Searching Worst Cases of a One-Variable Function Using Lattice Reduction

Damien Stehlé, Vincent Lefèvre, and Paul Zimmermann

Abstract—We propose a new algorithm to find worst cases for the correct rounding of a mathematical function of one variable. We first reduce this problem to the *real small value problem*—i.e., for polynomials with real coefficients. Then, we show that this second problem can be solved efficiently by extending Coppersmith's work on the *integer small value problem*—for polynomials with integer coefficients—using lattice reduction. For floating-point numbers with a mantissa less than N and a polynomial approximation of degree d, our algorithm finds all worst cases at distance less than $N^{\frac{-d^2}{2d+1}}$ from a machine number in time $O(N^{\frac{d+1}{2d+1}+\varepsilon})$. For $d = 2$, a detailed study improves on the $O(N^{2/3+\varepsilon})$ complexity from Lefèvre's algorithm to $O(N^{4/7+\varepsilon})$. For larger d, our algorithm can be used to check that there exist no worst cases at distance less than N^{-k} in time $O(N^{1/2+\varepsilon})$.

Index Terms—Computer arithmetic, multiple precision arithmetic, special function approximations.

1 Introduction

The IEEE-754 standard for binary floating-point arithmetic [11] requires that all four basic arithmetic operations ($+$, $-$, $\times$, $\div$) and the square root are correctly rounded. For a given function, floating-point inputs for which it is difficult to guarantee correct rounding, called *worst cases*, are numbers for which the exact result—as computed in infinite precision—is near a machine number or near the middle of two consecutive machine numbers. This is the famous "Table Maker's Dilemma" problem (TMD for short). Several authors [12], [13] have shown that, for the class of algebraic functions, such worst cases cannot be too close to a machine number or the middle of two consecutive machine numbers. Such bounds enable us to design some efficient algorithms that guarantee correct rounding for division and square root and less efficient algorithms for other algebraic functions.

However, for transcendental functions, number theoretic bounds—when they exist—are not sharp enough, which makes correct rounding much harder to implement. Defour et al. proposed in [7] to introduce different levels of quality for transcendental functions. However, an efficient implementation of Levels 1 and 2—correct rounding in the whole domain where the function is mathematically defined for Level 2 and in a subdomain for Level 1—requires first solving the TMD problem, i.e., to determine the worst cases for the given function in the given floating-point format.

Systematic work on the TMD was done by Lefèvre and Muller [15], who published worst cases for many elementary functions in double precision, over the full range for some functions. Alas, their approach is too expensive to deal with quadruple precision, which is included in the revision of the IEEE-754 standard. Thus, the only possible approaches for higher precisions are either to guess a reasonable bound on the precision required for the hardest to round cases and to write a library computing up to that precision or to write a generic multiple-precision library. For instance, Ziv's MathLib library does the former, where the guessed bound is 768 bits for double precision [20].

Having an efficient algorithm to find the hardest to round cases would help to replace guessed bounds—which are usually overestimated—by sharper and rigorous bounds. It would thus enable the design of very efficient libraries with correct rounding [6]. Then, there would no longer be a good reason to exclude those functions from the correct rounding requirements of the IEEE-754 standard.

Exhaustive search methods consist of finding the hardest to round cases of a given function in a given floating-point format. They give the best possible bound, but are very time-consuming. Moreover, a search for a given precision gives little knowledge for another precision. We propose here a new algorithm belonging to that class. It naturally extends Lefèvre's algorithm [14] and is based on Coppersmith's ideas.

Previous related work was done by Elkies, who gave in [8] a new algorithm using lattice reduction to find all rational points of small height near a plane curve; for example, his record:

$$5{,}853{,}886{,}516{,}781{,}223^3 - 447{,}884{,}928{,}428{,}402{,}042{,}307{,}918^2 = 1{,}641{,}843$$

corresponds to a worst case of the function $x^{3/2}$ for a 53-bit input and a 79-bit output; his other example:

$$2{,}220{,}422{,}932^3 - 283{,}059{,}965^3 - 2{,}218{,}888{,}517^3 = 30$$

corresponds to a worst case of $(x^3 + y^3)^{1/3}$ in 32-bit arithmetic. Gonnet [9] also used lattice reduction to find worst cases, however, his approach seems equivalent to Lefèvre's algorithm.

• The authors are with LORIA/INRIA Lorraine, Technopôle de Nancy-Brabois, 615 rue du jardin botanique, F-54602 Villers-lès-Nancy Cedex, France. E-mail: {stehle, lefevre, zimmerma}@loria.fr.

Manuscript received 1 Dec. 2003; revised 7 May 2004; accepted 16 Sept. 2004; published online 18 Jan. 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-0244-1203.

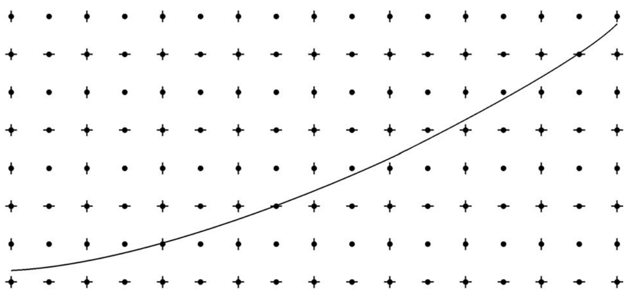


Fig. 1. A function graph and the grid of machine numbers. Worst cases correspond to grid points with a small vertical distance to the curve.

A preliminary version of this paper was presented at Arith 16 [18] and the technique described has also been used to break a cryptographic scheme proposed by Littlewood in the 1950s [19].

The paper is organized as follows: Section 2 explains in mathematical terms the problem we want to solve, recalls Lefèvre's algorithm, and analyzes its complexity. Section 3 is a short survey on lattice reduction and Coppersmith's work. Section 4 is the core of the paper: The new algorithm is described, proven, and analyzed. Section 5 explains in detail how the new algorithm can be tuned for the best practical choice of parameters. Section 6 presents experimental results for the 2^x function, in double, double-extended, and quadruple precisions.

2 Preliminaries

2.1 Definitions and Notations

In the following, we consider floating-point numbers with a mantissa of n bits. Let $N = 2^n$; for instance, $N = 2^{53}$ corresponds to double precision, $N = 2^{64}$ corresponds to double-extended precision, and $N = 2^{113}$ corresponds to quadruple precision. A worst case for a function f is a floating-point number x such that $f(x)$ has m identical bits after the round bit. If all those m bits equal (respectively, differ from) the round bit, x is a worst case for the directed rounding modes (respectively, for rounding-to-nearest mode).

For the sake of simplicity, we consider directed rounding only (toward $-\infty$, toward $+\infty$, toward zero) since worst cases at precision n for all rounding modes are worst cases at precision $n+1$ for directed rounding. To find worst cases for directed rounding, we throw away the first n significant bits of the result mantissa and we look for runs of at least m zeros or ones in the following bits. Equivalently, a worst case of length m corresponds to $|Nf(x) \bmod 1| < 2^{-m}$, where $x \bmod 1 := x - \lfloor x \rceil$ denotes the "centered" fractional part (see Fig. 1).

We also assume that both argument x and result $y = f(x)$ are normalized, i.e., $\frac{1}{2} \leq x, f(x) < 1$. This is easy to achieve by multiplying x or $f(x)$ by some fixed powers of 2, unless the exponent of $f(x)$ varies a lot in the considered range. This excludes the case of numerically irregular functions like $\sin x$ for large x. More generally, in the remainder of the paper, we assume that $f^{(i)}(x) = O(1)$ for any $i \geq 0$ and any x in the considered interval. This condition is verified in general, but it should be noted that there are cases where it is not fulfilled (for example, $\exp$ with large exponents).

Given a polynomial approximation $P(t)$ to $Nf(\frac{t}{N})$—for example, a Taylor expansion—the TMD problem can be reduced to the following:

Real Small Value Problem (Real SValP). Given positive integers M and T and a polynomial P with real coefficients, find all integers $|t| < T$ such that:

$$|P(t) \bmod 1| < \frac{1}{M}. \qquad (1)$$

Remark 1. The mantissa bound N does not appear explicitly in the real SValP, however, the polynomial $P(t)$ depends on N and so does the error made in the polynomial approximation.

Remark 2. If the fractional bits of the function behave randomly, we can expect $\approx \frac{T}{M}$ worst cases. Therefore, we may assume $T \ll M$ if we want only a few worst cases. The notation $x \ll y$ is equivalent to $x = O(y)$.

2.2 Lefèvre's Algorithm

Lefèvre's algorithm [14], [15] works as follows: One considers linear approximations to the function f on small intervals. Those approximations are computed from higher order polynomial approximations on larger intervals, using an efficient scheme based on the "table of differences" method. On each small interval, worst cases are found using a modified version of the Euclidean algorithm, which gives a lower bound for $|Nf(\frac{t}{N}) \bmod 1|$ on that interval.

Assume $f(x_0 + x) = a_0 + a_1 x + a_2 x^2 + O(x^3)$ around x_0. Since we neglect terms of order two or more in $Nf(\frac{t}{N})$, we need $|a_2 \frac{T^2}{N}| \ll \frac{1}{M}$ so that the error coming from the polynomial approximation does not exceed the distance $\frac{1}{M}$. Together with $T \ll M$, it follows $T \ll N^{1/3}$. Therefore, the complexity of Lefèvre's algorithm is $O(N^{2/3+\varepsilon})$ since we have to consider $\frac{N}{T} \approx N^{2/3}$ small intervals to check a complete mantissa range.

In practice, Lefèvre's algorithm is expensive but still feasible for double precision ($N^{2/3} \approx 4 \cdot 10^{10}$), near the limits of current processors for double-extended precision ($N^{2/3} \approx 7 \cdot 10^{12}$), and out of reach for quadruple precision ($N^{2/3} \approx 5 \cdot 10^{22}$).

3 Lattice Reduction and Coppersmith's Technique

In this section, we first state some basic facts about lattices—we refer to [17] for an introduction to that topic—and we explain Coppersmith's technique, on which our algorithm is based.

3.1 Some Basic Facts in Lattice Reduction Theory

A *lattice* L is a discrete subgroup of $\mathbb{R}^n$ or, equivalently, the set of all integer linear combinations of $\ell \leq n$ linearly independent vectors $\mathbf{b}_i$ over $\mathbb{R}$, that is:

$$L = \left\{ \sum_{i=1}^{\ell} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}.$$

We define the *determinant*, also called the *volume*, of the lattice L as: $\det(L) = \prod_{i=1}^{\ell} ||\mathbf{b}_i^*||$, where $||.||$ is the Euclidean norm and $[\mathbf{b}_1^*, \ldots, \mathbf{b}_\ell^*]$ is the Gram-Schmidt orthogonalization of $[\mathbf{b}_1, \ldots, \mathbf{b}_\ell]$. The *basis* $[\mathbf{b}_1, \ldots, \mathbf{b}_\ell]$ of L is not unique and, from an algorithmic point of view, only bases which consist of short linearly independent vectors of L are of interest. Those so-called *reduced bases* always exist and can be computed in polynomial time with the well-known LLL algorithm [16].

Theorem 1. *Given a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_\ell]$ of a lattice $L \subset \mathbb{Z}^n$, the LLL algorithm provides in time polynomial in the bit length of the input, a basis $[\mathbf{v}_1, \ldots, \mathbf{v}_\ell]$ satisfying:*[1]

1. $||\mathbf{v}_1|| \leq 2^{\ell/2} \det(L)^{1/\ell}$,
2. $||\mathbf{v}_2|| \leq 2^{\ell/2} \det(L)^{1/(\ell-1)}$.

Coppersmith (see [3], [4], or [5] for an overall description) found an important consequence of this theorem: One can compute, in time polynomial in $\log A$, the small roots of a multivariate polynomial modulo an integer A. His method proved very powerful to factorize integers when some bits of the factors are known and to forge some cryptographic schemes (see [1], [2], [3], for example). The algorithm described in Section 4 intensively uses that technique.

3.2 The Integer Small Value Problem

The problem that will prove relevant in our case is the following: Given a univariate polynomial $P \in \mathbb{Z}[x]$ of degree d, find on which small integer entries it has small values modulo a large integer A. Equivalently, we are looking for the small integer roots of the bivariate polynomial:

$$Q(x, y) = P(x) + y \pmod{A}.$$

We now explain how Coppersmith's technique helps to solve it. First, let α be a positive integer (which will later grow to infinity) and assume (x_0, y_0) is a root of Q modulo A. We consider the family of polynomials $Q_{i,j}(x, y) = x^i Q^j(x, y) A^{\alpha - j}$ with $0 \leq i + dj \leq d\alpha$. Then, (x_0, y_0) is a root modulo A^α of each $Q_{i,j}$, whence of each integer linear combination of them.

Our goal is to build two integer combinations of those polynomials, $v_1(x, y)$ and $v_2(x, y)$, which take small values —i.e., less than A^α—for any small x and y, more precisely, $|x| \leq X$ and $|y| \leq Y$ for some fixed bounds X and Y. Thus, if (x_0, y_0) is a small root of v_1 and v_2 modulo A^α, (x_0, y_0) is also a root of v_1 and v_2 over $\mathbb{Z}$. Finally, x_0 will be found by looking at the integer roots of the resultant $\mathrm{Res}_y(v_1, v_2) \in \mathbb{Z}[x]$.

It remains to explain how to find those two polynomials. For this, we consider the lattice of dimension $\frac{(\alpha+1)(d\alpha+2)}{2}$ spanned by the vectors associated with the polynomials $Q_{i,j}(X\tau, Y\upsilon)$: The vector linked to a bivariate polynomial $\sum_{i,j} a_{i,j} \tau^i \upsilon^j$ has its $\tau^i \upsilon^j$ coordinate equal to $a_{i,j}$. We give here the shape of the matrix we get in the case $d = 3$ and $\alpha = 2$.

$$\begin{array}{c|cccccccccccc}
i/j & & & & & & & & & & & & \\
0/0 & A^2 & & & & & & & & & & & \\
1/0 & & A^2X & & & & & & & & & & \\
2/0 & & & A^2X^2 & & & & & & & & & \\
3/0 & & & & A^2X^3 & & & & & & & & \\
4/0 & & & & & A^2X^4 & & & & & & & \\
5/0 & & & & & & A^2X^5 & & & & & & \\
6/0 & & & & & & & A^2X^6 & & & & & \\
0/1 & - & - & - & - & & & & AY & & & & \\
1/1 & & - & - & - & - & & & & AXY & & & \\
2/1 & & & - & - & - & - & & & & AX^2Y & & \\
3/1 & & & & - & - & - & - & & & & AX^3Y & \\
0/2 & - & - & - & - & - & - & - & - & - & - & - & Y^2
\end{array}.$$

Since we get a triangular matrix, the calculation of the determinant is obvious:[2]

$$\det(L) = A^{d\alpha^3/3 + o(\alpha^3)} X^{d^2\alpha^3/6 + o(\alpha^3)} Y^{d\alpha^3/6 + o(\alpha^3)},$$
$$\dim(L) \approx d\alpha^2/2.$$

Therefore, by Theorem 1, the LLL algorithm gives us two vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, of norms that are asymptotically below $A^{2\alpha/3 + o(\alpha)} X^{d\alpha/3 + o(\alpha)} Y^{\alpha/3 + o(\alpha)}$. Those vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ correspond to two polynomials $v_1(X\tau, Y\upsilon)$ and $v_2(X\tau, Y\upsilon)$. Moreover, if $|x| \leq X$ and $|y| \leq Y$, then

$$|v_k(x, y)| \leq \sum_{i,j} \left| v_{i,j}^{(k)} X^i Y^j \right| \frac{|x|^i |y|^j}{X^i Y^j} \leq c \cdot \max \left| v_{i,j}^{(k)} X^i Y^j \right|$$
$$\leq c \cdot ||\mathbf{v}_k||$$

for a certain constant c. Thus, to get $|v_k(x, y)| < A^\alpha$, it is sufficient that:

$$c \cdot A^{2\alpha/3 + o(\alpha)} \cdot X^{d\alpha/3 + o(\alpha)} \cdot Y^{\alpha/3 + o(\alpha)} < A^\alpha,$$

which asymptotically gives the bound $X^d Y \ll A$.

Using Coppersmith's technique, one can thus solve the integer SValP in polynomial time as long as $X^d Y < A^{1-\epsilon}$. In fact, this is not completely true because we used an argument we cannot prove: We assumed that $\mathrm{Res}_y(v_1, v_2) \neq 0$. This assumption is frequently made in cryptography (see [1], [2], [3]).

4 The New Algorithm and Its Analysis

In this section, we present the new algorithm, prove its correctness, and analyze its complexity.

4.1 The SLZ Algorithm

The real SValP is the following problem: Given a polynomial P, find for which integers t, $P(t)$ is near an integer. We solve this problem by reducing it to the integer SValP. The difficulty is that $P(t)$ has real coefficients and the LLL algorithm does not work well with real input. The following algorithm overcomes that difficulty (we present here a complete algorithm to solve the Table Maker's Dilemma, in which case, $P(t) = Nf(t/N)$, but the subalgorithm consisting of Steps 3 to 11 may be of interest to solve the real SValP itself).

1. This is not the strongest result, but is sufficient for our needs.

2. The $o()$ notation is meant when α grows to infinity.

Algorithm SLZ.
Input: a function f, positive integers N, T, M, d, α
Output: all $t \in [-T, T]$ such that $|Nf(\frac{t}{N}) \bmod 1| < 1/M$

```
1.  Let $P(t)$ be the Taylor expansion of $Nf(\frac{t}{N})$ up to order $d$, and $n = \frac{(\alpha+1)(d\alpha+2)}{2}$
2.  Compute $\varepsilon$ such that $|P(t) - Nf(\frac{t}{N})| < \varepsilon$ for $|t| \leq T$
3.  Let $M' = \lfloor \frac{1/2}{1/M+\varepsilon} \rfloor$, $C = (d+1)M'$, and[3] $\tilde{P}(\tau) = \lfloor CP(T\tau) \rceil$
4.  Let $\{e_1, \ldots, e_n\} \leftarrow \{\tau^i v^j\}$ for $0 \leq i + dj \leq d\alpha$
5.  Let $\{g_1, \ldots, g_n\} \leftarrow \{(T\tau)^i (\tilde{P}(\tau) + (d+1)v)^j C^{\alpha-j}\}$ for $0 \leq i + dj \leq d\alpha$
6.  Form the $n \times n$ integral matrix $L$ where $L_{k,l}$ is the coefficient of the monomial $e_k$ in $g_l$
7.  $V \leftarrow \mathrm{LatticeReduce}(L)$
8.  Let $\mathbf{v}_1, \mathbf{v}_2$ be the two smallest vectors from $V$, $Q_i(T\tau, v)$ the corresponding polynomials for $i = 1, 2$
9.  If there exists $(t, v) \in [-T, T] \times [-1, 1]$ with $|Q_1(t, v)| \geq C^\alpha$ or $|Q_2(t, v)| \geq C^\alpha$, return(FAIL)
10. $p(t) \leftarrow \mathrm{Res}_v(Q_1(t, v), Q_2(t, v))$
    if $p(t) = 0$ then return(FAIL)
11. For each $t_0$ in $\mathrm{IntegerRoots}(p(t), [-T, T])$ do
        if $|Nf(\frac{t_0}{N}) \bmod 1| < \frac{1}{M}$ then output $t_0$.
```

4.2 Correctness of the Algorithm

Theorem 2. *In case algorithm SLZ does not return FAIL, it behaves correctly, i.e., it outputs exactly all integers $t \in [-T, T]$ such that $|Nf(\frac{t}{N}) \bmod 1| < 1/M$.*

Proof. Because of the final check in Step 11, we only have to verify that no worst case is missed. Suppose there is $t_0 \in [-T, T]$ with $|Nf(\frac{t_0}{N}) \bmod 1| < 1/M$. From the definition of P, $|P(t_0) \bmod 1| < 1/M + \varepsilon \leq \frac{1}{2M'}$. Since $|CP(T\tau) - \tilde{P}(\tau)| \leq \frac{d+1}{2}$ for $|\tau| \leq 1$, by choosing $\tau = t_0/T$, we get $|\tilde{P}(t_0/T) \bmod C| < d + 1$.

Whence $\tilde{P}(t/T) + (d+1)v = 0 \bmod C$ has a root (t_0, v_0) with $|t_0| \leq T$ and $|v_0| < 1$. Since $Q_1(t, v)$ and $Q_2(t, v)$ are integer linear combinations of the g_is, then (t_0, v_0) is a common root of $Q_1(t, v)$ and $Q_2(t, v)$ modulo C^α and even over the reals since $|Q_1|, |Q_2| < C^\alpha$. Thus, t_0 is an integer root of $\mathrm{Res}_v(Q_1(t, v), Q_2(t, v))$ and will be found at Step 11. □

4.3 Choice of Parameters and Complexity Analysis

4.3.1 Coppersmith's Bound

Because of the use of Coppersmith's technique in our algorithm, to ensure the algorithm does not return FAIL at Step 9, the bound "$X^d Y \ll A$" has to be verified. In our case, X corresponds to T, Y to $(d+1)$, and A to C, so we get:

$$T \ll M^{1/d}.$$

4.3.2 Choice of the Degree d with Respect to T

Let $a_0, a_1, \ldots$ be the Taylor coefficients of f. Since we neglect Taylor coefficients of degree $d+1$ and greater, the error made in the approximation to $Nf(\frac{t}{N})$ by $P(t)$ is $\approx a_{d+1} T^{d+1} N^{-d}$. Since we are looking for worst cases with $|P(t) \bmod 1| < 1/M$, we want $T^{d+1} N^{-d} \ll 1/M$, i.e., $T^{d+1} \ll N^d/M$.

4.3.3 Complexity Analysis

We have two bounds for T: The first one, $T \ll M^{1/d}$, comes from Coppersmith's method, the second one, $T^{d+1} \ll N^d/M$, comes from the accuracy of the Taylor expansion. For $M \ll N^{\frac{d^2}{2d+1}}$, Coppersmith's bound wins and implies $T \ll M^{1/d}$, whereas, for $M \gg N^{\frac{d^2}{2d+1}}$, Taylor's bound gives $T^{d+1} \ll N^d/M$. The largest bound for T is obtained for $M \approx N^{\frac{d^2}{2d+1}}$, with $T \ll N^{\frac{d}{2d+1}}$. For $d = 1$, we find the constraint $T \ll N^{1/3}$ from Lefèvre's method; for $d = 2$, this gives $T \ll N^{2/5}$ with $M \approx N^{4/5}$; for $d = 3$, this gives $T \ll N^{3/7}$ with $M \approx N^{9/7}$. With $M \approx N^k$, we get a best possible interval length $T \approx N^{1/2 - 1/(8k) + o(1/k)}$.

4.3.4 Working Precision

In Step 1, we can use floating-point coefficients in the Taylor expansion $P(t)$ instead of symbolic coefficients as long as it introduces no error in Step 3 while computing $\tilde{P}(\tau)$. Let a_i be the ith Taylor coefficient of f. Then, to get $\tilde{P}(\tau)$ correct at Step 3, the error on $CN(T/N)^i a_i$ must be less than $1/2$, thus the error on a_i must be less than $1/(2CN)(N/T)^i$. Since $N \geq T$, it thus suffices to compute a_i with $\log_2(2CN) \leq 2n$ bits after the binary point.

Remark 3. When searching worst cases with $M \ll N$, degree-2 approximations suffice. Indeed, $N^{1-d} T^d \ll N^{1-d} M$ since $T^d \ll M$ (Coppersmith's bound) and, for $d \geq 3$, $N^{1-d} T^d \ll N^{2-d} \ll 1/N \ll 1/M$. Thus, all Taylor terms of degree ≥ 3 give a negligible contribution to $Nf(\frac{t}{N})$ and the largest value of T is $\approx N^{2/5}$, giving a complexity $\approx N^{3/5}$ to search a whole range of $N/2$ values. More generally, for $M \ll N^k$, degree-$2k$ approximations suffice, giving a complexity $\approx N^{\frac{2k+1}{4k+1}}$.

4.3.5 About the Size of the Coefficients of $\tilde{P}$

Here, we study the size of the coefficients of the polynomial $\tilde{P}$ obtained at Step 3 of the algorithm and used in the lattice reduction.

Theorem 3. *Suppose that the derivatives of f satisfy $|f^{(i)}(x)| = O(1)$ and that T is chosen to reach Taylor's bound, i.e., $T^{d+1} = \Theta(N^d/M)$. Then, the degree-d polynomial $\tilde{P}(\tau) = \tilde{p}_0 + \tilde{p}_1 \tau + \ldots + \tilde{p}_d \tau^d \in \mathbb{Z}[\tau]$ computed at Step 3 of the algorithm satisfies $\tilde{p}_i = O((N/T)^{d-i+1})$ for any $1 \leq i \leq d$.*

Proof. Let $1 \leq i \leq d$. Since $P(t) = p_0 + p_1 t + \ldots + p_d t^d$ is the degree-d Taylor expansion of $Nf(\frac{t}{N})$, we have $p_i = O(N^{1-i})$. $\tilde{P}$ is defined by

$$\tilde{P}(\tau) = \tilde{p}_0 + \tilde{p}_1 \tau + \ldots + \tilde{p}_d \tau^d = \lfloor CP(T\tau) \rceil.$$

Since $C = \Theta(M)$, we have $\tilde{p}_i = O(MT^i N^{1-i})$. Because T is chosen to reach Taylor's bound, we have $M = \Theta(N^d T^{-d-1})$, which concludes the proof. □

This result highlights the fact that the obtained instantiations of the integer SValP are very special in the sense that the high degree coefficients of the input polynomial are far smaller than expected. As we show in the next section, this weakness of the high degree coefficients can be used in order to improve Coppersmith's bound and, therefore, to obtain a better complexity bound.

3. The notation $\lfloor CP(T\tau) \rceil$ means that we round to the nearest integer each coefficient of $CP(T\tau)$. This gives an element of $\mathbb{Z}[\tau]$.

5 A Detailed Study of the Case $d = \alpha = 2$

This section gives improvements and details about the case $d = \alpha = 2$. This case is the smallest one that gives a bound for T larger than $N^{1/3}$. Indeed, $d = 1$ corresponds to Lefèvre's algorithm and $(d, \alpha) = (2, 1)$ gives a lattice of the form:

$$\begin{pmatrix} C & & & \\ & TC & & \\ & & T^2C & \\ a & b & c & 3 \end{pmatrix},$$

of determinant $3T^3C^3$, thus $T^3 \ll C$ is required to get sufficiently small vectors. Together with $C \approx M$ and Taylor's bound, this implies $T \ll N^{1/3}$.

5.1 The $d = \alpha = 2$ Lattice

If $f(x) = a_0 + a_1x + a_2x^2 + O(x^3)$ is the Taylor expansion of f at $x = 0$, we have:

$$\tilde{P}(\tau) = a + b\tau + c\tau^2,$$

with $a = \lfloor CNa_0 \rceil$, $b = \lfloor CTa_1 \rceil$, and $c = \lfloor CT^2a_2/N \rceil$. We have to reduce the lattice spanned by the rows of the following matrix:

$$\begin{pmatrix} C^2 & & & & & & & & \\ & TC^2 & & & & & & & \\ & & T^2C^2 & & & & & & \\ & & & T^3C^2 & & & & & \\ & & & & T^4C^2 & & & & \\ C & & Cb & Cc & & 3C & & & \\ & TCa & TCb & TCc & & & 3TC & & \\ & & T^2Ca & T^2Cb & T^2Cc & & & 3T^2C & \\ a^2 & 2ab & 2ac{+}b^2 & 2bc & c^2 & 6a & 6b & 6c & 9 \end{pmatrix}.$$

Assuming $a_2 = \Theta(1)$, T is chosen to reach Taylor's bound, and $C = \Theta(M)$, we find $c = O(\sqrt{MT})$ and it follows from a simple determinant calculation that Coppersmith's bound for this matrix is $T \ll N^{5/14}$. As explained in Section 4.3.5, c is weaker than it could be for a general instantiation of the integer SValP. This fact is used below to improve Coppersmith's bound to $T \ll N^{5/13}$ and to decrease the dimension of the lattice from 9 to 5, which significantly improves the efficiency of the lattice reduction.

5.2 Improving Coppersmith's Bound

It follows from the weakness of c that a certain subset of the rows of the above 9×9 matrix defines a sublattice with a nonzero vector far shorter than guaranteed by Theorem 1. More precisely, we erase rows 4, 5, and 8 and permute columns to obtain the nonsquare matrix:

$$L = \begin{pmatrix} C^2 & & & & & & & & \\ & TC^2 & & & & & & & \\ & & T^2C^2 & & & & & & \\ Ca & Cb & Cc & 3C & & & & & \\ & TCa & TCb & & TCc & 3TC & & & \\ a^2 & 2ab & 2ac + b^2 & 6a & 2bc & 6b & c^2 & 6c & 9 \end{pmatrix}.$$

We have the following result:

Theorem 4. *Given as input the rows of the matrix L above, the LLL algorithm outputs a vector $\mathbf{v}$ of length $\ll M^{19/12}T^{11/12}$.*

Proof. From Theorem 1, we know that, to obtain such a bound, it is sufficient to evaluate the determinant of the lattice. Recall that the determinant of a lattice spanned by $\mathbf{b}_1, \dots, \mathbf{b}_k$ is $\Pi_{i=1}^{k} ||\mathbf{b}_i^*||$, where the $\mathbf{b}_i^*$s are the Gram-Schmidt vectors associated with the $\mathbf{b}_i$s and, in particular, a lattice need not be written as a square matrix to have a determinant. Let $\mathbf{v}_1, \dots, \mathbf{v}_6$ be the rows of L. Obviously, $||\mathbf{v}_1^*|| = C^2$, $||\mathbf{v}_2^*|| = TC^2$, $||\mathbf{v}_3^*|| = T^2C^2$, and $||\mathbf{v}_4^*|| = 3C$. Since $\mathbf{v}_5^* = (0, 0, 0, 0, TCc, 3TC, 0, 0, 0)$, and c is large,

$$||\mathbf{v}_5^*|| = O(M^{3/2}T^{3/2}).$$

Finally, $\mathbf{v}_6^* = (0, 0, 0, 0, 0, 0, c^2, 6c, 9)$, which gives that $||\mathbf{v}_6^*|| = O(MT)$. As a consequence, we have

$$\det(L) = O(M^{19/2}T^{11/2}),$$

which ends the proof. □

Recall that Coppersmith's bound is derived from the inequality $||\mathbf{v}|| \ll C^{\alpha}$ so that, in our case, we obtain the improved bound $T \ll M^{5/11}$, i.e., $T \ll N^{5/13}$, which gives a complexity bound $O(N^{8/13})$.

5.3 Improving the Lattice Reduction Step

We now give some technical improvements for the lattice reduction step. Notice first that, in order to obtain a short vector of the lattice spanned by the rows of L, it is sufficient to reduce the rows of the matrix L' given by:

$$L' = \begin{pmatrix} C^2 & & & & & \\ & TC^2 & & & & \\ & & T^2C^2 & & & \\ & TCa & TCb & TCc & & \\ Ca & Cb & Cc & & 3C & \\ a^2 & 2ab & 2ac + b^2 & 2bc & 6a & c^2 \end{pmatrix}.$$

Let $\mathbf{v}'_1, \dots, \mathbf{v}'_6$ be the rows of L'. Then, for any $x_1, \dots, x_6$, the vector $x_1\mathbf{v}'_1 + \dots + x_6\mathbf{v}'_6$ is short in L' if and only if $x_1\mathbf{v}_1 + \dots + x_6\mathbf{v}_6$ is short in L. From a short vector $\mathbf{v}'$ of L', it is easy to recover a short vector in L since the missing coordinates are proportional to some of the remaining ones.

Moreover, the first row and the first column of L' can be erased, too. Indeed, suppose a short integer linear combination $(x_2, \dots, x_6)$ of the truncations of the remaining rows is found. With the notations of Section 3.2, this gives a bivariate polynomial $x_2Q_{1,0} + x_3Q_{2,0} + x_4Q_{0,1} + x_5Q_{1,1} + x_6Q_{0,2}$. Then, we reduce the constant term of this polynomial modulo C^2 and the remainder of the SLZ algorithm does not change.

It is thus sufficient to reduce the rows of the following 5×5 square matrix, with $T \approx N^{5/13}$, and $C, M \approx N^{11/13}$, $a \approx N^{11/13}$, $b \approx N^{16/13}$, $c \approx N^{8/13}$:

	N	M	T	est. time
double	2^{53}	2^{53}	2^{20}	11 days
double-extended	2^{64}	2^{64}	2^{24}	4 years
quadruple	2^{113}	2^{113}	2^{43}	11 Gyears

Fig. 2. The best experimental parameters for double, double-extended, and quadruple precision and the estimated cpu time for an exponent range of $N/2$ values on a 3GHz Pentium 4 running Linux for the 2^x function.

$$\begin{pmatrix} TC^2 & & & & \\ & T^2C^2 & & & \\ TCa & TCb & TCc & & \\ Cb & Cc & & 3C & \\ 2ab & 2ac+b^2 & 2bc & 6a & c^2 \end{pmatrix} \approx \begin{pmatrix} N^{27/13} & & & & \\ & N^{32/13} & & & \\ N^{27/13} & N^{32/13} & N^{24/13} & & \\ N^{27/13} & N^{19/13} & & N^{11/13} & \\ N^{27/13} & N^{32/13} & N^{24/13} & N^{11/13} & N^{16/13} \end{pmatrix}.$$

The LLL algorithm can be tuned for these particular lattices. In particular, the entries of the matrix can be truncated because the most significant bits suffice to find the same transformation matrix.

5.4 The General $d = 2$ Case

For $d = 2$, the lattice L from algorithm SLZ is spanned by the vectors

$$(T\tau)^i (a + b\tau + c\tau^2 + 3v)^j C^{\alpha-j}, \quad 0 \le i + 2j \le 2\alpha,$$

where $\tilde{P}(\tau) = a + b\tau + c\tau^2$. Keeping only the vectors with $i + j \le \alpha$ and decomposing the monomials by blocks where $i + 2j$ is constant, we find a sublattice of dimension $(\alpha + 1)(\alpha + 2)/2$ and determinant:

$$d \approx \begin{cases} (C^2T)^{\frac{\alpha(\alpha+1)(\alpha+2)}{6}} c^{\frac{\alpha(\alpha+2)(2\alpha+5)}{24}} & \text{for } \alpha \text{ even,} \\ (C^2T)^{\frac{\alpha(\alpha+1)(\alpha+2)}{6}} c^{\frac{(\alpha+1)(\alpha+3)(2\alpha+1)}{24}} & \text{for } \alpha \text{ odd.} \end{cases}$$

With $c \approx CT^2/N$, $C \approx M$, and $T^3 \ll N^2/M$, it follows that:

$$T \ll \begin{cases} N^{\frac{6\alpha+11}{14\alpha+27}} & \text{for } \alpha \text{ even,} \\ N^{\frac{6\alpha^2+9\alpha-3}{14\alpha^2+25\alpha-3}} & \text{for } \alpha \text{ odd.} \end{cases}$$

This gives the following exponents β for $T \ll N^\beta$:

α	1	2	3	4	∞
β	1/3	5/13	13/33	27/67	3/7
	0.333	0.385	0.394	0.403	0.429

When α grows to infinity, the best value we get is $T \approx N^{3/7+o(1)}$, giving a complexity of $N^{4/7+o(1)}$ for the worst-case search.

t_0	$N2^{-1/2+t_0/N} \bmod 1$
133850042647202529	$0.1\,1^{56}010111...$
406507340324366648	$0.1\,0^{57}101101...$
629850985640957850	$0.0\,0^{57}100001...$
182980980364065044	$0.0\,1^{58}000101...$

Fig. 3. Some worst cases for 2^x in double-extended precision ($N = 2^{64}$).

6 Experimental Results

We have implemented algorithm SLZ for $d = \alpha = 2$ using the GNU MP library [10] for the 2^x function.[4] Fig. 2 shows, for double, double-extended, and quadruple precisions, the best experimental parameters T and M for our method, together with the estimated time to check a whole exponent range of $N/2$ floating-point numbers.

Using that implementation, we started a search for worst cases of 2^x in double-extended precision. After one month of calendar time, using 10 processors (1.4GHz Athlons from the *Centre Charles Hermite*), we have completed about 10 percent of an exponent range of 2^{63} values and found 21 inputs with at least 56 identical bits after the round bit (Fig. 3). These experiments show that, with a carefully tuned implementation, and several computers running a few months, solving the Table Maker's Dilemma for double-extended precision is nowadays feasible for simple elementary functions.

References

[1] D. Boneh and G. Durfee, "Cryptanalysis of RSA with Private Key d Less than $n^{0.292}$," *Proc. Eurocrypt '99,* pp. 1-11, 1999.

[2] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $n = p^r q$ for Large r," *Proc. Eurocrypt '99,* pp. 326-337, 1999.

[3] D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation: Factoring with High Bits Known," *Proc. Eurocrypt '96,* pp. 178-189, 1996.

[4] D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation," *Proc. Eurocrypt '96,* pp. 155-165, 1996.

[5] D. Coppersmith, "Finding Small Solutions to Small Degree Polynomials," *Proc. Cryptography and Lattices Conf. (CALC '01),* pp. 20-31, 2001.

[6] D. Defour, F. de Dinechin, and J.-M. Muller, "Correctly Rounded Exponential Function in Double Precision Arithmetic," *Proc. SPIE 46th Ann. Meeting, Int'l Symp. Optical Science and Technology,* 2001.

[7] 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,* vol. 37, nos. 1-4, pp. 367-375, 2004, http://www.kluweronline.com/issn/1017-1398.

[8] N. Elkies, "Rational Points Near Curves and Small Nonzero $|x^3 - y^2|$ via Lattice Reduction," *Proc. Algorithmic Number Theory Symp. (ANTS-IV),* pp. 33-63, 2000.

[9] G. Gonnet, "A Note on Finding Difficult Values to Evaluate Numerically," http://www.inf.ethz.ch/personal/gonnet/FPAccuracy/NastyValues.ps, 2002.

[10] T. Granlund, *GNU MP: The GNU Multiple Precision Arithmetic Library,* 4.1.2 ed., 2002, http://www.swox.se/gmp/#DOC.

[11] "IEEE Standard For Binary Floating-Point Arithmetic," Technical Report ANSI-IEEE Standard 754-1985, 1985.

[12] C.S. Iordache and D.W. Matula, "Infinitely Precise Rounding for Division, Square Root, and Square Root Reciprocal," *Proc. 14th IEEE Symp. Computer Arithmetic,* pp. 233-240, 1999.

4. The code is available on http://www.loria.fr/~zimmerma/free/wclr-1.6.1.tar.gz.

[13] T. Lang and J.-M. Muller, "Bounds on Runs of Zeros and Ones for Algebraic Functions," *Proc. 15th IEEE Symp. Computer Arithmetic (ARITH 15),* N. Burgess and L. Ciminiera, eds., pp. 13-20, 2001.

[14] V. Lefèvre, "Moyens Arithmétiques pour un Calcul Fiable," thèse de doctorat, École Normale Supérieure de Lyon, 2000.

[15] V. Lefèvre and J.-M. Muller, "Worst Cases for Correct Rounding of the Elementary Functions in Double Precision," *Proc. 15th IEEE Symp. Computer Arithmetic (ARITH 15),* N. Burgess and L. Ciminiera, eds., pp. 111-118, 2001.

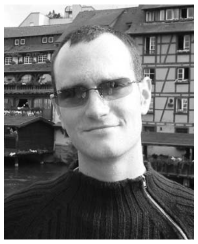
[16] A.K. Lenstra, H.W. Lenstra, and L. Lovász, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen,* vol. 261, pp. 515-534, 1982.

[17] L. Lovász, "An Algorithmic Theory of Numbers, Graphs and Convexity," *SIAM Lecture Series,* vol. 50, 1986.

[18] D. Stehlé, V. Lefèvre, and P. Zimmermann, "Worst Cases and Lattice Reduction," *Proc. 16th IEEE Symp. Computer Arithmetic,* pp. 142-147, 2003.

[19] D. Stehlé, "Breaking Littlewood's Cipher," *Cryptologia,* vol. 28, no. 4, pp. 341-357, 2004.

[20] A. Ziv, "Fast Evaluation of Elementary Mathematical Functions with Correctly Rounded Last Bit," *ACM Trans. Math. Software,* vol. 17, no. 3, pp. 410-423, 1991.



Damien Stehlé received the MSc degree in computer science from the École Normale Supérieure de Paris, France, in 2002. He is a PhD student working at LORIA, Nancy, France, on lattice reduction and correctly rounding mathematical functions. His main interests are computational number theory, computer arithmetic, cryptography, and complexity theory.

Vincent Lefèvre received the MSc and PhD degrees in computer science from the École Normale Supérieure de Lyon, France, in 1996 and 2000, respectively. Since 2000, he has been an INRIA researcher at LORIA, France. His research interests include computer arithmetic.

Paul Zimmermann received the PhD degree in computer science from the École Polytechnique, Palaiseau, France, in 1991. He has been an INRIA researcher at INRIA Rocquencourt and, since 1993, at INRIA Lorraine and LORIA, Nancy, France. His research interests include analysis of algorithms, computer algebra, and computer arithmetic.