ON AN AUTOMATIC COMPUTATION OF CHARACTERISTIC ROOTS OF A MATRIX

By Syamal Kumar Sen

(*Central Instruments and Services Laboratory, Indian Institute of Science, Bangalore 12, India*)

[Received: May 12, 1967]

ABSTRACT

A method consisting in finding a two dimensional elementary transformation for obtaining eigen values has been discussed with details of programming aspects and it has been shown with the help of numerical examples that this method is as efficient as L-R transformation so far as the number of iteration steps are concerned and moreover, it demands less machine time, less number of arithmetic operations (particularly divisions and multiplications) per iteration and less memory storage. Like L-R transformation this method is always stable for positive definite matrices and takes substantially less machine time than Q-R transformation and moreover, can be used both for symmetric and unsymmetric matrices.

INTRODUCTION

Jocobi's iteration method for finding eigen values and eigen vectors of symmetric matrices as revived by Von Neumann[19] for modern large computers, is well known and popular. A Jacobi-like method suggested by Eberlein[16, 10] for unsymmetric matrices has a wide range of application, but they require the calculation of several square-root functions in each iteration leading to rounding error and more computational time per iteration. The present method, however, does not involve calcution of aforesaid function thus obviating such rounding error and saving on operational time. It requires simple operations like addition, subtraction, multiplication and division while however utilizing, like the aforesaid methods, the principle of similar transformation. Like L-R transformation the present method also may not work in the case of singular matrices. In the following the method is discussed *vis-a-vis* L-R transformation stressing the programming aspects. Numerical examples are given to provide illustration. The relative computing time as well as memory requirements and number of arithmetic operations are also indicated.

METHODS

Method 1: (Based on two dimensional elementary transformation). This method consists in finding a sequence of two dimensional elementary transformations E_i such that if $E = \prod_i E_i$ then $E^{-1} A E$ are approximately diagonal with the approximations to the eigen values appearing on the diagonal, where A is a matrix of order n. The E_i which range over all two dimensional subspaces are determined at each step of the iteration to reduce to zero all the i-th row elements of A excluding, however, the first i elements. The matrix E_k will be an upper triangular[13] matrix of the form

$$e_{jj} = 1, \quad j = 1, 2, 3, \cdots, n$$
$$e_{ki} = -a_{ki}/a_{kk}, \quad i = k+1, k+2, k+3, \cdots, n-1, n$$
$$e_{pq} = 0, \quad p = 1, 2, 3, \cdots, n-2, n-1 ; q = p+1, p+2, \cdots, n$$
$$p \neq k$$

and consequently E_k^{-1} is also an upper triangular matrix of the form

$$e_{jj} = 1, \quad j = 1, 2, 3, \cdots, n-1, n$$
$$e_{ki} = a_{ki}/a_{kk}, \quad i = k+1, k+2, k+3, \cdots, n$$
$$e_{pq} = 0 \quad p = 1, 2, 3, \cdots, n-2, n-1 ; q = p+1, p+2, \cdots, n$$
$$p \neq k$$

Now

$$E_k^{-1} A_k E_k = A_{k+1}, \quad A_1 = A, \quad k = 1, 2, 3, \cdots, n-2, n-1$$

All the matrices A_k, $k = 1, 2, 3, \cdots, n-2, n-1$ will assume the same memory locations as the original matrix A. According to the principle of similar transformation, the roots of A_{k+1} and A_k will be the same. The process will continue till almost all the elements above the left diagonal of A_k turn out to be very small or zero depending on the accuracy desired.

Method 2: (L-R transformation). A square matrix A can be expressed uniquely as the product of a lower triangular matrix L and upper triangular matrix R, provided the diagonal elements of one of these matrices are specified. In this method suggested by Rutishauser[13], all the diagonal elements of L are taken as 1. If $A = A_1$, we can decompose A_1 into L_1 and R_1 such that $A_1 = L_1 R_1$. The elements of L_1 and R_1 are determined from the original matrix A_1. We then form the reverse product $R_1 L_1$. It will be different from A_1. Let it be denoted by A_2. We can decompose A_2 into L_2 and R_2 such that $A_2 = L_2 R_2$. Similarly $R_2 L_2 = A_3 = L_3 R_3$ and so on, where A_3 is the matrix formed by the product of R_2 and L_2.

This yields an infinite sequence of matrices A_k. After a large number of steps this process converges resulting in an upper triangular matrix A_k while the lower triangular matrix converges to an identity matrix ; the number of transformations required for such convergence depends on the nature of the elements of the original matrix.

This $L - R$ transformation is also a similarity transformation and hence keeps the characteristic equation of A_k's invariant.

PROGRAMMING ASPECTS*

Method 1. (*a*) Formulae of transformation: The elements of the matrix A_k will be transformed due to the post multiplication of A_k by E_k as

$$a_{ki} \Leftarrow 0 \qquad i \Leftarrow k+1,\ k+2,\ k+3,\ \cdots,\ n-1,\ n \tag{i}$$

$$a_{ij} = a_{ij} - (a_{kj}/a_{kk})\, a_{ik} \tag{ii}$$

$$i \Leftarrow k+1,\ k+2,\ k+3,\ \cdots,\ n-1,\ n$$

$$j \Leftarrow k+1,\ k+2,\ k+3,\ \cdots,\ n-1,\ n$$

and the remaining will remain unchanged. The a's on the r h s of [ii] are the elements of A_k throughout the transformation and not in any case the elements of $A_k E_k$.

Now $A_k E_k$ has the same locations as A_k and to find $E_k^{-1} A_k E_k$ we will refer to the elements of A_k, since $A_k = A_k E_k$. Due to the pre multiplication by E_k^{-1} the transformed elements of A_k (*i.e.* $A_k E_k$) will be

$$a_{kj} = \sum_{p=k+1}^{n} a_{pj}\,(a_{kp}/a_{kk}) + a_{kj}, \quad j \leq k \tag{iii}$$

$$a_{kj} = \sum_{p=k+1}^{n} a_{pj}\,(a_{kp}/a_{kk}), \qquad j > k \tag{iv}$$

$$j = 1, 2, 3, \cdots, n-1, n.$$

The rest of the elements of A_k (*i e.* $A_k E_k$) will remain unchanged for a particular k. The elements a_{kp}/a_{kk} were the elements of the original matrix A_k *i.e* before the formation of $A_k E_k$. The other elements on the right hand side of [iii] and [iv] were the elements of A_k (*i.e.* $A_k E_k$) and not in any case the elements of $E_k^{-1} A_k$. For a single iteration $k = 1, 2, 3, \cdots, n-2, n-1$.

(*b*) Checks. Trace check is performed

(*c*) Memory requirements. About $n^2 + n + 65$ words are necessary of which n^2 locations are required for the storage of matrix A of order n, n locations for the storage of one row (or column) of A, and 65 locations for the program.

(*d*) Computing time per iteration. (referring to formulae *i*, *ii*, *iii*, *iv*)

It is about
$$\sum_{k=1}^{n-1} (n-k)\,\nu_1 + 2\sum_{k=1}^{n-1} (n-k)^2\,\nu_2 + \sum_{k=1}^{n-1} (n-k)^2\,\nu_3 + \left\langle \left[\sum_{k=1}^{n-1} (n-k)(n-k-1)\right] + (n-1) \right\rangle \nu_4 + \tau_1(n)$$

* '$\Leftarrow$' will always mean 'is replaced by' in discussions under 'Programming Aspects'

where the first 4 terms with ν's deleted indicate number of divisions, multiplications, subtractions and additions respectively, and $\tau_1(n)$ is the time needed by input and output units inclusive of that due to logical operations. ν_1, ν_2, ν_3 and ν_4 are the division, multiplication, subtraction and addition time respectively.

Method 2. (*a*) Formulae of $L-R$ transformation: We determine the R and L matrices as follows

$$R \text{ matrix}: \qquad \underset{i\leq j}{r_{ij}} = a_{ij} - \sum_{p=1}^{i-1} l_{ip}\, r_{pj}, \text{ assuming } l_{ii}=1 \tag{v}$$

$$L \text{ matrix}: \qquad \underset{i>j}{l_{ij}} = \Big(a_{ij} - \sum_{p=1}^{j-1} l_{ip}\, r_{pj}\Big)\Big/ r_{jj} \tag{vi}$$

For actual computation we assume a fixed value for j (which has to be taken as 1 first) and then proceed varying $i=1, 2, 3, \cdots, n-1, n$ and as a result we find r_{11}, then l_{21}, l_{31}, l_{41}, $\cdots$, $l_{n-1,1}$, $l_{n,1}$ respectively. We then take $j=2$ and proceed varying $i=1, 2, 3, \cdots, n-1, n$ and as a result we find r_{12}, r_{22}; then l_{32}, l_{42}, l_{52}, $\cdots$, $l_{n-1,2}$, $l_{n,2}$ respectively; next $j=3$, $i=1, 2, 3, \cdots n-1, n$; we find r_{13}, r_{23}, r_{33}, then l_{43}, l_{53}, l_{63}, $\cdots$, $l_{n-1,3}$, $l_{n,3}$ respectively and so on. In case $r_{jj}=0$, the procedure will, however, collapse.

The general formula for the product matrix $S=RL$, the results of which will constitute the first iteration step, is given as follows

$$\underset{i\geq j}{s_{ij}} = r_{ii}\, l_{ij} + \sum_{p=i+1}^{n} r_{ip}\, l_{pj} \tag{vii}$$

$$\underset{i<j}{s_{ij}} = r_{ij} \quad + \sum_{p=i+1}^{n} r_{ip}\, l_{pj} \tag{viii}$$

In the computer the elements $[a_{ij}]$ are replaced by the elements $[s_{ij}]$, or in other words, no s_{ij} – storage is necessary. Therefore s_{ij} in (vii) and (viii) can be substituted by a_{ij} This makes the program more automatic and more efficient and reduces the computing time. Moreover n^2 locations are preserved. It is important to note in this connection that storage often plays a very important part in this type of problems.

In the second iteration the transformed matrix $[a_{ij}]$ will be treated in the same way as it has been done for the first iteration with the original matrix $[a_{ij}]$. This process will cantinue till the continuously transformed matrix $[a_{ij}]$ becomes almost an upper triangular matrix or L matrix becomes nearly an identity matrix.

(*b*) Check. Trace check is performed.

(*c*) Memory requirements. About $n^2 + n\,(n+1)/2 + n\,(n+1)/2 + 90$ locations are necessary of which n^2 locations are required for the matrix A, $n\,(n+1)/2$ for L, $n\,(n+1)/2$ for R and about 90 words for the program.

(*d*) Computing time per iteration. (referring to formulae *v, vi, vii, viii*) It is about

$$\nu_1\, n\,(n-1)/2 + \nu_2 \left\langle \left[\sum_{p=2}^{n} p\,(p-1)/2\right] + (n-1)\,(n-2)/2 \right.$$

$$\left. + \sum_{p=0}^{n-1} \left[n\,(n+1)/2 - p\right] - n \right\rangle + \nu_3\left[n\,(n-1)/2 + (n-1)\,(n-2)/2\right]$$

$$+ \left\langle \sum_{p=2}^{n-1} p\,(p-1)/2 + \sum_{p=0}^{n-2} \left[n\,(n-1)/2 - p\right] \right\rangle \nu_4 + \tau_2\,(n)$$

where the first 4 terms with ν's deleted indicate number of divisions, multiplications, subtractions and additions respectively and $\tau_2\,(n)$ is the time needed by input and output units inclusive of that due to logical operations.

The difference between $\tau_1\,(n)$ and $\tau_2\,(n)$ is small, so that in the course of comparison these quantities can be neglected without affecting the process much.

*When $n = 5$, *i. e.* for a matrix of order 5, computing time per iteration for method 1 is $70\nu'_1 + 54\nu'_3 + \tau_1\,(n)$ and for method 2 it is $96\nu'_1 + 60\nu'_3 + \tau_2\,(n)$. When $n = 7$, computing time per iteration for method 1 is $203\nu'_1 + 161\nu'_3 + \tau_1\,(n)$ and for method 2, it is $260\nu'_1 + 182\nu'_3 + \tau_2\,(n)$. When $n = 5$, total storage required for method 1 is about 95 words while for method 2 it is 140 words. Therefore method 1 obviously saves on operational time and storage.

RESULTS

Calculations in all the examples are carried out in 8 dit floating point arithmetic and the results are retained correct up to 4 decimal places, unless otherwise stated. Zeros on the least significant side are avoided.

Example. 1 Matrix with double latent roots

$$A = \begin{bmatrix} 6 & 4 & 4 & 1 \\ 4 & 6 & 1 & 4 \\ 4 & 1 & 6 & 4 \\ 1 & 4 & 4 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 5 & 0 & 5 & 0 \\ 1 & 3 & 3 & -1 \end{bmatrix} \begin{matrix} \text{12 passes,} \\ \text{method 1,} \\ |A| = -375 \end{matrix} \qquad A \rightarrow \begin{bmatrix} 15 & 5 & 5 & 1 \\ 0 & 5 & 0 & 3 \\ 0 & 0 & 5 & 3 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{matrix} \text{12 passes,} \\ \text{method 2} \end{matrix}$$

*ν'_1 = average time for one multiplication or one division
ν'_3 = average time for one addition or one subtraction.

Example. 2. Matrix with disorder of latent roots

$$A=\begin{bmatrix}5&4&1&1\\4&5&1&1\\1&1&4&2\\1&1&2&4\end{bmatrix}\rightarrow\begin{bmatrix}10&0&0&0\\-3.5109&1&-16.0218&0\\2&0&5&0\\1&0&1.5&2\end{bmatrix}\quad \begin{array}{l}\text{15 passes, method 1,}\\ |A|=100\end{array}$$

The (2, 3)th element varies with the number of iterations. It becomes, .6173, .346, .1837, .0947, .0481, .0242, .0121, .0059, .0010, −.0036, −.0249, −.1278, −.6407, −3.2043, −16.0218, −80.1091, −400.545 in the 1st, 2nd, 3rd, , ... , 16th and 17th iterations respectively and does not converge.

$$\rightarrow\begin{bmatrix}10&-6.7513&2&1\\0&1&0&0\\.0001&-22.5028&5&1.5\\0&0&0&2\end{bmatrix}\quad \text{15 passes, method 2}$$

The latent roots are 10, 1, 5, 2. The differences between A_k's in method 1 and method 2 are due to *rounding* errors only. Round-off errors are, however, more in method 2.

Example 3 Real matrix with one pair of complex roots

$$A=\begin{bmatrix}2&3&1\\1&2&3\\3&1&2\end{bmatrix}\rightarrow\begin{bmatrix}+6&0&0\\-1.3373&2.5582&3.2722\\3&-2&-2.5582\end{bmatrix}\quad \begin{array}{l}\text{11 passes, method 1,}\\ |A|=18\end{array}$$

The exact roots are 6, $\pm i\sqrt{3}$

$$\rightarrow\begin{bmatrix}6.0002&-41108.1277&1\\.0001&-82222.5661&2\\5.7987&-.338\times10^{10}&82222.5665\end{bmatrix}\quad \text{11 passses, method 2}$$

The 10th iteration step is $\begin{bmatrix}6.0001&3.5&1\\-.0001&-.0002&2\\.0004&7.8946&.0002\end{bmatrix}$

It can be seen that the last two diagonal terms in the 11th iteration are very large. This is evidently due to the fact that the last two diagonal elements of the 10th iteration step are very small and that they occur in denominators of (iv).

In both the methods the last two diagonal terms go on oscillating within the rounding error due to division as follows:

TABLE

No. of iterations	Method 1	Method 2
1	$-17, \quad 18$	$-17, \quad 18$
2	$-\cdot 9391, \quad -\cdot 2609$	$-\cdot 9391, \quad -\cdot 2609$
3	$137\cdot 9167, \quad -138\cdot 0001$	$137\cdot 9168, \quad -138\cdot 0001$
4	$\cdot 6143\times 10^{-1}, \quad \cdot 2076\times 10^{-1}$	$\cdot 6143\times 10^{-1}, \quad \cdot 2076\times 10^{-1}$
5	$-1732\cdot 5731, \quad 1732\cdot 5800$	$-17346\cdot 0023, \quad 17346\cdot 0718$
6	$-\cdot 5211\times 10^{-2}, \quad -\cdot 1740\times 10^{-2}$	$-\cdot 5234\times 10^{-2}, \quad -\cdot 1724\times 10^{-2}$
7	$21463\cdot 6185, \quad -21463\cdot 6192$	$18328\cdot 7695, \quad -18328\cdot 77$
8	$\cdot 3662\times 10^{-3}, \quad \cdot 122\times 10^{-3}$	$\cdot 4883\times 10^{-3}, \quad \cdot 2441\times 10^{-3}$
9	$46736\cdot 7895, \quad -46736\cdot 7895$	$-64673\cdot 3895, \quad 64673\cdot 3895$
10	$0, \quad 0$	$-\cdot 2441\times 10^{-3}, \quad \cdot 2441\times 10^{-3}$
11	$2\cdot 5582, \quad -2\cdot 5582$	$-82222\cdot 5661, \quad 82222\cdot 5665$
12	$\cdot 2739\times 10^{-4}, \quad -\cdot 3074\times 10^{-4}$	$0, \quad \cdot 2441\times 10^{-3}$

The above tables also shows the effect of rounding errors due to divisions in both the methods. The rounding errors appear to play more important part in method 2.

If a real matrix possesses a pair of complex roots in its i-th and $(i+1)$th rows, the elements in the i-th row and those in the i-th column below the diagonal[13] will not converge in the normal way; the diagonal elements in the i-th and $[i+1]$th row may go on oscillating as has been shown above.

Example 4. Matrix of order 4 with a pair of complex roots

$$A = \begin{bmatrix} 4 & -5 & 0 & 3 \\ 0 & 4 & -3 & -5 \\ 5 & -3 & 4 & 0 \\ 3 & 0 & 5 & 4 \end{bmatrix} \qquad \text{Exact roots; } 12,\ 1 \pm i\sqrt{5},\ 2$$

For real roots the first and last diagonal elements of A_k converge to 12 and 2 after 15th interation For complex roots the $[2, 2]$th, $[2, 3]$th, $[3, 2]$th and $[3, 3]$th elements do not appear to converge in the normal way. In the 7th iteration they become

$$B = \begin{bmatrix} \cdot 0072 & -5\cdot 1709 \\ 5\cdot 042 & 1\cdot 9336 \end{bmatrix}$$

Now the roots μ_2, μ_3 of B can be given by

$$\begin{vmatrix} \cdot 0072-\mu & -5\cdot 1709 \\ 5\cdot 042 & 1\cdot 9336-\mu \end{vmatrix} = 0$$

Therefore $\mu_{2,3} = \cdot 9704 \pm i\, 5\cdot 0145$

The divergence of μ_2, μ_3 from the exact roots are due to rounding errors. In the case of single or multiple pairs of complex roots rounding errors play a very important part. These errors can be reduced by increasing the number of digits of the floating point arithmetic[17].

Example 5: A symmetric matrix with distinct latent roots (convergence is slow for this particular matrix):

$$A = \begin{bmatrix} 2 & 1 & -1 & 2 \\ 1 & 3 & 2 & -3 \\ -1 & 2 & 1 & -1 \\ 2 & -3 & -1 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 7.4683 & 0 & 0 & 0 \\ -1.2335 & 3.2753 & 0 & 0 \\ -4.1801 & -4.4262 & -1.6405 & 0 \\ 2 & 2.357 & 9146 & .8969 \end{bmatrix} \quad \begin{array}{l} \text{15 passes,} \\ \text{method 1,} \\ |A| = -36 \end{array}$$

$$\rightarrow \begin{bmatrix} 7.4683 & -1.2335 & -4.1801 & 2 \\ -0.0003 & 3.2753 & -4.4261 & 2.3569 \\ 0 & 0 & -1.6405 & .9147 \\ 0 & 0 & 0 & .897 \end{bmatrix} \quad \begin{array}{l} \text{15 passes,} \\ \text{method 2} \end{array}$$

Example 6: A defective matrix:

$$A = \begin{bmatrix} 6 & -3 & 4 & 1 \\ 4 & 2 & 4 & 0 \\ 4 & -2 & 3 & 1 \\ 4 & 2 & 3 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 5.5713 & -0.024 & 0 & 0 \\ 4.6833 & 4.9008 & 0 & 0 \\ 2.5078 & -0.4179 & 0.825 & 0.0024 \\ 4 & 5.7137 & -1.5374 & 0.7029 \end{bmatrix} \quad \begin{array}{l} \text{14 passes,} \\ \text{method 1} \end{array}$$

$$\rightarrow \begin{bmatrix} 5.5713 & -1.141 & 5.4818 & 1 \\ 0.0985 & 4.9008 & -7.2257 & -2.0478 \\ 0 & 0 & 0.825 & 0.4358 \\ 0 & 0 & -0.0086 & 0.7029 \end{bmatrix} \quad \begin{array}{l} \text{14 passes,} \\ \text{method 2} \end{array}$$

Example 7: Wilson's matrix:

$$A = \begin{bmatrix} 10 & 9 & 7 & 5 \\ 9 & 10 & 8 & 6 \\ 7 & 8 & 10 & 7 \\ 5 & 6 & 7 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 30.2887 & 0 & 0 & 0 \\ 42.27 & 3.3581 & 0 & 0 \\ 10.0199 & 1.0071 & .8431 & 0 \\ 5 & .7022 & -.3168 & .0102 \end{bmatrix} \quad \begin{array}{l} \text{11 passes,} \\ \text{method 1,} \\ A = 1 \end{array}$$

$$\rightarrow \begin{bmatrix} 30.2887 & 42.27 & 10.0199 & 5 \\ 0 & 3.8581 & 1.0071 & .7022 \\ 0 & 0 & .8431 & -.3168 \\ 0 & 0 & 0 & .0102 \end{bmatrix} \quad \begin{array}{l} \text{11 passes,} \\ \text{method 2} \end{array}$$

Example 8 : A stochastic matrix of order 4

A square matrix $A = (a_{ij})$ with non-negative elements is called stochascic[20] if

$$\sum_{j=1}^{n} a_{ij} = 1 \qquad i = 1, 2, 3, \ldots, n-1, n$$

Such matrices are of great importance in the theory of probability.

$$A = \begin{bmatrix} .31 & .32 & .33 & .04 \\ .41 & .12 & .07 & .4 \\ .21 & .24 & .25 & .3 \\ .57 & .15 & .18 & .1 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & 0 & 0 \\ .9335 & -.1555 & -.0368 & 0 \\ -4.885 & 1.951 & -.0971 & 0 \\ .57 & -.2061 & 0129 & 0326 \end{bmatrix} \quad \begin{array}{l} \text{14 passes,} \\ \text{method 1,} \\ |A| = .0028 \end{array}$$

$$\to \begin{bmatrix} 1 & .5009 & .2608 & .04 \\ 0 & -.1555 & -.8832 & .36 \\ 0 & .0812 & -.097 & .0835 \\ 0 & 0 & 0 & .0326 \end{bmatrix} \quad \begin{array}{l} \text{14 passes,} \\ \text{method 2,} \end{array}$$

Example. 9. Hilbert's matrix

These matrices are typical examples of near singular matrices. Their singularity becomes more pronounced as their order is increased In the following Hilbert's matrices of order 3, 4 and 5 have been considered, retaining every element correct up to 8 decimal places.

Hilbert's matrix of order 3 (determinant value $= .463 \times 10^{-3}$)

$$\to \begin{bmatrix} 1.4083 & 0 & 0 \\ .8455 & .1223 & 0 \\ .3333 & .0647 & .2687 \times 10^{-4} \end{bmatrix} \quad \text{4 passes, method 1}$$

Method 2 gives the same results (correct up to 4 decimal places) in the same number of passes as method 1.

Hilbert's matrix of order 4 (determinant value $= .1653 \times 10^{-6}$)

$$\to \begin{bmatrix} 1.5002 & 0 & 0 & 0 \\ 1.1031 & .1491 & 0 & 0 \\ .7178 & .1483 & .6738 \times 10^{-2} & 0 \\ .25 & .5746 \times 10^{-1} & .3918 \times 10^{-2} & .967 \times 10^{-4} \end{bmatrix} \quad \text{6 passes, method 1}$$

Method 2 gives the same result as method 1.

Hilbert's matrix of order 5 (determinant value $= .3743 \times 10^{-11}$)

According to both the methods the roots are 1.5671, $.2085$, $.1141 \times 10^{-1}$ $.3059 \times 10^{-3}$, $.3282 \times 10^{-5}$ in 7 passes.

Example 10.

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} \qquad |A| = 16$$

Roots found by method 1 and method 2 are 2, 2, 2, 2. Roots obtained by 3.1 code (real code) of Eberlein[16] are shown underscored in the matrix.

$$\begin{bmatrix} \underline{2.008} & 0 & .008 & 0 \\ 0 & \underline{1.992} & 0 & -.008 \\ -.008 & 0 & \underline{2.008} & 0 \\ 0 & .008 & 0 & \underline{1.991} \end{bmatrix}, \quad 10 \text{ passes.}$$

Acknowledgement

The author wishes to express his sincere thanks to Prof P. L. Bhatnagar for taking keen interest in the preparation of this paper and to Prof. S. Dhawan for having kindly permitted the publication of this paper.

References

1. Wilkinson, J. H. .. *Computer J.*, 1958, **1**, 90
2. ———, .. *Ibid*, 1958, **1**, 148
3. ———, .. *J.A.C.M.*, 1959, **6**, 336
4. ———, .. *Computer J.*, 1960, **3**, 23
5. ———, .. *Num. Math.*, 1960, **2**, 319
6. ———, and Rollett .. *Computer J.*, 1961, **4**, 177
7. ———, .. *Ibid*, 1962, **5**, 61
8. Francis, J.G.F. .. *Ibid*, 1961, **4**, 265
9. ———, .. *Ibid*, 1961, **4**, 332
10. Forsythe G. E. and Henrici, P. .. *Trans. Am. Math. Soc.*, 1960, **94**, 1
11. ———, .. *Quart. J Mech.*, 1952, **5**, 191
12. Householder, A.S. and Bauer, F. L. .. *Num. Math.*, 1959, **1**, 29
13. Rutishauser, H. .. *Nat. Bur. Stand, Appl. Math. Ser.* 1958, No. 49, 47
14. Handscomb, D. C. .. *Computer J.*, 1962, **5**, 139
15. Givens, W. .. *J. Soc. Indust. Appl. Math.*, 1958, **6**, 26
16. Eberlein, P. J. .. *J. Soc. Indust. Appl. Math.*, 1962, **10**, 74
17. Sen, S. K. .. *J. Indian Inst. of Sci*, 1967, **49**, 37
18. Pope, D. A. and Tompkins, C. .. *J.A.C.M.*, 1957, **4**, 459
19. Goldstine, H. H., Murray, F. J. and Neumann von J. .. *Ibid*, 1959, **6**, 59
20. Brauer, A. .. "Recent Advances in Matrix Theory," edite by Schneider, H., Univ. of Wisconsin Press, 1964, 10.