Foundations of Exact Rounding

Chee K. Yap and Jihun Yu

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012
Email: {yap,jihun}@cs.nyu.edu,

Abstract. Exact rounding of numbers and functions is a fundamental computational problem. This paper introduces the mathematical and computational foundations for exact rounding. We show that all the elementary functions in ISO standard (ISO/IEC 10967) for Language Independent Arithmetic can be exactly rounded, in any format, and to any precision. Moreover, a priori complexity bounds can be given for these rounding problems. Our conclusions are derived from results in transcendental number theory.

1 Introduction

Modern scientific and engineering computation is predominantly based on floating point computations. Central to such computations is the now ubiquitous floating-point hardware that performs the four arithmetic operations and square-root to double or quadruple precision, exactly rounded. The IEEE 754 standard [11] specifies the various rounding modes that must be supported: round up or round down, round toward or away from zero, and round to nearest (with suitable tie-breaking rule).

To supplement these built-in hardware operations, many scientific computation also need a mathematical library in software for the elementary operations such as exponentiation, logarithm, trigonometric functions, etc. There is an ISO standard ISO/IEC 10967 for Language Independent Arithmetic (LIA) which is compatible with the IEEE 754 standard [18, 3]. The second part of this standard (known as LIA-2) specifies a list of elementary functions that should be implemented in such libraries. A very similar list of functions from Defour et al [3] is reproduced in Table 1.

An excellent general reference for the issues of exact rounding, including various algorithms for elementary functions is Muller [16]. Issues of hardware implementation and fixed precision formats are also addressed in this book. There is also a growing interest in arbitrary precision "big number" packages that need such algorithms. A much larger class of functions than elementary functions is the class of hypergeometric functions. In [5, 6], we provide algorithms for computing hypergeometric functions to any desired precision.

Logarithms:	$\log(x), \log_2(x), \log_{10}(x), \log_y(x), \log(1+x)$
Exponentials:	$\exp(x), \exp(x)-1, 2^x, 10^x, x^y$
Trigonometric:	$\sin(x), \cos(x), \tan(x), \cot(x), \sec(x), \csc(x)$
Inverse Trigonometric:	$\arcsin(x), \arccos(x), \arctan(x), \arctan(x/y)$, $\operatorname{arccot}(x), \operatorname{arcsec}(x), \operatorname{arccsc}(x)$
Hyperbolic:	$\sinh(x), \cosh(x), \tanh(x), \coth(x), \operatorname{sech}(x), \operatorname{csch}(x)$
Inverse Hyperbolic:	$\operatorname{arcsinh}(x), \operatorname{arccosh}(x), \operatorname{arctanh}(x), \operatorname{arctanh}(x/y)$, $\operatorname{arccoth}(x), \operatorname{arcsech}(x), \operatorname{arccsch}(x)$

Fig. 1. Elementary functions in the LIA-2

The ability to approximate a given function f to any desired precision is a prerequisite for the general problem[1] of exact rounding for the function f. But this ability alone is insufficient for the exact rounding of f. In general, this only be resolved by using nontrivial results from transcendental number theory. For instance, current knowledge does not allow us to exactly round the hypergeometric functions. According to [3], there is no IEEE standard analogous to LIA-2 because the theory of exact rounding for such functions is not understood.

Until the advent of modern computers, various elementary functions are precomputed and stored in tables for hand calculation. The **Table Maker's Problem** amounts to producing exactly rounded function values at a finite set G of inputs, with the function values also exactly rounded to G. We call G a "grid" for rounding. E.g., G can be the set of IEEE 754 double precision numbers in binary format. The associated **Table Maker's Dilemma** arises because it is not apparent that there are terminating algorithms to produce the table entries correctly [10, 16, 13, p. 166]. Indeed, this dilemma is another form of the Zero Problem [19, 23].

Why is exact rounding important? A exactly rounded math library is the foundation for trustworthy numerical computations, for code portability, for use in computed-assisted proofs, etc. See [3, 16] for other considerations. Another application arises in Computational Geometry, where the problem of numerical non-robustness is especially acute. A highly successful approach for achieving robust algorithms is **Exact Geometric Computation** (EGC) [23]. To speed up EGC computations, we need floating point filters [2, 15]. A basis for such filters is exactly rounded arithmetic at machine-level. In our Core Library (version 2), we introduced transcendental functions in expressions [4]. The corresponding filters need exactly rounded elementary functions in some fixed precision library, such as specified by LIA-2.

Let us briefly review some basic results on exact rounding of elementary functions. In Lefèvre et al [13], the exact rounding problem was solved for elementary functions for the IEEE double precision format in radix 2. Their general

[1] The problem is also known as "correct rounding". We prefer the terminology of "exact rounding" as there are situations when we would settle for less stringent notions of exactness, but consider it to be "correct" nevertheless. For instance, [3] describes 3 levels of rounding.

approach was to search for the worst case input numbers in the IEEE format. By exhaustive search, it is determined that the smallest gap between a function value and a breakpoint is greater than 2^{-119}. (A "break point" is either an possible input number, or the midpoint between consecutive input numbers.) It follows that if we evaluate the elementary functions to 119 bits of accuracy, we can round exactly.

The exhaustive search approach is important for hardware implementation of exactly rounded functions. But it's extendability is fragile: a new search must be mounted for input numbers in other precision, or in other formats. In short, each choice of a rounding grid G needs its own search. For example, Lefèvre et al [14] describe the search for the exponential function on 64-bit IEEE numbers in decimal format. They show that exact rounding can be achieved by approximating the exponential function to 47 digits. With 64-bit inputs, the search space is too large for naive search. Among the sophisticated search methods they employed is one based on lattice reduction. But exhaustive search for 128-bit formats is way beyond current techniques and computing power [16, p. 168].

Another important application for exact rounding is in multiprecision floating point libraries. Unlike hardware implementation applications, we do not need to determine the worst case precision for a fixed grid. We only need an algorithm that is guaranteed to produce the exactly rounded answer for any potential input number. One of the most widely used multiprecision libraries is `gmp` (Gnu Multiprecision library). The bigfloat subsystem in `gmp` does not guarantee exact rounding. The `mpfr` project [8] aims to remedy this shortcoming; but the underlying basis for such algorithms does not seem to have been published. Ziv and others [25, 9] have proposed to use arbitrary precision computation to achieve exact rounding. The problem with a naive application of arbitrary precision computation is termination. Only heuristic arguments have been used to argue for a high probability of termination.

The purpose of the present article is to clearly formulate the fundamental principles behind exact rounding. We show that there are terminating algorithms for computing exactly rounded values for the standard elementary functions, in any format and to any desired precision. In this sense, we solve the Table Maker's Dilemma for such functions. Worst case complexity estimates for some of these algorithms are also given. Our conclusions are based on basic results from transcendental number theory (TNT). Note that Muller [16, p. 167] had already observed that Lindemann's result in TNT can be used to justify exact rounding; he also gave quantitative bounds from Baker type theory in TNT. A general reference for TNT is [7].

¶*1. Computational Model.* We consider only $\mathbb{R}$ (the real numbers). Usually, extensions of our results to $\mathbb{C}$ is routine. In computing with reals, we use the computational model of [22, 24] in which numerical inputs and outputs for algorithms are members of some set $\mathbb{F} \subseteq \mathbb{R}$ such that $\mathbb{F}$ contains the integers $\mathbb{Z}$, and $\mathbb{F}$ is dense in $\mathbb{R}$ and is closed under the ring operations $(+, -, \times)$ as well as division by 2 (so $x \in \mathbb{F}$ implies $x/2 \in \mathbb{F}$), and allow exact comparisons. We may call $\mathbb{F}$ the **computational ring** of our model. There are several common

choices for this ring. Consider the following tower of rings

$$\mathbb{Z}[1/2] \subseteq \mathbb{Z}[1/10] \subseteq \mathbb{Q} \subseteq \mathbb{A} \cap \mathbb{R}$$

where $\mathbb{Z}[1/2] = \{m2^n : m, n \in \mathbb{Z}\}$ is the set of bigfloats or dyadic numbers, $\mathbb{Z}[1/10] = \{m10^n : m, n \in \mathbb{Z}\}$ is the set of decimal numbers, $\mathbb{Q}$ is the set of rational integers, and $\mathbb{A}$ is the set of algebraic numbers. We can choose $\mathbb{F}$ to be any one of these four rings. For these "standard" choices, our computational model can be implemented by standard Turing machines [24].

Unlike the computational model of computable analysis [12], ours allows exact comparison between members of $\mathbb{F}$; this is crucial for the rounding problem. Note only can we recognize if a given $x \in \mathbb{F}$ is 0, but we also determine if x is in $\mathbb{Z}$ (resp., $\mathbb{Q}$). E.g., this is needed for rounding for the function y^x where the case $x \in \mathbb{Q}$ is an exceptional case that is separately treated.

The most important computational ring is $\mathbb{F} = \mathbb{Z}[1/2]$; it is also the minimal computational ring by our axioms. In the following, $\mathbb{F} = \mathbb{Z}[1/2]$ is our default, unless otherwise noted. In order to formulate the rounding problem using $\mathbb{F}$ as the rounding target, we need to introduce a "scale" or grading on $\mathbb{F}$. A tower of proper inclusions $\mathbb{F}_0 \subseteq \mathbb{F}_1 \subseteq \mathbb{F}_2 \subseteq \cdots$ such that $\cup_{n\geq 0}\mathbb{F}_n = \mathbb{F}$ is called a **grading** of $\mathbb{F}$,

For our minimal ring $\mathbb{F} = \mathbb{Z}[1/2]$, we use the grading in which $\mathbb{F}_n = \mathbb{Z}/2^n := \{m2^{-n} : m \in \mathbb{Z}\}$ for each $n \in \mathbb{N}$. In general, for any set $G \subseteq \mathbb{R}$, let[2] $G/N := \{a/N : a \in G\}$ where $N \neq 0$. We can formulate similar grading for other choices of $\mathbb{F}$, with the proviso that $\mathbb{Z}/2^n \subseteq \mathbb{F}_n$. E.g., for $\mathbb{Q}$, we can choose $\mathbb{Q}_n = \{m/b : 1 \leq b \leq n+1, m \in \mathbb{Z}\}$.

Bigfloats are a convenient abstraction of the finite-precision binary formats of the IEEE numbers [11]. In particular, the IEEE numbers in double precision is a finite subset of $\mathbb{F}_{1075}$. The choice $\mathbb{F} = \mathbb{A} \cap \mathbb{R}$ is mostly of theoretical interest. It is conceivable that $\mathbb{F} = \mathbb{Z}[\sqrt{2}]$ has attractive computational properties.

Let $f : \mathbb{R} \to \mathbb{R}$ be any real function. An **approximation** for f is any function $\widetilde{f} : \mathbb{F} \times \mathbb{N} \to \mathbb{F}$ with the property that $\widetilde{f}(x, n) \in \mathbb{F}_n$ and $\widetilde{f}(x, n) = f(x) \pm 2^{-n}$ (i.e., $\widetilde{f}(x, n)$ is correct to n bits). In general, a numerical expression of the form "$x \pm \varepsilon$" (involving the symbol '$\pm$') denotes some value of the form $x + \theta\varepsilon$ where θ is an implicit variable satisfying $|\theta| \leq 1$.

Suppose we fix some rounding rule r and some grading $\{\mathbb{F}_n : n \in \mathbb{N}\}$ of $\mathbb{F}$. Then an **exactly rounded approximation** for f is

$$\widehat{f} : \mathbb{F} \times \mathbb{N} \to \mathbb{F} \tag{1}$$

such that (1) $\widehat{f}$ is an approximation for f, and (2) the value $\widehat{f}(x, n) \in \mathbb{F}_n$ is rounded following the rule r. In the next section, we will clarify the notion of rounding rules.

Our main task is to provide an algorithm for computing $\widehat{f}$. For this, we make two natural assumptions: First, we assume the availability of an algorithm to compute an approximation $\widetilde{f}$ for f. Such algorithms are well-known (e.g., [1, 22,

[2] $\mathbb{Z}/N$ should not be confused with integers mod N, a concept not used in this paper.

6]). Second, we assume that given $x \in \mathbb{F}$ and $n \in \mathbb{N}$, we know how to round x in $\mathbb{F}_n$. Both these assumptions are easily satisfied in our computational model, assuming the standard choices of f and $\mathbb{F}$. We will next see what else is needed to complete our task.

2 The Rounding Framework

We describe an abstract framework for rounding a real number y. In the context of our main task of computing $\widehat{f}$, the number y is the value of the function f at some point $x \in \mathbb{F}$.

Rounding of numbers takes place with respect to some discrete set G of points: a countable set $G \subseteq \mathbb{R}$ that is closed is called a (rounding) **grid**. Each $g \in G$ is called a **grid point**. Intuitively, the grid points serve as potential values for the approximation of real numbers. Note that G may be a finite set.

Examples: Typically, $G = \mathbb{F}_n$ in arbitrary precision arithmetic computation. For hardware-oriented algorithms, G can be the set of IEEE machine doubles. In evaluation of the arctangent function, it is desirable to output an element $y \in G$ that is in $[-\pi/2, \pi/2]$, to ensure monotonicity properties (see [16, 3]). One way to do this is to can all integer multiples of $\pi/2$ into G; it is possible to compute effective with this extended set [24].

Relative to grid G, we introduce the **floor** and **ceiling** functions on $y \in \mathbb{R}$:

$$\lfloor y \rfloor_G := \max\{x \in G : y \geq x\}, \qquad \lceil y \rceil_G := \min\{y \in G : y \leq x\}.$$

If y is less than the smallest grid point, $\lfloor y \rfloor_G = -\infty$ and if y is greater than the largest grid point, $\lceil y \rceil_G = +\infty$. Thus, $-\infty$ and $+\infty$ are implicitly included in all grids. A **rounding rule** for G is a function $r : \mathbb{R} \to G \cup \{-\infty, +\infty\}$ such that for all $y \in \mathbb{R}$, we have $r(y) \in \{\lfloor y \rfloor_G, \lceil y \rceil_G\}$. Here are 5 standard rounding rules:

r_1: Round Up	$r_1(y) = \lceil y \rceil_G$
r_2: Round Down	$r_2(y) = \lfloor y \rfloor_G$
r_3: Round toward Zero	$r_3(y) = \lfloor y \rfloor_G$ iff $y \geq 0$
r_4: Round away from Zero	$r_4(y) = \lceil y \rceil_G$ iff $y \geq 0$
r^*: Round To Nearest	$\lvert r^*(y) - y \rvert = \min\{\lvert y - \lfloor y \rfloor_G \rvert, \lvert \lceil y \rceil_G - y \rvert\}$

Note rounding to nearest $r^*(y)$ is sometimes denoted $\lfloor y \rceil$. But this rule r^* is incomplete as stated because when $y = (\lfloor y \rfloor_G + \lceil y \rceil_G)/2$, the round-to-nearest rule does not give a unique determination of $r^*(y)$. In this case, $r^*(y)$ can be decided by using one of the other four rules $(r_1, \ldots, r_4)$ as tie-breaker rule. We denote the corresponding rule by r_i^* $(i = 1, \ldots, 4)$.

There are two more rounding rules, based on an additional property of G: suppose there is **parity function** $par : G \to \{odd, even\}$. By definition, the parity function assigns different parity to any two adjacent y's in G. Clearly, there are only two possible parity functions for G. The standard parity function is the one where 0 has even parity. In case $G = \mathbb{F}_n$, this choice assigns to $g \in \mathbb{F}_n$

an even parity iff the mantissa of y is even (i.e., $y = m2^{-n}$ and m is even). We now have two more rounding rules: **Round to Even** (r_e) or **Round to Odd** (r_o). For instance, $r_e(y) = y$ if $y \in G$; otherwise $r_e(y)$ is the even parity grid point from $\{\lfloor y \rfloor_G, \lceil y \rceil_G\}$. These rules work because $\lfloor y \rfloor_G$ and $\lceil y \rceil_G$ have different parity when they are distinct. As before, we can also use these two rules as tie-breaker for round to nearest, and thus obtain the rounding rules denoted by r_e^* and r_o^*. We remark that rule r_e^* is empirically found to be a good rule for reducing error in a computation. So it is often the default rounding mode.

Naively, the problem of exact rounding is this: given a rounding rule $r : \mathbb{R} \to G$, construct an algorithm to compute r. This is naive because ordinary algorithms (in the sense of Turing) cannot possibly accept arbitrary real numbers, which are uncountably many, and which generally has no finite representation. Instead, suppose each real number y is given by some "black box" which, given $n \in \mathbb{N}$, will return an n-bit approximation of y. Formally, a **black box number** (or Cauchy function [12]) is a function $\widetilde{y} : \mathbb{N} \to \mathbb{F}$ such that there exists a $y \in \mathbb{R}$ with the property that $\widetilde{y}(n) = y \pm 2^{-n}$ for all $n \in \mathbb{N}$. Call $\widetilde{y}$ a black box for y. In computable analysis, y is called a **computable real number** if it has a black box $\widetilde{y}$ that is recursive. Turing machines can be extended to use such black boxes as inputs and outputs; these are called **oracle Turing machines** (see [12, 24]).

The **black box rounding problem** relative to some rounding function r on grid G is this: given a black box $\widetilde{y}$ for y, to compute $r(y) \in G$ by using $\widetilde{y}$ as an oracle. The black box $\widetilde{y}$ arise naturally when y is the value of an approximable function f at a point $a \in \mathbb{F}$. In this case, $\widetilde{y}(n)$ is just $\widetilde{f}(a, n)$ where $\widetilde{f}$ is any fixed approximation of f.

We call G an **effective grid** if (1) for all $y \in \mathbb{F}$, we can compute $\lceil y \rceil_G$ and $\lfloor y \rfloor_G$, and (2) for all $g \in G$, we can determine the next larger g^+ and next smaller g^- grid point, $g^- < g < g^+$. Typically $G \subseteq \mathbb{F}_n$ for some n, and the effectiveness of G is easy to see. But the introduction of non-algebraic grid points like π requires special considerations which we must address.

Recall the definition of the exactly rounded approximation $\widehat{f}$ in (1). We can now interpret the problem of computing $\widehat{f}(x, n)$ as rounding $f(x)$ to the grid $\mathbb{F}_n$.

3 Two Preconditions of Exact Rounding

Given an effective grid G, and $y \in \mathbb{R}$, our goal is to round y in G according to some rounding rule. Define $\|y\|_G := \inf_{x \in G} |x - y|$ to be the distance from y to the nearest grid point. Thus $y \in G$ iff $\|y\|_G = 0$. Clearly, one of these two equalities must hold:

$$\lfloor y \rfloor_G = y - \|y\|_G \quad \text{or} \quad \lceil y \rceil_G = y + \|y\|_G.$$

For any $\varepsilon \geq 0$, we say that y is **ε-discrete** in G provided that $y \notin G$ implies $\|y\|_G > \varepsilon$. We shall denote this condition by

$$\Delta_\varepsilon(y, G). \tag{2}$$

We now state two (alternative) preconditions on (y, G) for this:

– Precondition A: $y \notin G$.
– Precondition B: $\Delta_\varepsilon(y, G)$ holds for a known $\varepsilon > 0$.

Our first result shows that either of these two preconditions suffices to achieve exact rounding; it further shows that some precondition is unavoidable.

Theorem 1. *Let G be a fixed effective grid.*
(i) There are oracle Turing machines M_A and M_B such that for all black boxes $\overline{y}$ for y, the machine M_X ($X = A$ or B) on $\overline{y}$ will output $\lceil y \rceil_G$ if Precondition X holds.
(ii) For all oracle Turing machines M, there exists black boxes $\overline{y}$ such that M on $\overline{y}$ does not compute $\lceil y \rceil_G$.

Proof. The oracle Turing machines M_A and M_B will be given below. So we only prove (ii) here. Let M be an oracle Turing machine that takes an input oracle $\widetilde{y}$ (any black box for $y \in \mathbb{R}$) and outputs $\lceil y \rceil_G$ after finitely many steps. Let $g < g'$ be two consecutive grid points. Let $\underline{g}$ be a black box for g with $\underline{g}(n) = g - 2^{-n-1}$ for all n. Then M on input $\overline{g}$ must output g. Let n_0 be the largest value of n such that the computation of M on $\underline{g}$ queries the oracle $\underline{g}(n)$. Now let $y = g + 2^{-n_0-1}$, and choose any black box $\widetilde{y}$ for y that agrees with $\overline{g}$ for the first n_0 values. Clearly M on input $\widetilde{y}$ will still output g. This is wrong since $\lceil y \rceil_G = g'$. **Q.E.D.**

¶*2. METHOD A.* If $y \notin G$, we have the following method for computing $\lceil y \rceil_G$: Compute $\widetilde{y}(n) = y \pm 2^{-n}$ for $n = 0, 1, 2, \ldots$ until the interval $[\widetilde{y}(n) \pm 2^{-n}] = [\widetilde{y}(n) - 2^{-n}, \widetilde{y}(n) + 2^{-n}]$ contains no grid points,

$$[\widetilde{y}(n) \pm 2^{-n}] \cap G = \emptyset. \tag{3}$$

We then output $\lceil \widetilde{y}(n) \rceil_G$.

Correctness: By our assumption about the effective grid G, we can check (3) since this amounts to checking that $|\lceil \widetilde{y}(n) \rceil_G - \widetilde{y}(n)| > 2^{-n}$ and $|\lfloor \widetilde{y}(n) \rfloor_G - \widetilde{y}(n)| > 2^{-n}$. Furthermore, (3) ensures that $\lceil y \rceil_G = \lceil \widetilde{y}(n) \rceil_G$. Finally, observe that (3) will eventually hold since $y \notin G$.

¶*3. METHOD B.* In contrast to the precondition A, the ε-discreteness property $\Delta_\varepsilon(y, G)$ does not exclude the possibility that $y \in G$. The constant $\varepsilon > 0$ (or a positive lower bound) must be effectively known. We use the following lemma.

Lemma 1 (Discreteness Lemma). *Suppose $\Delta_\varepsilon(y, G)$ and $\widetilde{y} = y \pm \varepsilon/2$. Then: $\lfloor y \rfloor_G < y < \lceil y \rceil_G$ iff*

$$\lfloor y \rfloor_G + \varepsilon/2 < \widetilde{y} < \lceil y \rceil_G - \varepsilon/2.$$

Proof. It is enough (by symmetry) to show that $y < \lceil y \rceil_G$ iff $\widetilde{y} < \lceil y \rceil_G - \varepsilon/2$. If $y < \lceil y \rceil_G$, then by ε-discreteness, $y + \varepsilon < \lceil y \rceil_G$. Thus $\widetilde{y} \leq y + \varepsilon/2 < \lceil y \rceil_G - \varepsilon/2$. Conversely, if $\widetilde{y} < \lceil y \rceil_G - \varepsilon/2$, then $y \leq \widetilde{y} + \varepsilon/2 < \lceil y \rceil_G$. **Q.E.D.**

METHOD B goes as follows: Fix $N = \lceil \lg(2/\varepsilon) \rceil$. Compute $\widetilde{y}(n) = y \pm 2^{-n}$ for $n = 0, 1, 2, \ldots$ until the first time that one of the following cases hold:

Case (i): If (3) holds (as in METHOD A), we output $\lceil \widetilde{y}(n) \rceil_G$ (as before).
Case (ii): $[\widetilde{y}(n) - 2^{-n}, \widetilde{y}(n) + 2^{-n}] \cap G$ contains exactly one point g and $n \geq N$. We output g in this case.

Note that termination is assured since the interval $[\widetilde{y}(n) \pm 2^{-n}] \cap G$ will eventually have at most one point. The output in case (i) is clearly correct. In case (ii), by ε-discreteness, we know that $y = g$ and so $\lceil y \rceil_G = g$. It should be observed how ε-discreteness is used in an essential way.

¶*4. Other Rounding Modes.* The preceding METHODs A and B were for rounding up (or r_1). They can clearly be modified for rounding down (r_2). It can also be used for round to odd (r_o) and round to even (r_e): assuming that we can decide if a grid point is odd of even, we first compute $\lceil y \rceil_G$.

To extend them to r_3 and r_4, we need to know the sign of y. Actually, all we need to know if whether $y = 0$. If $y \neq 0$, we can then determine the sign of y from $\widetilde{y}$. Unfortunately, deciding if $y = 0$ from $\widetilde{y}$ is a well-known undecidable problem in computable analysis. However, in our applications below, y is the value of the elementary functions evaluated at dyadic points; we could explicitly detect when $y = 0$, and METHOD B will not be invoked if $y = 0$.

Finally, consider the round to nearest function r_i^* (various i). Here, we must extend the grid G to G' where $G \subseteq G'$ and the extra grid points of G' are the "breakpoints" that lie midway between two consecutive grid points of G. Method A extends directly to the grid G'. For Method B, we need some discreteness property, $\Delta_\varepsilon(y, G')$ for a suitable ε. This concludes our discussion of the rounding algorithms.

REMARK: we can extend the above rounding methods to complex grids, $G \subseteq \mathbb{C}$ with grading with $G_n = \{x + \mathbf{i}y : x, y \in \mathbb{F}_n\}$.

4 Rounding the Elementary Functions

We now consider exact rounding of the elementary functions found in Table 1. The functions there can be put into two groups: the "pure" functions are given by

$$\left.\begin{array}{ll} \textit{Exponentiation}: & \exp(x), \exp(x) - 1 \\ \textit{Trigonometric func.}: & \sin(x), \cos(x), \tan(x), \cot(x), \sec(x), \csc(x) \\ \textit{Hyperbolic func.}: & \sinh(x), \cosh(x), \tanh(x), \coth(x), \operatorname{sech}(x), \operatorname{csch}(x) \end{array}\right\} \quad (4)$$

together with all their inverses

$$\log(x), \log(1 + x); \arcsin(x), \arccos(x), \ldots; \operatorname{arcsinh}(x), \ldots.$$

The inverses (except for log) are multivalued functions: we will assume principal values for these functions. However, they are easily generalized to allow the specification of any desired branch, by introducing an extra integer argument. The second group of functions from Table 1 are

$$\left.\begin{array}{ll} \textit{Logarithm}: & \log_2(x), \log_{10}(x), \log_y(x) \\ \textit{Exponential}: & 2^x, 10^x, x^y \\ \textit{Inverse Trigonometric}: & \arctan(x/y) \\ \textit{Inverse Hyperbolic}: & \operatorname{arctanh}(x/y) \end{array}\right\} \quad (5)$$

This group is treated separately. We appeal to the following classic results from transcendental number theory:

Proposition 1.
(a) [Lindemann] *For any complex number $\alpha \neq 0$, either α or e^α is transcendental.*
(b) [Gelfond-Schneider] *For any complex numbers α, β where $\alpha(1-\alpha) \neq 0$ and β is irrational, one of the numbers in the set $\{\alpha, \beta, \alpha^\beta\}$ is transcendental.*

In order to apply this proposition, we need some basic facts about algebraic functions. A partial function $f : \mathbb{C} \to \mathbb{C}$ is **algebraic** if there is an integer polynomial $F(X,Y)$ such that $F(x, f(x)) = 0$ for all x in $\mathrm{dom}(f) := \{x : f(x) =\downarrow\}$, the domain of f. To ensure closure of algebraic functions under composition, it turns out that we also require that if $F(X,Y) = F_1(X)F_2(X,Y)$ then $F_1(X)$ must be a constant. Here $f(x) =\downarrow$ means f is defined at x. We say $F(X,Y)$ is a **defining polynomial** for f. The constant function $f(x) = c$ where $c \in \mathbb{A}$ is the simplest example of an algebraic function; another is $f(x) = \sqrt{x}$. If a function is not algebraic, we say it is **transcendental**. By an inverse function of f, we mean any partial function $g : \mathbb{C} \to \mathbb{C}$ such that $f(g(x)) = x$ holds for all $x \in \mathrm{dom}(g)$. Write $f^{-1}(x)$ to denote any choice of an inverse functions of f. (The notation f^{-1} will not be used for $1/f$ in this paper.) Also, let $f \circ g$ denotes function composition, $(f \circ g)(x) = f(g(x))$.

Our application of Prop. 1 is reduced to an application of the following:

Lemma 2. *Let f be an algebraic function. If $x \in \mathrm{dom}(f)$ is algebraic then $f(x)$ is algebraic.*

Proof. Let $F(X,Y)$ be a defining polynomial for f. For algebraic x, the relation $F(x, f(x)) = 0$ implies that $f(x)$ is the zero of a polynomial with algebraic coefficients. Thus $f(x)$ is algebraic. **Q.E.D.**

Combining this lemma with the results of Lindemann and Gelfond-Schneider Prop. 1, we conclude:

Corollary 1. *Let $y \in \mathbb{A}$ with $y(1-y) \neq 0$. The functions $f(x) = e^x$ and $g(x) = y^x$ are transcendental functions.*

In this corollary, we could have stated that $g(x,y) = y^x$ is a transcendental (i.e., non-algebraic) function. The definition of algebraic functions on more than one variable is a straightforward extension of the univariate case; but we avoided this for simplicity.

Here is the analogue of Lemma 2 for transcendental functions: *if f is transcendental, and $x \in \mathrm{dom}(f)$ is algebraic then $f(x)$ is transcendental.* Unfortunately, this is falsified even by $f(x) = e^x$: just take $x = 0$. We say x is an **exceptional argument** for a function f if both $f(x)$ and x are algebraic. Let $E_f \subseteq \mathbb{A}$ denote the set of exceptional arguments for a function f. Lindemann says that 0 is the only exceptional argument for the function e^x, $E_f = \{0\}$. Fortunately, all the elementary functions has exactly one exceptional argument and they are

easy to detect. Gelfond-Schneider says that the set of exceptional arguments for $g(x) = y^x$ is contained in $\mathbb{Q}$. We easily verify that $E_g = \mathbb{Q}$.

Lemma 3. *The exceptional arguments of f^{-1} are contained in the set $f(E_f) = \{f(x) : x \in E_f\}$. In particular, f^{-1} cannot have more exceptional arguments than f.*

Proof. Let y be exceptional for f^{-1}. We must show that $y = f(x)$ for some $x \in E_f$. Then $\{f^{-1}(y), y\} \subseteq \mathbb{A}$. Writing $x = f^{-1}(y)$, we also have $\{x, f(x)\} \subseteq \mathbb{A}$. So $x \in E_f$ and $y = f(x)$ as desired. **Q.E.D.**

The following allows us to conclude that certain functions are algebraic:

Theorem 2 (Closure of Algebraic Functions). *If f, g are algebraic functions, so are*

$$f + g,\ f - g,\ fg,\ 1/f,\ \sqrt{f},\ f^{-1},\ f \circ g.$$

Proof. Assume $F(x, f(x)) = 0$ and $G(x, g(x)) = 0$. Our basic tool is resultant theory [21, chap. 6]. View $F(X, Y) \in \mathbb{Z}[X, Y]$ as a polynomial in $D[X]$ where $D = \mathbb{Z}[X]$. Then we can view $f(x)$ as an element of $\overline{D}$ (algebraic closure of D). The constructions for $f + g, f - g, fg, 1/f$ are obtained from the corresponding resultants for adding, subtracting, multiplying and taking reciprocals of algebraic numbers (Lemma 6.16 in [21, p.158]). To see that $h = \sqrt{f}$ is algebraic, let $H(X, Y) = F(X, Y^2)$. Then $H(x, h(x)) = F(x, h(x)^2) = F(x, f(x)) = 0$. To see that the inverse $h = f^{-1}$ is algebraic, let $H(X, Y) = F(Y, X)$. Then for all y in the domain of $h = f^{-1}$, $H(y, h(y)) = F(f^{-1}(y), y) = F(f^{-1}(y), f(f^{-1}(y))) = F(x, f(x)) = 0$. For function composition $h = f \circ g$, let $H(X, Y) = \mathrm{res}_Z(F(Z, Y), G(X, Z))$. Then $H(x, f(g(x)) = 0$. **Q.E.D.**

An application of the preceding theorem shows that certain functions are transcendental:

Corollary 2. *If f is transcendental and g is algebraic, then the following are transcendental:*

$$f + g,\ f - g,\ fg,\ f/g,\ 1/f,\ \sqrt{f},\ f^{-1},\ f \circ g,\ g \circ f.$$

The next theorem assumes the general setting where the set $\mathbb{F}$ is the set of real algebraic numbers.

Theorem 3. *Assume $\mathbb{F} = \mathbb{A} \cap \mathbb{R}$. Let f or the inverse of f be an elementary function from the list (4).*
(a) f is transcendental with one exceptional argument.
(b) The exact rounding problem for f in an effective grid $G \subseteq \mathbb{F}$ is solvable.

Proof. Suppose we have shown part (a), i.e., f is transcendental, and determined its single exceptional argument x_0. Then part (b) follows in a generic way: the algorithm for exact rounding of f amounts to detecting if an input $x \in \mathbb{F}$ is

exceptional. If so, explicitly output the exact rounding of $f(x_0)$ in $G \subseteq \mathbb{F}$ (invariably, this is trivial). Otherwise, x is non-exceptional, and we use METHOD A to round $f(x)$ in the grid G.

Exponential function: If $f(x) = e^x$, we already saw that $f(x)$ is transcendental provided $x \neq 0$. If $x = 0$, we output $e^x = 1$. Otherwise, we use METHOD A to round $f(x) = e^x$ to G. Clearly, the method extends to the variant where $f(x) = e^x - 1$.

Trigonometric functions: Suppose $y = \sin(x)$ (the case $y = \cos(x)$ is very similar). If $x = 0$, we may output $\sin(0) = 0$. Otherwise, consider $f(y) = f(\sin x) := \sqrt{1 - \sin^2 x} + \mathbf{i} \sin x = \sqrt{1 - y^2} + \mathbf{i}y$. By Thm. 2, $f(y)$ is an algebraic function of y. But $f(y) = f(\sin x) = e^{\mathbf{i}x}$ and $\mathbf{i}x$ is non-zero algebraic; so Lindemann says $f(y) = e^{\mathbf{i}x}$ is transcendental. From this fact, and Lemma 2, we conclude that y is transcendental.

Suppose $y = \tan(x) = \frac{s}{\sqrt{1-s}}$ where $s = \sin(x)$. Thus, $\tan(x)$ is an algebraic function of $\sin(x)$. By Thm. 2, we conclude $\sin(x)$ is an algebraic function of $\tan(x)$. If $x = 0$ then we may output $\tan(0) = 0$. Otherwise, we already know that $\sin(x)$ in transcendental. Then $\tan(x)$ is transcendental.

Since $\cot(x) = 1/\tan(x)$, $\sec(x) = 1/\cos(x)$, $\csc(x) = 1/\sin(x)$, it follows from Corollary 2 that $\cot(x), \sec(x), \csc(x)$ are transcendental functions. Moreover, their exceptional argument are directly obtained from the exceptional argument of their reciprocals (Lemma 3). We can round exactly at these exceptional points.

The hyperbolic functions are seen to be transcendental by Corollary 2, since they are derived from transcendental trigonometric functions using the following algebraic relations:

$$\sinh(x) = \mathbf{i} \sin(\mathbf{i}x), \qquad \cosh(x) = \cos(\mathbf{i}x), \qquad \tanh(x) = -\mathbf{i} \tan(\mathbf{i}x).$$

Inverse functions: finally, we address the inverses of all the preceding functions. It is clear from the preceding proofs that these functions are transcendental and each has exactly one exceptional argument. By Corollary 2, the inverses are all transcendental. Moreover, by Lemma 3, the inverses has at most one exceptional argument. We can directly round the functions at the single exceptional argument. **Q.E.D.**

We now turn to the functions (5) of the second group.

Theorem 4. *Let $\mathbb{F} = \mathbb{A} \cap \mathbb{R}$ and $y \in \mathbb{A}$ and $y(y-1) \neq 0$. Consider the functions $f(x) = y^x$ and its inverse $g(x) = \log_y x$.*
(a) $f(x)$ and $g(x)$ are transcendental with exceptional values $E_f = \mathbb{Q}$ and $E_g = \{y^x : x \in \mathbb{Q}\}$.
(b) The exact rounding of $f(x)$ and $g(x)$ to grid $G = \mathbb{F}_n$ is solvable.

Proof. (a) We already know that $f(x) = y^x$ is transcendental with $E_f = \mathbb{Q}$. The transcendence of $g(x) = \log_y x$ with $E_g = \{y^x : x \in \mathbb{Q}\}$ follows from Lemma 3.
(b) To perform exact rounding for $f(x) = y^x$, we first check if $f(x) \in G$, and if so, we directly output $f(x)$. If not, we can use METHOD A.

To check if $f(x) \in G$, we proceed as follows: we check if $x \in \mathbb{Q}$. If not, $f(x) \notin G$. Otherwise let $x = a/b$ be a rational in lowest terms. If $f(x) = y^x \in G$, let $f(x) = m2^{-n}$ for some positive $m \in \mathbb{Z}$. We can compute some approximation $\widetilde{w} = 2^n y^x \pm 1/4$. Then $m = \lfloor \widetilde{w} \rfloor$ or $m = 1 + \lfloor \widetilde{w} \rfloor$. So we have to check if one of two cases hold: $2^n y^x = \lfloor \widetilde{w} \rfloor$ or $2^n y^x = 1 + \lfloor \widetilde{w} \rfloor$. Let us focus of testing the first case (the other case is similar). The height H of $E = 2^n y^x - \lfloor \widetilde{w} \rfloor$ is easy to determine [15]. It follows that if $E \neq 0$ then $|E| > 1/(1 + H)$. We compute an approximation $\widetilde{E} = E \pm (4(1 + H))^{-1}$. If $|\widetilde{E}| < (2(1 + H))^{-1}$, we know that $E = 0$, and we may output $f(x) = m2^{-n} \in G$. **Q.E.D.**

Finally, we address the two argument functions of $\arctan(x/y)$ and $\operatorname{arctanh}(x/y)$ in (5). Let $f(x) = \arctan(x/y)$ where $y \in \mathbb{A}$ is nonzero. By Corollary 2, $f(x)$ is a transcendental function. Moreover, its only exceptional argument is $x = 0$, and $f(0) = 0$. Thus METHOD A applies for rounding $f(x) \neq 0$. Similarly, we can treat $\operatorname{arctanh}(x/y)$.

All our algorithms above are based on METHOD A. It is also possible to base our algorithms on METHOD B but these require ε-discreteness properties. As we shall see in the next section, current estimates for ε (from TNT) is extremely pessimistic. So it is best to use METHOD A whenever possible.

5 Complexity of Exact Rounding

No complexity bounds can be deduced using only Precondition A. To deduce bounds, we need to invoke the ε-discreteness conditions of Precondition B (*even* when our algorithms are based on METHOD A). If $\Delta_\varepsilon(f(x), G)$ holds, the complexity of rounding $f(x)$ in G is basically the time to compute $f(x) \pm \varepsilon/2$. When f is an elementary function, Brent [1] tells us that the running time is $O(M(\log(1/\varepsilon)) \log(1/\varepsilon))$ where $M(n)$ is the time to multiply two n-bit numbers. One caveat is that Brent's result is "local", i.e., it is only applicable when x lies in a bounded range [20]. More global results are obtained in [6].

The ε-discreteness results are obtained from effective forms of the Lindemann or Gelfond-Schneider theorems, such as are provided by Baker's theory of linear form in logarithms [7]. In particular, we use some explicit bounds from Nesterenko and Waldschmidt [17], somewhat simplified.

Proposition 2 (Nesterenko-Waldschmidt, 1995). *Let $\alpha, \beta \in \mathbb{A}$ with $D = [\mathbb{Q}(\alpha, \beta) : \mathbb{Q}]$ and the heights of α and β are at most H. Define*

$$B_1(D, H) := 10550 \cdot D^2 A \cdot (H + \log(A) + \log(D) + 1)(D \log(D) + 1)$$

where $A = \max\{H, D^{-1}(1 + H)e\}$.

- *(i) If $\beta \neq 0$ then $|e^\beta - \alpha| \geq \exp(-B_1(D, H))$.*
- *(ii) If $\alpha \neq 0$ and $\log \alpha$ is any non-zero logarithm of α, then $|\beta - \log \alpha| \geq \exp(-B_1(D, H))$.*

Write $B_1(H) := B_1(1, H)$. If $D = 1$, then $A = (1 + H)e$ and we have

$$B_1(H) = 10550 \cdot (1 + H)e \cdot (H + \log(1 + H) + 2) \tag{6}$$

Since $H \geq 1$, we see that $B_1(H) > 10550$.

¶*5. Transfer Functions: Inhomogeneous Case.* To make our bounds as useful as possible, we state them as "transfer functions" that convert bounds such as $B_1(D, H)$ from TNT into ε-discreteness bounds. This is illustrated by the next lemma.

Lemma 4. *Let $G = \mathbb{Z}/N$ $(n \geq 0)$. If $x \in \mathbb{Q}$ $(x > 0)$ has height at most h_0, and $\log(x)$ is real, then*

$$\| \log(x) \|_G \geq \exp(-B_1(H))$$

where $H = \max\{h_0, N \log(e(1 + h_0))\}$.

Proof. Let $\Lambda_1 = g - \log x$ where $g \in G$ and $|\Lambda_1| = \| \log(x) \|_G$. So $\Lambda_1 \neq 0$. If $|\Lambda_1| \geq 1$, then our lemma is immediate. Otherwise, $|g| \leq 1 + |\log x| \leq 1 + \log(1 + h_0) = \log(e(1 + h_0))$. If $g = a/N \in \mathbb{F}_n$ then $h(g) \leq \max\{|a|, N\} \leq \max\{N \log(e(1 + h_0)), N\}$. So $h(g) \leq N \log(e(1 + h_0))$. Hence by Prop. 2(ii),

$$|g + \log x| \geq \exp(-B_1(H))$$

where $H = \max\{h(x), h(g)\} \leq \max\{h_0, N \log(e(1 + h_0))\}$. **Q.E.D.**

We can clearly obtain a similar transfer function for rounding the function $f(x) = e^x$. Suppose G is the IEEE double precision binary numbers and $x \in G$. Then $h_0 \leq 2^{1075}$ and $N = 2^{1075}$. Hence $H = 2^{1075}$. Thus it is clear that the bound $B_1(H)$ is nowhere near practical. On the other hand, the worst-case searches from Lefèvre et al [13, 14] gives very practical bounds for this case. This suggests that our ability to derive sharp transcendence bounds are rather limited.

¶*6. Transfer Function: Homogeneous Case.* An expression of the form

$$\Lambda = \beta_0 + \sum_{i=1}^{m} \beta_i \log \alpha_i$$

where $\alpha_i, \beta_i \in \mathbb{A}$ is called a "linear form in logarithms". The form is homogeneous when $\beta_0 = 0$; if, in addition, the β_i's are integers, then Λ is said to be in "algebraic form". This is the case (with $m = 2$) needed for bounding $\| \log_2(x) \|_G$.

Lemma 5. *Let $\Lambda_0 = \beta_1 \log \alpha_1 + \beta_2 \log \alpha_2$ where $\alpha_1, \alpha_2 \in \mathbb{A}$ and $\beta_1, \beta_2 \in \mathbb{Z}$. If $\alpha := {\alpha_1}^{\beta_1} {\alpha_2}^{\beta_2}$, then*

$$|\Lambda_0| \geq \exp\left(-B_1\left(D, H\right)\right). \tag{7}$$

where $D = [\mathbb{Q}(\alpha) : \mathbb{Q}]$ and $h(\alpha) = H$.

Proof. Note that $\alpha \in \mathbb{A}$ so that its degree D and height H is well-defined. Thus $|\Lambda_0| = |\beta_1 \log \alpha_1 + \beta_2 \log \alpha_2| = |\log \alpha|$. Then Prop. 2(ii), with $\beta = 0$, shows that $|\beta - \log \alpha| = |\log \alpha| \geq \exp(-B_1(D, H))$. **Q.E.D.**

We now apply this bound:

Lemma 6. *Let $G = \mathbb{Z}/N$ ($n \geq 0$). If $x \in \mathbb{Q}$ ($x > 0$) has height at most h_0, then*

$$\|\log_2(x)\|_G \geq \exp(-B_1(H))/N$$

where $H = 2^{2N \log(1+h_0)+2} h_0^N$.

Proof. Let $\Lambda_1 = g - \log_2 x$ where $|\Lambda_1| = \|\log_2 x\|_G$ and $g \in G$. So $Ng \in \mathbb{Z}$, and $(N \log 2)\Lambda_1 = \Lambda_0$ where

$$\Lambda_0 = Ng \log 2 - N \log x.$$

If $|\Lambda_0| \geq 1$, the lemma is true. Otherwise, it is easy to see that $|g| \leq 2/N + 2|\log(x)| \leq 2/N + 2\log(1 + h_0)$. We then use Lemma 5 to lower bound $|\Lambda_0|$. To do this, we need to bound the degree D and height H of $\alpha = e^{\Lambda_0}$. Clearly, $D = 1$. Moreover, $H = h(2^{Ng}x^{-N}) = h(2^{Ng})h(x^{-N}) \leq 2^{Ng} h_0^N \leq 2^{2N \log(1+h_0)+2} h_0^N$. **Q.E.D.**

¶7. *Non-algebraic Grid Points.* In evaluating the tangent function, it useful to know the relation of an input argument $x \in \mathbb{F}$ to multiples of $\pi/2$. The complexity of this comparison, can be deduced from the following result from [17, Theorem 2].

Proposition 3 (Nesterenko-Waldschmidt). *Let $L \geq 3$. Let ξ be a real algebraic number with $d = \deg \xi$ and $L(\xi) \leq L$. Then*

$$-\log|\pi - \xi| \leq B_\pi(d, L)$$

where $B_\pi(d, L) = 1.2 \cdot 10^6 d \cdot (\log L + d \log d)(1 + \log d)$.

We provide a corresponding transfer function:

Lemma 7. *If $G = \mathbb{F}_n = \mathbb{Z}/N$ then*

$$\|\pi\|_G \geq \exp(-B_\pi(5N))$$

Proof. The $\Lambda = g - \pi$ such that $g \in G$ and $|\Lambda| = \|\pi\|_G$. If $|\Lambda| \geq 1$, the result is immediate. Else, $|g| < 1 + \pi$. If $g = a/N$ then $L(g) \leq |a| + N \leq N(1 + \pi) + N < 5N$. **Q.E.D.**

¶*8. How to Use ε-discreteness for π.* Suppose we are given $x \in \mathbb{Z}/N$. We want to determine the sign of $\arctan(x)$. Suppose $\frac{m\pi}{2} < x < \frac{(m+1)\pi}{2}$. Then $\text{sign}(\arctan(x)) = 1 - 2 \cdot \text{parity}(m)$ where $\text{parity}(m) = 0$ if m is odd, and $\text{parity}(m) = 1$ if m is even. Thus, our goal is to determine m.

We first compute $\widetilde{m} = \frac{2x}{\pi} \pm 2^{-3}$. Let $m' = \lfloor m \rceil$ (rounding, with ties arbitrarily taken). If $|m' - \widetilde{m}| > 1/4$, we know that $m = \lfloor \widetilde{m} \rfloor$, so we can output $1 - 2 \cdot \text{parity}(\lfloor \widetilde{m} \rfloor)$.

Otherwise, notice that $m'\pi/2 > x$ iff $\pi > 2x/m'$. But $2x/m' \in \mathbb{Z}/(m'N)$. According to Lemma 7, $-\log|\pi - (2x/m')| \leq B_\pi(5m'N)$. Hence we compute $\widetilde{\pi} = \pi \pm 2^{-1-B_\pi(5m'N)}$. We then know that $\pi > 2x/m'$ iff $\widetilde{\pi} > 2x/m'$. The latter is easy to determine. If $\widetilde{\pi} > 2x/m'$, we output $1 - 2 \cdot \text{parity}(m')$. Otherwise, we output $2 \cdot \text{parity}(m') - 1$.

The above transfer functions illustrate the three types ε-discreteness bounds that are of interest. In a full paper, we will describe other transfer functions for the other elementary functions.

6 Conclusions

As this paper shows, the proper foundations for exact rounding are (1) the computational framework of arbitrary-precision approximation, and (2) the mathematical properties of transcendence.

We have shown that in the most important case of elementary functions, the exact rounding problem is solvable (thereby avoiding the Table Maker's Dilemma). However, pending much improved bounds from transcendental number theory, the worst-case complexity analysis of these algorithms is hopelessly pessimistic. Evidence (in the case of single and double precision IEEE number formats) suggests that the actual bounds are much better than we are able to prove. In any case, our algorithms based on METHOD A are naturally adaptive and runs in time (roughly) proportional to the actual bounds.

Acknowledgments

This work is supported by NSF Grant CCF-0728977.

References

1. R. P. Brent. Fast multiple-precision evaluation of elementary functions. *J. of the ACM*, 23:242–251, 1976.
2. H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. *Discrete Applied Mathematics*, 109(1-2):25–47, 2001.
3. D. Defour, G. Hanrot, V. Lefèvre, J.-M. Muller, N. Revol, and P. Zimmermann. Proposal for a standardization of mathematical function implementation in floating-point arithmetic. *Numerical Algorithms*, 37(1–4):367–375, 2004.

4. Z. Du. *Guaranteed Precision for Transcendental and Algebraic Computation made Easy*. Ph.D. thesis, New York University, Department of Computer Science, Courant Institute, May 2006. From `http://cs.nyu.edu/exact/doc/`.
5. Z. Du, M. Eleftheriou, J. Moreira, and C. Yap. Hypergeometric functions in exact geometric computation. In V.Brattka, M.Schoeder, and K.Weihrauch, editors, *Proc. 5th Workshop on Computability and Complexity in Analysis*, pages 55–66, 2002. Malaga, Spain, July 12-13, 2002. In Electronic Notes in Theoretical Computer Science, 66:1 (2002), `http://www.elsevier.nl/locate/entcs/volume66.html`.
6. Z. Du and C. Yap. Uniform complexity of approximating hypergeometric functions with absolute error. In S. Pae and H. Park, editors, *Proc. 7th Asian Symp. on Computer Math. (ASCM 2005)*, pages 246–249, 2006.
7. N. Fel'dman and Y. V. Nesterenko. *Number Theory IV: Transcendental Numbers*, volume 44 of *Encyclopaedia of Mathematical Sciences*. Springer-Verlag, Berlin, 1998. Translated from Russian by N. Koblitz.
8. L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2):13, 2007.
9. S. Gal and B. Bachelis. An accurate elementary mathematical library for the ieee floating point standard. *ACM Trans. on Math. Software*, 17(1):26–45, 1991.
10. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.
11. IEEE. ANSI/IEEE Standard 754-1985 for binary floating-point arithmetic, 1985. The Institute of Electrical and Electronic Engineers, Inc., New York.
12. K.-I. Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.
13. V. Lefèvre, J.-M. Muller, and A. Tisserand. Towards correctly rounded transcendentals. *IEEE Trans. Computers*, 47(11):1235–1243, 1998.
14. V. Lefèvre, D. Stehlé, and P. Zimmermann. Worst cases for the exponential function in the ieee 754r decimal64 format. In P. Hertling, C. Hoffmann, W. Luther, and N.Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, number 5045 in Lecture Notes in Computer Science, pages 114–126. Springer, 2008.
15. C. Li, S. Pion, and C. Yap. Recent progress in Exact Geometric Computation. *J. of Logic and Algebraic Programming*, 64(1):85–111, 2004. Special issue on "Practical Development of Exact Real Number Computation".
16. J.-M. Muller. *Elementary Functions: Algorithms and Implementation*. Birkhäuser, Boston, 1997.
17. Y. Nesterenko and M. Waldschmidt. On the approximation of the values of exponential function and logarithm by algebraic numbers. *Mat. Zapiski*, 2:23–42, 196. Diophantine Approximations, Proc. of papers dedicated to the memory of Prof. N.I. Feldman, Moscow. Available from http://arxiv.org/abs/math.NT/0002047.
18. I. S. O. ISO/IEC 10967-2-2001 for Language Independne Arithmetic: Elementary Numerical Functions, 2001. International Standards Organisation, Geneva, Switzerland.
19. D. Richardson. How to recognize zero. *J. of Symbolic Computation*, 24:627–645, 1997.
20. V. Sharma, Z. Du, and C. Yap. Robust approximate zeros. In G. S. Brodal and S. Leonardi, editors, *Proc. 13th European Symp. on Algorithms (ESA)*, volume 3669 of *Lecture Notes in Computer Science*, pages 874–887. Springer-Verlag, Apr. 2005. Palma de Mallorca, Spain, Oct 3-6, 2005.

21. C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000.
22. C. K. Yap. On guaranteed accuracy computation. In F. Chen and D. Wang, editors, *Geometric Computation*, chapter 12, pages 322–373. World Scientific Publishing Co., Singapore, 2004.
23. C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapman & Hall/CRC, Boca Raton, FL, 2nd edition, 2004.
24. C. K. Yap. Theory of real computation according to EGC. In P. Hertling, C. Hoffmann, W. Luther, and N.Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, number 5045 in Lecture Notes in Computer Science, pages 193–237. Springer, 2008.
25. A. Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. on Math. Software*, 17(3):410–423, 1991.