Numer. Math. 14, 252—263 (1970)

A Three-Stage Variable-Shift Iteration for Polynomial Zeros and Its Relation to Generalized Rayleigh Iteration*

M. A. JENKINS and J. F. TRAUB

Received July 11, 1968

Summary. We introduce a new three-stage process for calculating the zeros of a polynomial with complex coefficients. The algorithm is similar in spirit to the two-stage algorithms studied by Traub in a series of papers. We prove that the mathematical algorithm always converges and show that the rate of convergence of the third stage is faster than second order. To obtain additional insight we recast the problem and algorithm into matrix form. The third stage is inverse iteration with the companion matrix, followed by generalized Rayleigh iteration.

1. Introduction

We introduce a three-stage algorithm for calculating the zeros of a polynomial P,

$$P(z)=\sum_{i=0}^{n} a_i z^{n-i}, \qquad a_0=1,\ a_n\neq 0, \tag{1.1}$$

$$P(z)=\prod_{i=1}^{j}(z-\zeta_i)^{m_i}.$$

The condition $a_0=1$ is for convenience only. The coefficients are in general complex. The algorithm involves iteration in the complex plane. In [4] we analyze the appropriate analogue for polynomials with real coefficients (and complex conjugate zeros) which uses only real arithmetic.

The zeros are calculated one at a time and zeros of multiplicity m are found m times. The zeros are found in roughly increasing order of magnitude to avoid the instability arising from deflation with a large zero (Wilkinson [12]). (However this ordering of the deflation is not sufficient to ensure deflation stability for all polynomials. A discussion may be found in [4].)

The algorithm is similar in spirit to the two-stage algorithms proposed by Traub [8—10]. In [8] Traub gives a class of always convergent algorithms for calculating the largest zero. The appropriate modification of this algorithm for the case of a pair of complex conjugate zeros was announced in [9]. In [10] Traub gives an algorithm for calculating the smallest zero.

The implementation by Jenkins and Traub of a general polynomial solver based on two-stage algorithms is described in [5]. Separate procedures are used depending on whether there are one or two smallest zeros. If there are more than

* This work was supported in part by the National Science Foundation and the Office of Naval Research.

two distinct smallest zeros, a process of "double translation" described in Section 6 of [5] is used to break up the equimodularity.

The two-stage algorithm implemented in [5] performed very well and was fast except for polynomials with many nearly equimodular zeros. In this paper shifting is introduced to remedy this. Furthermore the methods for deciding when to terminate stages has been simplified from those presented in [5].

The three-stage algorithm introduced in this paper has the following desirable characteristics.

1. The mathematical algorithm is restriction-free that is, it converges for any distribution of zeros.

2. Zeros are calculated in roughly increasing order of modulus; this avoids the instability which occurs when the polynomial is deflated with a large zero.

3. The final stage is an iterative process and thus has the desirable stability features of iterative processes.

4. Few critical decisions have to be made by the program which implements the algorithm.

5. The algorithm is fast for all distributions of zeros.

6. Shifting is incorporated in the algorithm itself in a natural and stable way. Shifting breaks equimodularity and speeds convergence.

We summarize the contents of this paper. The mathematical algorithm is stated in Section 2. Global convergence for an arbitrary distribution of zeros is proven in Section 3 and the quadratic character of the convergence is established in Section 4.

In Section 5 we recast the problem and algorithm in matrix form and prove that Stage Three may be viewed as an efficient process for carrying out inverse powering using a companion matrix with shifted eigenvalues and generalized Rayleigh iteration. Although we are dealing with the case of a non-Hermitean matrix with nonlinear elementary divisors, the process does not suffer from the customary (Ostrowski [7]) slow convergence.

In Section 6 we prove that the third stage is precisely equivalent to Newton-Raphson iteration applied to a sequence of rational functions converging to a linear polynomial. It is a Newton-Raphson iteration even though *no differentiation* is performed.

Our focus in this paper is on the mathematical algorithm and its properties. Timing information and a numerical example are provided. A description of the implementation, an analysis of the effects of finite-precision arithmetic, an ALGOL program, the results of extensive testing, and a second program which clusters the zeros and provides a posteriori error bounds may be found in Jenkins' thesis [3]. In Section 7 we do discuss a number of important points pertaining to stability and decisions to be made by the implementing program. In the final section we give a small numerical example.

2. The Algorithm

We define the three-stage algorithm. Motivation may be found in [3].

The algorithm is used to calculate a zero of P. After each zero is found, the polynomial is deflated and then the algorithm is applied to the deflated poly-

nomial. Hence P represents either the original polynomial or a polynomial obtained by deflation.

Stage One. No-Shift Process

$$H^{(0)}(z) = P'(z),$$

$$H^{(\lambda+1)}(z) = \frac{1}{z}\left[H^{(\lambda)}(z) - \frac{H^{(\lambda)}(0)}{P(0)} P(z)\right], \quad \lambda = 0, 1, \ldots, M-1. \tag{2.1}$$

Stage Two. Fixed-Shift Process

Take β to be a positive number such that $\beta \leqq \min |\varrho_i|$ and let s be such that $|s| = \beta$ and such that

$$|s - \varrho_1| < |s - \varrho_i|, \quad i = 2, \ldots, j. \tag{2.2}$$

(The root labeled ϱ_1 depends on the choice of s.)

Let

$$H^{(\lambda+1)}(z) = \frac{1}{z-s}\left[H^{(\lambda)}(z) - \frac{H^{(\lambda)}(s)}{P(s)} P(z)\right], \quad \lambda = M, M+1, \ldots, L-1. \tag{2.3}$$

Stage Three. Variable-Shift Process

Take

$$s_L = s - \frac{P(s)}{\bar{H}^{(L)}(s)},$$

and let

$$\begin{aligned} H^{(\lambda+1)}(z) &= \frac{1}{z-s_\lambda}\left[H^{(\lambda)}(z) - \frac{H^{(\lambda)}(s_\lambda)}{P(s_\lambda)} P(z)\right], \\ s_{\lambda+1} &= s_\lambda - \frac{P(s_\lambda)}{\bar{H}^{(\lambda+1)}(s_\lambda)}, \quad \lambda = L, L+1, \ldots, \end{aligned} \tag{2.4}$$

where $\bar{H}^{(\lambda)}(z)$ is $H^{(\lambda)}(z)$ divided by its leading coefficient.

There are a number of iterative processes used in the algorithm. In each of the three stages there is an iteration producing a sequence of polynomials.

The first iteration is a reformulation of the classical Bernoulli iteration and performance of an analogous process on a sequence of polynomials of decreasing degree leads to Bauer's [1] Treppeniteration. The second iteration is the same process with a fixed origin shift. The iteration of Stage Three is related to an iteration suggested by Bauer and Samelson [2] which leads to a form of Treppeniteration with Newton-like convergence.

We prove in the next section that our algorithm, combining these processes, is globally convergent.

3. Proof of Global Convergence

We investigate the convergence of the three-stage algorithm. We begin by giving sufficient conditions for the convergence of the variable-shift iteration defined by (2.4). Let

$$H^{(L)}(z) = \sum_{i=1}^{j} c_i^{(L)} P_i(z), \quad P_i(z) = \frac{P(z)}{z-\varrho_i}. \tag{3.1}$$

Theorem. *Assume*

(i) $|s_L-\varrho_1|<\frac{1}{2}R$, *where* $R=\min_i|\varrho_1-\varrho_i|$,

(ii) $c_1^{(L)}\neq 0$,

(iii) $D_L=\sum_{i=2}^{j}\frac{|c_i^{(L)}|}{|c_1^{(L)}|}>\frac{1}{3}$.

Then $s_\lambda\to\varrho_1$.

Proof. We defer to the end of this proof the demonstration that the iteration is always defined. We show first that if the iteration is defined, it converges. It may be shown from (2.4) that for $\lambda\geqq L$,

$$H^{(\lambda)}(z)=\sum_{i=1}^{j}c_i^{(\lambda)}P_i(z),$$

where

$$c_i^{(\lambda)}=c_i^{(L)}\prod_{t=L}^{\lambda-1}(\varrho_i-s_t)^{-1}.$$

Some algebraic manipulation leads to

$$\frac{s_{\lambda+1}-\varrho_1}{s_\lambda-\varrho_1}=\frac{\sum_{i=2}^{j}[r_i^{(\lambda)}]^2d_i^{(\lambda)}-\sum_{i=2}^{j}r_i^{(\lambda)}d_i^{(\lambda)}}{1+\sum_{i=2}^{j}[r_i^{(\lambda)}]^2d_i^{(\lambda)}},\tag{3.2}$$

where

$$r_i^{(\lambda)}=\frac{s_\lambda-\varrho_1}{s_\lambda-\varrho_i},\quad d_i^{(\lambda)}=\frac{c_i^{(\lambda)}}{c_1^{(\lambda)}}.$$

Let

$$\frac{|s_{\lambda+1}-\varrho_1|}{|s_\lambda-\varrho_1|}=T_\lambda.$$

We prove convergence by showing there exists a τ_L such that for all $\lambda\geqq L$, $T_\lambda\leqq\tau_L<1$. The proof is by induction.

Observe that

$$|r_i^{(L)}|=\frac{|s_L-\varrho_1|}{|s_L-\varrho_i|}<1.$$

Hence

$$T_L\leqq\frac{\sum_{2}^{j}|d_i^{(L)}|+\sum_{2}^{j}|d_i^{(L)}|}{1-\sum_{2}^{j}|d_i^{(L)}|}=\frac{2D_L}{1-D_L}.$$

By hypothesis $D_L<\frac{1}{3}$. Let

$$\tau_L=\frac{2D_L}{1-D_L}.\tag{3.3}$$

Then $T_L\leqq\tau_L<1$.

Assume now that $T_L, T_{L+1},\ldots,T_{\lambda-1}\leqq\tau_L<1$. Hence for $t=L, L+1,\ldots,\lambda$,

$$|s_t-\varrho_1|\leqq|s_L-\varrho_1|<\tfrac{1}{2}R,$$
$$|s_t-\varrho_i|\geqq|\varrho_1-\varrho_i|-|s_t-\varrho_1|>\tfrac{1}{2}R.$$

Thus

$$|r_i^{(t)}|<1, \quad t=L, L+1, \ldots, \lambda. \tag{3.4}$$

Observe that

$$d_i^{(\lambda)}=r_i^{(\lambda-1)}\, d_i^{(\lambda-1)}.$$

Hence

$$\sum_{2}^{j} |d_i^{(\lambda)}| \leqq D_L. \tag{3.5}$$

From (3.2), (3.3), (3.4), (3.5) it follows that $T_\lambda \leqq \tau_L < 1$ for $\lambda \geqq L$ which completes the proof of convergence.

We now show that the sequence $\{s_\lambda\}$ is always well defined for $\lambda \geqq L$.

$$\bar{H}^{(\lambda+1)}(s_\lambda)=\frac{\sum_{i=1}^{j} c_i^{(\lambda)}(\varrho_i-s_\lambda)^{-1} P_i(s_\lambda)}{\sum_{i=1}^{j} c_i^{(\lambda)}(\varrho_1-s_\lambda)^{-1}}$$

$$=P_1(s_\lambda)\left[\frac{1+\sum_{i=2}^{j} d_i^{(\lambda)}[r_i^{(\lambda)}]^2}{1+\sum_{i=2}^{j} d_i^{(\lambda)} r_i^{(\lambda)}}\right].$$

$P_1(s_\lambda) \neq 0$ by hypothesis i and the contraction argument. Since, as we have seen,

$$\left|\sum_{i=2}^{j} d_i^{(\lambda)}[r_i^{(\lambda)}]^2\right|<\tfrac{1}{3},$$

$\bar{H}^{(\lambda+1)}(s_\lambda) \neq 0$ and the iteration is well defined. This completes the proof of the theorem.

We now investigate the convergence of the three-stage algorithm defined in Section 2. The major result of the paper is given by the following

Theorem. *Let s be such that $|s-\varrho_1|<|s-\varrho_i|$, $i=2, \ldots, j$. Then for all L sufficiently large and fixed, $s_\lambda \to \varrho_1$.*

Proof. It may be shown from (2.1) and (2.2) that

$$H^{(L)}(z)=\sum_{i=1}^{j} m_i \varrho_i^{-M}(\varrho_i-s)^{-(L-M)} P_i(z)$$

$$=\sum_{i=1}^{j} c_i^{(L)} P_i(z).$$

We have $c_1^{(L)} \neq 0$. Furthermore

$$\sum_{i=2}^{j} \frac{c_i^{(L)}}{c_1^{(L)}}=\sum_{i=2}^{j} \frac{m_i}{m_1}\left(\frac{\varrho_1}{\varrho_i}\right)^M\left(\frac{\varrho_1-s}{\varrho_i-s}\right)^{L-M}.$$

By hypothesis, $|\varrho_1-s|<|\varrho_i-s|$.

Fix M. Then by choosing L sufficiently large we can make

$$D_L=\sum_{i=2}^{j} \frac{|c_i^{(L)}|}{|c_1^{(L)}|}$$

as small as desired. Choose L so that

$$D_L < \tfrac{1}{3} \tag{3.6}$$

and

$$|s-\varrho_1|\frac{2D_L}{1-D_L} < \frac{1}{2}R. \tag{3.7}$$

The condition of (3.7) ensures that $|s_L-\varrho_1| < \frac{1}{2}R$. All the hypotheses of the first theorem of this section now hold and the conclusion follows.

4. Rate of Convergence

Let

$$C(\lambda) = \frac{|s_{L+\lambda+1}-\varrho_1|}{|s_{L+\lambda}-\varrho_1|^2}. \tag{4.1}$$

In the last section we proved the existence of a number τ_L such that for $\lambda \geqq 0$,

$$\frac{|s_{L+\lambda+1}-\varrho_1|}{|s_{L+\lambda}-\varrho_1|} = T_\lambda \leqq \tau_L < 1, \tag{4.2}$$

where $\tau_L = 2D_L/(1-D_L)$. We defined $R = \min_i |\varrho_1-\varrho_i|$. The rate of convergence of our algorithm is governed by the following

Theorem. *Let the hypotheses of the first theorem of the previous section hold. Then*

$$C(\lambda) \leqq \frac{2}{R}\tau_L^{\lambda(\lambda-1)/2}. \tag{4.3}$$

Proof. From (3.2),

$$\frac{s_{L+\lambda+1}-\varrho_1}{(s_{L+\lambda}-\varrho_1)^2} = \frac{\sum_{i=2}^{j} \frac{r_i^{(L+\lambda)} d_i^{(L+\lambda)}}{s_{L+\lambda}-\varrho_i} - \sum_{i=2}^{j} \frac{d_i^{(L+\lambda)}}{s_{L+\lambda}-\varrho_i}}{1+\sum_{i=2}^{j} [r_i^{(L+\lambda)}]^2 d_i^{(L+\lambda)}}. \tag{4.4}$$

One may verify that for all λ and $i > 1$,

$$|r_i^{(L+\lambda)}| \leqq \tau_L^\lambda, \quad \frac{1}{|s_{\lambda+L}-\varrho_i|} \leqq \frac{2}{R},$$

and

$$\sum_{i=2}^{j} |d_i^{(L+\lambda)}| \leqq \tfrac{1}{3}\tau_L^{\lambda(\lambda-1)/2}.$$

Substituting these bounds into (4.4) establishes the theorem.

Thus the process is second order with an error constant $C(\lambda)$ which approaches zero. This may be contrasted with the conventional Newton-Raphson iteration in which there is no control over the error constant.

Corollary. *Let the hypotheses of the first theorem of the previous section hold. Then for* $\lambda > 0$,

$$|s_{L+\lambda}-\varrho_1| \leqq \tfrac{1}{2}R\tau_L^\eta,$$
$$\eta = \tfrac{1}{2}[3\cdot 2^\lambda - (\lambda^2+\lambda+2)].$$

Proof. For $\lambda = 1$ this follows from (4.2). For $\lambda > 1$ it follows upon substituting (4.3) into (4.1).

5. Variable-Shift Iteration is Generalized Rayleigh Iteration

We now give a matrix interpretation of our algorithm. We show that in a matrix formulation the vector iteration of the third stage is inverse powering with a matrix whose eigenvalues have been shifted, while the scalar iteration is generalized Rayleigh iteration.

Let

$$A = \begin{pmatrix} 0 & 0 & \dots & 0 & -a_n \\ 1 & 0 & \dots & 0 & -a_{n-1} \\ 0 & 1 & \dots & 0 & -a_{n-2} \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ . & . & \dots & . & . \\ . & . & \dots & 1 & -a_1 \end{pmatrix}$$

be the companion matrix of P. Let

$$H^{(\lambda)}(z) = \sum_{i=0}^{n-1} h_i^{(\lambda)} z^{n-1-i}, \qquad [\underline{h}^{(\lambda)}]^T = (h_{n-1}^{(\lambda)}, \dots, h_0^{(\lambda)}).$$

Let

$$P_i(z) = \frac{P(z)}{z - \varrho_i} = \sum_{j=0}^{n-1} p_{ij} z^{n-1-j},$$
$$\underline{p}_i^T = (p_{in-1}, \dots, p_{i0}), \qquad \underline{q}_i^T = (1, \dots, \varrho_i^{n-1}).$$

One may easily verify that for the eigenvalue ϱ_i, the right and left eigenvectors are $\underline{p}_i$ and $\underline{q}_i^T$, respectively.

The fixed-shift recurrence

$$H^{(\lambda+1)}(z) = \frac{1}{z-s}\left[H^{(\lambda)}(z) - \frac{H^{(\lambda)}(s)}{P(s)} P(z)\right] \tag{5.1}$$

is equivalent to

$$\underline{h}^{(\lambda+1)} = (A - sI)^{-1} \underline{h}^{(\lambda)}. \tag{5.2}$$

Let

$$[\underline{s}^{(\lambda)}]^T = (1, s_\lambda, \dots, s_\lambda^{n-1}).$$

We have the following

Theorem. *The variable-shift recurrence*

$$H^{(\lambda+1)}(z) = \frac{1}{z-s_\lambda}\left[H^{(\lambda)}(z) - \frac{H^{(\lambda)}(s_\lambda)}{P(s_\lambda)} P(z)\right], \tag{5.3}$$

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\bar{H}^{(\lambda+1)}(s_\lambda)} \tag{5.4}$$

is equivalent to

$$\underline{h}^{(\lambda+1)} = (A - s_\lambda I)^{-1} \underline{h}^{(\lambda)}, \tag{5.5}$$

$$s_{\lambda+1} = \frac{[\underline{s}^{(\lambda)}]^T A \, \underline{h}^{(\lambda+1)}}{[\underline{s}^{(\lambda)}]^T \underline{h}^{(\lambda+1)}}. \tag{5.6}$$

Proof. Eq. (5.5) follows easily from (5.3). To prove (5.6), note that (5.4) may be written as

$$s_{\lambda+1} = \frac{s_\lambda H^{(\lambda+1)}(s_\lambda) - h_0^{(\lambda+1)} P(s_\lambda)}{H^{(\lambda+1)}(s_\lambda)}. \tag{5.7}$$

Observe that

$$H^{(\lambda+1)}(s_\lambda) = [\underline{s}^{(\lambda)}]^T \underline{h}^{(\lambda+1)}. \tag{5.8}$$

From (5.3),

$$h_0^{(\lambda+1)} = -\frac{H^{(\lambda)}(s_\lambda)}{P(s_\lambda)}.$$

Hence

$$\begin{aligned} s_\lambda H^{(\lambda+1)}(s_\lambda) - h_0^{(\lambda+1)} P(s_\lambda) &= s_\lambda H^{(\lambda+1)}(s_\lambda) + H^{(\lambda)}(s_\lambda) \\ &= [\underline{s}^{(\lambda)}]^T (A \underline{h}^{(\lambda+1)} - \underline{h}^{(\lambda)}) + [\underline{s}^{(\lambda)}]^T \underline{h}^{(\lambda)} \\ &= [\underline{s}^{(\lambda)}]^T A \underline{h}^{(\lambda+1)}. \end{aligned}$$

Substituting this result together with (5.8) into (5.7) completes the proof.

Eq. (5.5) is the formula for inverse iteration (Wielandt [11]). Observe that $\underline{h}^{(\lambda)} \to \underline{p}_1$ and $[\underline{s}^{(\lambda)}]^T \to \underline{q}_1^T$. Hence (5.5) is a generalized Rayleigh iteration (Wilkinson [13, p. 179], Ostrowski [7]) appropriate for non-Hermitean matrices.

However we are in a very favorable position as compared with the usual situation when inverse iteration and generalized Rayleigh iteration are applied.

1. No calculation has to be performed to determine the left eigenvector. It is simply $[\underline{s}^{(\lambda)}]^T$.

2. Multiple eigenvalues of a companion matrix imply nonlinear elementary divisors. In general this leads to (Ostrowski [7]) linear convergence. In our case multiple zeros do not affect the rate of convergence (it is still quadratic) and require no special attention.

3. The inverse iteration is carried out explicitly.

4. The initial vector $\underline{h}^{(0)}$ and all succeeding vectors $\underline{h}^{(\lambda)}$ lie in the subspace spanned by the eigenvectors of A. Furthermore the $\underline{h}^{(\lambda)}$ cannot be deficient in the eigenvector corresponding to the eigenvalue being calculated.

6. Newton-Raphson Iteration

We show that the formula

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\bar{H}^{(\lambda+1)}(s_\lambda)}$$

is precisely a Newton-Raphson iteration performed on a certain rational function. The word *precisely* in the previous sentence is to emphasize that the iteration is not merely of Newton-Raphson *type*. We prove the following

Theorem. *The formula*

$$s_{\lambda+1} = s_\lambda - \frac{P(s_\lambda)}{\bar{H}^{(\lambda+1)}(s_\lambda)}$$

is identical with

$$s_{\lambda+1} = s_\lambda - \frac{W^{(\lambda)}(s_\lambda)}{[W^{(\lambda)}(s_\lambda)]'}, \qquad W^{(\lambda)}(z) = \frac{P(z)}{H^{(\lambda)}(z)}.$$

Proof. Let

$$V^{(\lambda)}(z) = \frac{H^{(\lambda)}(z)}{P(z)}.$$

Hence the first equation of (2.4) may be written as

$$V^{(\lambda+1)}(z) = \frac{V^{(\lambda)}(z) - V^{(\lambda)}(s_\lambda)}{z - s_\lambda}.$$

Observe that

$$V^{(\lambda+1)}(s_\lambda) = [V^{(\lambda)}(s_\lambda)]', \qquad h_0^{(\lambda+1)} = -V^{(\lambda)}(s_\lambda).$$

Hence

$$\begin{aligned} s_{\lambda+1} &= s_\lambda - \frac{P(s_\lambda)}{\bar{H}^{(\lambda+1)}(s_\lambda)} = s_\lambda - \frac{P(s_\lambda)\, h_0^{(\lambda+1)}}{H^{(\lambda+1)}(s_\lambda)} \\ &= s_\lambda + \frac{V^{(\lambda)}(s_\lambda)}{[V^{(\lambda)}(s_\lambda)]'} = s_\lambda - \frac{W^{(\lambda)}(s_\lambda)}{[W^{(\lambda)}(s_\lambda)]'}. \end{aligned}$$

This permits us to regard variable shift iteration in the following manner. We are performing Newton-Raphson iteration on a sequence of rational functions $P(z)/H^{(\lambda)}(z)$. For λ sufficiently large, $P(z)/H^{(\lambda)}(z)$ is as close as desired to a linear polynomial whose zero is ϱ_1. This shows why the process is so powerful.

Note that no differentiation is performed in our calculation of the sequence $\{s_\lambda\}$. The division by $z - s_\lambda$ has the effect of differentiation.

7. Implementation of the Algorithm

The program implementing the algorithm, a discussion of how the program makes its decisions, stability of the algorithm in finite precision arithmetic, the method for efficiently calculating the H polynomials, the results of extensive testing, and a program which clusters the zeros and provides a posteriori error bounds may be found in [3]. Here we confine ourselves to a few observations.

The termination of Stage One, that is, the choice of M, is not crucial. Indeed Stage One is not necessary from theoretical considerations. The function of Stage One is to accentuate the smaller zeros. Numerical experimentation indicates that this makes the decision to terminate Stage Two more reliable. In the implementation, M is set at 5, a number arrived at by numerical experience.

The following three major decisions have to be made by the program:

1. Selection of the shift s.
2. Termination of Stage Two; that is, the choice of L.
3. Termination of Stage Three.

We indicate how these three decisions are made.

Selection of s. This parameter is chosen so that $|s| = \beta$, $\beta < \min |\varrho_i|$, $i = 1, 2, \ldots, j$ and so that

$$|s - \varrho_1| < |s - \varrho_i|, \qquad i = 2, \ldots, j. \tag{7.1}$$

A lower bound on the moduli of the zeros due to Cauchy (Marden [6, p. 98, ex. 1]) is given by the unique positive zero, β, of the polynomial

$$z^n + |a_1| z^{n-1} + \cdots + |a_{n-1}| z - |a_n|.$$

This number is easily calculated by Newton-Raphson iteration. The value of s is then chosen by using random numbers from a uniform distribution to pick a point on the circle of radius β. It is highly probable that the s so chosen will be closest to just one of the zeros of P and hence the condition (7.1) is satisfied. If the condition is not satisfied, the test described below may not be passed in which case a new value of s is chosen. Observe that s need not be closest to the smallest zero of P. It is easy to show that it will be closest to a zero whose modulus is at most three times the modulus of the smallest zero. Hence we guarantee that we will never perform a deflation using a zero which is large compared to other zeros of P. Thus we avoid a situation (Wilkinson [12]) which could lead to serious instability.

Termination of Stage Two. We do not attempt to carry out Stage Two far enough to assure that the conditions of the Theorem of Section 3 are satisfied. (These are only *sufficient* conditions.) Instead we test for the convergence of the sequence $s - P(s)/\bar{H}^{(\lambda)}(s)$. Experience has shown that it is efficient to terminate Stage Two after only a very weak test for convergence has been passed. Let $t_\lambda = s - P(s)/\bar{H}^{(\lambda)}(s)$. If t_λ, $t_{\lambda+1}$, $t_{\lambda+2}$ are defined and

$$|t_{\lambda+1} - t_\lambda| \leqq \tfrac{1}{2}|t_\lambda|, \quad |t_{\lambda+2} - t_{\lambda+1}| \leqq \tfrac{1}{2}|t_{\lambda+1}|$$

then we terminate Stage Two.

If the test is not passed by the time λ reaches a certain value (which itself is varied depending on how many shifts have been tried), a new value of s is generated with modulus β and random amplitude.

Termination of Stage Three. As in [5] we terminate Stage Three when the computed value of the polynomial at s_λ is less than or equal to a bound on the roundoff error in evaluating $P(s_\lambda)$. Numerical experience indicates this is a good stopping criterion for polynomial zero-finding as it stops the iteration just as the limiting accuracy has been reached.

8. Numerical Results

Extensive numerical experimentation, performed on an IBM 360/67, leads to the following timing results. For polynomials with degrees ranging from 20 to 50, the time required to calculate all zeros averages $8\,n^2$ milliseconds. The time for all the polynomials of degree 20 or greater which were tested ranges from $6\,n^2$ to $14\,n^2$ milliseconds. The polynomials used in the testing range from polynomials with randomly chosen zeros to polynomials with multiple zeros and clusters of near equimodular zeros. The fact that the time required is insensitive to the distribution of zeros is most encouraging.

The algorithm reported in this paper was not tailored for polynomials with real coefficients. In [4] we shall report on an algorithm designed for real coefficients. The real algorithm cuts the time by a factor of roughly four.

These figures were obtained from an ALGOL W implementation of the algorithm. A FORTRAN implementation has also been written and is faster.

For illustration we exhibit a low degree numerical example, which is not intended to prove anything about the efficacy of the algorithm and its implementation. This has been done through extensive testing [3].

The example given below has a zero of multiplicity two as well as three almost equimodular zeros, two of which form a near-multiple pair.

$$P(z) = z^5 - (13.999 + 5i)z^4 + (74.99 + 55.998i)z^3 - (159.959 + 260.982i)z^2 + (1.95 + 463.934i)z + (150 - 199.95i),$$

$$P(z) = (z - 1 - i)^2(z - 4 + 3i)(z - 4 - 3i)(z - 3.999 - 3i).$$

In calculating each of the zeros below, five no-shift steps were taken ($M = 5$). In the table we give the value of s used in Stage Two, the number of Stage Two steps ($L - M$), the value of s_L used to start Stage Three and the iterates s_{L+j} used in Stage Three. The program was written in ALGOL W and run on Stanford University's IBM 360/67.

Observe that the well-conditioned zero at $4 + 3i$ is calculated accurately even though the polynomial has already been deflated with three ill-conditioned zeros.

Table. *A numerical example*

Zero (1)	$s = -0.37087 + 0.17907i$,	$L - M = 2$, $s_L = 0.99996 + 1.0001i$
	j	s_{L+j}
	1	$1.00000\,00000\,0426 + 0.99999\,99999\,78201i$
Zero (2)	$s = -0.66572 - 0.093\,001i$,	$L - M = 2$, $s_L = 0.99975 + 1.0005i$
	j	s_{L+j}
	1	$0.99999\,99999\,74143 + 1.00000\,00000\,0192i$
	2	$0.99999\,99999\,95742 + 1.00000\,00000\,2180i$
Zero (3)	$s = 1.1968 + 0.92016i$,	$L - M = 4$, $s_L = 4.6023 + 3.0859i$
	j	s_{L+j}
	1	$3.99420\,18119\,1240 + 3.00623\,80363\,9207i$
	2	$3.99946\,02802\,2349 + 2.99995\,36718\,6768i$
	3	$3.99939\,73528\,8312 + 2.99988\,65887\,9128i$
	4	$3.99900\,06946\,9391 + 3.00038\,35031\,3524i$
	5	$3.99892\,67498\,2940 + 3.00000\,10128\,1734i$
	6	$3.99899\,96522\,0352 + 2.99999\,91830\,1853i$
	7	$3.99899\,99999\,7589 + 2.99999\,99999\,8516i$
Zero (4)	$s = 0.26797 - 2.3881i$,	$L - M = 4$, $S_L = 3.8372 - 2.6754i$
	j	S_{L+j}
	1	$3.99861\,69562\,4235 - 3.00234\,33418\,8375i$
	2	$4.00000\,00075\,7741 - 3.00000\,00043\,5632i$
	3	$4.00000\,00000\,0000 - 3.00000\,00000\,0000i$
Zero (5)		$4.00000\,00000\,2411 + 3.00000\,00000\,1484i$

References

1. Bauer, F. L.: Beiträge zur Entwicklung numerischer Verfahren für programmgesteuerte Rechenanlagen. II. Direkte Faktorisierung eines Polynoms. Bayer. Akad. Wiss. Math.-Nat. Kl. S.-B., 163—203 (1956).
2. — Samelson, K.: Polynomkerne und Iterationsverfahren. Math. Z. **67**, 93—98 (1957).

3. Jenkins, M. A.: Three-stage variable-shift iterations for the solution of polynomial equations with a posteriori error bounds for the zeros. Stanford Dissertation, 1969.
4. — Traub, J. F.: A three-stage algorithm for real polynomials using quadratic iteration. To appear in SIAM Journal of Numerical Analysis.
5. — — An algorithm for an automatic general polynomial solver. Constructive aspects of the fundamental theorem of algebra, edited by Dejon and Henrici, p. 151—180. New York: Wiley-Interscience 1969.
6. Marden, M.: The geometry of the zeros of a polynomial in a complex variable. Providence, Rhode Island: Amer. Math. Soc. 1949.
7. Ostrowski, A. M.: On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. IV. Generalized Rayleigh quotient for nonlinear elementary divisors. Arch. Rat. Mech. Anal. **3**, 341—347 (1959).
8. Traub, J. F.: A class of globally convergent iteration functions for the solution of polynomial equations. Math. Comp. **20**, 113—138 (1966).
9. — Proof of global convergence of an iterative method for calculating complex zeros of a polynomial. Notices Amer. Math. Soc. **13**, 117 (1966).
10. — The calculation of zeros of polynomials and analytic functions. Proceedings of Symposia in Applied Mathematics, volume 19, Mathematical aspects of computer science, p. 138—152. Providence, Rhode Island: Amer. Math. Soc., 1967.
11. Wielandt, H.: Bestimmung höherer Eigenwerte durch gebrochene Iteration. Ber. der Aerodynamischen Versuchsanstalt Göttingen, B 44/J/37, 1944.
12. Wilkinson, J. H.: Rounding errors in algebraic processes, chapter 2. Englewood Cliffs, New Jersey: Prentice-Hall 1963.
13. — The algebraic eigenvalue problem. Oxford: Clarendon Press 1965.

M. A. Jenkins
Stanford University
Stanford, California
and Queens University
Kingston, Canada

J. F. Traub
Bell Telephone Laboratories, Incorporated
Murray Hill, New Jersey 07974
USA