $Ax = \lambda Bx$ when A and B are symmetric and B is positive definite is considered. A generalization of the Lanczos algorithm for reducing the problem to a symmetric tridiagonal eigenproblem is given. A numerically stable variant of the algorithm is described. The new algorithm depends heavily upon the computation of elementary Hermitian matrices. An ALGOL W procedure and a numerical example are also given.

The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem

by

G. H. Golub, R. Underwood, and J. H. Wilkinson

1. Theoretical Background

In many fields of work the solution of the eigenproblem

$$Ax = \lambda Bx \tag{1}$$

is required where A and B are symmetric and B is positive definite. This problem can be reduced to the standard symmetric eigenproblem by making use of the Cholesky factorization of B defined by

$$B = LL^T . \tag{2}$$

Equation (1) is then equivalent to

$$L^{-1}AL^{-T}(L^Tx) = \lambda(L^Tx) \tag{3}$$

and $L^{-1}AL^{-T}$ is a real symmetric matrix.

When A and B are narrow symmetric band matrices of high order this reduction has the disadvantage that $L^{-1}AL^{-T}$ is, in general, a full matrix. However, L itself is of band form and hence we can certainly multiply an arbitrary vector by $L^{-1}AL^{-T}$ in an economical manner. In fact, if we write

$$z = L^{-1}AL^{-T}x \tag{4}$$

then z can be determined in the steps

$$L^T y = x\,, \quad Ay = w\,, \quad L z = w \tag{5}$$

and in this way we can take full advantage of the band forms of L and A . The total number of multiplications in the determination of z is only marginally greater than in the determination of both Ax and Bx taking advantage of the band forms of A and B .

Now in the Lanczos algorithm for a symmetric matrix C , the only way in which C is used is in the pre-multiplication of vectors. The algorithm may be described as follows. Let x_1 be an arbitrary unit vector ($\|x_1\|_2 = 1$) ; then determine sequences of vectors y_r and x_r defined by

$$y_2 = Cx_1 - \alpha_1 x_1\,, \quad \gamma_2 x_2 = y_2\,, \quad \|x_2\|_2 = 1\,, \quad \gamma_2 > 0 \tag{6}$$

$$y_{r+1} = Cx_r - \alpha_r x_r - \beta_r x_{r-1}\,, \quad \gamma_{r+1} x_{r+1} = y_{r+1}\,, \quad \|x_{r+1}\|_2 = 1\,, \quad \gamma_{r+1} > 0 \tag{7}$$

where the sequence is continued until $y_{r+1} = 0$. The α_r and β_r are determined so that y_{r+1} is orthogonal to x_r and x_{r-1} and α_1 so that y_2 is orthogonal to x_1 . These relations ensure that $\gamma_r = \beta_r$ and y_{r+1} is automatically orthogonal to $x_1, x_2, \ldots, x_r$. Notice that when y_{r+1} has been determined, Equations (6) and (7) imply that

$$C[x_1, x_2, \ldots, x_r] = [x_1, x_2, \ldots, x_r]T_r + [0, 0, \ldots, 0, y_{r+1}] \tag{8}$$

where T_r is the tridiagonal matrix with diagonal elements equal to the a_1, superdiagonal elements equal to the β_i and subdiagonal elements equal to the γ_i. This is true even if α_i and β_i are chosen arbitrarily! If $y_{r+1} = 0$, then (8) gives

$$C[x_1, x_2, \ldots, x_r] = [x_1, x_2, \ldots, x_r]T_r \tag{9}$$

and provided only the x_i are independent, T_r gives r of the eigenvalues of C and enables us to compute the corresponding eigenvectors. In the Lanczos algorithm, the orthonormality of the x_i ensures their independence and also the symmetry of T_r.

Since y_{r+1} is orthogonal to the r orthogonal vectors $x_1, x_2, \ldots, x_r$, the process must terminate with y_{n+1} if it has not done so before. In fact, we may regard the Lanczos algorithm as a method of ensuring that the x_i are independent and that the process does terminate. When the Lanczos algorithm terminates before y_{n+1}, then we have

$$0 = y_{r+1} = Cx_r - a_r x_r - \beta_r x_{r-1} . \tag{10}$$

If we choose x_{r+1} to be any unit vector orthogonal to $x_1, \ldots, x_r$, then Equation (10) gives

$$Cx_r - \alpha_r x_r - \beta_r x_{r-1} = 0 = 0 \cdot x_{r+1} . \tag{11}$$

Hence we may take $\gamma_{r+1} = 0$ and continue the algorithm with x_{r+1}, This again ensures that subsequent x_i are orthogonal to all earlier x_j. We may restart as often as necessary until we finally reach the null y_{n+1}. If the termination takes place after $r_1, r_1+r_2, r_1+r_2+r_3, \ldots, r_1+r_2+\ldots+r_k$ steps then we have

$$C[x_1, x_2, \ldots, x_n] = [x_1, x_2, \bullet \; \boxtimes\Omega\& \qquad (12)$$

where T is the direct sum of k symmetric tridiagonalmatrices of orders $r_1, r_2, \bullet \; \boxtimes\Omega\&$. The eigensystem of T gives the eigensystem of c . Premature termination of the sequence should not be regarded as a breakdown; in fact it leads to a simplification since it is easier to find the eigenvalues and eigenvectors of several smaller tridiagonal matrices than of one large one. The only disadvantage is the necessity for determining the restarting vectors.

At first sight the Lanczos algorithm is very attractive whenever C is sparse (including, in particular, the case when $C = L^{-1}AL^{-T}$) . Except when restarting, we need only the two vectors x_r and x_{r-1} to determine y_{r+1} and x_{r+1} and hence requirements on the high-speed store appear to be very modest. Unfortunately, if the algorithm is carried out as described, the later x_r may be very far from orthogonal to the earlier ones. When this is true, we have no guarantee that y_{n+1} will be null to working accuracy. Moreover, we may get near linear dependency of $x_1, x_2, \ldots, x_r$ for some $r < n+1$.

This departure from orthogonality is sometimes said to be the result of accumulation of rounding errors, but this is very misleading. It occurs when there is a good deal of cancellation when computing y_{r+1} from $Cx_r - \alpha_r x_r - \beta_r x_{r-1}$. This cancellation can occur even in the determination of y_2 when C is of order 2 (say), and in this case one could scarcely speak of accumulation of rounding errors. Before discussing how to obtain a complete set of orthonormal x_i we consider

the implication of Equation (8) when rounding errors are taken into account. We now have

$$C[x_1,x_2,\ldots,x_r] = [x_1,x_2,\ldots,x_r]T_r + [0,0,\ldots,y_{r+1}] - [\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_r] \qquad (13)$$

where ε_i is defined by the relation

$$\gamma_{i+1}x_{i+1} = Cx_i - \alpha_i x_i - \beta_{i-1} + \varepsilon_i \ , \qquad (14)$$

the x_{i+1} denoting the computed x_{i+1} . If λ and z are an eigenvalue and eigenvector of T_r we have

$$T_r z = \lambda z \qquad (15)$$

and hence

$$C[x_1,x_2,\ldots,x_r]z = \lambda[x_1,x_2,\ldots,x_r]z + z_r y_{r+1} - [\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_r]z \qquad (16)$$

where the z_i are the components of z . Now z may be taken to be a unit vector and $[\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_r]z$ is therefore of the order of machepsx$\|C\|$ where macheps is the machine precision. Hence if $z_r y_{r+1}$ is also of the order of machepsx$\|C\|$ we have

$$Cw = \lambda w + e \qquad (17)$$

where $w = [x_1,x_2,\ldots,x_r]z$ and $\|e\|$ is of order machepsx$\|C\|$. This shows that if we reach a y_{r+1} which is negligible to working accuracy, eigenvalues and eigenvectors of T_r give good approximations to eigenvalues and eigenvectors of C . (The latter must be interpreted in terms of exact eigenvalues of some C+E where E is small.) Also,

provided the $x_1,\ldots,x_r$ have not yet departed too far from orthogonality, we shall indeed get good approximations to r eigenvalues and r eigenvectors of C .

However, even if y_{r+1} is not itself negligible it may happen that for some eigenvalues λ of T_r, the corresponding z may have a moderately small z_r . Then $z_r y_{r+1}$ may be negligible even if y_{r+1} is not sufficiently small in itself. This situation may be induced to some extent by starting with an initial vector x_1 which consists mainly of a linear combination of a few dominant eigenvectors; such an initial vector is obtained if an arbitrary vector x_0 is premultiplied several times with C . It cannot be too strongly emphasized that the size of the vectors ε_i is not in any way affected by cancellation or by the normalization of y_{i+1} to give x_{i+1} ,the vector ε_i consists entirely of the rounding errors made in actually multiplying the computed x_i by C and subtracting multiples of x_i and x_{i-1} . Even the accuracy of the α_i and β_i is quite irrelevant in so far as it effects the size of the ε_i though if they were chosen at random it is unlikely that a small y_{r+1} would emerge. These considerations show why the Lanczos algorithm often gives remarkably accurate approximations to dominant eigenvalues and eigenvectors after quite a few steps.

In order to be certain of obtaining the full set of eigenvalues and eigenvectors it is necessary to ensure that the computed x_i are orthogonal to working accuracy. The conventional way of doing this is as follows. After computing y_{r+1} via Equations (7), it is reorthogonalized with respect to $x_r,x_{r-1},\ldots x_1$. (N.B. Since the lack of orthogonality

is caused by cancellation and not by the accumulation of rounding errors, it is just as necessary to reorthogonalize with respect to x_r and x_{r-1} (although y_{r+1} has just be orthogonalized with respect to these vectors) as to the earlier vectors.) We may write

$$y_{r+1} = y_{r+1} - \eta_1 x_1 - \eta_2 x_2 - \cdots - \eta_r x_r \tag{18}$$

where the η_i are chosen so as to give orthogonality. If cancellation takes place when deriving the $\bar{y}_{r+1}$, then $\bar{y}_{r+1}$ must be reorthogonalized yet again with respect to all earlier vectors. Moreover, if y_{r+1} or y_{r+1} vanishes, then a technique is needed for restarting. Although a perfectly satisfactory procedure may be constructed on these lines, it is not aesthetically pleasing. We now describe an alternative procedure for ensuring the orthogonality of the computed x_i to working accuracy.

Suppose $x_1, x_2, \ldots, x_r$ have already been determined and are orthogonal to working accuracy, and that the matrix $X_r = [x_1, x_2, \ldots, x_r]$ has been reduced to upper-triangular form by premultiplication with r elementary Hermitian matrices $P_1, P_2, \ldots, P_r$. Here

$$P_i = I - 2w_i w_i^T \quad , \quad \|w_i\|_2 = 1 \tag{19}$$

and the first i-1 components of w_i are zero. From the orthonormality of the x_i, this reduced form must consist of the first r columns of I. The vector y_{r+1} is then determined from Equation (7) and the vector z_{r+1} is determined from the relation

$$z_{r+1} = P_r \cdots P_2 P_1 y_{r+1} \quad . \tag{20}$$

Now with exact computation y_{r+1} would be orthogonal to $x_1,\ldots,x_r$ and hence z_{r+1} would be orthogonal to the r vectors $P_r\ldots P_1 x_i$, $(i = 1,\ldots,r)$. But these vectors are $e_1,e_2,\ldots,e_r$, and hence with exact computations z_{r+1} would have zero components in elements 1 to r . We now determine P_{r+1} so that $P_{r+1}z_{r+1}$ has zero components in elements $r+2,\ldots,n$. With exact computation $P_{r+1}z_{r+1}$ would be a multiple of e_{r+1} . Now define x_{r+1} by the relation

$$x_{r+1} = P_1P_2\ldots P_{r+1}e_{r+1} \tag{21}$$

so that x_{r+1} is automatically a unit vector. If all the computation had been exact x_{r+1} would merely be a multiple of y_{r+1} , the multiple being chosen so as to make x_{r+1} a unit vector. Notice that this technique gives us a method of continuing when $y_{r+1} = 0$. We merely define x_{r+1} by the relation

$$x_{r+1} = P_1P_2\ldots P_re_{r+1} \tag{22}$$

which corresponds to having taken I in place of P_{r+1} in Equation (21). From the derivation of the P_i , the vector x_{r+1} is automatically orthogonal to $x_1,x_2,\ldots,x_r$ since

$$x_i = P_1P_2\ldots P_ie_i = P_1P_2\ldots P_iP_{i+1}\ldots P_re_i \qquad (i \leq r) \tag{23}$$

giving

$$x_{r+1}^Tx_i = e_{r+1}^TP_r^T\ldots P_2^TP_1^TP_1P_2\ldots P_re_i = e_{r+1}^Te_i = 0 . \tag{24}$$

Notice that with this technique, when determining y_{r+1} we need only x_r and x_{r-1}. To determine an x_{r+1} accurately orthogonal to $x_1,\ldots,x_r$ we need only $P_1,\ldots,P_r$ and these may be stored via the corresponding w_i. Since the first i-1 components of w_i are zero, approximately $\frac{1}{2}n^2$ registers are needed to store full information on the P_i as against n^2 if we store the x_i. The full set of Equations (21) shows that

$$[x_1,x_2,\ldots,x_n] = P_1P_2\ldots P_n \ , \tag{25}$$

so that in retaining information on the P_i via the w_i we effectively have full information on the x_i.

The quantity γ_i may be derived via the relation

$$\gamma_i = \|y_i\|_2 \ . \tag{26}$$

We can take β_i to be equal to the γ_i derived in this way or we can determine it by making y_{i+1} orthogonal to x_{i-1}, that is, via the relation

$$\beta_i = x_{i-1}^T x_i \ . \tag{27}$$

Even after the reorthogonalization γ_i and β_i determined in this way will agree to within a small multiple of machepsx $\|C\|_2$. In practice, it is instructive to compare the β_i and γ_i obtained from Equations (26) and (27). We have finally

$$C[x_1,x_2,\ldots,x_n] = [x_1,x_2, \tag{28}$$

to working accuracy, where the x_i are orthogonal to working accuracy and T is a symmetric tridiagonalmatrix. If

$$Tz_i = \lambda_i z_i \tag{29}$$

then

$$C[x_1, x_2, \ldots, x_n]z_{ii} = \lambda_i[x_1, x_{2..} \tag{30}$$

giving

$$C(P_1P_2\ldots P_n)z_i = \lambda_i(P_{i-1}\ldots P_{ni}, \quad \text{or} \quad CQz_i = \lambda_i Qz_i \tag{31}$$

so that the P_j give sufficient information to enable us to determine eigenvectors of C . When $C = L^{-1}AL^{-T}$, we have

$$L^{-1}AL^{-T}Qz_i = \lambda_i Qz_i \quad \text{or} \quad A(L^{-T}Qz_i) = \lambda_i B(L^{-T}Qz) \tag{32}$$

and hence eigenvectors of $A-\lambda B$ may be determined using the matrix L . If the z_i are a set of orthonormal eigenvectors of T , then $p_i = L^{-T}Qz_i$ gives a set of eigenvectors for the problem $Ap = \lambda Bp$ such that

$$p_i^T Bp_i = z_i^T Q^T L^{-1} LL^T L^{-T} Qz_i = z_i^T z_i = 1 \tag{33}$$

$$p_i^T Bp_j = z_i^T Q^T L^{-1} LL^T L^{-T} Qz_j = z_i^T z_j = 0 \qquad (i \neq j) \tag{34}$$

2. Applicability

reducb may be used to reduce the eigenproblem $Ax = \lambda Bx$ to the standard symmetric eigenproblem $Ty = \lambda y$ where T is tridiagonal.

While reducb may be used whenever A and B are symmetric and positive definite, it is best used in problems in which the band width of A and B are small in comparison to their order.

The derived tridiagonal system may be solved by a variety of methods [4]. The eigenvalues of the derived standard problem are those of the original problem, but the vectors are related as indicated by Equations (31) and (32).

3. Formal Parameter List

Input to procedure reducb.

n order of matrices A and B .

ma number of lower diagonals of A .

mb number of lower diagonals of B .

a elements of the lower triangle of the symmetric matrix A stored as an $n \times (ma+1)$ array.

b elements of the symmetric matrix B stored as an $n \times (mb+1)$ array.

Output of procedure reducb.

alpha diagonal elements of the symmetric tridiagonalmatrix T similar to $L^{-1}AL^{-T}$.

beta codiagonal elements of T .

b the lower triangle of L such that $LL^T = B$, stored as an $n \times (mb+1)$ array (overwriting the original in b).

U information on the matrices $P_i = I - 2w_i w_i^T$ (This may be stored as an $n \times n$ array, but for economy it can be stored as a linear array of order $\frac{1}{2}n(n+1)$.)

fail exit used if B , possibly as the result of rounding errors, is not positive definite.

4. ALGOL W [3, 5] Procedures

```
procedure reducb (integer value n, ma, mb;
                  long real array a, b(*,*);
                  long real array alpha, beta(*);
                  long real array u(*,*);
                  procedure fail);

comment     Reduction of the symmetric eigenvalue problem

                        Ax = lambda*Bx

        with symmetric band matrix A and symmetric positive
        definite band matrix B, to symmetric tridiagonal form
        by the Lanczos method.
           The lower triangles of A and B are stored in the
        arrays a(1::n,0::ma) and b(1::n,0::mb), where ma
        and mb are the number of subdiagonals in A and B,
        respectively.  L, the Cholesky factor of B, is
        computed and overwritten on B in b.  u is used to
        store details of the transformation.  The diagonal
        of the result is stored in the array alpha(1::n)
        and the subdiagonal in the last n-1 stores of the
        array sub(1::n).
           The actual parameter corresponding to fail will
        be executed if B, perhaps on account of rounding
        errors, is not positive definite. ;

begin integer p,q,r,s;
   long real y0,y1,z;
   long real array v,x0,x1,y(1::n);

   comment Compute the Cholesky factor of B;
   for i:=1 step 1 until n do
   begin p:=(if i>mb then 0 else mb-i+1);
      r:=i-mb+p;
      for j:=p step 1 until mb do
      begin s:=j-1;
         q:=mb-j+p;
         z:=b(i,j);
         for k:=p step 1 until s do
         begin z:=z-b(i,k)*b(r,q);
            q:=q+1;
         end;
         if j=mb then
         begin if z<0 then fail;
            b(i,j):=longsqrt(z);
         end
         else b(i,j):=z/b(r,mb);
         r:=r+1;
      end j;
   end forml;
```

```
comment Compute tridiagonal form;
beta(1):=0l; y(1):=1l;
for i:=2 step 1 until n do y(i):=0l;
for k:=1 step 1 until n do
begin
    z:=0l;
    for j:=1 step 1 until n do
    begin x0(j):=x1(j);
        x1(j):=y(j);
        z:=z+y(j)*y(j);
    end;
    y0:=y1; y1:=z;

    comment Multiply x by inv(L)*A*inv(L');
    s:=mb-1;
    for i:=n step - 1 until 1 do
    begin p:=(if i <=n-mb then 0 else mb+i-n);
        q:=i;
        z:=x1(i);
        for j:=s step -1 until p do
        begin q:=q+1;
            z:=z-b(q,j)*v(q);
        end;
        v(i):=z/b(i,mb);
    end solve;
    for i:=1 step 1 until n do
    begin p:=(if i>ma then 0 else ma-i+1);
        q:=i-ma+p;
        z:=0l;
        for j:=p step 1 until ma do
        begin z:=z+a(i,j)*v(q);
            q:=q+1;
        end;
        p:=(if i<=n-ma then 0 else ma+i-n);
        for j:=ma-1 step -1 until p do
        begin z:=z+a(q,j)*v(q);
            q:=q+1;
        end;
        y(i):=z;
    end av;
    for i:=1 step 1 until n do
    begin p:=(if i>mb then 0 else mb-i+1);
        q:=i;
        z:=y(i);
        for j:=s step -1 until p do
        begin q:=q-1;
            z:=z-b(i,j)*v(q);
        end;
        v(i):=z/b(i,mb);
    end solve;
```

```
        comment Compute alpha(k) andbeta(k);
        z:=0l;
        for j:=1 step 1 until n do
        z:=z+x1(j)*v(j);
        alpha(k):=z:=z/y1;
        for j:=1 step 1 until n do
        y(j):=v(j)-z*x1(j);
        if k=1 then
        begin z:=0l;
           for j:=1 step 1 until n do
           z:=z+x0(j)*y(j);
           beta(k):=z:=z/y0;
           if k=n then go to 11;
           for j:=1 step 1 until n do
           y(j):=y(j)-z*x0(j);
        end;

        comment Normal ize and reorthogonal ize y with
                respect to previous col umns of X;
        for i:=2 step 1 until k do
        begin;
           for j:=i step 1 until n do z:=z+u(j,i)*y(j);
           z:=z/u(i,1);
           for j:=i step 1 until n do y(j):=y(j)-z*u(j,i);
        end;

        z:=0l;
        for i:=k+1 step 1 until n do z:=z+y(i)*y(i);
        if z=0l then
        begin,1):=11;
           for i:=k+1 step 1 until n do u(i,k+1):=0l;
        end else
        begin
           z:=if y(k+1)>=0l then. longsqrt(z) else -longsqrt(z);
           u(k+1,k+1):=y(k+1)+z; u(k+1,1):=u(k+1,k+1)*z;
           for j:=k+2 step 1 until n do u(j,k+1):=y(j);
        end;
        for j:=1 step 1 until n do
        y(j):=if j=k+1 then 11 else 0 1 ;

        for i:=k+1 step -1 until 2 do
        begin z:=0l
           for j:=i step 1 until n do z:=z+u(j,i)*y(j);
           z:=z/u(i,1);
           for j:=i step 1 until n do y(j):=y(j)-z*u(j,i);
        end;

     11:
     end k;
end reducb;
```

5. Organizational and Notational Details

The lower triangle of A is stored in such a manner that array element a(i, ma-i+j) contains the value of matrix element A(i,j), i=1,...n, and j=max(i-ma,1),...i. Thus, columns of a correspond to diagonals of A. B is stored similarly. L, the Cholesky factor of B, is lower triangular with the same number of diagonals as the lower triangle of B. L is stored as B is, overwriting B in b.

The initial vector x_1 is chosen to be e_1. The details of the elementary Hermitian matrices P_1 are contained in the vectors Ui and the scalars K_i, where

$$P_i = I - \frac{U_i U_i^t}{K_i}$$

If it should happen that $z_i = 0$, cf. (20), then $U_i = 0$, and $K_i = 1$, so that the corresponding P_i is the identity matrix. Also, since x_1 is e_1, P_1 is chosen to be the identity matrix, and information on P_1 is not stored. Otherwise, Ui is stored in the ith column of u, and K_i is stored in u(i, 1), i=2,...,n.

The diagonal of the reduced symmetric tridiagonal matrix is stored in the array alpha, and the off-diagonal in the last n-1 elements of the array beta. Beta (1) is set to zero.

6. Discussion of Numerical Properties

The behaviour of the reorthogonalization process is far from obvious. A detailed error analysis tends to obscure the essential simplicity of the underlying mechanism and we content ourselves with an exposition of the latter. For convenience it will be assumed that $\|C\|_2 = 1$.

We proceed by induction. Let us assume that on a computer with t-digit mantissa $x_1, x_2, \ldots, x_r$ have been determined and satisfy

$$\gamma_i x_{i+1} = Cx_i - \alpha_i x_i - \beta_i x_{i-1} + O(2^{-t}) \qquad i = 1, \ldots, r-1 \qquad \text{(i)}$$

and

$$x_i^T x_j = \delta_{ij} + O(2^{-t}) \ , \qquad i,j < r \ . \qquad \text{(ii)}$$

In other words we assume that the x_i produced by the reorthogonalization technique are orthogonal to working accuracy.

In the next step y_{r+1} is first determined and the computed vector satisfies the relation

$$y_{r+1} = Cx_r - \alpha_r x_r - \beta_r x_{r-1} + O(2^{-t}) \ . \qquad \text{(iii)}$$

If a great deal of cancellation takes place y_{r+1} will not be accurately orthogonal to x_r and x_{r-1} but the determination of α_r and β_r ensures that $y_{r+1}^T x_r$, $y_{r+1}^T x_{r-1} = O(2^{-t})$ (N.B. This will only imply accurate orthogonality if $\|y_{r+1}\|_2$ were of order unity; if $\|y_{r+1}\|_2$ is small, $y_{r+1}^T x_r$ can be of order 2^{-t} without y_{r+1} being orthogonal to x_r to working accuracy.) We now show that $y_{r+1}^T x_i = O(2^{-t})$ for all earlier x_i . In fact we have

$$y_{r+1}^T x_i = x_i^T(Cx_r - a_r x_r - \beta_r x_{r-1}) + O(2^{-t}) \qquad \text{(iv)}$$

$$= x_r^T C x_i + O(2^{-t})$$

$$= x_r^T(x_{i+1} + \alpha_i x_i + \beta_i x_{i-1}) + O(2^{-t})$$

$$= O(2^{-t}) \quad .$$

The essential point is that the inner-products of y_{r+1} with respect to $x_1,\ldots,x_r$ are all negligible. If in particular $\|y_{r+1}\|_2$ is of the order of unity y_{r+1} will already be accurately orthogonal to x_1, [illegible] and reorthogonalization will be unnecessary. In any case we may write

$$y_{r+1} = \alpha z + \eta_1 x_1 + \eta_2 x_2 + \ldots + \eta_r x_r \,, \qquad |\eta_i| = O(2^{-t}) \qquad \text{(v)}$$

and

$$\alpha = \|y_{r+1}\| + O(2^{-t}) = \gamma_{r+1} + O(2^{-t}) \quad . \qquad \text{(vi)}$$

The vector y_{r+1} is now multiplied by $P_1, P_2, \ldots, P_r$ successively and the resulting vector is used to determine P_{r+1}. From the derivation of the previous P_i it is evident that

$$P_{r+1}P_r \cdots P_1 y_{r+1} = \alpha e_{r+1} + \eta_1 e_1 + \ldots + \eta_r e_r + O(2^{-t})$$

$$= \gamma_{r+1} e_{r+1} + O(2^{-t}) \qquad \text{(vii)}$$

where e_i denotes the i-th column of I. Notice on the right-hand side of (vii), $\gamma_{r+1} e_{r+1}$ may not necessarily be much larger than the term

denoted by $O(2^{-t})$. If a great deal of cancellation took place when y_{r+1} was computed then y_{r+1} will be correspondingly small. However, independent of the size of y_{r+1} we have

$$y_{r+1} = \gamma_{r+1}P_1P_2\cdots P_{r+1}e_{r+1} + O(2^{-t}) \qquad \text{(viii)}$$

and substituting this in (iii) we have

$$\gamma_{r+1}P_1P_2\cdots [illegible] = Cx_r - \alpha_r x_r - \beta_r x_{r-1} + O(2^{-t}) . \qquad \text{(ix)}$$

Hence taking $x_{r+1} = P_1P_2\cdots P_{r+1}e_{r+1}$, Equation (ix) becomes

$$\gamma_{r+1}x_{r+1} = Cx_r - \alpha_r x_r - \beta_r x_{r-1} + O(2^{-t}) \qquad \text{(x)}$$

and since earlier x_i have been determined via the relation

$$x_i = P_1P_2\cdots P_i e_i \qquad \text{(xi)}$$

it is clear that x_{r+1} is orthogonal to all earlier x_i to working accuracy. If there had been exact computation throughout, x_{r+1} would have been $y_{r+1}/\|y_{r+1}\|_2 = z_{r+1}$ (say). If cancellation has taken place and $\|y_{r+1}\| = 2^{-k}$ (say), then (vii) shows that we can expect the computed $\|x_{r+1} - z_{r+1}\|$ to be of the order of 2^{k-t} . Hence as k becomes larger and approaches t , x_{r+1} increasingly diverges from z_{r+1} . However, since $\gamma_{r+1} = 2^{-k}$ the replacement of y_{r+1} , i.e., $\gamma_{r+1}z_{r+1}$ by $\gamma_{r+1}x_{r+1}$ on the left of Equation (iii) is merely a change of order 2^{-t} .

Having established these relations, we are now in a position to compare the computed β_i and γ_i. We have

$$\beta_i = x_{i-1}^T C x_i + O(2^{-t}) = x_i^T (C x_{i-1}) + O(2^{-t}) \qquad \text{(xii)}$$

$$= x_i^T (\alpha_{i-1} x_{i-1} + \beta_{i-1} x_{i-2} + \gamma_i x_i + O(2^{-t})) + O(2^{-t})$$

$$= \gamma_i + O(2^{-t}) \ .$$

(Without the normalization of C we have $\beta_i = \gamma_i + O(2^{-t} \|C\|_2)$.) Since β_i and γ_i are floating point numbers, the number of figures agreeing in the mantissa depends on the degree of cancellation. But it is clear that if we replace β_i by γ_i we still have, as before,

$$\gamma_{i+1} x_{i+1} = C x_i - \alpha_i x_i - \gamma_i x_{i-1} + O(2^{-t}) \qquad \text{(xiii)}$$

and hence we can take the derived tridiagonal matrix to be symmetric. In the case when γ_{r+1} is zero (or is considered to be negligible) we can clearly take x_{r+1} to be $P_1 P_2 \ldots P_r e_{r+1}$ and we have

$$0 \cdot x_{r+1} = \theta = \gamma_{r+1} = C x_r - \alpha_r x_r - \beta_r x_{r-1} + O(2^{-t}) \ .$$

In this case β_{r+1} will also turn out to be negligible to working accuracy.

An error analysis of the symmetric Lanczos process with Schmidt reorthogonalization has been given by Paige [2].

7. Test Results

To test reducb, the matrices

$$A = \begin{matrix} 10 & 2 & 3 & 1 & 1 \\ 2 & 12 & 12 & & 1 \\ 3 & 1 & 11 & 1 & -1 \\ 1 & 2 & 1 & 9 & 1 \\ 1 & 1 & -1 & 1 & 15 \end{matrix} \qquad B = \begin{matrix} 12 & 1 & -1 & 2 & 1 \\ 1 & 14 & 1 & -1 & 1 \\ -1 & 1 & 16 & -1 & 1 \\ 2 & -1 & -1 & 12 & -1 \\ 1 & 1 & 1 & -1 & 11 \end{matrix}$$

were used. A and B are of full width, so $ma = mb = 4$. On an IBM System 360 model 67 computer using floating point arithmetic with a 14 hexadecimal digit fraction, the following results were obtained (Although not necessary, the elements of u were initially zeroed.):

α

$$\begin{matrix} 0.8333333333333333_{10}+00 \\ 0.7268776335953 68_{10}+00 \\ 0.1162372359171 15_{10}+01 \\ 0.\ 1056929923237 69_{10}+01 \\ 0.8624334873006 40_{10}+00 \end{matrix}$$

8

$$\begin{matrix} 0.0000000000000 00_{10}+00 \\ -0.2885434037570 58_{10}+00 \\ -0.2178371544673 99_{10}+00 \\ 0.3029237276557 04_{10}+00 \\ 0.2196697066586 49_{10}+00 \end{matrix}$$

U

$$\begin{matrix} 0.0000000000000 00_{10}+00 & 0.0000000000000 00_{10}+00 & 0.0000000000000 00_{10}+00 \\ 0.\ 1093068146175 72_{10}+00 & 0.3788227808860 40_{10}+00 & 0.0000000000000 00_{10}+00 \\ 0.1773574632194 24_{10}-01 & 0.2715191299547 77_{10}+00 & 0.2223484163680 22_{10}+00 \\ 671791653895_{10}+00 & -0.3343417038780 02_{10}-01 & -0.1085033136702 29_{10}+00 \\ 0.\ 9650956004699 36_{10}-01 & -0.1632324220220 66_{10}-01 & 0.1888377751010 04_{10}+00 \end{matrix}$$

$$\begin{matrix} 0.0000000000000 00_{10}+00 \\ 0.0000000000000 00_{10}+00 \\ 0.0000000000000 00_{10}+00 \\ -0.5865231919231 92_{10}+00 \\ -0.1064618647404 83_{10}+00 \end{matrix} \qquad \begin{matrix} 0.0000000000000 00_{10}+00 \\ 0.0000000000000 00_{10}+00 \\ 0.0000000000000 00_{10}+00 \\ 0.0000000000000 00_{10}+00 \\ -0.4393394133172 97_{10}+00 \end{matrix}$$

The resulting tridiagonal system was solved using the procedure tql2. After the vectors of the tridiagonal system were transformed according to equations (31) and (32), the final results were essentially the same as those reported in [1] for the above matrices.

References

1. Martin, R. S., and Wilkinson, J. H., "Reduction of the Symmetric Eigenproblem $Ax = \lambda Bx$ and Related Problems to Standard Form," Num. Math. 11 (1968), pp. 99-110.

2. Paige, C. C., "Practical Use of the Symmetric Lanczos Process with Reorthogonalization," BIT 10 (1970), pp. 183-195.

3. Sites, R. L., "Algol W Reference Manual," Technical Report 230 (1971), Computer Science Department, Stanford University, Stanford, California,

4. Wilkinson, J. H., and Reinsch, C., Handbook for Automatic Computation, Volume II: Linear Algebra, Springer-Verlag, New York, 1971.

5. Wirth, N., and Hoare, C. A. R., "A Contribution to the Development of Algol," Comm. A.C.M. 9, No. 6 (1966), pp. 413-431.

Keywords

matrix

eigenvalue

eigenvector

symmetric

positive definite

band

generalized

tridiagonal

Lanczos

elementary Hermitian matrix

Householder

orthogonalization

Cholesky

error analysis