Accelerated approximation of the complex roots of a univariate polynomial

Victor Y. Pan, Elias Tsigaridas

Accelerated Approximation of the Complex Roots of a Univariate Polynomial

Victor Y. Pan[1] and Elias P. Tsigaridas[2]

[1] Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
and
Ph.D. Programs in Mathematics and Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA

victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

[2] INRIA, Paris-Rocquencourt Center, *PolSys*
Sorbonne Universités, UPMC Univ. Paris 06, *PolSys*,
UMR 7606, LIP6, F-75005, Paris, France
elias.tsigaridas@inria.fr

VP has been supported by NSF Grant CCF 1116736
and by PSC CUNY Award 67699-00 45.

ET has been partially supported by
GeoLMI (ANR 2011 BS03 011 06), HPAC (ANR ANR-11-BS02-013)
and an FP7 Marie Curie Career Integration Grant

The preliminary version of this paper has been presented at SNC 2014

Abstract

The known algorithms approximate the roots of a univariate polynomial in nearly optimal arithmetic and Boolean time. They are, however, are quite involved and require a high precision of computing when the degree of the input polynomial is large, which causes numerical stability problems. We observe that these difficulties do not appear at the initial stage of the algorithms, and in our present paper we extend this stage, analyze it, and avoid the cited problems, still achieving the solution at a nearly optimal estimated complexity, provided that some mild initial isolation of the roots of the input polynomial is ensured. The resulting algorithms promise to be of practical value for root-finding and can be extended to the problem of polynomial factorization, which has also interest on its own.

Keywords: polynomial equation, roots, root-finding, root-refinement, power sums, complexity

1 Introduction

The classical problem of univariate polynomial root-finding has been central in Mathematics and Computational Mathematics for about four millennia since the Sumerian times, and is still important for Signal and Image Processing, Control, Geometric Modeling, Computer Algebra, and Financial Mathematics. It is closely linked to the approximation of linear and nonlinear factors of a polynomial, which is also important on its own right because of the applications to the time series analysis, Weiner filtering, noise variance estimation, covariance matrix computation, and the study of multi-channel systems (see Wilson (1969) [34], Box and Jenkins (1976) [6], Barnett (1983) [1], Demeure and Mullis (1989 and 1990) [8], [9], Van Dooren (1994)) [33].

Nearly optimal algorithms have been developed for both problems (see Pan (1995) [21] and Pan (2002) [23]), but the algorithms are quite involved and their analysis in the cited papers ensures optimal arithmetic and Boolean complexity bounds (up to polylogarithmic factors) only assuming fairly long precision of computing, of the order exceeding the degree of the input polynomial.

The most popular packages of numerical subroutines for root-finding, such as MPSolve 2000 (see Bini and Fiorentino (2000) [3]), EigenSolve 2001 (see Fortune (2002) [10]), and MPSolve 2012 (see Bini and Robol (2014) [5]) employ alternative root-finders based on functional iterations (namely, Ehrlich–Aberth's and WDK, that is, Weierstrass', also known as Durand-Kerner's) and the QR algorithm applied to eigen-solving for the companion matrix of the input polynomial. The user considers these root-finders practically superior by relying on the empirical data about their excellent convergence, even though these data have no formal support.

We re-examine this subject and show that the cited deficiency of the algorithms of [21] and [23] disappears if we modify the initial stage of these algorithms. The resulting algorithms work under some mild assumptions about the initial isolation of the root sets of the input polynomial. Next we briefly comment on the cited results. In the next sections we elaborate upon this outline, analyze the resulting algorithms, and deduce the computational cost estimates.

Recall that polynomial root-finding iterations can be partitioned into two stages. At first a crude (although reasonably good) initial approximations to the roots or to a root are relatively slowly computed. Then these approximations are refined faster by means of the same or distinct iterations.

We partition the second stage into two substages, apply distinct efficient algorithms at these substages, and estimate their Boolean complexity. The estimates show that the computations are highly efficient and nearly optimal (under the assumed mild initial isolation of a root or a root set). Such initial isolation is routinely computed by most of the known root-finders (e.g., by the popular Ehrlich–Aberth' and WDK's iterations and the QR algorithm) on their way to the approximation of the roots, although formal Boolean complexity estimates at this stage are usually missing. Thus our study raises the algorithmic challenge of most efficient computation of such a mild initial isolation.

Having reached it, the known root-finders (e.g., Ehrlich–Aberth' and WDK's iterations) typically continue the iterations as before, but at this point, we can shift to our alternative algorithm, supporting superior and nearly optimal complexity estimates. This suggests that our techniques are promising to become the user's choice.

Besides root-finding applications, these techniques can be a valuable ingredient of the polynomial factorization algorithms. More common is the application of the factorization as the basic stage of root-finding (in particular see Schönhage (1982) [30], [21], and [23]), besides the various other cited applications, but Pan (2012) [24] has elaborated upon the converse reduction where having sufficiently close approximations to the roots of a polynomial one can produce its approximate factorization at a nearly optimal computational cost. Our techniques can be applied to produce the desired close approximations to the roots. An interesting research challenge is to combine our algorithms with the efficient but very involved algorithms algorithms of Kirrinnis (1998) [13] to produce polynomial factorization algorithms that both are simple enough for practical implementation and support factorization at a nearly optimal computational complexity.

Some definitions. Hereafter "ops" stand for "arithmetic operations". We use the norm $||u|| = \sum_{i=0}^{d} |u_i|$ for a polynomial $u = u(x) = \sum_{i=0}^{d} u_i x^i$, write $\mu(h) = O((h\log(h))\log\log(h))$ (cf. Fürer (2009) [11]), and write $\tilde{O}(\cdot)$ to denote the growth order $O(\cdot)$ up to polylogarithmic factors.

2 Isolation Ratio and Root-refinement

The following concept of the *isolation ratio* is basic for us, as well as for [21] and [23]. Assume a real or complex polynomial

$$p = p(x) = \sum_{i=0}^{d} p_i x^i = p_n \prod_{j=1}^{d} (x - z_j), \quad p_d \neq 0, \tag{1}$$

of degree d, an annulus $A(X, R, r) = \{x : r \leq |x - X| \leq R\}$ on the complex plane with a center X, and the radii r and R of the boundary circles. Then the internal disc $D(X, r) = \{x : |x - X| \leq r\}$ is R/r-*isolated*, and we call R/r its *isolation ratio* if the polynomial p has no roots in the annulus.

The isolation ratios for all discs $D(0, r)$ for all positive r can be approximated as long as we can approximate the root radii $|z_j|$ for $j = 1, \ldots, d$. The algorithms of [30] (cf. also Pan (2000) [22] and [23]) yield such approximations within the relative error 0.01, say, by using $O(d\log^2(d))$ ops, but involve the Dandelin's root squaring iteration (see Householder (1959) [12]), and this lead to numerical stability problems. Alternative heuristic algorithms of Bini (1996) [2] and [3] are slightly faster, but also cannot produce close approximation without using root squaring iteration. The Schur-Conn test does not use these iterations and can be applied to estimate the isolation ratio more directly. For discs $D(0, r)$ variant of this test in Renegar (1987) [29, Section 7] amounts to FFT at $d' = 2^h$ points for $16d \leq d' \leq 32d$ with the overhead of $O(n)$ ops and comparisons of real numbers with 0. This means a reasonably low precision of computing and reasonably low Boolean cost.

The next result from Tilli (1998) [31] shows that Newton's classical iteration converges quadratically to a single simple root of p if it is initiated at the center of a $3d$-isolated disc that contains just this root. The result softens the restriction that $s \geq 5d^2$ of [29, Corollary 4.5].

Theorem 1. *Suppose that both discs $D(c, r)$ and $D(c, r/s)$ for $s \geq 3d$ contain a single simple root α of a polynomial $p = p(x)$ of degree d. Then Newton's iteration*

$$x_{k+1} = x_k - p(x_k)/p'(x_k), k = 0, 1, \ldots \tag{2}$$

converges quadratically to the root α right from the start provided $x_0 = c$.

3 Increasing Crude Isolation Ratios of Polynomial Roots

Now suppose that we are given a disc with a single simple root of p having an isolation ratio $1 + \eta$ for a fixed constant $\eta > 0$. Can we increase the ratio to $3d$? Yes, we just need to apply the technique already used in [30] for the computation of the power sums of the roots lying inside such a disc. In our case this is a single root, the power sum is the root itself, and we just need its approximation c within an error of at most Δ such that $r\eta/\Delta \geq 3d$, in which case $\Delta \leq \frac{1}{3} r\eta/d$ and the disc $D(c, \Delta)$ is $3d$-isolated.

We can shift and scale the variable x, and so with no loss of generality we assume dealing with a $(1 + t)^2$-isolated disc $D(0, r)$ for $r = 1/(1 + t)$ for a fixed $t > 0$, and with polynomial p having a single simple root z_1 in this disc.

Consider the following Laurent expansion,

$$p'(x)/p(x) = \sum_{j=1}^{d} \frac{1}{x - z_j} = -\sum_{k=1}^{\infty} \sigma'_k x^{k-1} + \sum_{k=0}^{\infty} \sigma_k x^{-k-1} = \sum_{h=-\infty}^{\infty} c_h x^h \tag{3}$$

where $|x|=1$, $\sigma_0=1$, $\sigma_k=z_1^k$, $\sigma'_k=\sum_{i=2}^d z_i^{-k}$, $k=1,2,\dots$, that is, σ'_k is the kth power sum of the roots of the reverse polynomial $p_{\rm rev}(x)$ that lie in the disc $D(0,r)$. The leftmost equation of (3) is verified by the differentiation of $p(x)=p_n\prod_{j=1}^d(x-z_j)$. The middle equation is implied by the decompositions $\frac{1}{x-z_1}=\frac{1}{x}\sum_{h=0}^\infty\left(\frac{z_1}{x}\right)^h$ and $\frac{1}{x-z_i}=-\frac{1}{z_i}\sum_{h=0}^\infty\left(\frac{x}{z_i}\right)^h$ for $i>1$, provided $|x|=1$ for all i. Note a link of these expressions with the following quadrature formulae for numerical integration,

$$\sigma_m=\frac{1}{2\pi\sqrt{-1}}\int_\Gamma x^m p'(x)/p(x)dx,$$

where Γ denotes the unit circle $\{x:|x|=1\}, 0<m<q$.

We cover the case of any natural number k, although we only need the case where $k=1$. For a fixed natural number q we compute the approximations $\sigma_k^*\approx\sigma_k$ as follows,

$$\sigma_k^*=\frac{1}{q}\sum_{j=0}^{q-1}\omega^{j(k+1)}p(\omega^j)/p'(\omega^j),\quad k=1,2,\dots,q-1. \tag{4}$$

Here $\omega=\omega_q=\exp(2\pi\sqrt{-1}/q)$ is a primitive qth root of unity. Then the evaluation of the polynomial $p(x)$ at the qth roots of unity amounts to the same task for a polynomial $p_q(x)$ of degree at most $q-1$ with the coefficients $p_{q,i}=\sum_{j=0}^l p_{i+jq}$ for $l=\lfloor d/q\rfloor$ obtained by means of less than d additions of the coefficients of p.

Having computed the polynomial $p_q(x)$ we reduce the evaluation of all the desired approximations σ_k^* for $k=1,\dots,q-1$ essentially to performing three DFTs, each on q points, that is to a total of $O(q\log(q))$ ops. Namely, we apply two DFTs to compute $p(\omega^i)$ and $p'(\omega^i)$ for $i=0,1,\dots,q-1$ and a single DFT to multiply the DFT matrix $\Omega=[\omega^{hi}]_{h,i=0}^{q-1}$ by the vector $\mathbf{v}=[p(\omega^i)/p'(\omega^i]_{i=0}^{q-1}$.

Let us estimate the approximation errors. Equations (3) and (4) imply that

$$\sigma_k^*=\sum_{l=-\infty}^{+\infty}c_{-k-1+lq}.$$

Moreover, (3) for $h=-k-1, k\ge1$ implies that $\sigma_k=c_{-k-1}$, whereas (3) for $h=k-1, k\ge1$ implies that $\sigma'_k=-c_{k-1}$. Consequently

$$\sigma_k^*-\sigma_k=\sum_{l=1}^\infty(c_{lq-k-1}+c_{-lq-k-1}).$$

We assumed in (4) that $0<k<q-1$. It follows that $c_{-lq-k-1}=\sigma_{lq+k}$ and $c_{lq-k-1}=-\sigma'_{lq-k}$ for $l=1,2,\dots$, and we obtain

$$\sigma_k^*-\sigma_k=\sum_{l=1}^\infty(\sigma_{lq+k}-\sigma'_{lq-k}). \tag{5}$$

On the other hand $|\sigma_h|\le z^h$, $|\sigma'_h|\le(d-1)z^h$, $h=1,2,\dots$ where $z=\max_{1\le j\le d}\min(|z_j|,1/|z_j|)$, amd so $z\le\frac{1}{1+t}$ in our case. Substitute these bounds into (5) and obtain

$$|\sigma_k^*-\sigma_k|\le(z^{q+k}+(d-1)z^{q-k})/(1-z^q). \tag{6}$$

Therefore it is sufficient to choose q of order $\log(d)$ to decrease the error of the approximation to the root z_1 by a factor of gd^h for any pair of constants g and h, and so we can ensure the desired error bound Δ. To support this computation we only need less than d additions, followed by $O(\log(d))$ evaluations of the polynomial $p_q(x)$ of degree $q-1$ at the lth roots of unity for $l=O(\log(d))$, which involve $O(\log(d)\log(\log(d)))$ ops overall. Summarizing we obtain the following estimates.

Theorem 2. *Suppose the unit disc $D(0,r)=\{x:\ |x|\le1\}$ is $(1+\eta)^2$-isolated for $(1+\eta)r=1$ and a fixed $\eta>0$ and contains a single simple root z of a polynomial $p=p(x)$ of degree d. Then it is sufficient to apply less than d additions and $O(\log(d)\log(\log(d)))$ other ops to compute a $3d$-isolated subdisc of $D(0,r)$ containing this root.*

Combine Theorems 1 and 2 and obtain the following result.

Corollary 3. *Under the assumptions of Theorem 2 we can approximate the root z of the polynomial $p(x)$ within a fixed positive error bound $\epsilon < 1$ by using $O(\log(d)\log(\log(d)) + d\log(\log(1/\epsilon)))$ ops.*

Corollary 4. *Suppose that we are given d discs, each containing a single simple root of a polynomial $p = p(x)$ of degree d and each being $(1+\eta)^2$-isolated for a fixed $\eta > 0$. Then we can approximate all d roots of this polynomial within a fixed positive error bound $\epsilon < 1$ by using $O(d\log^2(d)(1 + \log(\log(1/\epsilon))))$ ops.*

Proof. Apply the same algorithm that supports Corollary 3 concurrently in all d given discs, but instead of the qth roots of unity use q equally spaced points at the boundary circle of each input disc (that is $dq = O(d\log d)$ points overall) and instead of FFT apply the Moenck–Borodin algorithm for multipoint polynomial evaluation Moenck and Borodin (1972) [19].

Also use it at the stage of performing concurrent Newton's iteration initialized at the centers of the $3d$-isolated subdiscs of the d input discs, each subdisc computed by the algorithm that supports Theorem 2. Here we work with the dth degree polynomial p rather than with the qth degree polynomials p_q because to support transition to polynomials p_q of the degree q for d discs we would need to perform d shifts and scalings of the variable x. Instead we employ the Moenck–Borodin algorithm, which still enables us to obtain a nearly optimal root-refiner.

Technically, in a relatively minor change of our algorithm, we replace the matrix $\Omega = [\omega^{j(k+1)}]_{j,k}$ in (4) by the matrix $[c + \omega^{j(k+1)}]_{j,k} = c[1]_{j,k} + \Omega$ where c is invariant in j and k. The multiplication of the new matrix by a vector $\mathbf{v}$ is still reduced to multiplication of the matrix Ω by a vector $\mathbf{v}$ with the additional $3d$ ops for computing the vector $c[1]_{j,k}\mathbf{v}$ and adding it to the vector $\Omega\mathbf{v}$. □

The Moenck–Borodin algorithm uses nearly linear arithmetic time, and [13] proved that this algorithm supports multipoint polynomial evaluation at a low Boolean cost as well (see also J. van der Hoeven (2008) [32], Pan and Tsigaridas (2013a,b) [27], [28], Kobel and Sagraloff (2013) [14], Pan (2015) [25], and Pan (to appear) [26]). Consequently *our algorithm supporting Corollary 4 can be extended to support a nearly optimal Boolean cost bound for refining all simple isolated roots of a polynomial.*

We can immediately relax the assumption that the roots are simple because our proof of Theorem 2 applies to a multiple root as well. Furthermore deduce from the Lucas theorem that the isolation ratio of the basic discs in our algorithms does not decrease when we shift from a polynomial to its derivative and higher order derivatives. Therefore we can just apply Newton's iteration to the derivative or a higher order derivative to approximate a double or multiple root, respectively.

Instead of Newton's, one can apply various other iterative root-finders McNamee (2002) [15], McNamee (2007) [16], McNamee and Pan(2013) [17]. They also support our complexity estimates as long as our power sum algorithms yield isolation of the roots sufficient in order to ensure superlinear convergence of the selected iterations. In particular Ehrlich–Aberth's and WDK iterations have local cubic and quadratic convergence, respectively.

4 Boolean Cost Bounds

Hereafter $\widetilde{O}_B$ denotes the bit (Boolean) complexity ignoring logarithmic factors. By $\lg(\cdot)$ we denote the logarithm with base 2. To estimate the Boolean complexity of the algorithms supported by Corollaries 3 and 4 we apply some results from [27] and [28], which hold in the general case where the coefficients of the polynomials are known up to an arbitrary precision. In our case the input polynomial is known exactly; the parameter λ, to be specified in the sequel, should be considered as the working precision.

At first we consider the algorithm of approximating one complex root, z, of a polynomial p up to any desired precision ℓ. We assume that the degree of p is d and that $\|p\|_\infty \le 2^\tau$.

At first, by following the discussion that preceded Theorem 2, we compute the polynomial p_q and then we apply two DFTs for p_q and p'_q and the inverse DFT for p_q/p'_q.

Assume that p is given by its λ-approximation $\widetilde{p}$ such that $\lg\|p-\widetilde{p}\|_\infty \le -\lambda$. Perform all the operations with $\widetilde{p}$ and keep track of the precision loss to estimate the precision of computations required in order to to obtain the desired approximation.

At first, we compute p_q by using d additions. This results in a polynomial such that

$$\lg\|p_q\|_\infty \le \tau + \lg(d)$$

and

$$\lg(\|p_q-\widetilde{p}_q\|_\infty) \le -\lambda + \tau\lg(d) + 1/2\lg^2(d) + 1/2\lg(d) = O(-\lambda+\tau\log(d)+\log^2(d)).$$

Similar bounds hold for p'_q, that is,

$$\lg(\|p'_q\|_\infty) \le \tau + 2\lg(d)$$

and

$$\lg(\|p'_q-\widetilde{p}'_q\|_\infty) \le -\lambda + \tau\lg(d) + 3/2\lg^2(d) + 1/2\lg(d) = O(-\lambda+\tau\log(d)+\log^2(d)).$$

The application of DFT on p'_q leads us to the following bounds,

$$|p'_q(\omega^i)| \le \tau + 2\lg(d) + \lg\lg(d) + 2 = O(\tau+\log(d))$$

and

$$|p'_q(\omega^i) - \widetilde{p'_q(\omega^i)}| \le -\lambda + \tau\lg(2d) + 3/2\lg^2(d) + 5/2\lg(d) + \lg\lg(d) + 5 = O(-\lambda+\tau\log(d)+\log^2(d))$$

for all i, [28, Lemma 16]. Similar bounds hold for $p_q(\omega^i)$.

The divisions $k_i = p_q(\omega^i)/p'_q(\omega^i)$ output complex numbers such that

$$|k_i| = |p_q(\omega^i)/p'_q(\omega^i)| \le \tau + 2\lg d + \lg\lg d + 2.$$

Define their approximations $\widetilde{k}_i$ such that

$$\lg(|k_i-\widetilde{k}_i|) \le -\lambda + \tau\lg(4d) + 3/2\lg^2 d + 9/2\lg d + 2\lg\lg d + 11 = O(-\lambda+\tau\log(d)+\log^2(d)).$$

The final DFT produces numbers such that the logarithms of their magnitudes are not greater than $\tau + 2\lg d + 2\lg\lg d + 4$ and the logarithms of their approximation errors are at most $-\lambda + \tau\lg(8d) + 3/2\lg^2 d + 13/2\lg d + 4\lg\lg d + 18 = O(-\lambda+\tau\log(d)+\log^2(d))$, [28, Lemma 16].

To achieve an error within $2^{-\ell}$ in the final result, we perform all the computations with accuracy $\lambda = \ell + \tau\lg(8d) + 3/2\lg^2 d + 13/2\lg d + 4\lg\lg d + 18$, that is $\lambda = O(\ell+\tau\log d+\log^2 d) = \widetilde{O}(\ell+\tau)$.

We perform d additions at the cost $O_B(d\,\lambda)$ and perform the rest of computations, that is the 3 DFTs, at the cost $O_B(\log(d)\,\log\log(d)\,\mu(\lambda))$ or $\widetilde{O}_B(d(\ell+\tau))$ [28, Lemma 16]. If the root that we want to refine is not in the unit disc, then we replace τ in our bounds with $d\tau$.

We apply a similar analysis from [27, Section 2.3] to the Newton iteration (see also [28, Section 2.3]) and arrive at the same asymptotic bounds on the Boolean complexity.

In [27] and [28] the error bounds of Newton operator have been estimated by using the properties of real interval arithmetic. In this paper we perform our computation in the field of complex numbers, but this affects only the constants of interval arithmetic, and so asymptotically, both the error bounds and the complexity bounds of the Newton iterations are the same. Thus, the overall complexity is $\widetilde{O}_B(d^2\tau + d\ell)$ and the working precision is $O(d\tau+\ell)$.

In our case we also assume the exact input, that is, assume the coefficients of the input polynomials known up to arbitrary precision; for example, they are integers. For the refinement of the root up to precision of L bits, we arrive at an algorithm that supports the following complexity estimates.

Theorem 5. *Under the assumptions of Theorem 2 we can approximate the root z of the polynomial $p(x) \in \mathbb{Z}[x]$, which is of degree d and $\|p\|_\infty \le 2^\tau$, up to precision of L bits in $\widetilde{O}_B(d^2\tau + dL)$.*

If we are interested in refining all complex roots, we cannot work anymore with the polynomial p_q of degree $q = O(\lg d)$ unless we add the cost of d shifts of the initial approximations to the origin. Instead we rely on fast algorithms for multipoint evaluation. Initially we evaluate the polynomial p of degree d at $O(d \lg d)$ points, and we assume that $\lg \|p\|_\infty \le \tau$. These d points approximate the roots of p, and so their magnitude is at most $\le 2^\tau$.

We use the following result of [28, Lemma 21]. Similar bounds appear in [13, 14, 32].

Theorem 6 (Modular representation)**.** *Assume $m+1$ polynomials, $F \in \mathbb{C}[x]$ of degree $2mn$ and $P_j \in \mathbb{C}[x]$ of degree n, for $j = 1, \dots, m$ such that $\|F\|_\infty \le 2^{\tau_1}$ and all roots of the polynomials P_j for all j have magnitude of at most 2^ρ. Furthermore assume λ-approximations of F by $\widetilde{F}$ and of P_j by $\widetilde{P}_j$ such that $\|F - \widetilde{F}\|_\infty \le 2^{-\lambda}$ and $\|P_j - \widetilde{P}_j\|_\infty \le 2^{-\lambda}$. Let $\ell = \lambda - O(\tau_1 \lg m + m\,n\,\rho)$. Then we can compute an ℓ-approximations $\widetilde{F}_j$ of $F_j = F \mod P_j$ for $j = 1, \dots, m$ such that $\|F_j - \widetilde{F}_j\|_\infty \le 2^{-\ell}$ in $\widetilde{O}_B(m\,n\,(\ell + \tau_1 + m\,n\,\rho))$.*

Using this theorem we bound the overall complexity of multipoint evaluation by $\widetilde{O}_B(d(L + d\tau))$. The same bounds holds at the stage where we perform Newton's iteration. We need to apply Newton's operator $\widetilde{O}(1)$ for each root. Each application of the operators consists of two polynomial evaluations. We perform the evaluations simultaneously and apply Theorem 6 to bound the complexity. On similar estimates for the refinement of the real roots see [28].

We have the following theorem, which complements Corollary 4 with the Boolean complexity estimates.

Theorem 7. *Suppose that we are given d discs, each containing a single simple root of a polynomial $p(x) \in \mathbb{Z}[x]$ of degree d and $\|p\|_\infty \le 2^\tau$, and each being $(1+\eta)^2$-isolated for a fixed $\eta > 0$. Then we can approximate all d roots of this polynomial within L bits in in $\widetilde{O}_B(d^2\tau + dL)$.*

5 Conclusions

References

[1] S. Barnett, *Polynomial and Linear Control Systems*, Marcel Dekker, New York, 1983.

[2] D. Bini, Numerical Computation of Polynomial Zeros by Means of Aberth's Method, *Numerical Algorithms* **13**, 179–200, 1996.

[3] D. A. Bini, G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms*, **23**, 127–173, 2000.

[4] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations*, Volume 1: *Fundamental Algorithms* (XVI + 415 pages), Birkhäuser, Boston, 1994.

[5] D.A. Bini, L. Robol, Solving Secular and Polynomial Equations: A Multiprecision Algorithm *J. Computational and Applied Mathematics*, **272**, 276–292, 2014.

[6] G. E. P. Box, G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco, 1976.

[7] C. Brunie, P. Picart, A Fast Version of the SchurCohn Algorithm, *Journal of Complexity*, **16**, **1**, 54–69, 2000.

[8] C. J. Demeure, C. T. Mullis, The Euclid Algorithm and the Fast Computation of Cross–covariance and Autocovariance Sequences, *IEEE Trans. Acoust., Speech, Signal Processing* **37**, 545–552, 1989.

[9] C. J. Demeure, C. T. Mullis, A Newton–Raphson Method for Moving-average Spectral Factorization Using the Euclid Algorithm, *IEEE Trans. Acoust., Speech, Signal Processing* **38**, 1697–1709, 1990.

[10] S. Fortune, An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials, *J. of Symbolic Computation*, **33**, **5**, 627–646, 2002.

[11] M. Fürer, Faster Integer Multiplication. *SIAM J. on Computing*, **39, 3**, 979–1005, 2009.

[12] A. S. Householder, Dandelin, Lobachevskii, or Graeffe, *American Mathematical Monthly* **66**, 464–466, 1959.

[13] P. Kirrinnis, Polynomial Factorization and Partial Fraction Decomposition by Simultaneous Newton's Iteration, *J. of Complexity* **14**, 378–444 (1998).

[14] A. Kobel and M. Sagraloff, Fast Approximate Polynomial Multipoint Evaluation and Applications, arXiv:1304.8069v1 [cs.NA] 30 April 2013.

[15] J.M. McNamee, A 2002 Update of the Supplementary Bibliography on Roots of Polynomials, *J. of Computational and Applied Math.* **142**, 433-434, 2002; also at web-site `www.yorku.ca/~mcnamee/`

[16] J.M. McNamee, *Numerical Methods for Roots of Polynomials (Part 1)*, Elsevier, Amsterdam, 2007.

[17] J. M. McNamee, V.Y. Pan, *Numerical Methods for Roots of Polynomials*, Part 2 (XXII + 718 pages), Elsevier, Amsterdam, 2013.

[18] K. Mehlhorn, M., Sagraloff, P. Wang, From Approximate Factorization to Root Isolation with Application to Cylindrical Algebraic Decomposition, in *Proc. International Symp. on Symbolic and Algebraic Computations, (ISSAC 2013)*, Boston, Massachusetts, June 2013, (M. Kauers, editor), 283-290, ACM Press, New York (2013).

[19] R. Moenck, A. Borodin, Fast Modular Transform via Division, *Proc. 13th Ann. Symp. Switching Automata Theory*, 90–96, IEEE Comp. Soc. Press, Washington, DC, 1972.

[20] V.Y. Pan, *How to Multiply Matrices Faster. Lecture Notes in Computer Science*, **179**, Springer, Berlin, 1984.

[21] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symp. on Theory of Computing (STOC '95)*, ACM Press, New York, 741–750 (1995).

[22] V. Y. Pan, Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration, *J. of Complexity* **16, 1**, 213–264, 2000.

[23] V. Y. Pan, Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding, *Journal of Symbolic Computations*, **33**, **5**, 701–733, 2002.

[24] V. Y. Pan, Root-refining for a Polynomial Equation, Proceedings of *CASC 2012* (V. P. Gerdt, V. Koepf, E. W. Mayr, and E. V. Vorozhtsov, editors), *Lecture Notes in Computer Science*, **7442**, 271–282, Springer, Heidelberg (2012).

[25] V. Y. Pan, Transformations of Matrix Structures Work Again, accepted by *Linear Algebra and Its Applications*, **465**, 1–32, 2015.

Also available at arXiv:1311.3729v1 [math.NA] 15 Nov 2013.

[26] V. Y. Pan, Fast Approximation Algorithms for Computations with Cauchy Matrices and Extensions, Tech. Report TR-2014005, *PhD Program in Comp. Sci.*, *Graduate Center, CUNY*, 2014

Proc. versions in Proceedings of *CASC 2013* (V. P. Gerdt, V. Koepf, E. W. Mayr, and E. V. Vorozhtsov, editors), Lecture Notes in Computer Science, **8136**, 273–287, Springer, Heidelberg (2013), and *CSR 2014* (E.A. Hirsch et al., editors), Lecture Notes in Computer Science **8476**, 287–300, Springer International Publishing, Switzerland 2014).

[27] V. Y. Pan and E. P. Tsigaridas, On the Boolean Complexity of the Real Root Refinement, in *Proc. International Symp. on Symbolic and Algebraic Computations, (ISSAC 2013)*, Boston, Massachusetts, June 2013, (M. Kauers, editor), 299–306, ACM Press, New York (2013).

[28] V. Y. Pan and E. P. Tsigaridas, Nearly Optimal Refinement of Real Roots of a Univariate Polynomial, Tech. Report, INRIA (2013).

[29] J. Renegar, On the Worst-case Arithmetic Complexity of Approximating Zeros of Polynomials, *J. of Complexity* **3, 2**, 90–113 (1987).

[30] A. Schönhage, The Fundamental Theorem of Algebra in Terms of Computational Complexity, manuscript, Univ. of Tübingen, Germany, 1982, URL: http://www.iai.uni-bonn.de/˜

[31] P. Tilli, Convergence Conditions of Some Methods for the Simultaneous Computations of Polynomial Zeros, *Calcolo*, **35**, 3–15, 1998.

[32] J. van der Hoeven, Fast composition of numeric power series, Tech. Report 2008-09, Université Paris-Sud, Orsay, France, 2008.

[33] P. M. Van Dooren, Some Numerical Challenges in Control Theory, *Linear Algebra for Control Theory, IMA Vol. Math. Appl.* **62**, 1994.

[34] G. T. Wilson, Factorization of the Covariance Generating Function of a Pure Moving-average Process, *SIAM J. on Numerical Analysis* **6**, 1–7, 1969.