Eigenvectors of a $2n \times 2n$ Matrix

S. CHARMONMAN
University of Alberta, Edmonton, Alberta, Canada

It has been known that the eigenvalues of a certain $2n \times 2n$ matrix can be obtained by use of two smaller matrices of order n which can be easily constructed. An algorithm is given to obtain the eigenvectors of the $2n \times 2n$ matrix by use of the eigenvectors of the smaller matrices.

1. Introduction

Let the matrix

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix},$$

where A and B are real square matrices of order n, be denoted by S, and the set of $2n$ eigenvalues of S by $\lambda(S)$. Friedman proved that $\lambda(S) = [\lambda(P = A + B), \lambda(Q = A - B)]$, [1, 2]. Therefore, the eigenvalues of the $2n \times 2n$ matrix S can be obtained by solving for the eigenvalues of two $n \times n$ matrices, P and Q. The use of the smaller matrices instead of S requires less computer storage and a smaller number of arithmetic operations which, in turn, results in less computer time and lower upper bound of roundoff error. For example, suppose the number of arithmetic operations for an $n \times n$ matrix is of order n^3. In this case the number of arithmetic operations in using two $n \times n$ matrices is of order $2n^3$ as opposed to $(2n)^3$ in using the original $2n \times 2n$ matrix. Therefore, the saving in the number of airthmetic operations in using the smaller matrices instead of the original one is about 75%.

Examples of matrices of the form S can be found in the theory of directional couplers [3] and in overlapping polymer chains [4].

2. Eigenvectors

The approach of using smaller matrices can also be extended to the problem of computing eigenvectors. Let $x_i' = (x_{i1}', x_{i2}')$, where primes denote transposes, be the eigenvector of S belonging to an eigenvalue λ_i. Thus $Sx_i = \lambda_i x_i$ can be written as

$$Ax_{i1} + Bx_{i2} = \lambda_i x_{i1} \tag{1}$$

$$Bx_{i1} + Ax_{i2} = \lambda_i x_{i2}. \tag{2}$$

Adding the above two equations and letting $P = A + B$ give

$$Py_i = \mu_i y \qquad (i \in I1)_i \tag{3}$$

where

$$y_i = x_{i1} + x_{i2},\ \mu_i = \lambda_i, \qquad (_i \in I1) \tag{4}$$

and $I1$ is the set of integers $(1, 2, \cdots, n)$. Similarly, subtracting (2) from (1) with $Q = A - B$ gives

$$Qz_i = \eta_i z_i \qquad (i \in I2) \tag{5}$$

where

$$z_i = x_{i1} - x_{i2}, \qquad \eta_i = \lambda_i \qquad (i \in I2) \tag{6}$$

and $I2$ is the set of integers $(n + 1, n + 2, \cdots, 2n)$. Note that the y_i and z_i above may be replaced by any nonzero multiples.

After the complete eigenvalue problems of P and Q are solved, it is possible to form the eigenvectors of S by inspection according to the following algorithm.

ALGORITHM 1. To obtain x_k belonging to λ_k of S, follow the following steps:

1. Is $k \in I1$? If not, go to step 3, otherwise proceed to step 2.

2. Is $\lambda_k \in \lambda(Q)$? If not, $x_k' = (y_k', y_k')$. If yes, search for the eigenvector z_j of Q belonging to $\eta_j = \lambda_k$. Then form $x_k' = (y_k' + cz_j', y_k' - cz_j')$, where c is an arbitrary constant. Stop or exit.

3. Is $\lambda_k \in \lambda(P)$? If not, $x_k' = (z_k', -z_k')$. If yes, search for the eigenvector y_j of P belonging to $\mu_j = \lambda_k$. Then form $x_k' = (z_k' + cy_j', cy_j' - z_k')$.

PROOF. Substracting (2) from (1) with λ_i replaced by μ_i for the case when $i \in I1$ gives

$$Qr_i = \mu_i r_i \tag{7}$$

where

$$r_i = x_{i1} - x_{i2} \qquad (i \in I1). \tag{8}$$

Now we observe that

(i) $\mu_i \notin \lambda(Q) \Rightarrow |Q - \mu_i I| \neq 0 \Rightarrow r_i = 0 \Rightarrow x_{i1} - x_{i2} =$

$0 \Rightarrow x_{i1} = x_{i2}$. Therefore, from the first part of (4), $x_{i1} = x_{i2} = y_i/2$. The factor $\frac{1}{2}$ need not be used because eigenvectors are obtained only up to a constant multiplier.

(ii) $\mu_i \in \lambda(Q) \Rightarrow \mu_i = \eta_j$, $(j \in I2)$, $\Rightarrow r_i = cz_j = x_{i1} - x_{i2}$, where c is an arbitrary constant. Therefore, $cz_j = x_{i1} - x_{i2}$ and $y_i = x_{i1} + x_{i2}$ give $x_{i1} = (y_i + cz_j)/2$ and $x_{i2} = (y_i - cz_j)/2$. Again the constant factor need not be used.

A proof for the case $i \in I2$ can be similarly constructed.

3. A Numerical Example

As an illustration of the use of Algorithm 1 we consider a 4×4 matrix

$$S = \begin{bmatrix} 0.25 & 3.25 & -1.25 & -1.25 \\ -1.25 & 0.75 & -1.75 & 3.25 \\ -1.25 & -1.25 & 0.25 & 3.25 \\ -1.75 & 3.25 & -1.25 & 0.75 \end{bmatrix}. \quad (9)$$

From (9) we have

$$P = A + B = \begin{bmatrix} -1 & 2 \\ -3 & 4 \end{bmatrix},$$
$$Q = A - B = \begin{bmatrix} 1.5 & 4.5 \\ 0.5 & -2.5 \end{bmatrix}. \quad (10)$$

Solving for the eigenvalues and eigenvectors of P and Q gives $\lambda(P) = (\lambda_1, \lambda_2) = (\mu_1, \mu_2) = (1, 2)$; $\lambda(Q) = (\lambda_3, \lambda_4) = (\eta_3, \eta_4) = (2, -3)$, $y_1' = (1, 1)$, $y_2' = (\frac{2}{3}, 1)$, $z_3' = (9, 1)$, and $z_4' = (-1, 1)$.

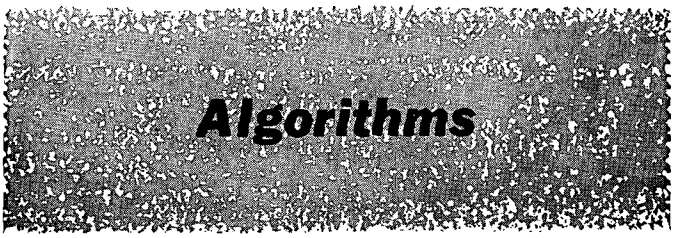
By use of Algorithm 1 the eigenvector of S belonging to $\lambda_1 = 1$ can be formed by noting that $\lambda_1 \notin \lambda(Q)$. Therefore, $x_1' = (y_1', y_1') = (1, 1, 1, 1)$. Similarly, to form x_2 belonging to $\lambda_2 = 2$ we note that $\lambda_2 \in \lambda(Q)$. In particular $\lambda_2 = \lambda_3$. Therefore, $x_2' = (y_2' + cz_3', y_2 - cz_3') = (\frac{2}{3} + 9c, 1 + c, \frac{2}{3} - 9c, 1 - c)$, where c is an arbitrary constant.

Acknowledgment. The preparation of this paper was supported by the National Research Council of Canada Grant No. A-3135.

RECEIVED DECEMBER, 1966; REVISED MAY, 1967

REFERENCES

1. FRIEDMAN, B. Eigenvalues of compound matrices. Research Rept. No. TW-16, Math. Res. Group. New York U., New York, 1951.
2. MARCUS, M. *Basic Theorems in Matrix Theory.* Appl. Math. Ser. 57, Nat. Bur. Standards, 1960, p. 17, Govt. Printing Office, Washington, D.C., p. 17
3. MONTGOMERY, C. G. ET AL. *Principles of Microwave Circuits.* Boston Technical Publishers, Inc., Lexington, Mass. 1964, pp. 437–452.
4. MONTROLL, E. W. Markoff chains and excluded volume effect in polymer chains. *J. Chem. Phys. 18* (1950), 734–743.



J. G. HERRIOT, Editor

ALGORITHM 318
CHEBYSCHEV CURVE-FIT (REVISED) [E2]

J. BOOTHROYD (Recd. 15 May 1967)
University of Tasmania, Hobart, Tas., Australia

```
procedure chebfit(x, y, n, a, m); value n, m;
  array x, y, a; integer n, m;
comment evaluates, in a[0] through a[m] of a[0:m+1], the co-
  efficients of an mth order polynomial P(x) = a0 + a1x + ··· amx^m
  such that the maximum error abs(P(xi)−yi)) is a minimum over
  the n(n>m+1) sample points x, y[1:n]. The x[i] must form a
  strictly monotonic sequence.
  This procedure is an extensive revision of Algorithm 91 (Albert
  Newhouse, Chebyshev Curve-Fit, Comm. ACM 5 (May 1961),
  281). The polynomial P(x) is a best-fit polynomial in the Cheby-
  shev sense as described by Stiefel (Numerical Methods of Tcheby-
  cheff Approximation), in Langer (Ed.), On Numerical Approxi-
  mation, U. of Wisconsin Press, 1959, pp. 217–232. Stiefel (p. 221)
  shows that the procedure must terminate after a finite number
  of steps. This is not always so with imperfect arithmetic, where
  roundoff errors may cause cycling of the chosen reference sets.
  This condition is detected by checking that the reference devia-
  tion is always raised monotonically. At exit the absolute value
  of a[m+1] yields the final reference deviation. Negative a[m+1]
  indicates that the procedure has been terminated following the
  detection of cycling;
begin
  integer i, j, k,mplus1, ri, i1, imax, rj, j1;
  real d, h, ai1, rhi1, denom, ai, rhi, xj, hmax, himax, xi, hi, abshi,
    nexthi, prevh;
  integer array r[0:m+1]; array rx, rh[0:m+1];
  mplus1 := m + 1; prevh := 0;
  comment index vector for initial reference set;
  r[0] := 1; r[mplus1] := n;
  d := (n−1)/mplus1; h := d;
  for i := 1 step 1 until m do
  begin r[i] := h + 1; h := h + d end;
start: h := −1.0;
  comment select m + 2 reference pairs and set alternating
    deviation vector;
  for i := 0 step 1 until mplus1 do
  begin
    ri := r[i];
    rx[i] := x[ri]; a[i] := y[ri];
    rh[i] := h := −h
  end i;
  comment compute m + 1 leading divided differences;
  for j := 0 step 1 until m do
  begin
    i1 := mplus1; ai1 := a[i1];
    rhi1 := rh[i1];
    for i := m step −1 until j do
    begin
      denom := rx[i1] − rx[i−j];
      ai := a[i]; rhi := rh[i];
      a[i1] := (ai1−ai)/denom;
```

(Continued on page 803)