A Stopping Criterion for Polynomial Root Finding

DUANE A. ADAMS
Stanford University, Stanford, California*

When searching for the root of a polynomial, it is generally difficult to know just when to accept a number as an adequate approximation to the root. In this paper an algorithm is presented which allows one to terminate the iteration process on the basis of calculated bounds for the roundoff error which occurs in evaluating the polynomial. This stopping criterion has been tested on numerous examples and has been found to serve as a satisfactory means for accepting a complex number as a zero of a real polynomial.

1. Introduction

When locating the zeros of a polynomial, it is usually difficult to know just when to terminate the iteration process. It is desirable to terminate the process when the zero is known to within roundoff accuracy. Various ad hoc stopping criteria have been used; however, such criteria do not take into account particular properties of the polynomial being evaluated. Such properties might include the condition of the polynomial, multiple zeros, or clusters of zeros. In this paper a stopping criterion is presented which requires that the value of the polynomial be smaller than a calculated bound for the roundoff error.

Bounds for the roundoff error can be obtained by using the methods of range arithmetic [1] or interval arithmetic [4], but such methods require a large amount of computation. The algorithm described here produces similar bounds, and offers the advantage of being easily calculated. Kahan and Farkas [3] have used this algorithm to bound the roundoff error for a real polynomial evaluated at a real point, but they do not explain why the algorithm works. In this paper, Kahan's bounds for a real polynomial evaluated at a real point are summarized, and then the analysis is extended to a real polynomial evaluated at a complex point. The use of this bound as a stopping criterion is discussed.

2. Summary of Results for a Real Polynomial Evaluated at a Real Point

This section contains a brief summary of Kahan's[1] results on bounding the roundoff error for a real polynomial evaluated at a real point. Consider the polynomial

$$P(Z) = a_0Z^n + a_1Z^{n-1} + \cdots + a_n .$$

The Horner recurrence to compute $P(x)$ is

$$b_0 = a_0 , \tag{1}$$
$$b_k = x \cdot b_{k-1} + a_k \quad \text{for} \quad k = 1, \cdots , n.$$

The last term of this recurrence, b_n, is $P(x)$.

Rounding errors in calculating the b_k will prevent (1) from being satisfied precisely. In order to bound these errors, we note that when the statement

$$s := u + v$$

is executed [7], the number s which is produced satisfies

$$| u + v - s |/| s | \leqq \sigma \leqq \tfrac{1}{2}\beta^{1-t},$$

and when the statement

$$p := u \times v$$

is executed, the number p satisfies

$$| uv - p |/| uv | \leqq \pi \leqq \tfrac{1}{2}\beta^{1-t}.$$

The numbers σ and π used above are bounds which hold for each addition and multiplication, respectively, in the evaluation of any given polynomial. β represents the base in which the machine's floating-point *rounded* arithmetic is performed, and t represents the number of digits in the mantissa. For the Burroughs B5500, an octal machine with a 39-bit mantissa, we have $\beta = 8$ and $t = 13$.

Associated with (1) we have a second recurrence, given by

$$e_0 = | a_0 | \pi/(\pi + \sigma),$$
$$e_k = | x | e_{k-1} + | b_k | \quad \text{for} \quad k = 1, \cdots, n, \tag{2}$$

where b_k represents the calculated quantity. Kahan shows that a bound for the roundoff error is given by

$$| P(x) - b_n | \leqq (\sigma + \pi)e_n - | b_n | \pi. \tag{3}$$

He also shows that a suitable criterion for accepting x as a zero of P to within the bounds given for the roundoff error is

$$| b_n | \leqq 2E \tag{4}$$

where

$$E = (\sigma + \pi)e_n - | b_n |\pi.$$

The factor 2 in (4) guarantees that there exists at least one number, representable in the computer, which satisfies (4). Note that the above criterion does not tell us how close we are to a zero, but only that we are in some interval about the zero where roundoff error may be dominating our calculations.

3. Rounding Error Bounds for a Real Polynomial Evaluated at a Complex Point

Now suppose that

$$P(Z) = a_0Z^n + a_1Z^{n-1} + \cdots + a_n$$

is a polynomial with real coefficients a_k, but that we wish

* Assigned to the Department of Computer Science by the Air Force Institute of Technology

[1] Lectures presented by Professor Kahan at Stanford University, Spring 1966.

to evaluate the polynomial at a point $Z = x + i \cdot y$. By dividing a quadratic factor into the polynomial we can obtain the well known recurrence [8] for evaluating this polynomial at a point in the complex plane which involves only real arithmetic and a total of $2 \cdot n$ multiplications. Thus if we write

$$P(Z) = (Z^2 + pZ + q)(b_0 Z^{n-2} + b_1 Z^{n-3} + \cdots + b_{n-2}) + R(Z - x) + S,$$

we obtain the recurrence

$$\begin{aligned} b_0 &= a_0, \\ b_1 &= a_1 - p \cdot b_0, \\ b_k &= a_k - p \cdot b_{k-1} - q \cdot b_{k-2} \quad \text{for} \quad k = 2, \cdots, n-1, \\ b_n &= a_n + x \cdot b_{n-1} - q \cdot b_{n-2}, \end{aligned} \tag{5}$$

where

$$p = -2x, \quad q = x^2 + y^2, \quad b_{n-1} = R \quad \text{and} \quad b_n = S.$$

Finally

$$P(x + i \cdot y) = b_n + i \cdot y \cdot b_{n-1}.$$

The coefficients a_k which appear in the machine may not be identical to the coefficients of the original problem because of the error in converting from decimal to binary. We shall not be concerned with this error, but rather with the errors which accumulate in attempting to evaluate the polynomial represented in the machine.

When the program which realizes (5) is executed, the numbers stored in cells called "b_k" will not satisfy (5) precisely but will instead satisfy

$$\begin{aligned} b_0 &= a_0, \\ b_1 &= (a_1 - \bar{p} \cdot b_0(1 + \pi_{11}))/(1 + \sigma_{11}), \\ b_k &= ((a_k - \bar{p} \cdot b_{k-1}(1 + \pi_{1k}))/(1 + \sigma_{1k}) \\ &\qquad - \bar{q} \cdot b_{k-2}(1 + \pi_{2k}))/(1 + \sigma_{2k}) \\ &\qquad \text{for} \quad k = 2, \cdots, n-1, \\ b_n &= ((a_n + x \cdot b_{n-1}(1 + \pi_{1n}))/(1 + \sigma_{1n}) \\ &\qquad - \bar{q} \cdot b_{n-2}(1 + \pi_{2n}))/(1 + \sigma_{2n}). \end{aligned} \tag{6}$$

Here the Greek letters σ_{jk} and π_{jk} represent the contributions due to roundoff. We can bound each of the quantities σ_{jk} and π_{jk} on the basis of the floating-point arithmetic of the computer being used; that is,

$$|\sigma_{jk}| \leqq \tfrac{1}{2}\beta^{1-t}, \quad |\pi_{jk}| \leqq \tfrac{1}{2}\beta^{1-t}.$$

In (6), $\bar{p}$ and $\bar{q}$ represent the calculated values of p and q, respectively, and hence have rounding errors associated with their calculations. For the sake of simplifying the analysis slightly, let us assume that $\bar{q}$ is calculated in double precision and then rounded to single precision. Then we may write

$$\bar{p} = p(1 + \pi_p)$$

and

$$\bar{q} = q(1 + \sigma_q) = (x^2 + y^2)(1 + \sigma_q),$$

where

$$|\pi_p| \leqq \tfrac{1}{2}\beta^{1-t}, \quad |\sigma_q| \leq \tfrac{1}{2}\beta^{1-t}.$$

In practice this double-precision calculation is not necessary.

Solving for the a_k in (6) we find

$$\begin{aligned} a_0 &= b_0, \\ a_1 &= b_1(1 + \sigma_{11}) + p \cdot b_0(1 + \pi_p)(1 + \pi_{11}), \\ a_k &= b_k(1 + \sigma_{1k})(1 + \sigma_{2k}) \\ &\qquad + q \cdot b_{k-2}(1 + \sigma_q)(1 + \pi_{2k})(1 + \sigma_{1k}) \\ &\qquad + p \cdot b_{k-1}(1 + \pi_p)(1 + \pi_{1k}) \\ &\qquad\qquad \text{for} \quad k = 2, \cdots n-1, \\ a_n &= b_n(1 + \sigma_{1n})(1 + \sigma_{2n}) \\ &\qquad + q \cdot b_{n-2}(1 + \sigma_q)(1 + \pi_{2n})(1 + \sigma_{1n}) \\ &\qquad - x \cdot b_{n-1}(1 + \pi_{1n}). \end{aligned} \tag{7}$$

Note that in (7), and for the rest of this analysis, the letters a_k and b_k shall represent the numbers within the machine, and any deviation from the true values is represented by the error bounds. p and q represent true values.

By substituting the a_k of (7) into P and simplifying we find

$$P(Z) = b_n + i \cdot y \cdot b_{n-1} - x \cdot b_{n-1}\pi_n + \sum_{k=1}^{n} \sigma_k b_k Z^{n-k} + \sum_{k=0}^{n-2} b_k(\pi_{k+1} \cdot p/Z + \omega_{k+2}\bar{Z}/Z) \cdot Z^{n-k},$$

where

$$\begin{aligned} 1 + \sigma_1 &= (1 + \sigma_{11}); \quad |\sigma_1| \leqq \tfrac{1}{2}\beta^{1-t} \\ 1 + \pi_n &= (1 + \pi_{1n}); \quad |\pi_n| \leqq \tfrac{1}{2}\beta^{1-t} \\ 1 + \sigma_k &= (1 + \sigma_{2k})(1 + \sigma_{1k}); \quad |\sigma_k| \leqq \beta^{1-t} \\ 1 + \pi_k &= (1 + \pi_{1k})(1 + \pi_p); \quad |\pi_k| \leqq \beta^{1-t} \\ 1 + \omega_k &= (1 + \sigma_q)(1 + \pi_{2k})(1 + \sigma_{1k}); \quad |\omega_k| \leqq \tfrac{3}{2}\beta^{1-t} \end{aligned}$$

and where $Z = x + i \cdot y$ and $\bar{Z} = x - i \cdot y$.

Recalling that the calculated value of the polynomial at $Z = x + i \cdot y$, as given by the recurrence, is supposed to be

$$P(x + i \cdot y) = b_n + i \cdot y \cdot b_{n-1},$$

we have

$$\begin{aligned} &|P(x + i \cdot y) - (b_n + i \cdot y \cdot b_{n-1})| \\ &\quad \leqq \pi |x| \cdot |b_{n-1}| + \sigma(|b_n| + |b_{n-1}| \cdot |Z|) \\ &\quad + (2\pi + \omega)|b_0| \cdot |Z^n| + (\sigma + 2\pi + \omega) \sum_{k=1}^{n-2} |b_k Z^{n-k}|, \end{aligned}$$

where

$$\sigma = \max_k |\sigma_k|, \quad \pi = \max_k |\pi_k|, \quad \omega = \max_k |\omega_k|.$$

Now choose

$$e_0 = |b_0|(2\pi + \omega)/(2\pi + \omega + \sigma),$$
$$e_k = |Z| e_{k-1} + |b_k| \quad \text{for} \quad k = 1, \cdots, n. \tag{8}$$

Hence

$$|b_0| = (2\pi + \omega + \sigma)\cdot e_0/(2\pi + \omega),$$
$$|b_k| = e_k - |Z|\cdot e_{k-1} \quad \text{for} \quad k = 1, \cdots, n,$$

and upon substituting into the above and simplifying we oabtain

$$|P(x + i\cdot y) - (b_n + i\cdot y\cdot b_{n-1})| \leqq (2\pi + \omega + \sigma)e_n \tag{9}$$
$$- (2\pi + \omega)(|b_n| + |b_{n-1}||Z|) + \pi|x||b_{n-1}|,$$

where

$$\pi \leqq \beta^{1-t}, \quad \sigma \leqq \beta^{1-t}, \quad \omega \leqq \tfrac{3}{2}\beta^{1-t}.$$

The formula given in (9) is a generalization of the formula given in (3). To complete the parallelism between the real and the complex cases, we give the generalization of (4). We accept $Z = x + i\cdot y$ as a complex zero of the real polynomial P if

$$|b_n + i\cdot y\cdot b_{n-1}| \leqq E, \tag{10}$$

where

$$E = (2\pi + \omega + \sigma)e_n$$
$$- (2\pi + \omega)(|b_n| + |b_{n-1}||Z|) + \pi|x||b_{n-1}|.$$

Section 4 contains a discussion of this criterion.

4. Use of the Error Bound in a Stopping Criterion

For each zero Z_j of P, let Ω_j denote the set of all machine representable points which lie in a neighborhood about Z_j and which satisfy (10). We may imagine that Ω_j defines a region about Z_j in the complex plane. This region may not be simply connected, and its size and shape will depend on both the polynomial P and the computer arithmetic. If our error bound is a good one, then we will not be able to distinguish any of the points in Ω_j from the true zero Z_j on the basis of calculated function values, for any nonzero values of $P(Z)$ for $Z \in \Omega_j$ will be mostly made up of "noise."

We have made rather extensive tests to see how the bound given in (10) compares with the actual roundoff errors. Included in our tests have been the polynomials given in Table 1 and Table 2 of Henrici [4]. The zeros of these polynomials were determined using the method suggested by Traub [5, 6]. The iteration process was terminated when (10) was satisfied. After all the zeros of each polynomial had been located, they were then resubstituted into the original polynomial and evaluated in both single and double precision. Any zeros which did not satisfy (10) were purified until (10) was satisfied using the original P. The roundoff error is then the difference between the evaluations in single and double precision.

Figure 1 shows a distribution of the ratios of roundoff error to the roundoff error bound when (10) was first satisfied for each zero. The calculations were performed on a Burroughs B5500, an octal machine. From Figure 1 we see that in nearly 85 percent of our examples the roundoff error is bigger than 0.01 times the error bound, and this we feel is a reasonable bound for the error.

The distribution shown in Figure 1 tells us how the roundoff error compares with the error bound, but not how close we are to a zero of P. When (10) is satisfied we know only that we are within the region Ω_j. However, our analysis of the data indicates that, in the majority of the examples we have tested, we are sufficiently close to the zero when the stopping criterion is satisfied that even one more iteration is unwarrented. The extra iteration may result in no change, there may be a perturbation in the roundoff error but the answer is not improved, or the answer may be improved by 2 or 3 units in the last decimal.

In referring to the region Ω_j about each zero, we have not dealt with the case where Ω_j may be empty. If there is no machine representable number which satisfies the error bound, then the algorithm would search endlessly for such a value unless terminated after a preassigned number of steps. We have not been able to prove that there always exists a machine number which satisfies (10). On the other hand, we have not found an example where there is no such number. For the real case, Kahan has shown that doubling the error bound is enough to make (4) satisfiable.

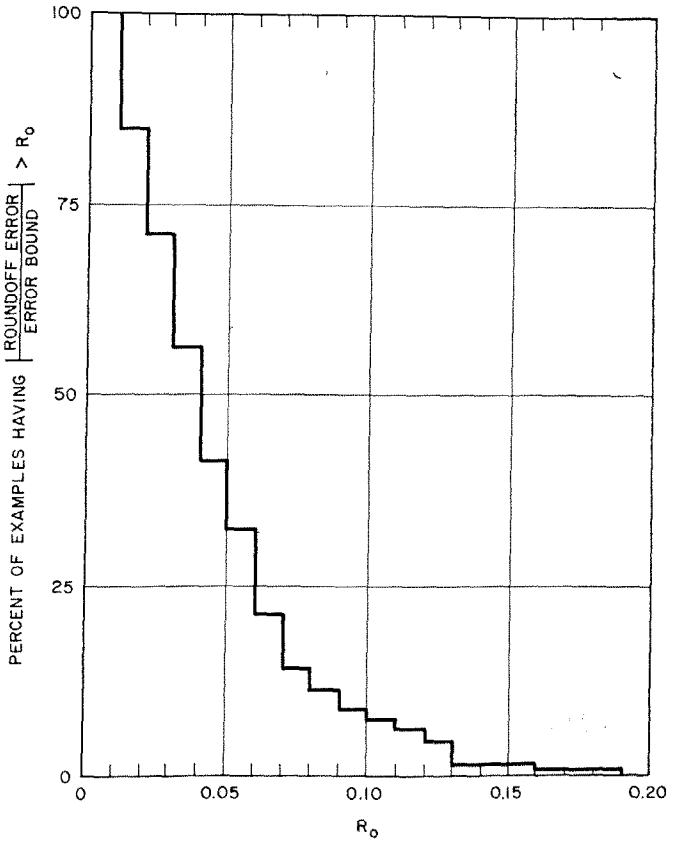

FIG. 1. Distribution of the ratio of roundoff error to error bound

For the complex case it can probably be shown that for some small multiple of the error bound, there is always a number Z in the machine which satisfies the bound (10). However, we have not shown this.

5. Conclusions

The stopping criterion given in (10) serves as a satisfactory means for accepting a number Z as a complex zero of a real polynomial. It is based on having a bound for the roundoff error, which is easily calculated along with the polynomial value by using the recurrence given in (8). Little is to be achieved by iterating beyond the stopping criterion. An open question at present is whether or not there always exists a machine representable number which satisfies (10).

Acknowledgments. I wish to thank Professor J. F. Traub for suggesting this problem to me and for his patience in reading the paper and suggesting improvements.

Also I wish to thank Professor W. Kahan for his many helpful suggestions.

RECEIVED JANUARY, 1967; REVISED JUNE, 1967

REFERENCES

1. GIBB, ALLAN. Algorithm 61 procedures for range arithmetic. *Comm. ACM 4* (July, 1961), 319–320.
2. HENRICI, P., AND WATKINS, BRUCE O. Finding zeros of a polynomial by the Q-D algorithm. *Comm. ACM 8* (Sept. 1965), 572–573; corrections given by THOMAS, RICHARD F. JR. Corrections to numerical data on Q-D algorithm. *Comm. ACM 9* (May 1966), 322.
3. KAHAN, W., AND FARKAS, I. Algorithm 168 and Algorithm 169. *Comm. ACM 6* (Apr. 1963), 165.
4. MOORE, R. E. Interval arithmetic and automatic error analysis in digital computing. Tech. Rep. No. 25, Appl. Math and Statistics Lab., Stanford U., Stanford, Calif., Nov. 1962, 134 pp.
5. TRAUB, J. F. A class of globally convergent iteration functions for the solution of polynomial equations. *Math. Comput. 20* (1966), 113–138.
6. ——. The calculation of zeros of polynomials and analytic functions. In *Proceedings of a Symposium on Mathematical Aspects of Computer Science*, American Mathematical Society, Providence, R. I., to appear. Also available as Tech. Rept. 36, Computer Science Dept., Stanford U., Stanford, Calif.
7. WILKINSON, J. H. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.
8. ——. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, England, 1965, pp. 447–449.

STANDARDS: On Draft COBOL Standard

TO: THE ACM MEMBERSHIP
FROM: R. W. BEMER, *ACM Standards Committee*

BEMA, sponsor of USASI X3, has published Document X3/85, the draft COBOL standard, via SICPLAN Notices [Vol. 2, #4, 1967 April] with NBS support as COBOL Information Bulletin #9. Further details were given on page 440 of the July issue of the *Communications*, specifying how the document could be obtained. This publication preceded a six-month balloting period for members of X3, which period closes 1967 December 31. However, it is desirable for the vote to be as complete as possible prior to the November 6–10 meeting in Paris of ISO/TC97/Subcommittee 5 on Programming Languages, to permit the U.S. delegation to report the latest status and give some indication of the U.S. position on this document. Several votes have already been cast. For purposes of SC5, this document is only a working paper for discussion as it could not, due to difficulties of formulation within X3.4, meet the requirements of submission to member countries four months prior to the meeting.

ACM is a voting member of X3, and the Steering Committee of the ACM Standards Committee recommended unanimously an affirmative vote at its August 4 meeting. Considering the existence span of the language and the vast amount of work that has gone into the preparation of this document, the ACM could cast a negative vote only on the basis of substantial new evidence of error or omission. Quite naturally there are already many suggestions received for improving the wording and/or substance of the many elements of this document. Some complaints have been heard that the various options provide too many combinatorial versions of COBOL to be controllable for interchange between processors, and that a few of these should be selected to the exclusion of the rest. The U.S. Department of Defense is currently choosing some specific levels, which have themselves been subject to attack for having too large a gap between levels to be economical to the user.

The purpose of this letter is to provide the opportunity to close this loop with the ACM membership. The present status is very advanced, and minor changes or improvements should (and probably will) be introduced only at the international level of SC5 or in the next revision of the U.S. standard. Another standard of comparable importance and complexity, USASCII, did in fact go through two revisions in the U.S. version to conform to international considerations. This indicates the impracticality and diminishing return of holding back a standard until a state of perfection is reached.

If no major objections can be advanced, not only will this document become a USA standard, but the possibility exists for it to be a Federal standard as well, and would thus be specifiable in government contracts. ACM members—the banns are hereby posted.—R.W.B.