Exact Algorithms for Polynomial Real Root Approximation Using Continued Fractions

A. G. Akritas, Lawrence, and **King H. Ng,** Houston

Received June 1, 1981

Abstract — Zusammenfassung

Exact Algorithms for Polynomial Real Root Approximation Using Continued Fractions. This paper discusses a set of algorithms which, given a polynomial equation with integer coefficients and without any multiple roots, uses exact (infinite precision) integer arithmetic, an idea by Lagrange (1767), and Vincent's theorem of 1836, to approximate the real roots of the polynomial equation to any degree of accuracy using continued fractions. Theoretical computing time bounds are developed for these algorithms and some empirical results are presented.

Algorithmen zur Approximation reeller Wurzeln von Polynomen mit Hilfe von Kettenbrüchen. Wir diskutieren eine Anzahl von Algorithmen, welche die reellen Wurzeln einer Polynomgleichung mit ganzzahligen Koeffizienten unter Verwendung von Kettenbrüchen mit jeder gewünschten Genauigkeit approximieren. Dabei benützen wir ganzzahlige, also exakte Arithmetik, eine Idee von Lagrange aus 1767 und die Theorie von Vincent aus 1836. Für die Algorithmen leiten wir theoretische Rechenzeitschranken her, ferner teilen wir empirische Ergebnisse mit.

1. Introduction

Recently, the process of isolating the real roots of a polynomial equation (or, simply, isolation) has been extensively studied [7], [3]. Isolation is the process of finding real, disjoint intervals such that each contains exactly one real root and every real root is contained in some interval. However, according to Fourier this is only the first (of the two) step involved in the computation of the real roots of a polynomial equation; the second step consists of approximating these roots to any desired degree of accuracy ε, that is, making the length of the isolating intervals less than or equal to ε. In the sequel, we shall restrict our attention to those approximation methods which have been implemented in computer algebra systems using exact (infinite precision) integer arithmetic algorithms. (For a survey of computer algebra systems see [15].) Moreover, we shall be dealing only with integral-coefficient polynomials.

Bisection is basically the only approximation method widely known and implemented in computer algebra systems (for a detailed description see [10]). In [10] it was shown that the bisection method will approximate one real root in time

$$0\left(n^2 L(h/\varepsilon)\ \left(L(eh/\varepsilon)\right)\left(L(eh\,|P|_\infty/\varepsilon)\right)\right) \tag{1.1}$$

0010-485X/83/0030/0063/$02.80

where n is the degree of the square-free polynomial P, h is the initial length of the interval, ε is the degree of accuracy (limit of approximation), $e=\max\{|a_1|, |a_2|, |b_1|, |b_2|\}$, where $(a_1/b_1, a_2/b_2]$ is the initial interval, and $|P|_\infty$ is the maximum coefficient in absolute value. Empirical results showed that bisection is actually a very slow method; its performance was later improved when it was combined with Newton's method [18].

Quite recently, extending previous work by the first author [3], we developed a method with polynomial computing time bound for the approximation of real roots using continued fractions. As we will see, using our method, one real root can be approximated in time

$$0\left(L\left(\frac{1}{\varepsilon}\right)(n^3 L(|P|_\infty)^3)\right). \tag{1.2}$$

From (1.2) it is obvious that, unlike bisection, our method does not depend on the length of the initial isolating interval. In the following sections we will study this method in detail. The necessary complexity notions as well as the computational model within which our algorithms are implemented can be found in [3] and ([1] p. 35). Here we simply remind the reader that for a given list $A=(a_1, a_2, \ldots, a_n)$ we define the following operations: invert (A) resulting in $A=(a_n, \ldots, a_1)$; prefix $\bar{a}_1, \ldots, \bar{a}_k$ to A, $k \geqq 1$, resulting in $A=(\bar{a}_1, \ldots, \bar{a}_k, a_1, \ldots, a_n)$; and advance $\bar{a}_1, \ldots, \bar{a}_k$ in A, $k \leqq n$, resulting in $\bar{a}_i$ pointing to a_i, $1 \leqq i \leqq k$ and $A=(a_{k+1}, \ldots, a_n)$.

2. Mathematical Background

The idea to approximate the real roots of a polynomial equation using continued fractions is due to Lagrange (1767) [11], whose objective was to develop a procedure free of the defects plaguing the well-known Newton's method of approximation. Lagrange's idea may be stated as follows (see also ([16] pp. 135–141) and [9] p. 223)): Suppose a root of the polynomial equation $P(x)=0$ lies between the consecutive integers a_1 and a_1+1; diminish the roots of the equation by a_1 (i.e. $x \leftarrow x+a_1$) and take the reciprocal equation $\left(\text{i.e. } x \leftarrow \frac{1}{x}\right)$. Find, *by trial*, a root of the last equation lying between a_2 and a_2+1, diminish the roots by a_2 and take the reciprocal equation. Proceed in this way. Then the continued fraction

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdot_{\cdot_{\cdot}}}}$$

approximates a root of the equation.

Clearly, Lagrange's idea has certain drawbacks. Notice that the partial quotients a_i are computed by trial, which means, that this computation is exponential in the length of the a_i's. Moreover, it should be observed that the procedure is straight forward if there is one, and only one, root between the consecutive integers a_i and

a_i+1. However, there was no proof, at the time, that if there are two or more roots within (a_i, a_i+1), the process will eventually separate them. This fact was proven in 1836 by the following

Theorem 2.1: *Let $P(x)=0$ be a polynomial equation of degree $n>1$, with rational coefficients and without multiple roots, and let $\Delta>0$ be the smallest distance between any two of its roots. Let m be the smallest index such that*

$$F_{m-1}\frac{\Delta}{2}>1 \quad \text{and} \quad F_{m-1}F_m\Delta>1+\frac{1}{\varepsilon_n} \tag{2.1}$$

where F_k is the k-th member of the Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, ... and

$$\varepsilon_n=\left(1+\frac{1}{n}\right)^{\frac{1}{n-1}}-1.$$

Then the transformation

$$x=a_1+\cfrac{1}{a_2+\cfrac{1}{\ddots+\cfrac{1}{a_m+\cfrac{1}{y}}}}, \tag{2.2}$$

(which is equivalent to the series of successive transformations of the form $x=a_i+\frac{1}{y}$, $i=1,2,\ldots m$) which arbitrary, positive, integral elements $a_1, a_2, \ldots, a_m$, transforms the equation $P(x)=0$ into the equation $\tilde{P}(y)=0$, which has not more than one sign variation in the sequence of its coefficients.

The proof of the above theorem is very long, and it is omitted since it can be found in the literature [5, 17 pp. 298 – 304] (see also [12]). The original form of Theorem 2.1 (that is, without specifying the quantity m) is due to Vincent alone [19, 7] and appeared in 1836; Uspensky [17] extended it in a somewhat erroneous manner, which was corrected in [2].

Theorem 2.1 can be used to isolate the real roots of a polynomial equation; from its statement we know that a transformation of the form (2.2) with arbitrary, positive integer elements $a_1, a_2, \ldots a_m$ transforms $P(x)=0$ into an equation $\tilde{P}(y)=0$ which has at most one sign variation. This transformation can also be written as

$$x=\frac{P_m y+P_{m-1}}{Q_m y+Q_{m-1}} \tag{2.3}$$

where P_k/Q_k is the k-th convergent to the continued fraction

$$a_1+\cfrac{1}{a_2+\ddots}$$

and as we recall

$$\begin{aligned} P_{k+1}&=a_{k+1}P_k+P_{k-1}\\ Q_{k+1}&=a_{k+1}Q_k+Q_{k-1}\end{aligned} \tag{2.4}$$

with

$$P_kQ_{k-1}-P_{k-1}Q_k=(-1)^k. \tag{2.5}$$

Since the elements $a_1, a_2, \ldots, a_m$ are arbitrary, there is obviously an infinite number of transformations of the form (2.2). However, with the help of Budan's theorem [4] we can easily determine those that are of interest to us; namely, there is a finite number of them (equal to the number of the positive roots of $P(x)=0$) which lead to an equation with exactly one sign variation in the sequence of its coefficients. Suppose $\tilde{P}(y)=0$ is one of those equations; then from the Cardano-Descartes rule of signs we know that it has one root in the interval $(0, \infty)$; if $\hat{y}$ was this positive root then the corresponding root $\hat{x}$ of $P(x)=0$ could be easily obtained from (2.3). We only know though that $\hat{y}$ lies in the interval $(0, \infty)$; therefore, substituting y in (2.3) once by 0 and once by ∞ we obtain for the positive root $\hat{x}$ its isolating interval whose unordered endpoints are P_{m-1}/Q_{m-1} and P_m/Q_m. In this fashion we can isolate all the positive roots of $P(x)=0$. If we subsequently replace x by $-x$ in the original equation, the negative roots become positive and, hence, they too can be isolated in the way mentioned above.

The calculation of the quantities $a_1, a_2, \ldots, a_m$ for the transformations of the form (2.2) – which leads to an equation with exactly one sign variation – constitutes the polynomial real root isolation procedure. There are two methods, Vincent's and the one due to the first author, corresponding to the two different ways in which the computation of the a_i's may be performed.

Vincent's method basically consists of computing a particular a_i by a series of unit incrementations (by trial); that is $a_i \leftarrow a_i + 1$, which corresponds to the substitution $x \leftarrow x+1$. This brute force approach results in a method which will behave exponentially when the values of the a_i's are big. Examples of this approach can be found in [19] and in [17].

On the contrary, the method due to the first author consists of immediately computing a particular a_i as the lower bound on the values of the positive roots of a polynomial; that is $a_i \leftarrow b$, which corresponds to the substitution $x \leftarrow x+b$ performed on the particular polynomial under consideration. This method is obviously independent of how big the values of the a_i's are. An unsuccessful treatment of the big values of the a_i's can be found in ([17] p. 136). (We assume that $b=\lfloor \alpha_s \rfloor$, where α_s is the smallest positive root.) The lower bound b on the values of the positive roots is computed with the help of the following rule ([14] pp. 50–51); notice that we are computing the upper bound on the values of the positive roots of $P\left(\frac{1}{x}\right)=0$.

Theorem 2.2 (*Cauchy's Rule*): *Let*

$$P(x)=x^n+c_{n-1}x^{n-1}+\ldots+c_1x+c_0=0$$

be an integral-coefficient, monic polynomial equation of positive degree n, and let λ be the number of its negative coefficients. Then

$$b=\max_{\substack{1\leq k\leq n\\ c_{n-k}<0}} |\lambda c_{n-k}|^{\frac{1}{k}}$$

is an upper bound on the values of the positive roots of $P(x)=0$.

Proof: From the way b is defined we conclude that

$$b^k \geqq \lambda |c_{n-k}|$$

for each k such that $c_{n-k}<0$; for these k's the last inequality is also written as

$$b^n \geqq \lambda |c_{n-k}| b^{n-k}.$$

Summing over all the appropriate k's we obtain

$$\lambda b^n \geqq \sum_{\substack{k=1 \\ c_{n-k}<0}}^{n} |c_{n-k}| b^{n-k}$$

or

$$b^n \geqq \sum_{\substack{k=1 \\ c_{n-k}<0}}^{n} |c_{n-k}| b^{n-k}.$$

From the last inequality we conclude that, if we substitute b for x in $P(x)=0$, the first term, i.e. b^n, will be greater than or equal to the sum of the absolute values of all the negative coefficients. Therefore, $P(x)>0$ for all $x>b$. //

Cauchy's rule has been implemented using exact integer arithmetic and it has been shown that its computing time is [6]

$$0(n^2 L(|P|_\infty)). \tag{2.6}$$

Pursuing studies in the direction outlined above, it was observed that Theorem 2.1 can be also used to approximate the real roots to any desired degree of accuracy. This is easily achieved by extending (computing more partial quotients of) the continued fraction (2.2) which transforms the original polynomial equation into one with exactly one sign variation in the sequence of its coefficients. Notice that now the approximation method depends heavily on the isolation process; that is, it cannot work if it is provided only with the isolating intervals of the roots.

Suppose that the limit of approximation is ε, and that we have computed k partial quotients. Then, from the preceeding discussion it becomes obvious that the root lies between the consecutive convergents

$$\frac{P_{k-1}}{Q_{k-1}}, \frac{P_k}{Q_k}$$

(obtained from (2.4)) whose difference in absolute value is

$$\frac{1}{Q_{k-1} Q_k}.$$

(Use (2.5) to obtain the difference.) Hence if x is the root we are approximating, we have

$$\left|\frac{P_k}{Q_k} - x\right| \leqq \frac{1}{Q_{k-1} Q_k} \leqq \frac{1}{Q_{k-1}^2}$$

and the method will terminate when

$$\frac{1}{Q_k^2} \leqq \varepsilon \tag{2.7}$$

for some k.

In what follows we describe two ways for extending the continued fraction (2.2), in order to approximate a real root. These two methods have the same theoretical computing time bound, but different empirical performance.

The first way to extend the continued fraction (2.2) is to compute each additional partial quotient with the help of Cauchy's rule (see Fig. 2.1). However, mainly due to Cauchy's rule, this approach is inefficient as can be seen from Table 4.2 (at the end of this paper). Actually, it is even slower than the bisection method [10], a method well-known for its slowness.

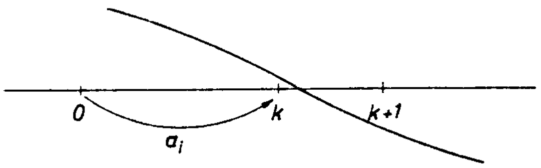

Fig. 2.1. Here k is computed with the help of Cauchy's rule

Trying to improve the empirical performance of our approximation method, we observed the special nature of the polynomials whose lower bounds b we are computing. These polynomials are special in the sense that they have one sign variation (and, hence, only one positive root) and consequently, they cross the x-axis only once. We then proceeded to compute each additional partial quotient of (2.2) by successively bisecting (and evaluating at midpoints) the interval $(0, b)$, where b is an easily computed upper bound on the value of the positive root (see Fig. 2.2 and Table 4.2 for the improvement).

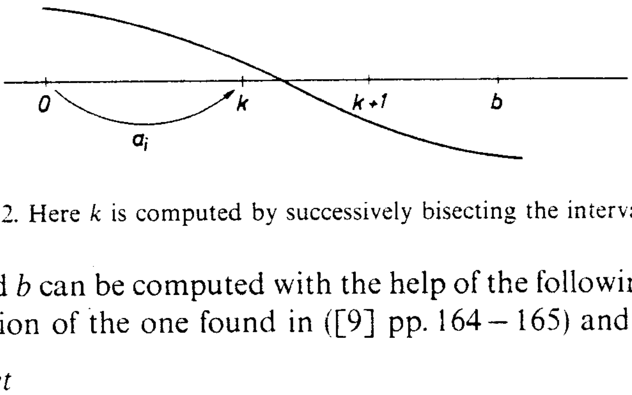

Fig. 2.2. Here k is computed by successively bisecting the interval $(0, b)$

This upper bound b can be computed with the help of the following theorem, which is a modern version of the one found in ([9] pp. 164–165) and ([16] p. 58).

Theorem 2.3: *Let*

$$P(x) = c_n x^n + c_{n-1} x^{n-1} \dots + c_0 = 0$$

be an integral coefficient polynomial equation of degree n. Then an upper bound on the values of its positive roots is given by

$$b = \max_{\substack{0 \leq r \leq n \\ c_r < 0}} \left(\frac{|c_r|}{\sum\limits_{\substack{i=r+1 \\ c_i > 0}}^{n} c_i} \right) + 1 \tag{2.8}$$

Theorem 2.3 is not actually used in our approximation method because it does not take advantage of the special nature of the polynomials discussed above. Instead we use Corollary 2.1 (which is much simpler to implement than Cauchy's rule).

Corollary 2.1: *Let*

$$P(x) = c_n x^n + \ldots + c_{r-1} x^{r+1} - c_r x^r - \ldots - c_0 = 0$$

be an integral-coefficient polynomial equation of degree n, with only one sign variation in the sequence of its coefficients. Then an upper bound on its (only one) positive root is given by

$$b = \frac{\max\limits_{0 \leqq j \leqq r} (|c_j|)}{\sum\limits_{i=r+1}^{n} c_i} + 1. \tag{2.9}$$

Following is a brief description of our approach:

Let $\tilde{P}_c(x) = 0$ be an integral-coefficient polynomial equation of degree n with one sign variation in the sequence of its coefficients. $\tilde{P}_c(x) = 0$ is obtained from an original equation $P(x) = 0$ after a continued fraction transformation of the form (2.2). Let P_c/Q_c be the convergent to the continued fraction from which $\tilde{P}_c(x) = 0$ is derived and P_0/Q_0 the immediately preceeding one. From the previous discussion it is clear that $\tilde{P}_c(x) = 0$ isolates one real root of $P(x) = 0$, and we are going to approximate this root to within ε. Obviously, if $1/Q_o^2 \leqq \varepsilon$ then we have nothing to do (see also (2.7)), whereas, otherwise we proceed as follows:

We first compute the lower bound b on the value of the positive root $\tilde{P}_c(x) = 0$ (it is assumed that $b = \lfloor \alpha \rfloor$, where α is the positive root; $b = a_i$ for some i), and then obtain $\tilde{P}_{\text{new}}(x) = \tilde{P}_c(x+b) = 0$ and the new convergent $P_{\text{new}}/Q_{\text{new}}$ using (2.4) and the pair P_c/Q_c, P_0/Q_0. If $\tilde{P}_{\text{new}}(0) = 0$ we have computed the root exactly and we terminate, whereas if $1/Q_c^2 \leqq \varepsilon$ we return an interval whose endpoints, $P_{\text{new}}/Q_{\text{new}}$ and P_c/Q_c, approximate the root to within the specified degree of accuracy ε. Otherwise, obtain $\tilde{P}'_{\text{new}}(x) = \tilde{P}_{\text{new}}\left(\dfrac{1}{x}\right) = 0$, update $\tilde{P}_c(x)$, P_0/Q_0, P_c/Q_c by $\tilde{P}'_{\text{new}}(x)$, P_c/Q_c, $P_{\text{new}}/Q_{\text{new}}$ respectively, and repeat the procedure.

Obviously, this procedure will approximate only the positive roots. For the negative roots we have to proceed in a similar fashion (after, of course, we substitute x by $-x$).

3. Computer Implementation of Our Approximation Method

We now present the algorithms which implement the main operation of our method. Short, informal descriptions are given for the various auxilliary algorithms; these are subsequently incorporated into one procedure which implements our root approximation method. From the discussion at the end of the previous section it is obvious that in order to keep track of the performed transformation, we have to associate with each polynomial equation, the function which corresponds to the transformation. The original polynomial equation is associated with the identity transformation $M(y) = y$. The substitutions performed on a polynomial equation are also performed on the corresponding function. We begin with the following (n refers always to the degree of a polynomial):

(i) *Computation of the Polynomial with One Sign Variation*

This is obtained using $\tilde{R} \leftarrow$ AMPRRA (P, N, D, F) which is a slight modification of algorithm $R \leftarrow$ AMPRRI(P, N, D) ([3] pp. 308 – 309). The inputs to AMPRRA are P, N, D and F. $P \neq 0$ is an univariate polynomial with integer coefficients and without multiple roots; moreover, zero is not a root ($P(0) \neq 0$). N is the special polynomial x, while D is the constant polynomial 1; they are the numerator and the denominator, respectively, of the function associated with P. F is a FORTRAN integer. We distinguish two cases:

(a) If $F = 0$, the output $\tilde{R}$ is a list of isolating intervals of the positive roots of P. Each interval is represented by a triplet (A, B, L), where A and B are the endpoints of the interval and L is the list of the partial quotients.

(b) If $F = 1$, the output $\tilde{R}$ is a list of two sublists, i.e., $\tilde{R} = (R_1, R_2)$. The first sublist R_1 is a list of isolating intervals. Each interval is represented by a quadruplet (P, N, D, L), where P is a polynomial with only one sign variation in its sequence of coefficients; N and D form the associated function; L, again, is the list of the partial quotients. The second sublist R_2 is a list of one-point isolating intervals representing the exact roots of the original polynomials.

From ([3] p. 309) it is obvious that $t_{\text{AMPRRA}}(P, x, 1, F) = 0(p\,n^4\,L(|P|_x)^3)$ where p is the number of the positive roots of $P(x) = 0$.

(ii) *Computation of the Lower Root Bound b*

Instead of using Cauchy's rule, we now use Corollary 2.1 to obtain an upper bound $\tilde{b}$ on the positive root α and then bisect the interval $(0, \tilde{b})$ to find $b = \lfloor \alpha \rfloor$. This is feasible because the polynomial we are dealing with has one positive root. Algorithm $b \leftarrow$ PPLRBD(P) is used; P is an integral-coefficient polynomial with one sign variation in the sequence of its coefficients, the leading one of which is positive, and $P(0) \neq 0$. b is an integer. It was shown [13] that $t_{\text{PPLRBD}}(P) = 0(n^2\,L(|P|_x)^3)$.

(iii) *Polynomial Translation*

For our method we need algorithms for the substitutions $x \leftarrow b + x$, $b \geqq 1$, and $x \leftarrow \dfrac{1}{1+x}$. The latter, however, is equivalent to the pair of transformations $x \leftarrow \dfrac{1}{x}$ and $x \leftarrow 1 + x$, so that we actually need algorithms for the transformations $x \leftarrow b + x$, $b \geqq 1$, and $x \leftarrow \dfrac{1}{x}$. In order to perform the substitution $x \leftarrow b + x$, $b \geqq 1$, we apply the Ruffini-Horner method and we consider two cases: if $b > 1$, we use algorithm $P' \leftarrow$ PTRNSL(P, B), where P is a polynomial, B the integer > 1, and $P'(x) = P(B + x)$; if $b = 1$ we use algorithm $P' \leftarrow$ PTRAN1(P). We have shown [8] that $t_{\text{PTRNSL}}(P, B) = 0(n^3\,L(B)^2 + n^2\,L(B)\,L(|P|_x))$. For the implementation of $x \leftarrow \dfrac{1}{x}$ we use algorithm $P' \leftarrow$ PINVCF(P), where P and P' are polynomials such that

$P'(x)=P\left(\frac{1}{x}\right)=0$. It is easily seen that $t_{\mathrm{PINVCF}}(P)\sim n$. Obviously, by applying the above mentioned algorithms separately on the numerator and denominator of the function associated with each polynomial equation, we can also transform the function by the previous substitutions. However, we can be more efficient by taking advantage of the fact that the numerator and denominator are both first degree (at most) polynomials. Therefore, we use algorithms $P'\leftarrow\mathrm{TUP}(P,B)$ and $P'\leftarrow\mathrm{TUP1}(P,F)$. With the first algorithm we obtain – from the polynomial $P(x)=c_1x+c_0$ – the transformed polynomial $P'(x)=P(B+x)=c_1x+(c_0+c_1B)$. With the second algorithm we have three options: if $F=0$ then, for the same polynomial P as above, we obtain $P'(x)=P(1+x)=c_1x+(c_0+c_1)$; if $F=1$ then $P'(x)=c_0x+(c_0+c_1)$ which corresponds to the equation $P'(x)=P\left(\frac{1}{1+x}\right)=0$, and if $F=2$ then $P'(x)=P\left(\frac{1}{x}\right)=c_0x+c_1$ (the second option is used in [3]).

(iv) *Formation of the Isolating Interval*

The isolating interval for a real root is obtained from a function of the form

$$M(x)=\frac{N(x)}{D(x)}=\frac{a_1x+a_0}{b_1x+b_0}$$

where a_0, b_0, a_1, b_1 are nonnegative integers such that $a_1+b_1>0$ and $a_0b_0>0$, if we replace x first by 0 and then by ∞. In fact, we can easily see that the isolating interval will be $\left(\frac{a_1}{b_1},\frac{a_0}{b_0}\right)$ or $\left(\frac{a_0}{b_0},\frac{a_1}{b_1}\right)$; ∞ is represented by -1. Algorithm $I\leftarrow\mathrm{FII}(N,D,F)$ is used, where if $F=1$ then I is an one-point interval, whereas, if $F=0$ we obtain one of the above mentioned intervals.

(v) *Approximation of a Real Root*

The following algorithm incorporates the previous algorithms into one procedure to approximate the given real root. The algorithm takes (P, N, D, EPS) as inputs where P is an univariate polynomial with integer coefficients which present only one sign variation (hence it has only one positive root), and $P(0)\neq0$; N and D are the numerator and denominator respectively of the associated function; EPS is the tolerance of the approximation to be achieved.

At first the floor function b of the root is computed using the algorithm PPLRBD(P). Then the polynomial P and the associated functions N, D are transformed by the substitution $x\leftarrow x+b$ into (P',N',D'). (P,N,D) is discarded. A check of the polynomial is made to determine whether an exact root has been reached or the tolerance has been achieved. If $P'(0)=0$, then an exact root has been reached and the one-point interval is obtained by calling $\mathrm{FII}(N',D',1)$; the algorithm is thus terminated. Otherwise, a test of tolerance is made, by obtaining the interval using $\mathrm{FII}(N',D',0)$, and comparing its length with EPS. If it is equal to or less than EPS, then the interval is saved for output and the algorithm is terminated. Otherwise (P', N', D') is transformed by the substitution $x \leftarrow \frac{1}{x}$ into (P, N, D). The triple (P', N', D') is then discarded. The algorithm is repeated until the requirements are satisfied. In the cource of our computation we also save the list of partial quotients and return this through a common statement.

$R \leftarrow$ APPROX$(P_0, N_0, D_0, \text{EPS})$: *Approx*imation of one real root.

Specifications: The inputs are P_0, N_0, D_0, EPS. $P_0 \neq 0$ is a univariate polynomial with integer coefficients represented by the list $P_0 = (C_r, e_r, C_{r-1}, e_{r-1}, \ldots, C_1, e_1)$, $r \geqq 1$, where each coefficient C_i is represented by the list $C_i = (c_{i1}, \ldots, c_{im_i})$, $m \geqq 1$. Moreover, the polynomial presents exactly one sign variation, and hence has one positive root. N_0 and D_0 form the associated functions, i.e. they are the numerator and denominator respectively of the continued fraction. They are both represented in the same way as P_0. EPS is the limit of the tolerance of the approximation to be achieved. It is represented by a rational number, i.e. EPS $= (C_1, C_2)$, where each integer C_i is represented by a list. The output is R representing the approximating interval; $R = (R_1, R_2)$ where R_1 and R_2 are rational numbers represented in the same way as EPS. This algorithm also saves a list PQ of the partial quotients of the continued fraction corresponding to the approximation. This is returned through a common statement.

Description:

```
1. [Initialize.] $P \leftarrow P_0$; $N \leftarrow N_0$; $D \leftarrow D_0$; PQ $\leftarrow 0$.
2. [Obtain lower bound.] $b \leftarrow$ PPLRBD($P$); prefix $b$ to PQ;
   if $b = 1$ then go to 4.
3. [$x \leftarrow x + b$.] $P' \leftarrow$ PTRNSL($P, b$); $N' \leftarrow$ TUP($N, b$); $D' \leftarrow$ TUP($D, b$); go to 5.
4. [$x \leftarrow x + 1$.] $P' \leftarrow$ PTRAN1($P$); $N' \leftarrow$ TUP1($N$,0); $D' \leftarrow$ TUP1(D,0).
5. [Update.] $P \leftarrow P'$; $N \leftarrow N'$; $D \leftarrow D'$.
6. [Check root.] $s \leftarrow P(0)$; if $s \neq 0$ then go to 7;
   IN $\leftarrow$ FII($N, D$, 1); go to 9.
7. [Check limit.] IN $\leftarrow$ FII($N, D$, 0); advance $L, R$ in IN;
   if $(R - L) \leqq$ EPS then go to 9.
8. [$x \leftarrow 1/x$.] $P' \leftarrow$ PINVCF($P$); if sign($P'$) $< 0$ then $P' \leftarrow -P'$;
   $N' \leftarrow$ TUP1($N$,2); $D' \leftarrow$ TUP1($D$,2); $P \leftarrow P'$; $N \leftarrow N'$; $D \leftarrow D'$; go to 2.
9. [Finish up.] $R \leftarrow$ IN; PQ $\leftarrow$ invert(PQ); return.
```

Theorem 3.1: *Let P_0 be an univariate polynomial of degree $n \geqq 1$ with exactly one sign variation in the sequence of its coefficients and $P_0(0) \neq 0$. If the tolerance of the approximation to be achieved for the positive root of P is* EPS $= \varepsilon$, *then*

$$t_{\text{APPROX}}(P_0, N_0, D_0, \text{EPS}) = 0\left(L\left(\frac{1}{\varepsilon}\right)(n^3 L(|P|_\infty)^3)\right).$$

$P'(x)=P\left(\frac{1}{x}\right)=0$. It is easily seen that $t_{\mathrm{PINVCF}}(P)\sim n$. Obviously, by applying the above mentioned algorithms separately on the numerator and denominator of the function associated with each polynomial equation, we can also transform the function by the previous substitutions. However, we can be more efficient by taking advantage of the fact that the numerator and denominator are both first degree (at most) polynomials. Therefore, we use algorithms $P'\leftarrow\mathrm{TUP}(P,B)$ and $P'\leftarrow\mathrm{TUP1}(P,F)$. With the first algorithm we obtain – from the polynomial $P(x)=c_1x+c_0$ – the transformed polynomial $P'(x)=P(B+x)=c_1x+(c_0+c_1B)$. With the second algorithm we have three options: if $F=0$ then, for the same polynomial P as above, we obtain $P'(x)=P(1+x)=c_1x+(c_0+c_1)$; if $F=1$ then $P'(x)=c_0x+(c_0+c_1)$ which corresponds to the equation $P'(x)=P\left(\frac{1}{1+x}\right)=0$, and if $F=2$ then $P'(x)=P\left(\frac{1}{x}\right)=c_0x+c_1$ (the second option is used in [3]).

(iv) *Formation of the Isolating Interval*

The isolating interval for a real root is obtained from a function of the form

$$M(x)=\frac{N(x)}{D(x)}=\frac{a_1x+a_0}{b_1x+b_0}$$

where a_0, b_0, a_1, b_1 are nonnegative integers such that $a_1+b_1>0$ and $a_0b_0>0$, if we replace x first by 0 and then by ∞. In fact, we can easily see that the isolating interval will be $\left(\frac{a_1}{b_1},\frac{a_0}{b_0}\right)$ or $\left(\frac{a_0}{b_0},\frac{a_1}{b_1}\right)$; ∞ is represented by -1. Algorithm $I\leftarrow\mathrm{FII}(N,D,F)$ is used, where if $F=1$ then I is an one-point interval, whereas, if $F=0$ we obtain one of the above mentioned intervals.

(v) *Approximation of a Real Root*

The following algorithm incorporates the previous algorithms into one procedure to approximate the given real root. The algorithm takes (P, N, D, EPS) as inputs where P is an univariate polynomial with integer coefficients which present only one sign variation (hence it has only one positive root), and $P(0)\neq 0$; N and D are the numerator and denominator respectively of the associated function; EPS is the tolerance of the approximation to be achieved.

At first the floor function b of the root is computed using the algorithm $\mathrm{PPLRBD}(P)$. Then the polynomial P and the associated functions N, D are transformed by the substitution $x\leftarrow x+b$ into (P',N',D'). (P,N,D) is discarded. A check of the polynomial is made to determine whether an exact root has been reached or the tolerance has been achieved. If $P'(0)=0$, then an exact root has been reached and the one-point interval is obtained by calling $\mathrm{FII}(N',D',1)$; the algorithm is thus terminated. Otherwise, a test of tolerance is made, by obtaining the interval using $\mathrm{FII}(N',D',0)$, and comparing its length with EPS. If it is equal to or less than EPS, then the interval is saved for output and the algorithm is terminated. Otherwise (P', N', D') is transformed by the substitution $x \leftarrow \frac{1}{x}$ into (P, N, D). The triple (P', N', D') is then discarded. The algorithm is repeated until the requirements are satisfied. In the cource of our computation we also save the list of partial quotients and return this through a common statement.

$R \leftarrow$ APPROX$(P_0, N_0, D_0,$ EPS$)$: *Approx*imation of one real root.

Specifications: The inputs are P_0, N_0, D_0, EPS. $P_0 \neq 0$ is a univariate polynomial with integer coefficients represented by the list $P_0 = (C_r, e_r, C_{r-1}, e_{r-1}, \ldots, C_1, e_1)$, $r \geqq 1$, where each coefficient C_i is represented by the list $C_i = (c_{i1}, \ldots, c_{im_i})$, $m \geqq 1$. Moreover, the polynomial presents exactly one sign variation, and hence has one positive root. N_0 and D_0 form the associated functions, i.e. they are the numerator and denominator respectively of the continued fraction. They are both represented in the same way as P_0. EPS is the limit of the tolerance of the approximation to be achieved. It is represented by a rational number, i.e. EPS $= (C_1, C_2)$, where each integer C_i is represented by a list. The output is R representing the approximating interval; $R = (R_1, R_2)$ where R_1 and R_2 are rational numbers represented in the same way as EPS. This algorithm also saves a list PQ of the partial quotients of the continued fraction corresponding to the approximation. This is returned through a common statement.

Description:

```
1. [Initialize.] $P \leftarrow P_0$; $N \leftarrow N_0$; $D \leftarrow D_0$; PQ $\leftarrow 0$.
2. [Obtain lower bound.] $b \leftarrow$ PPLRBD$(P)$; prefix $b$ **to** PQ;
   **if** $b = 1$ **then go to** 4.
3. [$x \leftarrow x + b$.] $P' \leftarrow$ PTRNSL$(P, b)$; $N' \leftarrow$ TUP$(N, b)$: $D' \leftarrow$ TUP$(D, b)$; **go to** 5.
4. [$x \leftarrow x + 1$.] $P' \leftarrow$ PTRAN1$(P)$; $N' \leftarrow$ TUP1$(N, 0)$; $D' \leftarrow$ TUP1$(D, 0)$.
5. [Update.] $P \leftarrow P'$; $N \leftarrow N'$; $D \leftarrow D'$.
6. [Check root.] $s \leftarrow P(0)$; **if** $s \neq 0$ **then go to** 7;
   IN $\leftarrow$ FII$(N, D, 1)$; **go to** 9.
7. [Check limit.] IN $\leftarrow$ FII$(N, D, 0)$; advance $L, R$ **in** IN;
   **if** $(R - L) \leqq$ EPS **then go to** 9.
8. [$x \leftarrow 1/x$.] $P' \leftarrow$ PINVCF$(P)$; **if** sign$(P') < 0$ **then** $P' \leftarrow -P'$;
   $N' \leftarrow$ TUP1$(N, 2)$; $D' \leftarrow$ TUP1$(D, 2)$; $P \leftarrow P'$; $N \leftarrow N'$; $D \leftarrow D'$; **go to** 2.
9. [Finish up.] $R \leftarrow$ IN; PQ $\leftarrow$ invert(PQ); **return.**
```

Theorem 3.1: *Let P_0 be an univariate polynomial of degree $n \geqq 1$ with exactly one sign variation in the sequence of its coefficients and $P_0(0) \neq 0$. If the tolerance of the approximation to be achieved for the positive root of P is EPS $= \varepsilon$, then*

$$t_{\text{APPROX}}(P_0, N_0, D_0, \text{EPS}) = 0\left(L\left(\frac{1}{\varepsilon}\right)(n^3 L(|P|_\infty)^3)\right).$$

Proof: Step 2 is computed in time $0(n^2 L(|P|_\infty)^3)$ while step 3 is computed in time $0(n^3 L(b)^2 + n^2 L(b) L(|P|_\infty))$. Since $b = 0(|P|_\infty)$ [3] the total computing time for step 2 and 3 is $0(n^2 L(|P|_\infty)^3 + n^3 L(|P|_\infty)^2)$ which dominates the whole procedure. The number of iterations m is obtained if we use the inequality

$$\frac{1}{Q_m^2} \leqq \varepsilon. \tag{3.1}$$

Recall that $Q_m \geqq F_m$ where F_m is the m-th member in the Fibonacci sequence, and that $F_m = \frac{\phi^m}{\sqrt{5}}$, $\phi = 1.618\ldots$.

Hence inequality (3.1) will also be satisfied if

$$\frac{5}{\phi^{2m}} \leqq \varepsilon.$$

From the latter we clearly see that

$$m = 0\left(\log_\phi \frac{1}{\varepsilon}\right) = 0\left(L\left(\frac{1}{\varepsilon}\right)\right).$$

Therefore the time for the whole procedure is

$$t_{\text{APPROX}}(P_0, N_0, D_0, \text{EPS}) = 0\left(L\left(\frac{1}{\varepsilon}\right)(n^2 L(|P|_\infty)^3 + n^3 L(|P|_\infty)^2)\right)$$

$$\sim 0\left(L\left(\frac{1}{\varepsilon}\right)(n^3 L(|P|_\infty)^3)\right). \quad //$$

In order to approximate all the real roots of a polynomial equation, we can easily incorporate the last algorithm into a more general procedure.

4. Empirical Results and Conclusions

In this section we present several tables comparing our continued fractions method with the bisection method. We compare both theoretical aspects as well as the actual computing times for Chebyshev polynomials. We first find out the number of partial quotients and bisections respectively, needed for each method under consideration, in order to approximate a root to within a specified degree of accuracy. Under the assumption that the original polynomial has only one positive root we can take $(0, \infty)$ as the initial interval for the continued fractions method; we take $(0, 1)$ as the initial interval for the bisection method. Moreover, for the continued fractions method we assume the worst possible case; that is, each partial quotient is 1, in which case we have $Q_m = F_m$ (see equation (2.1)) where F_m is the m-th member of the Fibonacci sequence. Under the above conditions, we can see from Table 4.1 that it takes more bisections than partial quotients in order to achieve the same degree of accuracy.

Table 4.1. *Comparison of the number of partial quotients and bisections needed to obtain the required degree of accuracy*

m	Continued Fraction $1/F_m^2$	Bisection $1/2^m$
1	1	0.5
5	0.04	0.03125
10	3.3×10^{-4}	9.7×10^{-4}
15	2.7×10^{-6}	3.1×10^{-5}
20	2.2×10^{-8}	9.5×10^{-7}
25	1.7×10^{-10}	1.9×10^{-8}
30	1.4×10^{-12}	9.3×10^{-10}
35	3.1×10^{-14}	1.5×10^{-11}
40	9.5×10^{-17}	2.3×10^{-13}

We now present a table showing the computing times for Chebyshev polynomials using our method and the bisection method. All the times are in seconds and were obtained using the SAC-1 (entirely implemented in FORTRAN) computer algebra system on the Honeywell 66/60 computer at the University of Kansas. The tolerance of the approximation is $\varepsilon = 10^{-15}$. We compare three versions of our approximation method; namely, the versions which utilize Cauchy's Rule, Corollary 2.1 with bisection, and preconditioning. In the last version, we assume that a list of partial quotients is supplied as input. In this way we spend no time in computing the floor functions. The result of this algorithm reflects the optimum time for the approximation of a real root using the continued fractions method. The difference between the versions using Corollary 2.1 with bisection and preconditioning reflects the time spent in computing the floor functions. We remind the reader that Cauchy's rule had to be applied a number of times in order to obtain the floor function of the root.

Table 4.2. *Computing times (in seconds) for the approximations of all the real roots of Chebyshev polynomials* ($\varepsilon = 10^{-15}$)

Degree	Bisection	Continued Fractions Using:		
		Cauchy's Rule	Corollary 2.1 with bisection	Preconditioning
2	17.2	11.5	6.7	5.4
3	17.9	10.3	4.9	3.8
4	42.3	38.7	15.7	10.3
5	45.8	40.0	16.4	10.8
6	83.1	99.8	46.2	29.2
7	90.9	105.1	44.6	27.0
8	146.3	277.8	93.0	50.2
9	170.6	257.6	106.2	62.2
10	243.2	524.3	202.8	116.2

Table 4.3. *Approximating the real roots of the Chebyshev polynomials of degrees* 2 – 10 ($\varepsilon = 10^{-15}$)

Degree	List of Partial Quotients for: Isolation	Approximation	Approximating Intervals
2	(0)	(0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)	0.7071067811865468338 0.70710678118654764296
3	(0)	(0, 1, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2)	0.86602540378443847925 0.86602540378443865879
4	(0, 1, 1)	(11, 7, 3, 2, 1, 1, 1, 1, 20, 5, 3, 11, 1, 7)	0.92387953251128634045 0.92387953251128676332
	(0, 2)	(0, 1, 1, 1, 1, 2, 2, 4, 3, 1, 19, 6, 8, 3, 2, 9)	0.38268343236508971655 0.38268343236509064218
5	(0, 1, 1, 1)	(1, 2, 1, 6, 1, 56, 1, 54, 1, 1, 1, 10, 1, 16)	0.58778525229247311679 0.58778525229247338805
	(0, 1, 2)	(17, 2, 3, 6, 5, 1, 1, 1, 3, 2, 1, 25, 2, 2, 1, 1)	0.95105651629515309991 0.95105651629515359558
6	(0, 1, 2, 1)	(1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)	0.70710678118654683338 0.70710678118654764296
	(0, 1, 3)	(25, 2, 1, 7, 21, 1, 8, 1, 3, 10, 1, 2, 3)	0.96592582628906811211 0.96592582628906894040
	(0, 2)	(1, 1, 6, 2, 1, 30, 5, 2, 9, 3, 1, 10, 5, 5)	0.25881904510252075804 0.25881904510252080914
7	(0, 1, 3, 1)	(0, 1, 2, 2, 24, 2, 3, 2, 2, 2, 2, 2, 8, 1, 8, 1)	0.78183148246802977984 0.78183148246803000835
	(0, 1, 4)	(34, 1, 7, 1, 2, 3, 1, 1, 2, 3, 1, 132, 2, 3)	0.97492791218182315610 0.97492791218182365484
	(0, 2)	(0, 3, 3, 1, 1, 3, 1, 29, 1, 3, 1, 18, 16, 1, 1, 2)	0.43388373911755805748 0.43388373911755880719
8	(0, 1, 1, 3, 1)	(0, 14, 14, 17, 1, 1, 10, 2, 1, 2, 18, 1)	0.83146961230254518379 0.83146961230254537502
	(0, 1, 1, 4)	(46, 23, 43, 8, 1, 2, 1, 3, 1, 15)	0.98078528040322959629 0.98078528040323045751
	(0, 1, 1, 1)	(2, 1, 840, 2, 1, 4, 1, 3, 21, 1, 17)	0.55557023301960177859 0.55557023301960224274
	(0, 2)	(3, 7, 1, 17, 1, 13, 3, 2, 7, 1, 1, 8, 1, 1, 1, 40)	0.19509032201612826749 0.19509032201612828909
9	(0, 1, 1, 5, 1)	(1, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2)	0.86602540378443847925 0.86602540378443865879
	(0, 1, 1, 6)	(57, 1, 4, 1, 1, 1, 6, 1, 1, 6, 10, 1, 1, 3, 1, 1, 2, 6)	0.98480775301220799222 0.98480775301220806644
	(0, 1, 1, 1)	(0, 3, 1, 72, 1, 1, 2, 3, 1, 1, 3, 14, 1, 1, 1, 16, 1)	0.64278760968653925445 0.64278760968653933737
	(0, 2)	(0, 1, 12, 8, 17, 1, 1, 2, 1, 1, 1, 49, 1, 20, 3)	0.34203014332566870745 0.34203014332566873472
10	(0, 1, 2, 6, 1, 1)	(3, 1, 14, 13, 1, 1, 10, 1, 72, 23, 1)	0.45399049973954674160 0.45399049973954687273
	(0, 1, 2, 6, 2)	(3, 2, 1, 1, 4, 1, 2, 2, 1, 35, 1, 2, 4, 1, 3, 4, 65)	0.15643446504023080945 0.15643446504023086913
	(0, 1, 2, 6, 1)	(4, 1, 2, 1, 1, 3, 1, 12, 5, 19, 11, 1, 1)	0.89100652418836719434 0.89100652418836801487
	(0, 1, 2, 7)	(71, 4, 2, 7, 3, 1, 1, 42, 1, 12, 18)	0.98768834059513772582 0.98768834059513783641
	(0, 1, 2, 1)	(1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)	0.70710678118654683338 0.70710678118654764297

These results show that the version using Corollary 2.1 with bisection is substantially better than the one using Cauchy's rule. On the other hand, comparing the results of the version using preconditioning, we note that there is still room for improvement. We believe that as the properties of the special polynomial equations with exactly one sign variation are better understood, they will result in improved algorithms.

As an illustration we present a table showing the derived list of partial quotients along with the approximating intervals for each root. (The tolerance of the approximation is $\varepsilon = 10^{-15}$.) Since the Chebyshev Polynomials are symmetric we only present the positive roots. For odd degree polynomials we omit the root $x = 0$.

References

[1] Aho, A. V., Hopcroft, J. E., Ullman, J. D.: The design and analysis of computer algorithms. Reading: Addison-Wesley 1976.

[2] Akritas, A. G.: A correction on a theorem by Uspensky. Bulletin of the Greek Mathematical Society *19*, 278 – 285 (1978).

[3] Akritas, A. G.: The fastest exact algorithms for the isolation of the real roots of a polynomial equation. Computing *24*, 299 – 313 (1980).

[4] Akritas, A. G.: On the Budan-Fourier controversy. ACM-SIGSAM Bulletin *15*, No. 1, 8 – 10 (1981).

[5] Akritas, A. G.: Vincent's forgotten theorem, its extension and application. International Journal of Computers and Mathematics with Applications *7*, 309 – 317 (1981).

[6] Akritas, A. G.: Exact algorithms for the implementation of Cauchy's rule. International Journal of Computer Mathematics *9*, 323 – 333 (1981).

[7] Akritas, A. G., Danielopoulos, S. D.: On the forgotten theorem of Mr. Vincent. Historia Mathematica *5*, 427 – 435 (1978).

[8] Akritas, A. G., Danielopoulos, S. D.: On the complexity of algorithms for the translation of polynomials. Computing *24*, 51 – 60 (1980).

[9] Burnside, W. S., Panton, A. W.: Theory of equations, 2nd ed. Dublin-London: Dublin University Press Series 1886.

[10] Heindel, L. E.: Integer arithmetic algorithms for polynomial real zero determination. Journal of the Association for Computing Machinery *18*, 533 – 548 (1971).

[11] Lagrange, J. L.: Traité de la Résolution des Equations Numériques. Paris: 1778.

[12] Mahler, K.: An inequality for the discriminant of a polynomial. Michigan Mathematical Journal *11*, 257 – 262 (1964).

[13] Ng, K. H.: Polynomial real root approximation using continued fractions. M. S. Research Report, University of Kansas, Department of Computer Science, Lawrence, Kansas (1980).

[14] Obreschkoff, N.: Verteilung und Berechnung der Nullstellen reeller Polynome. Berlin: VEB Deutscher Verlag der Wissenschaften 1963.

[15] Petricle, S. R. (ed.): Proceedings of the second symposium on symbolic and algebraic manipulation. ACM (1971).

[16] Todhunter, I.: Theory of equations. London: Macmillan 1882.

[17] Uspensky, J. V.: Theory of equations. New York: McGraw-Hill 1948.

[18] Verbaeten, P.: Computing real zeros of polynomials with SAC-1. ACM-SIGSAM Bulletin *9*, No. 2, 8 – 10 (1975).

[19] Vincent, A. J. H.: Sur la Résolution des Équations Numériques. Journal de Mathématiques Pures et Appliquées *1*, 341 – 371 (1836).

A. G. Akritas
University of Kansas
Department of Computer Science
Lawrence, KS 66045, U.S.A.

King H. Ng
Shell Oil Company
P. O. Box 991
Houston, TX 77001, U.S.A.

Printed in Austria

Druck: Paul Gerin, A-1021 Wien