(Extended Abstract)
A New Constructive Root Bound for Algebraic Expressions

Chen Li Chee Yap*
Courant Institute of Mathematical Sciences
New York University, New York, NY 10012, USA.
Email: {chenli, yap}@cs.nyu.edu

October 19, 2000

Abstract

Computing effective root bounds for *constant* algebraic expressions is a critical problem in the *Exact Geometric Computation* approach to robust geometric programs. Classical root bounds are often non-constructive. Recently, various authors have proposed bounding methods which might be called *constructive root bounds*. For the important class of *radical expressions*, Burnikel et al (BFMS) have provided a constructive root bound which, in the division-free case, is an improvement over previously known bounds and is essentially tight. In the presence of division, their bound requires a quadratic blowup in *root bit-bound* compared to the division-free case. We present a new constructive root bound that avoids this quadratic blowup and which is applicable to a more general class of algebraic expressions. This leads to dramatically better performance in some computations. We also give an improved version of the degree-measure bound from Mignotte and BFMS. We describe our implementation in the context of the Core Library, and report on some experimental results.

1 Introduction

There is considerable recent interest in robust implementation of geometric algorithms [17, 14]. Exact Geometric Computation (EGC) [19] is one general approach to achieve geometric robustness. This is the approach in, for instance, the LEDA [3, 8] and CGAL [6] libraries. A fundamental task in EGC is to determine the exact sign of a constant algebraic expression E. For example, the following expression arises in the implementation of Fortune's sweepline algorithm [5] for the Voronoi diagram of a planar point set:

$$E = \frac{a+\sqrt{b}}{d} - \frac{a'+\sqrt{b'}}{d'}, \tag{1.1}$$

where a, a', b, b', d, d' are integer constants. Typically, the sign determination task reduces to first finding some root bound b for E. We call $b > 0$ a *root bound* for E if the following holds: if $E \neq 0$ then $|E| \geq b$. As our choice of terminology suggests, we are mainly interested in bounding roots away from 0. With such a root bound b, we can determine the sign of E by computing a numerical approximation $\widetilde{E}$ such that $|E - \widetilde{E}| < \frac{b}{2}$. The sign of E is the same as that of $\widetilde{E}$ if $|\widetilde{E}| \geq \frac{b}{2}$, otherwise we can conclude that $E = 0$. In practice [19], the precision required in approximation can be progressively increased until one of two events occurs: either (i) the approximation $\widetilde{E}$ satisfies

$$|\widetilde{E}| > |E - \widetilde{E}|, \tag{1.2}$$

or (ii) the approximation satisfies

$$|E - \widetilde{E}| < \frac{b}{2}. \tag{1.3}$$

Note that if $|E|$ is large, then condition (i) will be reached first, and the root bound does not play a role in the actual complexity of the sign determination. However, if E is zero (as happens in, say, degenerate cases or theorem proving application [15]), then the root bound plays a critical role. The Real/Expr Package [19] is the first

*This work is supported in part by NSF Grant #CCR-9619846.

system to systematically exploit such root bounds. More recently, the LEDA Library [1] and the Core Library [9, 7] have made such sign determination techniques easily accessible to any programmer.

The problem of root bounds and more generally, root location, is a very classical one with an extensive literature (e.g., [10] or [12, chap. 2]). Some classical bounds are highly non-constructive. If a bound for a root α of a polynomial $P(x)$ is given in terms of some simple function of P's coefficients and degree, it can be considered constructive. For instance, Landau's bound says that any non-zero root α of $P(x)$ satisfies $|\alpha| \geq \|P(x)\|_2^{-1}$ where $P(x) = \sum_{i=0}^n a_i x^i$ and $\|P(x)\|_2 = \sqrt{\sum_{i=0}^n |a_i|^2}$. Unfortunately, in many applications, the coefficients of $P(x)$ are not explicitly given. For instance, in the LEDA and Core libraries, an algebraic number α is presented as a *radical expression* which is constructed from the integers, and recursively built-up using the four arithmetic operations $(+, -, \times, \div)$ and radical extraction $\sqrt[k]{\cdot}$ $(k \geq 2)$. Thus, the notion of "constructive" depends on the presentation of α; we call such a presentation an *expression*. If E is a presentation of α, we will write $\mathrm{val}(E) = \alpha$. But when there is no confusion, we simply write "E" in place of "$\mathrm{val}(E)$". The *constructive root bound problem* is this: given a set $\mathcal{E}$ of expressions (e.g., the radical expressions), give a set of inductive rules for computing a root bound for each expression in $\mathcal{E}$. If b is a root bound for an expression E, we will call $-\log_2 b$ a *root bit-bound* for E. In the worst case, the root bit-bound determines the complexity of our sign determination algorithms. It is important to realize that in our paper, the term "expression" roughly corresponds to a directed acyclic graph in which nodes are labeled by the appropriate constants and operations (see Section 3).

One of the best constructive root bounds for the class of radical expressions is from Burnikel et al [2] (hereafter called the "BFMS bound"). The BFMS approach is based on a well-known transformation of an expression E to eliminate all but one division, producing two associated division-free expressions $U(E)$ and $L(E)$ such that $\mathrm{val}(E) = \mathrm{val}(U(E))/\mathrm{val}(L(E))$. E.g., if $E = \frac{a}{b} + \frac{c}{d}$, then $U(E) = ad + bc$ and $L(E) = bd$. If E is division-free, then $L(E) = 1$ and $\mathrm{val}(E)$ is an algebraic integer. For an expression E having r radical nodes with indices $k_1, k_2, \ldots, k_r$, the BFMS bound is given by

(1.4)
$$\mathrm{val}(E) \neq 0 \Rightarrow |\mathrm{val}(E)| \geq (u(E)^{D(E)^2-1} l(E))^{-1},$$

where $D(E) = \prod_{i=1}^r k_i$, and $u(E)$ and $l(E)$ are (respectively) upper bounds on the absolute values of algebraic conjugates of $\mathrm{val}(U(E))$ and $\mathrm{val}(L(E))$. For division-free expressions, the BFMS bound improves to

$$\mathrm{val}(E) \neq 0 \Rightarrow |\mathrm{val}(E)| \geq (u(E)^{D(E)-1})^{-1}. \tag{1.5}$$

This was shown to be essentially sharp. Note that the root bit-bound in (1.4) is quadratic in $D(E)$, while in (1.5), it is linear in $D(E)$. Our experience is that this quadratic factor can be a serious efficiency issue. Consider a simple example: $E = (\sqrt{x} + \sqrt{y}) - \sqrt{x + y + 2\sqrt{xy}}$ where x, y are L-bit integers (i.e., $|x|, |y| < 2^L$). Of course, this expression is identically 0 for any x, y. The BFMS bound yields a root bit-bound of $7.5L + \mathcal{O}(1)$ bits. But in case, x and y are viewed as rational numbers (with denominator 1), the bit-bound becomes $127.5L + \mathcal{O}(1)$. This example shows that introducing rational numbers at the leaves of expressions has a major impact on the BFMS bound. This is an important and common situation: in practice, it is usual to have floating point numbers as input constants in an expression. Since these are special cases of rational numbers, the BFMS bound becomes quite pessimistic.

Main Results. In this paper, we address two issues raised by the BFMS bound. First, is the quadratic factor $D(E)^2$ in the root bit-bound of E essential for radical expressions? We show that it is not: $D(E)$ is sufficient. However, this requires a new approach in which we also need to maintain, among other things, upper bounds on the leading as well as tail coefficients of the minimal polynomial of the algebraic number E. The second issue is whether the BFMS technique can be extended to more general algebraic expressions. For instance, suppose we introduce a new kind of leaf into our expressions denoted by $\mathrm{Root}(P(x))$ where $P(x)$ is an integer polynomial. The framework of BFMS cannot handle this extension since there is no analogue of the $E \mapsto (U(E), L(E))$ transformation. But our new approach can handle such extensions.

For any algebraic number α, we will exploit the following relation:

$$(1.6) \qquad \alpha \neq 0 \Rightarrow |\alpha| \geq (\mu(\alpha)^{\deg(\alpha)-1}\mathrm{lead}(\alpha))^{-1},$$

where $\mu(\alpha) = \max\{|\xi| \ : \ \xi \text{ is a conjugate of } \alpha\}$, $\deg(\alpha)$ is the degree of the minimal polynomial $\mathrm{Irr}(\alpha)$ of α and $\mathrm{lead}(\alpha)$ is the leading coefficient of $\mathrm{Irr}(\alpha)$.

For radical expressions *without* divisions, our new bound turns out to be exactly same as the BFMS bound; but for those *with* divisions, our root bit-bound is only linear in $D(E)$. To see the effects of this improvement, recall the expression in Fortune's algorithm given in (1.1). Suppose the inputs to Fortune's algorithm are L-bit integers. Then it can be shown that the bit lengths for a and a' are $3L$, for b and b' are $6L$ and for d and d' are $2L$. The BFMS bit-bound for this predicate is $(79L + 30)$ bits. Our new bit-bound gives $(19L+9)$ bits. Sellen and Yap [16] have shown that $15L + O(1)$ bits are sufficient, and this is the best possible.

Improved degree-measure bound. Mignotte and Burnikel et al [2, 11, 12] presented a constructive root bound, called the *degree-measure bound*. As a side product of our new root bound, we obtain improvements to this degree-measure bound. This can lead to much improved performance, as seen in our theorem proving application [15]. We will also show below that the degree-measure bound is not comparable to our new bound and thus has independent interest.

Overview of Paper. In Section 2, we review previous work on constructive root bounds. Section 3 formalizes the constructive root bound problem. We present our new constructive root bound in Section 4, and give an improved degree-measure bound in Section 5. A comparative study of various root bounds is given in Section 6. In Section 7, experimental results are reported. We conclude in Section 8.

2 Previous Work

A number of constructive root bounds have been proposed. Here we briefly recall some of them.

Canny's bound. For a zero-dimensional system Σ of n polynomial equations with n unknowns, Canny [4] shows that if $(\alpha_1, \ldots, \alpha_n)$ is a solution, then $|\alpha_i| \geq (3dc)^{-nd^n}$ for each non-zero component α_i. Here c (resp., d) is an upper bound on the absolute value of coefficients (resp., the degree) of any polynomial in the system. An important proviso in Canny's bound is that the homogenized system $\widehat{\Sigma}$ has a non-vanishing U-resultant. Equivalently, $\widehat{\Sigma}$ has finitely many roots at infinity. Yap [18, p. 350] gives the treatment for the general case, based on the notion of "generalized U-resultant". Such multivariate root bounds are easily translated into a bound on expressions, as discussed in [2].

Degree-length and degree-height bounds. The degree-length bound [18] is a general bound for algebraic expressions, based on Landau's root bound. A similar degree-height bound based on Cauchy's root bound is found in [19]. Here "length" and "height" refer to the 2-norm and ∞-norm of a polynomial, respectively. These results are based on the resultant calculus.

Degree-measure bound. Given a polynomial $P(x) = a_m \prod_{i=1}^{m}(x - \alpha_i)$, with $a_m \neq 0$, the *measure* of P, $m(P)$, is defined as $|a_m| \cdot \prod_{i=1}^{m} \max\{1, |\alpha_i|\}$. Furthermore, the measure $m(\alpha)$ of an algebraic number α is defined as the measure of $\mathrm{Irr}(\alpha)$. It is known that if $\alpha \neq 0$, we have

$$(2.7) \qquad \frac{1}{m(\alpha)} \leq |\alpha| \leq m(\alpha).$$

Based on Mignotte's work, Burnikel et al [2] develop recursive rules to maintain the upper bounds for degrees and measures and call it the *degree-measure bound*. These rules are given in the last two columns of Table 5 where $M'(E)$ and $D'(E)$ are (respectively) upper bounds on $m(E)$ and $\deg(E)$. Similar rules are given in [12]. The degree-measure bound turns out to be always better than the degree-length bound.

BFMS bound. Burnikel et al [2] discovered a new bound for radical expressions (see (1.4) and (1.5)). Their bound for division-free expressions

(1.5) is proved to be better than any of the previous bounds. But in presence of divisions, the BFMS bound is not necessarily an improvement of the degree-measure bound (see Section 6). Conceptually the BFMS approach first transforms a radical expression E to a quotient of two division-free expressions $U(E)$ and $L(E)$. Two parameters $u(E)$ and $l(E)$, the upper bounds on the conjugates of $U(E)$ and $L(E)$, respectively, are maintained by the recursively rules in Table 1.

Scheinerman bound. This adopts an interesting approach based on matrix eigenvalues [13]. Let $\Lambda(n, b)$ denote the set of eigenvalues of $n \times n$ matrices with integer entries with absolute value at most b. It is easy to see that $\Lambda(n, b)$ is a finite set of algebraic integers. Moreover, if $\alpha \in \Lambda(n, b)$ is non-zero then $|\alpha| \geq (nb)^{1-n}$. Scheinerman gives a constructive root bound for division-free radical expressions E by maintaining two parameters, $n(E)$ and $b(E)$, satisfying the property that the value of E is in $\Lambda(n(E), b(E))$. These recursive rules are given by Table 2.

Note that the rule for $\sqrt{cd}$ is rather special, but it can be extremely useful. In Rule 6, the polynomial $\overline{P}(x)$ is given by $\sum_{i=0}^{d} |a_i| x^i$ when $P(x) = \sum_{i=0}^{d} a_i x^i$. This rule is not explicitly stated in [13], but can be deduced from an example he gave. An example given in [13] is to test whether $\alpha = \sqrt{2} + \sqrt{5 - 2\sqrt{6}} - \sqrt{3}$ is zero. Scheinerman's bound requires calculating α to 39 digits while the BFMS bound says 12 digits are enough.

3 The General Framework

We formalize the constructive root bound problem as follows. For the purposes of this paper, a "DAG" is an ordered, directed acyclic graph with a unique node that has out-degree 0, called the *root*. The DAG is ordered in the sense that the set of incoming edges to each node u is given a total ordering. Nodes with in-degree 0 are called *leaves*. Let Ω be a set of algebraic operations: each $\omega \in \Omega$ represents a partial function $f_\omega : \mathbb{C}^k \to \mathbb{C}$ where $\mathbb{C}$ are the complex numbers and $k = k(\omega)$ is called the *arity* of ω. If $k(\omega) = 0$ then ω may be identified with an element of $\mathbb{C}$ and is called a *constant*. An *expression* over Ω (or Ω-expression) is a DAG where each node u of in-degree k_u is labeled by an operation $\omega \in \Omega$ where $k(\omega)$ equals the in-degree of u. In particular the leaves are labeled by constants. In case[1] the DAG is a tree, we call it a *tree expression*. Each node in an expression induces a natural subexpression. Let $\mathcal{E}(\Omega)$ denote the set of Ω-expressions. The following classes of expressions are the main ones in this paper:

- $\Omega_0 = \{\pm, \times\} \cup \mathbb{Z}$ (where $\mathbb{Z}$ are the integers). Thus Ω_0-expressions are polynomials.
- $\Omega_1 = \Omega_0 \cup \{\div\}$. Thus Ω_1-expressions are rational expressions.
- $\Omega_2 = \Omega_1 \cup \{\sqrt[n]{\cdot} : n \geq 2\}$. Thus Ω_2-expressions are radical expressions.
- $\Omega_3 = \Omega_2 \cup \{\mathrm{Root}(P) : P \in \mathbb{Z}[x]\}$. Our main root bound applies to Ω_3-expressions. We assume the polynomial P is presented by its sequence of $n + 1$ integer coefficients if $\deg(P) = n$.

We need to clarify the $\mathrm{Root}(P)$ operation in Ω_3 above. This is intended to be a constant referring to some root α of P. In practice, we will need some method for identifying the root α. For instance, if α is real (as we assume in our applications) and it is the kth largest real root of P, we could identify α as "$\mathrm{Root}(P, k)$". Instead of k, we could also use, say, an isolating interval for α. It turns out that our root bounds do not depend on the choice of the root of P, and hence, we normally write "$\mathrm{Root}(P)$" instead of "$\mathrm{Root}(P, k)$".

For any set $\mathcal{E}$ of expressions, there is a partial function $\mathrm{val} : \mathcal{E} \to \mathbb{C}$ that is naturally defined by applying the appropriate functions f_ω ($\omega \in \Omega$) at each node of an expression. Notice that $\mathrm{val}(E)$ is undefined if any of its nodes has an undefined value, for instance, $\mathrm{val}(\mathrm{Root}(P, k))$ is undefined if $P(x)$ has less than k real roots. Having undefined values is not a new phenomenon, since this already arises when we divide by zero or take the square-root of a negative number (when values are assumed to be real). All our statements about $\mathrm{val}(E)$ are

[1] In some literature, our tree expressions are simply called "expressions" while our expressions are essentially "straight-line programs" or "circuits".

Table 1: BFMS Rules

	E	$u(E)$	$l(E)$
1.	integer a	$\|a\|$	1
2.	$E_1 \pm E_2$	$u(E_1)l(E_2) + l(E_1)u(E_2)$	$l(E_1)l(E_2)$
3.	$E_1 \times E_2$	$u(E_1)u(E_2)$	$l(E_1)l(E_2)$
4.	$E_1 \div E_2$	$u(E_1)l(E_2)$	$l(E_1)u(E_2)$
5.	$\sqrt[k]{E_1}$	$\sqrt[k]{u(E_1)}$	$\sqrt[k]{l(E_1)}$

Table 2: Scheinerman's Rules

	E	$n(E)$	$b(E)$
1.	integer a	1	$\|a\|$
2.	$\sqrt{cd}$	2	$\max\{\|c\|, \|d\|\}$
3.	$E_1 \pm E_2$	$n_1 n_2$	$b_1 + b_2$
4.	$E_1 \times E_2$	$n_1 n_2$	$b_1 b_2$
5.	$\sqrt[k]{E_1}$	kn_1	b_1
6.	$P(E_1)$	n_1	$\overline{P}(n_1 b_1)$

also conditioned on $\mathrm{val}(E)$ being defined. The *constructive root bound problem* for a class $\mathcal{E}$ of expressions is that of providing a *bounding function*

$$B : \mathbb{R}^m \to \mathbb{R} \quad (\mathbb{R} = \text{reals})$$

and a set of "recursive rules" to compute for each $E \in \mathcal{E}$ a set of real parameters $\{a_i(E) : i = 1, \ldots, m\}$, plus possibly other non-numeric parameters, such that the following holds:

$$\mathrm{val}(E) \neq 0 \Rightarrow |\mathrm{val}(E)| \geq B(a_1(E), \ldots, a_m(E)).$$

The rules are "recursive" in the sense that the parameters for each node in the DAG can be effectively computed from the parameters of its predecessors. Non-numeric parameters are needed, for instance, in the BFMS bound, where for each node u in an expression, we need to compute the set of predecessors of u that are radical nodes. In practice, the function B will be non-negative, with both B and the recursive rules relatively simple to compute. Another desirable property is that the bound $B(a_1(E), \ldots, a_m(E))$ should be as large as possible. Also, we call m the *order* of constructive root bound.

Example: In the original degree-measure bound, we compute two parameters, $a_1(E)$ and $a_2(E)$ where a_1 and a_2 are upper bounds on the degree and measure of E. Moreover, the bounding function $B : \mathbb{R}^m \to \mathbb{R}$ is given by $B(a, b) = 1/b$ (the first parameter is ignored by B). So the order of the degree-measure bound is $m = 2$.

Example: if there are 2 constructive root bounds using bounding function $B_1(x_1, \ldots, x_m)$ and $B_2(y_1, \ldots, y_n)$ then we can have a new constructive root bound using the bounding function $B(x_1, \ldots, x_m, y_1, \ldots, y_n) = \max\{B_1(x_1, \ldots, x_m), B_2(y_1, \ldots, y_n)\}$ of order $m + n$. For instance, in the Core Library, we actually choose the maximum of the BFMS bound, the improved degree-measure bound, and our new bound.

4 New Constructive Root Bound

In this section, we develop a constructive root bound for Ω_3-expressions. In order to obtain a root bound for an expression E using the relation (1.6), we need three parameters: $\deg(E), \mu(E)$ and $\mathrm{lead}(E)$. The definitions of these parameters involve the minimal polynomial of E, which is usually expensive to compute. Instead, we give recursive rules to maintain upper bounds

$$D(E), \quad \overline{\mu}(E), \quad \mathrm{lc}(E)$$

on the corresponding parameters.

First we consider $D(E)$, using the same approach as BFMS. Suppose that E has k radical nodes or root-of-polynomial nodes $\{r_1, r_2, \ldots, r_k\}$. Assume some topological sorting $r_1 \prec r_2 \prec \cdots \prec r_k$ of these nodes so that if r_i is a predecessor of r_j then $i < j$. Clearly, the degree of E over $\mathbb{Q}$ is no more than $\prod_{i=1}^{k} d_i$ where d_i is the degree of r_i over the extension field $\mathbb{Q}(r_1, \ldots, r_{i-1})$. Define $D(E) = \prod_{i=1}^{k} k_i$ where k_i is either the index of r_i if r_i is a radical node, or the degree of the polynomial if r_i is a polynomial-root node. Thus, $D(E)$ is an upper bound on $\deg(E)$ since $d_i \leq k_i$ for all i.

Given a non-zero polynomial $P(x)$, we denote its leading coefficient, its tail coefficient and its constant coefficient (respectively) by $\mathrm{lead}(P)$, $\mathrm{tail}(P)$, and $\mathrm{const}(P)$. Note that the $\mathrm{tail}(P)$ is defined to be the last non-zero coefficient of P. By definition, $\mathrm{lead}(P), \mathrm{tail}(P) \neq 0$. Also, let $m(P)$ denote the measure of P. Given an algebraic number α, we define $\mathrm{lead}(\alpha)$, $\mathrm{tail}(\alpha)$, etcetra, to be $\mathrm{lead}(\mathrm{Irr}(\alpha))$, $\mathrm{tail}(\mathrm{Irr}(\alpha))$, etcetra. Actually, $\mathrm{tail}(\alpha)$ is the same as $\mathrm{const}(\mathrm{Irr}(\alpha))$ (when $\alpha = 0$ this is true by definition).

Bound on Leading Coefficients and Table 3. We now consider $\mathrm{lc}(E)$, which is an upper bound on $|\mathrm{lead}(E)|$. The admission of divisions makes it necessary to bound tail coefficients as well. Moreover, we also need to bound the measure of E to help bound $|\mathrm{tail}(E)|$ (this is only used when E has the form $E = E_1 \pm E_2$). Let $\mathrm{tc}(E)$ and $M(E)$ denote upper bounds on $|\mathrm{tail}(E)|$ and $m(E)$. Table 3 gives the recursive rules to maintain $\mathrm{lc}(E), \mathrm{tc}(E)$ and $M(E)$.

The upper bound $M(E)$ on the measure of E is shown[2] in the last column. We note that the invariant $\mathrm{tc}(E) \leq M(E)$ is implicitly assumed. In other words, if the rule for $\mathrm{tc}(E)$ gives a value larger than $M(E)$, it is implicit in our rules to replace $\mathrm{tc}(E)$ by $M(E)$. Also note that we introduce a special node for the power operation $E = E_1^k$. This is not just a shortcut for $(k-1)$ multiplications; it leads to much better bounds too. For example, in computing $\mathrm{lc}(E)$ by naively expanding E into $(k-1)$ multiplications, we get $\mathrm{lc}(E) = \mathrm{lc}_1^{kD_1^{k-1}} \gg \mathrm{lc}_1^k$. Similar improvements can be shown for $\mathrm{tc}(E)$ and $m(E)$.

[2] This information is copied from column 2 in Table 5, and is discussed in conjunction with that table.

The justification of Table 3 is omitted in this extended abstract. The basic techniques come from resultant calculus. One subtlety arises for expressions of the form $E = E_1 \pm E_2$. In this case, resultant calculus gives us a polynomial $P_E(x)$ where $\mathrm{val}(E)$ vanishes. We can also deduce a bound on $\mathrm{const}(P_E)$. Unfortunately, this constant coefficient may vanish and tell us nothing about $\mathrm{tail}(E)$. Hence we need to resort to $m(E)$ as a bound for $|\mathrm{tail}(E)|$.

Bound on Conjugates and Table 4. Now we consider $\overline{\mu}(E)$, which is an upper bound on the absolute value of all the conjugates of $\mathrm{val}(E)$. Because of the admission of divisions, we also have to maintain $\underline{\nu}(E)$, which is a lower bound on the absolute value of all the conjugates of $\mathrm{val}(E)$ whenever $\mathrm{val}(E) \neq 0$. The recursive rules to maintain these two bounds are given in Table 4. The most noteworthy entry in Table 4 is the bound for $\underline{\nu}(E)$ when $E = E_1 \pm E_2$. In this case, we cannot obtain a lower bound on $\nu(E)$ based on $\underline{\nu}(E_1)$ and $\underline{\nu}(E_2)$. Instead, we can use either relation (1.6) or (2.7). As we will see in Section 6, neither bound is strictly better than the other. Hence, we take the maximum of the two bounds for $\underline{\nu}(E_1 \pm E_2)$. More generally, we could use (2.7) in all the entries of Table 4 if they give better bounds. We also note that our bounds on $\overline{\mu}(\mathrm{Root}(P))$ and $\underline{\nu}(\mathrm{Root}(P))$ are based on Cauchy's root bound [18, p. 148]. Of course, any of the classical root bounds can be used as convenient.

The justification of Table 4 is omitted in this abstract.

Finally, we obtain the new root bound in the following theorem:

THEOREM 4.1. *Given an Ω_3-expression E, if $E \neq 0$, then*

$$|E| \geq (\overline{\mu}(E)^{(D(E)-1)}\mathrm{lc}(E))^{-1}. \quad (4.8)$$

5 Improved Degree-Measure Bound

Let E be a Ω_3-expression. As explained in Section 2, $M'(E)$ and $D'(E)$ in Table 5 are the original degree-measure bound [11, 12, 2]. Recall the definition of $D(E)$ which gives an upper bound on

Table 3: Recursive rules for lc(E) (and associated tc(E) and $M(E)$)

	E	lc(E)	tc(E)	$M(E)$
1.	rational $\frac{a}{b}$	$\lvert b\rvert$	$\lvert a\rvert$	$\max\{\lvert a\rvert, \lvert b\rvert\}$
2.	Root(P)	$\lvert\mathrm{lead}(P)\rvert$	$\lvert\mathrm{tail}(P)\rvert$	$\lVert P\rVert_2$
3.	$E_1 \pm E_2$	$\mathrm{lc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$M_1^{D_2}M_2^{D_1}2^{D(E)}$	$M_1^{D_2}M_2^{D_1}2^{D(E)}$
4.	$E_1 \times E_2$	$\mathrm{lc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$\mathrm{tc}_1^{D_2}\mathrm{tc}_2^{D_1}$	$M_1^{D_2}M_2^{D_1}$
5.	$E_1 \div E_2$	$\mathrm{lc}_1^{D_2}\mathrm{tc}_2^{D_1}$	$\mathrm{tc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$M_1^{D_2}M_2^{D_1}$
6.	$\sqrt[k]{E_1}$	lc_1	tc_1	M_1
7.	E_1^k	lc_1^k	tc_1^k	M_1^k

Table 4: Recursive rules for $\overline{\mu}(E)$ and $\underline{\nu}(E)$

	E	$\overline{\mu}(E)$	$\underline{\nu}(E)$
1.	rational $\frac{a}{b}$	$\lvert\frac{a}{b}\rvert$	$\lvert\frac{a}{b}\rvert$
2.	Root(P)	$1 + \lVert P\rVert_\infty$	$(1 + \lVert P\rVert_\infty)^{-1}$
3.	$E_1 \pm E_2$	$\overline{\mu}(E_1) + \overline{\mu}(E_2)$	$\max\{M(E)^{-1}, (\overline{\mu}(E)^{D(E)-1}\mathrm{lc}(E))^{-1}\}$
4.	$E_1 \times E_2$	$\overline{\mu}(E_1)\overline{\mu}(E_2)$	$\underline{\nu}(E_1)\underline{\nu}(E_2)$
5.	$E_1 \div E_2$	$\overline{\mu}(E_1)/\underline{\nu}(E_2)$	$\underline{\nu}(E_1)/\overline{\mu}(E_2)$
6.	$\sqrt[k]{E_1}$	$\sqrt[k]{\overline{\mu}(E_1)}$	$\sqrt[k]{\underline{\nu}(E_1)}$
7.	E_1^k	$\overline{\mu}(E_1)^k$	$\underline{\nu}(E_1)^k$

$\deg(E)$. It is clear that $D(E)$ is never larger than $D'(E)$. Based on $D(E)$, we now give an improved upper bound (denoted by $M(E)$ in Table 5) on measures.

When $E = \mathrm{Root}(P)$ for some polynomial P, we use $\lVert P\rVert_2$ as the upper bound $M(E)$ because $\lVert P\rVert_2 \geq m(P)$. Besides the introduction of the new operations of Root(P) and power (E_1^k) in Table 5, we give a slightly improved rule for the measure of $E_1 \pm E_2$. Basically, we can replace the factor of $2^{D_1' D_2'}$ by 2^D. Here is the justification:

LEMMA 5.1. *If α and β are algebraic numbers with degrees m, n, respectively, then the measure of $\alpha \pm \beta$ is bounded by $2^d m(\alpha)^n m(\beta)^m$ where $d = \deg(\alpha \pm \beta)$.*

Proof is omitted for this abstract.

The improvement can be significant when there is sharing of subexpressions. For example, consider $E = ((\sqrt{x} + \sqrt{y}) - 2\sqrt{x + y + 2\sqrt{x}\sqrt{y}}) \cdot ((\sqrt{x} + \sqrt{y}) + 2\sqrt{x + y + 2\sqrt{x}\sqrt{y}})$ where x and y are L-bit integers. The original degree-measure bound for E is $2^{3584L+7148}$. But when all the common subexpressions of E are merged, our new bound gives $2^{896L+1408}$.

6 Comparison of the Root Bounds

We compare the various root bounds discussed in this paper. Because these bounds are rather different in recursive form, a direct comparison is sometimes not possible. Hence we compare their behavior on interesting classes $\mathcal{C}$ of expressions. For the ith constructive root bound (i=BFMS, degree-measure, etc) and expression $E \in \mathcal{C}$, let $B_i(E)$ denote the root bit-bound for E. Let $\mathcal{C}(L)$ denote the expressions in $\mathcal{C}$ whose input parameters are L-bit integers (or rational numbers, as the case may be). We want to study functions $\beta_i(L)$

$$\beta_i(L) = \max\{B_i(E) : E \in \mathcal{C}(L)\}.$$

Note that even if $\beta_i(L) \leq \beta_j(L)$ for all L, it does not necessarily mean that $B_i(E) \leq B_j(E)$ for all $E \in \mathcal{C}$. Sometimes such a stronger relation can be asserted.

1. By an examination of our tables, we can assert the following:

LEMMA 6.1. *For any division-free radical expression E, our new bound is exactly the same as the*

Table 5: The original and our improved degree-measure bounds

	E	$M(E)$ (new)	$M'(E)$ (old)	$D'(E)$ (old)
1.	rational $\frac{a}{b}$	$\max\{\lvert a\rvert, \lvert b\rvert\}$	$\max\{\lvert a\rvert, \lvert b\rvert\}$	1
2.	Root(P)	$\lVert P\rVert_2$	–	–
3.	$E_1 \pm E_2$	$M_1^{D_2} M_2^{D_1} 2^{D(E)}$	${M'_1}^{D'_2} {M'_2}^{D'_1} 2^{D'_1 D'_2}$	$D'_1 D'_2$
4.	$E_1 \times E_2$	$M_1^{D_2} M_2^{D_1}$	${M'_1}^{D'_2} {M'_2}^{D'_1}$	$D'_1 D'_2$
5.	$E_1 \div E_2$	$M_1^{D_2} M_2^{D_1}$	${M'_1}^{D'_2} {M'_2}^{D'_1}$	$D'_1 D'_2$
6.	$\sqrt[k]{E_1}$	M_1	M'_1	kD'_1
7.	E_1^k	M_1^k	${M'_1}^k$	–

BFMS bound.

2. In [2], it's shown that for division-free radical expressions, the BFMS bound is better than the degree-measure bound and Canny's bound. Here we strengthen this conclusion in showing that for this class of expressions, the BFMS bound is never worse than the Scheinerman bound.

Lemma 6.2. *For every division-free constant radical expression E, the BFMS bound for E is at least as tight as the Scheinerman bound.*

3. Next consider the sum of square roots of rational numbers. Suppose $E = \sum_{i=1}^{n}(\pm\sqrt{a_i})$ where a_i are L-bit rational numbers. Then the degree-measure root bit-bound is $2^{n-1}(n(L+2)-2)$. The BFMS bound is $2^{2n-1}(2\log_2 n + nL) - \log_2 n$. Our bound gives $2^{n-1}(2\log_2 n + (n+1)L) - (\log_2 n + L/2)$. It may be verified that both the degree-measure bound and our bound are better than the BFMS bound. The difference between our root bound and degree-measure bound in terms of bit-bound is

$$\Delta = 2^n(\log_2 n + (L/2) - n + 1) - (\log_2 n + L/2).$$

Note that Δ can be positive or negative, depending on the relative sizes of n and L. Thus our new bound is incomparable with the degree-measure bound.

4. Consider the expression $E = \frac{a+b^{2^{-r}}}{c} - \frac{d+e^{2^{-r}}}{f}$ with $2r$ ($r \geq 1$) square roots. This is a generalization of the Fortune predicate in Equation (1.1). Also, we now assume that a, b, c, d, e and f are all L-bit integers. The BFMS root bit-bound is $(5L+1)\cdot 2^{4r} - (L+2)$. The degree-measure root bit-bound is $(2L+3)\cdot 2^{2r} + L\cdot 2^{r+1}$. Our new bound gives $(2L+3)\cdot 2^{2r} - 3$ which is the best.

5. Consider the continued fraction expression:

$$E_n = a_n + \cfrac{1}{a_{n-1} + \frac{1}{a_{n-2}+\cdots}} = [a_n, a_{n-1}, a_{n-2}, \ldots],$$

where the a_i's are L-bit integers. Our new bound for E_n is $(n-1)L + (n-2)$. The degree-measure bound gives $(n+1)L - 1$. And the best bound for this example is given by the BFMS approach as $(n-1)L - n + 3$. This is close to the best possible bound $(n-2)L - n$. Note that the degree of E_n is 1. Now we modify E_n to get an expression E'_n with degree up to 2^n by replacing the integers a_i with square roots $\sqrt{a_i}$,

$$E'_n = [\sqrt{a_n}, \sqrt{a_{n-1}}, \sqrt{a_{n-2}}, \ldots].$$

Then our new bound for E'_n is $2^{2n-1}((n-1)L + 2n - 3)$. The BFMS bound gives $c\cdot 2^{2n}nL$, for some constant $c \geq 1$. Here the degree-measure approach gives the best bound $2^{n-1}(nL + 2n - 2)$

6. Next, we compare our new bound with the BFMS bound for a restricted class of radical expressions.

Proposition 6.3. *Given a radical expression E with rational values as the leaves but no division operations in the internal nodes, we have*

$$\overline{\mu}(E)l(E) = u(E) \tag{6.9}$$

where $\overline{\mu}(E)$ is as defined in our new bound, and $u(E), l(E)$ are defined as in the BFMS bound. Furthermore, if there are no shared radical nodes in E, then

$$\mathrm{lc}(E) \leq l(E)^{D(E)}. \tag{6.10}$$

Corollary 6.4. *Given a radical expression E with rational values at the leaves, if E has no divisions and* shared *radical nodes, our new root bound for E is never worse than the BFMS bound.*

It can be shown that our bound is strictly better than BFMS in this class whenever $u(E) > 1$ and $D(E) > 1$. If shared radical nodes are permitted, our bound can be proved to be better than the BFMS bound if $(D' - D)\log_2 l(E) < D(D-1)\log_2 u(E)$, where $D'(E)$ is a degree bound of E with the value $\prod_{i=1}^{k} k_i^{c_i}$, $c_i \geq 1$ is the number of different paths from the root to the radical node r_i.

7 Experimental Results

The new constructive root bound has been implemented in our Core Library [7]. In implementation, we maintain an upper bound on the root *bit*-bound, instead of the root bounds themselves. The logarithms are represented and manipulated as objects of the class extLong, which is basically a wrapper around the standard long number type, together with facilities to handle overflows. Such a class may be used to support "level arithmetic" in which for any integer x, we maintain the ceiling of $\lg^{(i)} x$ where $\lg^{(i)}$ denotes i iterations of $\log_2(\cdot)$, i is the smallest natural number such that $\lceil \lg^{(i)} x \rceil$ fits in built-in integer types. Therefore, the integer x is represented by a pair $(\lceil \lg^{(i)} x \rceil, i)$.

Our experiments, based on Version 1.3 of the Core library, will compare the performance of our new bound with the BFMS and degree-measure bounds. All the tests are performed on a Sun UltraSPARC with a 440 MHz CPU and 512MB main memory. All timings are in seconds.

1. Recall the critical test in Fortune's sweepline algorithm is to determine the sign of the expression $E = \frac{a+\sqrt{b}}{d} - \frac{a'+\sqrt{b'}}{d'}$ in Equation (1.1) where a's, b's and d's are $3L$-, $6L$- and $2L$-bit integers, respectively. The BFMS bound requires $(79L + 30)$ bits and the degree-measure (D-M) bound needs $(64L + 12)$ bits. Our root bit-bound improves the bound to $(19L+9)$ bits. We generate some random inputs with different L values which always make $E = 0$, and put the timings of the tests in Table 6. We also converted the Fortune's implementation of this algorithm to use Core library. We ran the program on two kinds of inputs: (1) First we test on a non-degenerate data set (100 random points provided in Fortune's code distribution). The time for our new bound is 3.62 seconds while the BFMS and D-M bounds take 3.75 and 3.64 seconds, respectively. This is not unexpected, since as explained in the introduction, our Core library exploits the progressive evaluation technique, and the signs of Fortune's predicate on non-degenerate inputs can be determined using the inequality (1.2). Thus the inequality (1.3) based on root bounds has no effect on the complexity. (2) We used highly degenerate inputs comprising points on a (32×32) uniform grid with coordinates being L bits long. The timings are reported in Table 7.

Table 6: Timings for Fortune's expression in (1.1)

L	10	20	50	100	200
NEW	0.01	0.03	0.12	0.69	3.90
BFMS	0.03	0.24	1.63	11.69	79.43
D-M	0.03	0.22	1.62	10.99	84.54

Table 7: Timings for Fortune's algorithm on degenerate inputs

L	10	20	30	50
NEW	35.2	41.7	47.5	112.3
BFMS	86.1	1014.1	1218.1	5892.2
D-M	418.5	1681.6	1874.4	> 2 hrs

2. The second test is to verify an expression which is identically zero. Let $x = \frac{a}{b}$ and $y = \frac{c}{d}$ (a, b, c, d are L-bit integers), and $E = (\sqrt{x} + \sqrt{y}) - \sqrt{x + y + 2\sqrt{xy}}$. Our new bound requires computing $(40L+38)$ bits, while the BFMS and the degree-measure (D-M) bounds require $(640L+510)$ and $(80L + 56)$ bits, respectively. The timings are in Table 8.

Table 8: Timings for Example 2

L	5	10	30	50
NEW	0.08	0.09	1.77	55.43
BFMS	88.16	91.03	3071.21	> 2 hrs
D-M	1.71	1.79	89.72	531.25

In comparing the timings, it is the *relative* speedup that should be stressed. We expect similar relative improvements to show up if the comparisons were made in other systems such as LEDA.

8 Conclusion

We have described a new constructive root bound for a large class of algebraic expressions. The main achievement is that our root bit-bound is only linear in the algebraic degree (or some upper bound thereof) of the expression.

We implemented the new bound in our Core library and experiments show that it can achieve remarkable speedup over previous bounds in the presence of division. Although we have described our bounds for the class of Ω_3-expressions, it should be clear that our methods extend to more general expressions. A comparative performance study of the BFMS bound, the degree-measure bound and our new bound shows that these are generally incomparable. So in practice, it may be worthwhile maintaining all of them simultaneously to choose the best.

A challenge for future work is to find better ways of bounding $\mathrm{tc}(E)$ for addition and subtraction nodes. The current bound on tail coefficients based on the polynomial measure seems to be too conservative for expressions with complex structure and large depth. Also, it may be possible to improve our current bound on $\underline{\nu}(E_1 \pm E_2)$.

References

[1] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. Exact geometric computation made easy. In *Proc. 15th ACM Symp. Comp. Geom.*, 1999.

[2] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. A strong and easily computable separation bound for arithmetic expressions involving radicals. *Algorithmica*, 27:87–99, 2000.

[3] C. Burnikel, J. Könnemann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Exact geometric computation in LEDA. In *Proc. 11th ACM Symp. Computational Geom.*, 1995.

[4] J. F. Canny. *The complexity of robot motion planning*. ACM Doctoral Dissertion Award Series. The MIT Press, 1988. PhD thesis, M.I.T.

[5] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[6] The CGAL Homepage. Computational Geometry Algorithms Library (CGAL) Project. URL `http://www.cs.uu.nl/CGAL/`.

[7] The CORE Project Homepage. URL `http://www.cs.nyu.edu/exact/`.

[8] The LEDA Homepage. URL `http://www.mpi-sb.mpg.de/LEDA/`.

[9] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A Core Library for robust numeric and geometric computation. In *Proc. 15th ACM Symp. on Computational Geometry*, pages 351–359, June 1999.

[10] M. Marden. *The geometry of the zeroes of a polynomial in a complex variable*. American Mathematical Society, 1949.

[11] M. Mignotte. Identification of algebraic numbers. *Journal of Algorithms*, 3(3), 1982.

[12] M. Mignotte and D. Ştefănescu. *Polynomials: An Algorithmic Approach*. Springer, 1999.

[13] E. R. Scheinerman. When close enough is close enough. *American Mathematical Monthly*, 107:489–499, 2000.

[14] S. Schirra. Robustness and precision issues in geometric computation. Research Report MPI-I-98-1-004, Max-Planck-Institut für Informatik, Saarbrücken, Germany, January 1996.

[15] D. Tulone, C. Yap, and C. Li. Randomized zero testing of radical expressions and elementary geometry theorem proving. In *Proceedings of the Third International Workshop on Automated Deduction in Geometry (ADG 2000)*, Sept. 2000.

[16] C. Yap. Lecture notes in robust geometric computation, 2000. URL `http://www.cs.nyu.edu/~yap/classes/exact/`.

[17] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 35, pages 653–668. CRC Press LLC, 1997.

[18] C. K. Yap. *Fundamental Problems in Algorithmic Algebra*. Oxford Univ. Press, Dec. 1999.

[19] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, pages 452–486. World Scientific Press, 1995. 2nd edition.