An Efficient Algorithm for the Real Root and Symmetric Tridiagonal Eigenvalue Problems

John H. Reif
Department of Computer Science
Duke University ‡

July 20, 1999

Abstract

Given a univariate complex polynomial $f(x)$ of degree n with rational coefficients expressed as a ratio of two integers $< 2^m$, the *root problem* is to find all the roots of $f(x)$ up to specified precision $2^{-\mu}$. In this paper we assume the arithmetic model for computation. We give an improved algorithm for finding a well-isolated splitting interval and for fast root proximity verification. Using these results, we give an algorithm for the *real root problem*: where all the roots of the polynomial are real. Our real root algorithm has time cost of $O(n \log^2 n(\log n + \log b))$, where $b = m + \mu$. Our arithmetic time cost is thus $O(n \log^3 n)$ even in the case of high precision $b \leq n^{O(1)}$. This is within a small polylog factor of optimality, thus (perhaps surprisingly) upper bounding the arithmetic complexity of the real root problem to nearly the same as basic arithmetic operations on polynomials.

The *symmetric tridiagonal* problem is: given an $n \times n$ symmetric tridiagonal matrix, with $3n$ nonzero rational entries each expressed as a ratio of two integers $< 2^m$, to find all the eigenvalues up to specified precision $2^{-\mu}$. Using known efficient reductions from the symmetric tridiagonal eigenvalue problem to the real root problem, we also get an $O(n \log^2 n(\log n + \log b))$ arithmetic time bound for the symmetric tridiagonal eigenvalue problem.

1 Introduction

1.1 Definition of the Root Finding Problem

Let the *log-precision* of a rational number x be m if x can be expressed as a ratio of two integers $< 2^m$. Let $f(x)$ be an univariate polynomial $f(x) = \sum_{i=0}^{n} c_i x^i$ of degree n with each coefficient c_i given within log-precision m. The *root finding problem* is: given b, approximate all the roots of $f(x)$ within given log-precision μ, where $b = m + \mu$. The *real root finding problem* is the root finding problem given a polynomial with all real roots.

‡surface mail: Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129; E-mail: reif@cs.duke.edu. A preliminary version of this paper appeared in Proceedings of the 34rd IEEE Symposium on Foundations of Computer Science, Palo Alto, CA (1993). Supported by NSF/DARPA CCR-9725021, CCR-96-33567, NSF IRI-9619647, ARO contract DAAH-04-96-1-0448, NSF-IRI-91-00681, Rome Labs Contracts F30602-94-C-0037, ARPA/SISTO contracts N00014-91-J-1985, and N00014-92-C-0182 under subcontract KI-92-01-0182.

1.2 Our Computational Model

For our model of computation, we assume the *algebraic random access machine (RAM)* where each arithmetic or logical operation such as addition, subtraction, multiplication, division, and comparison can be done in one step. The *time complexity* bound of an algorithm is defined to be the number of these steps of the algebraic RAM. We define *space complexity* as the number of memory locations used, where we can store a single rational number in each memory location.

1.3 Application of the Real Root Problem: the Symmetric Eigenvalue Problem

The real root problem has many applications, and one of the most important of these is the *symmetric eigenvalue problem*: given a symmetric matrix, find all the eigenvalues; which are all real in this case. The real eigenvalues are used for many engineering and scientific applications, including vibration analysis in structures, stability analysis, etc. The eigenvalues are the roots of the characteristic polynomial of the matrix. Many large matrices occurring in practice have a special structure which allow the characteristic polynomial as well as associated linear systems (which have the given matrix) to be computed quickly.

1. *Dense structured matrices* (see [BA 80, BGY 80, PR 87]). We define (this is a slight simplification of the usual terminology) an $n \times n$ matrix to have *displacement rank* r if it can be written as the sum of r terms, where each term is the product of a lower triangular Toeplitz matrix and an upper triangular Toeplitz matrix. A matrix A is *Toeplitz* if $A_{i,j} = A_{i+k,j+k}$ for each k where the matrix elements are defined, and Toeplitz have constant bounded displacement rank. For such bounded displacement rank matrices, there are known $O(n \log^2 n)$ algorithms ([BA 80, BGY 80, PR 87]) for solution of their linear systems, and $O(n^2 \log n \log\log n)$ algorithms (see Pan [Pa 90]) for computing all their eigenvalues.

2. *Sparse $n \times n$ matrices* with $O(n)$ non-zero coefficients. For these (e.g., see Canny, Kaltofen and Laksman [CKL 89]) there are known $O(n^2 \log^2 n)$ work algorithms for solution of their linear systems and for approximating all the eigenvalues algorithms.

3. *Tridiagonal and banded matrices*, for there are well known linear time algorithms for solution of their linear systems and $O(n^2 \log^2 n)$ algorithms for approximating all the eigenvalues.

For example, the *symmetric tridiagonal matrix eigenvalue problem* is the problem of finding all the eigenvalues of an $n \times n$ symmetric tridiagonal matrix

$$A = \begin{bmatrix} b_1 & a_2 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & a_3 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & a_4 & \dots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & & & & \\ & & & \ddots & \ddots & & & \\ & & & & & & & \\ 0 & & & & & a_{n-1} & b_{n-1} & a_n \\ 0 & \dots & & & & 0 & a_n & b_n \end{bmatrix}.$$

The real roots problem has an efficient reduction to and from the symmetric tridiagonal matrix eigenvalue problem, which has been attributed to Hald [H 76] and described in [KM 86, BP 91, BP 92, BG 92] and also by JáJá [J 92], p 428, homework 8.37. (This relationship is well-known among numerical analysts and they are encountered in different computational problems as inverse eigenvalue problems, orthogonal polynomials, Sturm sequences, three-term recurrences, Euclidean scheme and Lanczos algorithm.) This reduction from the symmetric tridiagonal matrix eigenvalue problem for the above matrix A to the real roots problem requires us to compute the characteristic polynomial det $(\lambda I - A)$. We sketch here this efficient reduction, with arithmetic cost $O(n \log^2 n)$. For each $i = 1, \ldots, n$ let $p_i(\lambda) =$ det $(\lambda I - A^{(i)})$, where $A^{(i)}$ is the $i \times i$ submatrix consisting of the first i rows and the first i columns. Note that $p_0(\lambda) = 1$, $p_1(\lambda) = \lambda - b_1$, and $p_i(\lambda) = (\lambda - b_i)p_{i-1}(\lambda) - a_i^2 p_{i-2}(\lambda)$. This recurrence equation (see JáJá [J 92], Heller [H 78]) can be solved for $p_n(\lambda)$ =det $(\lambda I - A)$ within arithmetic work $O(n \log^2 n)$, or in parallel time $O(\log^2 n)$ using $O(n \log n)$ processors.

(The reverse reduction of the polynomial root-finding problem for a polynomial $f(x)$ to the symmetric tridiagonal matrix eigenvalue problem, performed by means of the Euclidean remainder scheme, can be found in Hald [H 76]. This reverse reduction is used also in [BP 91, BP 92, BG 92], and shown to have Boolean cost $O(M(n)M(nm) \log n)$. The arithmetic cost for this reduction can easily be seen to be $O(n \log^2 n)$. The Euclidean scheme can be applied to $f(x)$ and $f'(x)$ or equivalently, to $f(x)$ and $g(x)$ where $f'(x_i)g(x_i) > 0$, $f(x_i) = 0$. The computation of this reduction can be performed by means of the "quotient-tree" procedure of [BT 90] (section 8.1), in fact the 2×2 matrices s_i yield all the quotients and the leading coefficients of the remainders, i.e., the entries of the tridiagonal. Recovering the coefficients of the polynomial, given the entries of the matrix, is described in [BP 91, BP 92, BG 92] (comments after algorithm 3.1.) and this problem can be equivalently solved by means of the technique of [KM 86]).

1.4 Previous Algorithms for the Real Root Problem in the Arithmetic Model

Pan [P 89] investigated the complexity of the real root problem in the arithmetic model, achieving processor bounds n^2 with $O(\log^2 n(\log^2 n + \log b))$ parallel time, but no improvement on his previous sequential bounds for the general root problem. Ben-Or and Tiwari [BT 90] investigated the bit complexity of the real root problem. Bini and Pan [BP 91, BP 92] developed an efficient algorithm (which they view as a matrix reformulation of the Ben-Or and Tiwari [BT 90] algorithm) for the symmetric tridiagonal matrix eigenvalue problem, which has arithemetic time cost $O(n \log^2 n \log^2 b)$. Using the known efficient reduction from the real root problem to the symmetric tridiagonal matrix eigenvalue problem described in Subsection 1.3, their resulting real root algorithm has the same arithemetic time cost of $O(n \log^2 n \log^2 b)$, thus has time bound $O(n \log^4 n)$ in the case of precision $b \leq n^{O(1)}$. In contrast, our real root algorithm has arithmetic time cost of $O(n \log^2 n(\log n + \log b))$ and thus $O(n \log^3 n)$ in the case of precision $b \leq n^{O(1)}$.

1.5 Definition of Root Splitting and Isolated Intervals

The following technical definitions will be of use in the discussion of root finding techniques below. For any fixed $\epsilon, 0 < \epsilon < 1/2$, an interval $[s, s']$ on the real line is an *ϵ-splitting interval* for the roots of $f(x)$ if the interval $[s, s']$ contains exactly i roots of $f(x)$, for some i, where $\epsilon n \leq i \leq (1-\epsilon)n$ (see Figure 1). The ϵ-splitting is *balanced* if ϵ is a constant independent of n. For any real $\alpha, \sigma, \delta > 0$, an interval $I = [\alpha - \delta, \alpha + \delta]$ is *σ-isolated* with respect to the roots of $f(x)$ if the larger interval $[\alpha - (1 + \sigma)\delta, \alpha + (1 + \sigma)\delta]$ has no further roots of $f(x)$ other than those contained already in I (see Figure 2). An interval is *well isolated* if it is *σ-isolated*, for $\sigma \geq \Omega(1/n)$ and is *highly isolated* if it is *σ-isolated*, for $\sigma \geq \Omega(1)$.

1.6 Known Techniques used to Solve Root Problems

The techniques used in root finding algorithms (also see the recent text of Mignotte [M 92] for an excellent survey) are:

1. Sturm defined in 1829 a sequence of polynomials known as the Sturm sequence, which can be used for counting the number of roots on interval of the real line. In 1988, Ben-Or, Feig, Kozen and Tiwari [BFK 86] showed that given a polynomial of all n real roots, a Sturm sequence can be used to find a point on the real line that gives a balanced splitting of the roots into sets whose size is a constant fraction of n (see Appendix 9.1) In particular, they found a 1/4-splitting interval. This gave them a divide and conquer NC algorithm for the real root problem. Ben-Or and Tiwari [BT 90] gave an efficient implementation of this parallel algorithm for the real root problem in the bit complexity model. Subsequently, Neff [N 90] generalized this technique to find all the roots in NC in the case where the polynomial has complex roots.

2. Geometric techniques for search and exclusion on the complex plane were developed by Lehmer and Weyl (see [H 70]) and require analytic mappings (such as translation, deflation and inversion) on the variable of a polynomial which can be efficiently computed by reduction to convolution and thus FFT.

3. A method due to Turin in 1968 [T 68, T 75, T 84] (see Appendix 9.3) can be used to determine approximations to the magnitudes of all the roots of a polynomial. Schönhage [S 82] gave an efficient implementation of Turin's method. Turin's method can also be used to determine if a given point is in the proximity of a root. This method requires at least quadratic time to determine which of $2n$ points are in the proximity of n roots.

4. Schönhage [S 82] developed efficient techniques based on the method of Turin for finding a well isolated interval (which, however, may not be a nontrivial splitting interval). Pan [P 89] gave an algorithm, requiring at least quadratic time, for finding a well isolated, balanced splitting interval.

5. An algorithm known as Graeff's Method,(see [H 70] and Appendix 9.3) but which is actually due to Dandelin, 1826, can be used to separate the roots by repeated powering while still maintaining the same degree. Graeff's Method can be used to find a highly isolated splitting interval. After the roots are determined in the powered polynomial then the roots must be extracted for the original polynomial, requiring $2n$ root proximity tests.

6. It was shown in the early 1800s (also see Henrici [H 74]) that the Cauchy contour integration formula gives the *power sums* of the roots within the integration contour. Schönhage [S 82] showed that if the roots are highly isolated and thus sufficiently far from the contour, then the contour integration can be done by reduction to DFT. This gives a relatively low precision approximation to the power sums (see Appendix 9.4). (Also, even if the roots are not sufficiently far from the contour, the contour integration can still be done within appropriate accuracy by careful choice of the discrete points used to approximate the contour integral; see Ben-Or, Feig, Kozen and Tiwari [BFK 86], and Neff [N 90].)

7. The coefficients of a factored polynomial can be efficiently determined from the power sums by use of a triangular linear system known as *Newton's identities* (see Appendix 9.5), which have constant displacement rank [BP 86]. Thus a polynomial can be split into two factor

polynomials by Cauchy contour integration followed by solution of a triangular linear system with constant displacement rank.

8. Newton's iteration method (see Appendix 9.6) for finding a root within high accuracy from an approximation with low accuracy was analyzed in the average case in the works of Smale [S 81, S 85, S 86] and Shube, Smale [SS 85, SS 86], and can be efficiently applied to improve the precision of approximations to all roots by use of multipoint evaluation. Also, Newton's iteration method was used by Schönhage [S 82] to exponentially improve the accuracy of polynomial splitting.

1.7 Organization of this paper

Section 1, the introduction, defines the root finding problem, discusses related history and previous work, looks at application to the symmetric eigenvalue problem, and examines known techniques used to solve root problems.

Section 2 presents some preliminary results for arithmetic on polynomials, translation, deflation, and inversion operations on polynomials, reduction to simplified root finding problems, bounds on roots (root separation), and bounded precision polynomial approximation.

Then we present two improved algorithms for basic subproblems required by our real root algorithm. Section 3 gives an improved algorithm for finding a well-isolated ϵ-splitting interval. Section 4 gives an improved algorithm for fast root proximity verification.

Section 5 presents our real root algorithm.

Section 6 describes applications of the real root problem to the symmetric eigenvalue problem in sparse and structured matrices. Finally, sections 7 and 8 are the conclusion and acknowledgments.

In addition, the Appendix Section 2 provides brief descriptions of some known complexity results for various subproblems encountered in finding real roots; their algorithms are used as subroutines by our real root algorithm. Subsection 9.1 deals with Sturm sequences and root isolation, while Subsection 9.2 deals with approximating the magnitudes of all the roots. Subsection 9.3 discusses obtaining a highly isolated interval by polynomial powering, while Subsection 9.4 discusses FFT approximations to contour integrals. Subsection 9.5 describes polynomial splitting using the Newton identities. Subsection 9.6 deals with high accuracy polynomial splitting using Newton iteration.

2 Preliminaries: Discussion of Known Results

2.1 Arithmetic on Polynomials

Here we describe known results that will be used in our real root finding algorithm.

All the polynomials considered in this paper are assumed univariate unless otherwise indicated. The following are well known results (see [AHU 74])

Lemma 2.1 *There is an $O(n \log n)$ time algorithm for multiplication and division of two polynomials.*

Lemma 2.2 *There is an $O(n \log^2 n)$ time algorithm for computing the n point evaluation and also greatest common divisor(gcd) of two polynomials.*

2.2 Translation, Deflation and Inversion Operations on Polynomials

There are efficient algorithms for analytic mappings such as translation, deflation, inversion, etc. of a polynomial. These will be used in our root finding algorithm. Let $f(x)$ be a polynomial of degree n. In each of these operations we wish to compute, without loss of precision, the coefficients of the new polynomial given the coefficients of the original polynomial.

Lemma 2.3 *There is an $O(n \log n)$ time algorithm for the variable translation mapping $x \mapsto x - \alpha$, for real α, resulting in translated polynomial $g(y) = f(y - \alpha)$.*

Lemma 2.4 *There is an $O(n)$ time algorithm for the variable deflation mapping $x \mapsto x/\alpha$, for real α, resulting in polynomial $g(y) = f(y/\alpha)$.*

Lemma 2.5 *There is an $O(n)$ time algorithm for the variable inversion mapping $x \mapsto 1/x$, for real α, resulting in translated polynomial $g(y) = y^n f(1/y)$.*

2.3 Reduction to Simplified Root Finding Problems

Here we review classical techniques for reduction to simplified root finding problems. A polynomial $f(x)$ is *monic* if its highest order coefficient is 1, so it can be written $\prod_{i=1}^{n}(x - r_i)$ where the r_i are the roots. Obviously, any polynomial can be made monic by dividing out the highest order coefficient without altering the roots. Also, any polynomial with rational coefficients can obviously be made into a polynomial with integer coefficients by simply multiplying the denominators of the rational coefficients.

Given a univariate polynomial $f(x)$ of degree n with complex coefficients, we can construct a polynomial $f(x)f^*(x)$ (where the coefficients of $f^*(x)$ are the complex conjugates of the coefficients of $f(x)$) of degree $2n$ with real coefficients containing all the roots of $f(x)$, thus efficiently reducing the problem of root finding in complex polynomials to the problem of root finding in polynomials with real coefficients.

A *simple root* r of a polynomial $f(x)$ is a root with no multiplicity. Given a univariate polynomial $f(x)$ of degree n with n' distinct zeros, we can construct a polynomial $\frac{f(x)}{gcd(f(x), f'(x))}$ with exactly the same distinct n' roots but no multiplicities, thus efficiently reducing the problem of root finding in polynomials with multiple roots to the problem of root finding in polynomials with only simple roots.

Thus, throughout this paper we generally assume the given polynomial $f(x)$ is monic and has only integer coefficients and simple roots, say $r_1 < r_2 < \ldots < r_n$.

2.4 Bounds on Roots: Root Separation

The *root separation* of a polynomial is the minimum distance between any two distinct roots. Mahler ([M 64]) proves:

Lemma 2.6 *The minimum root separation of a polynomial of degree n and coefficient log-precision m is lower bounded by $\sqrt{3}n^{-(n+2)/2}2^{-m(n-1)} \leq 2^{-(2n \log n + mn)}$.*

Cauchy in 1929 (see Barbeau [B 80],p 180, and Householder, 1970 [H 70]) showed that the maximum magnitude of any root of a monic polynomial is upper bounded by the sum of the magnitudes of it's coefficients. The resulting root bounds can be tightened if the polynomial has integer coefficients (Householder,1970 [H 70], [BFK 86]).

Lemma 2.7 *Given a polynomial of degree n with coefficient log-precision m, the maximum magnitude of any root is $\leq n2^m$. Also, if the polynomial has integer coefficients, then the maximum magnitude of any root is $\leq 2^m + 1$.*

2.5 Bounded Precision Polynomial Approximation

The following can be proven using the fact that a polynomial with real coefficients has roots which are either real or complex conjugate pairs.

Lemma 2.8 *([BFK 86]) Given a polynomial $f(x)$ of degree n with real coefficients with log-precision m, let $\tilde{f}(x)$ be an approximation of $f(x)$ up to log-precision π, that is, the coefficients of $\tilde{f}(x)$ differ from the coefficients of $f(x)$ by at most $2^{-\pi}$, where $\pi > n(n+m+\mu+2\log n+2)$. The roots of $\tilde{f}(x)$ differ from the roots of $f(x)$ by at most $2^{-\mu}$.*

Lemma 2.9 *(Householder,1970 [H 70], [BFK 86]) If the minimum root separation of $f(x)$ is $> 2^{-\mu}$ and $f(x)$ has all distinct real roots, then so does $\tilde{f}(x)$.*

The following Lemmas are due to Schönhage [S 82], and Ben-Or, Feig, Kozen and Tiwari, 1986 [BFK 86].

Lemma 2.10 *Given a monic polynomial $f(x)$ of degree n with coefficient log-precision m, and suppose $f_1(x), \ldots, f_n(x)$ are monic polynomials such that the polynomial $f(x) - \prod_{i=1}^{n} f_i(x)$ has degree $n-1$ and coefficient log-precision $\pi = n(m+\mu+2\log n+4)$, then $f_1(x), \ldots, f_n(x)$ need to have at most coefficient log-precision nm.*

Lemma 2.11 *Given a monic polynomial $f(x)$ of degree n with coefficient log-precision m, and suppose $f_1(x), \ldots, f_n(x)$ are the approximate linear factors of $f(x)$ using log-precision $O(n(\log n + m + \mu))$, then the coefficient size of $f(x) - \prod_{i=1}^{n} f_i(x)$ is $< 2^{-\mu}$.*

3 Finding A Well-Isolated ϵ-Splitting Interval

Recall the definitions for σ-isolated and well isolated intervals as well as balanced splitting intervals as given in Subsection 1.5. The following is an improvement of Pan's [P 89] quadratic time algorithm for finding a well isolated balanced splitting interval.

Lemma 3.1 *There is an $O(n\log^2 n)$ algorithm given a $1/4$-splitting point, for finding a σ-isolated $1/6$-splitting interval for the n real roots of $f(x)$, where $\sigma = 2/((1+\theta)n)$, for any constant $\theta > 0$.*

Proof:

Let $r_1 < r_2 < \ldots < r_n$ be the roots of $f(x)$. Let $\gamma = ln(1+\theta)/n$, where ln is the natural logorithm. Compute by Lemma 9.4 a rational s which is a 1/4-splitting point for the roots of $f(x)$. Construct the shifted polynomial $g(y) = f(y-s)$. (Note: the purpose of shifting by s is to insure the root distances are determined from s rather than 0. Also, note that it suffices for us to find a splitting interval for the shifted polynomial $g(y)$, since for any interval containing k roots of $g(y)$ the corresponding interval shifted by s contains k roots of $f(x)$.) Compute by Lemma 9.6 the approximate root magnitudes of $g(y)$, giving for each $i = 1, \ldots n$, a lower bound L_i and upper bound $U_i = L_i(1+\gamma)$ of the distance from s of each root r_i of $f(x)$: $L_i \leq |r_i - s| \leq U_i$, with $L_1 \leq L_2 \leq \ldots \leq L_n$. Let $U_0 = 0$. We will consider the *gaps* between consecutive bounding intervals for the roots of $g(y)$. We define *gap* i to be the interval $[U_i, L_{i+1}]$ if $L_{i+1} > U_i$ and otherwise the *empty* interval of length 0. The length of gap i is $G_i = max(0, L_{i+1} - U_i)$ for $0 \le i \le n-1$ and let the length of gap n be $G_n = \infty$. We say gap k *dominates gaps* $i, \ldots, j$, for $i \le j$, if gap k has the largest length among these gaps $i, \ldots, j$.

Claim: Suppose for $k+1 \le k'$, gap k dominates gaps $0, \ldots, k'-1$, but $k' = n$ or gap k does not dominate the gap k'. If there are exactly R roots of $g(y)$ of magnitude at least L_{k+1} and at most $U_{k'}$, where $\epsilon n \le R$ and $\epsilon < 1/2$, then one of

$$I^+ = [L_{k+1}, U_{k'}], I^- = [-U_{k'}, -L_{k+1}]$$

is a σ-isolated $\epsilon/2$-splitting interval for $g(y)$. If there are exactly R' roots of $g(y)$ of magnitude at least L_{k+1} and at most U_n, where $\epsilon' n \le R' \le (1-\epsilon')n$ and $\epsilon' < 1/2$, then one of

$$I' = [-U_k, U_{k'}], I'' = [-U_{k'}, U_k]$$

is a σ-isolated $\epsilon'/2$-splitting interval for $g(y)$, where $\sigma = 2/((1+\theta)n)$.

Proof:

In accordance with the assumptions of the claim, we have $G_k \ge G_i$ for each $i = 0, \ldots, k'-1$. For each $i = k+1, \ldots, k'-1$, we have the recurrence equation: $U_{i+1} \le (U_i + G_k)(1+\gamma)$, giving the bound

$$U_{k'} \le G_k \sum_{i=1}^{k'} (1+\gamma)^i \le nG_k(1+\gamma)^n \le nG_k(1+\theta)$$

since $(1+\gamma)^{(1/\gamma)} \le e$ and

$$(1+\gamma)^n \le (1+\gamma)^{(1/\gamma)(\gamma n)} \le e^{\gamma n} \le 1+\theta$$

for $\gamma = ln(1+\theta)/n$. Thus we have

$$|I^+| = |I^-| \le U_{k'} - U_k \le U_{k'} \le nG_k(1+\theta) \le G_k/(2\sigma),$$

for $\sigma = 2/((1+\theta)n)$, so the intervals I^+,I^- each have radius

$$\delta = |I^+|/2 = |I^-|/2 \le G_k/(2\sigma).$$

Also we have

$$|I'| = |I''| \le 2U_{k'} \le 2nG_k(1+\theta) \le G_k/\sigma,$$

so the intervals I',I'' each have radius $\delta' = |I'|/2 = |I''|/2 \le G_k/\sigma$, where $\delta' \ge \delta$. Hence $G_k \ge \sigma\delta'$, so the intervals $[U_{k'}, U_{k'} + \sigma\delta'], [-U_{k'} - \sigma\delta', -U_{k'}]$,

$$[L_{k+1} - \sigma\delta', L_{k+1}], [-L_{k+1}, -L_{k+1} + \sigma\delta']$$

contain no roots (see figure 4). Thus each of the intervals I^+, I^-, I', I'' are surrounded by empty intervals of size $\sigma\delta' \ge \sigma\delta$ containing no roots, so it follows that each of these intervals are σ-isolated. Suppose there are exactly R roots of magnitude at least L_{k+1} and at most $U_{k'}$, where $\epsilon n \le R$ for $\epsilon < 1/2$. By the pigeon hole principal, either (i) at least $R/2$ of these R roots are non-negative or (ii) at least $R/2$ of these R roots are non-positive. If $R \le (1-\epsilon)n$, then one of the intervals I^+ or I^- has at least $R/2 \ge (\epsilon/2)n$ roots and at most $R \le (1-\epsilon)n$ roots of $g(y)$. Else if $R > (1-\epsilon)n$, then one of the intervals I^+ or I^- has at least $R/2 \ge (1-\epsilon)n/2$ roots and (since s is a 1/4-splitting point, both $[-\infty, s]$ and $[s, \infty]$ have at most $3n/4$ roots each) at most $3n/4$ roots of $g(y)$. Since $\epsilon/2 \leq 1 - 3/4 = 1/4$ for $\epsilon < 1/2$, we conclude in either case that one of the intervals I^+ or I^- is a σ-isolated $\epsilon/2$-splitting interval of $g(y)$.

Now suppose there are exactly R' roots of magnitude at least L_{k+1} and at most U_n, for $\epsilon' n \leq R' \leq (1-\epsilon')n$ and $\epsilon' < 1/2$. By the pigeon hole principal, either (i) at most $R'/2$ of these R' roots are non-negative or (ii) at most $R'/2$ of these R' roots are non-positive. Then one of the intervals I' or I'' has at least $n - R' \geq \epsilon' n$ roots and at most $R'/2 + (n - R') = n - R'/2 \leq (1 - \epsilon'/2)n$ roots. Since $\epsilon' \geq 1 - (1 - \epsilon'/2) = \epsilon'/2$, we conclude one of the intervals I' or I'' is a σ-isolated $\epsilon'/2$-splitting interval of $g(y)$. This completes the proof of the Claim. ∎

We will use the above claim to complete our proof of Lemma 3.1 as follows. Fix a gap k_1, for $0 \leq k_1 \leq \lfloor 2n/3 \rfloor$, which dominates gaps $0, \ldots, \lfloor 2n/3 \rfloor$. Let k_2 be the minimum number $> \lfloor 2n/3 \rfloor$ such that gap k_1 does not dominate the gap k_2, but dominates the gaps $\lfloor 2n/3 \rfloor + 1, \ldots, k_2 - 1$ (note that if gap k_1 dominates all the gaps $\lfloor 2n/3 \rfloor + 1, \ldots, n-1$, then $k_2 = n$).

CASE 1. Consider the case that $k_1 \leq \lfloor n/3 \rfloor$. Let $I^+ = [L_{k_1+1}, U_{k_2}]$,$I^- = [-U_{k_2}, -L_{k_1+1}]$. Since there are R roots of magnitude at least L_{k_1+1} and at most U_{k_2}, where $n/3 \leq R$, all the requirements in first part of the claim are satisfied for $\epsilon = n/3$, so either I^+ or I^- is σ-isolated 1/6-splitting interval.

CASE 2. Next consider the case that $k_1 > \lfloor n/3 \rfloor$. Let $I' = [-U_{k_2}, U_{k_1}]$,$I'' = [-U_{k_1}, U_{k_2}]$. Since there are R' roots of magnitude at least L_{k_1+1} and at most U_n, where $n/3 \leq R' \leq 2n/3$, all the requirements in the second part of the claim are satisfied for $\epsilon' = n/3$, so either I' or I'' is σ-isolated 1/6-splitting interval.

In either case, we can determine which interval to use by applying Lemma 9.3.

∎

4 Fast Root Proximity Verification

Note that there may be two real N-th roots of a real number, for $N \geq 2$. Thus, to apply Lemma 9.7, we need a way to test, given a real point, whether it is close to a root of $f(x)$ Pan [P 87, P 89] utilized a costly proximity test due to Turin requiring $\Omega(n \log^2 n)$ time per test, and a total time of $\Omega(n^2 \log^2 n)$ time for the $2n$ tests. The following improves on Turin's test:

Theorem 4.1 *Let $f(x)$ be a polynomial of degree n with all real roots and with real coefficients of log-precision m. Let S be a set of real points where we wish to test which points are within a given sufficiently small distance $2^{-\pi}$ of a root of $f(x)$, where $\pi = O(n(\mu + n + m))$. Then all these tests can be done in time $O(n \log^2 n)$.*

Proof:

We reduce the task to multipoint evaluation, for which there are $O(n \log^2 n)$ time algorithms (Lemma 2.2).

Lemma 4.1 *Given a polynomial $f(x)$ of degree n with all real roots and coefficient log-precision m, if we find a real α such that $|f(\alpha)| < 2^{-\tau}$, for sufficiently large τ, then the closest distance Δ_n from α to a root of $f(x)$ is at most $2^{m-\tau}$ for $n = 1$ and at most $n2^{n-(\tau - m + n(n+1)/2)/n}$ for $n > 1$.*

Proof:

For fixed $m, \beta \geq 1$, we show by induction on n that if α is further than $\Delta_n = n2^{n-\beta}$ from a root of $f(x)$ then $|f(\alpha)| > 2^{-\tau_n}$ for $\tau_1 = \beta + m - 1$ and $\tau_n = n\beta + m - n(n+1)/2$ for $n > 1$ (see Figures 6.1,6.2,6.3).

In the base case $n = 1$, $f(x) = c_0 + c_1 x$ is a linear function with $c_1 \neq 0$, so $|f'(x)| = |c_1| \geq 2^{1-m}$. Thus $|f(x)|$ grows by a rate of at least 2^{-m}, so $|f(\alpha)| \geq |c_1|\Delta_1 \geq 2^{-m}2^{1-\beta} \geq 2^{-\tau_1}$, for $\tau_1 = \beta + m - 1$.

Now consider an $f(x)$ of arbitrary degree $n > 1$. Since we have assumed that $f(x)$ has real coefficients and all roots are real, by Rolle's result (Lemma 9.2), the roots of $f'(x)$ are all real and they strictly interleave the roots of $f(x)$. Choose r to be the root of $f(x)$ either just below α or just above α, such that $f'(x)$ is not zero between α and r. Let I be the interval between α and r, not containing r but containing α. Thus, there are at most two roots, say r_1, r_2 of $f'(x)$ where r_1, r_2 are not on I but are of distance $\leq \Delta_{n-1}$ from a point on I. Let I' be the points of I of distance $\geq \Delta_{n-1}$ from a root of $f'(x)$. Note that

$$|I'| \geq \Delta_n - 2\Delta_{n-1} > n2^{n-\beta} - (n-1)2 \cdot 2^{(n-1)-\beta} \geq 2^{n-\beta}(n - (n-1)) \geq 2^{n-\beta}.$$

Since $f'(x)$ has degree $n-1$, by the induction hypothesis, $|f'(x)| \geq 2^{-\tau_{n-1}}$ for all $x \in I$ of distance $> \Delta_{n-1} = (n-1)2^{n-1-\beta}$ from r_1 or r_2. Thus

$$|f(\alpha)| \geq 2^{-\tau_{n-1}}|I'| \geq (2^{-\tau_{n-1}})(2^{n-\beta}) \geq 2^{n-\beta-\tau_{n-1}} \geq 2^{-\tau_n},$$

for τ_n, requiring τ_n to satisfy the recurrence equation $\tau_n = \tau_{n-1} + \beta - n$ for $n > 1$, and $\tau_1 = \beta + m - 1$. This gives

$$\tau_n = \tau_1 + \sum_{i=2}^{n}(\beta - i) = (\beta + m - 1) + ((n-1)\beta + 1 - n(n+1)/2)$$

$= n\beta + m - n(n+1)/2$ for $n > 1$. ∎

5 Our Real Root Algorithm

In the following, we assume the input polynomial $f(x)$ is monic, has degree n, with each coefficient given within log-precision m, and we wish to approximate all the roots within given log-precision μ. Our main result is

Theorem 5.1 *There is an algorithm for the real root problem which has arithmetic time cost $O(n \log^2 n(\log n + \log b)$, where $b = m + \mu$, using arithmetic steps.*

Proof:

We will use all the techniques given in Sections 2, 3, and 4 as subroutines for our real root finding algorithm. Because of the large number of details of our Real Root Algorithm, we will present it in three stages of increasing detail and complexity.

A Naive Version of our Real Root Algorithm is summarized below:

INPUT A monic polynomial $f(x)$ with degree n and log-precision m.

OUTPUT A set S of n rationals approximating the roots of $f(x)$ within log-precision μ.

Goal: Split polynomial $f(x) = f_1(x)f_2(x)$ into factors $f_1(x), f_2(x)$ where $deg(f_1(x)) \geq \epsilon n$, $deg(f_2(x)) \leq (1-\epsilon)n$, for constant $\epsilon, 0 < \epsilon < 1$.

1. Get a high accuracy approximation to factors $f_1(x), f_2(x)$ with error $\leq 2^{-\pi}$.

2. Recursively factor $f_1(x)f_2(x)$, with error $\leq 2^{-\pi}$.

In fact we do not actually split $f(x)$, and instead recursively split a related polynomial $\hat{f}(y)$, and then recover the roots of $f(x)$ from the roots of $\hat{f}(y)$. We give the details of our real root algorithm below.

Algorithm Real Roots:

INPUT A monic polynomial $f(x)$ with degree n and log-precision m.

OUTPUT A set S of n rationals approximating the roots of $f(x)$ within log-precision μ.

1. Eliminate any multiple zeros.

2. Construct from $f(x)$ a poly $\hat{f}(y)$ of degree n with highly isolated balanced splitting interval $\hat{I}$.

3. Get high accuracy approximation to factors $\hat{f}_1(y), \hat{f}_2(y)$ of $\hat{f}(y)= \hat{f}_1(y)\hat{f}_2(y)$ with $deg(\hat{f}_1(y))$,$deg(\hat{f}_2(y))$ both bounded by $n5/6$.

4. Recursively approximate factor $\hat{f}_1(y), \hat{f}_2(y)$, giving set S' of n high accuracy approximation to the roots of $\hat{f}(y)$.

5. From S', using our fast root proximity test described in Section 4, construct a set S of n high accuracy approximates to the roots of $f(x)$.

We now outline the operations of Step [2]: Finding a highly isolated balanced splitting interval $\hat{I}$.

[2.1] Approximate magnitudes of all roots of $f(x)$ by Turin's method, as described in 9.3.

[2.2] Compute the Sturm sequence and sign sequence, as described in Appendix 9.1, so we can count roots on real intervals fast.

[2.3] Find a 1/4-splitting point, as is also described in Appendix 9.1.

[2.4] Find a well isolated balanced splitting interval I, again using our improved algorithm described in Section 4.

[2.5] Use Graeff's method, as described in Appendix 9.3 to construct a poly $\hat{f}(y)$ degree n with highly isolated balanced splitting interval $\hat{I}$.

[2.6] Compute by the Sturm sequence method, as described in Appendix 9.1, the number n' of roots of $\hat{f}(y)$ within $\hat{I}$.

Note: Step [2.4] uses our improved $O(n \log^2 n)$ time algorithm, and the final step [5] requires our fast root proximity test (This improves on previous quadratic time algorithms of [Pan,89]).

We next outline the operations of Step [3]: Get high accuracy approximation to factors $\hat{f}_1(y), \hat{f}_2(y)$ of $\hat{f}(y)$.

[3.1] Do Cauchy contour integration via FFT, using highly isolated balanced splitting interval $\hat{I}$, as described in Appendix Subsection 9.4.

[3.2] Solve Newtons equation, as described in Appendix Subsection 9.5, which is a triangular linear system with constant displacement rank [BP 86]. This gives an approximation to the coefficients of the factors of $\hat{f}(y)$ within error $\leq 2n/(1+\sigma)^n$.

[3.3] Using Newton Iterations, as described in Appendix Subsection 9.6, get high accuracy approximations to factors of $\hat{f}(y)$ within error $\leq 2^{-\pi}$.

We give further details of our real root algorithm below. (Note that we have renumbered the steps of algorithm, so the steps are in sequence.) The corresponding sections give detailed description and proof of each step.

INPUT A monic polynomial $f(x)$ with degree n and log-precision m.

OUTPUT A set S of n rationals approximating the roots of $f(x)$ within log-precision μ.

1. Eliminate any multiple zeros, by computing

$$\frac{f(x)}{gcd(f(x), f'(x))}$$

in time $O(n \log^2 n)$ by applying Lemmas 2.1 and 2.2 and replacing $f(x)$ with this new polynomial, which has the same distinct roots as $f(x)$, but no multiple roots. Reassign n to be the degree of the new $f(x)$. (See Subsection 2.3)

2. In time $O(n \log^2 n)$, find s a 1/4-splitting point for the roots of $f(x)$ (recall s is a real point between the ith and $(i+1)$th roots for some $n/4 \leq i \leq 3n/4$) guaranteed by Lemma 9.4. Here we first compute the Sturm sequence in time $O(n \log^2 n)$ by Lemma 9.1. Then we determine the sign sequence of the Sturm sequence, which is computable by multipoint evaluation in time $O(n \log^2 n)$ (Lemma 2.2). Then we apply Lemma 9.5 using $O(\log n)$ stages of binary search to determine the 1/4-splitting point from the linear coefficients of the Sturm sequence of $f(x)$. At each stage we use the precomputed sign sequence of the Sturm sequence (described in Lemma 9.2) to count the number of roots within an interval. Thus each stage takes $O(n)$ time. The total time for all $O(\log n)$ stages and including precomputation is $O(n \log^2 n)$.

3. Approximate the magnitudes of all the roots by Turin's method in time $O(n \log^2 n)$ by applying Lemma 9.6. This determines the magnitude of each root r_i of $f(x)$ to be within an interval

$$[r_i/(1 - \frac{1}{n^{O(1)}}), r_i(1 + \frac{1}{n^{O(1)}})].$$

4. Find a well isolated 1/6-splitting interval I, using the previously computed 1/4-splitting point and the approximate magnitudes of all the roots. Here we use our improved $O(n \log^2 n)$ time algorithm given by Lemma 3.1 (improving on the previous more than quadratic time algorithm of Pan [P 89]).

5. Use Graeff's method to construct in time $O(n \log^2 n)$ a polynomial $\hat{f}(y)$ of degree n with a highly isolated 1/6-splitting interval $\hat{I}$. $\hat{f}(y)$ is constructed by $\rho = O(\log n)$ stages of polynomial multiplication described in Lemma 9.7 requiring $O(n \log n)$ time per multiplication stage by Lemma 2.1. The variable transformation $\Phi(x) = y$ involves various scalar calculations including reciprocals and taking 2^ρ-th powers of scalars.

6. Compute by the Sturm sequence method the number n' of roots of $\hat{f}(y)$ within $\hat{I}$.

7. Let Γ be the circle of diameter $length(\hat{I})$ on the complex plane intersecting the real line at the end points of the interval $\hat{I}$. For each $k = 1, \ldots, n'$, approximately evaluate the (complex) Cauchy contour integral:

$$s_k = \frac{1}{2\Pi\sqrt{-1}} \int_{z \in \Gamma} z^k \frac{\hat{f}(y)'}{\hat{f}(y)} dy$$

giving an approximation to the k-th power sum s_k of the roots of $\hat{f}(y)$. To approximately evaluate the integral, shift and deflate the polynomial $\hat{f}(y)$ by the variable mapping

$$y \mapsto (y - center(\hat{I}))/radius(\hat{I})$$

so the resulting mapped interval is now $[-1, 1]$. This requires time $O(n \log n)$ by Lemmas 2.3, 2.4 and 2.5. Then evaluate the Cauchy contour integral of the transformed polynomial at the nth roots of unity, using a FFT in $O(n \log n)$ time. By Lemma 9.8 the error is $\leq 2n/(1+\sigma)^n$, for some constant $\sigma > 0$.

8. Using the Newton Identities given in Lemma 9.10 construct a triangular system relating the power sums of the n' roots of $\hat{f}(y)$ within the interval $\hat{I}$ and the coefficients of a monic polynomial with these same n' roots. Solve this triangular system, which has constant displacement rank [BP 86], in time $O(n \log n)$, thus determining approximate factor polynomial $\hat{f}_1(y)$ and $\hat{f}_2(y)= \hat{f}(y)/\hat{f}_1(y)$ where $deg(\hat{f}_1(y)) \geq n/6$, $deg(\hat{f}_2(y)) \leq n5/6$, with coefficient error between $\hat{f}(y)$ and $\hat{f}_1(y)\hat{f}_2(y)$ at most $2n/(1+\sigma)^n$.

9. Using $O(\log \pi)$ stages (where $\pi = O(n(\mu + m + n))$) of Newton's Iteration method (Lemma 9.12), compute a high accuracy polynomial splitting to exponentially improve the accuracy of the previous low accuracy polynomial splitting. This yields approximate factor polynomials $\hat{f}_1(y), \hat{f}_2(y)$ where $deg(\hat{f}_1(y)) \geq n/6$, $deg(\hat{f}_2(y)) \leq n5/6$, and the error between the coefficients of $\hat{f}(y)$ and $\hat{f}_1(y)\hat{f}_2(y)$ is at most $2^{-\pi}$. Each stage requires polynomial multiplication and division taking time $O(n \log n)$ by Lemma 2.1. Therefore, the total time is $O(n \log n \log \pi)= O(n \log n \log(\mu + m + n))$.

10. Apply our Real Root Algorithm recursively on polynomials $\hat{f}_1(y), \hat{f}_2(y)$, finding high accuracy approximations to the n roots $\hat{r}_1, \ldots, \hat{r}_n$ of $\hat{f}_1(y)$ and $\hat{f}_2(y)$ which differ from the roots of $\hat{f}_1(y)$ by at most $2^{-\pi}$.

11. Compute in $O(n \log n)$ time the set

$$\hat{S} = \bigcup_{i=1}^{n} \{\Phi^{-1}(\tilde{r}_i)\}$$

of at most $2n$ rational points, containing an n size subset $S = \{\tilde{r}_1, \ldots, \tilde{r}_n\}$ consisting of high accuracy approximations to the n roots $\{r_1, \ldots, r_n\}$ of $f(x)$, which differ from the roots of $f(x)$ by at most $2^{-\pi}$. The inverse variable transformation $\Phi^{-1}(y)$ involves taking 2^ρ-th roots. Since there can be two possible real 2^ρ-th roots of a real number, $\hat{S}$ contains at most $2n$ points. Each such scalar root computation can be done by $\rho = O(\log n)$ stages of taking square roots, which requires $O(\log n)$ time since square root is considered a basic operation in the arithmetic model of computation.

12. Find and output a set S of high accuracy approximations to the n roots of $f(x)$ by evaluating $f(x)$ at each point $x \in \hat{S}$, and testing if $f(x)$ is sufficiently small, say $< 2^{-\Omega(\pi)}$. This can be done by a multipoint evaluation algorithm (stated in Lemma 2.2) in time $O(n \log^2 n)$ (improving on the previous more than quadratic time root proximity test of Pan [P 87, P 89] based on Turin's test). We prove the correctness of our surprisingly simple root proximity verification test in Theorem 4.1.

Note that computing the 2^ρ-th roots in step 11 requires an additive term of $O(\rho) = O(\log n)$ further log-precision. Our algorithm will recurse at most $O(\log n)$ levels, thus requiring us to increase the required log-precision of calculations in the overall real root finding algorithm by an additive factor of only $O(\log^2 n)$. In addition to the above comment, the extensive precision analysis of Schönhage [S 82], Ben-Or, Feig, Kozen and Tiwari, [BFK 86] as stated by Lemmas 2.8, 2.9, 2.10, 2.11, limit the required precision of these calculations made by our algorithm to log-precision $\pi = O(n(\mu + m + n))$.

We have shown in the steps above that each recursive decomposition of a polynomial of degree n requires time at most

$$O(n(\log n)(\log n + \log \pi)) \leq O(n(\log n)(\log n + \log b)),$$

where $b = m + \mu$, since

$$\log \pi \leq O(\log(n(\mu + m + n))) \leq O(\log n + \log(n^2(\mu + m))) \leq O(\log n + \log b).$$

Thus, a recurrence equation for the arithmetic time complexity is:

$$T(n) \leq T(n') + T(n - n') + O(n \log n(\log n + \log b)),$$

where $n/6 \leq n' \leq n5/6$, and $T(1) = O(1)$. Thus, $T(n) \leq O(n \log^2 n(\log n + \log b))$, proving our main Theorem 5.1. ∎

By the known efficient reduction from the symmetric tridiagonal eigenvalue problem to the real root problem described in Subsection 1.3,

Corollary 5.1 *There is an algorithm for the symmetric tridiagonal eigenvalue problem which has arithmetic time cost* $O(n \log^2 n(\log n + \log b)$, *where* $b = m + \mu$.

6 Application of the Real Root Problem: the Symmetric Eigenvalue Problem in Sparse and Structured Matrices

Root finding has many applications, and one of the most important of these is finding the eigenvalues of a matrix. The eigenvalues are the roots of the characteristic polynomial of the matrix, which can found by a symbolic computation on the determinant. The *characteristic polynomial* of a matrix A is $det(A - I\lambda)$, where λ is an indeterminant.

Here consider various such classes of sparse and structured matrices occurring in practice which have special structure that allow the characteristic polynomial to be computed significantly faster than time $O(n^3)$. We can apply Theorem 5.1 to efficiently solve the symmetric eigenvalue problem for these classes of matrices. Thus, given efficient algorithms for the real root problem, we get efficient algorithms for the symmetric eigenvalue problem for these classes of matrices.

Eigenvalues are used in many engineering and scientific applications, including vibration analysis in structures, stability analysis, etc. The *symmetric eigenvalue problem* is: given a symmetric matrix (or more generally, a Hermitian matrix), find all the eigenvalues; which are all real in this case. Symmetric matrices and the corresponding symmetric eigenvalue problems occur naturally and frequently in applications and in fact most application programs for eigenvalues are for the symmetric eigenvalue problem. For example, in most of the cases where the matrices are derived from the solution of discretized PDEs, circuit problems, structure problems, and in signal processing, the resulting matrices are generally symmetric. Moreover, many large matrices occurring in practice have a special structure which allow the characteristic polynomial to be computed in nearly quadratic time. In addition to be generally symmetric, they often fall into one of two classes:

1. *Dense structured matrices* (see [BA 80, BGY 80, PR 87]), and in particular Toeplitz and bounded displacement rank matrices (defined in in Subsection 1.3), arise frequently in signal processing, coding theory, data compression, and algebraic computation applications. Pan [Pa 90] gave $O(n^2 \log n \log \log n)$ algorithms for computing the characteristic polynomial of these class of matrices.

2. *Sparse Matrices* (defined in in Subsection 1.3) arise from, for example, VLSI circuit problems, structure problems and discretization of $d = 2, 3$ dimensional PDEs, Canny, Kaltofen and Laksman [CKL 89] have applied Wiedemann's method of solving sparse linear systems (see [KS 91]) to computing the characteristic polynomial of sparse matrices in $O(n^2 \log^2 n)$ time.

7 Conclusion

We have upper bounded the arithmetic complexity of the real root problem to be nearly the same as basic arithmetic operations on polynomials. We have shown that there is an algorithm for the real root problem which has time cost $O(n \log^2 n(\log n + \log b))$ in the arithmetic model where $b = m + \mu$. This is with polylog factors of optimality. There an immediate extension to improving the parallel complexity of the real root problem, since our real root algorithm can easily seen to be executable in parallel. Our most costly operation which respect to parallel arithmetic complexity in the $O(\log n)$ recursive levels is the Sturm sequence evaluation. In a subsequent paper we show that parallelization of the techniques of this paper and application of a parallel algorithm of Reif [R 95] for inverse of various structured matrices including Toeplitz can be used to get an efficient parallel arithmetic algorithm for the real root problem.

It remains an open problem to determine a similarly efficient algorithm in the arithmetic model for the general root problem with complex roots.

8 Acknowledgments

The Author wishes to thank Kathy Redding and Ken Robinson for excellent help in editing the paper, Steve Tate, Hongyan Wang, Shenfeng Chen, Salman Azhar, and especially Deganit Armon for their help in providing technical edits for this paper. Thanks to Andrew Neff for helpful technical comments. Also, I wish to thank Dario Bini and Victor Pan for pointing out the known relationship between the real root problem and the symmetric tridiagonal eigenvalue problem.

References

[AHU 74] A.V.Aho, J.E.Hopcroft and J.D.Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, (1974).

[B 80] E.J. Barbeau, *Polynomials*, Springer-Verlag, (1980).

[BFK 86] M.Ben-Or, E.Feig, D.Kozen and P.Tiwari, *A Fast Parallel Algorithm for Determining All Roots of a Polynomial with Real Roots*, SIAM J. Comput., (1986).

[BT 90] M.Ben-Or and P.Tiwari, *Simple Algorithms for Approximating All Roots of a Polynomial with Real Roots*, Journal of Complexity **6**, (1990), 417–442.

[B 92] D. Bini, *Divide and conquer techniques for the polynomial root-finding problem*, Proceedings of the 1st World (International) Congress of Nonlinear Analysts, Tampa, (August, 1992).

[BG 92] D. Bini, L. Gemignani, *On the complexity of polynomial zeros*, SIAM J. Comput., **21**,4, 781-799, 1992.

[BG 93] D. Bini, L. Gemignani, *Iteration schemes for the divide-and-conquer eigenvalue solver*, Numerische Mathematik (to appear).

[BP 86] D. Bini and V. Pan, *Polynomial Division and its computational complexity*, Journal of Complexity **2**, (1986), 179-203.

[BP 91] Bini D., Pan V.,*Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem*, Proc. 2nd Annual ACM–SIAM Symp. on Discrete Algorithms, pp. 384–393, (1991).

[BP 92] Bini D., Pan V., *Practical Improvement of the Divide-and-Conquer Eigenvalue Algorithms*, Computing **48**, pp. 109-123, (1992).

[BA 80] R.R.Bitmead and D.O.Anderson, *Asymptotically fast solution of Toeplitz and related systems of linear equations*, Linear Algebra and its Applications **34**, (1980), 103–116.

[BGY 80] R.P.Brent, F.G.Gustavson, and D.Y.Y.Yun, *Fast solution of Toeplitz systems of equations and computation of Padé approximants*, J. Algorithms **1**, (1980), 259–295.

[BNS 78] Bunch J. R., Nielsen C. P., Sorensen D. C., *Rank-one modification of the symmetric eigenproblem*, Numer. Math. **31**, pp. 31–48, (1978).

[BP 60] W.S.Burnside and A.W.Panton, *The Theory of Equations, Vols. 1 and 2*, Dover, (1960).

[CKL 89] J.F.Canny, E.Kaltofen and Y.Laksman, *Solving System of Nonlinear Polynomial Equations Faster*, Proc. of International Symposium on Symbolic and Algebraic Computation, (1989).

[C 66] G.E.Collins, *Polynomial Remainder Sequences and Determinants*, Amer. Math. Monthly **73**, 708–712, (1966).

[CL 82] G.E.Collins and R.Loos, *Real Zeros of Polynomials*, Computing, (1982), Springer-Verlag.

[C 81] Cuppen J.J.M., *A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem*, Numer. Math., **36**, pp. 177-195, (1981).

[DS 87] Dongarra J. J., Sorensen D. C., *A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem*, SIAM J. on Sci. and Stat. Computing, 8, **2**, pp. 139–154, (1987).

[GM 77] S. Gal and W. Miranker, *Optimal Sequential and Parallel Search for Finding a Root*, Journal of Combinatoral Theory **23**, (1977).

[GH 72] I.Gargantini and P.Henrici, *Circular Arithmetic and the Determination of Polynomial Zeros*, Num. Math. **18**, (1972), 305–320.

[GL 83] G.H.Golub and C.F.van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, (1983).

[H 76] O.H. Hald, *Inverse Eigenvalue Problems for Jacobi Matrices, Linear Algebra Appl.*, 14, (1976) 63-85.

[H 78] D. Heller, *A survey of parallel algorithms in numerical linear algebra* SIAM Review, 20(4):740-777, (1978).

[H 74] P.Henrici, *Applied and Computational Complex Analysis, Vols. 1, 2, and 3*, Wiley, (1974).

[H 70] A.S.Householder, *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill, (1970).

[J 92] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, (1992).

[J 89] Jessup E. R., Parallel solution of the symmetric tridiagonal eigenproblem, YALEU/D CS /RR 728, Yale University, October, (1989).

[KM 86] A.S. Krishnakumar, M. Morf, *Eigenvalues of a symmetric tridiagonal matrix: a divide-and-conquer approach*, Numer. Math, **48**,(1986), 349-368.

[KS 91] E.Kaltofen and B.D.Saunders, *On Wiedemann's method of solving sparse linear systems*, in Proc. AAECC-5, Lecture Notes in Computer Science, **536**, 216–226, Springer-Verlag, 1991.

[LRT 79] R.J.Lipton, D.Rose and R.E.Tarjan, *Generalized Nested Dissection*, SIAM Journal on Numerical Analysis **16**, (1979), 346–358.

[M 64] K.Mahler, *An Inequality for the Discriminant of a Polynomial*, Mich. Math. J., **11**, (1964), 257–262.

[M 66] M.Marden, *Geometry of Polynomials*, American Mathematical Society, (1966).

[M 92] M.Mignotte, *Mathematics for Computer Algebra*, Springer-Verlag, (1992).

[N 90] C.A.Neff, *Specified Precision Polynomial Root Isolation is in NC*, Proc. 24th Annual IEEE Symp. on Foundations of Computer Science, 138–145, (1990).

[NR95] C.A.Neff and J.H.Reif, *An Efficient Algorithm for the Complex Roots Problem*, to appear in Journal of Complexity, (1996).

[P 87] V.Pan, *Sequential and Parallel Complexity of Approximate Evaluation of Polynomial Zeros*, Comput. Math. Applic. **14**, (1987), 591–622.

[Pa 87] V.Pan, *Fast and Efficient Parallel Toeplitz Computations*, S.U.N.Y. Albany Tech. Report TR 88-8, (1987).

[Pn 87] V.Pan, *Algebraic Complexity of Computing Polynomial Zeros*, Comput. Math. Applic. **14**, (1987), 285–304.

[P 89] V.Pan, *Fast and Efficient Parallel Evaluation of the Zeros of a Polynomial Having Only Real Zeros*, Computers Math. Applic. **17**, (1989), 1475-1480.

[Pa 90] V.Pan, *Parameterization of Newton's Iteration for Computations with Structured Matrices and Applications*, Tech. Report CUCS-032-90, Computer Science Dept., State Univ. of New York at Albany, Albany, NY, (1990).

[PR 87] V.Pan and J.H.Reif, *Some polynomial and Toeplitz matrix computations*, Proc. 28th Annual IEEE Symposium on Foundations of Computer Science(FOCS), (1987), 173–184.

[PS 85] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag,, (1985).

[R 93a] J.H.Reif, ed., *Synthesis of Parallel Algorithms*, Morgan Kaufmann, (1993).

[R 95] *Work Efficient Parallel Solution of Toeplitz Systems and Polynomial GCD*, Proc. 27th Annual ACM Symposium of Theory of Computing (STOC'95), Las Vegas, Nevada, (May, 1995). rewritten as *Efficient Parallel Factorization and Solution of Structured and Unstructured Linear Systems* (see http://www.cs.duke.edu/∼reif/paper/paper.html) and submitted for Journal publication.

[R 87] J.Renegar, *On the Worst-Case Arithmetic Complexity of Approximating Zeros of Polynomials*, Proc. 2nd Symp. on the Complexity of Approx. Solved Problems, (1987).

[S 82] A.Schönhage, *The Fundamental Theorem of Algebra in Terms of Computational Complexity*, Univ. of Tübingen, Tübingen, Germany, Unpublished Manuscript, (1982).

[SS 85] M.Shub and S.Smale, *Computational Complexity: On the Geometry of Polynomials and a Theory of Cost, Part I*, Annls. Scient. Ecole Norm. Sup., **4**, (1985), 107–142.

[SS 86] M.Shub and S.Smale, *Computational Complexity: On the Geometry of Polynomials and a Theory of Cost, Part II*, SIAM J. Computing **15**, (1986), 145–161.

[S 81] S.Smale, *The Fundamental Theorem of Algebra and Complexity Theory*, Bull. A.M.S. **4**, (1981), 1–36.

[S 85] S.Smale, *On the Efficiency of the Algorithms of Analysis*, Bull. A.M.S. **13**, (1985), 87–121.

[S 86] S.Smale, *Newton Method Estimates from Data at One Point*, Proc. Conf. in Honor of Gail Young, (1986).

[Sc 80] J.T.Schwartz, *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM **27**, (1980), 701–717.

[T 68] P.Turan, *On the Approximate Solution of Algebraic Equations*, Comm. Math. Phys. Class Hung. Acad. **XVIII**, (1968), 223–236.

[T 75] P.Turan, *Power Sum Method and Approximative Solution of Algebraic Equations*, Math. Computation **29**, (1975), 311–318.

[T 84] P.Turan, *On a New Method of Analysis and Its Applications*, Wiley, (1984).

9 Appendix: Known Algorithms Used as Subroutines by our Real Root Finding Algorithm

9.1 Sturm sequences and Root Isolation

A *Sturm sequence* of polynomials $f_0(x), f_1(x)$ is the sequence of polynomials $f_0(x), f_1(x), \ldots, f_k(x)$ where for $i = 1, 2, \ldots k-1$, $f_{i+1}(x) = q_i(x) f_i(x) - f_{i-1}(x)$, the $q_i(x)$ are linear, and $f_k(x)$ is constant. Sturm defined this sequence in 1829. See [BP 60] for a survey on Sturm sequences.

The (standard) *Sturm sequence* of polynomial $f(x)$ is defined to be the Sturm sequence of $f(x) = f_0(x), f'(x) = f_1(x)$. Since $f'(x)$ has degree $n-1$, the Sturm sequence of $f(x)$ has length $k = n$.

Note that the Sturm sequence of $f_0(x), f_1(x)$ is similar to the remainder sequence generated by the Euclidean algorithm for the $gcd(f_0(x), f_1(x))$ except that in the case of the Sturm sequence, $f_{i+1}(x)$ is the negative of the remainder of the division of $f_{i-1}(x)$ by $f_i(x)$. Therefore, by simple modification (see Schwartz [Sc 80] for details) of the usual HGCD algorithms (see [AHU 74]) used to compute gcd,

Lemma 9.1 *The Sturm sequence can be computed in time $O(n \log^2 n)$.*

Note: The precision required of the Sturm sequence as defined above can be quite high; but this can be easily remedied by use of additional indeterminants as linear factors, yielding a Sturm sequence computation requiring lower precision; see Collins [C 66] and [BFK 86] and also Ben-Or and Tiwari [BT 90] for details.

The applications of Sturm sequences use the following lemma, attributed to Rolle; see Marden, 1966 [M 66], Collins and Loos [CL 82], and Mignotte [M 92]:

Lemma 9.2 *If $f(x)$ has real coefficients and all roots are real, then the roots of $f'(x)$ are all real and they strictly interleave the roots of $f(x)$.*

For a real a, let V_a be the number of sign variations of the Sturm sequence $f_0(a), f_1(a), \ldots, f_k(a)$, that is, the number of times $f_i(a) \cdot f_{i+1}(a) < 0$. The following is proven using the result of Rolle:

Lemma 9.3 *(Jacobson, 1974) For any interval $[a, b]$ of the real line, the number of real roots in this interval is $V_a - V_b$.*

Let the zeros of the linear terms $q_i(x)$ be ordered $y_{i_1} \leq y_{i_2} \leq \ldots \leq y_{i_k}$ and let the roots of $f(x)$ be ordered $r_1 < r_2 < \ldots < r_n$. Recall the definitions for σ-isolated, well isolated, and balanced splitting intervals as given in Subsection 1.5. Ben-Or, Feig, Kozen and Tiwari [BFK 86] prove the remarkable result that:

Lemma 9.4 *There is a j such that y_{i_j} is a $1/4$-splitting point for the roots of $f(x)$*

Using a binary search of $O(\log n)$ stages on the sequence $y_{i_1} \leq y_{i_2} \leq \ldots \leq y_{i_k}$ and applying Lemma 9.3 to count the number of roots of $f(x)$ in the appropriate interval considered at each stage of this binary search, we get

Lemma 9.5 *There is an $O(n \log^2 n)$ algorithm for finding a $1/4$-splitting point for the roots of $f(x)$.*

9.2 Approximating The Magnitudes Of All the Roots

A method due to Turin in 1968 [T 68, T 75, T 84] can be used to determine approximations to the magnitudes of all the roots of a polynomial. Schönhage [S 82] describes an algorithm which uses Turin's method to approximating the magnitudes of all the roots by a certain ratio. This algorithm takes an input a polynomial $f(x)$ of degree n, and for each root r of $f(x)$, determines the magnitude of r to be within an interval $[L, U]$, where $U/(2n) \leq 2nL$. Pan [P 89] shows this algorithm takes $O(n \log n)$ time using a reduction to 2D convex hull, for which there are many known $O(n \log n)$ time algorithms[PS 85]. Pan [P 87] observes that $g = O(\log(\log(2n)/\log(1-\gamma^2)))$ iterations of Graeff's method (which is a technique involving polynomial powering described in Section 9.3) improves these root bounds to ratio $(1+\gamma)$. Since for $\gamma = 1/n^{O(1)}$, $g = O(\log n)$, this gives the following result:

Lemma 9.6 *Given a polynomial of degree n, there is a $O(n \log^2 n)$ algorithm which, for all $i = 1, \dots n$, determines the magnitude of each root r_i of $f(x)$ to be within an interval $[L_i, U_i]$, where $U_i \leq L_i(1+\gamma)$, for $\gamma = 1/n^{O(1)}$.*

9.3 Obtaining a Highly Isolated Interval by Polynomial Powering

The following is known as Graeff's Method, but is actually due to Dandelin, 1826. Given a monic polynomial $f_0(x) = \prod_{i=1}^{n}(x - r_i)$ of degree n, let $f_i(x) = f_{i-1}(\sqrt{x}) \cdot f_{i-1}(-\sqrt{x})$ for $i > 0$. Note that $f_1(x) = (-1)^n \prod_{i=1}^{n}(x - r_i^2)$, so $f_1(x)$ has the same degree n but the roots of $f_1(x)$ are the squares of the roots of $f_0(x)$. Thus, for any $i > 0$, $f_i(x) = (-1)^{ni} \prod_{i=1}^{n}(x - r_i^{2^i})$, so $f_i(x)$ has the same degree n but the roots of $f_i(x)$ are the 2^ith powers of the roots of $f_0(x)$.

Given a well isolated ϵ-splitting interval $I = [\alpha - \delta, \alpha + \delta]$ for the roots of $f(x)$, let $x = \alpha + y$, and let $g_0(y) = y^n f(\alpha + 1/y)/f(\alpha)$ be the monic polynomial of degree n derived from $f(x)$. Let $\rho = \sigma \log n$, for any constant $\sigma > 0$. Applying Graeff's Method to g_0, we compute $g_i(y) = g_{i-1}(\sqrt{y}) g_{i-1}(-\sqrt{y})$ for $i = 1, 2, \dots, \rho$, resulting in a degree n polynomial $\hat{f}(y) = g_\rho(y)$ which has roots which are the 2^ρth powers of the roots of $g_0(y)$. Note that the corresponding interval $\hat{I}$ for $\hat{f}(y)$ is σ-isolated, ϵ-splitting interval for the roots of $\hat{f}(y)$. The variable transformation $\Phi(x) = y$ involves scalar calculations including reciprocals and taking 2^ρ-th powers of scalars.

Lemma 9.7 *Given well isolated ϵ-splitting interval I for the roots of $f(x)$, for any constant $\sigma > 0$, there is an $O(n \log^2 n)$ algorithm for constructing a degree n polynomial $\hat{f}(y)$ and an interval $\hat{I}$ which is a σ-isolated ϵ-splitting interval for the roots of $\hat{f}(y)$. Furthermore, a set of $2n$ points containing all the roots of $f(x)$ can be obtained by the inverse variable transformation $\Phi^{-1}(y) = x$ involving scalar calculations including reciprocals and taking 2^ρ-th roots of scalars.*

9.4 FFT Approximations to Contour Integrals

Let $r_1, \dots, r_{n'}$ be the real roots of monic polynomial $f(x)$ within the interval I. Let Γ be a circle on the complex plane of diameter length(I) and such that the endpoints of interval I are on Γ (see figure 5.1).

The Cauchy formula states that if Γ is a closed curve on the complex plane, then

$$s_k = \frac{1}{2\pi\sqrt{-1}} \int_{z \in \Gamma} z^k \frac{f'(z)}{f(z)} dz$$

where $s_k = r_1^k + \dots + r_{n'}^k$ is the kth power sum of the roots in I.

Schönhage [S 82], (also see Pan [P 87]) shows

Lemma 9.8 *If Γ is the unit disk and interval I is σ-isolated, then the integral can be approximated by the Nth roots of unity within error $\leq 2n/(1+\sigma)^N$.*

First shift and deflate the polynomial $f(x)$ by the variable mapping $x \mapsto (x - center(I))/radius(I)$ so the resulting mapped interval is $[-1, 1]$. Then evaluate the Cauchy contour integral at the roots of unity by a FFT (see figure 5.2) in $O(n \log n)$ time, thus giving:

Lemma 9.9 *Given a degree n polynomial $f(x)$ and an interval I which is a σ-isolated interval for the roots of $f(x)$, for a constant $\sigma > 0$, then in time $O(n \log n)$ the n power sums of the roots of $f(x)$ can all be computed within precision $\leq 2n/(1+\sigma)^n$.*

9.5 Polynomial Splitting Using The Newton Identities

Given the power sums $s_1, \ldots, s_n$ of a monic polynomial $f(x) = \sum_{i=0}^{n} c_i x^i$, with $c_n = 1$, the coefficients of $f(x)$ can be related to the power sums $s_1, \ldots, s_n$ of the roots of $f(x)$ by the following linear system, known as Newton's Identities:

$$\begin{bmatrix} 1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \ddots & \ddots & \ddots & \\ s_n & s_{n-1} & \ldots & s_1 & n \end{bmatrix} \begin{bmatrix} c_{n-1} \\ c_{n-2} \\ c_{n-3} \\ \vdots \\ c_0 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix}.$$

Lemma 9.10 *Given power sums of a monic polynomial $f(x)$ within precision $\leq 2n/(1+\sigma)^n$, we can compute the coefficients of $f(x)$ within the same order of accuracy.*

This linear system is a triangular system, with constant displacement rank [BP 86], and can be solved by applying known $O(n \log n)$ algorithms [BP 86].

Lemma 9.11 *Given a degree n polynomial $f(x)$ and an interval I which is a σ-isolated ϵ-splitting interval for the roots of $f(x)$, for a constant $\sigma > 0, 0 < \epsilon < 1$, then in time $O(n \log n)$, $f(x)$ can be approximately separated into a product $g_1(x)g_2(x)$ where $\epsilon n \leq deg(g_1(x))$, $deg(g_2(x)) \leq (1-\epsilon)n$, and the coefficient error between $f(x)$ and $g_1(x)g_2(x)$ is at most $2n/(1+\sigma)^n$.*

9.6 High Accuracy Polynomial Splitting Using Newton Iteration

We will apply the Newton's iteration method developed by Schönhage ([S 82], Chapt 10-11, pages 26-32) to exponentially improve the accuracy of a low accuracy polynomial splitting. His algorithm requires $O(\log \pi)$ stages, where each stage requires polynomial multiplication and division taking arithmetic time $O(n \log n)$ by Lemma 2.1 (note that, in contrast, Schönhage analyzed his algorithm in the bit-complexity model). Therefore the total time is $O(n \log n \log \pi)$.

Lemma 9.12 *Given a degree n polynomial $f(x)$ and an interval I which is a σ-isolated ϵ-splitting interval for the roots of $f(x)$, for constants $\sigma > 0, 1 > \epsilon > 0$, and where $f(x)$ is approximately separated into $f_1(x)f_2(x)$ where*

$$\epsilon n \leq deg(f_1(x)), deg(f_2(x)) \leq (1-\epsilon)n,$$

and coefficient error between $f(x)$ and $f_1(x)f_2(x)$ is at most $2n/(1+\sigma)^n$, then in time $O(n \log n \log \pi)$, $f(x)$ can be approximately separated as $f_1(x)f_2(x)$, where $\epsilon n \leq deg(f_1(x))$,$deg(f_2(x)) \leq (1-\epsilon)n$, so that the coefficient error between $f(x)$ and $f_1(x)f_2(x)$ is at most $2^{-\pi}$.

We can apply this lemma of Schönhage to obtain arbitrarily high precision polynomial splittings.