4 Analysis

The algorithm works by binary search. The subproblems correspond to subintervals of $(0,1)$. Since the unit interval is repeatedly split in half, the subintervals have length 2^{-k} for integer k. For fixed k there are 2^k such subintervals which we denote

$$I_{c,k} = (\frac{c}{2^k}, \frac{c+1}{2^k})$$

with $0 \le c < 2^k$. The polynomial

$$P_{c,k} = 2^{kn} P(\frac{c+1}{2^k})$$

captures the roots of P in $I_{c,k}$. We use $Q_{c,k} = (1+x)^n P_{c,k}(\frac{1}{1+x})$ to denote the corresponding transformed polynomial.

Binary search explores a binary tree T. We use (c,k) to denote the node of the tree corresponding to subinterval $I_{c,k}$. Node (c,k) has children $(2c, k+1)$ and $(2c+1, k+1)$ and parent $(\lfloor c/2 \rfloor, k-1)$. The parent exists if $k \ge 1$. A node represents an interval $I_{c,k}$ and also the splitting value $\frac{2c+1}{2^{k+1}}$.

The algorithm explores a node (c,k) if $V(Q_{c',k'}) \ge 2$ for all proper ancestors (c',k') of (c,k). A node (c,k) is a leaf if it is explored and $V(Q_{c,k}) \le 1$. A node (c,k) is internal if it is explored and $V(Q_{c,k}) \ge 2$.

Every root z of P in $I_{0,0}$ defines a path p_z in T, namely, the search path for $\Re z$. A non-real root and its conjugate define the same path. The path is finite if the real part of z is a fraction whose denominator is a power of two. Otherwise, the path is infinite.

Lemma 5 *If (c,k) does not lie on one of the paths defined by one of the roots of P, the node (c,k) is not internal.*

Proof: . The strip $I_{c,k} + i \cdot \mathbb{R}$ does not contain a root of P. Hence $V(Q_{c,k}) = 0$ by Lemma 1. ∎

Let

$$\Delta = \min \{ |z' - z''| \; ; \; P(z') = P(z'') = 0 \text{ and } z' \ne z'' \} \; .$$

Lemma 6 *If $2^k \ge 2/\Delta$ then (c,k) is not an internal node.*

Proof: Clearly, the interval $I_{c,k}$ contains at most one real root of P. We also observe that $\Delta(P_{c,k}) = 2^k \Delta \ge 2$. We now distinguish cases.

If $I_{c,k}$ contains no real root of P, the disk defined in Lemma 1 contains no root of $P_{c,k}$. Otherwise, it would have to contain two roots, namely a non-real root and its conjugate, and these roots would have distance less than one. Thus $V(Q_{c,k}) = 0$ by Lemma 1.

If $I_{c,k}$ contains a real root, the disks defined in Lemma 2 contain no non-real root of $P_{c,k}$. Otherwise, they would have to contain two roots, namely a non-real root and its conjugate, and these roots would have distance less than two. Thus $V(Q_{c,k}) = 1$ by Lemma 2. ∎

Let us summarize what we know at this point. All internal nodes lie on n paths. These paths are explored up to depth at most $k_{max} := 1 + \log 1/\Delta$ and hence the number of internal nodes is bounded by $n(1 + \log 1/\Delta)$. We next need to discuss how much it costs to explore a node. We split this question into two parts. How large can the coefficients of $P_{c,k}$ be and how difficult is it to generate $Q_{c,k}$, $P_{2c,k+1}$ and $P_{2c+1,k+1}$ from $P_{c,k}$?

We define the *size* of an integer a to be the smallest integer t such that $|a| < 2^t$.

Lemma 7 *Assume that the size of the coefficients of P is bounded by t. Then the size of the coefficients of $P_{c,k}$ is bounded by $1 + t + (k+1)n = O(t + kn)$*

Proof: Let $S_k(x) = 2^{kn}P(x/2^k)$. Then $P_{c,k}(x) = S_k(c+x)$. We first bound the coefficient size of $S_k(x)$. We have

$$S_k(x) = \sum_i b_i x^i \quad \text{where} \quad b_i = a_i 2^{k(n-i)} \ .$$

The size of b_i is therefore bounded by $t + k(n-i)$. Let us next consider $P_{c,k} = \sum t_j x^j$. We have

$$\begin{aligned} P_{c,k}(x) &= \sum_i b_i (x+c)^i \\ &= \sum_i b_i \sum_{j \le i} \binom{i}{j} x^j c^{i-j} \\ &= \sum_j \left(\sum_{i \ge j} b_i \binom{i}{j} c^{i-j} \right) x^j \end{aligned}$$

Thus (using $c < 2^k$)

$$\begin{aligned} t_j &\le \sum_{i \ge j} a_i 2^{k(n-i)} \binom{i}{j} 2^{k(i-j)} \\ &\le 2^t 2^{k(n-j)} \sum_{i \ge j} \binom{i}{j} \\ &\le 2^t 2^{k(n-j)} \binom{n+1}{j+1} \\ &\le 2^t 2^{k(n-j)} 2^{n+1} \end{aligned}$$

where the third line uses the fact that $\sum_{i\leq n}\binom{i}{j}=\binom{n+1}{j+1}$. This well-known identity is easily shown by induction[1] on n. The last line uses the fact that $\binom{n+1}{j+1}\leq(1+1)^{n+1}=2^{n+1}$.

The size of the coefficients of $P_{c,k}$ is therefore bounded by $t+1+(k+1)n=O(t+kn)$. ∎

We next study the cost of computing the polynomials $Q_{c,k}$, $P_{2c,k+1}$ and $P_{2c+1,k+1}$ from $P_{c,k}$?

Lemma 8 *$Q_{c,k}$, $P_{2c,k+1}$ and $P_{2c+1,k+1}$ can be computed from $P_{c,k}$ by $O(n^2)$ additions and shifts.*

Proof: We leave $Q_{c,k}$ to the reader and concentrate on the polynomials associated with the children of (c,k). The coefficients of the polynomial associated with the left child are obtained by shifts. For the right child, we observe that $P_{2c+1,k+1}=P_{2c,k+1}(x+1)$. It therefore suffices the study the problem of shifting the argument by one.

For brevity, write $R(x)=P_{2c,k+1}(x)=\sum_{0\leq i\leq n}b_ix_i$ and define $h_j(x)=\sum_{i\geq j}b_i(x+1)^{i-j}$ for all j, $0\leq j\leq n$. Then $h_0(x)=R(x+1)$. Also $h_n(x)=b_n$ and for $j<n$ we have

$$\begin{aligned}h_j(x)&=\sum_{i\geq j}b_i(x+1)^{i-j}\\&=(x+1)\sum_{i>j}b_i(x+1)^{i-j-1}+b_j\\&=xh_{j+1}(x)+h_{j+1}(x)+b_j\end{aligned}$$

We can therefore compute h_j from h_{j+1} by n additions and hence h_0 by $O(n^2)$ additions. ∎

It remains to assemble the pieces. The cost to explore a node (c,k) is $O(n^2(t+kn))$. At each depth we explore at most n nodes and the maximal depth is k_{max}. Thus the total cost is bounded by $O(n^3tk_{max}+n^4k_{max}^2)$.

Theorem 3 *Let $|P|_1=\sum|a_i|\leq 2^{\log n+t}$. Then*

$$\Delta\geq\sqrt{3}\frac{1}{n^{(n+1)/2}}\frac{1}{|P|_1^{n-1}}$$

and hence

$$\log 1/\Delta=O(n(\log n+t))\ .$$

[1]For $n<j$ the sum is zero and the right hand side is zero. For $n\geq j$, we split the left summand from the sum and apply induction hypothesis. Thus

$$\sum_{i\leq n}\binom{i}{j}=\sum_{i<n}\binom{i}{j}+\binom{n}{j}=\binom{n}{j+1}+\binom{n}{j}=\frac{n!}{(j+1)!(n-j-1)!}+\frac{n!}{j!(n-j)!}=\frac{n!(n-j+j+1)}{(j+1)!(n-j)!}=\binom{n+1}{j+1}\ .$$

Proof: Proofs can be found in the books of Yap [Yap99] and Mignotte [Mig92]. ∎

Theorem 4 *The cost of Vincent's algorithm is*

$$O(n^3 t k_{max} + n^4 k_{max}^2) = O(n^6(t + \log n)^2)$$

where $k_{max} = 1 + \log 1/\Delta = O(n(t + \log n))$.

Remarks: The running time grows like $O(n^6)$ and hence the algorithms is only suitable for small n. Observe however that the running time estimate is a worst case bound.

Can one do an average case analysis? What are we averaging over? What is a random polynomial. Discuss this with Cyril.

One could do a more careful analysis. Observe that the path for a particular root z is followed to depth at most $\log \Delta_z$ where Δ_z is the minimum separation of z from any other root. For a non-real root, we might even stop earlier, if the root is far away from the real axis.

Let us consider a specific example. Wilkinson introduced the polynomials P_n with roots $1/i$ for $2 \le i \le n+1$. Then $P_n = \prod_{2 \le i \le n+1}(ix - 1)$ and hence the lead coefficient is $(n+1)! \approx (n/e)^n$. Thus $t = O(n \log n)$. Also $\Delta = O(1/n^2)$ and hence $k_{max} = O(\log n)$. We conclude that the running time is $O(n^4 \log^2 n)$. I implemented the algorithm. For $n = 100$, the running time is 0.98 sec, for $n = 200$, the running times was 13.38 sec. For $n = 400$, the running time is 242.47 seconds.

Rouillier and Zimmermann [RZ01] describe a very interesting optimization which substantially decreases the space requirement of the algorithm without increasing the running time. We explore the tree of subproblems by depth-first search. This requires to store k_{max} intermediate polynomials. Rouillier and Zimmermann cut this down to a single polynomial as follows. Assume that we have just finished the subproblem (c, k). If (c, k) is a left child, the next problem is $(c + 1, k)$. Also $P_{c+1,k}(x) = P_{c,k}(x + 1)$ and hence the required polynomial is easily computed. Assume next that (c, k) is a right child. Let (c', k') be the nearest ancestor of (c, k) which is a left child. Then the next subproblem is $(c' + 1, k')$.

It is easy to determine c' and k'. We have $k' = k - \ell$ where ℓ is the number of trailing ones in the binary representation of c. c' ends with a zero. There is a different way to obtain $c' + 1$. Add 1 to c and remove the trailing zeroes. Thus $c + 1 = (c' + 1)2^\ell$ and hence $P_{c+1,k}(x) = 2^{\ell n} P_{c'+1,k'}(x/2^\ell)$.

We obtain $P_{c'+1,k'}$ from $P_{c,k}$ as follows. We first compute $P_{c+1,k}(x) = P_{c,k}(x+1)$. Then $P_{c'+1,k'}(x) = 2^{-\ell n} P_{c+1,k}(2^\ell x)$ which amounts removing trailing zeroes from the coefficients.

What saving in space can we expect? Recall that the size of the coefficients of $P_{c,k}$ is $O(t + kn)$ and hence the space requirement for the polynomial is $O(tn + kn^2)$. When the tree is explored depth-first, the maximal space requirement is $O(tnk_{max} + n^2 k_{max}^2)$, when the tree is explored breadth-first, the maximal requirement is $O(tn^2 + k_{max} n^3)$. With Rouillier and Zimmermann's scheme the requirement is $O(tn + n^2 k_{max})$. For large k_{max}, the savings will be substantial.

Can we compute high-precision approximations for the roots and not just isolating intervals. Here is a way to do it. We do not stop the algorithm once we have found an isolating interval, but we proceed until we have determined strong isolating intervals. A strong isolating interval contains a root of P, but no root of the first and second derivative. Then P is monotone and convex in the interval and we can use Newton's method to compute a high precision approximation of the root. **What is the cost of determining all roots with error 2^{-m}? The problem is that a strong isolating interval by itself does not yet guarantee quadratic convergence of Newton's method.**

References

[Akr80] A. G. Akritas. The fastest exact algorithms for the isolation of the real roots of a polynomial equation. *Computing*, 24(4):299–313, 1980.

[CA76] G. E. Collins and A.-G. Akritas. Polynomial real root isolation using Descartes' rule of sign. In *SYMSAC*, pages 272–275, Portland, OR, 1976.

[CJ89] G. E. Collins and J. R. Johnson. Quantifier elimination and the sign variation method for real root isolation. In Gaston H. Gonnet, editor, *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pages 264–271, Portland, OR, July 1989. ACM Press.

[CL82] G. E. Collins and R. Loos. Real zeros of polynomials. *Computing*, Suppl. 4:83–94, 1982.

[Meh01] K. Mehlhorn. A remark on the sign variation method for real root isolation. www.mpi-sb.mpg.de/˜mehlhorn/ftp/Descartes.ps, 2001.

[Mig92] M. Mignotte. *Mathematics for Computer Algebra.* Springer, 1992.

[RZ01] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial real roots. Technical Report 4113, INRIA, 2001.

[Vin36] A. J. H. Vincent. Sur la résolution des equation numériques. *Journal de Mathématiques Pures et Appliquées*, pages 341–372, 1836.

[Yap99] C.K. Yap. *Fundamental Problems in Algorithmic Algebra.* Oxford University Press, 1999.