MATHEMATICS OF COMPUTATION, VOLUME 30, NUMBER 133
JANUARY 1976, PAGES 132–142

The QR Algorithm and Hyman's Method on Vector Computers*

By Robert C. Ward

Abstract. The implementation on vector computers of the QR algorithm and of iterative schemes based on obtaining the determinant and its derivatives by Hyman's method are presented. It is shown that iterative schemes based on Hyman's method will probably be more efficient than the QR algorithm on vector computers for large matrices. A theoretical comparison of the Laguerre iterative scheme with the QR algorithm is presented using the latest available CDC STAR-100 instruction execution times. In addition, the results of several test cases run on the Laguerre-Hyman algorithm on a serial computer are reported.

1. Introduction. The use of the QR algorithm for numerically solving the algebraic eigenvalue problem $Ax = \lambda x$, where A is an $n \times n$ upper Hessenberg matrix, has steadily increased since its introduction by Francis [1] in 1961. This algorithm and its many variants, such as the QL algorithm and the rational QR algorithm, have become fixtures in most mathematical subroutine libraries. In fact, most of the algorithms recommended by Wilkinson and Reinsch [6] for solving the various classes of eigenvalue problems are based upon the QR algorithm. Since the operations found in this algorithm are basically vector operations, one would expect that the QR algorithm would be the basic eigenvalue solver for the new vector computers, such as the Control Data Corporation STAR and the Texas Instruments ASC computers. The purpose of this paper is to examine the QR algorithm on vector computers and to compare these results with iterative schemes based on obtaining the determinant and its derivatives by Hyman's method. Particular attention will be paid to the Laguerre iterative scheme.

2. Comparison Basis on Vector Computers. On serial computers, the relative merits of linear algebra algorithms were determined primarily by counting the number of arithmetic operations. This operation count was usually expressed in terms of the highest power of n, the order of the matrix problem. Using the problem of solving a system of linear equations as an example, the operation count for the Gaussian elimination algorithm is $n^3/3$ while the operation count for the Gauss-Jordan elimination algorithm is $n^3/2$. Hence, the former algorithm should usually be preferred on serial computers.

The method of comparing algorithms on vector computers will be slightly different due to the nature of these computers. Since vector operations are expected to be, and should be, the dominant time-consuming operations for linear algebra algorithms,

Received January 20, 1975; revised June 4, 1975.

AMS (MOS) subject classifications (1970). Primary 65F15; Secondary 68A10, 15A18.

Key words and phrases. Vector computers, eigenvalues, QR algorithm, Hyman's method.

*A portion of this work was completed while the author was employed by NASA, Langley Research Center, Hampton, Virginia.

the relative merits of these algorithms on vector computers will be determined by summing the number of clocks (minor cycles) required for the vector instructions. One cannot merely count the number of arithmetic operations on these computers since the time required for a single arithmetic operation is not constant throughout the algorithm. This is easily seen by examining the timing equation for vector instructions. The general equation for determining the number of clocks required for a vector instruction is given by $S + c\lceil l/p \rceil$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, S is called the start-up time, c is called the incremental cost, p is called the increment size and is related to memory access, and l denotes the length of the vector.

The assumption that $S \gg c/p$ will be used throughout this paper. This assumption is true on the existing vector computers and is expected to be a general characteristic of all vector computers. Thus, considerable time is spent initially to start the vector instruction and this time is independent of the length of the vector. This indicates that the use of long vectors is advantageous on these computers. Note, however, that the maximum number of "free" arithmetic operations in a vector instruction is p and increasing the vector length without sufficient justification would be inefficient. This is a significant difference from parallel computers where there is no penalty for using all the available processors even if these extra generated answers are never used.

Since the values of S, c, and p are expected to depend upon the type of vector instruction to be executed, a subscript notation will be used in this paper to denote the various values of these parameters. A subscript of $+$, $*$, and Σ denotes addition/subtraction, multiplication, and dot product, respectively. For example, S_Σ denotes the start-up time for a vector dot product instruction, c_+ denotes the incremental cost for a vector addition or subtraction instruction, etc.

An important point which should not be overlooked is that some vector computers require the components of each of the vectors in a vector instruction to be in contiguous storage locations. This paper will only consider vector computers of this type (e.g. CDC STAR). Thus, algorithms must know whether the matrix is stored columnwise or rowwise and set up the vector instructions accordingly.

3. **QR Iteration Clock Count.** One double shift QR iteration on an $n \times n$ upper Hessenberg matrix consists of applying a sequence of $n - 1$ Householder similarity transformations to the matrix. The 4th transformation or step in this sequence will be briefly described for a matrix of order 8 stored columnwise in a vector computer. Before the 4th transformation is applied, the matrix A has the following form:

$$\begin{bmatrix}
x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x \\
 & & x & x & x & x & x & x \\
 & & x & x & x & x & x & x \\
 & & & & & x & x & x \\
 & & & & & & x & x
\end{bmatrix}$$

The 4th Householder transformation P_4 is given by the expression $I - uv^T$ where $u = (0, 0, 0, 1, u_2, u_3, 0, 0)^T$ and $v^T = (0, 0, 0, v_1, v_2, v_3, 0, 0)$. The vectors u and v are chosen such that a_{53} and a_{63} are annihilated. Since the matrix is stored columnwise, the premultiplication of A by P_4 requires one dot product, one vector multiplication (actually, a vector multiplied by a scalar which will be treated as a vector multiplication in this paper), and one vector subtraction for each of the last 5 columns of A. The vector length for each of these instructions is 3.

The matrix A now has the following form:

$$\begin{bmatrix}
x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x \\
 & & 0 & x & x & x & x & x \\
 & & 0 & x & x & x & x & x \\
 & & & & & x & x & x \\
 & & & & & & x & x
\end{bmatrix}$$

Recalling that the matrix is stored columnwise and denoting the kth column of A by $a^{(k)}$, the postmultiplication of A by P_4 begins by multiplying $a^{(5)}$ and $a^{(6)}$ by u_2 and u_3, respectively. Then, $a^{(4)}$ is added to $u_2a^{(5)}$; and this sum is then added to $u_3a^{(6)}$. This vector sum $(a^{(4)} + u_2a^{(5)} + u_3a^{(6)})$ is multiplied by v_1 and subtracted from $a^{(4)}$, then multiplied by v_2 and subtracted from $a^{(5)}$, and finally multiplied by v_3 and subtracted from $a^{(6)}$. Thus, the postmultiplication requires one vector multiplication and one vector addition of vector length 6, and four vector multiplications and four vector additions/subtractions of vector length 7.

TABLE 1

Computation of $A' = P_kA$ in QR *Algorithm*

Operation	*Serial Count*	*Vector Count*
$w_i = v^Ta^{(i)}$ $i = k, k + 1, \ldots, n$	$3(n - k + 1)$ multiplications $2(n - k + 1)$ additions	$(n - k + 1)$ dot products of vector length 3
$x^{(i)} = w_iu$ $i = k, k + 1, \ldots, n$	$2(n - k + 1)$ multiplications	$(n - k + 1)$ vector multiplications of vector length 3
$a'^{(i)} = a^{(i)} - x^{(i)}$ $i = k, k + 1, \ldots, n$	$3(n - k + 1)$ subtractions	$(n - k + 1)$ vector subtractions of vector length 3

For the kth step with $2 \leqslant k \leqslant n - 2$, the vector and serial operations required in the premultiplication of A by P_k is given in Table 1 with the following notation:

$$A' = (I - uv^T)A = P_kA,$$
$$u^T = (0, \ldots, 0, 1, u_2, u_3, 0, \ldots, 0),$$
$$v^T = (0, \ldots, 0, v_1, v_2, v_3, 0, \ldots, 0).$$

Similarly for $k = 1$, one dot product of vector length 2, $(n - 1)$ dot products of vector length 3 and n vector multiplications and n vector additions/subtractions of vector length 3 are required; and for $k = n - 1$, two dot products, two vector multiplications, and two vector additions/subtractions with a vector length of 2 are required. Thus, the total number of clocks required for the premultiplications in one QR iteration on a vector computer is given by

$$\sum_{k=1}^{n-2}\left[(n-k+1)\left\{S_\Sigma + c_\Sigma\left[\frac{3}{p_\Sigma}\right] + S_* + c_*\left[\frac{3}{p_*}\right] + S_+ + c_+\left[\frac{3}{p_+}\right]\right\}\right]$$
$$+ 2\left\{S_\Sigma + S_* + c_*\left[\frac{2}{p_*}\right] + S_+ + c_+\left[\frac{2}{p_+}\right]\right\} + 3c_\Sigma\left[\frac{2}{p_\Sigma}\right] - c_\Sigma\left[\frac{3}{p_\Sigma}\right]$$
$$= \left\{S_\Sigma + S_* + S_+ + c_\Sigma\left[\frac{3}{p_\Sigma}\right] + c_*\left[\frac{3}{p_*}\right] + c_+\left[\frac{3}{p_+}\right]\right\}\left(\frac{1}{2}n^2 + \frac{1}{2}n\right) + O(1).$$

The vector and serial operations required in the postmultiplication of A' by P_k for $k \leqslant n - 3$ is given in Table 2 where

$$A'' = A'(I - uv^T) = A'P_k.$$

Similarly for $k = n - 2$, five vector multiplications and five vector additions/subtractions of vector length n are required, and for $k = n - 1$, three vector multiplications and three vector additions/subtractions of vector length n are required. Thus, the total number of clocks required for the postmultiplications in one QR iteration on a vector computer is given by

$$\sum_{k=1}^{n-3}\left\{S_* + c_*\left[\frac{k+2}{p_*}\right] + S_+ + c_+\left[\frac{k+2}{p_+}\right] + 4\left(S_* + c_*\left[\frac{k+3}{p_*}\right] + S_+ + c_+\left[\frac{k+3}{p_+}\right]\right)\right\}$$
$$+ 8\left\{S_* + c_*\left[\frac{n}{p_*}\right] + S_+ + c_+\left[\frac{n}{p_*}\right]\right\}$$
$$= \frac{5}{2}\left[\frac{c_*}{p_*} + \frac{c_+}{p_+}\right]n^2 + 5\left[S_* + S_+ + \frac{1}{2}(c_* + c_+)\right]n + O(1).$$

The total number of clocks for one QR iteration eliminating the order of unity term is then given by

$$\frac{1}{2}\left[S_\Sigma + S_* + S_+ + c_\Sigma\left[\frac{3}{p_\Sigma}\right] + c_*\,\frac{3}{p_*} + c_+\left[\frac{3}{p_+}\right] + 5\left(\frac{c_*}{p_*} + \frac{c_+}{p_+}\right)\right]n^2$$
$$(1)\qquad + \frac{1}{2}\left[S_\Sigma + 11(S_* + S_+) + 5(c_* + c_+) + c_\Sigma\left[\frac{3}{p_\Sigma}\right] + c_*\left[\frac{3}{p_*}\right] + c_+\left[\frac{3}{p_+}\right]\right]n.$$

TABLE 2

Computation of $A'' = A'P_k$ *in* QR *Algorithm*

Operation	*Serial Count*	*Vector Count*
$w^{(2)} = u_2 a'^{(k+1)}$	$k + 2$ multiplications	1 vector multiplication of vector length $k + 2$
$w^{(3)} = u_3 a'^{(k+2)}$	$k + 3$ multiplications	1 vector multiplication of vector length $k + 3$
$x = a'^{(k)} + w^{(2)}$	$k + 2$ additions	1 vector addition of vector length $k + 2$
$y = x + w^{(3)}$	$k + 3$ additions	1 vector addition of vector length $k + 3$
$z^{(i)} = v_i y$ $i = 1, 2, 3$	$3(k + 3)$ multiplications	3 vector multiplications of vector length $k + 3$
$a''^{(k+i-1)} = a'^{(k+i-1)} - z^{(i)}$ $i = 1, 2, 3$	$3(k + 3)$ subtractions	3 vector subtractions of vector length $k + 3$

The significant point to note in this total is the start-up times appearing in the coefficient of the n^2 term. This indicates that one QR iteration, which required $5n^2$ multiplications and $5n^2$ additions/subtractions on a serial computer, requires $3n^2/2$ *vector* operations on a vector computer. Thus, the QR iteration does not "vectorize" very well and possibly could be slower on a vector computer than on a reasonably fast serial computer.

Using vectors of length 3 in the premultiplication is the reason for the poor vectorization. One could obviously alter this by storing the matrix rowwise, but this leads to using vectors of length 3 in the postmultiplication resulting in the same total number of clocks as in the columnwise algorithm.

4. Clock Count for Iterations Based on Hyman's Method.

In order to have a viable algorithm to compare with the QR algorithm, the timing formula for an iterative scheme where the determinant and its derivatives are obtained by Hyman's method will be presented in this section. The determinant of $(A - zI)$, aside from a nonzero constant multiplier, can be computed by Hyman's method by performing a single backsubstitution. This is illustrated by the following example where $n = 6$:

$$\begin{array}{ccccc} A - zI & & y & & b \\ \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ & x & x & x & x & x \\ & & x & x & x & x \\ & & & x & x & x \\ & & & & x & x \end{bmatrix} & & \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ 1 \end{bmatrix} & = & \begin{bmatrix} f(z) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

By setting $y_n = 1$, all the components of the y vector are found using rows n through 2 consecutively of $(A - zI)$. Using the first row of $(A - zI)$, the value of $f(z)$ which differs from $\det(A - zI)$ by the factor $\pm a_{21} a_{32} \cdots a_{n,n-1}$ is then computed. The values of $f'(z)$ and higher derivatives can be computed in a similar fashion requiring one backsubstitution per derivative (see Wilkinson [5]).

Howser and Lambiotte [3] have discussed in some detail the vectorization of the backsubstitution process for the CDC STAR computer. If the matrix is stored rowwise, the dot product instruction would be used and if the matrix is stored columnwise, a vector multiplication followed by a vector addition/subtraction would be used. Note that the former storage technique requires only one vector start-up while the latter requires two vector start-ups. But c_Σ / p_Σ is expected to be larger than $c_* / p_* + c_+ / p_+$; and hence, the matrix should be stored rowwise when n is "small" and columnwise when n is "large". In this paper, it is assumed that n is large and the matrix is stored columnwise. Thus, the total number of clocks required to compute $f(z)$ and the first $r - 1$ derivatives of $f(z)$ when z is real is given by

$$r \sum_{k=n-1}^{1} \left\{ S_* + c_* \left[\frac{k}{p_*} \right] + S_+ + c_+ \left[\frac{k}{p_+} \right] \right\} \tag{2}$$
$$= \frac{r}{2} \left[\frac{c_*}{p_*} + \frac{c_+}{p_+} \right] n^2 + r \left[S_* + S_+ - \left(\frac{c_*}{p_*} + \frac{c_+}{p_+} \right) + \frac{1}{2} (c_* + c_+) \right] n .$$

When z is complex, the number of clocks would be twice the above number using a vectorization of the technique described in Wilkinson [5]. This yields the type of result expected for linear algebra algorithms on a vector computer. That is, a process which requires on the order of n^2 operations on a serial computer requires on the order of n vector operations on a vector computer.

The Laguerre iteration scheme has been coded for the CDC 6000 series computers at NASA, Langley Research Center to use in a comparison with the QR algorithm. The basic roundoff error for these computers is approximately 10^{-14}. The salient features of this code are as follows:

(1) The determinant and its first two derivatives are required.

(2) The criterion for accepting an eigenvalue is based on passing two tests. The first test, which indicates that the iterates are beginning to converge, is

$$|x_k - x_{k-1}| \leqslant 10^{-7} \qquad \text{if } |x_{k-1}| < 10^{-4},$$

$$|x_k - x_{k-1}| \leqslant |x_{k-1}| 10^{-3} \quad \text{if } |x_{k-1}| \geqslant 10^{-4}.$$

After passing this test, the iteration stops and x_{k-1} is declared an eigenvalue when $|x_k - x_{k-1}| > |x_{k-1} - x_{k-2}|$.

(3) The suppression of previously computed eigenvalues is accomplished by iterating on the function $g_r(z)$ defined by

$$g_r(z) = \frac{f(z)}{\prod_{i=1}^{r} (z - \lambda_i)},$$

where $\lambda_1, \lambda_2, \ldots, \lambda_r$ are the previously computed zeros of $f(z)$.

(4) The starting value for the iteration to each eigenvalue may be given to the subroutine by the user. If not, the starting value for computing the first eigenvalue is taken to be $\|A\|_\infty$. Thereafter, the starting value for finding the $(r + 1)$th eigenvalue if $g_r'(\lambda_r) \neq 0$ is taken to be $\lambda_r - g_r(\lambda_r)/g_r'(\lambda_r)$, where λ_r is the rth eigenvalue or $\|A\|_\infty$ if $g_r'(\lambda_r) = 0$.

(5) If the convergence appears to be linear after the first test has been passed in the acceptance criterion, an attempt is made to obtain cubic convergence by trying the iteration assuming the eigenvalue which is being approached has multiplicity 2. If convergence still appears linear, a multiplicity of 3 is tried. If this also fails, the multiplicity is reset to 1 and the eigenvalue is found through linear convergence.

(6) The matrix is partitioned into submatrices if a negligible subdiagonal is found.

(7) After all the eigenvalues of a matrix or submatrix have been found, the sum of these eigenvalues are compared with the trace of the matrix or submatrix. If the test

$$\left| \sum_{i=1}^{n} \lambda_i - \text{trace}(A) \right| \leqslant 10^{-9} \qquad \text{if } |\text{trace}(A)| < 10^{-4},$$

$$\left| \sum_{i=1}^{n} \lambda_i - \text{trace}(A) \right| \leqslant |\text{trace}(A)| 10^{-5} \quad \text{if } |\text{trace}(A)| \geqslant 10^{-4}$$

fails, the iteration starts over again using a different starting value for each eigenvalue.

(8) A maximum number of thirty-two iterations are allowed to converge to each eigenvalue.

Most of these points are discussed in detail by Parlett [4] or Chapter 7 of Wilkinson [5].

The starting value described in item (4) is not crucial. If a starting value which is larger than the largest real eigenvalue is chosen, the real eigenvalues have a tendency to be found in algebraically decreasing order. This is the motivation for using $\|A\|_\infty$. Gerschgorin theorems could be employed to determine another reasonable starting value. Also, the simple choice of using 0 as the starting value has some merit.

5. Theoretical Comparison on the CDC STAR. Two CDC STAR computers have recently been delivered to Lawrence Livermore Laboratory at Livermore, California. The initial testing of these computers indicate that the actual execution times for the basic vector instructions are close to the execution times predicted by CDC. Thus, a theoretical comparison between the QR algorithm and the Laguerre-Hyman algorithm can be made using this information. The predicted STAR-100 execution times (number of clocks) for the vector instructions of interest are as follows:

Addition/subtraction:	$94 + 4\left[\frac{l}{8}\right]$,
Multiplication:	$154 + 8\left[\frac{l}{8}\right]$,
Dot product:	$100 + 4l$.

Based on this timing, the number of clocks required for one QR iteration from formula (1) is $189\frac{3}{4}n^2 + 1456n$, while the number of clocks for one Laguerre-Hyman iteration from Eq. (2) is $2\frac{1}{4}n^2 + 757\frac{1}{2}n$. Thus, for every large n, one Laguerre-Hyman iteration is faster than one QR iteration by a factor of roughly 84! These timing formulas also indicate why the term corresponding to the second highest power of n should be included. On vector computers, this term will probably contain most of the start-up times and usually will be of one or two orders of magnitude greater than the term corresponding to the highest power of n.

To carry this comparison further, one can make some reasonable assumptions concerning the QR and Laguerre-Hyman algorithms and obtain estimates for the number of clocks required to find all the eigenvalues of an $n \times n$ real matrix. The assumptions for the QR algorithm are first that the average number of iterations per eigenvalue is $1\frac{3}{4}$. This has been reported by Wilkinson and Reinsch [6], and it also has been this author's experience. Second, the average order of the deflated matrix used in the QR algorithm is $2n/3$. This, at first, appears to be an unrealistic assumption since the matrix begins with order n and is deflated by one or two until the order is zero. Using $2n/3$ for the average order instead of $\frac{1}{2}n$ attempts to offset the fact that more iterations are required while the matrix has high order than later in the algorithm. Under these two assumptions, the total number of clocks required by the vector instructions in the QR algorithm to find all the eigenvalues of an $n \times n$ matrix is given by

$$1\,\frac{3}{4}\,n\left[189\,\frac{3}{4}\left(\frac{2}{3}\,n\right)^2 + 1456\left(\frac{2}{3}\,n\right)\right] = 147\,\frac{7}{12}\,n^3 + 1698\,\frac{2}{3}\,n^2.$$

The assumption for the Laguerre-Hyman algorithm is that the average number of iterations per eigenvalue is seven. This has been this author's experience on computing the eigenvalues of twenty-five test cases by this algorithm on serial computers. These test cases all had distinct eigenvalues and the order of matrices ranged between 5 and 75. (Additional test cases are discussed in the next section.) Under this assumption, the total number of clocks required by the vector instructions in the Laguerre-Hyman algorithm to find all the eigenvalues of an $n \times n$ matrix is given by

$$7n\left[2\frac{1}{4}n^2 + 757\frac{1}{2}n\right] = 15\frac{3}{4}n^3 + 5302\frac{1}{2}n^2.$$

Thus, this comparison indicates that the QR algorithm would be faster for matrices of order 27 or smaller and the Laguerre-Hyman algorithm would be faster for matrices of order greater than 27. For a matrix of order 100, the Laguerre-Hyman algorithm would be faster than the QR algorithm by a factor of almost 2½.

When comparisons are actually made on the STAR computer, the order of the matrix for which the Laguerre-Hyman algorithm becomes faster will probably be larger than indicated in the previous paragraph. This will be due to the presence of considerable more logic and scalar arithmetic in the Laguerre-Hyman algorithm than in the QR algorithm. Also, the ability of the QR algorithm to detect consecutive "small" subdiagonals to decrease the computations in an iteration will tend to increase this value of n.

Another point which has not been discussed thus far is that of paging on the STAR computer. Paging is the process of bringing a block of storage into central memory or releasing a block of storage. On this point, the Laguerre-Hyman algorithm would have an advantage if the complete matrix could not be contained in central memory. Each column of the matrix is required only three times in a Laguerre-Hyman iteration, once for each of the three backsubstitutions. Since the backsubstitutions can be performed simultaneously, a matrix column need only be brought into central memory once for each iteration. In the QR algorithm, the rth column of the matrix for $r \neq n$ is required $r + 1$ times in only one QR iteration. The nth column is required n times. Thus, the nth column may have to be brought into and released from central memory n times in one QR iteration.

6. Test Results on Laguerre-Hyman Algorithm. In addition to the test cases mentioned in the previous section, several matrices from Gregory and Karney [2] which have eigenvalues that are numerically difficult to compute have been used as test matrices for the Laguerre-Hyman algorithm. These include nondefective matrices with multiple eigenvalues (examples 4.8 and 4.10), defective nonderogatory matrices (examples 5.11 and 5.12), a defective derogatory matrix (example 5.27), an orthogonal matrix (example 5.22 with $\epsilon = 1$), and matrices which have distinct ill-conditioned eigenvalues (examples 5.14 and 5.23 with $\epsilon = 10^{-10}$). In all cases, the Laguerre-Hyman algorithm performed satisfactorily in that the returned eigenvalues were within the domain of indeterminacy of the exact eigenvalues. In examples 4.10, 5.11, and 5.22, the algorithm reached a point where it repeatedly converged to the same eigenvalues and naturally missed some of the true eigenvalues. But the comparison with the trace detected this error and on the second try, using different starting values, the eigenvalues with the correct multiplicities were found. The average number of Laguerre iterations per eigenvalue for these special matrices was 5.9.

To indicate the type of results obtained from these test cases on a CDC 6400 serial computer, the results from example 5.12 are presented in this paper. The matrix is given by

$$\begin{bmatrix} 10 & -19 & 17 & -12 & 4 & 1 \\ 9 & -18 & 17 & -12 & 4 & 1 \\ 8 & -16 & 15 & -11 & 4 & 1 \\ 6 & -12 & 12 & -10 & 4 & 1 \\ 4 & -8 & 8 & -6 & 1 & 2 \\ 2 & -4 & 4 & -3 & 1 & 0 \end{bmatrix}$$

The eigenvalues are $1, \pm i$, and -1 with a multiplicity of 3 and only one eigenvector. The eigenvalues returned from the Laguerre-Hyman algorithm and the number of iterations necessary to compute them are given in Table 1.

TABLE 1

Eigenvalues		*Iterations*
Real Parts	*Imaginary Parts*	
.99999999999961	0.	5
5.93×10^{-13}	.99999999999941	6
5.93×10^{-13}	$-.99999999999941$	–
$-.99998566379579$	2.483376×10^{-5}	19
$-.99998566379579$	-2.483376×10^{-5}	–
-1.0000290725328	0.	9

The eigenvalues 1 and $\pm i$ are computed to within a few digits of machine accuracy and the eigenvalues -1 are computed to approximately cube root of machine accuracy.

On all the many test matrices which have been tried, the Laguerre-Hyman algorithm performed satisfactorily except on those matrices which contained eigenvalues so ill-conditioned that the first test for convergence could not be passed; that is, the eigenvalues could not be found accurately to three significant digits. On these matrices, the iteration count was exceeded and the appropriate error flag was set..

Acknowledgment. The author wishes to thank Dr. Jules J. Lambiotte for the helpful discussions on vector computers during the course of this investigation.

The author is also grateful to the referee for the valuable suggestions on improving the manuscript.

Union Carbide Corporation Nuclear Division
Computer Sciences Division
Oak Ridge, Tennesee 37830

1. J. G. F. FRANCIS, "The *QR* transformation: A unitary analogue to the *LR* transformation. I, II," *Comput. J.*, v. 4, 1961/62, pp. 265–271, 332–345. MR **23** #B3143; **25** #744.

2. R. T. GREGORY & D. L. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Wiley, New York, 1969. MR **40** #6752.

3. L. M. HOWSER & J. J. LAMBIOTTE, JR., *STAR Adaption for Two Algorithms Used on Serial Computers*, NASA Technical Memorandum X-3003, June 1974.

4. BERESFORD PARLETT, "Laguerre's method applied to the matrix eigenvalue problem," *Math. Comp.*, v. 18, 1964, pp. 464–485. MR 29 #2948.

5. J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965. MR 32 #1894.

6. J. H. WILKINSON & C. REINSCH, *Handbook for Automatic Computation*. Vol. II, Linear Algebra, Springer-Verlag, New York, 1971.