Toward accurate polynomial evaluation in rounded arithmetic (short report)

James Demmel[*], Ioana Dumitriu[†], and Olga Holtz[‡]

November 12, 2005

Abstract

Given a multivariate real (or complex) polynomial p and a domain $\mathcal{D}$, we would like to decide whether an algorithm exists to evaluate $p(x)$ accurately for all $x \in \mathcal{D}$ using rounded real (or complex) arithmetic. Here "accurately" means with relative error less than 1, i.e., with some correct leading digits. The answer depends on the model of rounded arithmetic: We assume that for any arithmetic operator $op(a, b)$, for example $a+b$ or $a \cdot b$, its computed value is $op(a, b) \cdot (1+\delta)$, where $|\delta|$ is bounded by some constant ϵ where $0 < \epsilon \ll 1$, but δ is otherwise arbitrary. This model is the traditional one used to analyze the accuracy of floating point algorithms.

Our ultimate goal is to establish a decision procedure that, for any p and $\mathcal{D}$, either exhibits an accurate algorithm or proves that none exists. In contrast to the case where numbers are stored and manipulated as finite bit strings (e.g., as floating point numbers or rational numbers) we show that some polynomials p are impossible to evaluate accurately. The existence of an accurate algorithm will depend not just on p and $\mathcal{D}$, but on which arithmetic operators and constants are available to the algorithm and whether branching is permitted in the algorithm.

Toward this goal, we present necessary conditions on p for it to be accurately evaluable on open real or complex domains $\mathcal{D}$. We also give sufficient conditions, and describe progress toward a complete decision procedure. We do present a complete decision procedure for homogeneous polynomials p with integer coefficients, $\mathcal{D} = \mathbb{C}^n$, using only arithmetic operations $+$, $-$ and $\cdot$.

1 Introduction

In actual computations "real numbers" are represented by floating point numbers $r = m \cdot \beta^e$, where m is a finite precision mantissa, β is a fixed radix (2 or 10), and e is an integer exponent. Viewing these as rational numbers makes it clear that all algebraic expressions can be evaluated exactly, but possibly at high cost. The usual alternative is to think of each arithmetic operation as introducing a multiplicative error $1 + \delta$ with $|\delta| \leq \epsilon \ll 1$, caused by rounding m with a relative error bounded by ϵ. Sometimes composite operations like $x + y \cdot z$ are carefully implemented so that they too produce the exact answer times some $1 + \delta$.

[*]Mathematics Department and CS Division, University of California, Berkeley, CA 94720. The author acknowledges the support of NSF under grants CCF-0444486, ACI-00090127, CNS-0325873 and of DOE under grant DE-FC02-01ER25478.

[†]Mathematics Department, University of California, Berkeley, CA 94720. The author acknowledges the support of the Miller Institute for Basic Research in Science.

[‡]Mathematics Department, University of California, Berkeley, CA 94720.

Dagstuhl Seminar Proceedings 05391
Algebraic and Numerical Algorithms and Computer-assisted Proofs
http://drops.dagstuhl.de/opus/volltexte/2006/447

We use this model as our starting point, and ask which expressions permit accurate evaluation, given a set of rounded arithmetic operations, possibly including an arbitrary set of "black-box" operations like $x + y \cdot z$. By treating δ as a tiny but otherwise arbitrary real (or complex) number, we will see that some expressions are in fact impossible to evaluate accurately. The practical implication is that higher precision arithmetic (in the form of more accurately implemented "black-box operations") is *necessary* for accurate evaluation of such expressions. Indeed, our goal is a *decision procedure* that takes any expression and identifies whether it can be evaluated accurately, and provides the algorithm if it exists. The impact would be both to formalize the process of accurate algorithm generation [9] and to systematize recent results [5] identifying apparently disparate classes of structured matrices for which efficient and accurate linear algebra algorithms exist.

We give some examples to illustrate our results. Consider the family of homogeneous polynomials $M_{jk}(x) = j \cdot x_3^6 + x_1^2 \cdot x_2^2 \cdot (j \cdot x_1^2 + j \cdot x_2^2 - k \cdot x_3^2)$ where j and k are positive integers, $\mathcal{D} = \mathbb{R}^n$, and we allow only addition, subtraction and multiplication of two arguments as basic arithmetic operations, along with comparisons and branching. When $k/j < 3$, $M_{jk}(x)$ is *positive definite,* i.e., zero only at the origin and positive elsewhere. This will mean that $M_{jk}(x)$ is easy to evaluate accurately using a simple method discussed in Section 3. When $k/j > 3$, then we will show that $M_{jk}(x)$ cannot be evaluated accurately by *any* algorithm using only addition, subtraction and multiplication of two arguments. This will follow from a simple necessary condition on the real variety $V_{\mathbb{R}}(M_{jk})$, the set of real x where $M_{jk}(x) = 0$, see Theorem 4.4. When $k/j = 3$, i.e., on the boundary between the above two cases, $M_{jk}(x)$ is a multiple of the Motzkin polynomial [8]. The real variety $V_{\mathbb{R}}(M_{jk}) = \{x : |x_1| = |x_2| = |x_3|\}$ of this polynomial satisfies the necessary condition of Theorem 4.4, and the simplest accurate algorithm to evaluate it that we know of has 8 cases depending on the relative values of $|x_i \pm x_j|$, one branch of which evaluates p by the nonobvious formula $p = j \cdot (x_3^4 \cdot [4((x_1 - x_3)^2 + (x_2 - x_3)^2 + (x_1 - x_3)(x_2 - x_3))] + x_3^3 \cdot [2(2(x_1 - x_3)^3 + 5(x_2 - x_3)(x_1 - x_3)^2 + 5(x_2 - x_3)^2(x_1 - x_3) + 2(x_2 - x_3)^3)] + x_3^2 \cdot [(x_1 - x_3)^4 + 8(x_2 - x_3)(x_1 - x_3)^3 + 9(x_2 - x_3)^2(x_1 - x_3)^2 + 8(x_2 - x_3)^3(x_1 - x_3) + (x_2 - x_3)^4] + x_3 \cdot [2(x_2 - x_3)(x_1 - x_3)((x_1 - x_3)^3 + 2(x_2 - x_3)(x_1 - x_3)^2 + 2(x_2 - x_3)^2(x_1 - x_3) + (x_2 - x_3)^3)] + (x_2 - x_3)^2(x_1 - x_3)^2((x_1 - x_3)^2 + (x_2 - x_3)^2))$. In contrast to the real case, when $\mathcal{D} = \mathbb{C}^n$ then Theorem 4.4 will show that $M_{jk}(x)$ is not accurately evaluable using only addition, subtraction and multiplication.

The necessary condition for accurate evaluability of $p(x)$ in Theorem 4.4 depends only on the variety of $p(x)$, but the variety alone is not enough to determine accurate evaluability, at least in the real case. Consider the irreducible, homogeneous, degree $2d$, real polynomial $p(x) = (x_1^{2d} + x_2^{2d}) + (x_1^2 + x_2^2)(q(x_3, ..., x_n))^2$, where $q(\cdot)$ is homogeneous of degree $d - 1$. The variety $V(p) = \{x_1 = x_2 = 0\}$ satisfies the necessary condition for accurate evaluability, but near $V(p)$ the polynomial $p(x)$ is "dominated" by $(x_1^2 + x_2^2)(q(x_3, ..., x_n))^2$, so accurate evaluability of $p(x)$ depends on $q(\cdot)$. Applying the same principle to $q(\cdot)$, we see that any decision procedure must be recursive, expanding $p(x)$ near the components of its variety and so on. We show current progress toward a decision procedure in Section 4.3. In particular, Theorem 4.12 shows that, at least for algorithms without branching, being able to compute dominant terms of p (suitably defined) accurately on $\mathbb{R}^n$ is a necessary condition for computing p accurately on $\mathbb{R}^n$. Furthermore, Theorem 4.14 shows that accurate evaluability of the dominant terms, along with branching, is sufficient to evaluate p accurately. In contrast to the real case, Theorem 4.5 shows that for the complex case knowing $V(p)$ is necessary and sufficient to decide.

The rest of this paper is organized as follows. Section 2 discusses further details of our algorithmic model. Section 3 discusses the evaluation of positive polynomials. Section 4 discusses

necessary conditions (for real and complex data) and sufficient conditions (for complex data) for accurate evaluability, when using only classical arithmetic. Section 4.3 describes progress toward devising a decision procedure for accurate evaluability in the real case using classical arithmetic. Section 5 extends Section 4's necessary conditions to arbitrary black-box arithmetic operations, and gives sufficient conditions in the complex case. Section 6 is devoted to open problems and future work.

2 Models of Algorithms

Now we state more formally our decision question. We write the output of our algorithm as $p_{comp}(x, \delta)$, where $\delta = (\delta_1, \delta_2, ..., \delta_k)$ is the vector of rounding errors made during the algorithm.

Definition 2.1. *We say that $p_{comp}(x, \delta)$ is an* accurate algorithm *for the evaluation of $p(x)$ for $x \in \mathcal{D}$ if*

$\forall\ 0 < \eta < 1$ *... for any η = desired relative error*
$\exists\ 0 < \epsilon < 1$ *... there is an ϵ = machine precision*
$\forall\ x \in \mathcal{D}$ *... so that for all x in the domain*
$\forall\ |\delta_i| \leq \epsilon$ *... and for all rounding errors bounded by ϵ*
$|p_{comp}(x, \delta) - p(x)| \leq \eta \cdot |p(x)|$ *... the relative error is at most η.*

Our ultimate goal is a decision procedure (a "compiler") that takes $p(\cdot)$ and $\mathcal{D}$ as input, and either produces an accurate algorithm p_{comp} (including how to choose the machine precision ϵ given the desired relative error η) or exhibits a proof that none exists.

To be more precise, we must say what our set of possible algorithms includes. The above decision question is apparently not Tarski-decidable [7, 10] despite its appearance, because we see no way to express "there exists an algorithm" in that format.

A more formal description of the algorithms that we consider is as follows.

1. We insist that the inputs x are given exactly, rather than approximately.

2. We insist that the algorithm computes the output $p_{comp}(x, \delta)$ always in finitely many steps and, moreover, computes the exact value of $p(x)$ when all rounding errors $\delta = 0$. In particular, we exclude iterative algorithms which might produce an approximate value of $p(x)$ even when $\delta = 0$.

3. We must describe the basic arithmetic operations we consider, beyond addition, subtraction and multiplication. We refer to the model with only those three operations, together with exact negation, as *classical arithmetic*. The case when additional polynomial operations are included is referred to as *black-box arithmetic.* We must also describe the constants available to our algorithms.

4. We consider algorithms both with and without comparisons and branching, since this choice may change the set of polynomials that we can accurately evaluate.

5. If the computed value of an operation depends only the values of its operands, i.e., if the same operands x and y of $op(x, y)$ always yield the same δ in $rnd(op(x, y)) = op(x, y) \cdot (1 + \delta)$, then we call our model *deterministic,* else it is *nondeterministic.* One can show that comparisons

and branching let a nondeterministic machine simulate a deterministic one, and subsequently restrict our investigation to the easier nondeterministic model.

6. What domains of evaluation $\mathcal{D}$ do we consider? In principle, any semialgebraic set $\mathcal{D}$ is a possibility, but for simplicity we mostly consider open $\mathcal{D}$, especially $\mathcal{D} = \mathbb{R}^n$ or $\mathcal{D} = \mathbb{C}^n$. We point out issues in extending results to other $\mathcal{D}$.

For further details of these assumptions, and comparisons with other models, see [4].

3 Evaluating positive polynomials accurately

Here we address the simpler case where the polynomial $p(x)$ to be evaluated has no zeros in the domain of evaluation $\mathcal{D}$. It turns out that we need more than this to guarantee accurate evaluability: we will require that $|p(x)|$ be bounded both above and below in an appropriate manner on $\mathcal{D}$.

We let $\bar{\mathcal{D}}$ denote the closure of $\mathcal{D}$. (For proofs of this and subsequent results, see [4].)

Theorem 3.1. *Let $p_{comp}(x, \delta)$ be* any *algorithm for $p(x)$ satisfying $p_{comp}(x, 0) = p(x)$, i.e. it computes the right value in the absence of rounding error. Let $p_{min} := \inf_{x \in \bar{\mathcal{D}}} |p(x)|$. Suppose $\bar{\mathcal{D}}$ is compact and $p_{min} > 0$. Then $p_{comp}(x, \delta)$ is an accurate algorithm for $p(x)$ on $\mathcal{D}$.*

Next we consider domains $\mathcal{D}$ whose closure is not compact. To see that merely requiring $p_{min} > 0$ is not enough, consider evaluating $p(x) = 1 + (x_1 + x_2 + x_3)^2$ on $\mathbb{R}^3$. Intuitively, $p(x)$ can only be accurate if its "dominant term" $(x_1 + x_2 + x_3)^2$ is accurate, once it is large enough, and this is not possible using only addition, subtraction and multiplication (as follows from results of Section 4.3).

Instead, we consider a homogeneous polynomial $p(x)$ evaluated on a homogeneous $\mathcal{D}$, i.e. one where $x \in \mathcal{D}$ implies $\gamma x \in \mathcal{D}$ for any scalar γ. Even though such $\mathcal{D}$ are unbounded, homogeneity of p will let us consider just the behavior of $p(x)$ on $\mathcal{D}$ intersected with the unit ball S^{n-1} in $\mathbb{R}^n$ (or S^{2n-1} in $\mathbb{C}^n$). On this intersection we can use the same compactness argument as above:

Theorem 3.2. *Let $p(x)$ be a homogeneous polynomial, let $\mathcal{D}$ be a homogeneous domain, and let S denote the unit ball in $\mathbb{R}^n$ (or $\mathbb{C}^n$). Let*

$$p_{min,homo} := \inf_{x \in \bar{\mathcal{D}} \cap S} |p(x)|$$

Then $p(x)$ can be evaluated accurately if $p_{min,homo} > 0$.

4 Classical arithmetic

In this section we sketch the way in which we deal with the classical arithmetic case over the real or complex fields, with the three basic operations $\{+, -, \cdot\}$, to which we add negation. The model of arithmetic is governed by the laws in Section 2. We remind the reader that this arithmetic model *does not allow* the use of constants.

We will need the following definition of allowability.

Definition 4.1. *Let p be a polynomial over $\mathbb{R}^n$ or $\mathbb{C}^n$, with variety $V(p) := \{x \ : \ p(x) = 0\}$. We call $V(p)$* allowable *if it can be represented as a union of intersections of sets of the form*

$$1. \quad Z_i = \{x \ : \ x_i \ = \ 0\} \ , \tag{1}$$

$$2. \quad S_{ij} = \{x \ : \ x_i + x_j \ = \ 0\} \ , \tag{2}$$

$$3. \quad D_{ij} = \{x \ : \ x_i - x_j \ = \ 0\} \ . \tag{3}$$

If $V(p)$ is not allowable, we call it unallowable.

4.1 Necessity: real and complex

Definition 4.2. *Given a polynomial p over $\mathcal{S}$ with unallowable variety $V(p)$, consider all sets W that are finite intersections of allowable hyperplanes defined by (1), (2), (3), and subtract from $V(p)$ those W for which $W \subset V(p)$. We call the remaining subset of the variety* points in general position *and denote it by $G(p)$. Note that if $V(p)$ is not allowable, then $G(p) \neq \emptyset$.*

Definition 4.3. *Given $x \in \mathcal{S}$, define the set* Allow(x) *as the intersection of all allowable hyperplanes going through x:*

$$\text{Allow}(x) := (\cap_{x \in Z_i} Z_i) \cap \left(\cap_{x \in S_{ij}} S_{ij}\right) \cap \left(\cap_{x \in D_{ij}} D_{ij}\right) ,$$

with the understanding that

$$\text{Allow}(x) := \mathcal{S} \qquad \text{whenever} \qquad x \notin Z_i, \ S_{ij}, \ D_{ij} \quad \text{for all} \quad i, j.$$

Note that Allow(x) *is a linear subspace of S, and that for each $x \in G(p)$,*

$$\text{Allow}(x) \not\subseteq V(p) \ .$$

We can now state the main necessity theorem.

Theorem 4.4. *Let p be a polynomial over a domain $\mathcal{D} \in \mathcal{S}$. Let $G(p)$ be the set of points in general position on the variety $V(p)$. If there exists $x \in \mathcal{D} \cap G(p)$ such that* Allow$(x) \cap$ Int$(\mathcal{D}) \neq \emptyset$*, then p is not accurately evaluable on $\mathcal{D}$.*

Sketch of proof. The proof of this theorem relies on tracing the zeros produced by the algorithm back to the nodes where they originate.

For the non-branching case, we think of the algorithm as a directed acyclic graph (DAG) with input nodes, branching nodes, and output nodes. One of the key facts in the proof is that each node outputs a polynomial in x and the error variables δ, which, for a given x, will either be exactly 0 for all δ, or it will be non-zero for almost all δ.

Roughly speaking, if the algorithm produces a "true" zero (i.e. a zero which does not depend on the error variables δ), we show that this zero can be traced back on the DAG to allowable conditions (multiplication by a perfect 0, or addition/subtraction of equal source variables). Thus, if the algorithm produces a 0 at x, it will also produce a 0 when run on Allow(x), for any choice of error variables δ. This is enough to prove Theorem 4.4 in the non-branching case (if $x \in G(p)$, then either $p_{comp}(x, \delta) \neq 0$ for almost all δ, or $p_{comp}(y, \delta) = 0$ for all $y \in \text{Allow}(x) \setminus V(p)$ and for all δ).

The branching case is based on the non-branching one and it is slightly more complicated. It involves proving a refinement of the above argument, namely, that arbitrarily close to any point x in general position there are sets S of positive measure such that the relative accuracy of the algorithm when run with inputs in S is either 1 or ∞. □

4.2 Sufficiency: the complex case

Suppose we now restrict input values to be complex numbers and use the same algorithm types and the notion of accurate evaluability from the previous sections. By Theorem 4.4, for a polynomial p of n complex variables to be accurately evaluable over $\mathbb{C}^n$ it is necessary that its variety $V(p) := \{z \in \mathbb{C}^n : p(z) = 0\}$ be allowable.

The goal of this section is to prove that this condition is also sufficient, as stated in the following theorem.

Theorem 4.5. *Let $p : \mathbb{C}^n \to \mathbb{C}$ be a polynomial with integer coefficients and zero constant term. Then p is accurately evaluable on $\mathcal{D} = \mathbb{C}^n$ if and only if the variety $V(p)$ is allowable.*

With the help of a little algebraic geometry, we obtain the following Lemma.

Lemma 4.6. *If $p : \mathbb{C}^n \to \mathbb{C}$ is a polynomial whose variety $V(p)$ is allowable, then it is a product $p = c \prod_j p_j$, where each p_j is a power of x_i, $(x_i - x_j)$, or $(x_i + x_j)$.*

Theorem 4.5 follows from Lemma 4.6.

4.3 Toward a necessary and sufficient condition in the real case

We now show that accurate evaluability of a polynomial over $\mathbb{R}^n$ is ultimately related to accurate evaluability of its "dominant terms". These are the terms of the polynomial that dominate its other terms in a particular semialgebraic set close to a particular component of its variety; thus which terms will dominate depends on how we approach the variety of a polynomial.

For reasons outlined in Section 3, we consider here only homogeneous polynomials. Futhermore, most of this section focuses on non-branching algorithms, but we do need branching for our statements at the end of the section.

4.3.1 Dominance

Given a polynomial p with an allowable variety $V(p)$, we fix an irreducible component of $V(p)$. Any such component is described by linear allowable constraints. It turns out (see [4]) that any given component of $V(p)$ can be put into the form $x_1 = x_2 = ... = x_k = 0$ using what we call a *standard change of variables.* Standard changes of variables are simple linear transformations of the variables, which, however, have a rather involved combinatorial description, which we here omit.

After a suitable allowable change of variables, we can assume that the polynomial $p(x)$ is written as $p(x) = \sum_{\lambda \in \Lambda} c_\lambda x_{[1:k]}^\lambda q_\lambda(x_{[k+1:n]})$, where we write $x_{[1:k]} := (x_1, \ldots, x_k)$, $x_{[k+1:n]} := (x_{k+1}, \ldots, x_n)$. Also, we let Λ be the set of all multi-indices $\lambda := (\lambda_1, \ldots, \lambda_k)$ occuring in the monomials of $p(x)$.

To determine all dominant terms associated with the component $x_1 = x_2 = ... = x_k = 0$, consider the Newton polytope P of the polynomial p with respect to the variables x_1 through x_k only, i.e., the convex hull of the exponent vectors $\lambda \in \Lambda$ (see, e.g., [6, p. 71]). Next, consider the normal fan $N(P)$ of P (see [11, pp. 192–193]) consisting of the cones of all row vectors η whose dot products with $x \in P$ are maximal for x on a fixed face of P. That means that for every nonempty face F of P we take $N_F := \{\eta = (n_1, \ldots, n_k) \in (\mathbb{R}^k)^* : F \subseteq \{x \in P : \eta x (:= \sum_{j=1}^k n_j x_j) = \max_{y \in P} \eta y\}\}$ and $N(P) := \{N_F : F \text{ is a face of } P\}$.

Finally, consider the intersection of the negative of the normal fan $-N(P)$ and the nonnegative quadrant $\mathbb{R}^k_+$. This splits the first quadrant $\mathbb{R}^k_+$ into several regions S_{Λ_j} according to which subsets Λ_j of exponents λ "dominate" close to the considered component of the variety $V(p)$, in the following sense:

Definition 4.7. *Let Λ_j be a subset of Λ that determines a face of the Newton polytope P of p such that the negative of its normal cone $-N(P)$ intersects $(\mathbb{R}^k)^*_+$ nontrivially (not only at the origin). Define $S_{\Lambda_j} \in (\mathbb{R}^k)^*_+$ to be the set of all nonnegative row vectors η such that*

$$\eta\lambda_1 = \eta\lambda_2 < \eta\lambda, \quad \forall\lambda_1, \lambda_2 \in \Lambda_j, \quad \text{and } \lambda \in \Lambda \setminus \Lambda_j.$$

Let $F_{\Lambda_j} \subseteq [-1, 1]^k$ be the set of all points $x_{[1:k]} \in \mathbb{R}^k$ such that

$$\eta := (-\log|x_1|, \ldots, -\log|x_k|) \in S_{\Lambda_j}.$$

Example 4.8. *Consider the following polynomial*

$$p(x_1, x_2, x_3) = x_2^8 x_3^{12} + x_1^2 x_2^2 x_3^{16} + x_1^8 x_3^{12} + x_1^6 x_2^{14} + x_1^{10} x_2^6 x_3^4.$$

We show below the regions F_{Λ_j} near the component $x_1 = x_2$ of $V(p)$.

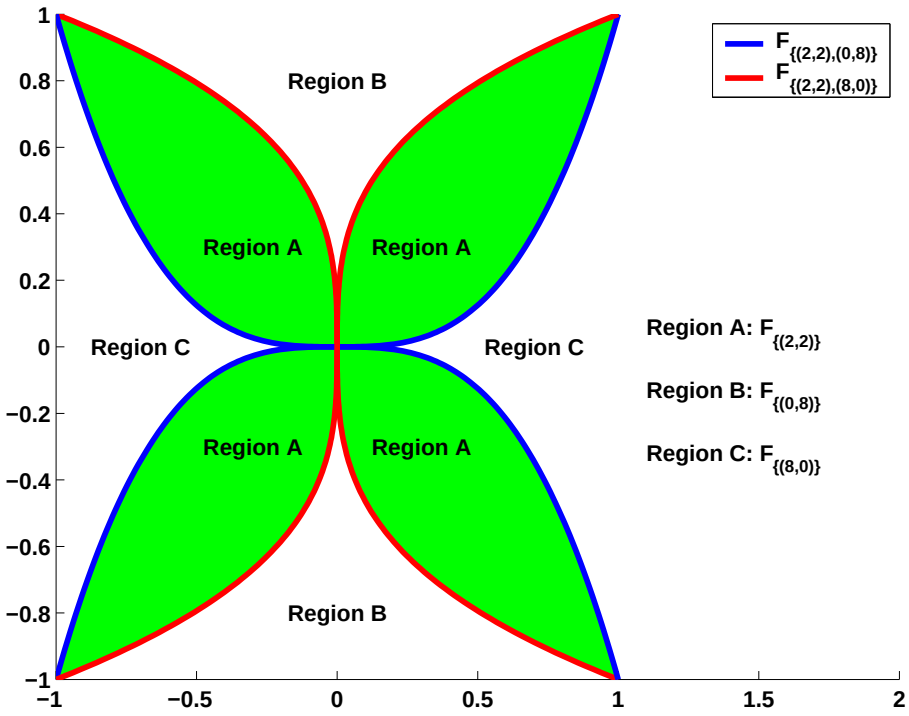

Figure 1. The regions F_{Λ_j}.

Definition 4.9. *We define the* dominant term *of $p(x)$ corresponding to the component $x_1 = \cdots = x_k = 0$ and the region F_{Λ_j} by*

$$p_{dom_j}(x) := \sum_{\lambda \in \Lambda_j} c_\lambda x^\lambda_{[1:k]} q_\lambda(x_{[k+1:n]}) \ .$$

We then prove that p_{dom_j} is the leading term along certain curves lying in the subset F_{Λ_j} as we approach 0. The next question is whether the dominant term p_{dom_j} indeed dominates the remaining terms of p in the region F_{Λ_j} in the sense that $p_{dom_j}(x)/p(x)$ is close to 1 sufficiently close to the component $x_1 = \cdots = x_k = 0$ of the variety $V(p)$. Indeed, we show that each dominant term p_{dom_j} such that the convex hull of Λ_j is a facet of the Newton polytope of p and whose variety $V(p_{dom_j})$ does not have a component strictly larger than the set $x_1 = \cdots = x_k = 0$ indeed dominates the remaining terms in p, not only in F_{Λ_j}, but in a certain "slice" $\widetilde{F}_{\Lambda_j}$ around F_{Λ_j}. These dominant terms, corresponding to larger sets Λ_j, are the useful ones, since they pick up terms relevant not only in the region F_{Λ_j} but also in its neighborhood.

Lemma 4.10. *Let p_{dom_j} be the dominant term of a homogeneous polynomial p corresponding to the component $x_1 = \cdots = x_k = 0$ of the variety $V(p)$ and to the set Λ_j whose convex hull is a facet of the Newton polytope N.*

Let $\widetilde{S}_{\Lambda_j}$ be any closed pointed cone in $(\mathbb{R}^k)^*_+$ with vertex at 0 that does not intersect other one-dimensional rays S_{Λ_l}, $l \neq j$, and contains $S_{\Lambda_j} \setminus \{0\}$ in its interior. Let $\widetilde{F}_{\Lambda_j}$ be the closure of the set

$$\{x_{[1:k]} \in [-1,1]^k : (-\log|x_1|, \ldots, -\log|x_k|) \in \widetilde{S}_{\Lambda_j}\}. \tag{4}$$

Suppose the variety $V(p_{dom_j})$ of p_{dom_j} is allowable and intersects $\widetilde{F}_{\Lambda_j}$ only at 0. Let $\|\cdot\|$ be any norm. Then, for any $\delta = \delta(j) > 0$, there exists $\varepsilon = \varepsilon(j) > 0$ such that

$$\left| \frac{p_{dom_j}(x_{[1:k]}, x_{[k+1:n]})}{p(x_{[1:k]}, x_{[k+1:n]})} - 1 \right| < \delta \quad \text{whenever} \quad \frac{\|x_{[1:k]}\|}{\|x_{[k+1:n]}\|} \leq \varepsilon \ \text{ and } \ x_{[1:k]} \in \widetilde{F}_{\Lambda_j}. \tag{5}$$

The above discussion of dominance was based on the transformation of a given irreducible component of the variety to the form $x_1 = \cdots = x_k = 0$. We must reiterate that the identification of dominant terms becomes possible only after a suitable change of variables C is used to put a given irreducible component into the standard form $x_1 = \cdots = x_k = 0$ and then the sets Λ_j are determined. Note however that the polynomial p_{dom_j} is given in terms of the original variables, i.e., as a sum of monomials in the original variables x_q and sums/differences $x_q \pm x_r$. We therefore use the more precise notation $p_{dom_j,C}$ in the rest of this section.

4.3.2 Pruning

We can convert an accurate algorithm that evaluates a polynomial p into an accurate algorithm that evaluates a selected dominant term $p_{dom_j,C}$. This process, which we will refer to as *pruning,* consists of deleting some vertices and edges and redirecting certain other edges in the DAG that represents the algorithm. Pruning allows us to track and extract leading terms as we approach a given branch of the variety $V(p)$ from within a set F_{Λ_j}. Here is an example intended to give an idea what is involved in the pruning process.

Example 4.11. *Figure 2 shows an example of pruning an algorithm that evaluates the polynomial*

$$x_1^2x_2^2 + (x_2 - x_3)^4 + (x_3 - x_4)^2x_5^2$$

using the substitution

$$(tx_1, x_2, tx_3 + x_2, tx_4 + x_2, x_5)$$

near the component

$$x_1 = 0, \;\; x_2 = x_3 = x_4.$$

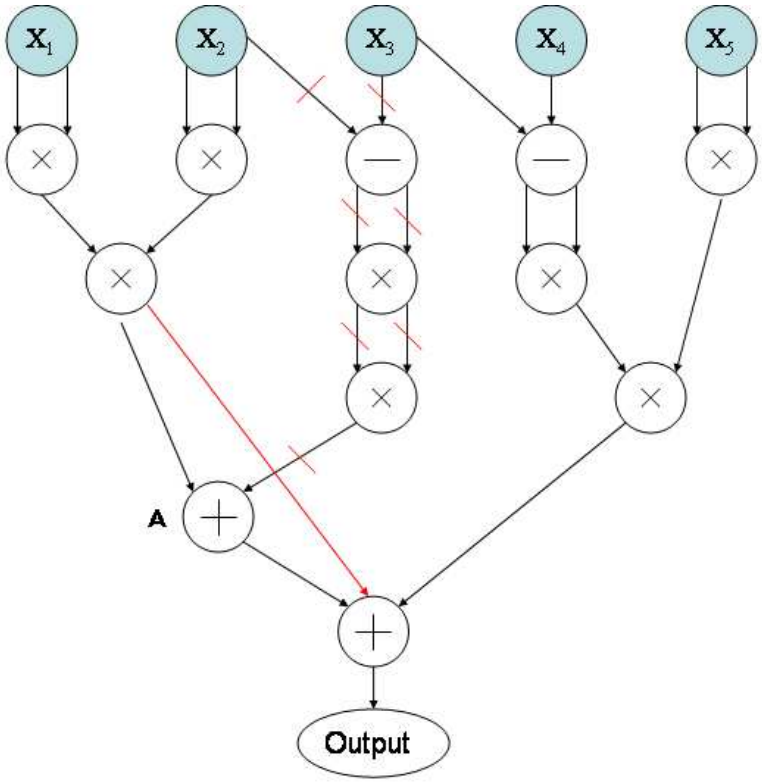

Figure 2. Pruning an algorithm for $p(x) = x_1^2x_2^2 + (x_2 - x_3)^4 + (x_3 - x_4)^2x_5^2$.

The result of pruning is an algorithm that evaluates the dominant term

$$x_1^2x_2^2 + (x_3 - x_4)^2x_5^2.$$

The output of the original algorithm is given by

$$\begin{aligned}\Big(\big(x_1^2(1+\delta_1)x_2^2(1+\delta_2)(1+\delta_3) + (x_2-x_3)^4(1+\delta_4)^4(1+\delta_5)^2(1+\delta_6)\big)(1+\delta_7)\\ +(x_3-x_4)^2(1+\delta_8)^2(1+\delta_9)x_5^2(1+\delta_{10})(1+\delta_{11})\Big)(1+\delta_{12}).\end{aligned}$$

The output of the pruned algorithm is

$$\big(x_1^2(1+\delta_1)x_2^2(1+\delta_2)(1+\delta_3))(1+\delta_7) + (x_3-x_4)^2(1+\delta_8)^2(1+\delta_9)x_5^2(1+\delta_{10})(1+\delta_{11})\big)(1+\delta_{12}).$$

The pruning process always produces an algorithm that accurately evaluates the corresponding dominant term.

Theorem 4.12. *Suppose a non-branching algorithm evaluates a polynomial p accurately on $\mathbb{R}^n$ by computing $p_{comp}(x, \delta)$. Suppose C is a standard change of variables associated with an irreducible component of $V(p)$. Let $p_{dom_j,C}$ be one of the corresponding dominant terms of p and let S_{Λ_j} satisfy some technical condition. Then the pruned algorithm with output $p_{dom_j,C,comp}(x, \delta)$ evaluates $p_{dom_j,C}$ accurately on $\mathbb{R}^n$. In other words, being able to compute all such $p_{dom_j,C}$ for all components of the variety $V(p)$ and all standard changes of variables C accurately is a condition necessary to compute p accurately.*

4.3.3 Sufficiency of evaluating dominant terms

Our next goal is to prove a converse of a sort to Theorem 4.12. Strictly speaking, our results do not provide a true converse, since branching is needed to construct an algorithm that evaluates a polynomial p accurately from algorithms that evaluate its dominant terms accurately.

We make two assumptions, viz., that our polynomial p is homogeneous and irreducible. The latter assumption effectively reduces the problem to that of accurate evaluation of a nonnegative polynomial, due to the following lemma.

Lemma 4.13. *If a polynomial p is irreducible and has an allowable variety $V(p)$, then it is either a constant multiple of a linear form that defines an allowable hyperplane or it does not change its sign in $\mathbb{R}^n$.*

From now on we therefore restrict ourselves to the nontrivial case when a (homogeneous and irreducible) polynomial p is nonnegative everywhere in $\mathbb{R}^n$.

Theorem 4.14. *Let p be a homogeneous nonnegative polynomial whose variety $V(p)$ is allowable. Suppose that all dominant terms $p_{dom_j,C}$ for all components of the variety $V(p)$, all standard changes of variables C, and all subsets Λ_j satisfying some technical condition are accurately evaluable. Then there exists a branching algorithm that evaluates p accurately over $\mathbb{R}^n$.*

Sketch of proof. We first show how to evaluate p accurately in a neighborhood of each irreducible component of its variety $V(p)$. We next evaluate p accurately off these neighborhoods of $V(p)$. The final algorithm will involve branching depending on which region the input belongs to, and the subsequent execution of the corresponding subroutine.

Consider a particular irreducible component V_0 of the variety $V(p)$. Using any standard change of variables C, we map V_0 to a set of the form $\widetilde{x}_1 = \cdots = \widetilde{x}_k = 0$. We create an ε-neighborhood of V_0 where we can evaluate p accurately. It is built up from semialgebraic ε-neighborhoods. More precisely, for each V_0, we can find a collection (S_j) of semialgebraic sets, all determined by polynomial inequalities with integer coefficients, and the corresponding numbers ε_j, so that the polynomial p can be evaluated with desired accuracy η in each ε_j-neighborhood of V_0 within the piece S_j. Moreover, testing whether a particular point x is within ε_j of V_0 within S_j can be done by branching based on polynomial inequalities with integer coefficients.

The final algorithm will be organized as follows. Given an input x, determine by branching whether x is in S_j and within the corresponding ε_j of a component V_0. If that is the case, evaluate $p(x)$ using the algorithm that is accurate in S_j in that neighborhood of V_0. For x not in any of the neighborhoods, evaluate p by Horner's rule. Since the polynomial p is strictly positive off the

neighborhoods of the components of its variety, the reasoning of Section 3 applies, showing that the Horner's rule algorithm is accurate. If x is on the boundary of a set S_j, any applicable algorithm will do, since the inequalities we use are not strict. Thus the resulting algorithm for evaluating p will have the desired accuracy η. □

4.3.4 Obstacles to full induction

Our results in the previous sections suggest that there could be an inductive decision procedure that would allow us to determine whether or not a given polynomial is accurately evaluable by reducing the problem for the original polynomial p to the same problem for its dominant terms, then their dominant terms, and so forth, going all the way to monomials or other polynomials that are easy to analyze. However, this idea would only work if the dominant terms were somehow "simpler" than the original polynomial itself, i.e., this would require an induction variable that would decrease at each step.

Two possible choices are the number of variables or the degree of the polynomial under consideration. Sometimes, however, neither of the two goes down, and moreover, the dominant term may even coincide with the polynomial itself. For example, if

$$p(x) = A(x_{[3:n]})x_1^2 + B(x_{[3:n]})x_1x_2 + C(x_{[3:n]})x_2^2$$

where A, B, C are nonnegative polynomials in x_3 through x_n, then the only useful dominant term of p in the neighborhood of the set $x_1 = x_2 = 0$ is the polynomial p itself. Thus no progress whatsoever is made in this situation.

Another possibility is induction on domains but we do not yet envision how to make this idea precise, since we do not know exactly when a given polynomial is accurately evaluable on a given domain. Further work to establish a full decision procedure is therefore highly desirable.

5 "Black-box" arithmetic

In this section we prove a necessary condition (for both the real and the complex cases) for a more general type of arithmetic, which allows for "black-box" polynomial operations. We describe the type of operations below.

Definition 5.1. *We call a black-box operation any type of operation that takes a number of inputs (real or complex) $x_1, \ldots, x_k$ and produces an output q such that q is a polynomial in $x_1, \ldots, x_k$.*

Example 5.2. $q(x_1, x_2, x_3) = x_1 + x_2x_3$.

Remark 5.3. *Note that $+, -$, and $\cdot$ are all black-box operations on two inputs.*

Consider a fixed set of multivariate polynomials $\{q_j : j \in J\}$ with real or complex inputs (this set may be infinite). In our model under consideration, the arithmetic operations allowed are given by the black-box operations $q_1, \ldots, q_k$, and negation. With the exception of negation, which is exact, all the others yield a $rnd(op(a_1, \ldots, a_l)) = op(a_1, \ldots, a_l)(1 + \delta)$, with $|\delta| < \epsilon$ (ϵ here is the machine precision). We consider the same arithmetical models as in Section 2, with this larger class of operations.

5.1 Necessity: real and complex

In order to see how the statement of the necessity Theorem 4.4 changes, we need to introduce a different notion of allowability. Recall that we denote by $\mathcal{S}$ the space of variables (which may be either $\mathbb{R}^n$ or $\mathbb{C}^n$). From now on we will denote the set $\{1, \ldots, n\}$ by $\mathcal{A}$.

Definition 5.4. *Let $p(x_1, \ldots, x_n)$ be a multivariate polynomial over $\mathcal{S}$ with variety $V(p)$. Let $\mathcal{A}_Z \subseteq \mathcal{A}$, and let $\mathcal{A}_D, \mathcal{A}_S \subseteq \mathcal{A} \times \mathcal{A}$. Modify p as follows: impose conditions of the type Z_i for each $i \in \mathcal{A}_Z$, and of type D_{ij}, respectively S_{ij}, on all pairs of variables in $\mathcal{A}_D$, respectively $\mathcal{A}_S$. Rewrite p subject to those conditions (e.g. set $X_i = 0$ for all $i \in \mathcal{A}_Z$), and denote it by $\tilde{p}$, and denote by $\mathcal{A}_R$ the set of remaining independent variables (use the convention which eliminates the second variable in each pair in $\mathcal{A}_D$ or $\mathcal{A}_S$).*

Choose a set $T \subseteq \mathcal{A}_R$, and let

$$V_{T,\mathcal{A}_Z,\mathcal{A}_D,\mathcal{A}_S}(p) = \cap_\alpha V(q_\alpha) \ ,$$

where the polynomials q_α are the coefficients of the expansion of $\tilde{p}$ in the variables x_T:

$$\tilde{p}(x_1, \ldots, x_k) = \sum_\alpha q_\alpha x_T^\alpha \ ,$$

with q_α being polynomials in $x_{\mathcal{A}_R \setminus T}$ only.

Finally, let $\mathcal{A}_N$ be a subset of $\mathcal{A}_R \setminus T$. We negate each variable in $\mathcal{A}_N$, and let $V_{T,\mathcal{A}_Z,\mathcal{A}_D,\mathcal{A}_S,\mathcal{A}_N}(p)$ be the variety obtained from $V_{T,\mathcal{A}_Z,\mathcal{A}_D,\mathcal{A}_S}(p)$, with each variable in $\mathcal{A}_N$ negated.

Remark 5.5. *$V_{\emptyset,\emptyset,\emptyset,\emptyset,\emptyset}(p) = V(p)$. We also note that, if we have a black-box computing p, then the set of all polynomials $\tilde{p}$ that can be obtained from p by permuting, repeating, and negating the variables (as in the definition above) is* exactly *the set of all polynomials that can be evaluated with a single rounding error, using that black box.*

Definition 5.6. *For simplicity, we denote a set $(T, \mathcal{A}_Z, \mathcal{A}_D, \mathcal{A}_S, \mathcal{A}_N)$ by $\mathcal{I}$, and a set $(T, \mathcal{A}_Z, \mathcal{A}_D, \mathcal{A}_S)$ by $\mathcal{I}_+$.*

Definition 5.7. *We define $q_{-2}(x_1, x_2) = x_1 x_2$, $q_{-1}(x_1, x_2) = x_1 + x_2$, and $q_0(x_1, x_2) = x_1 - x_2$.*

Remark 5.8. *The sets*

$$1. \quad Z_i = \{x \ : \ x_i \ = \ 0\} \ , \tag{6}$$

$$2. \quad S_{ij} = \{x \ : \ x_i + x_j \ = \ 0\} \ , \tag{7}$$

$$3. \quad D_{ij} = \{x \ : \ x_i - x_j \ = \ 0\} \tag{8}$$

describe all non-trivial (neither $\emptyset$ nor $\mathcal{S}$) sets of type $V_\mathcal{I}$, for q_{-2}, q_{-1}, and q_0.

We will assume from now on that the black-box operations q_{-2}, q_{-1}, q_0 defined in 5.7, and some arbitrary extra operations q_j, with $j \in J$ (J may be infinite) are given and fixed.

Definition 5.9. *We call any set $V_\mathcal{I}(q_j)$ with $\mathcal{I} = (T, \mathcal{A}_Z, \mathcal{A}_D, \mathcal{A}_S, \mathcal{A}_N)$ as defined above and q_j a black-box operation* basic q-allowable.

We call any set R irreducible q-allowable *if it is an irreducible component of a (finite) intersection of basic q-allowable sets, i.e., when R is irreducible and*

$$R \subseteq \cap_l \; Q_l \ ,$$

where each Q_l is a basic q-allowable set.

We call any set Q q-allowable *if it is a (finite) union of irreducible q-allowable sets, i.e.*

$$Q = \cup_j R_j \ ,$$

where each R_j is an irreducible q-allowable set.

Any set R which is not q-allowable we call q-unallowable.

Remark 5.10. *Note that the above definition of q-allowability is closed under taking union, intersection, and irreducible components. This parallels the definition of allowability for the classical arithmetic case – in the classical case, every allowable set was already irreducible (being an intersection of hyperplanes).*

Definition 5.11. *Given a polynomial p with q-unallowable variety $V(p)$, consider all sets W that are q-allowable (as in Definition 5.9), and subtract from $V(p)$ those W for which $W \subset V(p)$. We call the remaining subset of the variety* points in general position *and denote it by $\mathcal{G}(p)$.*

Remark 5.12. *Since $V(p)$ is q-unallowable, $\mathcal{G}(p)$ is non-empty.*

Definition 5.13. *Given $x \in \mathcal{S}$, define the set* $\mathrm{q{-}Allow}(x)$ *as the intersection of all basic q-allowable sets going through x:*

$$\mathrm{q{-}Allow}(x) := \cap_{j \in J \cup \{-2,-1,0\}} \left(\cap_{I \; : \; x \in V_{\mathcal{I}}(q_j)} \; V_{\mathcal{I}}(q_j) \right) ,$$

for all possible choices of $T, \mathcal{A}_Z, \mathcal{A}_D, \mathcal{A}_S, \mathcal{A}_N$.

The intersection in parentheses is $\mathcal{S}$ whenever $x \notin V_{\mathcal{I}}(q_j)$ for all possible $\mathcal{I}$.

Remark 5.14. *When $x \in \mathcal{G}(p)$,* $\mathrm{q{-}Allow}(x) \not\subseteq \mathcal{G}(p)$.

We can now state our necessity condition.

Theorem 5.15. *Given the black-box operations q_{-2}, q_{-1}, q_0, and $\{q_j \; : \; j \in J\}$, and the model of arithmetic described above, let p be a polynomial defined over a domain $\mathcal{D} \subset \mathcal{S}$. Let $\mathcal{G}(p)$ be the set of points in general position on the variety $V(p)$. If there exists $x \in \mathcal{D} \cap \mathcal{G}(p)$ such that* $\mathrm{q{-}Allow}(x) \cap \mathrm{Int}(\mathcal{D}) \neq \emptyset$, *then p is not accurately evaluable on $\mathcal{D}$.*

Sketch of proof. The proof mimics the proof of Theorem 4.4; once again, we trace back zeros to what we now call q-allowable conditions, and make use of the DAG structure of the algorithm. In the non-branching case, we obtain that if the algorithmis run on a point $x \in G(p)$, then either $p_{comp}(x, \delta) \neq 0$ for almost all δ, or $p_{comp}(y, \delta) = 0$ for all $y \in \mathrm{Allow}(x) \setminus V(p)$ and for all δ.

The proof for the branching case is again a refinement of the proof for the non-branching one, and we show that, arbitrarily close to any point $x \in \mathcal{G}(p)$, we can find sets S of positive measure such that the relative accuracy of the algorithm when run with inputs in S is either 1 or ∞. $\square$

5.2 Sufficiency: the complex case

In this section we obtain a sufficiency condition for the accurate evaluability of a complex polynomial, given a black-box arithmetic with operations q_{-2}, q_{-1}, q_0 and $\{q_j | j \in J\}$ (J may be an infinite set).

Throughout this section, we assume our black-box operations include q^c, which consists of multiplication by a complex constant: $q^c(x) = c \cdot x$. Note that this operation is natural, and that most computers perform it with relative accuracy.

We believe that the sufficiency condition we obtain here is sub-optimal in general, but it subsumes the sufficiency condition we found for the basic complex case with classical arithmetic $\{+, -, \cdot\}$.

We state here the best sufficiency condition for the accurate evaluability of a polynomial we were able to find in the general case, and a necessary and sufficient condition for the all-affine black-box operations case.

Theorem 5.16 (General case). *Given a polynomial $p : \mathbb{C}^n \to \mathbb{C}$ with $V(p)$ a finite union of intersections of hyperplanes Z_i, S_{ij}, D_{ij}, and varieties $V(q_j)$, for $j \in J$, then p is accurately evaluable.*

Theorem 5.17 (Affine case). *If all black-box operations q_j, $j \in J$ are affine, then a polynomial $p : \mathbb{C}^n \to \mathbb{C}$ is accurately evaluable iff $V(p)$ is a union of intersections of hyperplanes Z_i, S_{ij}, D_{ij}, and varieties $V_{\mathcal{I}}(q_j)$, for $j \in J$ and $\mathcal{I}$ as in Definition 5.4.*

The proofs follow easily from Lemma 5.18.

Lemma 5.18. *If $p : \mathbb{C}^n \to \mathbb{C}$ is a polynomial whose variety $V(p)$ is q-allowable, then it is a product $p = c \prod_j p_j$, where each p_j is a power of x_i, $(x_i - x_j)$, $(x_i + x_j)$, or q_j, and c is a complex constant.*

Remark 5.19. *Note that Theorem 5.17 is a more general necessary and sufficient condition than Theorem 4.5, which only considered having q_{-2}, q_{-1}, and q_0 as operations, and restricted the polynomials to have integer coefficients (thus eliminating the need for q^c).*

6 Open Problems

Building on the results obtained in [4] and described here, we would like to solve the following problems.

1. Complete the decision procedure outlined here, when the domain of evaluation $\mathcal{D} = \mathbb{R}^n$ or $\mathbb{C}^n$, initially for classical arithmetic and then for black-box arithmetic. We would like to implement this decision procedure in a practical way, to provide a "compiler" that will either produce an accurate algorithm for an input expression, or prove that one does not exist, or provide the smallest set of black-boxes that would make it accurately evaluable.

2. Extend these results to more general semialgebraic domains $\mathcal{D}$. It would be natural to consider only those $\mathcal{D}$ whose boundaries are allowable, so that membership in $\mathcal{D}$ is also decidable.

3. Apply these results to identify more structured matrix classes for which accurate linear algebra algorithms exist.

4. Incorporate division and rational functions in our analysis.

5. Understand the relationship of perturbation theory to accurate evaluability. For example, the problems evaluable in classical arithmetic so far seem to share a common perturbation theory, that the condition number grows proportionally to the reciprocal of the distance to the smallest problem with an infinite condition number [3].

6. Interval arithmetic [1] represents numbers by intervals, and does arithmetic with them by rounding the endpoints "outward" so as to provably include the true answer. It is natural to ask whether accurate evaluability of $p(x)$ in our sense is related to the existence of interval algorithms that provide analogously narrow intervals when evaluating $p(x)$.

7. Ultimately we want to understand the bit-complexity of floating point computation, for which any real model can only give hints. For example, our model shows that the determinant of a matrix with independent entries can only be evaluated accurately if the determinant itself is one of our black-box operations. Thus we are led to suspect determinant evaluation of unstructured floating point matrices to have high complexity, in contrast to the case where the entries are rational, or have bounded exponents [2].

References

[1] G. Alefeld and J. Herzberger. *Introduction to interval computations*. Academic Press, 1983.

[2] K. Clarkson. Safe and effective determinant evaluation. In *33rd Annual Symp. on Foundations of Comp. Sci.*, pages 387–395, 1992.

[3] J. Demmel. On condition numbers and the distance to the nearest ill-posed problem. *Num. Math.*, 51(3):251–289, 1987.

[4] J. Demmel, I. Dumitriu, and O. Holtz. Toward accurate polynomial evaluation in rounded arithmetic. To appear in Found. Comput. Math.

[5] J. Demmel and P. Koev. Accurate and efficient algorithms for floating point computation. In *Proceedings of the 2003 International Congress of Industrial and Applied Mathematics*, Sydney, 2004.

[6] E. Miller and B. Sturmfels. *Combinatorial commutative algebra*. Springer-Verlag, 2005.

[7] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals: Parts I, II and III. *J. Symb. Comp.*, 13, 1992.

[8] B. Reznick. *Some concrete aspects of Hilbert's 17th problem*, volume 253 of *Contemporary Mathematics*. Amer. Math. Society, 2000.

[9] J. R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18:305–363, 1997.

[10] A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, Berkeley, 1951.

[11] G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.