New Resultant Inequalities and Complex Polynomial Factorization

Victor Pan

TR-91-026 (revised)*


*TR-91-026 formerly titled "Randomized Incomplete Numerical Factorization of a Polynomial Over the Complex Field"
New Resultant Inequalities and Complex Polynomial Factorization

Victor Pan

Lehman College, CUNY, Bronx, NY 10468.
Supported by NSF Grant CCR 9020690 and PSC-CUNY Award 662478

Abstract. We deduce some new probabilistic estimates on the distances between the zeros of a polynomial $p(x)$ by using some properties of the discriminant of $p(x)$ and apply these estimates to improve the fastest deterministic algorithm for approximating polynomial factorization over the complex field. Namely, given a natural $n$, positive $\epsilon$, such that $\log(1/\epsilon) = O(n \log n)$, and the complex coefficients of a polynomial $p(x) = \sum_{i=0}^{n} p_i x^i$, such that $p_n \neq 0$, $\sum_i |p_i| \leq 1$, we compute (within the error norm $\epsilon$) a factorization of $p(x)$ as a product of factors of degrees at most $n/2$, by using $O(\log^2 n)$ time and $n^3$ processors under the PRAM arithmetic model of parallel computing or by using $O(n^2 \log^2 n)$ arithmetic operations. The algorithm is randomized, of Las Vegas type, allowing a failure with a probability at most $\delta$, for any positive $\delta < 1$ such that $\log(1/\delta) = O(\log n)$. Except for a narrow class of polynomials $p(x)$, these results can be also obtained for $\epsilon$ such that $\log(1/\epsilon) = O(n^2 \log n)$.

1 Introduction

Randomization is already a classical tool for designing effective numerical and algebraic algorithms over finite fields (see, for instance, [18]), but over infinite fields, randomization is usually limited to verification of polynomial identities and to avoiding singularities (see [15], [21]). We are going to demonstrate the power of randomization in a new area, for approximate factorization (over the complex field) of a monic univariate polynomial of degree $n$,

$$p(x) = \sum_{i=0}^{n} p_i x^i = p_n \prod_{j=1}^{n} (x - x_j), \quad ||p(x)|| = \sum_i |p_i|,$$

(1)

that is, given a set of complex coefficients $p_0, p_1, \ldots, p_n$ and a positive $\epsilon$, we seek approximations $x_1^*, \ldots, x_n^*$ to the zeros $x_1, \ldots, x_n$ of $p(x)$ such that

$$||p(x) - p_n \prod_{j=1}^{n} (x - x_j^*)|| \leq \epsilon,$$

(2)

assuming the polynomial norm

$$|| \sum_i u_i x^i || = \sum_i |u_i|.$$

The requirement (2) is motivated by the observation that in practice of computation, the coefficients of $p(x)$ are most frequently available only within certain
truncation errors. For the worst case input polynomial \( p(x) \), where \( |x_j| \leq 1 \), for all \( j \) (this assumption is no loss of generality, since we may scale \( x \)), \((2)\) implies the more classical requirement that

\[
|x_j - z_j^*| < \tilde{e}, \quad \text{for all } j,
\]

as long as \( \tilde{e} \geq 4.4|p_n|(2\epsilon)^{1/n}, 2|p_n|\epsilon < 40^{-n} \) (see Proposition 11 in the appendix below). Moreover, for many polynomials \( p(x) \), the bound \((2)\) implies \((3)\) already for \( \tilde{e} = O(\epsilon/\max |p'(z_j)|) \).

For polynomials \( p(x) \) with multiple or clustered zeros, \((2)\) implies \((3)\), with \( \tilde{e} \) tending to be larger than \( \epsilon \) as \( \epsilon \to 0 \).

**Example 1.** \( p(x) = x^n \) has the only zero \( z_1 = z_2 = \ldots = z_n = 0 \), whose multiplicity is \( n \). \( p(x) + \epsilon = x^n + \epsilon \) has the zeros \( z_j = \epsilon^{1/n} \omega^j, j = 1, \ldots, n; \omega = \exp(2\pi \sqrt{-1}/n) \), so that \( \tilde{e} = \epsilon^{1/n} \).

Example 1 suggests that, for polynomials with multiple or clustered zeros, computing the factorization under \((2)\) is numerically better conditioned than the zero finding problem under \((3)\).

More important for us, the same example also suggests that after a small random perturbation of the coefficients of \( p(x) \), the zeros of the resulting polynomial \( p(x) + \Delta(x) \), with \( \|\Delta(x)\| = \epsilon \), tend to stay apart from each other at the distance at least of the order of \((1/n)^{1/n} \) [even if they correspond to the multiple or clustered zeros of \( p(x) \)], so that such a perturbation promises to eliminate a major obstacle to a rapid convergence of many known algorithms for polynomial factorization, namely, those which converge slowly on the input polynomials whose zeros form clusters.

Substantiation of this intuitive argument is not straightforward, however.

In this paper, we achieve some progress towards such a substantiation, by exploiting some properties of the resultant of \( p(x) \) and its derivative [or of the discriminant of \( p(x) \)]. Based on these properties, we estimate the \( m \)-th diameter

\[
d_m = d_m(x_1, \ldots, x_n)
\]

for a random complex \( \Delta \) uniformly distributed on the circle \( |\Delta| = \rho \), and for a fixed small positive \( \rho \). We have set \( m = n/2 \) (assuming \( n \) even) and proved that, with a probability at least \( 1/(2n+1) \), the reciprocal of the \( m \)-th diameter of the set of the zeros of \( p(x) + x^h \Delta \) is bounded by \( O(1/\rho^{n/(n-2)}) \) (as \( \rho \to 0 \)) if \( h = 0 \), and moreover, for a large class of the input polynomials \( p(x) \) [not including \( p(x) = x^n \) of example 1], the maximum of such a reciprocal over all integers \( h \), \( 0 \leq h \leq n \), is bounded by \( O(1/\rho^{n/(n-2)n}) \) [see (18), (53) and Proposition 4 below, for more specific estimates].

\[
d_m = \min_{|\sigma| = m, i,j \in \mathbb{R}} |x_i - x_j|, \quad m = 2, 3, \ldots, n,
\]
We also examined the resulting effect of the perturbation of \( p(x) \) on the computation of the factorization of \( p(x) \) by means of the effective factorization algorithm of [20] and of its further improvement based on some techniques of [16]. The resulting hybrid algorithm recursively computes a complete numerical factorization \( (2) \) of \( p(x) \) and supports the record running time bounds for this computation, both in terms of arithmetic and Boolean (bit) operations involved. The intermediate steps compute incomplete numerical factorizations of \( p(x) \). If the degrees of the output factors are less than, say, \( 1/2 \) of the input degree in every recursive step, then in at most \( \lfloor \log_2 n \rfloor \) recursive steps, a complete numerical factorization \( (2) \) is computed, and the worst case bounds on the sequential and parallel running times of the entire algorithm decrease roughly by the factors of \( n \) and \( n/\lfloor \log_2 n \rfloor \), respectively. This is because, in the worst case, for each \( i \), recursive step \( i \) splits its input polynomial of degree \( n + 1 - i \) into two factors of degrees \( 1 \) and \( n - 1 \), respectively, so that \( n - 1 \) recursive steps are required to compute a complete recursive factorization \( (2) \).

By using the cited lower bounds on the distances among the zeros of the slightly perturbed polynomial, \( p(x) + x^k \Delta \), we prove that, with a probability bounded away from 0, for a random choice of \( \Delta \), practically the same algorithm splits such a perturbed polynomial into factors of degrees less than \( n/2 \). Thus, the randomization alone strictly improves the performance of the original algorithms.

The same randomization techniques can be recursively applied to split the computed factors of \( p(x) \). In each new recursive step, however, the randomization gives us less and less advantages (see section 8 on the details), and we may end up with shifting to the original deterministic factorization algorithm at some recursive step at which using randomization gives us no advantage anymore.

Below, we will formally state our main results in the form of Theorem 1, which presents our estimates for the error and for the parallel and sequential computational complexity of a single step of splitting \( p(x) + \Delta \) (for a random \( \Delta \), \( |\Delta| = \rho \)) into factors of degrees less than \( n/2 \). We note that our randomization is of Las Vegas type, where no undetected errors may occur (the algorithm may output \textsc{failure} but with a low and controlled probability).

For simplicity, we will use the arithmetic (rather than Boolean) complexity estimates, which is adequate since our deviation from the original algorithms of [20] and [16] does not actually affect the required precision of the computation. We will assume the customary RAM and PRAM models of sequential and parallel arithmetic computations, respectively (see [1], [11]), and will adopt the customary notation \( O_A(t, p) \), which means that \( O(t) \) parallel arithmetic steps and \( p \) processors suffice in the parallel implementation of the solution algorithm. (Here and hereafter, we use the "\( O \)" notation assuming that \( n \to \infty \) and \( \epsilon, \rho \to 0 \).) \( O_A(t, p) \) also implies \( O_A(ts, p/s) \), for any integer \( p/s > 0 \), which gives us the sequential time bound \( O_A(tp, 1) \), for \( s = p \) (Brent's principle [7]).

We will assume hereafter that

\[
\|p(x)\| = 1 ,
\]

\[
1/4 \leq r(C, p(x)) = \max_{j} |x_j - C| \leq 1, C = (1/n) \sum_j x_j = -p_{n-1}/(np_n) .
\]
We may closely approximate \( r(C, p(x)) \) at the cost \( O_A(\log n, n) \) (see Proposition 5 below), and if \( r(C, p(x)) > 1 \) or if \( r(C, p(x)) < 1/4 \), we may shift to a polynomial

\[
q(y) = ay + b = q_n \prod_{j=1}^{n} (y - y_j),
\]

for fixed complex \( a, b \) and \( c \) such that \( ab \neq 0 \),

\[
\|q(y)\| = 1, \quad 1/4 < r(\overline{C}, q(y)) = \max_{j} |y_j| \leq 1, \quad b\overline{C} = \sum_j (x_j - c).
\]

The cost of shifting from \( p(x) \) to \( q(y) \) is also \( O_A(\log n, n) \) (see [2]).

Here is our main result, which we will prove in sections 6 and 7 (also compare Remark 8 below).

**Theorem 1.** Suppose that we are given complex coefficients \( p_0, \ldots, p_n \) of the polynomial \( p(x) \) of (1), satisfying (5), (6), and four positive \( \alpha, \beta, \delta \) and \( \epsilon \) such that

\[
\alpha > 1 > \beta > 0, \quad \beta > 1, \quad n p_n 20^{-n} > \epsilon, \quad \log(1/\epsilon) = O(n^\alpha(\log n)\beta).
\]  

Then there exists a randomized algorithm (of Las Vegas type) that either reports a failure with a probability at most \( \delta \) or, otherwise, computes an incomplete numerical factorization of \( p(x) \) (within error norm \( \epsilon \)) into a product of polynomials, each of degree less than \( \frac{n}{2} \). The algorithm supports the following parallel and sequential complexity estimates for this computation:

\[
c_A = O_A(n^{\alpha - 1}(\log n)^{\beta + 1}, n^3 \log(1/\delta)/(\log n)^\beta),
\]

\[
(sc)_A = O_A(n^{\alpha + 1}(\log n)^{\beta + 1} + n(\log n)^2 \log(1/\delta), 1).
\]

**Remark 1.** If

\[
\log(1/\delta) = O(\log n),
\]

then the estimates (8) and (9) are simplified as follows:

\[
c_A = O_A(n^{\alpha - 1}(\log n)^{\beta + 1}, n^3(\log n)^{1-\beta}),
\]

\[
(sc)_A = O_A(n^{1+\alpha}(\log n)^{\beta + 1}, 1).
\]

[In the summary, we cited (11) and (12) for \( \alpha = \beta = 1 \).]

**Remark 2.** We also show (in section 9) how the complexity bounds (8) and (9) can be ensured, for a large class of input polynomials \( p(x) \), even provided that the upper bound of (7) on \( \log(1/\epsilon) \) is increased by the factor of \( n \), so

\[
\log(1/\epsilon) = O(n^{\alpha + 1}(\log n)^{\beta}).
\]

We organize the remaining part of the paper as follows: In section 2 we estimate some correlations between the absolute value of the discriminant of \( p(x) \) [or of the resultant \( R(p, p') \) of \( p(x) \) and \( p'(x) \)] and the \( m \)-th diameter of the set of zeros of \( p(x) \). In section 3 we estimate \( |R(p + \Delta, p')| \) as a function in \( \Delta \). In section 4 we show our hybrid deterministic algorithm. In section 5 we combine the results of sections 2-4 into our basis lemma (Proposition 9). In section 6 we summarize our study in the
form of a parallel algorithm. In section 7 we modify this algorithm for the sequential computation and will deduce the complexity estimates of Theorem 1. In section 8 we comment on the recursive application of our algorithms. In section 9 we improve our estimated error bounds over a large class of the input polynomials $p(z)$. Finally, in the appendix we recall some auxiliary results on perturbation of polynomial zeros.

Hereafter, all logarithms are to the base 2, and each polynomial zero of multiplicity $\mu$ is counted as $\mu$ zeros.

Acknowledgements. I wish to thank Joseph Lewittes and John Reif, for helpful comments and stimulating discussions, and also C. Engle for her assistance with typing the manuscript.

2 The Magnitude of the Resultant and the Distances between Polynomial Zeros

In this section we will relate the distances between some zeros of a polynomial $p(z)$ of (1) to the absolute value of $R$, the resultant of this polynomial and its derivative $p(z)$. We will make use of the two following expressions for $R$:

$$R = R(p_0, \ldots, p_n) = (-1)^{n(n-1)/2} p_n^{2n-1} \prod_{1 \leq i < j \leq n} (x_i - x_j)^2 ;$$  \hspace{1cm} (13)

$R$ equals the determinant of the $(2n - 1) \times (2n - 1)$ Sylvester (resultant) matrix,

$$R = \det S, \quad S = \begin{bmatrix} p_n & 0 & (np_n) & 0 \\ \vdots & \vdots & \vdots & \vdots \\ p_2 & \cdots & p_n & (2p_2) \\ p_1 & \cdots & p_{n-1} & p_1 \cdots (np_n) \\ p_0 & \cdots & p_{n-2} & p_1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & p_0 & 0 & p_1 \end{bmatrix}.$$  \hspace{1cm} (14)

(14) implies, in particular,

**Proposition 2.** The resultant $R$ is a polynomial in $p_1, p_2, \ldots, p_n$ with integer coefficients, having the total degree $2n - 1$.

We will recall (4), ignore the case of multiple zeros, assuming for a moment that $R \neq 0$, and will immediately deduce from (13) the following

**Proposition 3.** For any integer $m$, $1 \leq m \leq n$, let us denote that

$$D = D(p_0, \ldots, p_n) = \left| R(p_0, \ldots, p_n) / p_n^{2n-1} \right| .$$  \hspace{1cm} (15)

Then

$$D \leq d_n^{n(n-1)} (d_m/d_n)^{m-1}, \quad m = 1, 2, \ldots, n .$$  \hspace{1cm} (16)
We deduce from (16) that
\[ D \leq d_n^{(n-1)}(d_m/d_n)^{m(m-1)}. \]  
(17)

Hereafter, we will assume that \( n \) is even, \( n = 2m \), so that \( m(m-1) = n(n-2)/4. \) Then we will deduce from (17) that
\[ D/d_n^{(3n-2)/4} \leq d_m^{(n-2)/4}, \quad D^2/d_n^{(3n-2)(n-2)} \leq d_m. \]  
(18)

3 Probabilistic Lower bound on the Magnitude of the Resultant

Let \( \Delta \) denote a random complex parameter under the uniform probability distribution on the circle
\[ |\Delta| = \rho \]  
(19)

where the value \( \rho \) will be fixed later on. We are going to deduce some probabilistic lower bound on \( |R_h(\Delta)| = |R(p_0, \ldots, p_{h-1}, p_h + \Delta, p_{h+1}, \ldots, p_n)|, \] \( 0 \leq h \leq n \) [compare (13)]. Recall Proposition 2 and rewrite \( R_h(\Delta) \) as a polynomial in \( \Delta \),
\[ R_h(\Delta) = \sum_{i=0}^{n} r_{h,i} \Delta^i = r_{h,k} \prod_{j=1}^{k}(\Delta - z_{j,h}(\Delta)), \quad k = k(h) \leq n, \]  
(20)

where \( r_{h,i} \) are polynomials in \( p_0, \ldots, p_n \) with integer coefficients and have total degrees at most \( 2n - 1 - i \), \( i = 0, 1, \ldots, n \). Furthermore,
\[ r_{h,1} = \partial R/\partial p_h, \quad h = 0, 1, \ldots, n, \]  
(21)
\[ r_{0,n} = 0, \quad r_{0,n-1} = (-1)^{n(n-1)/2}(np_n)^n. \]  
(22)

The next simple result is the basis for our algorithms.

**Proposition 4.** For a random \( \Delta \), under the uniform probability distribution on the circle (19), for any pair of nonnegative integers \( h \leq n, u \leq n \), and for any fixed positive \( \rho \), we have:
\[ |R_h(\Delta)| > 0.5|r_{h,u}|\rho^u \]  
(23)
with a probability at least \( 1/(2k(h)+1) \geq 1/(2n+1) \), where \( k(h) \) is defined by (20).

**Proof.** Let \( j \) be an integer such that
\[ |r_{h,u}|\rho^u \leq |r_{h,j}|\rho^j = \max_{0 \leq i \leq n} (|r_{h,i}|\rho^i), \quad 0 \leq j \leq n - 1. \]  
(24)

It is well-known from analytic function theory that
\[ \int_{|\Delta| = \rho} (R_h(\Delta)/\Delta^{j+1}) \, d\Delta = 2\pi\sqrt{-1} r_{h,j}. \]

Representing \( \Delta \) as \( \rho \exp(2\pi\sqrt{-1} \phi) \), we obtain that
\[ (1/\rho^j) \int_0^{2\pi} |R_h(\Delta)| \, d\phi = \int_0^{2\pi} |R_h(\Delta)/\Delta^{j+1}| \rho \, d\phi \geq 2\pi |r_{h,j}|. \]
\[ \frac{1}{2\pi} \int_{0}^{2\pi} |R_{h}(\Delta)| \, d\phi \geq |r_{h,j}| \rho^{j} , \]
so that $|r_{h,j}| \rho^{j}$ is a lower bound on the average value of $|R_{h}(\Delta)|$ on the circle (19).
To complete the proof, we compare this bound with the following upper bound:
$|R_{h}(\Delta)| \leq \sum_{i=0}^{k_{h}(h)} |r_{h,i}| \rho^{i} \leq (k_{h} + 1)|r_{h,j}| \rho^{j}$, and recall (24).

Hereafter, denote that
\[ \tilde{p}_{h,n} = p_{n} + \Delta \text{ if } h = n; \quad \tilde{p}_{h,n} = p_{h,n} \text{ otherwise.} \]
Assume that $\tilde{p}_{h,n} \neq 0$ for all $h$. Recall (20) and, similarly to the definitions (15) and (4), denote that
\[ D_{h}(\Delta) = \frac{|R_{h}(\Delta)|}{\rho^{n_{h}-1} \tilde{p}_{h,n}} , \]
\[ d_{h,k}(\Delta) = \min_{\rho \in [1,2], i,j \in I} \max_{k_{h}} |z_{h,i}(\Delta) - z_{h,j}(\Delta)| , \quad k = 2, 3, \ldots, n , \quad h = 0, 1, \ldots, n , \]
and rewrite (18) as
\[ (D_{h}(\Delta))^{4/n(n-2)} / (d_{h,m}(\Delta))^{3(n-2)/(n-2)} \leq d_{h,m}(\Delta) , \quad m = n/2 . \]
Rewrite $3n - 2)/(n - 2)$ as $3 + 4/(n - 2)$, combine (23) and (25), and obtain that
\[ d_{h,m}(\Delta) > (0.5 |r_{h,u}|^{1-2n} \rho^{n})^{4/n(n-2)} / (d_{h,n}(\Delta))^{3+4/(n-2)} , \]
$u, h = 0, 1, \ldots, n$, with a probability at least $1/(2n + 1)$.
In particular, substitute (22) into (26), for $h = 0, u = n - 1$, and arrive at the inequality
\[ d_{0,m}(\Delta) > (0.5 n^{\frac{1}{2}} \rho^{n})^{4/n(n-2)} / (d_{0,n}(\Delta))^{3+4/(n-2)} , \]
where $\lim_{n \to \infty} (0.5 n^{\frac{1}{2}} \rho^{n})^{3/n(n-2)} = 1$.
Also, substitute (21) into (26), for $u = 1$ and $h = 0, 1, \ldots, n$, to obtain that
\[ d_{h,m}(\Delta) > (0.5 |r_{h,1}|^{2n} \rho^{n})^{4/n(n-2)} / (d_{h,n}(\Delta))^{3+4/(n-2)} . \]

4 A Deterministic Algorithm for Polynomial Factorization

Hereafter, $S(C, r)$ will denote the square on the complex plane, with the center $C$ and with the sides parallel to the real and imaginary axes and having length $2r$. $D(C, r)$ will denote the disc on the complex plane, with the center $C$ and radius $r$. We assume that $S(C, r)$ and $D(C, r)$ denote closed complex domains.

$\bar{R} / r$ is called the isolation ratio of the square $S(C, r)$ or of the disc $D(C, r)$ if $\bar{R}$ is the minimum value such that the domains $S(C, \bar{R}) - S(C, r)$ or, respectively, $D(C, \bar{R}) - D(C, r)$ contain a zero of $p(x)$.

We next recall some known results.

**Proposition 5.** Given the coefficients $p_{0}, \ldots, p_{n}$, a complex $C$ and a constant $\theta$, $0 < \theta < 1$, it is possible, at the cost $O(\log n, n)$, to evaluate $r$ such that
\[ \theta r \leq r(C, p(x)) = \max_{1 \leq j \leq n} |C - x_{j}| \leq r . \]
**Remark 3.** \(d_n \leq 2r(C, p(x)) \leq 2r\), under the notation of (29).

Proposition 5 relies on an old algorithm of Turan (see [22], [23]) complemented with more recent estimates for the complexity of its blocks.

**Proposition 6.** ([16]). Let a complex \(C\) and a positive \(r\) be two given values that satisfy (6) and (29). Then there exists a natural \(H = H(n, r) = O(\log n)\) such that, for every \(K > H\), it is possible, at the cost \(O_A(K \log n, n^2)\), to compute complex \(C_i\), real \(r_i\), and natural \(g \geq 2\) and \(k_i \geq 1\), \(i = 1, \ldots, g\), such that for all \(i\),

\[
r_i \leq r n^{2K},
\]

and \(S(C_i, r_i)\) has an isolation ratio at least 3 and contains exactly \(k_i\) zeros of \(p(x)\); furthermore, \(\sum_{i=1}^{g} k_i = n\), and \(S(C_j, r_j) \cap S(C_h, r_h) = \text{empty}\) if \(1 \leq j < h \leq g\).

The proof of Proposition 6 (in [16]) exploits and improves the search-and-exclusion construction of Weyl for approximating polynomial zeros (compare [9] and [19] on other applications of Weyl’s construction).

**Proposition 7.** Let \(S(C_i, r_i)\), the squares defined in Proposition 6, for \(i = 1, \ldots, g\), be given to us, together with the associated integers \(k_i\) and with a positive \(\epsilon^*\). Let us denote

\[
b = b(\epsilon^*) = (1/n) \log(1/\epsilon^*).
\]

Then, at the cost \(O_A(\log n \log(\log bn), gn \log b/\log(\log bn))\), the coefficients of \(g \leq n\) monic polynomials \(p_1(x), \ldots, p_g(x)\) can be computed such that the polynomial \(p_i(x)\) has exactly \(k_i\) zeros, all lying in \(S(C_i, r_i)\), \(i = 1, \ldots, g\), and

\[
||p(x) - p_1(x) - \ldots - p_g(x)|| \leq \epsilon^*.
\]

Proposition 7 follows from the results of [20] (Corollary 4.3 and section 12) applied to the discs \(D(C_i, r_i \sqrt{2})\), \(i = 1, \ldots, g\); such a disc \(D(C_i, r_i \sqrt{2})\) contains exactly \(k_i\) zeros of \(p(x)\) (lying in the square \(S(C_i, r_i)\)) and has an isolation ratio at least \(3/\sqrt{2} > 2\) (compare Remark 4 below).

Combining Propositions 5–7, we devise

**Algorithm 1.** Input: the complex coefficients \(p_0, \ldots, p_{n-1}\) of a polynomial \(p(x)\) of (1) and a positive \(\epsilon^*\).

Output: positive integers \(g > 1, k_1, k_2, \ldots, k_g\) (such that \(k_1 + \ldots + k_g = n\)) and the coefficients of monic polynomials \(p_i(x)\) of degrees \(k_i\), \(i = 1, \ldots, g\), satisfying (32).

**Stage 1.** Compute \(C\) and \(r\) satisfying (6) and (29).

**Stage 2.** Compute the squares \(S(C_i, r_i)\) defined in Proposition 6.

**Stage 3.** Compute a factorization (32), by following Proposition 7.

We immediately deduce the bounds

\[
C_A^* = O_A(\log n, n^2)
\]

on the parallel cost of the computation by algorithm 1, where \(b\) has been defined in Proposition 7 and where we may choose any \(K\) exceeding \(H = O(\log n)\). (Later on, we will see some advantages of choosing a larger value of \(K\).)

We may recursively apply algorithm 1 in whose input set we replace \(p(x)\) by the polynomials \(p_1(x), \ldots, p_g(x)\), as long as they have degrees exceeding 1 (these \(g\)
applications can be performed concurrently, and similarly in the further recursive steps). The maximum degree of the output polynomials of algorithm 1 is \( n - g + 1 \) or less, so that in at most \( n - 1 \) its applications and at the overall cost bounded by \( O_A((K + \log(bn))n \log n, n^3) \), we will arrive at the factorization (2) of \( p(x) \) into linear factors. Here, we impose the same error bound \( \epsilon^* \) for the numerical factorization of all the auxiliary polynomials that we factorize in this recursive process. We choose this bound \( \epsilon^* \) sufficiently small, so as to ensure the error bound \( \epsilon \) for the complete numerical factorization (2). Specifically, we are guided by the next proposition from section 5 of [20], which relates \( \epsilon^* \) to \( \epsilon \):

Proposition 8. Let

\[
\|p(x) - p_1(x) \cdots p_h(x)\| \leq \epsilon h/n, \quad \|p_2(x) - f(x)g(x)\| \leq \epsilon^*, \quad \epsilon^* = \frac{\epsilon \|p_1(x)\|}{n2^n\|p(x)\|}.
\]

Then

\[
\|p(x) - f(x)g(x)p_2(x) \cdots p_h(x)\| \leq (h + 1)\epsilon/n.
\]

Due to Proposition 8, it is sufficient to choose

\[
\epsilon^* = \frac{\epsilon \|w(x)\|}{n2^n\|p(x)\|} \geq \frac{\epsilon}{n2^n\|p(x)\|}
\]

whenever algorithm 1 is applied to a monic polynomial \( w(x) \) in the recursive process described above.

Next suppose that, by modifying the first step of this recursive process, we may ensure a more rapid decrease of the degrees of the factors, such that

\[
\max_{1 \leq s \leq g} \deg p_s(x) < 0.5 \deg p(x) = n/2,
\]

and similarly at all the next recursive steps. Then it would have sufficed to use \( [\log_2 n] \) (rather than \( n - 1 \)) concurrent recursive steps of application of algorithm 1 [as before, we ensure the desired output error bound \( \epsilon \) by imposing the error bound \( \epsilon^* \) of (34) at all the recursive steps].

We easily estimate that in this case the complexity of recursive step \( i \) is \( O_A((K + \log(bn))n \log n, n^2/2^i) \), for \( i = 1, 2, \ldots, [\log n] \), which implies that the overall complexity of the complete numerical factorization of \( p(x) \) into linear factors is \( O_A((K + \log(bn))n^2 \log n, n^2/\log n) \). (To arrive at this bound, we apply Brent's principle of [7] by slowing down those recursive steps that otherwise would have used more than \( n^2/\log n \) processors.)

These estimates show that algorithm 1 performs more effectively in the cases where the factors computed at its recursive steps have smaller degrees. In the next sections we will apply a randomization technique to split \( p(x) \) into factors of degrees less than \( n/2 \).

Remark 4. Algorithm 1 is a simplification (based on the results of [16]) of the algorithm of [20]. The recursive step of [20] splits \( p(x) \) into two factors, whose zeros lie inside and, respectively, outside a fixed disc that has an isolation ratio of the order of \( 1 + 1/n \). Since we deal with the squares \( S(C_i, \rho_i) \) of Propositions 6, 7, having isolation ratios at least 3, the computation of the splitting by using algorithm 1 is substantially simpler than the similar computation in [20].
5 How to Ensure a Balanced Partition of the Set of Zeros

In the next result, we use \( D \) and \( r \) defined by (15) and (29).

**Proposition 9.** Let the integer \( K \) of Proposition 6 be chosen such that
\[
2^{K-0.5} > n(2r)^{(n-1)/(n-2)} D^{4/(n(n-2))}.
\]

Then \( 2\sqrt{2} \, r_i < d_m, \) for \( i = 1, \ldots, g, \) that is, every square \( S(C_i, r_i), \) \( i = 1, \ldots, g, \) of Proposition 6 contains less than \( m = n/2 \) zeros of \( p(x) \).

**Proof.** Proposition 9 immediately follows from the comparison of the upper bound (30) on \( r_i \) with the lower bound (18) on \( d_m. \) [Compare also Remark 3 and observe that the diameter of the square \( S(C_i, r_i) \) equals \( 2\sqrt{2} \, r_i. \)]

**Remark 5.** As an exercise, the reader may extend (18) and Proposition 9, as well as our subsequent study, by choosing a smaller \( m, \) say, \( m = \sqrt{n}. \)

Proposition 9 can be immediately extended to the case where \( p(x) \) is replaced by \( p(x) + \Delta, \) for \( \Delta \) of (19), \( D \) by \( D_0(\Delta) \) (see section 3) and similarly \( r \) of (29) by \( r_0(\Delta); \) in particular, we may rewrite the assumption (35) as follows:
\[
D_0(\Delta) > (2r_0(\Delta))^{(n-1)n(n-2)/n} 2^{-(K-0.5)(n-2)n/4}.
\]

Now let \( d, K \) and \( \rho \) be such that \( d \geq 2r_0(\Delta), \)
\[
0.5|r_{0,n-1}p^n_{n-2}\rho^{n-1} \geq d^{(n-1)n(n-2)/4} 2^{-(K-0.5)(n-2)n/4}.
\]

Apply Proposition 4 for \( h = 0, u = n - 1 \) and deduce that [with a probability at least \( 1/(2n + 1) \)] we have:
\[
D_0(\Delta) = |R_0(\Delta)/p^n_{n-2}| > 0.5|r_{0,n-1}/p^n_{n-1}|\rho^{n-1} \geq d^{(n-1)n(n-2)/4} 2^{-(K-0.5)(n-2)n/4}.
\]

Since \( d \geq 2r_0(\Delta), \) the latter inequality implies (36), and, consequently, implies the extension of Proposition 9 to the case where the polynomial \( p(x) + \Delta \) replaces \( p(x). \)

6 The Factorization Algorithm

Now we are ready to summarize our previous study, by devising the following randomized (of Las Vegas type) algorithm:

**Algorithm 2. Input:** the complex coefficients \( p_0, \ldots, p_{n-1} \) of a polynomial \( p(x) \) of (1), a positive \( \rho \) such that
\[
r(C, p(x)) \leq 2r(C, p(x) + \Delta) \leq 3r(C, p(x)) \quad \text{if} \quad |\Delta| = \rho, \quad C = -p_{n-1}/(np_n)
\]
(\( \text{compare Remark 6 below,} \) and a tolerance \( \delta \) to the probability of a failure, \( 0 < \delta < 1 \) [\( \text{compare (7)} \)].

**Output:** FAILURE with a probability at most \( \delta \) or, otherwise, an integer \( g, \ g > 2, \) a complex value \( \Delta, \ |\Delta| = \rho, \) and the coefficients of \( g \) polynomials \( p^i(x, \Delta) \) having degrees less than \( n/2 \) and satisfying the inequality (40) below.
Stage 1. Successively compute the values

$$
\nu = \left\lceil \frac{(2n+1) \log(1/\delta)}{\log 3} \right\rceil,
$$

then $C$ and $r$ satisfying (6) and (29), for $\Theta = 0.99$ (compare Remark 6 below), then $d = 3.1 r$, and finally, the minimum positive integer $K$ exceeding $H$ of Proposition 6 and satisfying (37). Choose $\nu$ independent random values $\Delta$ on the circle $|\Delta| = \rho$, under the uniform probability distribution.

Stage 2. For all such values $\Delta$ and for $\Delta = 0$, concurrently compute $D_0(\Delta)$. If (36) has been satisfied for none of the values $\Delta$, end this computation and output FAILURE. Otherwise, fix some $\Delta$ for which (36) holds, apply algorithm 1 to the polynomial $p(x) + \Delta$ in order to compute and to output a natural $g > 2$ and the coefficients of the polynomials $p_i(x, \Delta)$ that approximate $g$ distinct nonconstant factors $p_i(x, \Delta)$ of $p(x) + \Delta$, for $i = 1, \ldots, g$, such that

$$
p(x) + \Delta = \prod_{i=1}^{g} p_i(x, \Delta); \quad ||p(x) + \Delta - \prod_{i=1}^{g} p_i(x, \Delta)|| \leq \rho;$$

$$
||p(x) - \prod_{i=1}^{g} p_i(x, \Delta)|| \leq 2\rho,
$$

and end the computations. [Note that $2\rho$ plays the role of $\epsilon^*$ of (32), (34) and algorithm 1.]

Let us next show the correctness of the algorithm. First observe that $d = 3.1 r \geq 3.1 r(C, p(x))$ [due to (29)]. Recall (38) and obtain that $d \geq 2.05 r(C, p(x) + \Delta)$. Extend (29) by replacing $p(x)$ by $p(x) + \Delta$ and $r$ by $r_0(\Delta)$, and obtain that $d \geq 2.05 \Theta r_0(\Delta)$. Substitute $\Theta = 0.99$ and obtain that

$$
d > 2r_0(\Delta).
$$

Now apply Proposition 4, for $h = 0$, $u = n-1$, and deduce that, with a probability at least $1/(2n+1)$,

$$
|R_0(\Delta)| > 0.5r_{0,n-1}\rho^{n-1},
$$

which implies that

$$
D_0(\Delta) = |R_0(\Delta)/p_n^{2n-1}| > 0.5r_{0,n-1}|p_n^{1-2n}|\rho^{n-1}.
$$

The latter inequality implies (36) since we have chosen the value of $K$ satisfying (37). Therefore, we may apply Proposition 9 extended to the case of the input polynomial $p(x) + \Delta$. Then, application of algorithm 1 at stage 2 of algorithm 2 gives us a numerical factorization of $p(x) + \Delta$ satisfying (40), and we observe that the probability of a failure in an application of stage 2 to $p(x) + \Delta$ for all the $\nu$ values $\Delta \neq 0$ is at most $(1 - 1/(2n+1))^\nu < 1/2^{\log(1/\delta)/\log 3} = \delta$ [see (39)].

Remark 6. The assumption (38) surely holds for any sufficiently small positive $\nu$. In particular, we may deduce from Proposition 11 of Appendix A below that it is sufficient if

$$
2\rho < 40^{-n}|p_n|.
$$
Of course, the reader may replace the "magic" numbers 2 and 3 in (38) and the value \( \Theta = 0.99 \) by various other candidate values, so as to preserve the correctness of algorithm 2, respectively modifying (42).

**Remark 7.** We may further modify stage 2 of algorithm 2 by first computing the coefficients of \( D_0(\Delta) \) as above and then by computing \( \max_{|\Delta| = \rho} |R_0(\Delta)| \). The problem can be immediately reduced to computing all the real zeros of a polynomial of degree \( O(n) \). This problem may generally be not simpler than the original problem, but the approach may serve as a heuristic algorithm, for we may already be satisfied with having a rough approximation to the maximum of \( |R_0(\Delta)| \) on the circle \( |\Delta| = \rho \).

7 Computational Complexity Estimates

Next, we will assume that \( p(x) \) and \( z \) have been scaled so that

\[
d \leq 1
\]  

(43)

and will estimate the parallel complexity \( c_\lambda \) of performing algorithm 2. Clearly, \( c_\lambda \) is dominated by the complexity of computing \( D_0(\Delta) \) for all the \( x \) values \( \Delta \) [bounded by \( O_A(\log^2 n, \nu n^2 / \log n) \), due to [17] ] and of applying algorithm 1 [see (33)], so

\[
c_\lambda = O_A \left[ \left(K + \log \log n\right) \log n, n^2(1 + \nu / (K + \log \log n)) \right]
\]  

(44)

where \( K \) denotes the minimum integer exceeding \( H \) (defined in Proposition 6) and satisfying (37), \( bn = -1 + \log(1/\rho) \) [compare (31)] and \( \nu \) is defined by (39).

Now observe that (40) implies (32), for any \( \rho \leq \epsilon^* / 2 \) and for \( p_i(x) = p_i^*(x, \Delta) \), \( i = 1, \ldots, 9 \). Suppose that \( \alpha, \beta, \delta \) and \( \epsilon \) are given as in Theorem 1, such that (7) holds. Then (38) holds for \( \rho = \epsilon^* / 2 \) [see (5), (34) and Remark 6]. Set \( \rho = \epsilon^* / 2 \) and apply algorithm 2. By the definition of \( K \) given at stage 1 of this algorithm,

\[
K = \max\{H + 1, [K^*]\}
\]  

(45)

\( H = O(\log n) \), and setting \( K = K^* \) turns (37) into an equation, that is,

\[
2^{(K^* - 0.5) n (n-2)/4} = 2^{d(n-1)n(n-2)/8} |p_n|^{2n-1} / \tau_{0,n-1} \rho^{n-1}.
\]

Substitute (22), take logarithms on both sides, divide the resulting equation by \( n(n-2)/4 \) and obtain that

\[
K^* = 0.5 + 4/(n-2)n + (1 - 4/(n-2)) \log n + ((2n-4) / (n-2)n) \log |p_n|^{2n-1} / \rho.
\]  

(46)

Replace \( \rho \) by \( \epsilon^* / 2 \), recall (5), (34), (43) and Proposition 9, and deduce that

\[
K = O(\log n + (1/n) \log(1/\epsilon^*)) = O(\log(n/\epsilon^*)^{1/\kappa})
\]  

(47)

We also have that

\[
bn = O(\log(1/\epsilon^*))
\]  

(48)

\[
\nu = O(n \log(1/\delta))
\]  

(49)
[compare (31), (39)]. Combining (44)-(49), we obtain that \( K + \log bn = \phi(n, \epsilon) = O(\log(\log(1/\epsilon) + (n/\epsilon^{1/n}))) \), and furthermore that
\[
c_A = O_A(\phi(n, \epsilon) \log n, n^3 \log(1/\delta) / \phi(n, \epsilon)).
\] (50)

In particular, if
\[
\log(1/\epsilon) = O(n^\alpha \log(n)^\beta), \quad \text{for a fixed } \beta \geq 1,
\]
then \( \log(1/\epsilon) = O(\log n) \); \( \log(1/\epsilon^{1/n}) = O(n^{-\alpha-1} \log n)^\beta \), and we arrive at (8).

To deduce the bound (9), we modify stage 2 of algorithm 2 as follows:

a) compute \( R_0(\Delta) \) for \( n \) values \( \Delta \) equal to all the \( n \)-th roots of 1; for each value of \( \Delta \), such an evaluation amounts to computing the determinant of a Sylvester matrix, for the sequential cost \( O_A(n(\log n)^2, 1) \) ([4], [5], [13]); this means \( O_A((n \log n)^2, 1) \) for all the values \( \Delta \);

b) compute the coefficients of \( R_0(\Delta) \) as a polynomial in \( \Delta \) [see (14) and (20)] by means of the inverse FFT, for the cost \( O_A(\log n, n) \) or \( O_A(n \log n, 1) \) ([1]);

c) define \( \nu \) by (39) and compute \( D_0(\Delta) \), for \( \nu \) random and independent values \( \Delta \) on the circle \( |\Delta| = \rho \), for the overall cost \( O_A((\nu + n) \log^2 n, 1) \) ([1], [4], [6]);

d) among these \( \nu \) values, choose the value \( \Delta \) that maximizes \( |D_0(\Delta)| \) and output \( \text{FAILURE} \) if (36) does not hold, for this value of \( \Delta \);

e) otherwise, if (36) holds, apply algorithm 1 to the polynomial \( p(x) + \Delta \), at the cost \( C_A^* = O_A((K + \log bn) \log n, n^2) \), bounded according to (33), (45)-(48); by Brent's principle of [7], this implies the sequential cost bound \( (sC)^*_A = O_A(n^2(K + \log bn) \log n, 1) \).

We will refer to the resulting algorithm as to \textit{algorithm 3}.

Summarizing all these estimates for the complexity of performing algorithm 3 and assuming (7), (38), (39), we arrive at the bound (9) on the sequential complexity of computing the approximate factorization (2), thus proving Theorem 1.

\textbf{Remark 8.} We may perform stage a) of algorithm 3 at the cost \( O_A(\log^2 n, n^3 \log \log n) \) by concurrently applying the parallel algorithm of [17] for all the \( n \) values of \( \Delta \). Stage c) can be performed at the parallel cost \( O_A(\log^2 n, (\nu + n) \log \log(\nu + n)) \) ([4]), which turns into \( O_A(\log^2 n, n \log(1/\delta) \log \log(\log(1/\delta))) \) under (39) and which is dominated by the above bound on the cost at stage a) under the mild condition that \( \log(1/\delta) = o(n^2 / \log n) \). Assuming this condition and recalling the bound \( O_A(n^{\alpha-1} \log n)^{\beta+1}, n^2 \) on the cost at stage e), we obtain the overall cost bound
\[
O_A\left(n^{\alpha-1} \log n)^{\beta+1}, n^2\left(1 + n^{2-\alpha} / \log n\right)\right)
\]
which slightly improves (8).

\section{Extension to Recursive Factorization}

Unless \textit{FAILURE} has been output, stages 1 and 2 of algorithms 2 or 3 can be repeated with each of the polynomials \( p_i(x, \Delta) \) of degree \( k_i > 1 \) replacing \( p(x) \), with \( k_i \) replacing \( n \) in (37), and with
\[
\nu_i = \left[(2k_i - 1) \log(1/\delta) / \log 3\right]
\]
replacing $\nu$ of (39). Then the failure probability is at most $\delta$ at this factorization stage for all the $p^*_i(x, \Delta)$. The difficulty with this extension of algorithms 2 and 3 is that in order to compute the factorization of $p^*_1(x, \Delta)$, we need to replace $n$ by $k_i$ in both (37) and (46), so that at this step, our estimate gives us $K = O((1/k_i) \log(1/\rho))$, rather than $K = O((1/n) \log(1/\rho))$. At the subsequent recursive steps, we need to replace $k_i$, in this expression for $K$, by smaller and smaller values. This means that we need to choose larger and larger values of $K$ to ensure the desired bound on the approximation error of the factorization. This way we increase the running time of the algorithm. Thus, at some recursive step, the estimated time of our randomized computation may exceed the estimated time bound for the recursive step of the deterministic algorithm (based on algorithm 1), to which we shall shift at this point. It is desired, of course, to decrease the error of the factorization, so as to take advantage of the randomization as long as possible, reaching the factors of $p(x)$ that have smaller degrees. To achieve this, we will next slightly modify our randomization approach.

9 Decreasing Upper Bounds on the Factorization Error

Algorithms 2 and 3 and their analysis in section 7 relied on (22) and (27). For a large class of input polynomials $p(x)$, we may reach the same asymptotic complexity bounds (8) and (9) assuming that, roughly,

$$\log(1/\epsilon) = O(n^{1+\alpha}(\log n)^\beta),$$  \hspace{1cm} (51)

that is, for a large class of inputs, we may arrive at a superior approximation to the factorization of $p(x)$ by using roughly the same amount of computational resources. [We refer to (55), (56) below, for a more precise statement of the assumption (51).] The improvement is important in the recursive application of the factorization algorithms (see the previous section).

Let us supply some details. Denote

$$w_h = 0.5|p_{h,n}^{1-2n} \partial R/\partial p_h|, \quad h = 0, 1, \ldots, n, \hspace{1cm} (52)$$

substitute this expression on the right side of (28), also replacing $d_{h,n}(\Delta)$ by $d$ of (41), $d \geq d_{h,n}(\Delta)$, and obtain that

$$d_{h,m}(\Delta) > (w_h \rho)^{3/5(n+2)} / c^{3+4/(n-2)}. \hspace{1cm} (53)$$

We need to choose $K$ such that $d_{\nu^{1/2}}/2^K < d_{h,m}(\Delta)$, due to Proposition 6. Assume that $w_h \neq 0$ and extend Proposition 9 replacing (45), (46) by the following expression:

$$K = \max\{H + 1, [0.5 - \frac{4}{n(n-2)} \log(w_h \rho) + \log n + \frac{4n-4}{n-2} \log d]\}. \hspace{1cm} (54)$$

Assume that $\rho = \epsilon^*/2$ and $d \leq 1$ [compare (34), (43)], and deduce that

$$K = O(\log n + (1/n^2) \log(1/(w_h \epsilon))) = O(\log n / (w_h \epsilon)^{1/n^2}), \text{ unless } w_h = 0. \hspace{1cm} (55)$$
This is a considerable improvement of (47) unless \( w_h \) is small. Clearly, \( w_h \) does not decrease as \( \epsilon \to 0 \), and (52) and Proposition 2 imply that

\[
w_h \geq 0.5 \cdot \epsilon |z|^2 \cdot p_{h,n} |1 - 2^n|
\]

if \( z \neq 0 \), \( z \) are integers for all \( i \) and \( \partial R / \partial p_h \neq 0 \). Furthermore, we may evaluate \( \partial R / \partial p_h \), for \( h = 0, 1, \ldots, n \), at the same asymptotic computational cost [bounded by \( O_A(\log^2 n, n^2 / \log n) \) or alternatively by \( O_A(n \log^2 n, 1) \)] that we need for the evaluation of \( R \). We achieve this just by applying to \( R \) the parallel algorithm of [10], which extends the sequential algorithm of [3], [12]. [The algorithm of [10] computes all the first order partial derivatives of any polynomial \( p(y_1, \ldots, y_n) \) at the cost \( O_A(t, p) \), provided that an algorithm is available that computes \( p(y_1, \ldots, y_n) \) at the cost \( O_A(t, p) \), by using only arithmetic operations, with no branchings.]

Thus, initially, we may compute \( \partial R / \partial p_h \) and \( w_h \), for \( h = 0, 1, \ldots, n \), then choose \( h \) for which \( w_h \) is maximum, and finally compare \( w_h \) with \( 0.5 \cdot \epsilon |z| / p_n |1 - 2^n| \) [see (22), (23)]. If the latter value is greater, we shall go to algorithms 2 or 3 with no changes. Otherwise, we shall apply one of these algorithms, replacing \( D_0(\Delta) \) by \( D_h(\Delta) \) and \( K \) of (45), (46) by \( K \) of (54).

In the latter case, we shall replace \( \phi(n, \epsilon) \) in (50) by \( \log(1/\epsilon) + n/(\epsilon w_h)^{1/n^2} \), and thus shall arrive at the overall complexity estimates (8) and (9) assuming that

\[
\log(1/\epsilon) = O((\log n)^\beta),
\]

\[
\log(1/\epsilon w_h) = O(n^{\alpha+1}(\log n)^\beta), \quad \text{for fixed} \quad \alpha \geq 1, \quad \beta \geq 1.
\]

((55) and (56) turn into (51) if, say, \( 1/w_h = O(1) \).]

Remark 9. Instead of computing all the partial derivatives, we may just try to factorize the polynomial \( \sum_{i=0}^n (p_i + c_i \Delta) x^i \) where \( c_i \) have been chosen at random, say, in the real interval from 1 to 2 and \( \Delta \) has been chosen as before, at random on the circle (19) for an appropriate \( \rho \). This approach (for appropriate \( \rho \)) shall work as long as \( o^h \cdot (\det S) \) for some \( h = O(1) \) and for some \( i \geq 1 \) (not necessarily for \( i = 1 \)).

Appendix A. Perturbation of Polynomial Zeros

Approximation to polynomial zeros and numerical polynomial factorization involve some nontrivial estimates of how much the zeros are perturbed by the perturbation of the coefficients [compare (2), (3) and Proposition 8]. This problem is closely related to our study, and next we will recall some known results on this topic.

We will first recall a simple corollary (see [9], [14]) from a perturbation theorem by Ostrowski. This theorem bounds the magnitude of the perturbation of the coefficients so that the magnitude of the resulting perturbation of the zeros of \( p(x) \) never exceeds \( 2^{-\mu} \).

Let us denote that

\[
||u_i||_\infty = \max_{i} |u_i|.
\]

Proposition 10. For two monic polynomials \( p(x) \) and \( p^*(x) \) of degree \( n \), where \( ||p(x)||_\infty < 2^{n} \), \( ||p(x) - p^*(x)||_\infty = 2^{-h} \), the zeros \( z_1, \ldots, z_n \) of \( p(x) \) and \( z_1^*, \ldots, z_n^* \) of \( p^*(x) \) can be enumerated so that \( \max_j |z_j - z_j^*| < 2^{m + \log n + 2^{-h/n}} \).
For a large class of polynomials, we may improve the latter estimates if we separately consider the zeros of \( p(x) \) inside and outside the unit circle \( |x| \leq 1 \) (see [20]). Alternatively, we may just scale \( x \) so as to place all the zeros inside this circle (as we have assumed this in (6)), then argue similarly to [20], and arrive at the next improved estimates.

**Proposition 11.** Given a natural \( N \), a positive \( \epsilon < 0.5/40^N \), and the complex coefficients of two monic polynomials of degree \( N \), \( P(x) \) and \( P^*(x) \), such that \( P(x) = \prod_{j=1}^N (x - z_j) \),

\[
\|P(x) - P^*(x)\| < \epsilon ,
\]

\[
\max_j |x_j| \leq 1 ,
\]

we may enumerate the zeros of \( P^*(x) \) so that \( P^*(x) = \prod_{j=1}^N (x - x_j^*) \) and

\[
|x_j - x_j^*| < 4.4(2\epsilon)^{1/N} , \quad j = 1, \ldots, N
\]

**Proof.** First observe that \( P^*(x) = 0 \) implies that

\[
|P(x)| = |P(x) - P^*(x)| \leq \|P(x) - P^*(x)\| \max \{1, |x|^N \}.
\]

Therefore, the open set

\[
T = \{ z : |P(z)| < \epsilon \max \{1, |z|^N \} \}
\]

contains all the zeros of \( P(x) \) and \( P^*(x) \). Furthermore, the homotopy argument shows that in a fixed component \( T_k \) of \( T \), the polynomial \( P(t, z) = P(z) + t(P^*(z) - P(z)) \) has the same number of zeros for all \( t \) from 0 to 1; in particular, the two polynomials \( P(z) = P(0, z) \) and \( P^*(z) = P(1, z) \) have the same number of zeros in \( T_k \).

Now, let \( P(x) = 0 \), for some \( x \in T_k \), so that \( |x| \leq 1 \) [see (58)]. Let

\[
arc(x, y) = \{ x(u) = x + u z(u), |z(u)| = 1, 0 \leq u \leq d \}
\]

be an arc in \( T_k \) with the endpoints \( x = x(0), y = x(d) \). Then \( |P(x + u z(u))| = \prod_j |x - x_j + u z(u)| \). On the other hand, by applying first (60) and then (57) and (61), we deduce that \( |P(x + u z(u))| < \epsilon \max \{1, |x + u z(u)|^N \} \leq (1 + u)^N \leq (1 + d)^N \). Therefore,

\[
|s(u)| = \prod_{j=1}^N |u - |x - x_j|| \leq \prod_{j=1}^N |x - x_j + u z(u)| < \epsilon (1 + d)^N , \quad \text{for} \quad 0 \leq u \leq d.
\]

The monic polynomial \( s(u) \) has degree \( N \), so that \( 2 \max_{0 \leq u \leq d} |s(u)| \geq (d/4)^N \) ([8], p. 241). Consequently, \( (d/4)^N < 2\epsilon (1 + d)^N, d < 4(1 + d)(2\epsilon)^{1/N} \). It follows that

\[
d \leq 0.1(1 + d) , \quad d \leq 1/9 , \quad \text{for} \quad 2\epsilon \leq 1/40^N
\]

and therefore,

\[
d < 4.4(2\epsilon)^{1/N},
\]

which implies (60).
Let us now apply Proposition 11 under (42), to ensure (38). (42) and (59) together imply that

$$|r(C, p(x) + \Delta) - r(C, p(x))| < 4.4(2p/|p_n|)^{1/n} < 0.11,$$

which together with (6) implies (38). [Note that the alternate application of Proposition 10 (in lieu of Proposition 11) would require that one replace (42) by a stronger bound, of the order of $\rho = (1/n)O(n)$, in order to deduce (38).]

References