INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Worst Cases and Lattice Reduction

Damien Stehlé — Vincent Lefèvre — Paul Zimmermann

N° 4586

15 octobre 2002

THÈME 2

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4586--FR+ENG

Worst Cases and Lattice Reduction

Damien Stehlé * , Vincent Lefèvre † , Paul Zimmermann ‡

Thème 2 — Génie logiciel
et calcul symbolique
Projet Spaces

Rapport de recherche n° 4586 — 15 octobre 2002 — 10 pages

Abstract: We propose a new algorithm to find worst cases for correct rounding of an analytic function. 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 [4, 5, 6].

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 $< N^{\frac{-d^2}{2d+1}}$ from a machine number in time $O(N^{\frac{d+1}{2d+1}+\varepsilon})$. For $d = 2$, this improves on the $O(N^{2/3+\varepsilon})$ complexity from Lefèvre's algorithm [12, 13] to $O(N^{3/5+\varepsilon})$. We exhibit some new worst cases found using our algorithm, for double-extended and quadruple precision. For larger d, our algorithm can be used to check that there exist no worst cases at distance $< N^{-k}$ in time $O(N^{\frac{1}{2}+O(\frac{1}{k})})$.

Key-words: Exact rounding, table maker's dilemma, worst case, IEEE-754, lattice reduction, Coppersmith's theorem

* ENS Paris, 45 rue d'Ulm, 75005 Paris (France), `damien.stehle@ens.fr`

† LORIA/INRIA Lorraine, Technopôle de Nancy-Brabois, 615 rue du Jardin Botanique, 54602 Villers-lès-Nancy Cedex (France), `lefevre@loria.fr`

‡ LORIA/INRIA Lorraine, Technopôle de Nancy-Brabois, 615 rue du Jardin Botanique, 54602 Villers-lès-Nancy Cedex (France), `zimmerma@loria.fr`

Blabla.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Pires cas et réduction des réseaux

Résumé : Nous proposons un nouvel algorithme trouvant les pires cas pour l'arrondi d'une fonction mathématique, en calcul à virgule flottante. Dans un premier temps, nous réduisons ce problème à la recherche d'une petite valeur d'un polynôme à coefficients réels. Nous montrons ensuite que ce second problème peut être résolu efficacement, en étendant le travail de Coppersmith — utilisant la réduction des réseaux — sur la recherche de petites valeurs d'un polynôme à coefficients entiers [4, 5, 6].

Pour des nombres flottants avec une mantisse bornée par N, et une approximation polynomiale de degré d, notre algorithme trouve tous les pires cas à distance $< N^{\frac{-d^2}{2d+1}}$ d'un nombre machine en temps $O(N^{\frac{d+1}{2d+1}+\varepsilon})$. Pour $d = 2$, cela améliore la complexité de l'algorithme de Lefèvre [12, 13] — $O(N^{2/3+\varepsilon})$ — avec une complexité de $O(N^{3/5+\varepsilon})$. Nous exhibons de nouveaux pires cas trouvés avec notre algorithme, pour la double précision étendue et la quadruple précision. Pour un plus grand degré d, notre algorithme peut montrer qu'il n'existe pas de pire cas à distance $< N^{-k}$ en temps $O(N^{\frac{1}{2}+O(\frac{1}{k})})$.

Mots-clés : arrondi exact, dilemme du fabricant de table, pire cas, norme IEEE-754, réduction des réseaux, théorème de Coppersmith

1. Introduction

The IEEE-754 standard for binary floating-point arithmetic [9], approved in 1985 by the IEEE Standards Board and the American National Standards Institute, 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, in particular Iordache and Matula [10], Lang and Muller [11], have shown that for the class of algebraic functions, such worst cases cannot be too near from a machine number or the middle of two consecutive machine numbers. Such bounds enable one to design some efficient algorithms that guarantee correct rounding for algebraic functions.

However, for non-algebraic functions, number theory bounds are not sharp enough, which makes correct rounding harder to implement. This is probably the reason why the IEEE-754 standard does not require correct rounding for those functions. Muller and other authors proposed in [16] to introduce different levels of quality for transcendental functions. This proposal was presented by Markstein at the May 2002 meeting of the IEEE-754 revision group, but the conclusion was that "*we're not yet ready to standardize*".

Systematic work on the TMD was done by Lefèvre and Muller [13], who published worst cases for many elementary functions in double precision ($N = 2^{53}$), over the full range for some functions. Alas, their approach is too expensive to deal with the quadruple precision, which is included in the current revision of the IEEE 754 standard. Thus currently 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 [17].

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

Exhaustive search methods consist in finding the hardest to round cases of the given function in the given range. 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 the first algorithm proposed by Lefèvre [12], and is based on Coppersmith's ideas.[1]

Previous related work was done by Elkies, who gives in [7] a new algorithm using lattice reduction to find all rational points of small height near a plane curve; for example his record:

$$5853886516781223^3 - 447884928428402042307918^2 = 1641843$$

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

$$2220422932^3 - 283059965^3 - 2218888517^3 = 30$$

corresponds to a worst case of the function $(x^3 + y^3)^{1/3}$ in 32-bit arithmetic. More recently Gonnet [8] also used lattice reduction to find worst cases, however his approach seems equivalent to Lefèvre's algorithm.

[1]To our best knowledge, this is the first non-cryptographic application of Coppersmith's work.

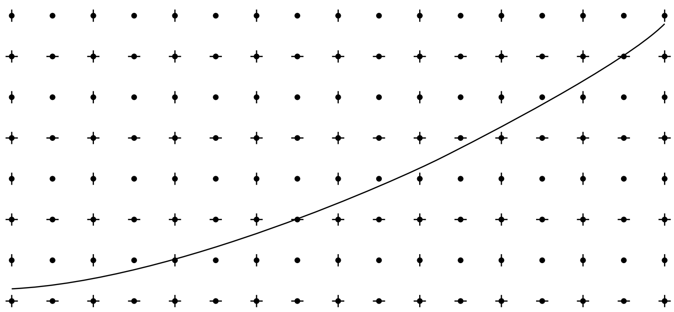

FIGURE 1. A function graph and the grid of machine numbers. Worst cases correspond to grid points with a small vertical distance to the curve.

Our 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 describes our new algorithm, after a short survey on lattice reduction and Coppersmith's work, which we heavily use. Section 4 presents some new worst cases found with our algorithm for the 2^x function, in double-extended precision and quadruple precision. Section 5 discusses some ideas for possible improvements and open questions.

2. PRELIMINARIES

2.1. **Definitions and Notations.** We assume we work here with 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 those m bits equal (resp. differ from) the round bit, x is a worst case for directed rounding (resp. rounding to nearest).

For sake of simplicity, we consider here directed rounding only (towards $-\infty$, towards $+\infty$, towards zero), since a worst case for rounding to nearest at precision n corresponds to a worst case 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. Then 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 consider 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. Given a polynomial approximation $P(t)$ to $Nf(\frac{t}{N})$ (for example a Taylor expansion), the TMD can be reduced to the following problem.

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

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

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 few worst cases.[2]

[2]The notation $x \ll y$ is equivalent to $x = O(y)$.

2.2. **Lefèvre's Algorithm.** Lefèvre's algorithm [12, 13] works as follows. One considers a linear approximation 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 $f(x) = a_0 + a_1 x + a_2 x^2 + O(x^3)$ around $x = 0$. Since we neglect the terms of order two or more in $Nf(\frac{t}{N})$, we must have $|a_2 \frac{T^2}{N}| \ll \frac{1}{M}$ 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 feasible for the double precision ($N^{2/3} \approx 4 \cdot 10^{10}$), near from the limits of the 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. A New Algorithm Using Lattice Reduction

In that section, we first state some basic facts about lattices — we refer to [15] for an introduction to that subject — and we explain Coppersmith's theorem, on which our algorithm is based. Then we introduce the algorithm, we prove its correctness and we analyze its complexity.

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 integral combinations of $\ell \leq n$ linearly independent vectors over $\mathbb{R}$, that is:

$$L = \{\sum_{i=1}^{\ell} n_i \mathbf{b}_i | n_i \in \mathbb{Z}\}.$$

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 $[\mathbf{b}_1^*, ..., \mathbf{b}_\ell^*]$ is the Gram-Schmidt orthogonalization of $[\mathbf{b}_1, ..., \mathbf{b}_\ell]$. The *basis* $[\mathbf{b}_1, ..., \mathbf{b}_\ell]$ of L is not unique and on an algorithmic point of view, only bases which consist of small 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 [14].

Theorem 1. *Given a basis $[\boldsymbol{b}_1, ..., \boldsymbol{b}_\ell]$ of a lattice $L \subset \mathbb{Z}^n$, the LLL algorithm provides in polynomial time in ℓ and in the bit-lengths of the $||\boldsymbol{b}_i||$'s, a basis $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_\ell\}$ satisfying:*

1. $||\boldsymbol{v}_1|| \leq 2^{\frac{\ell}{2}} \det(L)^{\frac{1}{\ell}}$;
2. $||\boldsymbol{v}_2|| \leq 2^{\frac{\ell}{2}} \det(L)^{\frac{1}{\ell-1}}$.

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

Coppersmith (see [4, 5], or [6] for a better description) found recently an important consequence of this theorem: one can compute the small roots of a multivariate polynomial modulo an integer N in polynomial time. His method proved very powerful to forge cryptographic schemes (see [2, 3, 4] for example). Our new algorithm intensely uses that technique.

3.2. **The Integer Small Value Problem.** The problem that will prove interesting 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 N. Equivalently, we are looking for the small integer roots of the bivariate polynomial:

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

We now explain how Coppersmith's technique helps solving it. First let α be a positive integer (that will grow later to infinity), and assume (x_0, y_0) is a root of Q modulo N. We consider the family of polynomials $Q_{i,j}(x,y) = x^i Q^j(x,y) N^{\alpha - j}$ with $0 \leq i + dj \leq d\alpha$. Then (x_0, y_0) is a root modulo N^α of each $Q_{i,j}$, whence of each 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 N^α — for small x and y, more precisely $|x| \leq X$ and $|y| \leq Y$ for fixed bounds X and Y. Thus, if (x_0, y_0) is a small root of v_1 and v_2 modulo N, (x_0, y_0) is also a root of v_1 and v_2 over $\mathbb{Z}$, and (x_0, y_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 $\frac{(\alpha+1)(d\alpha+2)}{2}$ generated by the vectors associated with the $Q_{i,j}(Xx, Yy)$: the vector associated with a bivariate polynomial $\sum_{i,j} a_{i,j} x^i y^j$ has its $x^i y^j$ coordinate equal to $a_{i,j} X^i Y^j$. We give here the shape of the matrix we get in the case $d = 3$ and $\alpha = 2$.

$$
\begin{array}{c}
i/j \\ 0/0 \\ 1/0 \\ 2/0 \\ 3/0 \\ 4/0 \\ 5/0 \\ 6/0 \\ 0/1 \\ 1/1 \\ 2/1 \\ 3/1 \\ 0/2
\end{array}
\begin{pmatrix}
N^2 & & & & & & & & & & & \\
 & N^2X & & & & & & & & & & \\
 & & N^2X^2 & & & & & & & & & \\
 & & & N^2X^3 & & & & & & & & \\
 & & & & N^2X^4 & & & & & & & \\
 & & & & & N^2X^5 & & & & & & \\
 & & & & & & N^2X^6 & & & & & \\
- & - & - & - & & & & NY & & & & \\
 & - & - & - & - & & & & NXY & & & \\
 & & - & - & - & - & & & & NX^2Y & & \\
 & & & - & - & - & - & & & & NX^3Y & \\
- & - & - & - & - & - & - & - & - & - & - & Y^2
\end{pmatrix}
$$

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

$$\det(L) = N^{\frac{d}{3}\alpha^3 + o(\alpha^3)} \cdot X^{\frac{d^2}{6}\alpha^3 + o(\alpha^3)} \cdot Y^{\frac{d}{6}\alpha^3 + o(\alpha^3)}.$$

Therefore, by Theorem 1, where here the lattice dimension satisfies $\ell \sim \frac{d}{2}\alpha^2$, the LLL algorithm gives us two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ of norm less than $N^{\frac{2}{3}\alpha + o(\alpha)} \cdot X^{\frac{d}{3}\alpha + o(\alpha)} \cdot Y^{\frac{1}{3}\alpha + o(\alpha)}$ when α grows to infinity. Those vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ correspond to two polynomials $v_1(x,y)$ and $v_2(x,y)$. Moreover if $|x| \leq X$ and $|y| \leq Y$, $|v_k(x,y)| \leq \sum_{i,j} |v_{i,j}^{(k)}| \frac{|x|^i |y|^j}{X^i Y^j} \leq C \cdot \max |v_{i,j}^{(k)}| \leq C \cdot ||\mathbf{v}_k||$ for a certain constant C. Thus, to get $|v_k(x,y)| < N^\alpha$, it is sufficient that:

$$C \cdot N^{\frac{2}{3}\alpha + o(\alpha)} \cdot X^{\frac{d}{3}\alpha + o(\alpha)} \cdot Y^{\frac{1}{3}\alpha + o(\alpha)} < N^\alpha,$$

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

Using Coppersmith's technique, one can thus solve the integer SValP in polynomial time as long as $X^d Y < N^{1-\epsilon}$. 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 heuristic has been made very often in cryptography (see [2, 3, 4]).

3.3. **The SLZ Algorithm.** Substituting N by 1, X by T, and Y by $1/M$ in the integer SValP, we find exactly the real SValP. The only difficulty is that $P(x)$ 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 TMD, but the sub-algorithm consisting of steps 3 to 11 may be of interest to solve the real SValP itself).

Input: a function f, positive integers N, T, M, d, α
Output: all worst cases at distance $< 1/M$ for $f(\frac{t}{N})$ for $|t| \leq T$
1. Let $P(t)$ the Taylor expansion of $Nf(\frac{t}{N})$ up to order d, and $n = \frac{(\alpha+1)(d\alpha+2)}{2}$
2. Compute a bound ε 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] $P'(x) = \frac{1}{C}\lfloor CP(Tx) \rceil$

4. Let $\{e_1, \ldots, e_n\} \leftarrow \{x^i y^j\}$ for $0 \le i + dj \le d\alpha$
5. Let $\{g_1, \ldots, g_n\} \leftarrow \{C^\alpha (Tx)^i (P'(x) + \frac{y}{M'})^j\}$ for $0 \le i + dj \le d\alpha$
6. Form the $n \times n$ matrix L where $L_{k,l}$ is the coefficient of the monomial e_k in g_l

7. $V \leftarrow C^{-\alpha}$LatticeReduce(L)
8. Let v_1, v_2 the two smallest vectors from V, and $p_1(x, y)$ and $p_2(x, y)$ the corresponding polynomials

9. **if** $\exists x, y, \in [-1, 1]$ with $|p_1(x, y)| \ge 1$ or $|p_2(x, y)| \ge 1$, **then** return(FAIL)
10. $p(t) \leftarrow \mathrm{Res}_y(p_1(t/T, y), p_2(t/T, y))$; **if** $p(t) = 0$ **then** return(FAIL)
11. **for** each t_0 in IntegerRoots$(p(t), [-T, T])$ **do**
12. **if** $|Nf(\frac{t_0}{N}) \bmod 1| < 1/M$ **then** output t_0.

(Note that the matrix L has integer entries since $CP'(x)$ has integer coefficients, and M' divides C.)

3.4. Correctness of the Algorithm.

Theorem 2. *In case algorithm SLZ does not return FAIL, it behaves correctly, i.e. it prints 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 12, we only have to check that no worst case is missed. Suppose there is $t_0 \in [-T, T]$ with $|Nf(\frac{t_0}{N}) \bmod 1| < 1/M$. Then $|P(t_0) \bmod 1| < 1/M + \varepsilon \le \frac{1}{2M'}$, and $|P'(x) - P(Tx)| \le \frac{d+1}{2C}$ for $|x| \le 1$, thus $|P'(t_0/T) \bmod 1| < \frac{1}{2M'} + \frac{d+1}{2C} \le 1/M'$. Whence $P'(t/T) + u = 0 \bmod 1$ has a root (t_0, u_0) with $|u_0| < 1/M'$. Since $p_1(t, y)$ and $p_2(t, y)$ are linear combinations of $P'(t/T) + \frac{y}{M'}$ and its powers, then $(t_0, M'u_0)$ is a common root of $p_1(t, y)$ and $p_2(t, y)$ modulo 1, and even over the reals since $|p_1|, |p_2| < 1$. Thus t_0 is an integer root of $\mathrm{Res}_y(p_1, p_2)$, and will be found at line 11. □

3.5. Choice of Parameters and Complexity Analysis.

3.5.1. *Coppersmith's Bound.* Because of the use of the Coppersmith's technique in our algorithm, to insure the algorithm does not return FAIL at step 9, the bound "$X^d Y \ll N$" has to be verified. In our case, X corresponds to T, Y to $1/M'$ and N to 1, so we get:

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

3.5.2. *Choice of the Degree d With Respect to T.* Let $(a_i)_i$ 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. $MT^{d+1} \ll N^d$.

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

[3]The notation $\lfloor CP(Tx) \rceil$ means that we round to the nearest integer each coefficient of $CP(Tx)$.

	N	M	T	d	α	est. time
double	2^{53}	2^{28}	2^{15}	1	1	560 days
	2^{53}	2^{53}	2^{20}	2	2	120 days
precision	2^{53}	2^{106}	2^{25}	4	2	45 days
double	2^{64}	2^{32}	2^{19}	1	1	140 years
extended	2^{64}	2^{64}	2^{24}	2	2	43 years
precision	2^{64}	2^{128}	2^{30}	4	2	9 years
quadruple	2^{113}	2^{70}	2^{35}	1	1	1600 Gyears
	2^{113}	2^{113}	2^{43}	2	2	94 Gyears
precision	2^{113}	2^{226}	2^{53}	4	2	1.6 Gyears

FIGURE 2. Best experimental parameters for double, double-extended and quadruple precision, and estimated time for an exponent range of $N/2$ values.

3.5.4. *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 $P'(x)$. With d-bit floating-point coefficients, a necessary condition is that $2^d > CN$ to ensure the constant coefficient from $P'(x)$ is correct.

REMARK 3: When searching worst cases with $M \ll N$, degree 2 is enough. 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 $N^{2/5}$, giving a complexity of $N^{3/5}$ to search a whole range of $N/2$ values. More generally, for $M \ll N^k$, degree $2k$ is enough, giving a complexity of $N^{\frac{2k}{4k+1}}$.

4. Experimental Results

We have implemented algorithm SLZ in the Pari/GP system (version 2.2.4-alpha) [1] and experimented it on a Athlon XP 1600+ under Linux. We have chosen the 2^x function since it is the easiest one, with only one exponent range to study. Fig. 2 shows for each target precision (double, double-extended, quadruple), and for $M \approx N$ and $M \approx N^2$, the best parameters (T, d, and α) for our method, together with the estimated time to check the whole exponent range, i.e. $N/2$ floating-point numbers. For each precision, the first row gives the best parameters for the $d = \alpha = 1$ case, which is what Gonnet considers in [8]; comparing that first row to the following ones shows the speedup obtained. For $M \approx N$, the speedup increases from 3 to 17, whence is not dramatic. However for $M \approx N^2$, we get a speedup of about 1000 in quadruple precision with respect to the naive method ($d = \alpha = 1$), with $(d, \alpha) = (4, 2)$. Fig. 3 shows a few worst cases found using algorithm SLZ for double-extended and quadruple precision. These experiments tend to show that with a carefully tuned implementation, and several computers running a few months, solving the TMD for the double-extended precision is nowadays feasible.

5. Possible Improvements and Open questions

We have presented a new algorithm, based on lattice reduction, to search for worst cases for correct rounding of analytic functions. The first experimental results show that algorithm SLZ is quite efficient, especially to detect worst cases at distance much less than 2^{-n}, where n is the target precision. However the efficiency largely depends on the function considered, like in Lefèvre's algorithm.

Several open questions remain. Does this approach extend like in the modular case ([4]) to functions of two variables like x^y or $\arctan \frac{x}{y}$?

Our algorithm is complementary to that of Elkies [7], which works well when $M \ll N$ (in our notation), i.e. when we expect many worst cases, whereas our algorithm is more efficient when

N	t_0	$N2^{-1/2+t_0/N} \bmod 1$
2^{64}	586071771766963	$0.1\,1^{47}001111\ldots$
2^{64}	594068190588573	$0.0\,0^{48}100010\ldots$
2^{64}	891586182147388	$0.1\,1^{50}001000\ldots$
2^{64}	9014384889202147	$0.0\,1^{53}010011\ldots$
2^{64}	9602866023852631	$0.0\,0^{54}111001\ldots$
2^{113}	1119374922072865495	$0.0\,1^{63}000000\ldots$
2^{113}	8923960372306650064	$0.0\,0^{64}101011\ldots$
2^{113}	43616445401128570224	$0.0\,1^{65}011110\ldots$
2^{113}	53608038600996804036	$0.0\,1^{67}000001\ldots$

FIGURE 3. Some worst cases found for the 2^x function in double-extended and quadruple precision.

$M \gg N$, i.e. when we expect only few worst cases, or none. However, in the case of $f(x) = x^{3/2}$, related to Hall's conjecture, Elkies proposes a special-purpose algorithm to find all worst cases at distance $< 1/N$ in $O(N^{1/2+\varepsilon})$. Does this algorithm generalize to other algebraic functions?

What is the best complexity one may obtain for the TMD using Coppersmith's method? Coppersmith gives in [6] some arguments giving evidence that $T \ll M^{1/d}$ might be the best possible bound for finding in polynomial time the roots of general modular univariate polynomials of degree d. Could one improve the technique by considering the shape of our polynomials, like is done by Boneh and Durfee [2] for the small inverse problem with $|k| < N^{0.292}$?

References

[1] Batut, C., Belabas, K., Bernardi, D., Cohen, H., and Olivier, M. *User's Guide to PARI/GP*, 2000. `ftp://megrez.math.u-bordeaux.fr/pub/pari/manuals/users.pdf`.

[2] Boneh, D., and Durfee, G. Cryptanalysis of RSA with private key d less than $N^{0.292}$. In *Proceedings of Eurocrypt'99* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–11.

[3] Boneh, D., Durfee, G., and Howgrave-Graham, N. Factoring $N = p^r q$ for large r. In *Proceedings of Eurocrypt'99* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 326–337.

[4] Coppersmith, D. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Proceedings of Eurocrypt'96* (1996), U. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 178–189.

[5] Coppersmith, D. Finding a small root of a univariate modular equation. In *Proceedings of Eurocrypt'96* (1996), U. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 155–165.

[6] Coppersmith, D. Finding small solutions to small degree polynomials. In *Proceedings of CALC'01* (2001), vol. 2146 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 20–31.

[7] Elkies, N. Rational points near curves and small nonzero $|x^3 - y^2|$ via lattice reduction. In *Proceedings of ANTS-IV* (2000), W. Bosma, Ed., vol. 1838 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 33–63.

[8] Gonnet, G. A note on finding difficult values to evaluate numerically. `http://www.inf.ethz.ch/personal/gonnet/FPAccuracy/NastyValues.ps`, Sept. 2002. 3 pages.

[9] IEEE standard for binary floating-point arithmetic. Tech. Rep. ANSI-IEEE Standard 754-1985, New York, 1985. Approved March 21, 1985: IEEE Standards Board, approved July 26, 1985: American National Standards Institute, 18 pages.

[10] Iordache, C. S., and Matula, D. W. Infinitely precise rounding for division, square root, and square root reciprocal. In *Proceedings of 14th IEEE Symposium on Computer Arithmetic* (1999), pp. 233–240.

[11] Lang, T., and Muller, J.-M. Bounds on runs of zeros and ones for algebraic functions. In *Proceedings of ARITH'15* (Vail, Colorado, 2001), N. Burgess and L. Ciminiera, Eds., IEEE Computer Society, pp. 13–20.

[12] Lefèvre, V. *Moyens arithmétiques pour un calcul fiable*. PhD Thesis, École Normale Supérieure de Lyon, Jan. 2000.

[13] Lefèvre, V., and Muller, J.-M. Worst cases for correct rounding of the elementary functions in double precision. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH'15)* (2001), N. Burgess and L. Ciminiera, Eds., IEEE Computer Society, pp. 111–118.

[14] Lenstra, A. K., Lenstra, H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen 261* (1982), 515–534.
[15] Lovàsz, L. An algorithmic theory of numbers, graphs and convexity. *SIAM lecture series 50* (1986).
[16] Muller, J.-M. Proposals for a specification of the elementary functions. In *Abstracts of SCAN'2002* (2002), J.-L. Lamotte and F. Rico, Eds., Laboratory LIP6, Paris, France, pp. 54–55.
[17] Ziv, A. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Softw. 17*, 3 (1991), 410–423.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399