On Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration

Victor Y. Y. Pan

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration

Victor Y. Pan

N° 2894

Mai 1996

THÈME 2

Rapport de recherche

ISSN 0249-6399

On Approximating Complex Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration

Victor Y. Pan

Thème 2 — Génie logiciel
et calcul symbolique
Projet SAFIR

Rapport de recherche n ˚ 2894 — Mai 1996 — 36 pages

Abstract: The known record complexity estimates for approximating polynomial zeros rely on geometric constructions on the complex plane, which achieve initial approximation to the zeros and/or their clusters as well as their isolation from each other, and on the subsequent fast analytic refinement of the initial approximations. We modify Weyl's classical geometric construction for approximating all the n polynomial zeros in order to more rapidly achieve their strong isolation. For approximating the isolated zeros or clusters of zeros, we propose a new extension of Newton's iteration to yield quadratic global convergence (right from the start), under substantially weaker requirements to their initial isolation than one needs in the known algorithms.

Key-words: Complex polynomial zeros, algorithms, computational complexity.

(Résumé : tsvp)

The results of this paper have been presented at the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, Virginia, January 1994.

Mathematics and Computer Science Department, Lehman College, CUNY, Bronx, NY 10468, USA. Email: vpan@lcvax.bitnet. Supported by NSF Grant CCR 9020690 and PSC-CUNY Awards Nos. 662478 and 664334.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Sur l'Approximation des Zéros Complèxes d'un Polynôme : Une Modification de l'Algorithme de Weyl par Quadtrees et une Amélioration de la Méthode de Newton

Résumé : Les meilleures bornes connues sur la complexité pour approcher les zéros d'un polynôme dépendent, prémièrement, de constructions géométriques dans le plan complexe qui donnent une approximation initiale aux zéros et/ou nuages de zéros tout en les séparant et, deuxièment, d'une convergence analytique rapide à partir de ces approximations. Nous modifions la construction géométrique classique de Weyl pour approcher tous les zéros d'un polynôme afin d'accélerer leur isolation. Pour approcher les zéros ou nuages isolés, nous proposons une extension nouvelle de la méthode de Newton avec une convergence quadratique globale (dès le debut). Les hypothèses sur la séparation initiale sont significativement plus faibles que dans les algorithmes connus.

Mots-clé : Zéros complexes de polynômes, algorithmes, complexité de calcul

1 Introduction.

The record upper bounds on the computational complexity of the classical problem of approximating complex zeros of an n-th degree polynomial $p(x)$ have been obtained by combining some *computational geometry constructions* for search and exclusion on the complex plane with some *analytical tools,* such as Newton's iteration and numerical integration [Sc82], [P85], [PR], [P87], [R], [BFKT], [P89], [B-OT], [BP91]. In this paper we continue using such technical patterns but also apply some new techniques and make further progress.

As in almost all of the cited papers (with the only exception of the important unpublished paper [Sc82]), we assume the arithmetic computational model, measuring the computational complexity by the number of arithmetic operations and comparisons involved; we also allow (relatively fewer) evaluations of the k-th roots of complex numbers for natural k, and our computation performed with $\mathbf{O}(bn)$ bits ensures the output precision of $\mathbf{O}(b)$ bits, for a polynomial $p(x) = \sum_{i=0}^{n} p_i x^i$ with $\|p(x)\| = \sum_{i=0}^{n} |p_i| = 1$. (We refer the reader to the argument in [P87] supporting the latter upper bound on the precision of computation by our algorithms, and we note that $o(bn)$-bit precision of computation cannot generally ensure the b-bit output precision, for approximating polynomial zeros, no matter which algorithm has been applied: for instance, deleting the (bn)-th bit of the x-free term of the polynomial $x^n - 2^{-bn}$ turns it into the polynomial x^n and thus changes the b-th bit of the root $x = 2^{-b}$.)

The record estimate for the arithmetic computational complexity of this problem, $\mathbf{O}(n^2 \log n \log(bn))$ operations, has been obtained in [P87] by means of first isolating the zeros of $p(x)$ and/or their clusters from each other (based on Weyl's geometric construction for search and exclusion on the complex plane, which in [P87] was complemented with the application of Turan's proximity test) and then rapidly refining the resulting rough approximations to the zeros (by means of the techniques of numerical integration and of the recursive splitting of $p(x)$ into smaller degree factors). Weyl's construction is a 2-dimensional extension of the bisection of a line and is also known as *quadtree* algorithm, effectively used in various areas of computing, for instance, in image processing, template matching, and the solution of the n-body problem of particle simulation [HS], [Ga], [Sa], [Se], [G]. We refer the reader to [P95] on the recent extension of Weyl's algorithm to the unsymmetric eigenvalue problem. Some elements of Weyl's approach also appeared in [Sc82] and [R], although [Sc82] uses Lehmer's (rather than Weyl's) construction, and [R] applies Schur-Conn's (rather than Turan's) proximity test and Newton's iteration for higher order derivatives of $p(x)$ (rather than numerical integration). Both algorithms [Sc82] and [R] support the order of n^3 arithmetic cost bounds, although further improvements are possible within both frameworks.

In this paper we revisit some fundamental techniques involved in the algorithms for approximating polynomial zeros, in particular, the geometric techniques for the isolation of the zeros and the analytical techniques for the refinement of the approximations to the isolated zeros. To link these two areas, we rely on the useful concept of an isolation ratio, due to [P87] and giving a quantitative measure for the isolation of polynomial zeros or their clusters from each other. Furthermore, we contribute new effective algorithms in both areas.

Our first contribution is *a new variant of Weyl's (quadtree) construction*, which enables us to isolate groups (clusters) of the zeros of $p(x)$, achieving a constant isolation ratio, $f \geq 8$ (say), at the cost of using $\mathbf{O}(n^2 \log n)$ operations. Having achieved such an initial isolation of (the clusters of) the zeros, one may apply various techniques for the subsequent rapid improvement of the approximation. In this paper we present new techniques for such an improvement.

This stage of our computation begins with performing $\mathbf{O}(n^2 \log n \log\log n)$ operations in order to lift the constant isolation ratio f so as to make it linear in n or $\sqrt{n}$. Specifically, we need to have an initial isolation ratio $f \geq (3+\gamma n)\sqrt{2}$ or $f \geq (2+\sqrt{\gamma n})\sqrt{2}$, where we may fix any $\gamma > 6$, in order to insure that our *new modifications of Newton's iteration* rapidly improve the initial approximations to (the clusters of) the zeros of $p(x)$ and converge with the quadratic rate, right from the start, so that we achieve the desired refinement of the initial approximations to the zeros (up to within the error bound 2^{-b}) in $\mathbf{O}(n^2 \log n \log b)$ operations. (For comparison, such a rapid convergence has been achieved in [R] for a distinct variant of Newton's iteration, but only for an initial isolation ratio as large as $80n^5$, which required to increase the work at the isolation stage, respectively.) In this way, we arrive at the overall arithmetic complexity estimate $\mathbf{O}(n^2 \log n \log(b \log n))$, for the approximation of all the n zeros of $p(x)$ within 2^{-b}, by using the precision of $\mathbf{O}(bn)$ bits. This gives alternative algorithms for establishing the complexity bound of [P87], which is the current record in terms of b and n and which is supported by $\mathbf{O}(bn)$-bit precision of computing (asymptotically optimum). [The arithmetic bound has been stated as $\mathbf{O}(n^2 \log n \log b)$ in [P87], which is justified if and only if the error bound 2^{-b} is not greater than some fixed positive power of $1/n$; the latter assumption, however, is very mild and covers all cases of interest.] Letting $m(s) = \mathbf{O}(s \log s \log\log s)$ denote the Boolean (bit-operation) complexity of multiplication of two integers modulo 2^s, we now immediately arrive at the bound $\mathbf{O}(n^2 \log n \log(b \log n) m(bn))$ on the Boolean complexity of approximating polynomial zeros. The latter bound can be slightly decreased [by the factor $\log(bn)$] if one performs the computations using the special techniques of binary segmentation, that is, splitting the 0-1 string that represents a binary value into several substrings (compare [FP], [Sc82], [P92a] and section 9 of chapter 3 of [BP94] on this subject).

The arithmetic complexity bound $\mathbf{O}(n^2 \log n \log(bn))$ (close to the bound of this paper) can be also obtained by making a hybrid of our modification of Weyl's geometric construction and the techniques of [Sc82] for splitting a polynomial into two lower degree factors. In [Sc82], the computational complexity has been estimated only in terms of the number of bit-operations involved, assuming variation of the precision of computation and binary segmentation, but simple inspection shows that the order of roughly n^3 arithmetic operations are required. Specifically, Schönhage in [Sc82] first applies some geometric techniques (which he calls splitting circle method), in order to reach an isolation ratio of the order $1 + 1/n$, and then applies FFT based numerical integration and Newton's iteration (in two versions) to split $p(x)$ into factors. The integration stage of splitting requires the order of n^3 arithmetic operations in [Sc82]. By applying our modification of Weyl's construction, rather than the splitting circle method of [Sc82], one may ensure a higher initial isolation ratio, of

$f = 8$ or $f = 4$ (say), and then reach a smaller arithmetic complexity bound (cited above) after some adjustment of the splitting algorithm of [Sc82] (also compare Remark 1.2 in this section).

The asymptotic bound $\mathbf{O}(n^2 \log n \log(b \log n))$ on the complexity of approximating the zeros of $p(x)$ is optimum in terms of b [R], but very recently (already after submission of this paper) has been improved in terms of n, to the nearly optimum level of $\mathbf{O}((\log n)^2((\log n)^2 + \log b)n)$ (see [P95a], [P,a]). For practical implementation, however, the algorithms of the present paper still have an advantage of being much simpler to code and to implement, beside being a very robust and defining an easily controllable iterative process.

Our techniques and our results can be extended to approximating the zeros of an analytic function in a square, if these zeros are isolated from all other zeros of the function.

For an extension to approximating the eigenvalues of a symmetric tridiagonal matrix, see [P93] and [PL].

We will present the results in the following order. Section 2 is devoted to some preliminaries. In section 3 we recall Weyl's exclusion algorithm (quadtree algorithm). In section 4 we define our main algorithm in the flowchart form, as a recursive process of isolation and contraction. Section 5 describes the geometric construction used at the isolation stages. In section 6 and Appendices D and E we present some analytic techniques needed at the contraction stages. Appendices A, B and C contain several auxiliary results needed in sections 3–6. Appendix F contains figures.

Remark 1.1 *In this paper, we have not cited numerous papers that proposed various techniques for approximating all the complex zeros of a polynomial but did not supply the competitive worst case estimates for the computational complexity. Among these techniques, we wish to single out the Durand-Kerner method and its various extensions and modifications, such as ones by Aberth, Maehly, Werner, Pasquini-Trigiante, Farmer-Loizou and by others (see [BP]), which have been proved to be practically most effective among all the known approaches to the problem of approximating all the complex zeros of a polynomial. These methods also require to obtain an initial set of approximations to the n zeros of $p(x)$, for which purpose one may apply the techniques of our section 5.*

Remark 1.2 *In some papers, notably in [Sc82] and [KS], the error of the approximation to the zeros of $p(x)$ is measured by the maximum magnitude $E(p)$ of the coefficients of the error polynomial $p(x) - p_n \prod_{j=1}^{n}(x - z_j^*)$, z_j^* denoting the computed approximations to the zero z_j of $p(x)$ and p_n being the leading coefficient of $p(x)$. In this case the problem is called* complex factorization *[versus* approximating the zeros *of $p(x)$, where the output error is measured by $E_z(p) = \max_j |z_j^* - z_j|$]. Recall from the example of the polynomial $x^n - 2^{-bn}$ that one needs to have $E(p) = \mathbf{O}(2^{-bn})$ in order to ensure that $E_z(p) \leq 2^{-b}$ in the worst case, and the algorithms of [Sc82] and [KS] support substantially smaller upper bounds on the complexity of the complex factorization than approximating the zeros. In particular, Schönhage in [Sc82] reaches the Boolean complexity bounds $\mathbf{O}((b + n \log n)n^2 \log(bn) \log\log(bn))$ and $\mathbf{O}((b + \log n)n^3 \log(bn) \log\log(bn))$ for the problems of complex factorization, with $E(p) \leq 2^{-b}$, and of approximating the zeros, with $E_z(p) \leq 2^{-b}$, respectively. [Note that the latter bound can be* *also reached (and insignificantly improved) by relying on the algorithms of the present paper and employing the same techniques of binary segmentation as in [Sc82].) A distinct approach of [KS] uses* $\mathbf{O}((b+n)n\log^2 n)$ *and* $\mathbf{O}(bn^2\log^2 n)$ *arithmetic operations and comparisons in order to solve the complex factorization problem and approximating the zeros problem, respectively [with the error bounds in terms of* $E(p)$ *and* $E_z(p)$*, respectively] provided that* $n=\mathbf{O}(b)$, $\log E(p) = -b$*. To reach these bounds, however, multipoint polynomial evaluation need be involved in the algorithm of [KS], which usually leads to numerical stability problems. (One may avoid these problems by using the known numerically stable polynomial evaluation algorithms; then, however, the above complexity bound should be increased by the factor* $n/\log n$*.) Technically, the algorithm of [KS] is very interesting since it exploits the path lifting method, originated in [S81], [S85] and more recently developed into a highly effective algorithm for a system of polynomial equations, [SS], [SS,a], [SS,b], [SS,c], [SS,d].*

Acknowledgment. A. Sadikou has made several useful comments on the original draft of this paper.

2 Definitions and an Auxiliary Algorithm.

We will present the complexity estimates by using the notation $\mathbf{O_A}(t)$, for $\mathbf{O}(t)$ operations, which may include arithmetic operations, pairwise comparisons of positive numbers and the evaluation of the k-th roots of complex numbers, for natural k. log denotes logarithms to the base 2. We will count the polynomial zeros with their multiplicities, not distinguishing between clustered and multiple zeros. By saying "computing a polynomial" we will mean "computing its coefficients" (unless we explicitly specify otherwise). $p(x)$ denotes a fixed polynomial of degree n,

$$p(x) = \sum_{i=0}^{n} p_i x^i = p_n \prod_{j=1}^{n} (x - z_j), \ \ p_n \neq 0. \tag{2.1}$$

Definition 2.1 *$D = D(X,R)$ denotes the disc of radius $\rho(D) = R$ with the center X on the complex plane; $S = S(X,R)$ denotes the square with the side length $2\rho(S) = 2R$ and with the vertices $X + R(1+\sqrt{-1})$, $X - R(1+\sqrt{-1})$, $X + R(1-\sqrt{-1})$, $X - R(1-\sqrt{-1})$.*

Definition 2.2 *Two complex sets U and V are called equivalent if they contain exactly the same zeros of $p(x)$. Transformation of a set U into its equivalent subset is called shrinking or contraction. If U denotes a square or a disc, we define its* **rigidity ratio,** *$r.r.(U)$, and its* **isolation ratio,** *$i.r.(U)$, as follows (see Figure 1): $r.r.(U) = inf(\rho(U^-)/\rho(U))$, $i.r.(U) = sup(\rho(U^+)/\rho(U))$, where $\rho(U)$ is defined in Definition 2.1 and where the infimum and the supremum are over all the squares (or discs) U^- and U^+ that are equivalent to the square (respectively, disc) U and such that U^+ and U are concentric, $U^- \subseteq U \subseteq U^+$. A disc or a square U and every its equivalent subset are* **f-isolated** *if $i.r.(U) \geq f$.*

Definition 2.3 $d(U) = \max_{z_g, z_h} |z_g - z_h|$ *for a complex set* U,
$d^*(U) = \max_{z_g, z_h} \max\{|Re\ z_g - Re\ z_h|, |Im\ z_g - Im\ z_h|\}$,
where $\max_{z_g, z_h}$ *denotes the maximum over all the pairs* z_g, z_h *of the zeros of* $p(x)$ *in* U *and we use the customary notation* $c = Re\ c + \sqrt{-1}\ Im\ c$, *with real* $Re\ c$ *and* $Im\ c$, *for any complex* c.

Proposition 2.1 $r.r.(S) = d^*(S)/(2\rho(S))$ *for a square* S; $r.r.(D) = d^*(D)/(c\rho(D))$, $\sqrt{2} \le c \le 2$, *for a disc* D.

Definition 2.4 *For a complex* X, *for an* $f > 1$ *and for a nonnegative* ε, *the disc* $D(X, \varepsilon)$ *is called* **an** f**-isolated** ε*-neighborhood of a zero* z_j *of* $p(x)$ *if this disc contains* z_j *and is* f*-isolated.*

Definition 2.5 $i(p(x), U)$, *the* **index** *of* $p(x)$ *in* U, *is the number of the zeros of* $p(x)$ *lying in a complex set* U *and counted with their multiplicities.*

Definition 2.6 *The* n *distances* $r_1(X) \ge r_2(X) \ge \cdots \ge r_n(X)$ *from a point* X *to the* n *zeros of* $p(x)$ *are called the* **root radii** *of* $p(x)$ *at* X; *in particular,* $r_s(X)$ *is called the* s**-th root radius** *of* $p(x)$ *at* X, *and we set* $r_s(X) = \infty$ *for* $s \le 0$, $r_s(X) = 0$ *for* $s > n$.

Proposition 2.2 $1/r_s(X)$ *equals the* $(n + 1 - s)$*-th root radius at* X *of the (shifted and reversed) polynomial* $(x - X)^n p(X + \frac{1}{x-X})$.

Definition 2.7 *For fixed positive* $\overline{R}$ *and* ε, *denote that*

$$b = \log(\overline{R}/\varepsilon), \tag{2.2}$$

provided that $\overline{R}$ *is an upper bound on the moduli of all the zeros of* $p(x)$,

$$\overline{R} \ge |z_j|, \ \ for\ j = 1, \ldots, n. \tag{2.3}$$

In particular(see [He], section 6.4a), the relations (2.3) hold for

$$\overline{R} = 2 \max_{h=1,\ldots,n} |p_{n-h}/p_n|^{1/h}. \tag{2.4}$$

Remark 2.1 *Hereafter, all the rectangles and squares on the complex plane have their sides parallel to the coordinate axes.*

Algorithm 2.1 (superscription of a square about a closed set) .
Input: *a closed set* U *on the complex plane.*
Output: *the side lengths and the real and imaginary coordinates of the center of the smallest rectangle containing the set* U *(and having its sides parallel to the coordinate axes).*
Computation: *Compute the maximum* M *and the minimum* m *of the real parts of the points of* U. *Output* $M - m$ *and* $\frac{M+m}{2}$. *Repeat the same computations for the set of the imaginary parts of the points of* U.

Proposition 2.3 *The two half-sums in the output of Algorithm 2.1 equal the real and imaginary parts of the center X of the minimum rectangle containing the set U. The two differences equal the lengths of the sides of this rectangle. The center X and the length $2r$ of the longer side define the smallest square $S(X,r)$ containing the set U.*

In this paper, we will use the known algorithms for the basic operations with polynomials [such as their multiplication and division with a remainder, discrete Fourier transform (DFT), scaling and the shift of the variable]; the arithmetic cost of these computations is $\mathbf{O_A}(n\log n)$ (see [AHU], [P92a], [BP94]).

Remark 2.2 *Our results stated for the unit disc $D(0,1)$ immediately extend to the disc $D(X,r)$, for any complex X and positive r: it suffices to shift and to scale the variable.*

3 Weyl's Exclusion Algorithm.

In this section, we will recall Weyl's exclusion algorithm, where we will incorporate Turan's proximity test of Appendix A ([He], pp. 517–521). Later on, we will use this algorithm in order to isolate the clusters of the zeros of $p(x)$, although historically such an algorithm has been first introduced for approximating the zeros.

Algorithm 3.1 (Weyl) *(see Figure 2).*
Input: *positive integers k, G, $n \geq k$, and $N \geq 32$, the coefficients of the polynomial $p(x)$ of (2.1), a complex X and a positive R such that the square $S(X,R)$ contains exactly k (not necessarily distinct) zeros of $p(x)$ and is $5^{1/N}$-isolated.*
Output: *a positive integer $H \leq 4n$ and complex values X_h, $h=1,\dots,H$, such that the union $\bigcup_{h=1}^{H} S(X_h, R/2^G)$ lies in $S(X,R)$ and contains k zeros of $p(x)$.*
Initialization: *Call the square $S(X,R)$* suspect *in Stage 0.*
Stage g**,** $g=1,\dots,G$. *Divide each square that is suspect in Stage $g<G$ (such a square has sides of length $R/2^g$) into four subsquares, with the side length $R/2^{g+1}$. Then, within the factor $5^{1/N}$ [which is less than 1.052 for $N=32$, see (A.2)], approximate the distances from the centers of the subsquares to the nearest zeros of $p(x)$. [Do this by applying Algorithm A.1 of Appendix A at the four centers of the four subsquares, where each center plays the role of the input value X of Algorithm A.1, whose other input values, N, n, $p_0,\dots,p_n$, are shared with Algorithm 3.1.] Call a square* suspect *in Stage g unless the latter proximity test proves that the square contains no zeros of $p(x)$. Output the centers X_h of all the squares $S(X_h,r)$ that are suspect in Stage G (where $r=R/2^G$). Output H, the overall number of the output squares. (The next remark shows that $H \leq 4n$.)*

Remark 3.1 *Each zero of $p(x)$ may make at most 4 squares suspect in Stage g, for any g [note that our proximity test computes $r_n(X)$ within 6% error, due to (A.1), (A.2) and since $N \geq 32$].*

Remark 3.2 *Instead of Algorithm A.1, any other proximity test can be applied. In particular, replacing Algorithm A.1 by algorithm supporting part a) of Corollary B.1 and based on the relations (B.8) of Appendix B would only require to increase the resulting overall bound on the arithmetic complexity by the small factor* $\log\log n$*. In fact, in spite of such a minor increase of the arithmetic complexity estimates, the latter proximity test is more suitable for practical implementation since it has better numerical stability.*

4 Combining Approximation and Isolation of Polynomial Zeros.

Our algorithm for polynomial zeros will rely on the solution of the following problem:

Problem 4.1
Input: *two real constants* $f > 1$ *and* $\varepsilon = 2^{-b} > 0$*, complex coefficients of a polynomial* $p(x)$ *of (2.1), a natural* k*,* $2 \le k \le n$*, and an* f*-isolated square* S *that contains exactly* k *zeros of* $p(x)$*.*
Output: *an integer* k_0*,* $0 \le k_0 \le k$*,* f*-isolated* ε_h*-neighborhoods of* k_0 *zeros* z_h *of* $p(x)$ *in* S *for* $\varepsilon_h \le \varepsilon$*,* $h = 1, \dots, k_0$*, an integer* $u \ge 0$ *(*$u \ge 2$ *if* $k_0 = 0$*) and* u *squares* $S_1, \dots, S_u$ *that are disjoint,* f*-isolated, and such that* $S_g \subseteq S$*,* $k_g = i(p(x), S_g) \ge 1$*,* $g = 1, \dots, u$*,* $\sum_{g=0}^{u} k_g = k$*.*

The computation of the indices of $p(x)$ in each square S_g, $g = 1, \dots, u$, can be based on Proposition C.1 of Appendix C.

Algorithm 4.1 *To compute* f*-isolated* ε_h*-neighborhoods of all the zeros* z_h *of* $p(x)$*, proceed as follows: Initially set* $k = n$*,* $S = S(0, \overline{R})$*, for* $\overline{R}$ *of (2.4), and solve Problem 4.1 for such a square* S*. Then recursively, for each output square* S_g*,* $g = 1, \dots, u$*, set* $S = S_g$ *and solve Problems 4.1 (for* $S = S_g$ *playing the role of the input square). Recursively repeat this process until the desired* f*-isolated* ε_h*-neighborhoods of all the zeros of* $p(x)$ *have been computed.*

Since $k_g < k$, for $g = 1, \dots, u$, we will arrive at the desired ε_h-neighborhoods of all the n zeros of $p(x)$ in S in at most $n - 1$ recursive steps of solving Problem 4.1. It remains to specify the solution of Problem 4.1 and to estimate its complexity.

We will partition the solution of Problem 4.1 into two stages:
Problem 4.1a.
Input: *as for Problem 4.1.*
Output: *either a common* f*-isolated* ε_h*-neighborhood of all the* k *zeros* z_h *of* $p(x)$ *in* S *for some* $\varepsilon_h \le \varepsilon$ *or, else, an equivalent subsquare* $S^* = S(X^*, R^*)$ *of* S*, such that*

$$e \ge 1/r.r.(S^*) \tag{4.1}$$

for a fixed positive e*.*

Problem 4.1b.
Input: *the output subsquare S^* of Problem 4.1a, satisfying (4.1).*
Output: *as for Problem 4.1.*

We will consider Problem 4.1b in the next section and Problem 4.1a in section 6.

5 Isolation of the Zeros.

In this section we will assume available a black box algorithm **(Algorithm 6.1)** that solves Problem 4.1a (see section 6) and will specify **Algorithm 5.1**, that is, the blocks of Algorithm 4.1 where Problem 4.1b is solved. We will proceed as in (Weyl's) Algorithm 3.1 applied to the input square $S^* = S(X^*, R^*)$ of Algorithm 5.1 (which is the output square of Algorithm 6.1), so that for every i we define the i-th stage of Algorithm 5.1 as identical to the i-th stage of Algorithm 3.1, except that we now add some blocks for the verification if the requirements to the output of Algorithm 5.1 have already been satisfied for some squares suspect in this stage; if so, we stop partitioning these squares and keep them invariant until the end of the computations, when we output these invariant squares. Specifically, for every g, we will partition the union of all the $w(g)$ squares that are suspect in Stage g into $v(g)$ maximal connected components, $C_1^{(g)}, \ldots, C_{v(g)}^{(g)}$, each component contains at least one zero of $p(x)$, so that $1 \leq v(g) \leq k$, $g = 1, 2, \ldots$; $v(1) = 1$, $S^* = C_1^1$. Let the $(1 + g_0)$th stage be the separation stage of Algorithm 5.1, that is,

$$v(g) = 1 \ for \ g = 1, 2, \ldots, g_0; \ v(g_0 + 1) > 1. \tag{5.1}$$

In every Stage g, for $g > g_0$, apply Algorithm 2.1 to the set of all the vertices of all the suspect squares of the component $C_u^{(g)}$, for $u = 1, \ldots, v(g)$, and arrive at the minimum square, $S_u^{(g)} = S(X_u^{(g)}, R_u^{(g)})$, covering $C_u^{(g)}$ (compare Remark 2.1 and Proposition 2.3). Call $C_u^{(g)}$ an *f-isolated component* if the square $S_u^{(g)}$ is equivalent to $C_u^{(g)}$ and if

$$i.r.(S_u^{(g)}) \geq f.$$

Rewrite the latter assumption as follows:

$$d_u^{(g)} = \min_{v \neq u}(\max\{|ReX_u^{(g)} - ReX_v^{(g)}|, |ImX_u^{(g)} - ImX_v^{(g)}|\} - R_v^{(g)}) \geq fR_u^{(g)}. \tag{5.2}$$

For each u, $u = 1, \ldots, v(g)$, test if

$$\sqrt{2}R_u^{(g)} \leq \min\{\varepsilon, d_u^{(g)}/f\}. \tag{5.3}$$

If so, output the disc $D(X_u^{(g)}, R_u^{(g)}\sqrt{2})$ as an f-isolated $(\sqrt{2}R_u^{(g)})$-neighborhood of the zeros of $p(x)$ lying in the component $C_u^{(g)}$ and define $C_v^{(g+i)} = S(X_u^{(g)}, R_u^{(g)}) = S_v^{(g+i)} = S(X_v^{(g+i)}, R_v^{(g+i)})$, for an appropriate $v = v(g + i)$, as an invariant component that remains unchanged in all Stages $g + i$ of Algorithm 5.1 for $i > 0$. Otherwise test if (5.2) holds. If so, output the square $S_u^{(g)}$ as an input square S for Algorithm 6.1 and define $C_v^{(g+i)} = S(X_u^{(g)}, R_u^{(g)}) = S_v^{(g+i)} = S(X_v^{(g+i)}, R_v^{(g+i)})$ for appropriate $v = v(g+i)$ as an invariant component that remains unchanged in all Stages $g+i$ of Algorithm 5.1 for $i > 0$. Apply Stage $g+1$ of Algorithm 3.1 to all the squares that lie in the remaining noninvariant components and that are suspect in Stage g. Stop the computation by Algorithm 5.1 when all the components have become invariant.

Let us next analyze Algorithm 5.1. Hereafter, $w(C_u^{(g)})$ denotes the number of squares that are suspect in Stage g and lie in the component $C_u^{(g)}$, so that

$$w(g) = \sum_{u=1}^{v(g)} w(C_u^{(g)}),\ \ g = 1, 2, \ldots .$$

Our next goal is to deduce that

$$W = \sum_{g=g_0+1}^{G} w(g) = \mathbf{O}((1 + \log f)k). \tag{5.4}$$

We define a tree T with $G + 1 - g_0$ levels of nodes, where g_0 is defined by (5.1). The $v(g)$ components $C_1^{(g)}, \ldots, C_{v(g)}^{(g)}$ are identified with the $v(g)$ nodes forming the $(g - g_0)$th level of the tree. The component $C_1^{(g_0)}$ is identified with the root of the tree, which lies at level 0. The leaves are identified with the invariant components $C_u^{(g)}$, such that (5.2) and/or (5.3) hold for the values $X_u^{(g)}, R_u^{(g)}$. The edges of the tree go from each component $C_u^{(g)}$ to all the components [of the $(g+1-g_0)$th level] formed by the squares that are suspect in Stage $g+1$ and lie in $C_u^{(g)}$. We let w denote $\sum w(C_u^{(g)})$ where the sum is over all the leaves $C_u^{(g)}$; we deduce from Remark 3.1 that

$$w \leq 4k. \tag{5.5}$$

Hereafter, we will assume that every zero of $p(x)$ that makes two or several squares suspect in Stage g shall lie in or coincide with the intersection of the boundaries of these squares. [Clearly, we may always move the zeros of $p(x)$ on the complex plane, so as to satisfy this assumption without any decrease of the value W of (5.4).] Then, each node $C_u^{(g)}$ that is not a leaf contains not more suspect squares than its children do, that is,

$$w(C_u^{(g)}) \leq \sum_{r(u)} w(C_{r(u)}^{(g+1)}), \tag{5.6}$$

where the summation is over all the children $C_{r(u)}^{(g+1)}$ of the node $C_u^{(g)}$. We now recall that, besides the leaves, none of the components $C_u^{(g)}$ for $g \geq g_0$ is f-isolated and deduce the following proposition:

Proposition 5.1 *Every component $C_u^{(g)}$, $g > g_0$, has an ancestor-fork (that is, an ancestor with at least two children) at level g_1, $g_1 = g - g_0 - \log\ ((f-1)w(C_u^{(g)})) + y + \delta$ for some $y > 0$, where $\delta = 0$ if $C_u^{(g)}$ is an invariant output component, $\delta = 1$ otherwise.*

Proof. Observe that $R_u^{(g)} \le R^* w(C_u^{(g)})/2^g$, whereas the distance from $X_u^{(g)}$ to the nearest zero of $p(x)$ lying outside $C_u^{(g)}$ is at least $R_u^{(g)} + 2R^*/2^{g_1+g_0}$, because such a zero of $p(x)$ must be separated from $X_u^{(g)}$ by both the square $S(X_u^{(g)}, R_u^{(g)})$ and some square $S(X, R^*/2^{g_0+g_1})$ discarded in Stage g_0+g_1. Consequently, $i.r.(S_u^{(g)}) \ge 1 + \frac{2^{g-g_1-g_0+1}}{w(C_u^{(g)})}$. Therefore, $i.r.(S_u^{(g)}) \ge f$ if $(f-1)w(C_u^{(g)}) \le 2^{g-g_1-g_0+1}$ or, equivalently, if $\log((f-1)w(C_u^{(g)})) \le g - g_1 - g_0 + 1$. On the other hand, $i.r.(S_u^{(g)}) < f$ unless $C_u^{(g)}$ is a leaf of T, and in the latter case $i.r.(S_v^{(g-1)}) < f$ for the parent $C_v^{(g-1)}$ of $C_u^{(g)}$. By combining the above relations we complete the proof. □

To derive (5.4), we will also use the following estimates:

Fact 5.1 *Every component $C_u^{(g)}$, for $u = 1, \dots, v(g)$, $g = 3, \dots, G$, is covered by*

$$s(g,u) \le 2 + \lfloor w(C_u^{(g)})/2 \rfloor$$

squares that are suspect in Stage $g-2$.

The proof is rather straightforward (although tedious): it appeals, on one hand, to the fact that a square that is suspect at Stage $g-2$ has the 4-fold increase of the side length versus the side length of a square that is suspect at Stage g and, on the other hand, to the connectivity of the component $C_u^{(g)}$ consisting of the chain of $w(C_u^{(g)})$ smaller suspect squares. To complete the proof, it essentially remains to classify all possible configurations of these $w(C_u^{(g)})$ squares (the worst case configuration is shown in Figure 3). We will omit further details but will note that for the proof of the asymptotic bound (5.4), it actually suffices to prove a weaker version of Fact 5.1, where Stage $g-2$ is replaced by Stage $g-\mathbf{O}(1)$ and the right side of the inequality of Fact 5.1 is turned into $\mathbf{O}(1) + a\, w(C_u^{(g)})$ for some fixed $a < 1$. The latter version of Fact 5.1 immediately follows from the 2 above observations (about the 4-fold increase of the side length and the component connectivity).

Corollary 5.1 $s(g,u) \le 0.75 w(C_u^{(g)})$ *if* $w(C_u^{(g)}) > 6$.

Now let $w_6 = \sum_{u,g} w(C_u^{(g)})$, $w_6^* = \sum_{u,g}^* w(C_u^{(g)})$ where $\sum$ and $\sum^*$ denote the sums in all the pairs g and u such that $g > g_0$, $1 \le u \le v(g)$, $w(C_u^{(g)}) \le 6$, provided that $\sum^*$ consists of forks and leaves, that is, includes no pairs (g,u) such that the node $C_u^{(g)}$ of the tree T has exactly one child.

By combining the bounds (5.5) and (5.6) with Corollary 5.1, we obtain that

$$W - w_6 \le 8w \le 32k, \tag{5.7}$$

and it remains to estimate w_6 in order to prove (5.4).

Proposition 5.1 implies that in the tree T every node $C_u^{(g)}$ such that $w(C_u^{(g)}) \le 6$ has at most $\log(6(f-1)) - 1$ its successive predecessors with a single child. Due to the bound (5.6), each of these predecessors contributes at most $w(C_u^{(g)})$ to the sum w_6. It follows that

$$w_6 \le \log(6(f-1)) w_6^*. \tag{5.8}$$

It remains to estimate w_6^*. We first observe that the tree T has at most k leaves; therefore, it has at most $k-1$ forks (the nodes with more than one child). The set of all the forks and leaves has a cardinality of at most $2k-1$ and has a subset of nodes $C_u^{(g)}$ such that $w(C_u^{(g)}) \leq 6$. This subset is formed by exactly w_6^* squares, each of which is suspect in at least one of Stages $g_0+1, \ldots, G$. Consequently,

$$w_6^* \leq 6(2k-1) = 12k - 6.$$

Combining this bound on w_6^* with (5.8) and (5.7), we arrive at (5.4).

We have deduced the bound (5.4) for a single application of Algorithm 5.1, which is actually called $\mathbf{O}(n)$ times in the process of performing Algorithm 4.1. In every application of Algorithm 5.1, its every output square $S_u^{(g)}$, satisfying the relation (5.2) but not (5.3) (that is, f-isolated but not embedded in an equivalent and f-isolated disc of radius at most ε), serves as the input square in the subsequent application of Algorithm 6.1, which, in turn, generally outputs an input square for Problem 5.1 and Algorithm 5.1. We then use the tree T generated in the latter application of Algorithm 5.1 to replace the respective leaf of the tree generated in the preceding application of Algorithm 5.1. In this way, we construct a single tree associated with all the $\mathbf{O}(n)$ applications of Algorithm 5.1 within Algorithm 4.1 and having at most n leaves. Proposition 5.1 and Fact 5.1 are also extended to the entire computational process, represented by the latter tree, and in this way we extend (5.4) to the same asymptotic estimate (with k replaced by n) for the total number of suspect squares processed in all these calls for Algorithm 5.1, provided that in each call for Algorithm 5.1 we only count the suspect squares processed after the first splitting of the single input component into disjoint connected components.

Finally, we complement this estimate with the bound

$$W_0 = \mathbf{O}(n + n \log e), \tag{5.9}$$

where e is defined by (4.1) and W_0 denotes the overall number of squares that are suspect in all Stages g such that $v(g) = 1,\ g \leq g_0$ [see (5.1)], (that is, $C_1^{(g)}$ is the single connected component output in Stage g), in all the applications of Turan-Weyl's Algorithm 3.1, within Algorithm 4.1.

To deduce (5.9) we first consider a single application of Algorithm 5.1 and recall that the length of the sides of the squares that are suspect in Stage g decreases by twice as g grows by 1. It is sufficient to consider the cases, where

$$w(C_1^{(g)}) = w(g) \leq 6, \tag{5.10}$$

since we may extend Fact 5.1 and Corollary 5.1 and also recall that

$$w(g) \leq 4k,\ \ g = 1, 2, \ldots, G. \tag{5.11}$$

We now combine (5.10) and Proposition 2.1 and deduce that

$$w_6^* \leq 6\lceil \log e \rceil + 15. \tag{5.12}$$

Indeed, due to (5.6) and (5.10), $w(C_1^{(g)})$ may grow with g at most in 5 stages g, for $1 \leq g \leq g^*$, which contributes the term $1+2+3+4+5=15$ on the right side of (5.12). On the other hand, as g increases by 1, $r.r.(S_1^{(g)})$ never decreases and grows by twice, if $w(C_1^{(g)})$ stays invariant. This implies (5.12), since $r.r.(S_1^{(g)})$ never exceeds 1 and since we initially have (4.1).

Combine Fact 5.1 and Corollary 5.1 with (5.11) and (5.12) and deduce the bound

$$\sum_{g=1}^{g_0} w(g) = \mathbf{O}(k + \log e),$$

for a single application of Algorithm 5.1, and similarly we arrive at (5.9) for all of at most $n-1$ applications of Algorithm 5.1 within Algorithm 4.1. More specifically, to deduce (5.9), we observe that there are at most $n-1$ forks in the tree associated with the entire Algorithm 4.1 (since this tree has at most n leaves), and that for each fork, we have to go through at most $15 + 6\lceil \log e \rceil$ its successive descendants before we reach either the next fork or a node C with $w(C) > 6$. We exclude the suspect squares associated with the node C such that $w(C) > 6$ since to them we may apply or extend Corollary 5.1 and may apply the bound (5.11). For other nodes, we arrive at (5.9), by $n-1$ times applying the bound (5.12).

We now recall that each proximity test in Algorithm 3.1 (one for each suspect square) can be performed by means of Algorithm A.1 at the cost bounded by $\mathbf{O_A}(n \log n)$. Therefore, from (5.4) and (5.9), we deduce the bound

$$\mathbf{O_A}((1 + \log f + \log e) n^2 \log n)$$

on the computational cost of all the applications of Algorithm 5.1 within Algorithm 4.1. In particular, for $e = \mathbf{O}(1)$, $\log f = \mathbf{O}(\log n)$, the latter cost bound turns into $\mathbf{O_A}((n \log n)^2)$, whereas for $e = \mathbf{O}(1)$, $f = \mathbf{O}(1)$, this cost bound turns into $\mathbf{O_A}(n^2 \log n)$.

6 Contraction of a Complex Square.

We will now present the following algorithm for Problem 4.1a, where we assume that $1/e \leq 0.24$, $f = f^*\sqrt{2}$, $f^* > 2 + 6n(e-1)/(e-2)$, and $f^* > \sqrt{50}$ (compare Remark 6.1 and the correctness proof in Appendix D), so that it suffices to set $e = 5$, $f = 3 + 8n\sqrt{2}$, or $e = 8$, $f = 3 + 7n\sqrt{2}$, or $e = 26$, $f = 3 + (25/4)n\sqrt{2}$:

Algorithm 6.1
Initialization: *Let σ denote a set of complex numbers available from the previous application of Algorithm 6.1 and chosen to be empty at the first application of this algorithm. Compute the f^*-isolated disc $D = D(x_0, R)$ whose boundary circle passes through the four vertices of the input square S. Compute the complex value $Y = x_0 + 2R$.*
Computations:

1. *Compute the value* $p'(Y)$. *If* $p'(Y) \neq 0$, *set* $t = Y$. *Otherwise, add the complex point* Y *to the set* σ *and choose a point* t *near* Y *such that* $p'(t) \neq 0$. *(It suffices to test at most* $n - |\sigma|$ *candidate points* $t \notin \sigma$ *to find one for which* $p'(t) \neq 0$, *where* $|\sigma|$ *denotes the cardinality of the set* σ. *Each candidate point* t *for which* $p'(t) = 0$ *is added to the set* σ.*)*

2. *Compute the value*
$$u = t - \frac{kp(t)}{p'(t)}. \tag{6.1}$$

3. *If* $p(u) = 0$, *set* $X = u$ *and go to Stage 10. Otherwise, apply Algorithm A.1 of Appendix A (Turan's proximity test), for* $X = u$, $N \geq 32$, *which outputs* $r = r(u)$ *and* $r^*(u) = 5^{1/N} r$, *or achieve the same goal based on the relations of (B.8) and on the iteration (A.3), in the latter case by using slightly more arithmetic operations [by factor* $O(\log\ \log\ n)$*] but achieving better numerical stability.*

4. *Compute* $v_h = v_h(u) = u + 8\delta_h r^*(u)$, *for* $\delta_h = (\sqrt{-1})^h$, $h = 0, 1, 2, 3$.

5. *If* $p(v_h) = 0$ *for some* h, $0 \leq h \leq 3$, *set* $X = v_h$ *and go to Stage 10. Otherwise, apply Algorithm A.1 for* $X = v_h$, $N = 32$, $h = 0, 1, 2, 3$; *output* $r(v_h)$, $r^*(v_h)$, $h = 0, 1, 2, 3$ *[or apply the cited alternative of Algorithm A.1, based on the relations (B.8) and the iteration (A.3)].*

6. *Compute* $\tilde{r}_h = r^*(v_h) + r^*(v_{h+2})$, *for* $h = 0, 1$. *If* $\tilde{r}_h \leq 15 r^*(u)$ *for* $h = 0$ *or* $h = 1$, *then go to Stage 10 for* $X = u$.

7. *Otherwise, for every* h, $h = 0, 1, 2, 3$, *write* $D_h = D(v_h, r_n(v_h))$, *observe that the pair of the discs* D_h *and* $D_{h+1 \bmod 4}$ *has two points* α_h *and* β_h *of the intersection of their boundaries [this follows from the comparison of the distances*
$$|v_h - v_{h+1 \bmod 4}| = 8\sqrt{2} r^*(u), \ \ h = 0, 1, 2, 3, \tag{6.2}$$
with the bounds (E.1) and (E.3) of Appendix E], and choose one of these two points, α_h, *that lies closer to* u *(see Figure 4).*

8. *Compute the complex* c *and the nonnegative* ρ *satisfying*
$$2Re\ c = \min_h Re\ \alpha_h + \max_h Re\ \alpha_h,$$
$$2Im\ c = \min_h Im\ \alpha_h + \max_h Im\ \alpha_h,$$
$$\rho = \max_h |c - \alpha_h|.$$
Here and hereafter we use the notation $\min_h$ *and* $\max_h$ *for the minimums and maximums in* h, *for* $h = 0, 1, 2, 3$, *and* $z = Re\ z + \sqrt{-1}\ Im\ z$ *for all complex* z. *[Note that the closed disc* $\tilde{D} = D(c, \rho)$ *contains* α_0, α_1, α_2, α_3 *and, consequently, at least*

one zero of $p(x)$; furthermore, the disc $\tilde{D}$ is f-isolated, for some fixed $f = f(N) > 1$, in particular, it is f-isolated for $f > 2.38$, if N is sufficiently large (see Claim E.1 and Remark E.1 in Appendix E).]

9. *Apply one of the two algorithms supporting Proposition C.1 of Appendix C (compare Remark C.1) in order to compute $i(p(x), \tilde{D})$, the index of $p(x)$ in $\tilde{D}$. If $i(p(x), \tilde{D}) < k$ (that is, if the disc $\tilde{D}$ does not contain all the k zeros of $p(x)$ lying in the input square S), go to Stage 10 for $X = c$. Otherwise, set $Y = c + 2\rho$ and go to Stage 1.*

10. *Shift to the polynomial $q(y) = p(y + X)$, set $s = n + 1 - k$, $\Delta = 0.01$, and apply Algorithm B.3 of Appendix B to compute an upper bound $\hat{r}$ on r_s for $q(y)$, such that $r_s \le \hat{r} \le 1.01 r_s$. Then*

    a) *if $\hat{r} \le \varepsilon$, output the disc $D(X, \hat{r})$ as a common f-isolated $\hat{r}$-neighborhood of all the k zeros of $p(x)$ lying in the input square S and stop;*

    b) *otherwise, set $X^* = X$, $R^* = \hat{r}$, output the square $S^* = S(X^*, R^*)$ [satisfying (4.1) for $1/e < 0.24$, that is, for $e > 25/6$] and go to Algorithm 5.1.*

In both cases the set σ is saved for the next application of Algorithm 6.1.

Let us next show that $r.r.(D(X,\hat{r})) > 0.24 \ge 1/e$ in the case of the transition to Stage 10 from Stage 6. (Similarly, the lower bound 0.24 follows in the case of the transition to Stage 10 from Stage 9, where c and ρ play the role of u and $r^*(u)$; in the case of the transition to Stage 10 from Stages 3 and 5, a similar but simpler argument gives a stronger bound of $r.r.(D(X,\hat{r})) > 0.49$.) By the definition of $\tilde{r}_h$ (see Stage 6), the distance between the two zeros of $p(x)$ that are the closest to v_h and v_{h+2}, respectively, is at least $|v_h - v_{h+2}| - \tilde{r}_h = 16r^*(u) - \tilde{r}_h$, which is not less than $r^*(u)$ if $\tilde{r}_h \le 15r^*(u)$, that is, in the case of the transition from Stage 6 to Stage 10. Therefore, in this case, $d^*(D(X,\hat{r})) \ge r^*(u)$, $X = u$, so that $d^*(D(X,\hat{r})) \ge \hat{r}/2$ if $\hat{r}/2 \le r^*(u)$. On the other hand, by the virtue of Theorem A.1, $p(x)$ has a zero $u + cr^*(u)$ for $|c| \le 1$, so that $d^*(D(X,\hat{r})) \ge r_s - r^*(u) \ge \hat{r}/1.01 - r^*(u)$, for the r_s defined at Stage 10. If $r^*(u) < \hat{r}/2$, then it follows that $d^*(D(X,\hat{r})) > \hat{r}(1/1.01 - 1/2) = \frac{99}{202}\hat{r}$. We have shown that this bound also holds if $r^*(u) \ge \hat{r}/2$. Therefore, $d^*(D(X,\hat{r})) > 0.24\hat{r}$, and hence, due to Proposition 2.1, $r.r.(D(X,\hat{r})) > 0.24$.

Correctness of the algorithm now follows from rapid (quadratic) convergence of the value u of (6.1) to the smallest disc, D_{min}, equivalent to the input square; such convergence is shown in Appendix D.

Remark 6.1 *If $k = n$, then we do not have to apply Algorithm 6.1, since we may first compute and output $X = -p_{n-1}/(np_n) = \sum_{j=1}^{n} z_j/n$ and then apply the algorithms of Appendix B to compute and to output a desired approximation R^* to $r_{n+1-k}(X)$.*

Algorithm 6.1 is recursively applied in its combination with Algorithm 5.1, so that all the input squares of Algorithm 6.1 can be arranged in the form of a tree with $\mathbf{O}(n)$ nodes. In each recursive step, one may concurrently apply Algorithm 6.1 to all the squares (nodes) of the current level of the tree starting with the root, at the first step.

The computational cost of performing Algorithm 6.1 is estimated as follows:

Stages 1, 2: $\mathbf{O_A}(n)$ [dominated by the cost of computing $\mathbf{O}(1)$ values of $p(x)$ and $p'(x)$];

Stages 3, 5: $\mathbf{O_A}(n \log n)$ (the cost of $\mathbf{O}(1)$ applications of Algorithm A.1);

Stages 4, 6, 7, 8: $\mathbf{O_A}(1)$;

Stages 9: $\mathbf{O_A}(n \log n)$, since the disc D computed at Stage 8 is f-isolated for $f > 2$;

Stage 10: $\mathbf{O_A}(n \log n)$ for an application of Algorithm B.3, due to Proposition B.2 and since the input disc is f-isolated for $f > 2 + 6n$; the same estimate, $\mathbf{O_A}(n \log n)$, holds for the complexity of shifting to the polynomial $q(y) = p(y + X)$.

According to Appendix D, we have quadratic convergence of the value u of (6.1) to the disc D_{min}; the approximation to the zeros of $p(x)$ in D_{min} is then improved in the subsequent applications of Algorithms 5.1 and 6.1. Due to the quadratic convergence of Algorithm 6.1, we need $\mathbf{O}(\log\log(R/\varepsilon))$ recursive calls to its Stage 2, in order to decrease the approximation error bound from R to ε. Based on this observation, we arrive at the overall cost bound $\mathbf{O_A}(n^2 \log n \log\log(R/\varepsilon))$ at the stages of refining the initial approximation of all the zeros of $p(x)$, which turns into the bound $\mathbf{O_A}(n^2 \log n \log b)$, for $b = \log(R/\varepsilon)$.

By combining the bounds of sections 5 and 6, for $e > 25/6$, $f > 2\sqrt{2} + 6n\sqrt{2}(e-1)/(e-2)$, we arrive at the cost bound $\mathbf{O_A}(n^2 \log n \log(bn))$, for approximating all the zeros of $p(x)$ within $\varepsilon = 2^{-b}$. The cost bound can be improved, to $\mathbf{O_A}(n^2 \log n \log(b \log n))$, if we apply Algorithm 5.1, say, for $f = 8$, and then lift f to $2\sqrt{2} + 6n\sqrt{2}(e-1)/(e-2)$, by means of the recipe of Remark A.2 of Appendix A.

To conclude this section, we will show a modification of Algorithm 6.1, where the expression (6.1), is replaced by the following one:

$$u = t - \frac{k}{q}, \ q = \frac{p'(t)}{p(t)} + \sum_j^{*} \frac{1}{z_j^* - t} \tag{6.3}$$

and where z_j^* denotes the current approximation to the zero z_j of $p(x)$, whereas $\sum^*$ denotes the sum in j over all the natural j corresponding to the zeros z_j of $p(x)$ that lie outside the input square S and the initial disc D. Instead of avoiding the points x and t that annihilate $p'(x)$, we shall now avoid the points t that annihilate $q = q(t)$; we will use a strategy similar to one of Algorithm 6.1. The presented modification of Algorithm 6.1 will be called **Algorithm 6.1a** and will always be applied to the largest of the currently unprocessed suspect squares. [This choice should insure the greatest acceleration of the convergence to the disc D_{min}.] The analysis of Appendix D shows that the quadratic convergence of Algorithm 6.1a can be achieved already for the input square S with an isolation ratio $f = f^*\sqrt{2}$, $f^* > 2 + \sqrt{6n(e-1)/(e-2)}$ [see (D.16)], so that it would suffice to set $e = 5$, $f = 3 + \sqrt{16n}$, or $e = 8$, $f = 3 + \sqrt{14n}$, or $e = 26$, $f = 3 + \sqrt{12.5n}$. To arrive at such a smaller initial isolation ratio, we need roughly by twice fewer iterations of Algorithm 5.1. This saving should actually be weighted against the additional arithmetic operations required in order to compute the value u via (6.3) [rather than via (6.1)]; for each such an evaluation, we need $3(n-k)$ additional operations.

References

[Ah] L. Ahlfors, *Complex Analysis*, McGraw-Hill, New York, 1979.

[AHU] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Mass., 1976.

[BFKT] M. Ben-Or, E. Feig, D. Kozen, P. Tiwari, A Fast Parallel Algorithm for Determining All Roots of a Polynomial with Real Roots, *SIAM J. on Computing*, 17(1989) 6, pp. 1081–92 (proceedings version in *Proc. 18th Ann. ACM Symp. on Theory of Computing* (1986), pp. 340–349, ACM Press, New York).

[B-OT] M. Ben-Or, P. Tiwari, Simple Algorithm for Approximating All Roots of a Polynomial with Real Roots, *J. of Complexity*, 6(1990) 4, pp. 417–442.

[BM] A. Borodin, I. Munro, *The Computational Complexity of Algebraic and Numerical Problems*, American Elsevier, New York, 1975.

[BP86] D. Bini, V. Y. Pan, Polynomial Division and Its Computational Complexity, *J. of Complexity*, 2(1986), pp. 179–203.

[BP91] D. Bini, V. Y. Pan, Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem, *Proc. 2-nd Ann. ACM-SIAM Symposium on Discrete Algorithms* (1991), pp. 384–393, ACM Press, New York.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrices Computations, vol. 1: Fundamental Algorithms,* Birkhauser, Boston, 1994.

[BP] D. Bini, V. Y. Pan, *Polynomial and Matrices Computations, vol. 2, Selected Topics*, Birkhauser, Boston, to appear.

[DH] B. Dejon, P. Henrici (editor), *Constructive Aspects of the Fundamental Theorem of Algebra*, Wiley, London, 1967.

[FP] M. J. Fischer, M. S. Paterson, String Matching and Other Products, *SIAM-AMS Proc.*, 7(1974), pp. 113–125.

[G] L. Greengard, *Rapid Evaluation of Potential Fields in Particle Systems*, M.I.T. Press, Cambridge, 1988.

[Ga] I. Gargantini, An Effective Way to Represent Quadtrees, *Communication of the ACM*, 25(1982) 12, pp. 905–910.

[H] A. S. Householder, *The Numerical Treatement of a Single Nonlinear Equation*, McGraw-Hill, New York, 1970.

[He] P. Henrici, *Applied and Computational Complex Analysis*, Wiley, New York, 1974.

[HS] G. M. Hunter, K. Steiglitz, Operations on Images using Quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1(1979)* 2, pp. 145–153.

[KS] M.-H. Kim, S. Sutherland, Polynomial Root-finding Algorithms and Branched Covers, *SIAM J. on Computing*, 23(1994), 2, pp. 415–436.

[M] M. Mignotte, An Inequality about Factors of Polynomials, *Math. of Computations*, 28(1974), pp. 1153–1157.

[P85] V. Y. Pan, Fast and Efficient Algorithms for Sequential and Parallel Evaluation of Polynomial Zeros and of Matrix Polynomials, *Proc. 26th Ann. IEEE Symp. on Foundation of Computer Science* (1985), pp. 522–531, IEEE Computer Science Press.

[P87] V. Y. Pan, Sequential and Parallel Complexity of Approximate Evaluation of Polynomial Zeros, *Computers and Math. (with Applications)*, 14(1987) 8, pp. 591–622.

[P89] V. Y. Pan, Fast and Efficient Parallel Evaluation of the Zeros of a Polynomial Having Only Real Zeros, *Computers and Math (with Applications)*, 17(1989) 11, pp. 1475–1480.

[P90] V. Y. Pan, Parallel Least-Squares Solution of General and Toeplitz-like Linear Systems, *Proc. 2nd Ann. ACM Symp. on Parallel Algorithms and Architectures* (1990), pp. 244–253, ACM Press, New York.

[P92a] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review*, 34(1992) 2, pp. 225–262.

[P92b/94] V. Y. Pan, New Resultant Inequalities and Complex Polynomial Factorization, *Proc. ISTCS' 92, Springer's Lecture Notes in Computer Science*, 601(1992), pp. 122–136, and *SIAM J. on Computing*, 23 (1994) 5, pp. 934–950.

[P93] V. Y. Pan, Accelerated Solution of the Symmetric Tridiagonal Eigenvalue Problem, Tech. Report TR 93–016, *Intern. Computer Science Institute*, Berkeley, CA, 1993.

[P95] V. Y. Pan, Weyl's Quadtree Algorithm for the Unsymmetric Eigenvalue Problem, *Applied Math. Letters*, 8 (1995) 5, pp. 87–88.

[P95a] V. Y. Pan, Optimal (up to Polylog Factor) Sequential and Parallel Algorithms for Approximating Polynomial Zeros, *in Proc. 27th Annual ACM Symposium on the Theory of Computing*, (1995), pp. 741–750, ACM Press, New York.

[P,a] V. Y. Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros, *Computers and Math. (with Applications)*, (1996),to appear.

[PL] V. Y. Pan, E. Linzer, A New Approach to Bisection Acceleration for the Symmetric Eigenvalue Problem, manuscript, 1995.

[PR] V. Y. Pan, J. H. Reif, Some Polynomial and Toeplitz Matrix Computations, *Proc. 28th Ann. IEEE Symp. on Foundation of Computer Science* (1987), pp. 173–184, IEEE Computer Science Press.

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

[S81] S. Smale, The Fundamental Theorem of Algebra and Complexity Theory, *Bull. of the Amer. Math. Soc.*, 4(1981) 1, pp. 1–36.

[S85] S. Smale, On the Efficiency of Algorithms of Analysis, *Bull. of the Amer. Math. Soc.*, 13 (1985) 2, pp. 87–121.

[Sa] H. Samet, The Quadtree and Related Hierarchical Data Structures, *Computing Surveys*, 16(1984) 2, pp. 187–260.

[Sc82] A. Schönhage, The Fundamental Theorem of Algebra in Terms of Computational Complexity, Manuscript, *Dept. of Math., Univ. of Tübingen*, Tübingen, Germany, 1982.

[Sc85] A. Schönhage, Quasi-GCD Computations, *J. of Complexity*, 1(1985), pp. 118–137.

[Se] H. Senoussi, A Quadtree Algorithm for Template Matching on Pyramid Computer, *Theoretical Computer Science*, 136(1994), pp. 387–417.

[SS] M. Shub, S. Smale, Complexity of Bezout's Theorem I: Geometric Aspects, *J. of the Amer. Math. Soc.*, 6(1993), pp. 459–501.

[SS,a] M. Shub, S. Smale, Complexity of Bezout's Theorem II: Volumes and Probabilities, *Computational Algebraic Geometry* (F. Eyssette, A. Galligo, Editors), *Progress in Mathematics*, 109(1993), pp. 267–285, Birkhauser.

[SS,b] M. Shub, S. Smale, Complexity of Bezout's Theorem III: Condition Number and Packing, *J. of Complexity*, 9(1993), pp. 4–14.

[SS,c] M. Shub, S. Smale, Complexity of Bezout's Theorem IV: Probability of Success, Extensions, *SIAM J. on Numerical Analysis*, to appear.

[SS,d] M. Shub, S. Smale, Complexity of Bezout's Theorem V: Polynomial Time, *Theoretical Computer Science*, 133 (1994) 1, pp. 141–164.

[Tu] P. Turan, *On a New Method of Analysis and Its Applications*, Wiley and Sons, New Jersey, 1984.

Appendices

A Squaring the Zeros and an Isolation Ratio. Turan's Proximity Test.

In this appendix, we recall Turan's proximity test, which enables us to compute the distance from a complex point X to the nearest zero of $p(x)$, with a relative error at most $5^{1/N}-1$ and at the cost $\mathbf{O_A}(n(1+\log N)\log n)$. If $z_1, z_2, \ldots, z_n$ denote the zeros of $p(x)$, in the order of nonincreasing their absolute values, and if s_K denotes $\sum_{j=1}^{n} z_j^K$, then, clearly, $|z_1| = r_1(0)$, $1 \leq \frac{n|z_1|^K}{|s_K|}$ for $K \geq 0$. By using some sophisticated tools from the number theory, Turan [Tu] proved an *effective upper bound* on the latter ratio, namely, $r_1(0) \leq 5^{1/N} \max_{g=1,\ldots,n} |s_{gN}/n|^{1/N}$ for all natural N. For large N, the value $5^{1/N}$ is close to 1, and we may closely approximate $r_1(0)$ via the computation of the power sums s_{gN}. For a proximity test at a point X, we need to know the value $r_n(X)$, rather than $r_1(0)$, but $r_n(X)$ for $p(x)$ equals $r_1(0)$ for $q(y) = y^n p(X + 1/y)$, where $y = x - X$ (Proposition 2.2).

Algorithm A.1 (Turan's proximity test) .
Input: *Natural N and n, the coefficients of polynomial $p(x)$ of (2.1), having degree n, and a complex number X that is not a zero of $p(x)$.*
Output: *Two nonnegative numbers $r = r(X)$ and $r^* = r^*(X) = 5^{1/N} r$ such that*

$$1 \leq r^*/r_n(X) \leq 5^{1/N},\ 1 \leq r_n(X)/r \leq 5^{1/N},\ r_n(X) = \min_{j=1,\ldots,n} |z_j - X|. \tag{A.1}$$

Stage 1. *Compute the values of the power sums,*

$$s_{gN} = \sum_{j=1}^{n} y_j^{gN},\ y_j = 1/(z_j - X),\ j = 1, \ldots, n.$$

Stage 2. *Compute and output $r^* = \left[\max_{g=1,\ldots,n} |s_{gN}|^{1/(gN)}\right]^{-1}$ and $r = r^*/5^{1/N}$.*

Theorem A.1 *[Tu, p. 299]. The output values r and r^* of Algorithm A.1 satisfy (A.1).*

It suffices for our purpose to choose $N = 32$; then

$$1.051581 < 5^{1/N} < 1.051582. \tag{A.2}$$

Turan chooses $N = 2^h$ to be a power of 2, and here is his algorithm for Stage 1:

Subalgorithm A.1.

Stage a). *Shift the variable by setting $y = x - X$ and compute the coefficients of the n - th degree polynomial $q(y)$ with the zeros $y_j = 1/(z_j - X)$, $j = 1, \dots, n$, such that*

$$p(x) = p(X + y) = \sum_{i=0}^{n} p_i(X) y^i,$$

$$q(y) = y^n p(X + 1/y) = \sum_{i=0}^{n} p_i(X) y^{n-i} = p_0(X) \prod_{j=1}^{n} (y - y_j).$$

Stage b), *(Dandelin-Lobachevsky-Graeffe, [H]). Let $q_0(y) = q(y)/p_0(X)$ and successively compute the coefficients of the polynomials*

$$q_{i+1}(y) = (-1)^n q_i(\sqrt{y}) q_i(-\sqrt{y}),\ i = 0, 1, \dots, h-1. \tag{A.3}$$

Stage c). *Numerically compute the power sums s_{gN} for $g = 1, \dots, n$, by solving the triangular Toeplitz system of Newton's identities in the variables $s_N, s_{2N}, \dots, s_{nN}$:*

$$q_{n,h} s_N + q_{n-1,h} = 0,$$

$$q_{n,h} s_{2N} + q_{n-1,h} s_N + 2 q_{n-2,h} = 0,$$

$$\vdots$$

$$q_{n,h} s_{nN} + q_{n-1,h} s_{(n-1)N} + \dots + n q_{0,h} = 0.$$

We note that each step i of Stage b) squares the zeros of the polynomial $q_i(y)$, so that

$$q_h(y) = \prod_{j=1}^{n} (y - y_j^N) = \sum_{i=0}^{n} q_{i,h} y^j,\ N = 2^h. \tag{A.4}$$

This makes the iteration (A.3) a powerful tool for separating the zeros of $p(x)$ from each other, because the logarithm of the ratio $|y_j/y_i|$ grows linearly in N, if $|y_j| > |y_i|$; furthermore, higher powers of the absolutely and strictly largest zero of $p(x)$ dominate in the respective power sums of all the zeros.

At Stage a), we just shift the variable x; every iteration i of Stage b) is a polynomial multiplication; Stage c) of solving a triangular Toeplitz system amounts to polynomial division (see [P92a]). Thus the overall cost of performing Algorithm A.1 is

$$\mathbf{O_A}(n(1 + h) \log n),\ h = \log N. \tag{A.5}$$

Remark A.1 *Each iteration step (A.3) squares* $i.r.(D(0,r))$ *for any positive* r. *This enables us to increase an* $i.r.(D)$ *from* $f>1$ *to* f^{2^h} *at the cost* $\mathbf{O_A}((h+\delta)n\log n)$, *where* $\delta=0$ *if the disc* D *has center 0.*

Remark A.2 *By applying Remark A.1 for* $h=\mathbf{O}(\log\log n)$, *we may increase the ratio* $f=\mathbf{O}(1)$ *(say,* $f=4$*) to* $f\geq 2\sqrt{2}+6n\sqrt{2}(e-1)/(e-2)$, *at the overall cost bounded by* $\mathbf{O_A}(n^2\log n\log\log n)$, *for the computation performed simultaneously for several discs that together contain all the zeros of* $p(x)$. *Thus, remaining within this cost bound, we may always satisfy the assumptions required for the application of Algorithm 6.1. Then we may compute isolated* ε_h^**-neighborhoods of the* d*-th powers of all the zeros of* $p(x+u)$ *for a pair or a triple of fixed shift values* u, *for* $d=\mathbf{O}(\log n)$, *and for appropriate positive* ε_h^*. *For every* u *and for every* ε_h^**-neighborhood of a zero* z_h^d *of* $q(x)$, *we then compute* d *candidate* ε_h^**-neighborhoods of* z_h, *for* $\varepsilon_h^*\leq\varepsilon$, *of which we need to select a true* ε_h*-neighborhood of* z_h. *We choose sufficiently small* ε_h^* *(they only need to be moderately small relative to* ε*) and appropriate shifts* u, *so as to identify unique* ε_h*-neighborhoods of* z_h, *for all* h, *by intersecting the candidate* ε_h*-neighborhoods (for all fixed* h *and for all the shift values* u*).*

B Root Radii Computations.

In this appendix, we will approximate the root radii $r_s(X)$, for $s=1,\ldots,n$, by following and a little simplifying [Sc82], section 14. We will keep assuming that $X=0$ (otherwise, we would shift the variable by letting $y=x-X$) and will denote $r_s=r_s(X)$,

$$r_0=\infty,\ r_{n+1}=0. \tag{B.1}$$

Consider the two following tasks (note the redundancy in their statements):
Task r. *Given positive* r *and* Δ, *find a (generally non-unique) integer* s *such that*

$$r_{s+1}/(1+\Delta)<r<(1+\Delta)r_s. \tag{B.2}$$

Task s. *Given a positive integer* s, $1\leq s\leq n$, *and a positive* Δ, *find a positive* r *such that*

$$r/(1+\Delta)<r_s<(1+\Delta)r. \tag{B.3}$$

We will solve Tasks r and s for $1+\Delta=2n$. The extension to an arbitrary positive Δ is immediate, by means of

$$g=g(\Delta)=\lceil\log(\frac{\log(2n)}{\log(1+\Delta)})\rceil \tag{B.4}$$

iteration steps (A.3); indeed, such an iteration step amounts to squaring $1+\Delta$ in the context of Tasks r and s, and we have $(1+\Delta)^{2^g}\geq 2n$, under (B.4). The computational cost of performing these iteration steps (as well as the cost of shifting the variable x, if needed) should be added to the overall cost of the solution, of course. Note that

$$g(\Delta)=0\ if\ 1+\Delta\geq 2n;\ g(\Delta)=\mathbf{O}(\log\log n)\ if\ 1/\Delta=\mathbf{O}(1), \tag{B.5}$$

$$g(\Delta) = \mathbf{O}(\log n) \text{ } if \text{ } 1/\Delta = n^{\mathbf{O}(1)}. \tag{B.6}$$

Most frequently, we need to solve Task s for $s = n$, and we have the following corollary of Theorem A.1 [compare (A.5)]:

Corollary B.1 *a) For $s = 1$ and $s = n$, Task s can be solved by means of the application of Theorem A.1 at the cost $\mathbf{O_A}(n(1+g)\log n)$, where g is defined by (B.4)-(B.6).*
b) Moreover, the cost decreases to $\mathbf{O_A}(n \log n)$ if $1/\Delta \le \mathbf{O}(1)$.

For part a) we have an alternative simpler proof from [Sc82]. Recall the well-known inequalities (compare [He], pp.451, 452 and 457):

$$t_1^*/n \le r_1 < 2t_1^*, \; t_1^* = \max_{k>0} |p_{n-k}/p_n|^{1/k}. \tag{B.7}$$

Apply Proposition 2.2 for $X = 0$ and extend the bounds (B.7) as follows:

$$t_n^*/2 \le r_n < nt_n^*, \; t_n^* = \min_{k>0} |p_0/p_k|^{1/k}. \tag{B.8}$$

Therefore, $r = t_1^*\sqrt{2/n}$ is a solution to Task s for $s = 1$, whereas $r = t_n^*\sqrt{n/2}$ is a solution to Task s for $s = n$; in both cases, we may choose any Δ such that $1 + \Delta > \sqrt{2n}$. This can be extended to the solution of Task s, for $s = 1$ and $s = n$ and for an arbitrary positive Δ, at the cost $\mathbf{O_A}(ng \log n)$ of g iteration steps (A.3), where g is defined by (B.4)-(B.6). This implies the cost bound of part a) of Corollary B.1.

We also need to solve Task s for $1 < s < n$ at Stage 10 of Algorithm 6.1 and Task r in Remark C.1. Next, we will show solution algorithms relying on the following useful and elegant result:

Theorem B.1 *([He, pp. 458–462], [Sc82]). If $1 \le m \le n$ and if $|p_{n+1-m-i}/p_{n+1-m}| \le av^i$ for $i = 1, \dots, n+1-m$, then $r_m < m(a+1)v$.*

Proof. Due to Van Vleck's theorem ([He], p. 459), we have

$$|p_{n+1-m}|r_m^{n+1-m} \le \binom{n}{n+1-m}|p_0| + \binom{n-1}{n-m}|p_1|r_m + \dots + \binom{m}{1}|p_{n-m}|r_m^{n-m}.$$

Divide both sides by $|p_{n+1-m}|r^{n+1-m}$, apply the assumed bound on the ratios $|p_{n+1-m-i}/p_{n+1-m}|$, and deduce for $x = v/r_m$ that

$$1 \le ax^{n+1-m}\binom{n}{m-1} + ax^{n-m}\binom{n-1}{m-1} + \dots + ax^2\binom{m+1}{m-1} + ax\binom{m}{m-1}. \tag{B.9}$$

If $x \ge 1$, then $r_m \le v$, and Theorem B.1 follows. Otherwise (B.9) implies that

$$1 + a < a(1 + x + x^2 + x^3 + \cdots)^m = a/(1-x)^m.$$

By applying the Lagrange formula to y^d for $y = 1 + a$, we obtain that $y^d - a^d = du^{d-1}$ for some u, $a \leq u \leq y$. Substitute this expression and deduce that

$$1/x < \frac{(a+1)^d}{(a+1)^d - a^d},\ d = 1/m,$$

so that $r_m/v = 1/x < (a+1)/d$, and then again Theorem B.1 follows. □

Theorem B.1 and Proposition 2.2 also imply a similar upper bound on $1/r_m$, which is the $(n+1-m)$-th root radius of the reverse polynomial $x^n p(1/x)$.

Such bounds immediately imply the solution of Task r for $r = 1$ and $1 + \Delta = 2n$. Indeed, compute the natural m such that $|p_{n+1-m}| = \max_{0 \leq i \leq n} |p_i|$. If $m = n+1$, then $t_n^* \geq 1$, $n > r_n$, and $r_{n+1} = 0$ [see (B.1) and (B.8)]. Moreover, for the reverse polynomial, $q(x) = x^n p(1/x)$, we have $t_1^* \leq 1$, and, therefore, $r_1 < 2$ [see (B.7)], which implies that $r_n > 1/2$ for $p(x)$. It follows that $s = n$ is a desired solution to Task r, for $r = 1$ and $1 + \Delta = n$, as well as for $r = \sqrt{n/2}$ and $1 + \Delta = \sqrt{2n}$. Otherwise, $1 \leq m \leq n$. Then we apply Theorem B.1 (for $a = v = 1$) to $p(x)$ and $q(x) = x^n p(1/x)$ and deduce that $\frac{1}{2(n+1-m)} < r_m < 2m$, so that $1/(2n) < r_m < 2n$, and $s = m - 1$ is a solution to Task r where $r = 1$ and $1 + \Delta = 2n$ [take into account (B.1) where $m = 1,\ s = 0$]. The extensions to arbitrary r is by means of scaling the variable x and to arbitrary Δ by means of the iteration (A.3). We arrive at

Proposition B.1 *Task r can be solved at the cost $\mathbf{O_A}(n(1+g)\log n)$ where g is defined by (B.4)-(B.6); the cost bound can be decreased to $\mathbf{O_A}(n)$ if $1 + \Delta \geq 2n$.*

We could have solved Task s by recursively applying Proposition B.1 in a binary search algorithm, but we will prefer a more direct algorithm outlined in [Sc82].

Algorithm B.1 *Given the coefficients p_u of $p(x)$ and an integer s, $1 \leq s \leq n$, choose two integers t and h that satisfy the following inequalities:*

$$t < n + 1 - s \leq t + h \leq n \tag{B.10}$$

(so that $h > 0,\ t < n$), and the following convexity property: *there exists no integer u in the considered range such that the point $(u, w(u))$ on the plane $\{(u, w)\}$ lies above the straight line passing through the two points $(t, w(t))$ and $(t+h, w(t+h))$, where $w(u)$ denotes* $\log |p_u|$. *Compute and output*

$$r = |p_{t+h}/p_t|^{1/h}. \tag{B.11}$$

The relations (B.10) and (B.11) and the above convexity property immediately imply that

$$\begin{aligned} p_{t+g}/p_t &\leq r^g, \quad for \quad g = 1, \ldots, n-t, \\ p_{t+h-g}/p_{t+h} &\leq 1/r^g, \quad for \quad g = 1, \ldots, t+h. \end{aligned} \tag{B.12}$$

Proposition B.2 *The output r of Algorithm B.1 is a solution to Task s for $1 + \Delta = 2n$.*

Proof. Due to the first and the second inequalities of (B.12), we may apply Theorem B.1 to $p(y)$ with $a = 1$, $v = 1/r$, $i = g$, $m = n + 1 - t - h$ and to $y^n p(1/y)$ with $a = 1$, $v = r$, $i = g$, $m = t + 1$, respectively, and arrive at the desired bounds,

$$r_{n+1-t-h} < 2(n + 1 - t - h)/r, \ 1/r_{n-t} < 2(t + 1)r.$$

Since $h > 0$, $t < n$, it follows that

$$1/(2nr) < r_{n-t} \leq r_s \leq r_{n+1-t-h} < 2n/r. \tag{B.13}$$

□

Let us next specify the computations in Algorithm B.1 as follows:

Algorithm B.2 *Compute the values* $\log|p_u|$, $u = 0, 1, \ldots, n$, *with a prescribed precision; then compute the convex hull* CH *of the set* $\{(u, \log|p_u|),\ u = 0, 1, \ldots, n\}$ *in the two-dimensional real space, and then find the edge in the upper part of the boundary of* CH *whose orthogonal projection onto the* u*-axis is an interval including the point* $n + 1 - s$. *Choose* t *and* $t + h$ *to be the endpoints of this interval and, finally, compute* r *by using (B.11).*

Note that we compute the convex hull CH of the same single set when we solve Task s for all s. Thus we arrive at the following result:

Proposition B.3 *Task* s *for all* s *can be solved at the cost of* $c_{\mathbf{A}}(CH) + \mathbf{O}_{\mathbf{A}}(gn \log n)$ *where* g *is defined by (B.4)-(B.6) and where* $c_{\mathbf{A}}(CH)$ *denotes the cost of computing the values* $\log|p_u|$ *for* $u = 0, 1, \ldots, n$ *and the convex hull of the set* $\{(u, \log|p_u|),\ u = 0, 1, \ldots, n\}$ *of* $n + 1$ *points on the plane.*

Let us next simplify the solution of Task s for $1 + \Delta = 2n$ and for a disc D, with the center 0, under the additional assumption that

$$i(p(x), D) = n + 1 - s, \ i.r.(D) \geq (1 + \Delta)^2. \tag{B.14}$$

In this case, the cost of the solution is $\mathbf{O}_{\mathbf{A}}(n)$ with a small overhead constant.

Proposition B.4 *Let the relations (B.14) hold for a fixed integer* s *(such that* $0 < s \leq n$*), for* $1 + \Delta = 2n$*, and for a disc* D *having the center 0 and an unknown radius. Then Task* s *for* $1 + \Delta = 2n$ *can be solved at the cost* $\mathbf{O}_{\mathbf{A}}(n)$ *if (B.14) holds.*

Proof. Let t and h denote the two integers defined in Algorithm B.1 and thus satisfying (B.10) and (B.13). (B.14) implies that $r_{s-1}/r_s \geq (1 + \Delta)^2 = 4n^2$. On the other hand, the first inequality of (B.10) implies that $r_s \geq r_{n-t}$, so that $r_{s-1}/r_{n-t} \geq (1 + \Delta)^2 = 4n^2$. It follows that either $r_{s-1} > r_{n+1-t-h}$, and then $n + 1 - t - h > s - 1$, $t + h \leq n + 1 - s$, or, otherwise, $r_{n+1-t-h}/r_{n-t} \geq r_{s-1}/r_{n-t} \geq (1 + \Delta)^2 = 4n^2$. The latter inequalities imply that $r_{n+1-t-h} \geq 4n^2 r_{n-t}$, which contradicts (B.13), so that $t + h \leq n + 1 - s$. Therefore, (B.10) implies that $t + h = n + 1 - s$. Since $t + h$ has been defined, it remains to choose $h > 1$ such that the convexity property of Algorithm B.1 holds, or, equivalently, such that the values

$(1/g)\log|\frac{p_{t+h}}{p_{t+h-g}}| = \log(|\frac{p_{t+h}}{p_{t+h-g}}|^{1/g})$ and, therefore, also $|\frac{p_{t+h}}{p_{t+h-g}}|^{1/g}$ reach their maximums where $g = h$, provided that g is the integer parameter ranging from 1 to n. The integer h can be computed at the cost $\mathbf{O_A}(n)$, and we arrive at Proposition B.4. □

We now observe that, for $g(\Delta)$ of (B.4)–(B.6), $g = g(\Delta)$ iteration steps (A.3) suffice to reduce the solution of Task s, for any fixed $1 + \Delta > 1$ and for any disc D satisfying (B.14), to the case where $1 + \Delta = 2n$, and, by shifting the variable X, we may ensure that the disc D has center 0.

Corollary B.2 *Let the relations (B.14) hold for a fixed integer s, $0 < s \leq n$, for a fixed positive Δ, and for a disc D having the center 0 and an unknown radius. Then Task s can be solved at the cost bounded by $\mathbf{O_A}(n + g(\Delta)n\log n)$, where $g(\Delta)$ is defined by (B.4) [and satisfies (B.5) and (B.6)].*

The algorithm supporting Corollary B.2 is referred to as **Algorithm B.3**.

C Computing the Number of Polynomial Zeros in a Disc.

Proposition C.1 *Let $i.r.(D) = f > 1$ for a disc $D = D(X, r)$. Then the index $i(p(x), D)$ of $p(x)$ in D (defined in Definition 2.5) can be computed at the cost $\mathbf{O_A}(hn\log n)$, where*

$$h = 1 + \left\lceil \log\lceil\frac{\log 9}{\log f}\rceil\right\rceil. \tag{C.1}$$

Proof (compare Remark C.1). The well known *winding number algorithms* (see [R], [He], pp. 239–241) compute the index $i(p(x), D)$ of $p(x)$ in a disc D at the cost $\mathbf{O_A}(n\log n)$, provided that all the zeros of $p(x)$ lie far enough from the boundary of such a disc; specifically, it suffices if $i.r.(D) \geq 9$ ([R]). Proposition C.1 now follows due to Remark A.1. □

Remark C.1 *Given a disc $D = D(0, \rho)$ such that $i.r.(D) = f \geq (1+v)^2$, we may compute $s = n + 1 - i(p(x), D)$ by solving Task r of Appendix B for $r = (1+v)\rho$ and for any $\Delta \leq v$. The cost of the solution is $\mathbf{O_A}(n(1+g)\log n)$, where g is defined by (B.4)-(B.6), so that $g = \mathbf{O}(h + \log\log n)$, [compare (B.4) with (C.1)]. This implies an alternative proof of Proposition C.1, provided that the cost bound in its statement changes into $\mathbf{O_A}(gn\log n)$.*

D Analysis of Algorithm 6.1 and 6.1a.

We will next analyze Algorithm 6.1 and will prove its correctness. With no loss of generality, we will assume that $x_0 = 0$ and that we are given an f^*-isolated disc $D = D(0, R)$ containing exactly k zeros of $p(x)$, $z_1, \ldots, z_k$, $1 \leq k \leq n$, R being an upper bound on $r_{n+1-k}(0)$. [Indeed, we may reenumerate the zeros of $p(x)$ and may shift from any disc on the complex plane to the disc D by shifting the variable x.] Thus we have our initial relations:

$$Y = 2R,\ |z_j| \leq R,\ j = 1, \ldots, k, \tag{D.1}$$

$$|z_j| \geq f^* R, \;\; j = k+1, \ldots, n. \tag{D.2}$$

We now designate that

$$Q(x) = \sum_{j=1}^{k} \frac{1}{x - z_j}, \;\; V(x) = \sum_{j=k+1}^{n} \frac{1}{x - z_j} \tag{D.3}$$

and deduce the equation

$$p'(x)/p(x) = Q(x) + V(x). \tag{D.4}$$

Since we may choose the value t arbitrarily close to Y, we will simplify our analysis by assuming that $t = Y = 2R$.

Now recall that D_{min} denotes the minimum disc that is equivalent to the disc D and contains the zeros $z_1, z_2, \ldots, z_k$ of $p(x)$, so that

$$D_{min} = D(X_{min}, R_{min}), \;\; R_{min} = (r.r.(D))R, \;\; X_{min} \in D. \tag{D.5}$$

Let us denote

$$q(x) = x - k/Q(x), \;\; q = q(t), \tag{D.6}$$

$$\Delta(x) = (x - q(x))V(x)/k, \;\; \Delta = \Delta(t). \tag{D.7}$$

We will next prove two auxiliary results.

Lemma D.1 *The complex point $q(x)$ of (D.6) lies in the closed disc D_{min} provided that x lies outside D_{min} and that $Q(x) \neq 0$.*

Proof. Set $y_x(z) = 1/(x - z)$ and let $D_x = y_x(D_{min})$ denote the image of the disc D_{min}, that is, $D_x = \{y_x(z) : \; z \in D_{min}\}$. Since $y_x(z)$ is the reciprocal of a linear function in z and since $x \notin D_{min}$, we apply a known result from the analytic function theory ([Ah]) and deduce that D_x is a disc. On the other hand, $y_x(z_j) \in D_x$, for $j = 1, 2, \ldots, k$, since $z_j \in D_{min}$, for $j = 1, 2, \ldots, k$. Therefore, the average value $Q(x)/k = (1/k)\sum_{j=1}^{k} y_x(z_j)$ lies in D_x. The inverse map, $y_x^{-1}(w) = x - 1/w$, transforms D_x into the disc D_{min}, and therefore, $q(x) = y_x^{-1}(Q(x)/k) \in D_{min}$. $\square$

Lemma D.2 *$u - q = (t - q)\Delta/(1 + \Delta)$, where u, q, t and Δ satisfy the equations (6.1), (D.6) and (D.7).*

Proof. Due to (6.1) and (D.4), we have

$$u = u(t) = t - k/(Q(t) + V(t)). \tag{D.8}$$

Deduce from (D.6) that $Q(t)/k = 1/(t - q)$, substitute this expression into (D.8) and obtain that

$$u = t - \frac{1}{1/(t-q) + V(t)/k} = t - \frac{t - q}{1 + (t-q)V(t)/k} =$$

$$t - \frac{t-q}{1+\Delta} = \frac{t\Delta + q}{1+\Delta}.$$

Therefore,

$$u - q = \frac{t\Delta + q}{1+\Delta} - q =$$

$$\frac{t\Delta - q\Delta}{1+\Delta} = \frac{(t-q)\Delta}{1+\Delta}. \quad \square$$

Hereafter denote that

$$R_x = \max_{j=1,\dots,k} |x - z_j|, \ D_x = D(x, R_x), \ for\ x = t, \ x = u. \tag{D.9}$$

Next obtain the following estimate:

Lemma D.3 *Let $t \notin D_{min}$, $Q(t) \neq 0$. Then*

$$R_u/R \leq 2R_{min}/R + \frac{(R_t/R)^2 \mu}{k - (R_t/R)\mu}, \tag{D.10}$$

where

$$\mu = R|V(t)|. \tag{D.11}$$

Proof. By definition of D_x in (D.9), $z_j \in D_x$, $j = 1, \dots, k$, and, therefore, $D_{min} \subseteq D_x$, for $x = t$, $x = u$. Consequently, $R_x - 2R_{min} \leq |x - a| \leq R_x$ if $a \in D_{min}$. On the other hand, $q \in D_{min}$, due to Lemma D.1 and since $q = q(t)$, $t \notin D_{min}$, $Q(t) \neq 0$. Therefore, $R_x - 2R_{min} \leq |x - q| \leq R_x$, for $x = t$ and $x = u$. Combine the former of these inequalities, for $x = u$, with the equation of Lemma D.2, then apply the latter of them, for $x = t$, and obtain that $R_u - 2R_{min} \leq |u - q| = |t - q| \ |\Delta/(1+\Delta)| \leq R_t|\Delta/(1+\Delta)|$. To estimate $|\Delta/(1+\Delta)|$ from above, set $x = t$, apply the equations (D.7), recall that $|t - q| \leq R_t$, and deduce that

$$|\Delta| = |t - q| \ |V(t)|/k \leq R_t|V(t)|/k,$$

$$1/|1 + \Delta| \leq 1/(1 - |\Delta|) \leq 1/(1 - R_t|V(t)|/k).$$

Combine these bounds on $|\Delta|$ and $1/(1+|\Delta|)$ and obtain that $|\frac{\Delta}{1+\Delta}| \leq \frac{R_t|V(t)|}{k - R_t|V(t)|}$. Substitute the latter inequality into the upper bound on $R_u - 2R_{min}$ above and obtain that

$$R_u - 2R_{min} \leq \frac{R_t^2|V(t)|}{k - R_t|V(t)|},$$

which immediately implies (D.10). $\square$

The next lemma extends Lemma D.3.

Lemma D.4 *For a fixed positive e, let*

$$\theta = \frac{1}{(1-2/e)((f^*-2)k/(n-k)-3)} \leq 1/3, \tag{D.12}$$

$$R_u \geq eR_{min}. \tag{D.13}$$

Then

$$R_u/R \leq \theta(R_t/R)^2 \leq R_t/R\ . \tag{D.14}$$

Proof. From (D.10) and (D.13), obtain that

$$(1-2/e)R_u/R \leq \frac{(R_t/R)^2\mu}{k - R_t\mu/R} = \frac{(R_t/R)^2}{(k/\mu) - R_t/R}. \tag{D.15}$$

Next obtain from the equations (D.3) and (D.11) that

$$\mu \leq R \sum_{j=k+1}^{n} \frac{1}{|t-z_j|} \leq (n-k)R/\min_{j>k}|t-z_j|.$$

The relations (D.2) and $t = 2R$ imply that $|t-z_j| \geq (f^*-2)R$, for $j > k$. Therefore,

$$\mu \leq \frac{n-k}{f^*-2}.$$

Substitute this bound into (D.15) and obtain that

$$(1-2/e)R_u/R \leq \frac{(R_t/R)^2}{(f^*-2)k/(n-k)-(R_t/R)}.$$

We have $R_t \leq 3R$, since $t = 2R$, and will choose f^* such that $(f^*-2)k/(n-k) > 3$. Hence we obtain that

$$(1-2/e)R_u/R \leq \frac{(R_t/R)^2}{(f^*-2)k/(n-k)-3}.$$

Substitute the expression for θ from (D.12) and obtain that $R_u/R \leq \theta(R_t/R)^2$, and, therefore, $R_u/R \leq R_t/R$, since $\theta \leq 1/3$ and $R_t \leq 3R$. This completes the proof of (D.14). □

Correctness of Algorithm 6.1 follows, and, moreover, $\mathbf{O}(\log\log(R/\varepsilon))$ its recursive steps suffice, since, for $f^* > 2 + 6n(e-1)/(e-2)$ and for $e > 2$, the inequality of (D.12) holds, and we may recursively apply Lemma D.4.

Finally, we will analyze the convergence of the computed values u to the disc D_{min} provided that we shift from Algorithm 6.1 to Algorithm 6.1a, by replacing the equations (6.1) by (6.3). The transition from (6.1) to (6.3) amounts to subtracting $V^*(x) = \sum_{j=k+1}^{n} \frac{1}{x_j^*-x}$ from both sides of the equation (D.4), which gives us that

$$q(x) = p'(x)/p(x) - V^*(x) = Q(x) + V(x) - V^*(x).$$

The analysis of Algorithm 6.1 is extended, with the replacement of $V(x)=\sum_{j=k+1}^{n}\frac{1}{x-z_j}$ by

$$V(x)-V^*(x)=\sum_{j=k+1}^{n}\left(\frac{1}{x-z_j}-\frac{1}{x-z_j^*}\right)=\sum_{j=k+1}^{n}\frac{z_j-z_j^*}{(x-z_j)(x-z_j^*)}.$$

The resulting increase of the estimated convergence rate is quantitatively expressed by the following relations for the main parameter θ of Lemma D.4:

$$\theta=\frac{1}{(1-2/e)((f^*-2)^2k/(n-k)-3)}\leq 1/3.$$

In other words, the statement of this basis lemma should remain unchanged, except that, in the expressions for θ, the quantity f^*-2 should be replaced by $(f^*-2)^2$. This enables us to preserve the quadratic convergence of the algorithm even where the upper bound on the initial isolation ratio f^* for the disc D decreases from $f^*>2+6n(e-1)/(e-2)$, for Algorithm 6.1, to

$$f^*>2+\sqrt{6n(e-1)/(e-2)}, \tag{D.16}$$

for Algorithm 6.1a. This enables us to save about 50% of the preceding iterations of Algorithm 5.1, for large n, at the expense of performing $3(n-k)$ additional operations, for each evaluation of u via (6.3), rather than via (6.1).

E Isolation Ratio of an Auxiliary Disc.

Claim E.1 *The disc $\tilde{D}$ defined at Stage 8 of Algorithm 6.1 is f-isolated for $f>2.38$ provided that N is chosen sufficiently large in the applications of Algorithm 3.1 to Algorithm 6.1.*

Proof. Note that, in the applications of Algorithm A.1, we have $r^*(X)\longrightarrow r(X)=r_n(X)$, for any complex X, as $N\longrightarrow\infty$. Since $r^*(X)$ lies close to $r(X)=r_n(X)$, for any complex X and for a sufficiently large N, we will simplify our estimates and our notation by assuming that $r^*(X)=r(X)$, for all X, throughout the proof. In particular, we will write $r(u)$ instead of $r^*(u)$ and $r(v_h)$ instead of $r_n(v_h)$ and $r^*(v_h)$, $h=0,1,2,3$. We will now use these assumption and definitions in the following relations, basic for our proof:

$$r(v_h)\leq|v_h-u|+r(u)=9r(u),\ h=0,1,2,3, \tag{E.1}$$

$$r(v_h)+r(v_{h+2 \bmod 4})=\tilde{r}_h>15r(u),\ h=0,1 \tag{E.2}$$

(see description of Stage 6 of Algorithm 6.1). Consequently, we have

$$r(v_h)\geq 6r(u),\ h=0,1,2,3. \tag{E.3}$$

Hereafter, for $h=0,1,2,3$, let ν_h denote the intersection point of the two line segments, $[\alpha_h,\beta_h]$ and $[v_h,v_{h+1 \bmod 4}]$ (see Figure 4). We will also designate that

$$x_h=|v_{h+1 \bmod 4}-\nu_h|,$$

$$y_h = |v_h - \nu_h|,$$
$$w_h = |\alpha_h - \nu_h| = |\beta_h - \nu_h|,$$

for $h = 0, 1, 2, 3$ (see Figure 4) and will let ω_h denote the angle between the lines passing through the pairs of points $(v_h, v_{h+1 \bmod 4})$ and $(v_{h+1 \bmod 4}, \alpha_h)$, so that $x_h = r(v_{h+1 \bmod 4}) \cos \omega_h$, $w_h = r(v_{h+1 \bmod 4}) \sin \omega_h$. Our next objective is to prove that

$$m_h \sqrt{2} = \min\{x_h + w_h, y_h + w_h\} > 5.951198575 r(u). \tag{E.4}$$

We first observe that

$$(x_h + w_h)^2 = x_h^2 + w_h^2 + 2x_h w_h = r^2(v_{h+1 \bmod 4})\,(1 + 2 \sin \omega_h \cos \omega_h) =$$
$$r^2(v_{h+1 \bmod 4})\,(1 + \sin(2\omega_h)) \leq 36 r^2(u)\,(1 + \sin(2\omega_h))$$

[compare (E.3)]. We now note that $\sin(2\omega_h)$ takes on its minimum value where $r(v_h) = 6r(u)$, $r(v_{h+1 \bmod 4}) = 9r(u)$. Therefore, $m_h \sqrt{2}$ is bounded from below by $x_h + w_h$ provided that x_h, y_h and w_h satisfy

$$r^2(v_h) = x_h^2 + w_h^2 = 36 r^2(u),$$
$$r^2(v_{h+1 \bmod 4}) = y_h^2 + w_h^2 = 81 r^2(u).$$

Subtracting these equations from each other gives us that

$$y_h^2 - x_h^2 = 45 r^2(u).$$

Recall (6.2) and obtain that

$$y_h + x_h = 8\sqrt{2} r(u), \quad y_h - x_h = 45 r^2(u)/(y_h + x_h) = 45 r(u)/(8\sqrt{2}).$$

>From these two equations we have

$$x_h = (4\sqrt{2} - 45\sqrt{2}/32) r(u) = 83 r(u) \sqrt{2}/32.$$

Consequently,

$$w_h^2 = 36 r^2(u) - x_h^2 = (36 - (83)^2/2^9) r^2(u) = 11543 r^2(u)/2^9,$$
$$(w_h + x_h)/\sqrt{2} = (83 + \sqrt{11543}) r(u)/32 > 5.951198575 r(u),$$

which implies a lower bound of (E.4) on m_h for all h. On the other hand, for the radius ρ of the disc $\tilde{D}$ defined at Stage 8 of Algorithm 6.1, we have

$$\rho \leq (8r(u) - \min_h m_h)\sqrt{2},$$

and therefore,

$$\rho < 2.8975 r(u). \tag{E.5}$$

Now let d denote the minimum distance from the disc $\tilde{D}$ to the points β_0, β_1, β_2 and β_3. Then

$$d \geq 2 \min_h w_h. \tag{E.6}$$

On the other hand, the distance from $\tilde{D}$ to the nearest zero of $p(x)$ lying outside $\tilde{D}$ is at least d (see Figure 4). Therefore,

$$i.r.(\tilde{D}) \geq 1 + d/\rho. \tag{E.7}$$

Finally, we observe that w_h reaches its minimum value, where

$$x_h = y_h = (4\sqrt{2})r(u),$$

$$r(v_h) = r(v_{h+1 \bmod 4}) = 6r(u)$$

(see Figure 4). In this case,

$$w_h^2 = r^2(v_{h+1 \bmod 4}) - x_h^2 = (36 - (4\sqrt{2})^2)r^2(u) = 4r^2(u), \ w_h = 2r(u).$$

By combining the latter lower bound on w_h with (E.5)–(E.7), we obtain that

$$i.r.(\tilde{D}) \geq 1 + 4r(u)/\rho > 2.38,$$

which proves Claim E.1. □

Remark E.1 *The proof of Claim E.1 can be extended to show that the disc $\tilde{D}$ is f-isolated for some constant $f > 1$ already for any $N \geq 32$ (we omit the details of estimating such f).*

F Figures.

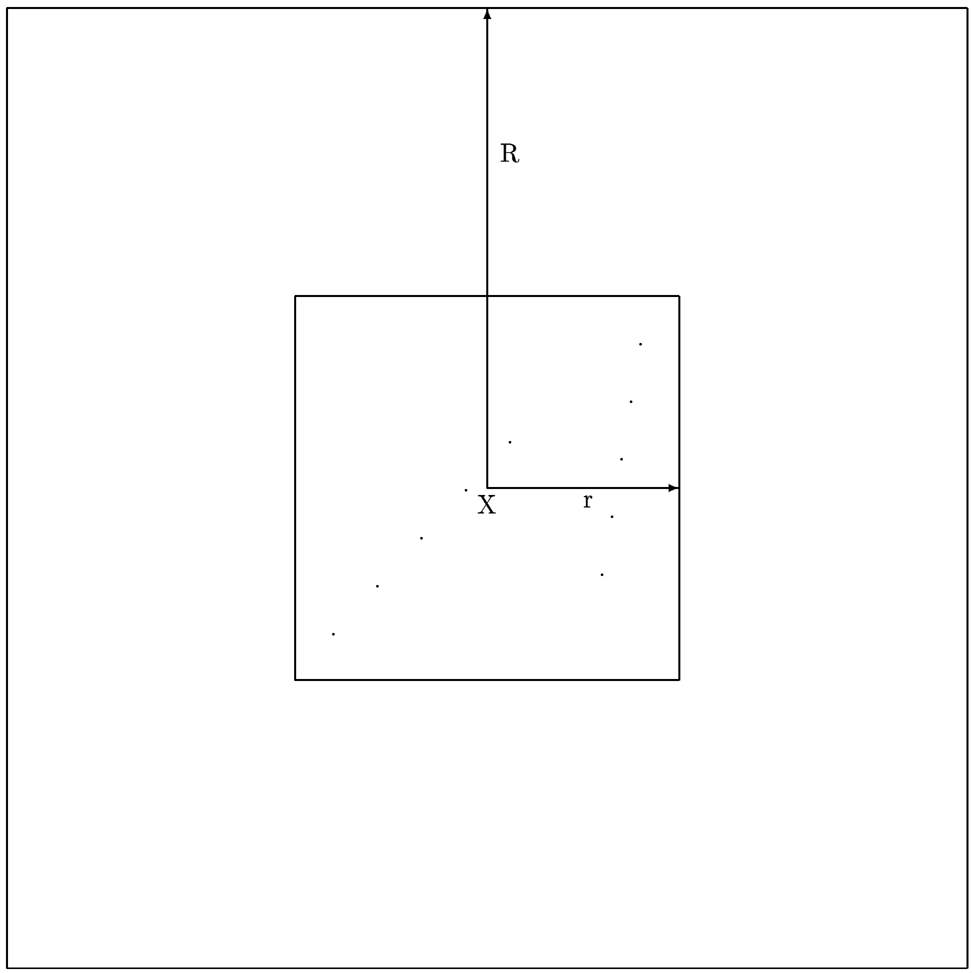

Figure 1: Isolation and rigidity ratios for squares. $i.r.(S(X,r)) \geq R/r$, $r.r.(S(X,R)) \geq r/R$. The dots show all the zeros of $p(x)$ lying in the larger square. They also lie in the smaller square.

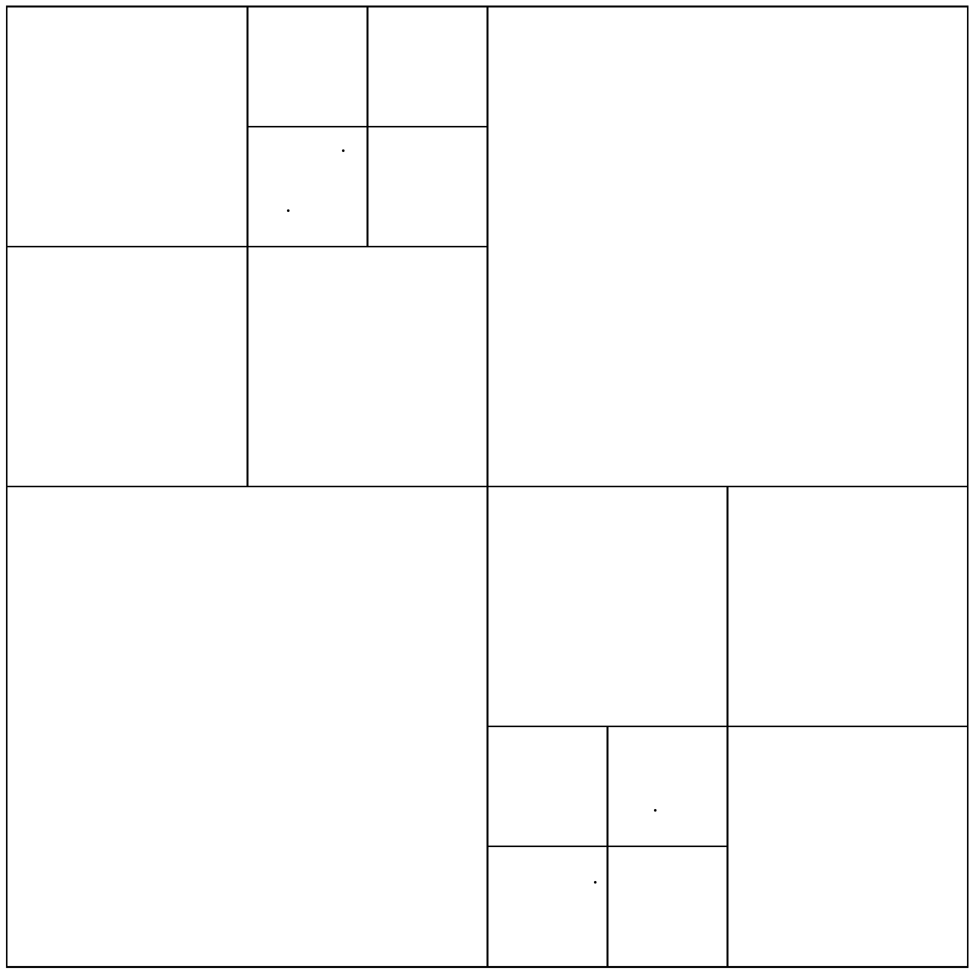

Figure 2: Turan-Weyl's algorithm. The dots show all the zeros of $p(x)$ lying in the large square.

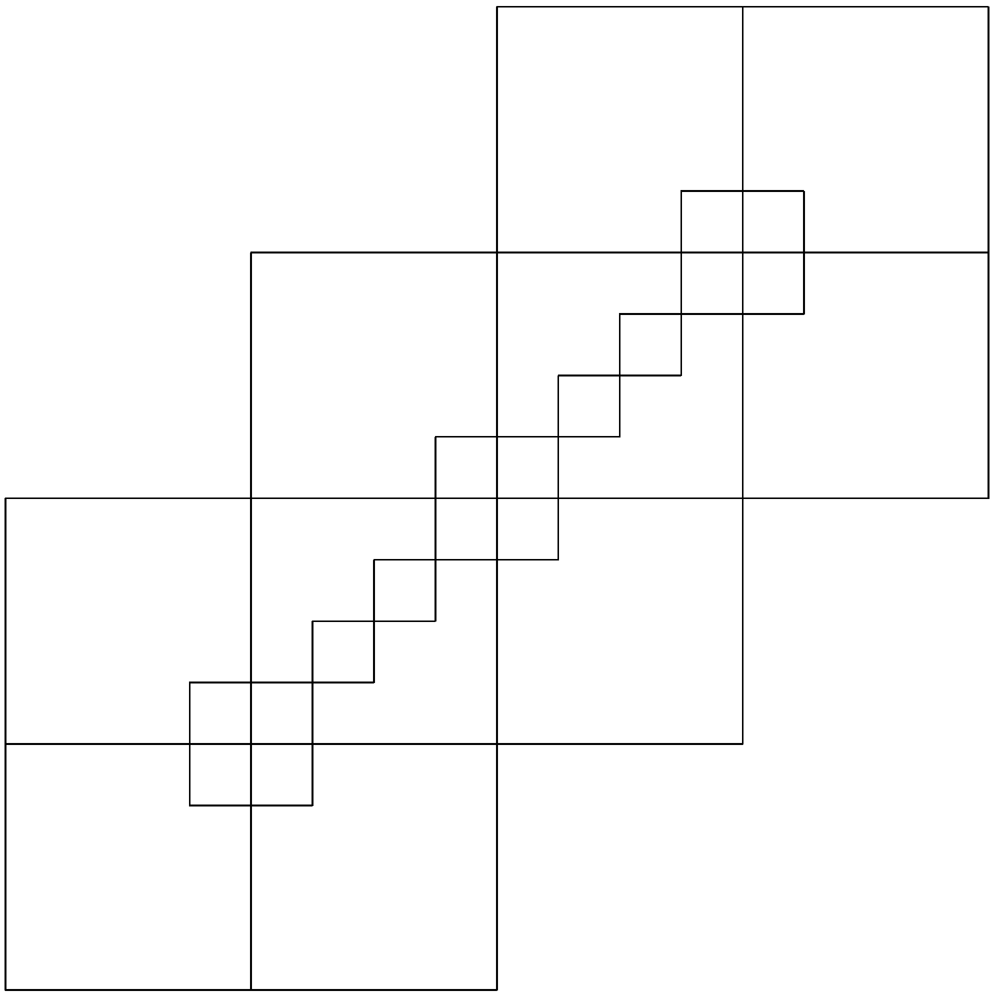

Figure 3: 16 smaller suspect squares at Stage $g+2$ versus 10 larger ones at Stage g of Turan-Weyl's algorithm.

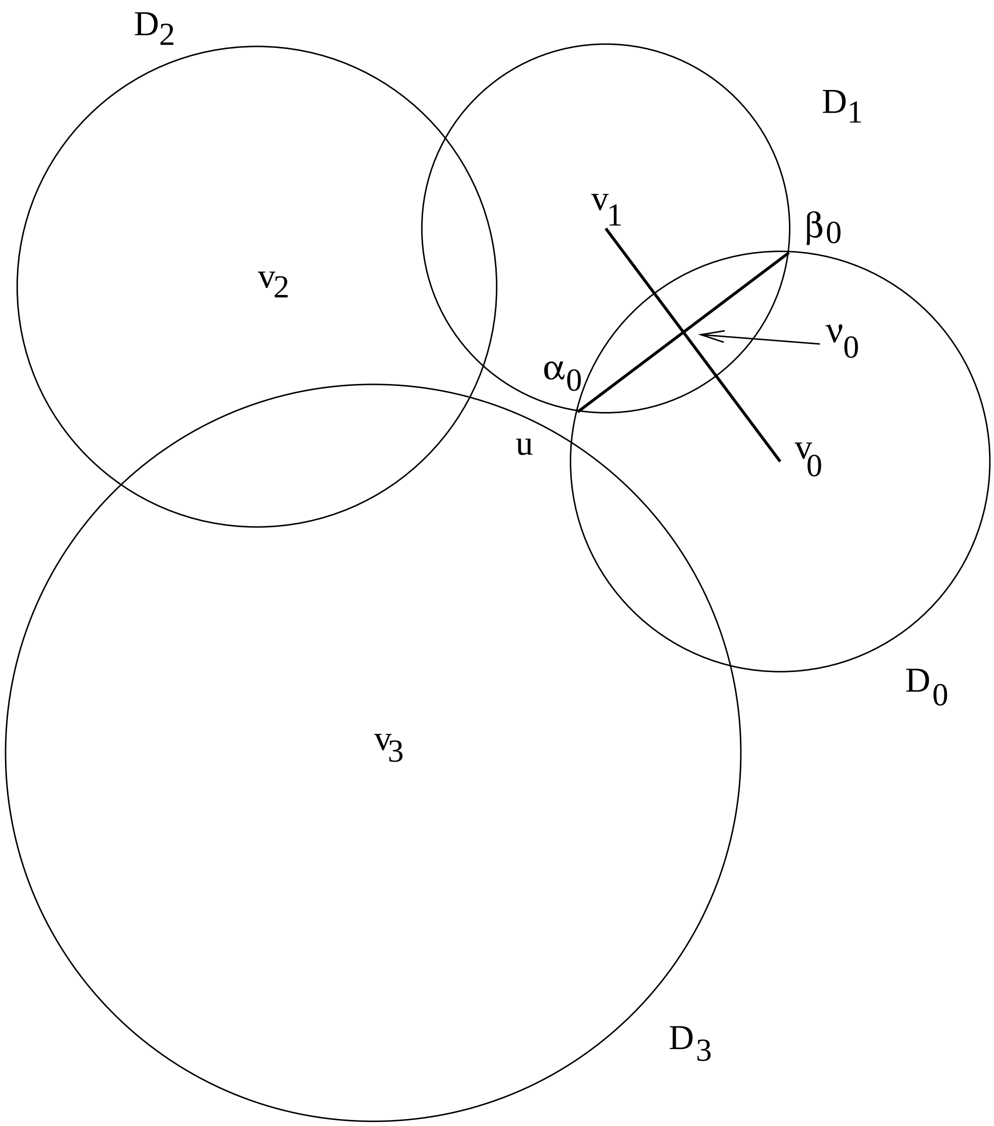

Figure 4: $r(v_h) + r(v_{h+2 \bmod 4}) > 15r(u)$ for $h = 0, 1$; $D_h = D(v_h, r(v_h))$ for $h = 0, 1, 2, 3$; $x_0 = |v_1 - \nu_0|$, $y_0 = |\nu_0 - v_0|$, $w_0 = |\alpha_0 - \nu_0| = |\nu_0 - \beta_0|$.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399