NORTH-HOLLAND

Polynomial Roots: The Ultimate Answer?

L. Brugnano and D. Trigiante*
Dipartimento di Energetica
Viale C. Lombroso 6 / 17
50134 Firenze, Italy

Submitted by Peter Lancaster

ABSTRACT

We show that it is always possible to transform the problem of finding the roots of a generic polynomial to the problem of determining the eigenvalues of tridiagonal matrices having only simple eigenvalues. Since this problem is very efficiently solved, for example with the familiar QR method, it seems that the present very simple approach has the potentiality to supplant the existing ones.

1. INTRODUCTION

The problem of finding simultaneously all the roots of a polynomial has received an increasing amount of attention in the last few decades, perhaps because of an increasing need of robust methods in the applications. The problem, however, is attractive in itself because of its wonderful history, which goes back to Sumerians ($\approx$ 3000 B.C.) [10]. At the present, the situation is the following. For the special subset of orthogonal polynomials, it has recently been realized that the most efficient method to approximate their roots is to solve an eigenvalue problem for an associate tridiagonal matrix. In the general case the following three main ideas are driving the current lines of research.

*E-mail: mbxtrigiante@mail.cnuce.cnr.it

LINEAR ALGEBRA AND ITS APPLICATIONS 225:207–219 (1995)

655 Avenue of the Americas, New York, NY 10010

0024-3795/95/$9.50
SSDI 0024-3795(93)00341-V

1. *Homotopy methods.* These reach the roots of the polynomial essentially by following a curve implicitly or explicitly defined by a differential equation. They are interesting because they can easily be generalized to systems of polynomials and for their potentiality for parallel computation. They are crucially dependent on the initial points and on the multiplicity of the roots. For a single polynomial they don't seem competitive with the methods belonging to the following two classes.

2. *Methods which transform the polynomial equation to an equivalent system of equations.* This is generally obtained by using some relations among symmetric functions of the roots and the coefficients. The obtained system is then solved by using the Newton method. Examples of such methods are the Durand-Kerner-Weierstrass method [2, 7, 8] and the Pasquini-Trigiante method [12]. Because of the use of the Newton method, such methods are potentially dependent on the choice of the starting point and on the multiplicities of the roots, unless information about these quantities is provided. This information may actually be obtained as the iteration proceeds by means of semiempirical techniques; the dependence on the starting point may also be essentially eliminated, but then the overall procedure becomes cumbersome (see e.g. LPT [9] and BTP [1]). Even if these methods work very well, the proof of their convergence has been done only for special cases, e.g. when the roots are all real and simple.

3. *Methods which transform the problem to an eigenvalue problem.* The easiest way of doing this is to construct the companion matrix associated to the given polynomial and then look for its eigenvalues, for example by taking advantage of the high performance of the QR method. Although this procedure may seem rough, it always gives an approximation of the roots (see the function `roots` in Matlab). However, in the case of multiple roots, the obtainable approximations are not sharp. For special subsets of polynomials, for example orthogonal polynomials, this idea may be refined by using the fact that these polynomials satisfy a three-term recursive relation, and this permits the problem to be transformed into an eigenvalue problem for a symmetric tridiagonal matrix with simple eigenvalues. The performance of the QR method on such matrices is excellent, and today there is unanimity in considering this technique the best.

In this paper we will show that the third idea may be generalized. Mainly, we will show that for a generic real or complex polynomial $p(x)$ the problem of finding its roots may always be transformed to one or more eigenvalue problems for tridiagonal matrices with only simple eigenvalues.

The procedure which transforms the original problem is very simple and gives rise to a very efficient method whose main features are:

1. It is possible to get exact information about the multiplicities.
2. One obtains the same precision for both simple and multiple roots.

3. There is no need of starting points.

4. The order of convergence is that of the QR method for tridiagonal matrices and for simple eigenvalues. In the case of all real roots, the tridiagonal matrix can be taken symmetric, and this further improves the rate of convergence of the QR method [4].

2. THE EUCLIDEAN ALGORITHM

The Euclidean algorithm is usually used to find the common divisor between two polynomials, but it can also be regarded as a three-term recurrence relation similar to that satisfied by orthogonal polynomials. Let

$$p_0(x) = \sum_{i=0}^{n} c_i x^{n-i}, \qquad c_0 = 1,$$

be the polynomial we are interested in. Let $p_1(x)$ be any other polynomial of degree $n-1$. The Euclidean algorithm performs iteratively the division between successive polynomials according to the following scheme:

$$\begin{aligned} p_0(x) &= q_1(x)p_1(x) - p_2(x), \\ p_1(x) &= q_2(x)p_2(x) - p_3(x), \\ &\vdots \\ p_{m-2}(x) &= q_{m-1}(x)p_{m-1}(x) - p_m(x), \\ p_{m-1}(x) &= q_m(x)p_m(x), \end{aligned} \tag{1}$$

where $m \leqslant n$ and $0 \leqslant \deg p_{j+1} < \deg p_j$, $j = 1, \ldots, m-1$. One easily realizes that $p_m(x)$ is a common factor of $p_0(x), p_1(x), \ldots, p_{m-1}(x)$. It is, indeed, the gcd of p_0 and p_1. Then, if $p_m(x) \not\equiv \text{const}$, the functions

$$f_i(x) = \frac{p_i(x)}{p_m(x)}, \qquad i = 0, \ldots, m,$$

are polynomials as well. Since the degree of $p_1(x)$ has been taken equal to $n-1$, the degree of $q_1(x)$ has to be one. Concerning the degree of $p_2(x)$, what can be said is that it is not greater than $n-2$. In general, when it happens that

$$\deg p_i(x) = n - i, \qquad i = 0, \ldots, m \equiv n, \tag{2}$$

we shall say that the algorithm (1) *terminates regularly*. When it does not, i.e. when the gcd of p_0 and p_1 is nontrivial, or when $k \in \{1, \ldots, m\}$ exists such that

$$\begin{aligned} &\deg p_i(x) = n - i, \qquad i = 0, \ldots, k-1, \\ &\deg p_i(x) < n - i, \qquad i \geqslant k, \end{aligned} \tag{3}$$

we say that a *breakdown* has occurred at the kth step of (1). We use this term because this situation seems analogous to what happens when one tries to transform a general square matrix to tridiagonal form [3, 11, 13].

If we are not forced to use a specific $p_1(x)$, it is always possible to choose it such that the algorithm (1) terminates regularly: in fact, it is enough to consider that the conditions on the coefficients which lower the degrees of the polynomials $p_i(x)$, $i \geqslant k$, are necessarily a finite number, while the coefficients of $p_1(x)$ can be chosen in ∞^{n-1} possible ways.

3. THE NEW METHOD

When (2) is satisfied, the relation (1) defines a complete three-term recurrence relation among the polynomials $p_i(x)$, $i = 0, \ldots, n$. It follows that all the polynomials $q_i(x)$ are linear. By setting $q_i(x) = x - \alpha_i$ and supposing that $p_0(x)$ and $p_1(x)$ are monic polynomials, one obtains

$$\begin{aligned} p_0(x) &= (x - \alpha_1) p_1(x) - \beta_1 p_2(x), \\ p_1(x) &= (x - \alpha_2) p_2(x) - \beta_2 p_3(x), \\ &\vdots \\ p_{n-1}(x) &= (x - \alpha_n) p_n(x), \end{aligned} \tag{4}$$

where $p_n(x) \equiv 1$. The quantities β_i have been introduced in order to have the successive polynomials $p_i(x)$ monic. Then, if (1) terminates regularly, p_0 and p_1 are coprime, and the polynomials generated by (1), when normalized by the leading coefficient, coincide with those obtained from (4). On the other hand, if the procedure (1) has a breakdown at the kth step, it means that the procedure (4) is able to produce only the polynomials $p_0, \ldots, p_k$. These polynomials coincide again with the first $k + 1$ ones obtained from (1) scaled by the leading coefficient. It follows that we can switch, if necessary,

from the procedure (4) to the procedure (1). Moreover, we can extend the definitions of "regular termination" and "breakdown" to the procedure (4).

The relations (4) can be placed in matrix form by introducing the matrices

$$T_i = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{i-1} \\ & & 1 & \alpha_i \end{pmatrix}, \qquad i = 1, \ldots, n. \tag{5}$$

One obtains

$$x \begin{pmatrix} p_1(x) \\ p_2(x) \\ \vdots \\ p_n(x) \end{pmatrix} = T_n \begin{pmatrix} p_1(x) \\ p_2(x) \\ \vdots \\ p_n(x) \end{pmatrix} + \begin{pmatrix} p_0(x) \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

from which it follows that the roots of $p_0(x)$ are eigenvalues of T_n and vice versa.

The above results show that one may always transform the original problem to a problem of finding the eigenvalues of a tridiagonal matrix. This eigenvalue problem, however, may have multiple eigenvalues, and that would degrade the QR method. To avoid multiple eigenvalues we shall start with a specific choice for $p_1(x)$, that is

$$p_1(x) = \frac{p_0'(x)}{n}. \tag{6}$$

This choice, even if it does not guarantee that (2) is verified, has the advantage of giving valuable information as the process goes on. In fact, the following general result applies to this case (see e.g. [5]):

THEOREM 1. *If the procedure* (1) *with* (6) *is used, then if* x^* *is a root of multiplicity* k *for* $p_0(x)$, *it is a root of multiplicity* $k - 1$ *for* $p_m(x)$, *and vice versa.*

As a consequence, we have the following

COROLLARY 1. *With the hypotheses of Theorem* 1, *it follows that the roots of* $f_0(x) = p_0(x)/p_m(x)$ *are all simple.*

If the process (1) with (6) terminates regularly, that means that $p_0(x)$ has no multiple roots, since $p_n(x)$ is of degree zero, and then the starting polynomial and its derivative have no common roots. It follows that also the process (4) terminates regularly; moreover, the matrix T_n [see (5)] will have only simple eigenvalues.

Conversely, if a breakdown occurs at the $(r+1)$st step in the procedure (4), then two cases are possible:

$$p_{r+1}(x) \equiv 0,$$

$$p_{r+1}(x) \not\equiv 0.$$

In the first case, it follows that the roots of $p_0(x)$ are given by the roots of $f_0(x) = p_0(x)/p_r(x)$, which has only simple roots, and those of $p_r(x)$. Moreover, the roots of $f_0(x)$ are the eigenvalues of the matrix T_r [see (5)] already formed. It follows that it is sufficient to apply the same procedure to the polynomial $p_r(x)$ alone.

In the second case, we switch from the procedure (4) to the procedure (1). Two subcases may then occur:

$$\deg p_m = 0,$$

$$\deg p_m > 0.$$

In the first subcase p_m is necessarily a constant. This implies that $p_0(x)$ and $p_0'(x)$ have no common factors, and then all the roots of $p_0(x)$ need to be simple. In this case we are no longer interested in (6). As a matter of fact, as stated in the previous section, it is possible to find infinitely many polynomials $p_1(x)$ such that the procedure (4) applied to $p_0(x)$ terminates regularly. In general, it is sufficient to choose $p_1(x)$ as a random monic polynomial of degree $n-1$.

In the second subcase one applies the whole procedure to the two polynomials $p_m(x)$ and $f_0(x) = p_0(x)/p_m(x)$ as starting polynomials. As before, $f_0(x)$ has only simple roots.

The previous considerations may be summarized as follows.

THEOREM 2 (Main result). *The roots of a real or complex polynomial $p(x)$ of degree n are the eigenvalues of a block diagonal matrix*

$$T^* = \begin{pmatrix} T^{(1)} & & \\ & \ddots & \\ & & T^{(s)} \end{pmatrix}. \tag{7}$$

Each block $T^{(j)}$ has only simple eigenvalues, and has the form

$$T^{(j)} = \begin{pmatrix} \alpha_1^{(j)} & \beta_1^{(j)} & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k_j-1}^{(j)} \\ & & 1 & \alpha_{k_j}^{(j)} \end{pmatrix}, \qquad j = 1, \ldots, s,$$

where

$$d = k_1 \geqslant k_2 \geqslant \cdots \geqslant k_s \geqslant 1$$

and

$$\sum_{j=1}^{s} k_j = n.$$

The number of diagonal blocks s is equal to the maximum multiplicity of the roots of $p(x)$, while d is the number of distinct roots. Moreover, if a root appears in the jth block and not in the $(j+1)$st one, it has exact multiplicity j and must appear in all the previous blocks.

The method based on the previous main result has the potential to furnish the exact multiplicities of the roots. This is never achieved by usual methods, although some methods (for example IPT [9]) may give this information with good reliability.

It is known that for the existing methods, in the case where the roots of $p_0(x)$ are all real and simple, the problem simplifies. This is essentially due to the fact that in this case Rolle's theorem applies in the usual form. The counterpart for our procedure is the following:

THEOREM 3. *Suppose that $p_0(x)$ is a monic polynomial. Then it has only real and simple roots iff the procedure* (4) *with* (6) *terminates regularly and $\beta_i > 0$, $i = 1, \ldots, n-1$.*

Let us prove the first implication. Although the proof could be done by observing that in this case the sequence $\{p_i\}$ is a Sturm sequence, we prefer to prove the result in a different way which shows the meaning of the quantities β_i. From Rolle's theorem one has that the roots of $p_1(x)$ separate those of $p_0(x)$. Let $x_1 < x_2 < \cdots < x_{n-1}$ be the roots of $p_1(x)$. Then the quantities

$$g_i = \frac{p_0(x_i)}{\prod_{j \neq i}(x_i - x_j)}, \qquad i = 1, \ldots, n-1,$$

have constant sign, that is,

$$\begin{aligned}\operatorname{sign} g_i &= \operatorname{sign} p_0(x_i) \operatorname{sign} \prod_{j\neq i}(x_i - x_j)\\ &= \operatorname{sign} p_0(x_1) \operatorname{sign} \prod_{j=2}^{n-1}(x_1 - x_j)\\ &= (-1)^{n-1}(-1)^{n-2}\\ &= -1,\end{aligned}$$

where $\operatorname{sign} p_0(x_1) = (-1)^{n-1}$ follows from the hypothesis that $p_0(x)$ is monic. By considering the divided difference of $p_0(x)$ over $x_1, \ldots, x_{n-1}$, from the first relation in (4) one has

$$p_0[x_1, \ldots, x_{n-1}] = -\beta_1 = \sum_{i=1}^{n-1} g_i < 0,$$

that is, $\beta_1 > 0$. Moreover one has

$$\frac{p_0(x_i)}{p_2(x_i)} = -\beta_1 < 0, \qquad i = 1, \ldots, n-1.$$

Therefore $p_2(x)$ has $n-2$ real roots, which separate those of $p_1(x)$. The proof is completed by induction. The next β_i, $i = 2, \ldots, n-1$, are

$$\beta_i = -p_{i-1}[z_1, \ldots, z_{n-i}],$$

where $z_1 < \cdots < z_{n-i}$ are the roots of $p_i(x)$.

Concerning the converse implication, since all the β_i are positive, the matrix T_n [see (5)] may be transformed to the symmetric tridiagonal matrix

$$\hat{T}_n = \begin{pmatrix} \alpha_1 & \gamma_1 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{n-1} \\ & & \gamma_{n-1} & \alpha_n \end{pmatrix},$$

where

$$\gamma_i = \sqrt{\beta_i}, \qquad i = 1, \ldots, n-1.$$

Since the matrix $\hat{T}_n$ is symmetric and unreduced, its eigenvalues have to be real and distinct. ■

4. NUMERICAL EXAMPLES

In the previous sections we have described a procedure which is able to find all the roots of a polynomial along with the respective multiplicities. The procedure consists in the construction of the block diagonal matrix T^* in (7), whose blocks are tridiagonal with only simple eigenvalues.

Owing to the logical simplicity of the procedure, the tridiagonal matrix T^* may be constructed either numerically or formally. Of course, the numerical

TABLE 1

btroots	μ
1.000000000000000E00	6
−1.000000000000000E00	2
1.110223024625157E − 16 + 1.000000000000000E00*i*	3
1.110223024625157E − 16 − 1.000000000000000E00*i*	3
2.000000000000000E00	1

roots
1.999999999999988E00
−9.999999999999982E − 01 + 1.741703483489851E − 08*i*
−9.999999999999982E − 01 − 1.741703483489851E − 08*i*
2.784917664012954E − 06 + 1.000004549925426E00*i*
2.784917664012954E − 06 − 1.000004549925426E00*i*
−5.332793704612704E − 06 + 1.000000136873613E00*i*
−5.332793704612704E − 06 − 1.000000136873613E00*i*
2.547876037664598E − 06 + 9.999953132009639E − 01*i*
2.547876037664598E − 06 − 9.999953132009639E − 01*i*
1.001555590371782E00 + 2.702808664651358E − 03*i*
1.001555590371782E00 − 2.702808664651358E − 03*i*
9.984395503134930E − 01 + 2.694392347437423E − 03*i*
9.984395503134930E − 01 − 2.694392347437423E − 03*i*
1.003120952472778E00
9.968887661566825E − 01

implementation may have difficulties, in pathological cases, in recognizing whether a breakdown has occurred in the procedure (4) or not. This problem disappears if a formal implementation for constructing T^* is used. However, once the diagonal blocks of the matrix T^* are obtained, no more formal operations are needed, since the QR method works quite well.

As usual, in finite-precision arithmetic one must distinguish the following two cases:

1. the coefficients of the polynomial are exactly represented;
2. the coefficients of the polynomial are not exactly represented.

In the latter case, the represented polynomial may have different multiplicities from the exact one. A typical example is given by the polynomial x^n if represented as $x^n + \varepsilon$, where a multiple root splits into n simple roots. In this case the method will fail in the determination of the multiplicities.

In the first case, troubles are expected in some pathological cases, even though some of them can be overcome. For example, the polynomial $(x + \varepsilon)(x + 2\varepsilon)$ can be transformed to the more favorable $(y + 1)(y + 2)$ with the change of variable $x = \varepsilon y$, but we do not go into details of the effective implementation of the method.

TABLE 2

btroots	μ
1.000000000000000E00	10
2.000000000000002E00	2
2.359223927328458E − 16 + 1.000000000000000E00i	1
2.359223927328458E − 16 − 1.000000000000000E00i	1
roots	
2.000000000139061E00 + 6.670978040752351E − 06i	
2.000000000139061E00 − 6.670978040752351E − 06i	
4.996003610813204E − 15 + 9.999999999999956E − 01i	
4.996003610813204E − 15 − 9.999999999999956E − 01i	
1.057137071108160E00 + 1.961984122875980E − 02i	
1.057137071108160E00 − 1.961984122875980E − 02i	
1.032737676591875E00 + 4.933406972009342E − 02i	
1.032737676591875E00 − 4.933406972009342E − 02i	
9.970809692262279E − 01 + 5.734748579346576E − 02i	
9.970809692262279E − 01 − 5.734748579346576E − 02i	
9.654271879090582E − 01 + 4.377197476036213E − 02i	
9.654271879090582E − 01 − 4.377197476036213E − 02i	
9.476170950256143E − 01 + 1.614499615703074E − 02i	
9.476170950256143E − 01 − 1.614499615703074E − 02i	

We shall report the results obtained from a preliminary version of the algorithm here described, which has been coded in Matlab and called `btroots`. The obtained results are compared with the output of the Matlab function `roots`, which uses the QR method applied to the companion matrix. We mention that other known solvers such as the Jenkins-Traub [6] and the Durand-Kerner method are much more entangled, and they do not furnish any information about multiplicities.

In all the examples, the Matlab function `poly` has been used to obtain the coefficients of the polynomial. The outputs of the functions `btroots` and `roots` are reported (for `btroots`, the second column in each table contains the found multiplicity μ):

TABLE 3

`btroots`	μ
0 + 1.000000000000051E00i	5
0 − 1.000000000000051E00i	5
0 + 5.000000000002114E − 01i	4
0 − 5.000000000002114E − 01i	4
0 + 7.500000000017798E − 01i	1
0 − 7.500000000017798E − 01i	1

`roots`
− 8.677727078576503E − 10 + 1.001364240450085E00i
− 8.677727078576503E − 10 − 1.001364240450085E00i
− 1.297904655584797E − 03 + 1.000422932206438E00i
− 1.297904655584797E − 03 − 1.000422932206438E00i
1.297903969489856E − 03 + 1.000422933904355E00i
1.297903969489856E − 03 − 1.000422933904355E00i
− 8.037626826402702E − 04 + 9.988949461055121E − 01i
− 8.037626826402702E − 04 − 9.988949461055121E − 01i
8.037642364997766E − 04 + 9.988949473339882E − 01i
8.037642364997766E − 04 − 9.988949473339882E − 01i
3.608224830031759E − 16 + 7.500000000007786E − 01i
3.608224830031759E − 16 − 7.500000000007786E − 01i
− 1.464917736879456E − 04 + 5.001463361006820E − 01i
− 1.464917736879456E − 04 − 5.001463361006820E − 01i
1.464917536932175E − 04 + 5.001463361205132E − 01i
1.464917536932175E − 04 − 5.001463361205132E − 01i
− 1.461809791177826E − 04 + 4.998536638787507E − 01i
− 1.461809791177826E − 04 − 4.998536638787507E − 01i
1.461809991131213E − 04 + 4.998536638988948E − 01i
1.461809991131213E − 04 − 4.998536638988948E − 01i

EXAMPLE 1.

$$p(x) = (x-1)^6(x+1)^2(x+i)^3(x-i)^3(x-2).$$

The results are shown in Table 1.

EXAMPLE 2.

$$p(x) = (x-1)^{10}(x-2)^2(x-i)(x+i).$$

The results are shown in Table 2.

EXAMPLE 3.

$$p(x) = (x-i)^5(x+i)^5(x-0.5i)^4(x+0.5i)^4(x-0.75i)(x+0.75i).$$

The results are shown in Table 3.

TABLE 4

btroots	μ
1.000000000000000E00	3
−1.000000000000000E00	4
4.999999999999961E − 01 + 1.000000000000003E00i	3
4.999999999999961E − 01 − 1.000000000000003E00i	3
4.999999999999963E − 01 + 5.000000000000068E − 01i	2
4.999999999999963E − 01 − 5.000000000000068E − 01i	2
roots	
− 1.000081182476063E00	
−1.000000000770496E00 + 8.118324186043912E − 05i	
−1.000000000770496E00 − 8.118324186043912E − 05i	
−9.999188159829470E − 01	
4.999958489694322E − 01 + 1.000012649037798E00i	
4.999958489694322E − 01 − 1.000012649037798E00i	
5.000130304823061E − 01 + 9.999972704487297E − 01i	
5.000130304823061E − 01 − 9.999972704487297E − 01i	
4.999911205482621E − 01 + 9.999900805135149E − 01i	
4.999911205482621E − 01 − 9.999900805135149E − 01i	
1.000004941876991E00 + 8.559855154400441E − 06i	
1.000004941876991E00 − 8.559855154400441E − 06i	
9.999901162460169E − 01	
5.000000692175076E − 01 + 4.999999709731102E − 01i	
5.000000692175076E − 01 − 4.999999709731102E − 01i	
4.999999307824918E − 01 + 5.000000290268498E − 01i	
4.999999307824918E − 01 − 5.000000290268498E − 01i	

EXAMPLE 4.

$$p(x) = (x - 1)^3(x + 1)^4(x - 0.5 - i)^3(x - 0.5 + i)^3$$
$$\times(x - 0.5 - 0.5i)^2(x - 0.5 + 0.5i)^2.$$

The results are shown in Table 4.

Note added in proof. After the manuscript was accepted, the authors were advised that a result similar to Theorem 3 had been obtained by Schmeisser and published in this *Journal* 193:14 (1993).

REFERENCES

1 L. Brugnano, BTP: Un Algoritmo per l'Approssimazione Simultanea delle Radici di un Polinomio, Report 6/88 IRMA-CNR, Dept. of Mathematics, Univ. of Bari, 70125 Bari, Italy, 1988.

2 E. Durand, *Solution Numérique des Equations Algébriques*, Vol. I, Masson, Paris, 1968.

3 G. A. Geist, Reduction of a general matrix to tridiagonal form, *SIAM J. Matrix Anal. Appl.* 12:362–373 (1991).

4 G. H. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins U.P., Baltimore, 1989.

5 P. Henrici, *Applied and Computational Complex Analysis*, Vol. 1, Wiley, New York, 1974.

6 M. A. Jenkins and J. F. Traub, A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration, *Numer. Math.* 14:252–263 (1970).

7 I. O. Kerner, Ein Gesamtschittverfahren zur Berechnund der Nullstellen von Polynomen, *Numer. Math.* 8:290–294 (1966).

8 I. O. Kerner, Algorithm 283, *Comm. ACM* 9:273 (1966).

9 M. Lo Cascio, L. Pasquini, and D. Trigiante, Simultaneous determination of polynomial roots and multiplicities: An algorithm and related problems, *Ricerche Mat.* 38:283–305 (1989).

10 O. Neugebauer, *The Exact Sciences in Antiquity*, 2nd ed., Brown U.P., Providence, 1957.

11 B. N. Parlett, Reduction to Tridiagonal Form and Minimal Realizations, Report PAM-486, Center for Pure and Applied Mathematics, Univ. of California, Berkeley, 1990.

12 L. Pasquini and D. Trigiante, A globally convergent method for simultaneously finding polynomial roots, *Math. Comp.* 44:135–149 (1985).

13 J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford U.P., Oxford, 1965.

Received 12 April 1993; final manuscript accepted 28 November 1993