On the Power of Discontinuous Approximate Computations [1]

Karl Aberer

International Computer Science Institute
1947 Center Street, Suite 600, Berkeley, CA 94704
email: aberer@icsi.berkeley.edu

Bruno Codenotti

Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche, Pisa, Italy
email: codenotti@iei.pi.cnr.it

Abstract. Comparision operations are used in algebraic computations to avoid degeneracies, but are also used in numerical computations to avoid huge roundoff errors. On the other hand, the classes of algorithms using only arithmetic operations are the most studied in complexity theory, and are used, e.g., to obtain fast parallel algorithms for numerical problems. In this paper, we study, by using a simulation argument, the relative power of different sets of operations for computing with approximations. We prove that comparisions can be simulated efficiently and with the same error bounds for most inputs by arithmetic operations when divisions are present. To develop our simulation strategy we combine notions imported from approximation theory and topology with complexity and error bounds.

1 Introduction

In this paper we show that approximate computations over the reals do not suffer very much from the lack of conditional statements, provided that division is allowed. On the other hand we show that the set of operations without division is very poor, if one wants to achieve the goal of keeping the roundoff error small.

The problem studied in this paper has relations with several different research fields, and we attack it by using techniques and ideas related , e.g., to algebraic complexity tools used to remove degeneracies [EC91], topology notions used by numerical analysts to evaluate the distance of a path in a computation from degenerate (ill-conditioned) regions [SS91], and techniques, like arithmetization and simulation, used in complexity to find lower bounds on the size of a circuit [BF91, C91].

To the best of our knowledge, this is the first nontrivial result on the relative power of different sets of operations when roundoff errors are present. Previous work on the complexity of numerical problems can be divided into three categories: complexity results for obtaining ε-approximations without roundoff [CF89, CL91, MST89, TWW88, PR85], complexity bounds obtained in terms of problem conditioning [SS91, S90], and development of theories of computation over the reals [BSS89]. In this paper, we embed the topological notions needed to evaluate the effect of problem conditioning and the approximation notions needed to find bounds on the degree of the rational functions (or polynomials) involved in the simulation process onto the framework of the classical circuit complexity measures (size, depth, width).

The set of operations $S_1 = \{+, -, *, /, >\}$ is used in algebraic computations to avoid degeneracies (i.e. division by zero), but is also used in numerical computations to avoid huge roundoff errors. On the other hand, the class of algorithms using operations from the set $S_2 = \{+, -, *, /\}$ or from the set $S_3 = \{+, -, *\}$ is extremely important, and is the one actually used, e.g., to obtain fast parallel algorithms for numerical problems. In this paper, we study, by using a simulation argument, the relative power of the sets S_1, S_2, and S_3. We prove that S_2 does very efficiently simulate S_1, while S_3 does not; this fact shows and measures the crucial role of division in computations introducing roundoff errors. We also show how to construct algorithms using operations $\{+, -, *, /\}$ which achieve for most inputs the same bounds on the roundoff error as algorithms using operations $\{+, -, *, /, >\}$.

Here we adopt a model of computation which essentially consists of a "layered version" of an arithmetic network [G86], which is simpler to analyze. Arithmetic network and branching

[1] The work of Bruno Codenotti was partially supported by the Italian National Research Council under the "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 2" and was done while he was at the International Computer Science Institute, Berkeley.

programs are generally analyzed (see, e.g., [S90]) to measure the extra cost necessary to avoid degeneracies, e.g., divisions by zero. As already mentioned, our goals are more general, since we view branching instructions (conditionals) as a tool to avoid huge errors in a computation, and we study the conditions under which adaptive computations, i.e. those allowing branching instructions, can be simulated by oblivious ones, i.e. straight-line programs.

The main contributions of this paper are expressed by the results of section 3. The parameters we simultaneously deal with are the length of the input n, the degree of the approximating functions m, the maximum value M an intermediate result can achieve, the local approximation error ε_{loc}, the global approximation error ε_{glob}, the number k of polynomials defining degenerate regions, which we call *decision polynomials*, the maximum degree d of the above polynomials, and the quantity δ which measures the extension of the degenerate regions. The basic bound, that allows us to find several tradeoffs among the above parameters, is shown in Lemma 8, where it is proved that

$$f(V_\delta) \leq 12knd M e^{-\sqrt{m}} \varepsilon_{loc}^{-1},$$

where $f(V_\delta)$ denotes the intersection of the unit ball with the volume of the degenerate regions. Using the above bound together with other results of section 3, we prove that algorithms using operations from the set S_2 can carry out a simulation by using, under reasonable conditions, rational functions of degree at most $O(\log^2 n)$. On the contrary, we show that algorithms using operations in S_3 require the use of polynomials of degree at least $n^{O(1)}$ in the input size. This gives a measure of the importance of division.

This paper is organized as follows. Section 2 contains the description of the models of computation adopted, the definition of adversary which is used to find bounds on the error, some preliminary results on the error introduced by oblivious computations and some theorems which we import from approximation theory and topology. Section 3 contains the description of our simulation strategy, which gives upper and lower bounds on the cost of avoiding conditionals. In section 4 we report some concluding remarks and open problems.

2 Preliminaries

2.1 Models of Computation

The models of computations adopted here are a slight modification of arithmetic circuits and networks, and branching programs. More precisely, we adopt models which are a *structured* version of the above models, which turn out to be particularly useful for a detailed analysis (see [AC92]). Before describing the models of computation, it is convenient to define which kind of algorithms we consider.

An algorithm using operations $\{+, -, *, /\}$ is called *oblivious rational*. An algorithm using operations $\{+, -, *\}$ is called *oblivious polynomial*. An algorithm using operations $\{+, -, *, /, >\}$ is called *adaptive rational*. An algorithm using operations $\{+, -, *, >\}$ is called *adaptive polynomial*.

The simplest model consists of an arithmetic circuit, with gates $\{+, -, *, /\}$ or $\{+, -, *\}$ fed by a routing network, with gates $\{>\}$ (see Figure 1a). The input data are fed to the routing network which can compute any permutation of these data, and then provide the arithmetic circuit with the chosen permutation. The routing network is a network of comparators (see Figure 1b). We call *arithmetic stage* the part of the computation carried out by the arithmetic circuit, and *combinatorial stage* the part of the computation carried out by the routing network. A slight modification of this model consists of a cascade of "modules" of the previous kind, as shown in Figure 1c. In this case adaptivity can also be used to decide upon partial results. The relations of this model to arithmetic network are obvious (see [G86]). The cost of a computation can be analyzed by using the natural measures of circuit complexity (size, depth, and width). We will also analyze the error that is introduced in the arithmetic part by means of a "layered"

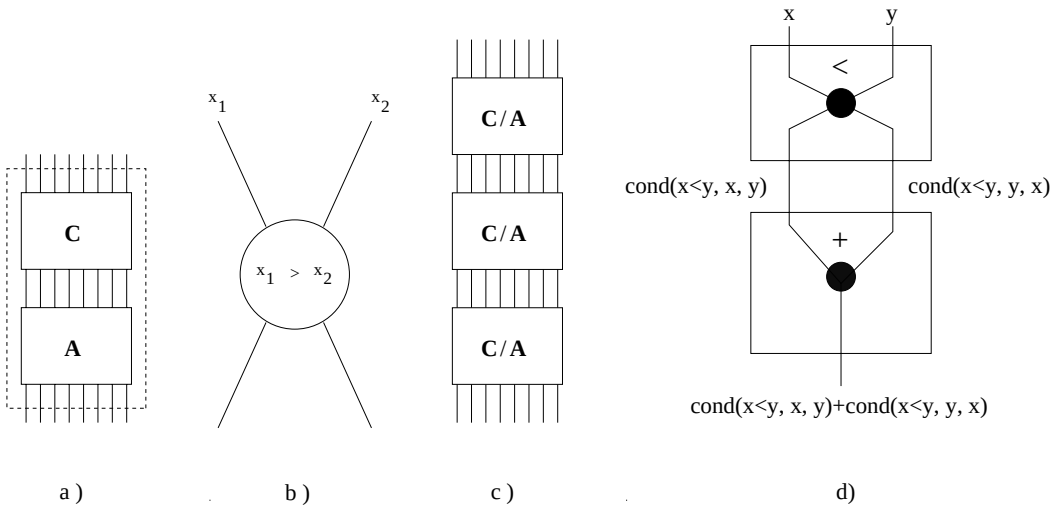

Figure 1: a) A combinatory (C) and arithmetic (A) stage.
b) The combinatory stage consists of comparators.
c) A cascade of modules. C/A is a module of the type introduced in a).
d) Translating a circuit into an expression.

error analysis, in which for any level i of the circuit, we evaluate the maximum attainable error (see section 2).

It is easy to see that any arithmetic circuit corresponds in a straightforward way to an arithmetic expression. On the other hand, by taking into account parentheses, an arithmetic expression uniquely defines a computation, which can be easily interpreted as the evaluation of an arithmetic circuit. To describe a routing network as an expression, it is sufficient to introduce *conditional functions* of the form

$$\mathrm{cond}(P, e_1, e_2) = \begin{cases} e_1, \text{ if } P \text{ true}, \\ e_2, \text{ otherwise}, \end{cases} \tag{1}$$

where e_1 and e_2 are expressions built up from arithmetic and conditional functions and P is a primitive predicate depending on one or more input data, e.g., $P(x) \equiv (x > 0)$ or $P(x, y) \equiv (x > y)$.

A numerical computation (consisting of one combinatorial and one arithmetic stage) taking k input values and computing m output values, is described by the expressions $e_1(x_1, \ldots, x_k), \ldots, e_m(x_1, \ldots, x_k)$. The combinatorial stage consists of evaluating all conditional functions according to the inputs and the arithmetic stage consists of evaluating all arithmetic functions according to the input values. Several combinatorial/arithmetic stages can be combined by composing the corresponding expressions. The relation of this model to the circuit-model is illustrated in Figure 1d.

In this model, the cost of a computation can be evaluated according to the maximum depth of all the expressions in the stage, the maximum number of different subexpressions at one level, and the overall size of the expressions.

2.2 Errors and Adversaries

It is very difficult to express lower bounds on the error an algorithm produces, because the notion of error is heavily instant-dependent. Typically, a lower bound which has to hold for all instances should be zero. Therefore we adopt the following strategy, which is based on an adversary argument (a similar approach can be found in [CLR91], or [TWW88]). In other words we assume an adversary can decide about the input distribution. This leads to the following definition of *unavoidable roundoff error*.

Definition 1 *Let $\mathcal{A}$ be a set of oblivious algorithms for solving a problem, I the set of problem instances, and $\varepsilon_{A(i)}$ the absolute value of the error occurring by executing the algorithm $A \in \mathcal{A}$ on the input $i \in I$. Then the roundoff error for oblivious and adaptive algorithms can be expressed as* $\min_{A\in\mathcal{A}}(\max_{i\in I}\varepsilon_{A(i)})$, *and* $\max_{i\in I}(\min_{A\in\mathcal{A}}\varepsilon_{A(i)})$, *respectively.*

Theorem 1 $\min_{A\in\mathcal{A}}(\max_{i\in I}\varepsilon_{A(i)}) \geq \max_{i\in I}(\min_{A\in\mathcal{A}}\varepsilon_{A(i)})$.

We now define a function which quantifies the relative power of adaptive and oblivious algorithms, with respect to the error they introduce.

Definition 2 *The quantity* $\log \dfrac{\max_{i\in I}(\min_{A\in\mathcal{A}}\varepsilon_{A(i)})}{\min_{A\in\mathcal{A}}(\max_{i\in I}\varepsilon_{A(i)})}$ *is called the pay-off function.*

2.3 Roundoff Errors

A numerical process gives rise to roundoff errors. These errors depend on the system chosen to represent the real numbers. Since our goal is to analyze roundoff errors, we do not take into account the question of under/overflow, and we may assume that a countable subset R of the real numbers, e.g., floating point numbers with arbitrary integer exponents, rational numbers or fixed point numbers of arbitrary size, is given. Given $R \subseteq \mathbb{R}$ we describe the roundoff error by means of a rounding function $r : \mathbb{R} \to R$ with the properties

$$r(x) = x + \varepsilon_R(x),\ |\varepsilon_R(x)| \leq h, \tag{2}$$

where h is a constant depending only on R. Such a rounding function exists, for the representation systems mentioned above. In the models of computation of section 2.1, the arithmetic stages were essentially straight-line programs built up in terms of primitive functions from a set $F = \{f_1, \ldots, f_k\}$. The roundoff errors lead to local errors in the evaluation of these primitive functions. We assume that the primitive functions satisfy a condition of the form

$$f_i(x_1 + \varepsilon_1, \ldots, x_n + \varepsilon_n) = f_i(x_1, \ldots, x_n) + \varepsilon_1\alpha_1(x_1, \ldots, x_n) + \ldots + \varepsilon_n\alpha_n(x_1, \ldots, x_n). \tag{3}$$

Note that for differentiable primitive functions, as for example addition and multiplication, such a property is naturally satisfied. A numerical operation is then described by the function

$$f_i^r(x_1, \ldots, x_n) := r(f_i(x_1, \ldots, x_n)),\ x_1, \ldots, x_n \in R,\ i = 1, \ldots, k.$$

2.4 Layered Error Analysis

In this section, we prove some bounds on the error introduced by oblivious algorithms. We will use these bounds in section 3.

Lemma 1 *Let f be a composition of primitive functions. Given a computation graph of depth t associated to the function f, the "roundoff error" ε_t can be evaluated according to the formula:*

$$\begin{aligned} \varepsilon_i &= \alpha_i\varepsilon_{i-1} + \varepsilon_{R_i},\ i = 1, \ldots, t,\ |\varepsilon_{R_i}| \leq h, \\ \varepsilon_0 &= \begin{cases} \varepsilon_{R_0}, & \text{if we include the perturbation error,} \\ 0, & \text{otherwise,} \end{cases} \end{aligned}$$

where the α_i, $i = 1, \ldots, t$, are quantities depending on a primitive function and partial results.

Proof. We follow one path in the computation graph from the input to the output nodes. Let t be the length of the path, and let v_{j_i}, $i = 1, \ldots, t$, be the sequence of values computed along this path. Then the error is evaluated using (2) and (3)

$$\begin{aligned} \varepsilon_i &= f_i^r(x_1 + e_1, \ldots, v_{j_{i-1}} + \varepsilon_{j_{i-1}}, \ldots, x_n + e_n) - f_i(x_1, \ldots, v_{j_{i-1}} \ldots, x_{n_i}) \\ &= e_1\alpha_1 + \ldots + \varepsilon_{j_{i-1}}\alpha_{j_i} + \ldots + e_{n_i}\alpha_{n_i} + \varepsilon_{R_i} = \varepsilon_{j_{i-1}}(\alpha_{j_i} + \frac{e_1}{\varepsilon_{j_{i-1}}}\alpha_1 + \ldots + \frac{e_{n_i}}{\varepsilon_{j_{i-1}}}\alpha_{n_i}) + \varepsilon_{R_i}. \end{aligned}$$

∎

Corollary 1 *Under the assumptions of lemma 1, if $\varepsilon_0 = 0$, then*

$$\varepsilon_t = \sum_{i=1}^{t} \varepsilon_{R_i} \prod_{j=i+1}^{t} \alpha_j, \quad \text{and} \quad |\varepsilon_t| \leq h \sum_{i=1}^{t} \prod_{j=i+1}^{t} |\alpha_j|.$$

2.5 Bounds from Approximation Theory and Topology

In section 3 we will make use of the following lemmas, which we import from approximation theory ([R69, D88]) and topology ([G82, W39]). The first three lemmas provide bounds on the best rational and polynomial approximation to the function $|x|$.

Lemma 2 ([N64]) *Let $a(x) = |x|$ and $r(x)$ be a best-approximating rational function of degree m. Then we have*

$$\frac{e^{-9\sqrt{m}}}{2} < \max_{x \in [-1,1]} |a(x) - r(x)| \leq \begin{cases} 3e^{-\sqrt{m}}, & m \text{ even} \\ 3e^{-\sqrt{m-1}}, & m \text{ odd.} \end{cases}$$

Lemma 3 *A function $r(x)$ achieving the upper bound of lemma 2 is*

$$r(x) = \frac{x(p(x) - p(-x))}{p(x) + p(-x)}, \text{ where } p(x) = \prod_{j=0}^{m-1} (x + t^j), \ t = e^{\frac{-1}{\sqrt{m}}}.$$

Proof. (Hint.) Follows from [R69] page 128. ∎

Lemma 4 *Let $p(x)$ be a polynomial of degree m and $a(x) = |x|$, and let K be a constant. Then we have*

$$\frac{K}{m} < \max_{x \in [-1,1]} |a(x) - p(x)| \leq \frac{2}{(2m+1)\pi}.$$

Proof. (Hint.) The upper bound can be obtained by using Tchebycheff or Legendre polynomials. The lower bound follows from [L52]. ∎

We now report a result on the volumes of tubes from [O85], which we will need to estimate the volumes of the *degenerate regions* created by the approximate simulation process (see section 3).

Theorem 2 *Suppose M is a real, purely d-dimensional variety in $\mathbb{R}^n$. Suppose further that M is the complete intersection of the polynomials $p_1, \ldots, p_{n-d}$. Let $D = \max_{1 \leq i \leq n-d} deg(p_i)$, and $f(V_\delta)$ be the fraction of the volume of the unit ball in $\mathbb{R}^n$ which is within distance $\delta \leq 1$ of M. Then*

$$f(V_\delta) \leq 2(n-d) \sum_{k=n-d}^{n} \binom{n}{k} (2D\delta)^k.$$

3 Main Results

This section contains our main results. We analyze adaptive computations for which the payoff function does not tend to zero, and we show a simulation procedure which allows us to investigate the relative power of different sets of operations.

We first show how to simulate the expressions $\text{cond}(x > 0, y, z)$ by rational functions, and analyze the cost of this process in terms of the degree of the rational functions (i.e. the maximum degrees of the denominator and numerator). The goal of this simulation is to find oblivious algorithms behaving like adaptive ones, for most inputs. We associate to the decisions corresponding to adaptive algorithms suitable semi-algebraic varieties which describe singularities. Analogously, we introduce semi-algebraic varieties which describe the degenerate regions "close"

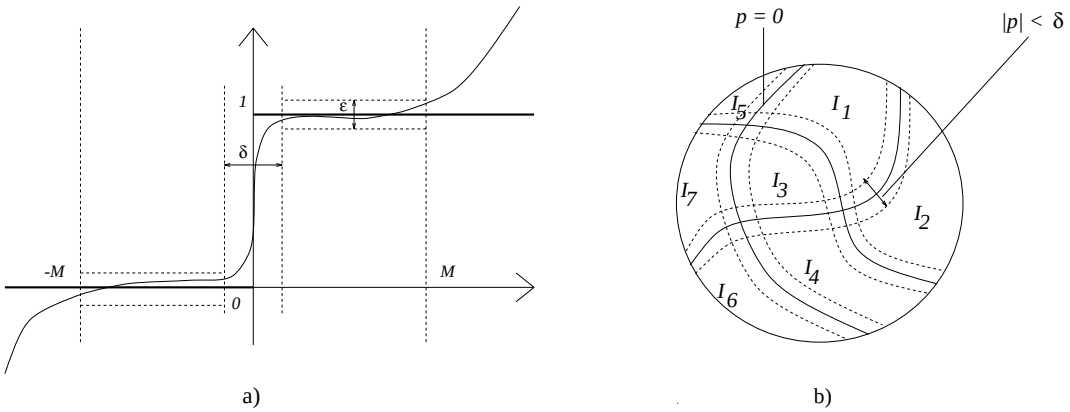

Figure 2: a) Approximation of the step-function.
b) The critical regions.

to singularities. These regions correspond to the inputs for which the approximate simulation cannot take place. We then prove lower and upper bounds on the volumes of such regions, and from these bounds, we are able to estimate the fraction of inputs for which we can approximately simulate adaptive algorithms. We also extend the above results taking care of the effects of roundoff errors, and by using estimates of the error introduced by oblivious algorithms. It turns out that the simulation is feasible at a very reasonable price, provided that it involves rational functions. The restriction to polynomial computations is proved to be very expensive.

3.1 Simulation of Conditionals

In section 2 we introduced the notation $\mathrm{cond}(a > 0, 1, 0)$ to denote expressions like *if* $a > 0$ *then* 1 *else* 0. We replace the expression $\mathrm{cond}(a > 0, 1, 0)$ by an arithmetic circuit (oblivious algorithm) simulating it. From this simulation - to be described below - we immediately get, by arithmetization, a simulation of a general conditional $\mathrm{cond}(x > 0, y, z)$. In fact

$$\mathrm{cond}(x > 0, y, z) = \mathrm{cond}(x > 0, 1, 0) * y - (\mathrm{cond}(x > 0, 1, 0) - 1) * z.$$

Definition 3 *A function* $f : \mathbb{R} \to \mathbb{R}$ *simulates cond*$(a > 0, 1, 0)$ *iff we have*

$$|f(x) - s(x)| < \varepsilon, \ \forall x \in [-M, -\delta) \cup (\delta, M],$$

where $s(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$ *, and* $M > \delta > 0$ *(see figure 2a).*

We use lemmas 2 and 4 to approximate the function $s(x)$. We do this by observing that,

$$s(x) = \frac{x - |x|}{2x}, \text{ for } x \neq 0.$$

Lemma 5 *Let* $M > 0$. *Then there exists a rational function* $r(x)$ *such that*

$$M e^{-9\sqrt{m}} \leq max_{x \in (-M,M)} |r(x) - |x|| \leq M\gamma_m, \ \gamma_m = \begin{cases} 3e^{-\sqrt{m}}, & m \text{ even} \\ 3e^{-\sqrt{m-1}}, & m \text{ odd} \end{cases}$$

Proof. (Sketch.) Follows from lemma 2 , by using the substitution $r(x) := \frac{1}{M}\tilde{r}(\frac{x}{M})$, where $\tilde{r}$ is the rational function of lemma 2. Note that r and $\tilde{r}$ have the same degree. ∎

Lemma 6 $\frac{e^{-9\sqrt{m}}}{2} \leq \max_{x \in [-M,-\delta) \cup (\delta,M]} |r(x) - s(x)| \leq \frac{1}{2}\gamma_m M \delta^{-1}$.

If we want the rational approximation to be an ε-approximation we have the following.

Corollary 2 *If* $\delta \geq \frac{1}{2}\gamma_m M \varepsilon^{-1}$ *then* $\max_{x \in [-M,-\delta) \cup (\delta,M]} | \ r(x) - s(x) \ | \leq \varepsilon$.

A correspondent lemma could be also proven for polynomial approximations, by using lemma 4. We omit it here for the sake of brevity.

3.2 Approximate Simulation of Adaptive Computations

The above arguments naturally suggest a procedure of *approximate simulation.*

Definition 4 *An algorithm A for computing $g : I \to O$ ε-approximately simulates an algorithm B for computing $g : I \to O$ on a subset $I' \subseteq I$ if there is ε such that $\forall x \in I' \; |A(x) - B(x)| \leq \varepsilon$.*

Definition 5 *An adaptive algorithm taking n input values defines the semi-algebraic varieties $V = \bigcup_{i=1}^{k} \{x_1, \ldots, x_n : p_i(x_1, \ldots, x_n) = 0\}$, and $V_\delta = \bigcup_{i=1}^{k} \{x_1, \ldots, x_n : |p_i(x_1, \ldots, x_n)| < \delta\}$, where $p_i(x_1, \ldots, x_n)$ is a polynomial for which a decision $p_i(x_1, \ldots, x_n) > 0$ is made during the computation. We call V the critical and V_δ the δ-critical variety of the adaptive algorithm.*

Definition 6 *We denote with $M(r)$ the maximum absolute value of all intermediate quantities created during the computation on inputs $(x_1, \ldots, x_n) \in B(r)$, where $B(r) \in \mathbb{R}^n$ is the ball of radius r centered in the origin.*

Theorem 3 *An adaptive algorithm, associated with V and $M(r)$, can be approximately simulated by an oblivious algorithm for inputs in $I' = V_\delta \cap B(r)$.*

Proof. (Hint.) Use lemma 6 together with the simulation arguments. ∎

We now adapt the theorems on the volume of tubes reported in section 2 to estimate how large degenerate regions (critical varieties) are. We do this, by first finding bounds on the volume of the regions $\{x_1, \ldots, x_n : |p(x_1, \ldots, x_n)| < \delta\}$.

Lemma 7 *Let n be the length of the input and let d be the degree of p. Let $f(V_\delta)$ be the intersection of $B(1)$ with the set $\{x_1, \ldots, x_n : |p(x_1, \ldots, x_n)| < \delta\}$. Then*

$$f(V_\delta) \leq 2 \sum_{i=1}^{n} \binom{n}{i} (2d\delta)^i = 2(1 + 2d\delta)^n - 2.$$

Proof. It follows from the application of theorem 2 to the case of hypersurfaces. ∎

Theorem 4 *Let n be the length of the input and let be d the maximum degree of p_i in V_δ. Let $f(V_\delta)$ be the intersection of $B(1)$ with the set $\bigcup_{i=1}^{k} \{x_1, \ldots, x_n : |p_i(x_1, \ldots, x_n)| < \delta\}$, then*

$$f(V_\delta) \leq 2k \sum_{i=1}^{n} \binom{n}{i} (2d\delta)^i = 2k(1 + 2d\delta)^n - 2k.$$

Corollary 3 *Let $f(V_\delta)$ be as in theorem 4. If $d\delta n < \frac{\tau}{2}$, then $f(V_\delta) < \alpha \leq 1$, for n large enough, where $\tau = \log(\frac{\alpha}{2k} + 1)$.*

Lemma 8 *$f(V_\delta) \leq 4knd M \gamma_m \varepsilon_{loc}^{-1}$, where ε_{loc} denotes the local approximation error.*

Now we can give a lower bound on the degree of the approximating function. We assume that the global error ε_{glob}, as given in definition 4, satisfies $\varepsilon_{glob} = n^c \varepsilon_{loc}$ for large n, where c is a constant.

Lemma 9 *The global error ε_{glob} satisfies $\varepsilon_{glob} < \varepsilon$ for $x \in V_\delta$, with $f(V_\delta) < \alpha$, if*

$$m \geq c \log^2(\log^{-1}(\frac{\alpha}{2k} + 1) nd M \varepsilon^{-1}), \quad \textit{where } c \textit{ is a constant.}$$

Theorem 5 *If the global approximation error ε_{glob}, as given in definition 4 satisfies $\varepsilon_{glob} = n^c \varepsilon_{loc}$, where c is a constant, and if $k, M, d \leq n^{O(1)}$, then the degree m of the rational functions used in the simulation satisfies $m = O(\log^2 n)$.*

This theorem tells us that the simulation is relatively cheap as long as the algorithm we simulate is numerically stable.

3.3 Lower Bounds for Oblivious Algorithms

We give conditions on the data representation and primitive functions (which are satisfied by the systems used in practice) for which the adversary has the power sufficient to produce nontrivial lower bounds on the error.

Condition 1: We assume that an adversary can, for any path in any computation graph, provide input data for which there exists $\tilde{h}$, $|\tilde{h}| < h$, such that, $|\varepsilon_{R_i}| > \tilde{h}$, and $sign(\tilde{h}) = sign(\varepsilon_{R_i})$.

Condition 2: a) There exists a subset $R' \subseteq R$ such that for all $\tilde{y} \in R'$, for all n-ary primitive functions f and all $i \in \{1, \ldots, n\}$ there exist arguments $\tilde{x_1}, \ldots, \tilde{x_n} \in R$ such that $\tilde{x}_i \in R'$, $f^r(\tilde{x_1}, \ldots, \tilde{x_n}) = \tilde{y}$, $f(\tilde{x_1}, \ldots, \tilde{x_n}) = y$, $|y - \tilde{y}| \geq \tilde{h} > 0$, and $sign(y - \tilde{y}) = const.$ b) Furthermore for all primitive functions f and $\tilde{y} \in R$ there exist $\tilde{x_1}, \ldots, \tilde{x_n} \in R$ such that $f^r(\tilde{x_1}, \ldots, \tilde{x_n}) = \tilde{y}$.

Lemma 10 *If the computation is carried out on a tree, then Condition 2 implies Condition 1.*

Lemma 11 *If Condition 1 is satisfied, then, in the worst case, we have*

$$|\varepsilon_t| \geq \tilde{h} \sum_{i=1}^{t} \prod_{j=i+1}^{t} \alpha_j = \tilde{h}t + \tilde{h} \sum_{i=1}^{t} \beta_i, \text{ where } \varepsilon_{R_i} = \tilde{h} - \gamma_i, \text{ and } \prod_{j=1}^{i} \alpha_j = 1 + \beta_i.$$

Proof. We have $\varepsilon_t = \tilde{h}t + \tilde{h} \sum_{i=1}^{t} \beta_i - \sum_{i=1}^{t} \gamma_i \beta_i - \sum_{i=1}^{t} \gamma_i$. The proof follows from the fact that Condition 1 allows the adversary to provide data such that $sign(\gamma_i) \neq sign(\tilde{h})$, $i = 1, \ldots, t$. ∎

Corollary 4 *Under the hypothesis of lemma 11, the representation error for oblivious algorithms is in the worst case lower bounded by* $|\tilde{h}|C$ *if* $\alpha_i \geq 1$, $i = 1, \ldots, t$, *or by* $|\tilde{h}|(1+\delta)^C$, *if* $\alpha_i \geq 1 + \delta$, $\delta > 0$, *where* C *is the longest path in the computation graph of minimum depth.*

Note that the conditions $\alpha_i \geq 1$ or $\alpha_i \geq 1 + \delta$ are assumptions on primitive functions and/or intermediate data.

Corollary 5 *Assume that only one primitive function is used, and* $\alpha_i = \alpha > 0$, $i = 1, \ldots, t$, *is independent of the intermediate data. Then we have*

$$|\epsilon_C| \geq |\tilde{h}| \, |\sum_{i=1}^{C} \alpha^i| = \begin{cases} |\tilde{h}| \; |\frac{(\alpha - \alpha^{C+1})}{(1-\alpha)}|, & \alpha \neq 1, \\ |\tilde{h}| \; C, & \alpha = 1. \end{cases}$$

Theorem 6 *Under the hypothesis of lemma 11, let* g *be a differentiable function finitely composed of primitive functions. Let* D *be a lower bound on the minimal depth of a computation graph for the computation of* g. *Then the error introduced by an oblivious algorithm for* g *is in the worst case at least* $\tilde{h}(|g'| + D)$, *if* $\alpha_i \geq 1$ *or* $\tilde{h}(|g'| + \gamma^D)$, $\gamma > 1$, *if* $\alpha_i > 1$.

3.4 From Approximation and Topology to Complexity and Conditioning

In this section, we make a simplifying assumption on the nature of the computations we are considering. It is possible to prove that this is not a loss of generality. We assume that the simulation has the goal of approximating a computation carried out on a circuit (see section 2) consisting only of two layers, one combinatorial stage and one arithmetic stage. Let the roundoff error associated to this computation be such that the payoff function is strictly greater than zero. This means that the error introduced is $c\varepsilon_R$, with $c < 1$, where ε_R is the error unavoidable by oblivious algorithms performing the same computation. The above scenario leads to the following result.

Lemma 12 *Let* D_c *be the depth of the part of the circuit implementing the combinatorial stage. Let* g *be the (differentiable) function to be computed by the overall circuit. Then, we have* $\varepsilon_{loc} < \frac{\varepsilon_R}{|g'| D_c}$.

Proof. The error introduced in the approximation of the combinatorial stage is $\varepsilon_{loc} D_c$, so that the error of the overall computation is $|g'| \varepsilon_{loc} D_c$, where the factor $|g'|$ plays the role of amplifying the error. The proof now follows from the assumption that the payoff function is strictly positive. ∎

We can now state a very important result, concerning the feasibility of the simulation by rational functions for $x \in V_\delta$, in the presence of roundoff errors.

Theorem 7 $\frac{e^{-9\sqrt{m}}}{2} \leq \varepsilon_{loc} < \frac{\varepsilon_R}{|g'|D_c}$.

Proof. Follows from lemma 6 and lemma 12. ∎

From this we can also evaluate the cost of the rational simulation in the presence of roundoff errors in order to achieve a strictly positive payoff function.

Corollary 6 $m > 9\log^2(2\varepsilon_R^{-1}|g'|D_c)$.

If we further assume that an oblivious algorithm achieves the error defined in section 3.3, then we get the bound

$$m > 9\log^2(\frac{2D_c|g'|}{hD_a}),$$

where D_a denotes the minimal depth of an oblivious algorithm for computing g.

Finally, we get the following result for the degree m in the presence of roundoff errors.

Theorem 8 *If g is such that $|g'| \leq n^{O(1)}$, then $m = O(\log^2 n)$.*

Theorem 8 essentially says that the approximate simulation is "cheap" provided that we are not trying to solve an ill-posed problem.

For the lack of space we did not include the corresponding theorems and results for a polynomial simulation, which lead to the conclusion that polynomials of degree at least $n^{O(1)}$ are necessary. Another issue that did not fit into these pages is that of computations over complex numbers. In this case, the results obtained were similar to those presented for computations over the reals, with some additional features related to the fact that the non negligible volumes of tubes over complex varieties lead to non trivial lower bounds [D88, W39].

4 Conclusions and Further Work

In this paper we have shown that approximate computations over the reals do not suffer very much from the lack of conditionals, provided that division is allowed. By simulation, we have proved that it is possible to construct oblivious algorithms which, almost everywhere, behave like adaptive algorithms. The cost of this construction is very low if division is allowed and is very high if division is not allowed. The techniques employed here are very general and we believe will be used in several different areas.

One of the open problems we hope to see solved by using our techniques is related to the development of numerically stable NC algorithms. For example, Gaussian elimination with pivoting is P-complete [V89], whereas Gaussian elimination can be performed in NC. Can the simulation process described in this paper be applied to Gaussian elimination with pivoting, so that we obtain an algorithm almost everywhere reliable and, as the same time, parallelizable? More in general, assume that an adaptive algorithm is P-complete. Is its oblivious counterpart, obtained by approximate simulation, also P-complete?

The continuation of this work will include a detailed analysis of the polynomial approximation process, as well as the analysis carried out for computations over the complex numbers, where the regions of degeneracies have non trivial lower bounds [D88].

References

[AC92] Aberer, K., Codenotti, B. (1992). Towards a Complexity Theory of Approximation. *TR-92-012, International Computer Science Institute.*

[BF91] Babai, L., Fortnow, L., (1991). Arithmetization: A New Method in Structural Complexity Theory. *Computational Complexity, Vol.* **1**, *No. 1, pp 41-67.*

[BSS89] Blum, L., Shub, M., Smale, S. (1989). On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness. Recursive Functions and Universal Machines, *Bulletin of AMS, Vol.* **21**, *No. 1, pp 1-36.*

[C91] Cleve, R., (1991). Towards Optimal Simulation of Formulas by Bounded-width Programs. *Computational Complexity, Vol.* **1**, *No. 1, pp 91-105.*

[CF89] Codenotti, B., Flandoli, F., (1989). A Monte Carlo Method for the Parallel Solution of Linear Systems. *Journal of Complexity, Vol.* **5**, *pp 107-117.*

[CL91] Codenotti, B., Leoncini, M., (1991). Matrix Inversion in RNC^1. *Journal of Complexity, Vol.* **76**, *No. 3, pp 282-295.*

[CLR91] Codenotti, B., Leoncini, M., Resta, E., (1991). Oracle Computations in Parallel Numerical Linear Algebra. *TR ICSI 91-060.*

[D88] Demmel, J., (1988). The Probability that a Numerical Analysis Problem is Difficult. *Mathematics of Computation, Vol.* **50**, *No. 182, pp 449-481.*

[EC91] Emiris, I., Canny, J., (1991). A General Approach to Removing Degeneracies. *Proc. 32nd Annual Symposium on Foundation of Computer Science, pp 405-413.*

[G86] von zur Gathen, J., (1986). Parallel Arithmetic Computations: A Survey. *Lecture Notes in Computer Science, Vol.* **233**, *Springer-Verlag, New-York, pp 93-122.*

[G82] Gray, A. (1982) Comparison Theorems for the Volumes of Tubes as Generalizations of the Weyl Tube Formula. *Topology, Vol* **21**, *No. 2, pp 201-228.*

[L52] La Vallee Poussin, C.J. de, (1952). Lecons sur l'Approximation des Fonctions d'une Variable reelle. *Paris: Gauthier-Villars.*

[MST89] Mansour, Y., Schieber, B., Tiwari, P., (1989). The Complexity of Approximating the Square Root. *Proc. 30th Annual Symposium on Foundation of Computer Science, pp 325-330.*

[N64] Newman, D.J., (1964). Rational Approximation to $|x|$. *Michigan Math. J.* **11**, *pp 11-14.*

[O85] Ocneanu, A., (1985). On the Volumes of Tubes about a Real Variety. *Report, Mathematical Sciences Research Institute, Berkeley.*

[PR85] V. Pan and J. Reif, (1985). Efficient Parallel Solution of Linear Systems. *Proc. 17th Annual ACM Symposium on Theory of Computing, pp 143-152.*

[R69] Rivlin, T.J., (1969). An Introduction to the Approximation of Function. *Dover Publications, New York, NY.*

[SS91] Shub, M., Smale, S., (1991). Complexity of Bezout's Theorem I. Geometric Aspects. *manuscript.*

[S90] Smale, S., (1990). Some Remarks on the Foundation of Numerical Analysis. *SIAM Review, Vol.* **32**, *No. 2, pp 211-220.*

[S90] Strassen, V., (1990). Algebraic Complexity Theory. *Handbook of Theoretical Computer Science, Vol. A, pages 635-672.*

[TWW88] Traub, J.F., Wasilkowski G.W., Wozniakowski H., (1988). Information Based Complexity. *Academic Press, New York.*

[V89] Vavasis, S., (1989). Gaussian Elimination with Pivoting is P-Complete. *SIAM J. Disc. Math., Vol.* **2**, *No.3, pp 413-423.*

[W39] Weyl, H., (1939). On the Volume of Tubes. *Amer. J. of Math. Vol* **61**, *pp 461-472.*