SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1059–1063, September 1994

REMARK ON ALGORITHMS TO FIND ROOTS OF POLYNOMIALS*

S. GOEDECKER†

Abstract. The problem of finding the roots of a polynomial is equivalent to finding the eigenvalues of an upper Hessenberg matrix, which can be done with the QR algorithm. It is shown that the QR algorithm has considerable advantages over other standard algorithms to find the roots of a polynomial.

Key words. roots of polynomials

AMS subject classifications. 11C++, 65Y25

1. The QR algorithm to find roots of polynomials.

It has already been realized in the literature [1] that the problem of finding the roots of the polynomial

$$a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0 = 0 \tag{1}$$

is equivalent to finding the eigenvalues of the upper Hessenberg matrix A

$$\begin{pmatrix} \frac{a_{n-1}}{-a_n} & \frac{a_{n-2}}{-a_n} & \frac{a_{n-3}}{-a_n} & \frac{a_{n-4}}{-a_n} & \dots & \frac{a_0}{-a_n} \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ .. & . & . & . & . & . \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}.$$

If λ is a root of the polynomial (1), λ is an eigenvalue of A and the associated left eigenvector v is given by

$$\begin{pmatrix} \lambda^{n-1} \\ \lambda^{n-2} \\ \lambda^{n-3} \\ \dots \\ \lambda^0 \end{pmatrix}.$$

Plugging in the above eigenvector in the eigenvalue equation

$$Av = \lambda v$$

immediately gives (1). This equivalence has been used in [1] to find the roots by a procedure related to the well-known inverse iteration.

Instead of using inverse iteration, the eigenvalues of an upper Hessenberg matrix can of course also be found by the QR algorithm [2].

---

*Received by the editors October 13, 1992; accepted for publication (in revised form) July 5, 1993. The author was supported in part by a grant from the Swiss National Foundation.

†Ecole Polytechnique Féderale de Lausanne, Institut de Physique Appliquée, CH-1015 Lausanne, Switzerland. Current address: Theory Center, Cornell University, Ithaca, New York 14853-3801 (goedeck@titan.tc.cornell.edu).

2. Test criteria.

We compared the QR algorithm, as it is implemented in the Eispack library, with standard root finders of the Numerical Algorithms Group (NAG) and International Mathematics and Statistical Libraries (IMSL) under the following criteria of reliability, accuracy, and speed for both vector and nonvector machines.

• *Reliability.* We tested whether the program can find all the roots without overflow. This is of course not only a characteristic of the algorithm but depends also on the range of floating point numbers that can be represented on the computer. Overflows are a serious problem on VAX computers, where the largest double precision number is of the order of 10^{38}.

• *Accuracy.* The exact roots were not known for most of the test polynomials. To assess the accuracy of the numerical solution we therefore calculated the value of the polynomial at the values of the numerically found roots. In the interval $[-1,1]$ the derivative of the polynomial is bound by $\sum_i i\ |a_i|$. Since this bound is reasonably small in the cases we considered, the value of the polynomial at a numerical root gives an indication of the precision of the numerical root. The situation is however quite different outside the interval $[-1,1]$. The derivative can grow exponentially with the degree of the polynomial and the above procedure cannot be used to estimate the accuracy of the roots. Only one class of test polynomials, namely random polynomials, had several roots outside the interval and we did not check their accuracy. The HQR routine we used did not query the machine constants to optimize accuracy. Implementations of the HQR algorithm, which do this, gave however only a precision which was better by a factor of 2 in the cases we checked.

• *Speed.* The speed was measured by introducing CPU timer calls into the test program. In the case of the NAG and IMSL libraries we made calls to the libraries that were installed on these machines. In the case of the Eispack routine, the Fortran routine was included in the program. Thus exactly the same code was run on all machines. On most machines, implementations of the Eispack routines that are optimized for the architecture are available. Their use would somewhat reduce the timings of the HQR algorithm.

The (double precision) programs we used were the IMSL rootfinder ZPORC [3], which is based on the Jenkins–Traub method [1], [4]; the NAG rootfinder C02AGF [5], which is based on Laguerre method [6]; and the subroutine HQR from the Eispack library [7], which is an implementation of the QR algorithm. The IMSL rootfinder is limited to polynomials of degree less than 100. The test were performed on an IBM RS/6000-550, a Cray2, and a VAX 4000-300. Several classes of polynomials with different properties were tested.

3. Test polynomials.

3.1. Fibocacci polynomials.

The name Fibocacci polynomials is not standard terminology, but it is useful to have a name to refer to this class of polynomials. Their coefficients are $a_0 = a_1 = \cdots = a_{n-1} = 1$, $a_n = -1$. The Fibocacci polynomials have n (mostly complex) roots of modulus less than one and one root $\lambda_{\max}$ with modulus greater than one and that has the asymptotic value $\lambda_{\max} = 2 - 2^{-n}$. This distribution of the roots makes them extremely useful for test purposes. The accuracy of the roots in the interval $[-1,1]$ can be assessed by the method discussed above. The values of the Fibocacci polynomial for arguments of modulus less than one can be calculated with nearly machine precision by recursion. In the case of the Fibocacci polynomials, it is however also possible to check the accuracy of the largest root. Since the largest root $\lambda_{\max}$ is well separated from the other roots, it can be calculated with nearly machine precision through the Bernoulli recursion formula. The detailed result are shown in Tables 1 and 2. We separately report the error for the largest root and the largest absolute value $\max(|p(\lambda_i|)$ that the polynomials take on when evaluated at all the other numerical roots. If an entry in the table reads "fail," overflow occurred. "False" means that the errors were so large that they could not be assigned to different roots. For low-order polynomials the HQR method is faster than both the NAG and IMSL library. For high-order polynomials the NAG library is fastest,the HQR retains the second place, and the IMSL library is slowest on the RS/6000 workstations. On the vector computer Cray2, the HQR is always the fastest. For high-order polynomials only the HQR algorithm gives accurate results.

TABLE 1
Fibocacci polynomials on IBM RS/6000-550.

n	5	10	15	20	30	50	100	150
Time in $m$sec NAG	3.2	7.5	11	15.9	26.5	54.5	120	fail
Time in $m$sec IMSL	2.4	12.8	25.3	88.3	377	1160	3860	fail
Time in $m$sec EISPACK	.40	2.2	4.2	8.9	22.5	72.5	410	1180
$\max(\lvert p(\lambda_i)\rvert)$ NAG	$5\ 10^{-16}$	$3\ 10^{-14}$	$2\ 10^{-14}$	$8\ 10^{-15}$	$5\ 10^{-14}$	$1\ 10^{-12}$	false	fail
$\max(\lvert p(\lambda_i)\rvert)$ IMSL	$3\ 10^{-16}$	$6\ 10^{-15}$	$3\ 10^{-13}$	$1\ 10^{-13}$	$7\ 10^{-12}$	$5\ 10^{-6}$	$8\ 10^{-2}$	fail
$\max(\lvert p(\lambda_i)\rvert)$ EISPACK	$2\ 10^{-15}$	$7\ 10^{-15}$	$1\ 10^{-14}$	$6\ 10^{-14}$	$2\ 10^{-13}$	$2\ 10^{-12}$	$3\ 10^{-12}$	$1\ 10^{-11}$
Error in $\lambda_{\max}$ NAG	$1\ 10^{-17}$	$4\ 10^{-16}$	$4\ 10^{-16}$	$4\ 10^{-16}$	$4\ 10^{-16}$	$7\ 10^{-16}$	false	fail
Error in $\lambda_{\max}$ IMSL	$1\ 10^{-17}$	$2\ 10^{-16}$	$2\ 10^{-16}$	$2\ 10^{-14}$	$7\ 10^{-16}$	$1\ 10^{-17}$	$7\ 10^{-15}$	fail
Error in $\lambda_{\max}$ EISPACK	$1\ 10^{-15}$	$9\ 10^{-16}$	$2\ 10^{-16}$	$7\ 10^{-16}$	$2\ 10^{-15}$	$4\ 10^{-16}$	$1\ 10^{-15}$	$6\ 10^{-15}$

TABLE 2
Fibocacci polynomials on Cray2.

n	5	10	15	20	30	40	50	100
Time in $m$sec NAG	4.7	12	18	25	42	60	fail	fail
Time in $m$sec IMSL	4.3	24	70	98	320	520	940	4560
Time in $m$sec EISPACK	.63	2.5	5.0	8.9	18	36	48	220

3.2. Random polynomials. Random polynomials have roots inside and outside the interval $[-1,1]$. We systematically only checked the accuracy of the roots in this interval. Whenever the root with the largest modulus was well separated we checked it against the Bernoulli recursion. The results were completely analogous to the results we obtained for the Fibocacci polynomials. For low-order polynomials both methods give highly accurate results, but the HQR algorithm is faster. For high-order polynomials the HQR algorithm is somewhat slower on serial machines but always gives high precision results, whereas the precision rapidly degrades with the other methods.

3.3. Legendre polynomials. Legendre polynomials have n real roots in the interval $[-1, 1]$. The values of the polynomial in this interval can be calculated with nearly machine precision by the well-known stable recursion relations. The coefficients of the Legendre polynomials can be calculated with machine precision only for polynomials up to degree 24 (double precision). The results are shown in Table 3. Again the HQR algorithm is the winner because of its speed and precision.

3.4. The polynomials $x^n - 1$. This polynomial is particularly difficult for the HQR method since it requires the application of the exceptional shift to the resulting matrix. In this case all roots have modulus 1 and they can be calculated analytically. Therefore the error in the root itself is listed in Table 4. The NAG library is very fast for this example, but again the HQR algorithm gives the best precision for high-order polynomials and is therefore the winner.

3.5. Multiple roots: Powers of the Fibocacci polynomials. Multiple roots are a numerically more difficult than single roots. Since the slope at the root is zero, the root can be

TABLE 3
Legendre polynomials on IBM RS/6000-550.

n	5	10	15	20	24	50	100
Time in $m$sec NAG	3.0	13.0	22.2	35.0	44.3	78.0	148
Time in $m$sec IMSL	1.3	12.6	23.4	35.2	49.1	330	997
Time in $m$sec EISPACK	0.2	1.5	3.4	6.5	10.1	101	336
$\max(\|p(\lambda_i)\|)$ NAG	$1\ 10^{-15}$	$1\ 10^{-14}$	$3\ 10^{-12}$	$3\ 10^{-10}$	$3\ 10^{-9}$		
$\max(\|p(\lambda_i)\|)$ IMSL	$7\ 10^{-15}$	$2\ 10^{-14}$	$8\ 10^{-12}$	$9\ 10^{-11}$	$6\ 10^{-9}$		
$\max(\|p(\lambda_i)\|)$ EISPACK	$7\ 10^{-15}$	$1\ 10^{-13}$	$2\ 10^{-11}$	$3\ 10^{-10}$	$3\ 10^{-9}$		

TABLE 4
$x^n - 1$ *on* IBM RS/6000-550.

n	5	10	20	50	100
Time in $m$sec NAG	0.8	1.8	4.9	23	48
Time in $m$sec IMSL	5.0	13	40	504	3710
Time in $m$sec EISPACK	0.8	2.6	9.8	65	320
$\max(err(\lambda_i))$ NAG	$3\ 10^{-16}$	$3\ 10^{-15}$	$1\ 10^{-14}$	$1\ 10^{-14}$	$7\ 10^{-1}$
$\max(err(\lambda_i))$ IMSL	$2\ 10^{-16}$	$8\ 10^{-15}$	$7\ 10^{-14}$	$2\ 10^{-8}$	$6\ 10^{-1}$
$\max(err(\lambda_i))$ EISPACK	$5\ 10^{-16}$	$1\ 10^{-15}$	$7\ 10^{-15}$	$1\ 10^{-15}$	$2\ 10^{-13}$

located with much less precision. We generated polynomials that had only double or quadruple roots by taking the Fibocacci polynomials to the second or fourth power. The results are shown in Tables 5 and 6. Both the NAG and IMSL rootfinder showed a rather unpredictable behavior. The precision did not uniformly decay for higher degree polynomials, but showed some unexpected outliers. None of the methods gives acceptable results for polynomials of degree higher than 50.

TABLE 5
Double roots on IBM RS/6000-550.

n	4	8	16	24	32	48
Time in $m$sec NAG	1.0	2.7	7.0	12	18	32
Time in $m$sec IMSL	1.1	6.0	45	140	270	680
Time in $m$sec EISPACK	0.2	2.0	8.0	18	33	77
$\max(\|p(\lambda_i\|)$ NAG	$5\ 10^{-15}$	$8\ 10^{-14}$	$1\ 10^{-5}$	$3\ 10^{-8}$	$2\ 10^{-5}$	.4
$\max(\|p(\lambda_i\|)$ IMSL	$2\ 10^{-14}$	$7\ 10^{-14}$	$7\ 10^{-12}$	$7\ 10^{-9}$	$6\ 10^{-8}$	.5
$\max(\|p(\lambda_i\|)$ EISPACK	$3\ 10^{-15}$	$2\ 10^{-14}$	$5\ 10^{-11}$	$2\ 10^{-10}$	$3\ 10^{-6}$	$5\ 10^{-3}$

4. Test results.

Let us summarize the results.

• *Reliability.* For polynomials of high degree overflow occurs frequently in the root finders (especially on VAX), whereas the QR algorithm never gave overflow.

• *Accuracy.* For low-order polynomials, all the subroutines gave highly accurate results. For all high-order polynomials without multiple roots, the rootfinders rapidly lost accuracy, whereas the QR algorithm gave still accurate results. In the case of multiple roots the QR algorithm also gave best overall performance.

• *Speed.* The fact, that the QR algorithm has a complexity of n^3 compared to n^2 for the rootfinders lets one expect that it will be slower for polynomials of high degree. In practice, however, one is usually interested in polynomials of low degree and for these the QR algorithm is faster than the other algorithms. In spite of its higher complexity, the QR algorithm was

TABLE 6
Quadruple roots on IBM RS/6000-550.

n	8	16	24	32	48
Time in $m$sec NAG	2.6	5.8	12	17	27
Time in $m$sec IMSL	100	78	160	325	780
Time in $m$sec EISPACK	1.9	8.1	18	31	78
$\max(\lvert p(\lambda_i)\rvert)$ NAG	$2\ 10^{-11}$	$4\ 10^{-8}$	$2\ 10^{-7}$	$3\ 10^{-4}$	10
$\max(\lvert p(\lambda_i)\rvert)$ IMSL	$2\ 10^{-7}$	$1\ 10^{-14}$	$3\ 10^{-8}$	$7\ 10^{-9}$	20
$\max(\lvert p(\lambda_i)\rvert)$ EISPACK	$2\ 10^{-14}$	$2\ 10^{-12}$	$4\ 10^{-8}$	$1\ 10^{-6}$	$2\ 10^{-1}$

not significantly slower than the rootfinders on the IBM RS/6000. On a vector machine like a Cray2, a crossover of the required time due to the cubically increasing number of operations in the QR algorithm cannot be seen since the increase of the length of the vectorized inner loops results in an higher speed. On a vector machine the QR algorithm is therefore significantly faster for any reasonable value of n. The IMSL routine is under all circumstances by far the slowest.

5. Conclusions. It is surprising to realize that the standard libraries for such an old and presumably rather easy problem, namely the determination of polynomial roots, are not optimal both from the point of accuracy and efficiency. A different approach based on the QR algorithm is preferred. The use of the QR algorithm can be recommended without restriction for polynomials of any degree. For low-order polynomials it is faster than rootfinders and for high-order polynomials it gives the correct result without overflow and in most cases with higher precision. If roots of high multiplicity exist, the HQR method as well as any other method have to be used with caution.

Acknowledgement. I thank Dr. K. Maschke for the careful reading of the manuscript.

Note added in proof. The Lapack library, which replaces the Eispack library, is typically two times faster for matrix eigenvalue problems on a workstation. Unfortunately, the Lapack library was not yet available when the tests were done.

REFERENCES

[1] M. A. JENKINS AND J. F. TRAUB, Numer. Math., 14 (1970), p. 252; SIAM J. Numer. Anal., 7 (1970), p. 545.
[2] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.
[3] *IMSL User's manual*, version 1.0, (1987), chapter 7.
[4] M. A. JENKINS, ACM Trans. Math. Software, 1 (1975), p. 178.
[5] *NAG Fortran Library Manual*, Mark 13 (1988), vol. 1.
[6] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes*, Cambridge University Press, Cambridge, 1989.
[7] *Eispack Guide*, Springer Lecture Notes in Computer Science, Springer, Berlin, 1974.