SqFreeEVAL: An (almost) optimal real-root isolation algorithm

Michael A. Burr[a,c], Felix Krahmer[b,c]

[a] *Fordham University, 441 East Fordham Road, Bronx, NY 10458, USA*
[b] *Hausdorff Center for Mathematics, Universität Bonn, Endenicher Allee 60, 53115 Bonn, Germany*
[c] *Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA*

Abstract

Let f be a univariate polynomial with real coefficients, $f \in \mathbb{R}[X]$. Subdivision algorithms based on algebraic techniques (e.g., Sturm or Descartes methods) are widely used for isolating the real roots of f in a given interval. In this paper, we consider a simple subdivision algorithm whose primitives are purely numerical (e.g., function evaluation). The complexity of this algorithm is adaptive because the algorithm makes decisions based on local data. The complexity analysis of adaptive algorithms (and this algorithm in particular) is a new challenge for computer science. In this paper, we compute the size of the subdivision tree for the SqFreeEVAL algorithm.

The SqFreeEVAL algorithm is an evaluation-based numerical algorithm which is well-known in several communities. The algorithm itself is simple, but prior attempts to compute its complexity have proven to be quite technical and have yielded sub-optimal results. Our main result is a simple $O(d(L+\ln d))$ bound on the size of the subdivision tree for the SqFreeEVAL algorithm on the benchmark problem of isolating all real roots of an integer polynomial f of degree d and whose coefficients can be written with at most L bits.

Our proof uses two amortization-based techniques: First, we use the algebraic amortization technique of the standard Mahler-Davenport root bounds to interpret the integral in terms of d and L. Second, we use a continuous amortization technique based on an integral to bound the size of the subdivision tree. This paper is the first to use the novel analysis technique of continuous amortization to derive state of the art complexity bounds.

Keywords: Continuous Amortization, Adaptive Analysis, Subdivision Algorithm, Integral Analysis, Amortization, Root Isolation

Email addresses: `mburr1@fordham.edu` (Michael A. Burr), `felix.krahmer@hcm.uni-bonn.de` (Felix Krahmer)

1. Introduction

In this paper, we show that the size of the subdivision tree for the simple, evaluation-based, numerical algorithm `SqFreeEVAL` has size $O(d(L + \ln d))$ for the benchmark problem of isolating all of the real roots of an integer polynomial of degree d whose coefficients can be represented by at most L bits. Under the mild assumption that $L \geq \ln d$, this complexity simplifies to the optimal size of $O(dL)$, see (Eigenwillig et al., 2006, Section 3.3) for a proof of optimality. The optimality and simplicity of the `SqFreeEVAL` algorithm imply that it may be a useful algorithm in practical settings. The bound on the size of the subdivision tree is achieved via a straight-forward and elementary argument. The two main techniques which are used in the computation are algebraic amortization, in the form of Mahler-Davenport bounds, and continuous amortization, in the form of an integral technique as presented in (Burr et al., 2009).

1.1. `EVAL`-type algorithms

The `SqFreeEVAL` algorithm which we study in this paper is a specific example of what we call an `EVAL`-type algorithm. These algorithms are so named because they are based on function evaluation: `EVAL`-type algorithms take, as input, functions which allow some subset of the following two predicates: First, these functions and their derivatives can be evaluated at a countable dense subset of their domain. In this paper, the domain will be the real numbers and the countable dense subset will be the dyadic integers. Second, these functions and their derivatives can be approximated on intervals in such a way that the approximation converges as the input intervals converge to a point. In this paper, the approximation is derived from interval arithmetic on a Taylor sequence. The simplest and most well-known example of an `EVAL`-type algorithm is Lorensen and Cline's marching cube algorithm (Lorensen and Cline, 1987).

`EVAL`-type algorithms are typically studied because of their simplicity and generality. These algorithms are fairly general because their inputs can be extended to more general analytic functions. In particular, many analytic functions have interval arithmetic available to them, and, therefore, it is possible to approximate these functions on intervals. In addition, with the limited predicates available to `EVAL`-type algorithms, most of the techniques which are used in these algorithms are analytically based (as opposed to algebraically based). These algorithms are simple because, in many cases, `EVAL`-type algorithms are based on simple recursive bisection algorithms. Such algorithms iteratively subdivide an initial domain until each set in the resulting partition of the initial domain satisfies a (usually simple) terminal condition. Bisection algorithms are common in computer graphics (Boier-Martin et al., 2005) as well as in computational science and engineering applications (International Conference on Domain Decomposition Methods). Bisection algorithms are of particular interest because they are adaptive; they perform more bisections near difficult features and fewer bisections elsewhere. However, this adaptivity makes the complexity analysis of such algorithms more difficult because the subdivision tree may have a few deep paths while the remainder of the tree remains modest in size.

`EVAL`-type algorithms have been studied in the univariate case in (Henrici, 1970; Yakoubsohn, 2005; Sagraloff and Yap, 2009; Yap and Sagraloff, 2011; Burr et al., 2011, 2009), in the bivariate and trivariate cases in (Lorensen and Cline, 1987; Snyder, 1992; Plantinga and Vegter, 2004; Plantinga, 2006; Lin and Yap, 2009; Burr et al., 2010), and in the multivariate case in (Galehouse, 2009; Dedieu and Yakoubsohn, 1992). All of these algorithms are devoted to approximating algebraic (and in some cases analytic varieties) in the real or complex settings. The algorithms in (Burr et al., 2011, 2009) are designed to find all real roots of a polynomial or analytic function while the algorithms in (Henrici, 1970; Yakoubsohn, 2005; Sagraloff and Yap, 2009; Yap and Sagraloff, 2011) are designed to find the complex roots of a polynomial or analytic function (note that (Henrici, 1970) is only designed to find a single root of a polynomial). Each of these algorithms is very closely related to the `SqFreeEVAL` algorithm considered in this paper; the main differences are in the setting, in the type of subdivisions performed, and in various preprocessing steps. We give a more detailed account of these algorithms in the next section. The two-dimensional `EVAL`-type algorithm (Plantinga and Vegter, 2004; Plantinga, 2006) was presented for approximating smooth and bounded varieties. It was extended to singular and unbounded varieties in (Burr et al., 2010); in addition, the tests performed by the algorithm were improved in (Lin and Yap, 2009).

1.2. The `SqFreeEVAL` algorithm

There are many bisection algorithms for finding roots, see Section 1.5 for references, but among such algorithms, the `SqFreeEVAL` algorithm is one of the simplest and most widely applicable, see (Burr et al., 2011). There are two distinct paths in the literature which arrive at algorithms similar to the `SqFreeEVAL` algorithm: one path proceeds through the consideration of magnitudes of derivatives and the other path proceeds via interval arithmetic.

We begin by discussing the history from the magnitudes of derivatives perspective. In (Henrici, 1970), the author presents an algorithm for finding a single complex root of a polynomial. The test T_3 from the paper is essentially used here. In (Yakoubsohn, 2005), the test is developed into a bisection algorithm and to find all complex roots of entire functions, not just polynomials. In the paper, however, the test from (Henrici, 1970) is used only as a one-sided test; therefore, the algorithm can only exclude regions from containing roots and does not confirm that roots exist in the final regions. There, the algorithm was termed a *bisection-exclusion* algorithm to reflect this drawback. Finally, in (Sagraloff and Yap, 2009; Yap and Sagraloff, 2011), the algorithm from (Yakoubsohn, 2005) was adapted to polynomials in order to confirm that roots exist in the final regions; there, the authors studied both an algorithm for finding complex roots as well as one for finding real roots. The `SqFreeEVAL` algorithm is a natural restriction of these complex root-finding algorithms to the real line.

On the other hand, from the interval arithmetic community, a bisection algorithm using interval methods was suggested in (Moore, 1966; Mitchell, 1990). In these papers, any interval function can be used; if the standard centered form for polynomials is used, see (Ratschek and Rokne, 1984), then the exclusion conditions are identical (when f and f' are square free) to those for the `SqFreeEVAL` algorithm.

In this paper, we study the `SqFreeEVAL` algorithm on the standard benchmark problem of finding all of the real roots of a polynomial. We show that, in this case, the subdivision tree has the favorable size of $O(d(L + \ln d))$ which simplifies to the optimal size of $O(dL)$ under the mild assumption that $L \geq \ln d$. Since this algorithm uses only local data to find roots, it is an adaptive algorithm and may be more efficient than the standard exact algorithms in certain cases, see (Burr et al., 2011). In addition, the `SqFreeEVAL` algorithm can handle analytic varieties, see (Burr et al., 2011), which extends its reach beyond that of more standard exact algorithms which require sophisticated algebraic primitives and are specialized to polynomials. These advantages of the `SqFreeEVAL` algorithm imply that it may be more practical than other standard root isolation algorithms in practice.

1.3. Previous complexity results

The computational complexity of `EVAL`-type algorithms has proven to be quite a challenging problem because the algorithms are adaptive and the analytic primitives do not carry much information about the global structure (unlike algebraic information). Here, we survey the methods of complexity analysis of the precursors to the `SqFreeEVAL` algorithm. In most situations, the complexity is computed in terms of the size of the subdivision tree of the specific `EVAL`-type algorithm (this is almost equivalent to counting the number of tests performed by the algorithm). There have been two main techniques to find the size of the subdivision tree: by finding the width of the subdivision tree at various subdivision levels or by finding the local depth of the subdivision tree.

In (Henrici, 1970), the author is searching for only a single root, and, therefore, retains a single disk containing a root at each stage of the algorithm. Many tests are performed in the algorithm, however, because at each stage of the algorithm, tests are performed on a covering of the previously retained disk. The final stopping criterion for this algorithm is based on a precision $\epsilon > 0$ which is chosen *a priori* by the user. When the worst-case root separation bound for a polynomial is used, the complexity of the subdivision tree becomes $O(d^3(L + \ln d))$.

In (Yakoubsohn, 2005), the author is searching for all of the complex roots of an analytic function. In the computation, a bound on the width of the tree is computed to bound the number of subdivisions performed. Since this algorithm only excludes regions and lacks an inclusion test, it is possible that the final output regions do not contain roots or do not separate roots. The final stopping criterion for this algorithm is based on a precision $\epsilon > 0$ which is chosen *a priori* by the user. When the worst-case root separation bound for a polynomial is used, the complexity of the subdivision tree becomes either $O(d^4(L + \ln d))$ or $O(d^3(L + \ln^3 d))$ after $\lceil \ln d \rceil$ steps of the Graeffe iteration.

In (Burr et al., 2009), we search for all of the real roots of a polynomial. Here, the computation is based on the depth of the tree over each point of the initial interval. In the paper, we introduced the idea of continuous amortization via an integral and showed how to use it to bound the size of the subdivision tree. In particular, we proved a complexity bound of $O(d^3(L + \ln d))$ for the subdivision tree.

In (Sagraloff and Yap, 2009; Yap and Sagraloff, 2011), the authors present algorithms to find all of the real or all of the complex roots of a polynomial. In the computation, a bound on the width of the subdivision tree is used to compute the number of subdivisions performed. The authors show that the complexity of the subdivision tree is $O(d(L + \ln d)(\ln L + \ln d))$ in the real case and $O((d \ln d)^2(L + \ln d))$ in the complex case. In addition, the authors show that the bit complexity of both of these algorithms is $\widetilde{O}(d^4L^2)$ where the $\widetilde{O}$ means that logarithmic factors in d and L have been suppressed.

Each of the methods of analysis in (Yakoubsohn, 2005; Burr et al., 2009; Sagraloff and Yap, 2009; Yap and Sagraloff, 2011) are quite technical, complicated, and require several constants to be defined whose use becomes justified only after the completion of the complexity analysis. In contrast, the computation in this paper is quite simple and provides the better bound of $O(d(L+\ln d))$. It should be noted that although this is the best bound known, it does not directly replace the bounds presented in these papers because some are in the different setting of the complex plane and others use different preprocessing steps. In the case where the polynomial and its derivative are both square free and we are searching for the real roots, all of these algorithms are identical and our bound on the subdivision tree is the best.

1.4. Algebraic and continuous amortization

In this paper, we use amortization in two forms: algebraic and continuous. Algebraic amortization originated with Davenport (Davenport, 1985) where the individual real root separation bounds are replaced by a product of root separations. This bound was then studied in (Johnson, 1991; Du et al., 2007; Eigenwillig et al., 2006; Mignotte, 1995; Emiris et al., 2008) where it was generalized to other root separation products including complex roots. This technique has proven useful to compute the complexity of the subdivision tree for many other root isolation techniques, see Section 1.5. We introduced continuous amortization in (Burr et al., 2009) to bound the size of the subdivision tree of an `EVAL`-type algorithm. In this paper, we show that continuous amortization can be used to significantly simplify complexity calculations.

In continuous amortization, we use a complexity charge ϕ whose domain is the input region, and, for each x in the input region, $\phi(x)$ is a lower bound on the size of any leaf interval containing x. We call such a function a *stopping function* for the algorithm. In this situation, $1/\phi(x)$ is related to the depth of the subdivision tree for an interval which contains x. Note that a stopping function ϕ is theoretical and is never meant to be implemented: it is used only in the computation of the size of the subdivision tree, and it is not used in the algorithm itself. Functions similar to stopping functions also appeared in (Henrici, 1970) where they were called *inner* and *outer convergence* functions.

In (Yakoubsohn, 2005) such functions were also termed *exclusion functions*. In both cases, these stopping functions were used to compute the complexity of the algorithm, but they were not used in a continuous amortization computation.

Continuous amortization was first used in (Burr et al., 2009), to compute the size of a subdivision tree for an `EVAL`-type algorithm. In the paper, an upper bound on the integral resulting from continuous amortization is computed by using algebraic amortization-based bounds. In Section 4, we greatly simplify the computation to provide a complexity bound of $O(d(L + \ln d))$ for the size of the subdivision tree for the `SqFreeEVAL` algorithm.

1.5. Other root isolation algorithms

There is an extensive amount of literature on the complexity of root isolation, see (Pan, 1997, 1996) for surveys of the previous literature, which we will not attempt to cover here. Most algorithms are compared by their performance on the benchmark problem of finding all real roots of a polynomial of degree d and whose coefficients can be represented by at most L bits. For this problem, the bit-complexity of $O(d^3(L + \ln d))$ for complex roots was first achieved by Schönhage (Schönhage, 1982). In many algorithms, the size of the subdivision tree is smaller than this bound because, for each node in the subdivision tree, additional calculations must be performed. Davenport (Davenport, 1985) proved that the the subdivision tree for the Sturm method is $O(d(L+\ln d))$, see (Reischert, 1997; Lickteig and Roy, 2001; Du et al., 2007; Emiris et al., 2008; Johnson, 1991). More recently, it has been shown in (Eigenwillig et al., 2006; Emiris et al., 2008) that the Descartes method also achieves this bound, see (Collins and Akritas, 1976; Eigenwillig et al., 2006; Krandick and Mehlhorn, 2006; Collins et al., 2002; Sagraloff, 2011; Johnson, 1991). These methods are optimal under the weak assumption that $L \geq \ln d$. In addition, related exact techniques using continued fractions were shown to have a tree size of $\widetilde{O}(dL)$ when an ideal root bound is used and $\widetilde{O}(d^2L)$ when a more practical bound is used (Sharma, 2008); in the expected case, the tree was also shown in (Tsigaridas and Emiris, 2008) to have an expected size of $O(d^2 + dL)$. In the algebraic computing community, the Descartes method appears to be one of the more practical algorithms, see (Collins et al., 2002; Johnson, 1998; Rouillier and Zimmermann, 2004; Mourrain et al., 2005; Rouillier and Zimmermann, 2004). For an empirical comparison of these methods, see (Hemmer et al., 2009). In this paper, we show that the subdivision tree for the `SqFreeEVAL` algorithm also achieves this bound; therefore, the `SqFreeEVAL` algorithm should also be considered on equal footing with the other more well-known root finding algorithms via the Sturm or Descartes methods. The `SqFreeEVAL` algorithm may, in addition, be considered practical because its computations are numerical and hence easy to implement and its subdivision tree has a favorable size.

1.6. Organization of this paper

In Section 2, we introduce the `SqFreeEVAL` algorithm and discuss the main condition we will use for an interval to be `SqFreeEVAL` terminal. In Section 3, we review the use of stopping functions to bound the size of the subdivision from (Burr et al., 2009) and create a stopping function for the `SqFreeEVAL` algorithm. In Section 4, we compute the size of the `SqFreeEVAL` algorithm's subdivision tree using continuous amortization via the stopping function technique and achieve the main result of this paper, the $O(d(L + \ln d))$ bound on the size of the subdivision tree for the `SqFreeEVAL` algorithm. Finally, we conclude in Section 5.

The authors would like to thank the following people for many useful discussions: Benjamin Galehouse, Michael Sagraloff, and Chee Yap.

2. The `SqFreeEVAL` algorithm

Given an interval $I = [a, b]$ with integer endpoints and a polynomial f with integer coefficients, i.e., $f \in \mathbb{Z}[X]$, the `SqFreeEVAL` algorithm returns a collection of intervals which cover and isolate the real roots of f in (a, b), i.e., every root appears in an output interval and each output interval contains exactly one root (ignoring multiplicities). In the `SqFreeEVAL` algorithm, if the interval $[c, d]$ is output, then (c, d) contains exactly one root of f and if $[c, c]$ is output, then c is a root of f. The `SqFreeEVAL` algorithm maintains a (finite) partition P of the interval I, i.e., a finite collection of intervals whose interiors are disjoint and whose union is I. The `SqFreeEVAL` algorithm iteratively bisects the elements of P until the intervals of the partition P are each small enough to pass the `SqFreeEVAL` termination conditions (see Section 2.1). Of interest to us is the size $\#P$ of the partition, i.e., the number of intervals in P.

We begin with some terminology: For an interval $J = [c, d]$ the *width* of J is $w(J) = d - c$ and the *midpoint* of J is $m(J) = (c + d)/2$. Also, to *bisect* an element of the partition P means to replace the interval $J = [c, d] \in P$ by the two subintervals $[c, m(J)]$ and $[m(J), d]$. Note that each bisection increases the number of intervals in the partition by exactly one; this implies that $\#P$ is one more than the number of bisections done by the `SqFreeEVAL` algorithm. The subdivision tree is a full binary tree, and, therefore, has $2\#P - 1$ nodes. In fact, the internal nodes of the subdivision tree are in bijective correspondence with the splits in the final partition. All of the calculations done by the `SqFreeEVAL` algorithm will be performed on the dyadic integers $\mathbb{Z}[1/2]$ so that all of the standard operations are exact. This prevents well-known implementation errors from arising in practice.

2.1. Statement of the `SqFreeEVAL` algorithm

In the `SqFreeEVAL` algorithm, we first replace f by its square free component, which we briefly call g. Then, we replace f' by its square free and relatively prime to f component, i.e., we first take the square free component of f' and then take the portion of this polynomial which is relatively prime to f. We briefly call this h. Note that $g|f$ and $h|f'$, and, moreover, the roots of g are separated by roots of h by Rolle's theorem. In the case where f is square free, the zeros of h partition f into monotonic regions; in the case where f is not square free, the zeros of h no longer have this property, but they still partition the roots of f (and hence the roots of g). Throughout the remainder of this paper, except for a brief note in Section 4.2, we use these square free substitutions for f and f' without mention. The bounds on the subdivision tree, however, will be in terms of the data for original the f and not for any replacements.

The `SqFreeEVAL` algorithm creates a partition of I and determines which intervals in the partition contain roots. Initially, the partition of I is $P = \{I\}$, the trivial partition.

Algorithm 2.1: The `SqFreeEVAL` algorithm

Repeatedly subdivide each $J \in P$ until one of the following conditions holds:

$$(\mathrm{C}_0) \qquad |f(m(J))| > \sum_{i=1}^{d} \frac{|f^{(i)}(m(J))|}{i!} \left(\frac{w(J)}{2}\right)^i \quad \textit{or}$$

$$(\mathrm{C}_1) \qquad |f'(m(J))| > \sum_{i=1}^{d-1} \frac{|f^{(i+1)}(m(J))|}{i!} \left(\frac{w(J)}{2}\right)^i$$

If, when subdividing, $f(m(J)) = 0$, then output $[m(J), m(J)]$.

For each interval $J = [c, d] \in P$ where C_1 holds and $f(c) \cdot f(d) < 0$, output J

The termination proof for the `SqFreeEVAL` algorithm is very similar to the corresponding statement in (Burr et al., 2009; Sagraloff and Yap, 2009; Yap and Sagraloff, 2011). The correctness proof is slightly different from the corresponding proofs for other `EVAL`-type algorithms. The correctness follows from the Taylor polynomial centered at $m(J)$: if one of the conditions holds, then it follows that f (for condition C_0) or f' (for condition C_1) is never zero in J since the inequalities are equivalent to a reverse triangle inequality on the Taylor polynomial. The first condition implies that f has no zeros in J. The second condition implies that f has at most one zero in J since roots of f' separate zeros of f (even though f might not be monotonic due to the replacements above).

2.2. `SqFreeEVAL` terminal intervals

In this section, we provide a sufficient condition for the `SqFreeEVAL` algorithm to terminate without subdividing on a given interval, i.e., for the interval to be `SqFreeEVAL` terminal.

Definition 2.1. For any polynomial g of degree d, define $\alpha_g = \{\alpha_1, \cdots, \alpha_d\}$ to be the multiset of the roots of g. In addition, define the function Σ_g to be the sum of the reciprocals of the distances from its argument to the roots of g:

$$\Sigma_g(x) = \sum_{\alpha \in \alpha_g} \frac{1}{|x - \alpha|}.$$

Note that this function can be represented in a simple form using the harmonic mean HM. Then, one has

$$\frac{1}{\Sigma_g(x)} = \frac{\mathrm{HM}(|x-\alpha_g|)}{d}$$

where $|x-\alpha_g|$ is the set of distances from x to the roots of g.

Σ_f and $\Sigma_{f'}$ will be our main objects of study. We begin with the following lemma which connects $\Sigma_f(x)$ and $\Sigma_{f'}(x)$ with conditions C_0 and C_1, respectively:

Lemma 2.1. The following inequality holds for $i \geq 0$:

$$\left|\frac{f^{(n)}(x)}{f(x)}\right| \leq (\Sigma_f(x))^n .$$

The proof is a straight-forward computation. See the proof of (Burr et al., 2009, Lemma 6.2) or (Sagraloff and Yap, 2009, Section 5.2) for details.

We use this lemma to show that a simple upper bound on the width of an interval will ensure that the conditions in the `SqFreeEVAL` algorithm hold. For example, in condition C_0, divide both sides of the inequality by $|f(m(J))|$ and apply Lemma 2.1 to derive the following inequality:

$$\sum_{i=1}^{d} \frac{|f^{(i)}(m(J))|}{i!|f(m(J))|}\left(\frac{w(J)}{2}\right)^i \leq \sum_{i=1}^{d}\frac{1}{i!}\left(\frac{\Sigma_f(m(J))w(J)}{2}\right)^i .$$

If $w(J) \leq \frac{1}{\Sigma_f(m(J))}$, then the sum on the RHS is bounded above by a geometric series with $r = 1/2$, and, therefore, the sum is bounded by 1. This implies that condition C_0 holds. We explicitly state the conclusion of this argument in the following lemma:

Lemma 2.2. If $w(J) \leq \frac{1}{\Sigma_f(m(J))}$ then C_0 holds on J and J is `SqFreeEVAL` terminal. Similarly, if $w(J) \leq \frac{1}{\Sigma_{f'}(m(J))}$, then C_1 holds on J and J is `SqFreeEVAL` terminal.

3. Stopping functions

In this section, we show how stopping functions can be used to compute the size of the subdivision tree of the `SqFreeEVAL` algorithm. The construction in Section 3.1 was originally presented in (Burr et al., 2009), but we include it here for completeness and because the construction in Section 3.2 requires a detailed understanding of the method.

3.1. Basic properties

The use of stopping functions promises to be an important tool for bounding the complexity of subdivision algorithms. Most of the numerical algorithms appearing in the introduction may benefit from this type of analysis; more algorithms of this type are mentioned in the Conclusion, Section 5. We begin by formulating an abstract algorithm called the `Bisection` algorithm, which is intended to be the prototype of these types of algorithms in one dimension. The notion of stopping functions and the `Bisection` algorithm both easily generalize to higher dimensions.

Fix a predicate B (i.e., a Boolean function) on intervals with the following property: if $K \subseteq J$ and $B(J)$ is true, then $B(K)$ is also true. The `Bisection` algorithm is the following algorithm: given an interval I, the algorithm maintains a partition P of I. Initially, let the partition be the trivial partition $P = \{I\}$ and let $P_{\texttt{Bisection}}(I)$ be the final partition.

Algorithm 3.1: The `Bisection` algorithm

```
Repeatedly subdivide each $J \in P$ until the following condition holds:
        $B(J)$ is true
```

A *stopping function* for the `Bisection` algorithm with predicate B is a real-valued function F with the following property: if, for a given interval J, there exists a point $p \in J$ such that $w(J) \leq F(p)$, then $B(J)$ is true. The following theorem, which also appears as (Burr et al., 2009, Theorem 3.5), bounds the number of subdivisions performed by the `Bisection` algorithm.

Theorem 3.1. (Burr et al., 2009, Theorem 3.5) Let F be a stopping function for the `Bisection` algorithm, then

$$\#P_{\texttt{Bisection}}(I) \leq \max\left\{1, \int_I \frac{2dx}{F(x)}\right\}.$$

If the `Bisection` algorithm does not terminate, then the integral is infinite.

Proof. If $\#P_{\texttt{Bisection}} = 1$, then the bound is immediate. If $\#P_{\texttt{Bisection}} > 1$, then an examination of the `Bisection` algorithm shows that for $J \in P_{\texttt{Bisection}}$ there is a lower bound on $w(J)$ since the `Bisection` did not terminate at the parent of J:

$$\forall c \in J, w(J) \geq \frac{1}{2}F(c).$$

In addition, $\int_I \frac{2dx}{F(x)} = \sum_{J \in P_{\texttt{Bisection}}} \int_J \frac{2dx}{F(x)}$, and it, therefore, suffices to show that for every $J \in P_{\texttt{Bisection}}$, $\int_J \frac{2dx}{F(x)} \geq 1$. Let $y \in J$ be such that $F(y)$ is maximal in J. Then

$$\int_J \frac{2dx}{F(x)} \geq \int_J \frac{2dx}{F(y)} = \frac{2}{F(y)}w(J) \geq \frac{2}{F(y)} \cdot \frac{F(y)}{2} = 1.$$

In the case when the `Bisection` algorithm does not terminate, we can look at the partition P at any moment in time. The above argument shows that $\#P$ is still bounded by the integral $\int_I 2dx/F(x)$. Since $\#P$ can be chosen to be arbitrarily large, this shows that the integral is unbounded. $\square$

3.2. A stopping function for `SqFreeEVAL`

The next goal is to transform the inequality $w(J) \leq \frac{1}{\Sigma_f(m(J))}$ into a stopping function. Currently, it is not a stopping function because the function on the RHS is not for an arbitrary point of J, but for a specific point, the midpoint. We begin to turn this into a stopping function via the following lemma:

Lemma 3.1. Let $\overline{z} = (z_1, \cdots, z_d)$ with $z_i > 0$ and $y \in \mathbb{R}$ such that $y > 0$ and $z_i > y$ for all i. Then

$$\mathrm{HM}(\overline{z} - y) \geq \mathrm{HM}(\overline{z}) - d \cdot y \tag{1}$$

$$\mathrm{HM}(\overline{z}) \leq d \cdot z_i \qquad \forall i. \tag{2}$$

Proof. For Inequality (1), we expand each of the harmonic means and get the following equivalent inequality:

$$\frac{d}{\sum \frac{1}{z_i - y}} \geq \frac{d}{\sum \frac{1}{z_i}} - d \cdot y.$$

Noting that all of the denominators are positive, clearing fractions gives that this inequality is equivalent to the following inequality:

$$y\left(\sum \frac{1}{z_i - y}\right)\left(\sum \frac{1}{z_i}\right) \geq \sum \frac{1}{z_i - y} - \sum \frac{1}{z_i}.$$

This inequality is easily justified by combining similar terms on the RHS to obtain a sum with general term

$$\frac{1}{z_i - y} - \frac{1}{z_i} = \frac{y}{z_i(z_i - y)},$$

which is a term that appears on the LHS. Since the remaining terms on the LHS are positive, this proves the first inequality.

For Inequality (2), we expand the harmonic mean to get the following equivalent inequality:

$$\frac{d}{\sum \frac{1}{z_j}} \leq d \cdot z_i.$$

Once again, the denominator is positive, so by clearing fractions we have that this inequality is equivalent to the following inequality:

$$\frac{1}{z_i} \leq \sum \frac{1}{z_j}.$$

Since all of the terms on the RHS are positive and include the term on the LHS, this proves the second inequality. $\square$

Let $G_0(x) = \frac{2}{3\Sigma_f(x)}$, then G_0 is a stopping function for EVAL: Let J be an interval such that J contains x and let m be the midpoint of J; then, $|x-m| \leq \frac{w(J)}{2}$. Assume now that $w(J) \leq \frac{2}{3\Sigma_f(x)}$. Then, inequality (2) in Lemma 3.1 implies that $|x-\alpha| \geq \frac{1}{\Sigma_f(x)} > \frac{w(J)}{2}$ for all $\alpha \in \alpha_f$. This setup implies the following inequalities:

$$w(J) \leq \frac{1}{\Sigma_f(x)} - \frac{w(J)}{2} \leq \frac{\mathrm{HM}(|x-\alpha_f| - \frac{w(J)}{2})}{d} \leq \frac{\mathrm{HM}(|x-\alpha_f| - |x-m|)}{d} \leq \frac{1}{\Sigma_f(m)}.$$

The second inequality follows from Lemma 3.1 and the fact that the terms of the harmonic mean $\mathrm{HM}(|x-\alpha_f| - |x-m|)$ are all positive (because of the bound on $w(J)$ above). The remaining inequalities follow from the monotonicity of the harmonic mean. The last inequality also uses that $|x-\alpha_j| - |x-m| \leq |m-\alpha_j|$ by the triangle inequality. When combined with Lemma 2.2, this implies that J is `SqFreeEVAL` terminal. Similarly, let $G_1(x) = \frac{2}{3\Sigma_{f'}(x)}$ where $\Sigma_{f'}$ is the corresponding function for f', then G_1 is also a stopping function for the `SqFreeEVAL` algorithm. Finally, let $G(x) = \max\{G_0(x), G_1(x)\}$, then G is an everywhere positive stopping function for the `SqFreeEVAL` algorithm.

4. Size of the subdivision tree of the `SqFreeEVAL` algorithm for the benchmark problem

In this section, we prove that the size of the subdivision tree of the `SqFreeEVAL` algorithm is $O(d(L+\ln d))$ where L is the number of bits needed to write the coefficients of f (hence the absolute value of the coefficients is strictly less than 2^L). In this case, the absolute value of all the roots is bounded by 2^L (Yap, 2000) (this bound comes from the original f, not from the square free substitution). Hence, we can assume wlog that $b = -a = 2^L$. By Theorem 3.1, the number of intervals in the final partition of I from the `SqFreeEVAL` algorithm is bounded by $\int_I \frac{2}{G(x)}dx$. The crossover points of G are difficult to determine, however, so we replace this integral by a slightly larger one which is easier to evaluate: For any $x \in I$, let R_x be the set of roots in $\alpha_{ff'}$ which are closest to x. Similarly, for $\alpha \in \alpha_{ff'}$, let I_α be the set of $x \in I$ such that no other root in $\alpha_{ff'}$ is closer to x than α. Note that $x \in I_\alpha$ iff $\alpha \in R_x$ and that two of the I_α's are either disjoint (except for endpoints) or coincide (in the case of complex conjugates). Therefore, these I_α's determine a partition of I. Also, let S be the set of endpoints of the I_α's; then, for all points $x \in I \setminus S$, one has $R_x \subseteq \alpha_f$ or $R_x \subseteq \alpha_{f'}$ because f and f' do not share roots. We define another function $F(x)$:

$$F(x) = \begin{cases} G_1(x) & x \notin S \text{ and } R_x \subseteq \alpha_f \\ G_0(x) & x \notin S \text{ and } R_x \subseteq \alpha_{f'} \\ G(x) & x \in S \end{cases}.$$

For some intuition in this definition, note that when $R_x \subseteq \alpha_{f'}$, x is far from roots of f. Hence, G_0 (which is related to C_0 by the discussion in Section 3.2) should be applied in this case. A corresponding statement holds for G_1. Note that although S might not correspond to the crossover points of G, pointwise, $F(x) \leq G(x)$ since G is a maximum of the terms which can occur in F. This inequality implies the following inequalities:

$$\int_I \frac{2}{G(x)} dx \leq \int_I \frac{2}{F(x)} dx \leq \int_I \sum_{\alpha \in \alpha_{ff'} \setminus R_x} \frac{3dx}{|x-\alpha|} = \sum_{\alpha \in \alpha_{ff'}} \int_{I \setminus I_\alpha} \frac{3dx}{|x-\alpha|}. \qquad (3)$$

For the second inequality let $x \notin S$, then x is either closest to a root of f or a root of f'. If x is closest to a root of f, then $R_x \subseteq \alpha_f$ and $\frac{2}{F(x)} = 3\Sigma_{f'}(x)$. In this case, the sum to the right of the inequality includes all of the roots in $\alpha_{f'}$ as well as some roots in α_f. Thus, at least all of the terms of $\Sigma_{f'}(x)$ appear on the RHS of the inequality. The case where x is closest to a root of f' is similar. This implies the inequality because the set of points for which this inequality may fail is a measure zero subset of S.

4.1. Evaluating the integrals

Consider the shape of each of the regions where we integrate: since all of the integrals are of the form $\int_r^s \frac{3dx}{|x-\alpha|}$, we evaluate a general integral of this form where r and s lie on the same side of $\mathrm{Re}(\alpha)$.

- In the case where α is real:

  if $s > r > \alpha$
  $$\int_r^s \frac{3dx}{|x-\alpha|} = \int_r^s \frac{3dx}{x-\alpha} = 3\ln(|s-\alpha|) - 3\ln(|r-\alpha|)$$

  if $r < s < \alpha$
  $$\int_r^s \frac{3dx}{|x-\alpha|} = \int_r^s \frac{3dx}{\alpha - x} = 3\ln(|r-\alpha|) - 3\ln(|s-\alpha|)$$

  These logarithms will be bounded in the next section.

- In the case where α is not real:
  $$\begin{aligned}\int_r^s \frac{3}{|x-\alpha|} dx &= \int_r^s \frac{3}{\sqrt{(x-\mathrm{Re}(\alpha))^2 + \mathrm{Im}(\alpha)^2}} dx \\ &= \int_{(r-\mathrm{Re}(\alpha))/|\mathrm{Im}(\alpha)|}^{(s-\mathrm{Re}(\alpha))/|\mathrm{Im}(\alpha)|} \frac{3}{\sqrt{y^2+1}} dy \\ &= 3\,\mathrm{arcsinh}\left(\frac{s-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|}\right) - 3\,\mathrm{arcsinh}\left(\frac{r-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|}\right)\end{aligned}$$

  This is now bounded via the relationship between $\mathrm{Re}(\alpha)$ and r, s. If $s > r > \mathrm{Re}(\alpha)$, then:

  $$\begin{aligned}&3\,\mathrm{arcsinh}\left(\frac{s-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|}\right) - 3\,\mathrm{arcsinh}\left(\frac{r-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|}\right) \\ &\quad = 3\ln\left(\frac{s-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|} + \sqrt{\left(\frac{s-\mathrm{Re}(\alpha)}{|\mathrm{Im}(\alpha)|}\right)^2 + 1}\right)\end{aligned}$$

$$
\begin{aligned}
&\qquad - 3\ln\left(\frac{r-\mathrm{Re}(\alpha)}{|\,\mathrm{Im}(\alpha)|} + \sqrt{\left(\frac{r-\mathrm{Re}(\alpha)}{|\,\mathrm{Im}(\alpha)|}\right)^2+1}\right)\\
&= 3\ln(s-\mathrm{Re}(\alpha)+\sqrt{(s-\mathrm{Re}(\alpha))^2+\mathrm{Im}(\alpha)^2})\\
&\qquad - 3\ln(r-\mathrm{Re}(\alpha)+\sqrt{(r-\mathrm{Re}(\alpha))^2+\mathrm{Im}(\alpha)^2})\\
&\le 3\ln(2|s-\alpha|) - 3\ln(|r-\alpha|).
\end{aligned}
$$

If $r < s < \mathrm{Re}(\alpha)$, then the computation is similar, and the integral is bounded above by $3\ln(2|r-\alpha|) - 3\ln(|s-\alpha|)$. These logarithms will also be bounded in the next section.

4.2. Finishing the bound on the `SqFreeEVAL` *algorithm*

In this section, we use the computation from the previous section to prove the main result of this paper. To do this, we consider the roots $\alpha \in \alpha_{ff'}$ with two different cases depending on if α is real or not.

- If α is real; then $\alpha \in I_\alpha$ and let $I_\alpha = [c, d]$. Then, the term corresponding to α in the RHS of Inequality (3) consists of $\int_{I\setminus I_\alpha} \frac{3dx}{|x-\alpha|} = \int_{-2^L}^{c} \frac{3dx}{|x-\alpha|} + \int_{d}^{2^L} \frac{3dx}{|x-\alpha|}$. Note that integrals may be zero which happens when $c = -2^L$ or $d = 2^L$. Then, using the bounds derived in the preceding section on these integrals, it follows that they are bounded by:

$$3\ln(|-2^L-\alpha|) - 3\ln(|c-\alpha|) + 3\ln(|2^L-\alpha|) - 3\ln(|d-\alpha|).$$

  The positive terms are bounded by $O(L)$ (the leading term is $6\ln(2)L$) and for the negative terms, note that c and d are points which are equidistant from α and another root of ff', e.g., c is equidistant from α and $\beta \in \alpha_{ff'}$ where I_β is the interval immediately to the left of I_α. Then, $\ln(|c-\alpha|)$ is bounded below by the logarithm of half the distance from α to β.

- If α is not real, then the term corresponding α in the RHS of Inequality (3) consists of $\int_{I\setminus I_\alpha} \frac{3dx}{|x-\alpha|}$ which is bounded above by $\int_I \frac{3dx}{|x-\alpha|}$. By splitting this integral at $\mathrm{Re}(\alpha)$, the integral is equal to $\int_{-2^L}^{\mathrm{Re}(\alpha)} \frac{3dx}{|x-\alpha|} + \int_{\mathrm{Re}(\alpha)}^{2^L} \frac{3dx}{|x-\alpha|}$. Using the bounds derived in the preceding section on these integrals, it follows that these integrals are bounded by:

$$3\ln(2|-2^L-\alpha|) - 3\ln(|\,\mathrm{Re}(\alpha)-\alpha|) + 3\ln(2|2^L-\alpha|) - 3\ln(|\,\mathrm{Re}(\alpha)-\alpha|).$$

  The positive terms are bounded by $O(L)$ (the leading term is $6\ln(2)L$) and for the negative terms, note that $|\,\mathrm{Re}(\alpha)-\alpha| = |\,\mathrm{Im}(\alpha)|$, which is the logarithm of half the distance between α and $\overline{\alpha}$.

Combining all of the $O(L)$'s which appear in the integrals results in a bound of $O(dL)$ (the leading term is $6(\ln 2(2d-1))L$). The sum of the logarithmic distances between roots are bounded simultaneously via the standard Mahler-Davenport lower bound on distances between roots, see (Davenport, 1985; Mignotte, 1995; Du et al., 2007; Eigenwillig et al., 2006). To do this, we construct a directed graph whose nodes are the roots in $\alpha_{ff'}$ and whose edges represent the logarithms which must be calculated. In this graph, the edges satisfy the conditions of the Mahler-Davenport bound and are chosen so that the in-degree of any node is at most 2. For each pair of complex roots, $(\alpha, \overline{\alpha})$, connect them with two directed edges, one from α to $\overline{\alpha}$, the other in the opposite direction. On the other hand, if α is real, then let β be a root where I_β lies immediately to the right of I_α and γ be a root where I_γ lies immediately to the left of I_α (provided I_α is not the rightmost or leftmost interval in the partition, respectively). Those of β or γ which are real are connected to α so that the arrow points in the direction of decreasing absolute value. If β is not real, then connect α to either β or $\overline{\beta}$, whichever has *positive* imaginary component. On the other hand, if γ is not real, then connect α to either γ or $\overline{\gamma}$, whichever has *negative* imaginary component. Again, these edges are directed so that the arrow points in the direction of decreasing absolute value. By inspection, we find that the maximum in-degree of this directed graph is 2. The Mahler-Davenport bound can then be applied twice to find the result. The bound implies that the sum of the negative logarithmic distances between the roots appearing in this construction is bounded above by:

$$12 \cdot \ln\left(\frac{1}{\sqrt{|\operatorname{Disc}(ff')|}} M(ff')^{2d-2} \left(\frac{2d-1}{\sqrt{3}}\right)^{2d-1} (2d-1)^d\right).$$

The discriminant will be an integer and therefore the discriminant term is bounded above by 1. The Mahler measure of ff' is bounded in terms of the 2-norms of the coefficients of the original f and f': $M(ff') = M(f)M(f') \leq \|f\|_2\|f'\|_2 \leq (2^L\sqrt{d+1})(d2^L\sqrt{d})$. Therefore, this portion is bounded by $O(dL + d\ln d)$ (the leading term is bounded $24\ln(2)dL + 42d\ln d$)). Thus, the complexity of the `SqFreeEVAL` algorithm is $O(d(L + \ln d))$ (the leading term is bounded $36\ln(2)dL + 42d\ln d \leq 25dL + 42d\ln d$).

If f or f' was replaced by a square free version, we used the original f and f' because the square free versions of f and f' divide the original functions, and, therefore, the Mahler measure of the product of the square free versions is bounded above by $M(ff')$. In fact, the 2-norms of the coefficients of the original functions are often smaller than the 2-norms of the square free versions: naïvely using the divisibility relationship, the coefficients of the square-free versions may be larger by a factor of 2^{cd}, where c is a constant, see (Yap, 2000).

5. Conclusion

In this paper, we provided a complexity analysis of the `SqFreeEVAL` algorithm and showed it to be optimal under the weak assumption that $L \geq \ln d$. To accomplish this, we used the novel technique of continuous amortization through stopping functions. The simplicity of this argument exhibits the utility of this technique: the proof of the next closest complexity bound for an `EVAL`-type algorithm in (Sagraloff and Yap, 2009; Yap and Sagraloff, 2011) is significantly more complex.

The `SqFreeEVAL` algorithm is very easy to implement (Moore, 1966; Mitchell, 1990; Plantinga and Vegter, 2004; Plantinga, 2006; Kamath, 2010) and it now joins the Sturm and Descartes methods by having a subdivision tree which grows at the rate $O(d(L + \ln d))$. It, therefore, may become more prevalent in practical situations because it has several desirable properties. This also answers a question raised in (Henrici, 1970) concerning the good behavior of this technique.

In addition, the continuous amortization technique can be used to bound the number of subdivisions over any interval, and, therefore, may find many more applications for different types of questions about subdivision algorithms. For example, in many practical applications, the question is to find the roots in a given domain, not just for the benchmark domain; continuous amortization may provide a comparison of different algorithms in these situations.

We close with some continuing research and questions:

- The algorithm for finding complex roots appearing in (Sagraloff and Yap, 2009; Yap and Sagraloff, 2011) is very similar to the `SqFreeEVAL` algorithm. We are currently preparing a simplification of their work using the results from this paper.

- There are many bisection algorithms where continuous amortization may be useful, see, for example, (Henrici, 1970; Yakoubsohn, 2005; Sagraloff and Yap, 2009; Yap and Sagraloff, 2011; Plantinga and Vegter, 2004; Plantinga, 2006; Snyder, 1992; Galehouse, 2009; Burr et al., 2010; Eigenwillig et al., 2006; Sagraloff, 2011; Du et al., 2007; Lin and Yap, 2009). We plan on extending our techniques to these cases. In particular, stopping functions which are appropriate for the two dimensional cases treated in (Plantinga and Vegter, 2004; Plantinga, 2006; Galehouse, 2009) would be very useful because current techniques have not been fruitful in establishing complexity bounds of these algorithms.

- If f' was not square free, then the test for condition (C_1) in Algorithm 2.1 is based on the square free part of f', not the original function. The `SqFreeEVAL` algorithm, however, will continue to terminate and be correct even when this substitution does not occur, i.e., when the original f' is used. For this reason, it is likely that the above substitution is extraneous. For example, in the simplest cases where f' is not square free and the integral in Inequality (3) can be calculated by hand, the result is $O(d(L + \ln d))$.

I. Boier-Martin, D. Zorin, and F. Bernardini. A survey of subdivision-based tools for surface modeling. In Ravi Janardan, Michiel Smid, and Debasish Dutta, editors, *Geometric and Algorithmic Aspects of Computer-Aided Design and Manufacturing*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2005.

M. Burr, S.W. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*, 2010. To appear, Special Issue for ISSAC 2008.

Michael Burr, Felix Krahmer, and Chee Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. Technical Report TR09-136, Electronic Colloquium on Computational Complexity (ECCC), December 2009.

Michael Burr, Vikram Sharma, and Chee Yap. Evaluation-based root isolation, 2011. In preparation.

George E. Collins and Alkiviadis G. Akritas. Polynomial real root isolation using Descartes' rule of signs. In R. D. Jenks, editor, *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation*, pages 272–275. ACM Press, 1976.

George E. Collins, Jeremy R. Johnson, and Werner Krandick. Interval arithmetic in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 34:145–157, 2002.

James H. Davenport. Computer algebra for cylindrical algebraic decomposition. Tech. Rep., The Royal Inst. of Technology, Dept. of Numerical Analysis and Computing Science, S-100 44, Stockholm, Sweden, 1985. Reprinted as Tech. Report 88-10 , School of Mathematical Sci., U. of Bath, Claverton Down, Bath BA2 7AY, England. URL http://www.bath.ac.uk/ masjhd/TRITA.pdf.

Jean-Pierre Dedieu and Jean-Claude Yakoubsohn. Localization of an algebraic hypersurface by the exclusion algorithm. *Applicable Algebra in Engineering, Communication and Computing*, 2(4):239–256, 1992.

Zilin Du, Vikram Sharma, and Chee Yap. Amortized bounds for root isolation via Sturm sequences. In Dongming Wang and Lihong Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pages 113–130. Birkhäuser Verlag AG, Basel, 2007. Proc. Int'l Workshop on Symbolic-Numeric Computation, Xi'an, China, Jul 19–21, 2005.

Arno Eigenwillig, Vikram Sharma, and Chee Yap. Almost tight complexity bounds for the Descartes method. In *Proc. Int'l Symp. Symbolic and Algebraic Computation (ISSAC'06)*, pages 71–78, 2006. Genova, Italy. Jul 9-12, 2006.

Ioannis Z. Emiris, Bernard Mourrain, and Elias P. Tsigaridas. Real algebraic numbers: Complexity analysis and experimentation. In *Reliable Implementation of Real Number Algorithms*, pages 57–82, 2008.

Benjamin Galehouse. *Topologically Accurate Meshing Using Spatial Subdivision Techniques*. Ph.D. thesis, New York University, Department of Mathematics, Courant Institute, May 2009. From `http://cs.nyu.edu/exact/doc/`.

Michael Hemmer, Elias P. Tsigaridas, Zafeirakis Zafeirakopoulos, Ioannis Z. Emiris, Menelaos I. Karavelas, and Bernard Mourrain. Experimental evaluation and cross-benchmarking of univariate real solvers. In *Proceedings of the 2009 Conference on Symbolic Numeric Computation*, pages 45–54, 2009.

Peter Henrici. Methods of search for solving polynomial equations. *Journal of the Association for Computing Machinery*, 17(2):273–283, April 1970.

International Conference on Domain Decomposition Methods, 1987–2011.

Jeremy R. Johnson. *Algorithms for polynomial real root isolation.* Ph.D. thesis, The Ohio State University, 1991.

J.R. Johnson. Algorithms for polynomial real root isolation. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and monographs in Symbolic Computation, pages 269–299. Springer, 1998.

Narayan Kamath. Subdivision algorithms for complex root isolation: Empirical comparisons. Master's thesis, Oxford University, August 2010.

Werner Krandick and Kurt Mehlhorn. New bounds for the Descartes method. *J. Symbolic Computation*, 41(1):49–66, 2006.

Thomas Lickteig and Marie-Françoise Roy. Sylvester-Habicht sequences and fast Cauchy index computation. *Journal of Symbolic Computation*, 31:315–341, 2001.

Long Lin and Chee Yap. Adaptive isotopic approximation of nonsingular curves: the parametrizability and non-local isotopy approach. In *Proceedings of the 25th annual Symposium on Computational Geometry*, pages 351–360, June 2009.

W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

Maurice Mignotte. On the distance between the roots of a polynomial. *Applicable Algebra in Engineering, Communication and Computing*, 6(6):327–332, 1995.

Don P. Mitchell. Robust ray intersection with interval arithmetic. In *Graphics Interface'90*, pages 68–74, 1990.

Ramon E. Moore. *Interval Analysis.* Prentice Hall, Englewood Cliffs, NJ, 1966.

Bernard Mourrain, Fabrice Rouillier, and Marie-Françoise Roy. The Bernstein basis and real root isolation. In Jacob E. Goodman, János Pach, and Emo Welzl, editors, *Combinatorial and Computational Geometry*, number 52 in MSRI Publications, pages 459–478. Cambridge University Press, 2005.

V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers & Mathematics with Applications*, 31(12):97–138, 1996.

Victor Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Review*, 39(2):187–220, 1997.

Simon Plantinga. *Certified Algorithms for Implicit Surfaces.* Ph.D. thesis, Groningen University, Institute for Mathematics and Computing Science, Groningen , Netherlands, December 2006.

Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. Eurographics Symposium on Geometry Processing*, pages 245–254, New York, 2004. ACM Press.

Helmut Ratschek and Jon Rokne. *Computer Methods for the Range of Functions.* Horwood Publishing Limited, Chichester, West Sussex, UK, 1984.

Daniel Reischert. Asymptotically fast computation of subresultants. In *ISSAC 97*, pages 233–240, 1997. Maui, Hawaii.

Fabrice Rouillier and Paul Zimmermann. Efficient isolation of [a] polynomial's real roots. *J. Computational and Applied Mathematics*, 162:33–50, 2004.

Michael Sagraloff. On the complexity of real root isolation. Technical Report arxiv:1011.0344, Mathematics Arxiv, 2011.

Michael Sagraloff and Chee K. Yap. An efficient exact subdivision algorithm for isolating complex roots of a polynomial and its complexity analysis, July 2009. Submitted.

Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity, 1982. URL `www.informatik.uni-bonn.de/ schoe/fdthmrep.ps.gz`. Manuscript , Department of Mathematics, University of Tübingen. Updated 2004.

Vikram Sharma. Complexity of real root isolation using continued fractions. *Theoretical Computer Science*, 409(2), 2008.

J. M. Snyder. Interval analysis for computer graphics. *SIGGRAPH Comput.Graphics*, 26(2):121–130, 1992.

Elias P. Tsigaridas and Ioannis Z. Emiris. On the complexity of real root isolation using continued fractions. *Theoretical Computer Science*, 392(1-3): 158–173, 2008.

Jean-Claude Yakoubsohn. Numerical analysis of a bisection-exclusion method to find zeros of univeriate analytic functions. *Journal of Complexity*, 21(5): 652–690, 2005.

Chee Yap and Michael Sagraloff. A simple but exact and efficient algorithm for complex root isolation. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation (ISSAC 2011)*, pages 353–360, 2011. San Jose, CA, June 8-11, 2011.

Chee K. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, 2000.