CS 71

AN ALGORITHM FOR AN AUTOMATIC GENERAL POLYNOMIAL SOLVER

BY

M. A. JENKINS

J. F. TRAUB

TECHNICAL REPORT NO. CS 71

JULY 21, 1967

COMPUTER SCIENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UNIVERSITY

AN ALGORITHM FOR AN AUTOMATIC GENERAL POLYNOMIAL SOLVER*

by

M. A. Jenkins
Stanford University
Stanford, California

and

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

Presented at a Symposium on Constructive Aspects of the Fundamental Theorem of Algebra at Rüschlikon, Switzerland, held June 5-7, 1967.

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

TABLE OF CONTENTS

1. INTRODUCTION

A general automatic equation solver should be based on a restriction-free mathematical algorithm. By this we mean the algorithm should be suitable for all polynomials and not depend on the properties of certain classes of polynomials. In this paper we will describe a restriction-free algorithm and discuss a program which implements it.

The algorithm enjoys a basic simplicity and requires few decisions. We devise procedures by which the computer may automatically make the major decisions required. We do not concern ourselves here with programs used in an interactive environment. Routines to be used in such an environment might have different characteristics.

We summarize a few of the desirable characteristics of the algorithm. It is basically iterative with a preprocessing stage which guarantees that the iteration will converge. Often the most difficult problem associated with an iterative method is the value of the initial iterate. This is easy for us to handle because the mathematical algorithm will converge for essentially all initial approximations while our implementation of the algorithm actually supplies us with a good initial approximation. Multiple zeros require no special handling. Finally, the importance of finding the zeros in roughly increasing order of magnitude to ensure stable deflation

has been stressed by Wilkinson [11, p. 465] who observes there seems no reliable method for ensuring this. Our algorithm does find the zeros in roughly increasing order of magnitude.

The last point merits some amplification. If the zeros are found in decreasing order of magnitude, then the backward deflation is stable. What is really crucial is that at each stage of the deflation either one of the smallest or one of the largest zeros is calculated.

The algorithm may be applied to polynomials with real or complex coefficients. Polynomials with complex coefficients are easier to deal with for the following reason. Let k be the number of distinct smallest zeros of equal magnitude. Although theoretically the kind of method which we will describe could be extended to handle zero distributions with any value of k, the simplicity of the implementation depends on k being small. In the case of a polynomial with complex coefficients we can, after a complex translation, ensure $k = 1$. For a polynomial with real coefficients, we are left with the cases $k = 1$ and $k = 2$ if we restrict ourselves to real translations.

The algorithm to be introduced in this paper is a member of a class of two-stage methods introduced by Traub. This type of method was first announced in [5]. The calculation of the largest zero of a polynomial was discussed in detail in [6] and global convergence was proven for a class of methods. For the largest zero the first stage involves the generation of G polynomials. The proof of global convergence of an algorithm for computing complex conjugate zeros was announced in [7] while the calculation of the smallest zero and of multiple zeros as well as the extension to analytic functions appears in [8].

The calculation of the smallest zero involves H polynomials. G polynomials and H polynomials have a simple relation and any result involving one can be translated into a result involving the other. Calculating the smallest zero first makes translation more effective. Hence, we shall be involved with H polynomials.

Bibliographic remarks and rather extensive bibliographies may be found in Traub [6], [8].

The papers cited above deal with finding one zero or a complex conjugate pair and focus on mathematical properties. In this paper, we focus on a particular algorithm out of a class of possible algorithms and discuss its feasibility as the basis for a general automatic polynomial equation solver.

2. THE MATHEMATICAL ALGORITHM

Let

$$P(t) = \sum_{j=0}^{n} a_j t^{n-j}, \qquad a_0 = 1, \quad a_n \neq 0$$

be a polynomial with ℓ distinct zeros ρ_i of multiplicity m_i.

Stage One

We generate a sequence of polynomials as follows. Let

$$H(0,t) = P'(t)$$

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

Observe that the polynomials are of degree at most $n - 1$.

Stage Two

Let k be the number of distinct zeros of smallest magnitude. If $k > 2$, translate the polynomial so that $k = 1$ or 2. Observe that the distinction between $k = 1$ and $k = 2$ is of importance only for the case of real coefficients.

We introduce the following notation to help us describe the Stage Two iteration. Let $h(t)$ be a polynomial of degree r. Then $\bar{h}(t)$ is the polynomial $h(t)$ divided by the coefficient of t^r.

Let t_0 be the initial iterate. Then we generate a sequence of iterates by $t_{i+1} = \psi_k(t_i,f)$ where f is the function whose zero we seek. (In this notation, Newton-Raphson iteration is defined by $\psi(t,f) = t - f/f'$.)

We can now give the formulas of the iteration functions for k = 1 and 2.

<u>k = 1</u>

Let

$$t_{i+1} = \psi_1(t_i,f)$$

where

$$\psi_1(t,f) = t - f/f',$$

and

$$f \equiv V(\Lambda,t) = P(t)/H(\Lambda,t) .$$

<u>k = 2</u>

Let

$$t_{i+1} = \psi_2(t_i,f)$$

where

$$\psi_2(t,f) = t - \frac{2f}{f' \pm [(f')^2-4f]^{1/2}}$$

and

$$f \equiv W(\Lambda,t) = P(t)/I(\Lambda,t) ,$$

$$I(\Lambda,t) = \delta(\Lambda-1)H(\Lambda,t)-\delta(\Lambda)H(\Lambda-1,t)$$

with $\delta(\Lambda)$ the coefficient of t^{n-1} in $H(\Lambda,t)$. Let the zero be labeled α. If $k = 1$, the polynomial $P(t)/(t-\alpha)$ is formed. If $k = 2$, the polynomial $P(t)/[(t-\alpha)(t-\bar{\alpha})]$ is formed. We then return to Stage One with the new polynomial.

3. PROPERTIES OF THE MATHEMATICAL ALGORITHM

We shall now state a number of results which exhibit the power of the mathematical algorithm. Results analogous to results stated for the case $k = 1$ may be found in Traub [6]. Proofs of results for the case $k = 2$ will appear elsewhere. The notation is the same as in Section 2. We shall use λ as a running index and Λ as a fixed integer.

Most of our results follow from the formula given in

THEOREM 1.

For all λ,

$$\frac{H(\lambda,t)}{P(t)} = \sum_{i=1}^{\ell} \frac{m_i \rho_i^{-\lambda}}{t-\rho_i} \quad .$$

The key property of $H(\lambda,t)$ is given in the following two theorems.

THEOREM 2.

Let $|\rho_1| < |\rho_i|$, $i > 1$. Then for all finite t,

$$\lim_{\lambda \to \infty} V(\lambda,t) = t - \rho_1 \quad .$$

THEOREM 3.

Let $|\rho_1| < |\rho_i|$ and $|\rho_2| < |\rho_i|$, $i > 2$. Then for all finite t,

$$\lim_{\lambda \to \infty} W(\lambda,t) = (t-\rho_1)(t-\rho_2) \quad .$$

Note that the hypothesis of Theorem 3 includes the case $k = 2$. Generalizations of these theorems hold for the case of k smallest zeros in magnitude. The rate of convergence is of Bernoulli type.

Results concerning the zeros and poles of $V(\lambda,t)$ and $W(\lambda,t)$ are given in the following group of theorems. Note that no restrictions have been imposed on the multiplicities of the zeros of P. The following theorem is a generalization of the statement that the rational function P/P' has only simple zeros. Observe that $V(0,t)$ is proportional to P/P'.

THEOREM 4.

For all finite λ, $V(\lambda,t)$ and $W(\lambda,t)$ have only simple zeros and these are the zeros of P.

THEOREM 5.

Let $|\rho_1| < |\rho_i|$, $i > 1$. Let K_1 be the union of circles with arbitrarily small fixed radii centered at the ρ_i, $i > 1$. Then for λ sufficiently large, the poles of $V(\lambda,t)$ are contained in K_1.

THEOREM 6.

Let $|\rho_1| < |\rho_i|$, $|\rho_2| < |\rho_i|$, $i > 2$. *Let* K_2 *be the union of circles with arbitrarily small fixed radii centered at the* ρ_i, $i > 2$. *Then for* λ *sufficiently large, the poles of* $W(\lambda,t)$ *are contained in* K_2.

We now state some theorems concerning the iteration functions ϕ_1 and ϕ_2. As usual we define the order of the iteration as follows. Let $t_i \to \alpha$. Then if there exists a constant p and a nonzero constant C such that

$$\lim_{t_i \to \alpha} \frac{(t_{i+1}-\alpha)}{(t_i-\alpha)^p} = C$$

then p is called the order and C the asymptotic error constant. For our iteration functions, $C = C_k(\Lambda)$. We then have

THEOREM 7.

ϕ_1 and ϕ_2 are second order iteration functions. Furthermore,

$$\lim_{\lambda \to \infty} C_k(\lambda) = 0, \quad k = 1,2 . \tag{3.2}$$

We comment on this result. The iteration is done for a fixed value of $\lambda = \Lambda$. Theorem 7 shows that if Λ is large, $C_k(\Lambda)$ will be small. Hence, although the iteration is of second order, the error at each iteration will be the product of three small numbers and hence will appear faster than the usual quadratic convergence.

Additional discussion of $C(\Lambda)$ may be found in Traub [6, Section 6] and [8, Section 7].

The speed of convergence is illustrated by the following simple example which we take from Traub [6]. In this example an earlier program is used which calculates the biggest zero first and which does not make decisions automatically.

Let

$$P(t) = t^4 - 46t^3 + 528t^2 - 1090t + 2175.$$

The largest zero is 29. Take $\Lambda = 16$. Let

$$t_0 = 100\ 000.$$

Then

$$t_1 = \quad 28.99963$$

$$t_2 = \quad 28.9999999999997$$

We hope the following discussion will offer some insight into the choice of ϕ_1 and ϕ_2 and will clarify the reason why (3.2) holds.

Let $v(t) = (t-\rho_1)$, $w(t) = (t-\rho_1)(t-\rho_2)$. Then Theorems 2 and 3 may be restated as

$$\lim_{\lambda \to \infty} V(\lambda,t) = v(t) \quad ,$$

$$\lim_{\lambda \to \infty} W(\lambda,t) = w(t) \quad .$$

Now,

$$\rho_1 \equiv \phi_1[t,v(t)] \quad ,$$

$$\rho_1,\rho_2 \equiv \phi_2[t,w(t)] \quad .$$

Thus if $k = 1$ and we have taken λ to ∞, then starting with any t_0, the iteration with ϕ_1 would have delivered the exact zero in one iteration and analogously for $k = 2$ and ϕ_2.

We now state theorems on global convergence of the iterations defined by ϕ_1 and ϕ_2.

THEOREM 8.

Let $|\rho_1| < |\rho_i|$, $i > 1$. Let t_0 be an arbitrary point in the extended complex plane such that $t_0 \neq \rho_i$, $i > 1$ and let

$t_{i+1} = \psi_1(t_i, V)$. Then for Λ sufficiently large but fixed, the sequence t_i is defined for all i and $t_i \to \rho_1$.

THEOREM 9.

Let $|\rho_1| < |\rho_i|$, $|\rho_2| < |\rho_i|$, $i > 2$. *Let* t_0 *be an arbitrary point in the extended complex plane such that* $t_0 \neq \rho_i$, $i > 2$ *and let* $t_{i+1} = \psi_2(t_i, W)$. *Then for* Λ *sufficiently large but fixed, the sequence* t_i *is defined for all* i *and* $t_i \to \rho_1$.

These theorems require a few words of comment. The formulas for ψ_1 and ψ_2 as given above make it appear as if these functions are not defined at ∞. However, ψ_1 and ψ_2 may be rewritten so that they are defined at ∞. (Observe that this is not a property shared by the Newton-Raphson iteration function.)

The iteration ψ_2 is multivalued because of the $\pm$ sign. However, a strategy is available for making the iteration converge to either ρ_1 or ρ_2. A discussion of this in a somewhat different setting is given by Traub [8, Section 12].

A proof of Theorem 8 for the case where P has only simple zeros is given by Traub [6, pp. 121-123]. The extension to multiple zeros is not difficult.

These theorems show that if we apply our two stage algorithm to any polynomial, with perhaps a translation to ensure $k = 1$ or 2, then provided Λ is sufficiently

large, the mathematical algorithm is guaranteed to converge. For the remainder of this paper we discuss the implementation of this algorithm on a digital computer.

4. DECISIONS TO BE MADE IN THE PROGRAM

We enumerate the major decisions that have to be made automatically by a program implementing this algorithm. A number of the decisions are not crucial and are made on an ad hoc basis. Other decisions are crucial and are made on the basis of certain calculations.

We summarize the major decisions to be made in the calculation of each zero or pair of zeros:

a) What is Λ, the value of λ for which we terminate Stage One and switch to Stage Two?

b) Is $k = 1, 2,$ or is $k > 2$?

c) If $k > 2$, by how much should we translate?

d) What value should be assigned to t_0, the initial iterate for Stage Two?

e) What is the termination criterion for the Stage Two iteration?

Decisions a, b, and d are made as the result of the same calculation. Indeed, Stage One is terminated when k can be determined as equal to 1 or 2. If such a determination cannot be made by the time that λ has reached a certain value λ_f, a translation is carried out.

Decision d, which is often the most difficult decision to make, is available here as a byproduct. However, Theorems 8 and 9 show that the choice of t_0 is not crucial.

The methods for making decisions c and e are described in Sections 6 and 7, respectively.

5. THE TERMINATION OF STAGE ONE

If there are k smallest zeros in magnitude, then for λ sufficiently large

$$\frac{H(\lambda,t)}{P(t)} \sim \sum_{i=1}^{k} \frac{m_i \rho_i^{-\lambda}}{t-\rho_i} \quad .$$

Hence $H(\lambda,t)$, ... , $H(\lambda+k,t)$ will approximately satisfy the k-th order recurrence

$$\sum_{i=0}^{k} c_{k-i} H(\lambda+i,t) = 0, \qquad c_0 = 1, \tag{5.1}$$

where the c_i are related to the zeros of P by

$$\sum_{i=0}^{k} c_{k-i} t^i = \prod_{i=1}^{k} (t-\rho_i^{-1}) \quad .$$

We wish to test the hypothesis that the $H(\lambda,t)$ satisfy a recurrence of the form (5.1) with $k = 1$ or 2 . For a fixed value of λ we test the hypothesis $k = 1$ and if that seems to be false we test $k = 2$. If that is also false, we increase λ by a certain amount and test again. This is continued until a preset upper limit of λ is reached. At that point the polynomial is translated and we start again.

We describe the test for $k = 2$. The test for $k = 1$ is the appropriate simplification. Let

$$\delta(\lambda) = \sum_{i=1}^{\ell} m_i \rho_i^{-\lambda}$$

denote the leading coefficient of $H(\lambda,t)$. Let $\underset{\sim}{h}_\lambda$ denote the vector of coefficients of the polynomial $H(\lambda,t)$. We apply two tests, the second being more expensive than the first and applied only if the first is passed.

We first test for a second order scalar recurrence. If this is passed, we test for a vector recurrence.

Let

$$D(\lambda) = \begin{vmatrix} \delta(\lambda+2) & \delta(\lambda+1) \\ \delta(\lambda+1) & \delta(\lambda) \end{vmatrix}$$

$$R(\lambda) = \frac{D(\lambda+1)}{D(\lambda)} \quad .$$

If $\delta(\lambda)$ satisfies a second order scalar recurrence, $R(\lambda)$ converges. Hence the first test is

$$\frac{| R(\lambda+1) - R(\lambda) |}{R(\lambda+1)} < \epsilon \quad . \tag{5.2}$$

If this test is passed, we test for the vector recursion as follows.

Choose $\hat{c}_1$, $\hat{c}_2$ so that the quantity

$$\underset{\sim}{r}_\lambda = \underset{\sim}{h}_{\lambda+1} + \hat{c}_1 \underset{\sim}{h}_\lambda + \hat{c}_2 \underset{\sim}{h}_{\lambda-1} \tag{5.3}$$

is minimized in the L_2 norm and test if

$$\frac{|| \underset{\sim}{r}_\lambda ||_2}{|| \underset{\sim}{h}_\lambda ||_2} < \varepsilon \ . \tag{5.4}$$

That is we solve a $n \times 2$ least squares problem. If (5.4) holds, we calculate the zeros q_1 and q_2 of $t^2 + \hat{c}_1 t + \hat{c}_2$ and use q_1^{-1} or q_2^{-1} as the value of the initial iterate t_0 in the second stage.

A similar least squares technique is used by Zurmühl [12] whose purpose it is to calculate approximations of equimodular eigenvalues using vectors generated by the power method. Zurmühl proves that he obtains Rayleigh approximations in this way. He does not use this as a criterion for termination.

We emphazise that the test tells us

a) When to switch from Stage One to Stage Two.

b) The value of k .

c) The value of t_0 .

6. THE TRANSLATION OF THE POLYNOMIAL

If the tests described in Section 5 have not been passed by a certain value of $\lambda = \lambda_f$, we translate the polynomial. Since we wish to calculate the zeros in the order of increasing magnitude, we do not want to shift by an amount which would place near the origin a zero with a significantly larger modulus than the smallest zeros. To ensure that this does not happen, we calculate a lower bound on the moduli of the zeros and use this quantity for a shift along the real axis. The lower bound we use, (Marden [4, p. 98]), is the unique positive zero of

$$Q(t) = -\sum_{j=0}^{n-1} |a_j| t^{n-j} + |a_n| . \tag{6.1}$$

The positive zero of (6.1) is easily found by Newton-Raphson iteration. It need not be found very accurately.

One may construct examples which show that the smallest zero of the translated polynomial need not be the translated smallest zero of the original polynomial. However, these examples are based on near equimodular zero distributions and hence will not effect our statement that we can ensure stable deflation. However, this shows that the translated polynomial may have more than two equimodular smallest zeros. In the program we try shifts in both directions. An example of this may be seen in Example 3 of Section 8.

We now show how we may perform the translation and still use the original polynomial in the Stage Two iteration. This is clearly desirable numerically.

Let $p(t) = P(t-s)$ be the translated polynomial. With $s > 0$, this means the zeros of P are shifted s units to the right. Let $\{h(\lambda,t)\}$ be the "H polynomial sequence" for $p(t)$ and let η_1 be the smallest zero of $p(t)$. [We assume for simplicity of exposition that $p(t)$ has a smallest zero. This is not essential.] Let $\bar{h}(\lambda,t) = h(\lambda,t+s)$. Let

$$\sigma(\lambda,t) = \frac{P(t)}{\bar{h}(\lambda,t)} .$$

Then we have

THEOREM 10.

Let $p(t)$ *have zeros* η_i *with* $|\eta_1| < |\eta_i|$, $i > 1$. *Let* $t_{i+1} = \psi_1(t_i,\sigma)$. *Then* $t_i \to \eta_1 - s$.

An analogous result holds if the translated polynomial has two smallest zeros. After the zero has been calculated, the deflation is carried out in the original polynomial. Although this scheme requires two translations, it is not sensitive to roundoff error since the iteration is done in the original polynomial and hence the translations need not be done in higher precision than the rest of the calculation.

7. SCALING

We turn first to the scaling of the H polynomials. From (3.1), we see that as λ increases the coefficients of the $H(\lambda,t)$ grow or diminish depending on whether $|\rho_1| < 1$ or $|\rho_1| > 1$. Thus the coefficients must be periodically scaled. To minimize roundoff error it is desirable to scale by a power of the radix (a power of 8 on the Burroughs B5500). This seems preferable to the scaling strategy proposed by Traub [6, Section 9].

We scale the least squares problem by replacing the problem of minimizing (5.3) with the problem of minimizing

$$\underset{\sim}{s}_\lambda = D\underset{\sim}{h}_{\lambda+1} + \hat{c}_1 D\underset{\sim}{h}_\lambda + \hat{c}_2 D\underset{\sim}{h}_{\lambda-1} \tag{7.1}$$

where D is the diagonal matrix whose j-th diagonal element is the average of the j-th component of $\underset{\sim}{h}_{\lambda-1}$, $\underset{\sim}{h}_\lambda$, $\underset{\sim}{h}_{\lambda+1}$. If this scaling is not done, the minimization of (5.3) may reflect only one very large component which might satisfy a three-term recurrence whereas the vector does not.

8. TERMINATION OF SECOND STAGE ITERATION

A problem common to all iterative methods is when to terminate the process. Generally, the decision to terminate has been based on an ad hoc criterion such as $| t_{i+1}-t_i |/| t_i | < \varepsilon$, with the parameter ε chosen a priori.

In our program we terminate iteration on the basis of a technique due to Kahan. He derives an a posteriori bound on the roundoff error in evaluating a real polynomial at a real point and suggests that iteration be stopped when the computed value of the polynomial is less than a small integer multiple of this bound. Kahan's technique appears without explanation in Kahan and Farkas [3]. Adams [1] analyzes the case of a real polynomial evaluated at a complex point and shows that the bound is tight enough so that the iteration is not stopped prematurely. Our experience with this leads us to the conclusion that it is an excellent way in which to terminate the second stage.

9. NUMERICAL RESULTS

An ALGOL program has been written for the Burroughs B5500 to test the algorithm described in this paper. The Stage One calculation is done in single precision (13 octal digits) and the Stage Two iteration is done in double precision (26 octal digits).

The program makes good use of the recursive facility of ALGOL. Flowcharts of the program may be found in the Appendix.

The procedures for automatically making the important decisions listed in Section 4 have been described earlier. A number of other parameters, whose values do not play a critical role, are chosen on an ad hoc basis. We discuss the values assigned these parameters in our program.

The switchover test is applied each time that λ has been increased by 4. The maximum value of λ permitted in Stage One is $\lambda = 200$. If the switchover test has not been passed by this time, we translate.

The maximum number of iterations permitted in Stage Two is 6. This choice is based on the assumption that there will be at least 1 correct figure in the initial approximation and with quadratic convergence 6 iterations will produce a double precision answer. (Double precision on the Burroughs B5500 is 23 decimal digits.)

The number ε appearing in (5.2) and (5.4) is initially set at .001. If the switchover test is passed and then iteration does not converge we replace ε by $\varepsilon/10$ and restart the Stage One calculation.

If the switchover test is not passed for $\lambda = 200$ and iterations following translation in both directions fail, then we increase the upper limit on λ and restart Stage One with ε replaced by 10ε.

We turn to three numerical examples. We tried the program on some of the hardest problems we could find. Examples of how this type of method does on simple problems may be found in Traub [6, Appendix].

What are hard problems for zero finders? Multiple and near multiple zeros cause difficulty for most methods. Wilkinson [9] points out the difficulty in solving a polynomial whose zeros lie in an arithmetic progression. Equimodular and near equimodular zeros are difficult for methods involving zero separation such as Graeffe's method and power methods. Since Stage One of our algorithm can be interpreted as a power method, this presents us with our hardest problem.

Example 1. This is the 20-th degree polynomial with zeros at $1, 2, \ldots, 20$ discussed by Wilkinson. All the zeros are found to at least 10 decimal places of accuracy. Table I gives the zeros in the order in which they were found. Note that the zeros were calculated in strictly increasing order and that except for the zeros at 17 and 18 the value of Λ required to pass the switchover test increases as the ratio of the smallest to the next-to-smallest zero increases. As this ratio increases, the initial approximations become less accurate but

Table I

Zero	Λ	Estimate from Stage One	Number of Iterations
1	12	1.0002	2
2	16	2.0014	2
3	16	3.0096	3
4	20	4.011	3
5	20	5.026	3
6	24	6.025	3
7	24	7.042	3
8	24	8.064	3
9	28	9.055	3
10	28	10.075	3
11	28	11.097	3
12	28	12.12	3
13	28	13.15	4
14	32	14.12	4
15	32	15.15	4
16	32	16.17	4
17	28	17.25	4
18	4	18.14	5
19,20	Found Directly From the Final Quadratic Factor.		

in all cases are within 2% of the zero. Thus our automatic switch-over criterion is working very well. Since $k = 1$, ϕ_1 is used throughout.

Example 2. This is the 19-th degree polynomial whose zeros are $.025 \pm .035i$, $-.04 \pm .03i$, $.27 \pm .37i$, $-.4 \pm .3i$, $2.9 \pm 3.9i$, $-4 \pm 3i$, $10 \pm 2i$, -20, 20, 30, 30, 30. The first six pairs of complex conjugate zeros were chosen to test how the algorithm behaves with complex zeros of nearly equal modulus which are not clustered. The zeros at 20 and -20 test the algorithm on equal real roots with opposite sign and the triple zero at 30 tests the behavior of multiple zeros. Table II gives the results in the order the zeros were found. The zeros are found by the algorithm to eleven significant figures except for the multiple zero at 30 for which the last two approximations agree only to 7 significant figures. This is all one expects for a triple zero. The results show that as the ratio of the smallest modulus to the next-to-smallest modulus increases, the switchover value of Λ increases. For the first two pairs of zeros the ratio is $.860$ and Λ is 68. For the next two pairs the ratio is $.916$ and Λ is 100. For the last two pairs in this group of zeros the ratio is $.972$ and the test is not passed for any $\lambda \leq 200$. In this case the program automatically shifts the zeros by 2.93, the test is now passed with $\Lambda = 12$, and the algorithm converges to the zero at $-4 + 3i$.

Table II

Zero	Modulus	Λ	Estimate From Stage One	Number of Iterations
.025 + .035i	.0430	68	.0250015 + .0350012i	3
-.04 + .03i	.0500	8	-.03999988 + .0300002i	2
.27 + .37i	.458	100	.26997 + .36993i	3
-.4 + .3i	.500	8	-.400002 + .2999996i	2
-4. + 3.i	5.00	12*	-4.0005 + 3.0003i	3
2.9 + 3.9i	4.86	12	2.9004 + 3.90009i	3
10. + 2.i	10.198	20	9.996 + 1.996i	3
-20.	20	12	-20.0089	2
20.	20	4	20.0000006	2
30.	30	4	29.999999995	0
30,30	Found Directly From the Quadratic			

*Switchover test not satisfied at $\lambda = 200$. Shift the zeros by 2.93 and the test is passed with $\Lambda = 12$.

Example 3. This is the 36-th polynomial whose coefficients were chosen randomly by Henrici and Watkins [2]. All its zeros lie close to the unit circle which make this example a difficult one for our algorithm. The polynomial is $P = -9265.3t^{36} + 6468t^{35} - 42.015t^{34} + 70.311t^{33} + 3072.4t^{32} + 2.953t^{31} + 5.6163t^{30} + 870.73t^{29} - 7.9141t^{28} - 74.110t^{27} - 22.964t^{26} + 9.2252t^{25} - 2.4987t^{24} - 39.063t^{23} + 6.5810t^{22} - 6.8461t^{21} - 7.8867t^{20} - 32.151t^{19} - 34.637t^{18} + 67.916t^{17} - 390.57t^{16} + 60.247t^{15} + 265.74t^{14} - 453.86t^{13} - 7015.6t^{12} - 309.67t^{11} - 2.0574t^{10} - 85.581t^{9} - 99.394t^{8} - 20.775t^{7} + 49.225t^{6} + 3924.5t^{5} - .083830t^{4} + 73.941t^{3} + 0.049060t^{2} + 88.312t - 993.56$.

Many of the zeros required shifts before they were found. The column "number of shifts" in Table III has the following meaning. If the entry is 1 a shift of the zeros to the right has succeeded. If 2, then the first shift has failed to produce a polynomial whose smallest root or pair of roots could be found, but a subsequent shift of the zeros to the left succeeded. If the entry is 2^{+}, then both the shifts have failed and the original Stage One calculation has been restarted. Note that one of the largest zeros has been calculated first. Since all the zeros are of comparable magnitude this does not cause trouble during deflation. Purification in the original polynomial produces no change in the approximate zeros in the 10 significant figures which are quoted in Table III.

Table III

Zero	Modulus	Number of Shifts	Λ	Number of Iterations
-.9828508293 + .1124801357i	.989	1	44	3
-.5871954694 + .4819686407i	.760	1	48	3
.2208328178 + .7000044671i	.734	0	60	3
.8127598823 + .0614797464i	.815	0	96	3
-.8845981101 + .3015196625i	.935	1	108	4
-.8372852532 + .3962960916i	.926	1	72	3
-.7466486856 + .6041234511i	.960	1	144	4
.6330847696 + .6931412366i	.939	2	92	4
-.2094825011 + .8999616203i	.924	0	152	3
-.5831113093 + .7852246824i	.978	1	184	4
-.3772294487 + .8980728714i	.974	1	96	4
-.0725823577 + .9908757215i	.994	1	164	4
.5788786350 + .7720099987i	.965	0	120	3
.3926236740 + .9249986413i	1.005	2+	384	5
.1585296127 + .1000329321i	1.013	1	68	4
.8264593743 + .5773500167i	1.008	0	188	5
.9538469054 + .3607127224i	1.020	0	92	4
1.053012577 + .0877037032i	1.057	Found Directly From the Quadratic		

10. SUMMARY

Our major conclusion is that the algorithm described here can be used as the basis for a program which automatically calculates all the zeros of a polynomial and finds them in roughly increasing order of magnitude.

It is clear that our program could not compete in terms of computer time with a program which simply always uses Muller or Newton-Raphson iteration, when these iterations converge. Note, however, that if a problem is easy, then the switchover from Stage One to Stage Two is made early. Hence, easy problems are handled relatively cheaply. Time is spent on the hard zeros. One may view the technique as one which involves a spectrum of iteration functions with the appropriate iteration automatically selected.

Our program is designed to be a general polynomial solver. It must be able to handle all polynomials. If one knows a priori that one is dealing with a special polynomial, such as one with all distinct real zeros, then Newton-Raphson or Laguerre iteration may be used safely. Furthermore, a computer library should probably contain special routines for handling quadratic, cubic, and perhaps quartic equations. A general polynomial solver is needed to handle the cases where special properties of the polynomial are not known or if there is to be only one polynomial solver in the library.

The only difficult case for our algorithm is when there are near equimodular zeros such that our translations don't break the near equimodularity. We are studying methods for handling this

problem either by a suitable modification of the algorithm or by switching to another method in case this difficulty occurs.

We are considering the feasibility of using complex translations even in the case of real polynomials. This would mean only the case $k = 1$ need be considered. It would also offer greater flexibility in translation.

Our program is incomplete in that no attempt is made to give a posteriori estimates of how good the calculated zeros are.

The numerical examples exhibited here, as well as other examples we have run, indicate that the switchover test works quite efficiently. In almost every case the test is not passed until λ is large enough so that the Stage Two iteration converges. The value of λ at the switchover point increases as the ratio of the magnitude of the smallest zero to the magnitude of the next smallest zero increases. Usually the approximation from Stage One to Stage Two is good to between two to four figures which indicates the first stage has not been carried too far.

Observe that very few iterations are required in Stage Two as we would expect from the discussion of Section 3. Our numerical results confirm Wilkinson's conclusion [10, pp. 65] that there is little to be gained by purification in the original polynomial provided that the zeros have been deflated in the proper order. We have not found a single case where a zero is significantly improved by purification. This also indicates that our procedure for terminating Stage Two is working well.

11. ACKNOWLEDGMENTS

It is a pleasure to express our appreciation to Professor W. Kahan of the University of Toronto who made many valuable comments and suggestions on the work reported in this paper. One of his contributions was to suggest the method of "double translation". Much of the research for this paper was done while one of us (JFT) was enjoying the hospitality of Professor G. E. Forsythe's Computer Science Department at Stanford University. The authors would like to thank the National Science Foundation and the Office of Naval Research for partial support of this research.

BIBLIOGRAPHY

[1] Adams, D., A Stopping Criterion for Polynomial Root Finding. To appear Comm. ACM, 1967. Also available as Technical Report 55, Computer Science Department, Stanford University.

[2] Henrici, P. and Watkins, Bruce O., Finding Zeros of a Polynomial by the Q-D Algorithm. Comm. ACM 8 (1965), pp. 572-573. See also Thomas, Richard F. Jr., Corrections to Numerical Data on Q-D Algorithm Comm. ACM 9 (1966), p. 322.

[3] Kahan, W. and Farkas, I., Algorithm 168 and Algorithm 169. Comm. ACM 6 (1963), p. 165.

[4] Marden, M., The Geometry of the Zeros of a Polynomial in a Complex Variable. Amer. Math. Soc., Providence, Rhode Island, 1949.

[5] Traub, J. F., A Class of Globally Convergent Iteration Functions for the Solution of Polynomial Equations. Proc. IFIP Congress 65, Vol. 2, pp. 483-484. Spartan Books, Washington, D.C.

[6] Traub, J. F., A Class of Globally Convergent Iteration Functions for the Solution of Polynomial Equations. Math. Comp. 20 (1966), pp. 113-138.

[7] Traub, J. F., Proof of Global Convergence of an Iterative Method for Calculating Complex Zeros of a Polynomial, Notices Amer. Math. Soc. 13 (1966), p. 117.

[8] Traub, J. F., The Calculation of Zeros of Polynomials and Analytic Functions. To appear in Proceedings of a Symposium on Mathematical Aspects of Computer Science, Amer. Math. Soc., Providence, Rhode Island, 1967. Also available as Technical Report 36, Computer Science Department, Stanford University.

[9] Wilkinson, J. H., The evaluation of the Zeros of Ill-Conditioned Polynomials. Part I. Num. Math. 1 (1959), pp. 150-166.

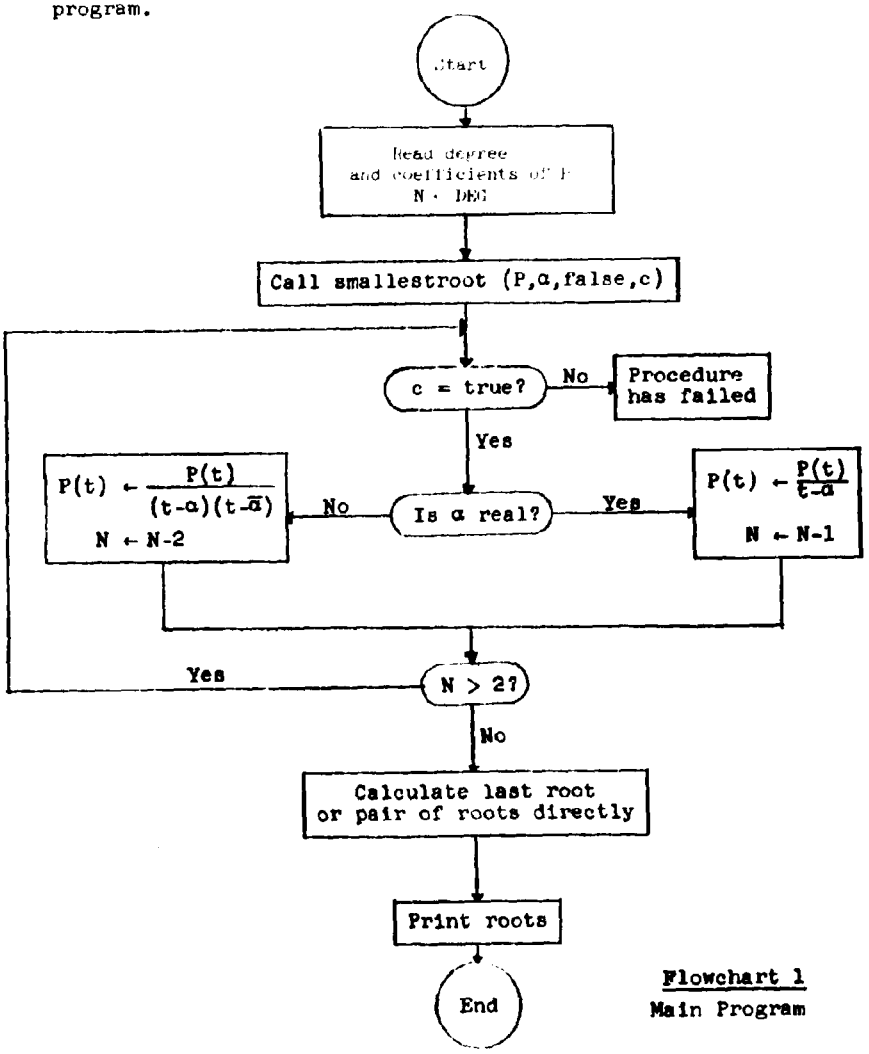
[10] Wilkinson, J. H., Rounding Errors in Algebraic Processes. Prentice-Hall, 1963.

[11] Wilkinson, J. H., The Algebraic Eigenvalue Problem. Clarendon Press, 1965.

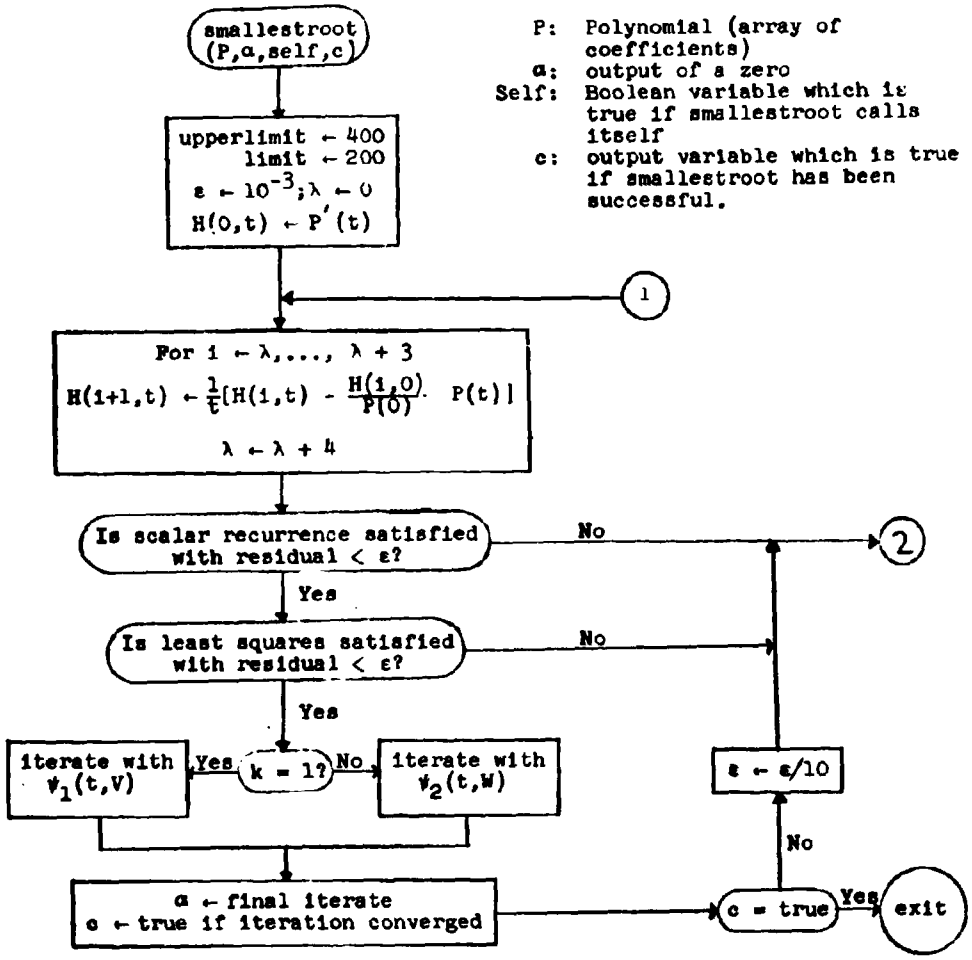
[12] Zurmühl, R., Rayleigh-Näherungen für Simultan-Iteration an betragsgleichen Eigenwerten einer Matrix. ZAMM 42 (1962), pp. 210-213.

Flowcharts

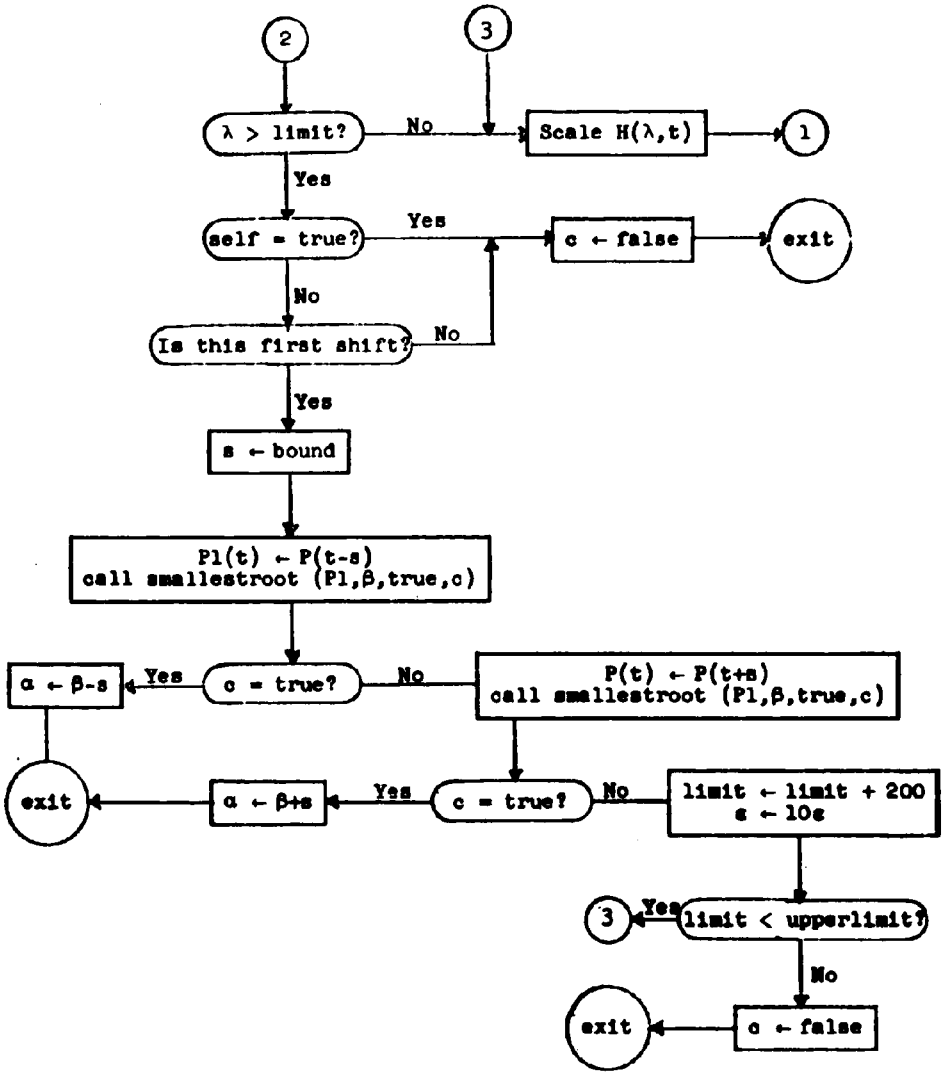
These flowcharts are intended only to give the general flow of the program.



Flowchart 1
Main Program



Flowchart 2 - Procedure smallestroot



Flowchart 3 - Procedure smallestroot (continued)

BLANK PAGE

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Computer Science Department Stanford University Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP ---

3. REPORT TITLE

AN ALGORITHM FOR AN AUTOMATIC GENERAL POLYNOMIAL SOLVER

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Manuscript for Publication (Technical Report)

5. AUTHOR(S) (Last name, first name, initial)

Jenkins, M. A., and Traub, J. F.

6. REPORT DATE 7-21-67	7a. TOTAL NO. OF PAGES 58	7b. NO. OF REFS 12
8a. CONTRACT OR GRANT NO. Nonr-225(37) b. PROJECT NO. NR-044-211 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) CS71 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) none	

10. AVAILABILITY/LIMITATION NOTICES

Releasable without limitations on dissemination.

11. SUPPLEMENTARY NOTES ---	12. SPONSORING MILITARY ACTIVITY Office of Naval Research Code 432 Washington, D. C. 20360

13. ABSTRACT

A restriction-free algorithm for a general automatic polynomial equation solver is described and its implementation as an ALGOL program is discussed. The First Stage of the algorithm is a preprocessing step which quarantees that the iterative Second Stage will converge. The zeros are found one or two at a time and in increasing order of magnitude which quarantees stable deflation. The algorithm automatically decides when to switch to Stage Two and the decision is made easily for a zero which is "easy" to calculate. Flowcharts for the program are given and numerical results are presented for 3 hard problems. The present program is a research program and certain improvements will have to be made before the program could serve as a general library routine.

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
polynomial						
equation solver						
automatic algorithm						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.

DD FORM 1 JAN 64 1473 (BACK)

Unclassified

Security Classification