Gal's Accurate Tables Method Revisited

Damien Stehlé, Paul Zimmermann

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Gal's Accurate Tables Method Revisited

Damien Stehlé, Paul Zimmermann

N° 5359

Octobre 2004

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--5359--FR+ENG

INRIA
LORRAINE

Gal's Accurate Tables Method Revisited

Damien Stehlé, Paul Zimmermann

Thème 2 — Génie logiciel
et calcul symbolique
Projet Spaces

Rapport de recherche n° 5359 — Octobre 2004 — 23 pages

Abstract: Gal's accurate tables algorithm aims at providing an efficient implementation of elementary functions with correct rounding as often as possible. This method requires an expensive pre-computation of a table made of the values taken by the function – or by several related functions – at some distinguished points. Our improvements of Gal's method are two-fold: on the one hand we describe what is the arguably best set of distinguished values and how it improves the efficiency and correctness of the implementation of the function, and on the other hand we give an algorithm which drastically decreases the cost of the pre-computation. These improvements are related to the worst cases for the correct rounding of mathematical functions and to the algorithms for finding them. We show that the whole method can be turned into practice by giving complete tables for 2^x and $\sin x$ for $x \in [\frac{1}{2}, 1[$, in double precision.

Key-words: Gal's method, elementary functions, lattice reduction, simultaneous worst cases.

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

Améliorations de la méthode de Gal

Résumé : L'algorithme de Gal a pour but de fournir une implantation efficace des fonctions mathématiques élémentaires avec arrondi correct aussi souvent que possible. Cette méthode nécessite un pré-calcul très coûteux d'une table de valeurs prises par la fonction considérée – ou de plusieurs fonctions en relation avec celle-ci – en un ensemble de points distingués. Nous améliorons la méthode de Gal de deux manières: d'une part nous décrivons quel est le meilleur ensemble de points distingués et comment il améliore l'efficacité et la correction de l'implantation de la fonction, et d'autre part nous donnons un algorithme qui décroit considérablement le coût du pré-calcul. Ces améliorations sont liées aux pires cas pour l'arrondi correct des fonctions mathématiques et aux algorithmes qui les calcule. Nous montrons que nos améliorations sont raisonnables en pratique en donnant des tables complètes pour 2^x et $\sin x$ avec $x \in [\frac{1}{2}, 1[$, en double précision.

Mots-clés : Méthode de Gal, fonctions mathématiques élémentaires, réduction de réseaux, pires cas simultanés.

1 Introduction

The IEEE-754 standard for floating-point arithmetic [12] specifies that the four basic arithmetic operations and the square root should be correctly rounded, but does not require a correct rounding for any other elementary mathematical function, like trigonometric, exponential and logarithmic functions. For this reason, there are libraries which produce incorrectly rounded results, or which do not guarantee a correctly rounded result, and, for example, this causes serious difficulties for the portability and reproducibility of scientific calculations. The reticence to extend the standard to the elementary mathematical functions comes from the fact that an implementation guaranteeing a correct rounding is too expensive. Citing David Hough in the context of the revision of the IEEE-754 standard[1]:

> *[Elementary functions] are too hard to standardize now because nobody knows how to trade-off precision vs performance. If less than correct rounding is specified [...] then [it] is always possible that somebody will come up with a non-standard algorithm that is more accurate AND faster. This can't happen with a correct rounding specification, but correctly-rounded transcendental functions seem to be inherently much more expensive than almost-correctly-rounded. One could instead standardize properties that approximate functions are supposed to obey - but anomalies still abound. All these points argue against standardizing transcendental functions now.*

In the present paper, we improve a routine commonly used in the implementation of elementary functions, namely Gal's accurate tables method (see [8, 9] and [17] pp58–62), in the hope it will shrink the efficiency gap between elementary functions libraries in use and those guaranteeing a correct rounding (such as [1, 25]), and give more support for proposals of standardization of these functions [18, 7].

The implementation of an elementary function over its full domain for some given precision usually requires two phases (see [6] for the exponential function): a *quick phase* giving a correctly rounded result for an overwhelming proportion of the entries and an *accurate phase* which is considerably slower but performed only in the few cases for which the quick phase was not sufficient. The quick phase often uses the input-output precision as working precision (or eventually extends the precision or uses a few more bits in very few steps), while the accurate phase is often based on Ziv's strategy [24] which extends the working precision until the result can be guaranteed correctly rounded (eventually, one may also use a sharp bound on the

[1] `http://grouper.ieee.org/groups/754/meeting-minutes/01-03-14.html`

required precision if such a bound is known [13, 14]). The quick phase is itself often subdivided into three sub-phases:

- *First range reduction*: The full domain of the function is restricted to a smaller one by using the mathematical properties of the considered function, *e.g.* $\exp(x + k\ln 2) = 2^k \cdot \exp(x)$ giving the range $[0, \ln 2[$, $2^{x+k} = 2^k \cdot 2^x$ giving the range $[0, 1[$, $\sin(x + k\frac{\pi}{2}) = f_k(x)$ with $f_k = \pm\sin x$ or $f_k = \pm\cos x$ depending on $k \bmod 4$, giving the range $[-\frac{\pi}{4}, \frac{\pi}{4}[$, ...
- *Second range reduction*: By a table lookup, the range is shrunk further. For example, we write $2^x = 2^{x_0} \cdot 2^h$, (resp. $\sin x = \sin x_0 \cdot \cos h + \cos x_0 \cdot \sin h$) where $(x_0, 2^{x_0})$ (resp. $(x_0, \sin x_0, \cos x_0)$) is stored in the table and $h = x - x_0$ is small ($|h|$ is approximately smaller than the length of the range obtained after the first reduction, divided by the number of distinguished elements). We call *related functions* the functions used in this range reduction: for 2^x there is one related function (namely 2^x), but for $\sin x$ there are two related functions (namely $\sin x$ and $\cos x$).
- *Polynomial evaluation*: The remaining terms (*e.g.* $2^h, \cos h, \sin h, \ldots$) are evaluated by using a polynomial approximation of the function (or the related functions) over the restricted range.

There are very satisfactory answers for the first [20] and last sub-phases [2]. Gal's method addresses the second sub-phase. The original technique is based on a table of "almost regularly spaced" distinguished points which images by the function (or the related functions) are unusually close to machine numbers. The table, of size a few kilobytes [5], is obtained *via* a pre-computation based on a naive search.

We improve Gal's method in two ways. First we notice that the best set of distinguished points is made of the values for which the function (resp. the related functions) is hard to round (resp. to round simultaneously) in the case of directed rounding (towards 0 or ∞): this problem is therefore closely related to the Table Maker's Dilemma (TMD for short). A direct consequence of this fact is that we can adapt the methods which find the worst cases of a one-variable function [13, 14, 22, 23] in order to construct the tables, if there is a single related function. This is much more efficient than Gal's naive search. In the case of one related function, Gal's method can be adapted for the accurate phase to guarantee a correct rounding for any input. The second improvement is an algorithm to construct the table when there are two related functions: we modify the lattice-based algorithm of [22, 23] to find simultaneously bad cases for two functions.

In the paper, although the method can be easily generalized to other functions and to arbitrary precision, we focus on 2^x and $\sin x$ for $x \in [\frac{1}{2}, 1[$ in double precision.

For this choice of parameters, our improvements towards Gal's original method are the following:

1. The table for $\sin x$ is constructed much more efficiently than by naive search (the cost of the naive search would have been 2^{52} calls to $\sin x$ and $\cos x$ in extended precision).
2. For $\sin x$, the proportion of entries for which the quick phase fails decreases from $\approx 2^{-10}$ to $\approx 2^{-20}$.
3. The accurate phase for 2^x can be based on Gal's method by using the work of Lefèvre [13, 14]. It requires only quadruple precision calculations.

The rest of the paper is organized as follows. In Section 2 we recall some basic facts about the TMD and Gal's accurate tables method. In Section 3 we describe what is the best set of distinguished values in the table and how this choice improves the quick and accurate phases. In Section 4 we explain how to obtain these tables. Finally, in Section 5 we show that the method is practical for double precision by giving parts of the tables for 2^x and $\sin x$.

Notations: We define $[|a, b|]$ (resp. $[|a, b[|$) as the set of integers in $[a, b]$ (resp. $[a, b[$). For any integer n, $[a, b[_n$ denotes the set of the $\frac{m}{2^n}$'s where $a \leq \frac{m}{2^n} < b$ and m is an integer. For example, $[\frac{1}{2}, 1[_{53}$ corresponds to the doubles with exponent -1. For any reals x and c, x cmod c denotes $x - \lfloor \frac{x}{c} \rceil$, and if $\frac{x}{c}$ is half an odd integer, we choose any of the possibilities. In particular, x cmod 1 is the "centered" fractional part of x. We denote by $o(x)$ a machine number closest to x in double precision. Finally, vectors are denoted in bold and for a vector $\mathbf{x} \in \mathbb{R}^n$, $||\mathbf{x}||$ and $||\mathbf{x}||_1$ are respectively its L_2 and L_1 norms, i.e. $||(x_1, \ldots, x_n)|| = \sqrt{\sum_{i=1}^n x_i^2}$ and $||(x_1, \ldots, x_n)||_1 = \sum_{i=1}^n |x_i|$.

2 Preliminaries

This section gives the necessary background to describe our improvements of Gal's accurate tables method. We describe an informal model where the functions we consider are regarded as black-boxes returning uniformly distributed outputs. Experiments corroborate very well this model. We use it to describe Gal's original scheme and to give the basic ideas concerning the Table Maker's Dilemma.

2.1 The Random Model

We consider that a function behaves as a random black-box when the first bits of its output are thrown away. Such an assumption is of course very strong and completely

heuristic. Nevertheless experiments tend to validate this hypothesis and in particular the experiments of Section 5 do not contradict it.

A given function $f : [\frac{1}{2}, 1[\to [a, b[$ for some $a < b$ (*e.g.* $2^x : [\frac{1}{2}, 1[\to [1, 2[$) is seen as a random black-box: for any n and $k_1 < k_2$ larger than some thresholds, if x is chosen randomly and uniformly in $[\frac{1}{2}, 1[_n$, then the bits at positions k_1 and k_2 of the binary expansion of $f(x)$ are independent random variables uniformly distributed in $\{0, 1\}$. This implies some useful properties:

- The probability of obtaining a run of p consecutive identical bits– starting from some given position – in the binary expansion of $f(x)$ is 2^{1-p}.
- If we consider 2^p consecutive x's, there is $O(1)$ of them such that the binary expansion of $f(x)$ has a run of p consecutive identical bits starting from some given position.
- The set of the $x \in [\frac{1}{2}, 1[_n$ such that $f(x)$ has a run of p consecutive identical bits – starting from some given position – is of cardinality around 2^{n-p}, and the maximum distance between two consecutive elements is less than $\approx \left(1 + n - \frac{p}{2}\right) \cdot 2^{n-p}$ (see the appendix for a proof).

We generalize this model to two functions $f_1, f_2 : [\frac{1}{2}, 1[\to [a, b[$ for some $a < b$ (*e.g.* $\sin x$ and $\cos x$). They are seen as random independent black-boxes: firstly they both are black-boxes, and secondly, for any n and k_1, k_2 larger than some thresholds, if x is chosen randomly and uniformly in $[\frac{1}{2}, 1[_n$, then the bit at position k_1 of the binary expansion of $f_1(x)$ and the bit at position k_2 of the binary expansion of $f_2(x)$ are independent random variables. From such an hypothesis we can also derive some properties:

- The probability of obtaining a run of p consecutive identical bits– starting from some given position – in the binary expansions of both $f_1(x)$ and $f_2(x)$ is 2^{2-2p}.
- If we consider 2^{2p} consecutive x's, there is $O(1)$ of them such that the binary expansions of both $f_1(x)$ and $f_2(x)$ have a run of p consecutive identical bits starting from some given position.
- The set of the $x \in [\frac{1}{2}, 1[_n$ such that $f_1(x)$ and $f_2(x)$ have a run of p consecutive identical bits starting from some given position is of cardinality around 2^{n-2p}, and the maximum distance between two consecutive elements is less than $\approx (1 + n - p) \cdot 2^{n-2p}$.

Notice that it is easy to make all these statements rigorous and to generalize the model to more than two functions.

2.2 Gal's Accurate Tables Method

Gal's accurate tables method [8, 9] addresses the second range reduction of the quick phase of the calculation of $f(x)$. The method is general, but here we take as examples $f(x) = 2^x$ and $f(x) = \sin x$. The idea of the second range reduction is to write:

$$2^x = 2^{x_0} \cdot 2^h,$$

$$\sin x = \sin x_0 \cdot \cos h + \cos x_0 \cdot \sin h,$$

where x_0 is in a table of distinguished points and is stored with an approximation of its corresponding 2^{x_0} (resp. $\sin x_0$ and $\cos x_0$); $h = x - x_0$ is small: roughly speaking, $|h|$ is smaller than $\frac{1}{2}$ divided by the number of distinguished points. After this table-based range reduction, 2^h (resp. $\cos h$ and $\sin h$) is computed approximately by using a low degree polynomial (resp. two low degree polynomials) which closely approximates 2^h (resp. $\sin h$ and $\cos h$) when h is close to 0. To fix the ideas, we can suppose that these polynomials are the Taylor expansions of the functions at 0, but it is possible to do better [2]. If we consider the double precision (i.e. 53 bits of mantissa), since we are in the quick phase, the calculations should be made in double precision as often as possible, and it is interesting to have tables with approximately 2^{10} distinguished elements, for cache optimization reasons (see [5, 1]).

The naive way of choosing the distinguished elements is to take them equally spaced, thus minimizing the maximum value taken by $|h|$. Gal's idea is to relax very slightly the maximum value taken by $|h|$ in order to choose better distinguished points: he takes almost regularly spaced distinguished points x_0 such that the stored approximation of 2^{x_0} (resp. $\sin x_0$ and $\cos x_0$) is unusually close to its true value. More precisely, in double precision, for any $0 \leq i < 2^{10}$, x_0^i is a machine number closest to $\frac{1}{2} + \frac{i}{2^{11}}$ such that:

$$\left|2^{52} \cdot 2^{x_0^i} \bmod 1\right| \leq 2^{-10} \quad \text{for } 2^x$$

$$\left|2^{53+e} \cdot \sin x_0^i \bmod 1\right| \leq 2^{-10} \quad \text{and} \quad \left|2^{53} \cdot \cos x_0^i \bmod 1\right| \leq 2^{-10} \quad \text{for } \sin x,$$

where $e = 1$ if $|x_0^i| \leq \frac{\pi}{6}$ and 0 otherwise. According to the random model, x_0^i should be extremely close to $\frac{1}{2} + \frac{i}{2^{11}}$, implying a negligible increase of the value that can be taken by $|h|$. Moreover, this set of distinguished points gives a more accurate implementation of the function: since $|x - x_0|$ is bounded by $\approx 2^{-12}$, the final output for $f(x)$ can be made accurate with ≈ 63 bits of precision although the calculations are made with doubles. Again in the random model, this implies that the value

output for $f(x)$ is correctly rounded with probability around $1 - 2^{-10}$, which is higher than 99.9%.

The tables are constructed by exhaustive search, so that there are approximately 2^9 trials for any distinguished element of the 2^x-table, and approximately 2^{18} trials for any distinguished element of the $\sin x$-table.

2.3 Some Reminders on the TMD

Let n be the precision and $f : [\frac{1}{2}, 1[\to [\frac{1}{2}, 1[$ be the considered function. A *m-bad case* x for f is a n-bit machine number such that at least $n + m$ bits are needed to round $f(x)$ correctly. More precisely these are the x's in $[\frac{1}{2}, 1[_n$ such that:

- $|2^n \cdot f(x) \bmod 1| \leq 2^{-m}$ for the directed rounding modes (towards 0, $+\infty$ or $-\infty$),
- $|2^n \cdot f(x) - \frac{1}{2} \bmod 1| \leq 2^{-m}$ for the rounding to nearest mode.

Notice that in this definition, m can be any real number. Here the functions we consider have output ranges different from $[\frac{1}{2}, 1[$, so we adopt the definition to our case. We say that $x \in [\frac{1}{2}, 1[_{53}$ is a m-bad case for 2^x if $|2^{52+x} \bmod 1| \leq 2^{-m}$, a m-bad case for $\cos x$ if $|2^{53} \cdot \cos x \bmod 1| \leq 2^{-m}$, a m-bad case for $\sin x$ if $|2^{53} \cdot \sin x \bmod 1| \leq 2^{-m}$ when $x \in]\frac{\pi}{6}, 1[$ and $|2^{54} \cdot \sin x \bmod 1| \leq 2^{-m}$ if $x \in [\frac{1}{2}, \frac{\pi}{6}[$.

The knowledge of the *hardness to round* f, i.e. the maximum m such that f admits a m-bad case, makes it possible to implement f efficiently, because the maximum number of bits which are needed is known in advance. The drawback of this approach is that this value is hard to compute. The exhaustive search is by far infeasible because of its $\approx 2^n$ complexity. Lefèvre and Muller proposed a $\approx 2^{2n/3}$ algorithm [13, 14] , which was sufficient to perform some systematic work in double precision. Stehlé, Lefèvre and Zimmermann [22] gave a generalization of this algorithm by using lattice-based Coppersmith's work to find small roots of modular polynomials [3, 4]. They obtained an algorithm with a heuristic $\approx 2^{3n/5}$ complexity, which was later improved to $\approx 2^{5n/7}$ [23]. Although these complexity bounds are better than the one of Lefèvre's algorithm, it seems that the practical cut-off between both methods is around the double extended precision, i.e. $n = 64$.

3 Gal's Accurate Tables Method Revisited

Our improvement is based on the idea that it is possible to require runs of more than 10 consecutive identical bits as Gal does. We first describe what are the best sets of distinguished points for 2^x and $\sin x$, and then show that for 2^x, Gal's method based

on this set can be used for the whole evaluation scheme (both quick and accurate phases).

3.1 A Table Made of Bad Cases for the Directed Rounding Modes

Under the random model assumption, it is possible to strengthen the requirements of Gal on the lengths of the runs of consecutive identical bits. For the 2^x function, we can take as set of distinguished points all the 42-bad cases. There are approximately 2^{12} of them and the maximum distance between two consecutive ones is below $2^{-9.914}$, see Section 5. Choosing such a set of distinguished values decreases the error made on 2^{x_0} in

$$2^x = 2^{x_0} \cdot 2^h,$$

but since the maximum distance between two distinguished values is not decreased, the error made while evaluating the polynomial approximating 2^h for h close to 0 is roughly the same, i.e. $\approx 2^{-10}$: since the coefficients of the polynomial are doubles and since this is also the working precision, the degree-1 coefficient is correct with error bounded by $\approx 2^{-53-10}$. As a consequence, the error made in the quick phase is roughly the same. But as we will see in the next subsection, this choice of distinguished points makes the accurate phase more efficient.

For the $\sin x$ function, since there are two related functions in the second range reduction (namely $\sin x$ and $\cos x$), we cannot expect runs of consecutive identical bits as long as for 2^x. Indeed, the best choice of distinguished values is made of all the x's that are simultaneously 21-bad cases for $\sin x$ and $\cos x$. There are approximately 2^{12} of them, and the average distance between two consecutive ones is below $2^{-9.977}$ as shown by the experiments of Section 5. This means that the errors made on $\sin x_0$ and $\cos x_0$ in

$$\sin x = \sin x_0 \cdot \cos h + \cos x_0 \cdot \sin h \qquad (1)$$

are decreased from $\approx 2^{-63}$ to $\approx 2^{-74}$. We argue that the errors made in the polynomial evaluations corresponding to $\cos h$ and $\sin h$ can also be made that small, in a very efficient way. We define

$$S(h) = h - a_3 h^3 + a_5 h^5 \quad \text{and} \quad C(h) = 1 - \frac{1}{2}h^2 + a_4 h^4,$$

where a_i is the double closest to $\frac{1}{i!}$ for $i \in \{3, 4, 5\}$. Using the fact that $|h| \leq 2^{-10.977}$ along with Lagrange's bound, we obtain that:

$$|\sin h - S(h)| \leq \left|\sin h - \left(h - \frac{h^3}{6} + \frac{h^5}{120}\right)\right| + \left|a_3 - \frac{1}{6}\right| |h|^3 + \left|a_5 - \frac{1}{120}\right| |h^5|$$

$$\begin{aligned}
&\leq 2^{-76.839} \cdot 2^{-12.299} + 2^{-56.584} \cdot 2^{-32.931} + 2^{-62.906} \cdot 2^{-54.885} \\
&\leq 2^{-88.314}, \\
|\cos h - C(h)| &\leq \left|\cos h - \left(1 - \frac{h^2}{2} + \frac{h^4}{24}\right)\right| + \left|a_4 - \frac{1}{24}\right| |h|^4 \\
&\leq 2^{-65.862} \cdot 2^{-9.491} + 2^{-58.584} \cdot 2^{-43.908} \\
&\leq 2^{-75.352}.
\end{aligned}$$

We now explain how to evaluate equation (1). Let c_0 and s_0 be the two machine numbers approximating $\cos x_0$ and $\sin x_0$. By construction we have $|c_0 - \cos x_0|, |s_0 - \sin x_0| \leq 2^{-74}$. Let $C^*(h) = C(h) - 1$ and $S^*(h) = S(h) - h$. Recall that $h \leq 2^{-10.977}$. We first compute approximations of $S^*(h)$ and $C^*(h)$ with Horner's method in double precision:

$k \leftarrow o(h^2)$	$\lvert k - h^2\rvert \leq 2^{-54-21} \leq 2^{-75}$	$k \leq 2^{-21.953}$
$k' \leftarrow o(h \cdot k)$	$\lvert k' - h^3\rvert \leq 2^{-10.977-75} + 2^{-54-32} \leq 2^{-84.988}$	$\lvert k'\rvert \leq 2^{-32.930}$
$S^* \leftarrow o(a_5 \cdot k)$	$\lvert S^* - a_5 \cdot h^2\rvert \leq 2^{-6.906-75} + 2^{-54-28} \leq 2^{-80.952}$	$\lvert S^*\rvert \leq 2^{-28.859}$
$S^* \leftarrow o(S^* - a_3)$	$\lvert S^* - (-a_3 + a_5 \cdot h^2)\rvert \leq 2^{-80.952} + 2^{-54-2} \leq 2^{-55.999}$	$\lvert S^*\rvert \leq 2^{-2.584}$
$S^* \leftarrow o(k' \cdot S^*)$	$\lvert S^* - S^*(h)\rvert \leq 2^{-84.988-2.584} + 2^{-55.999-32.930} + 2^{-54-35} \leq 2^{-86.754}$	$\lvert S^*\rvert \leq 2^{-35.513}$
$C^* \leftarrow o(a_4 \cdot k)$	$\lvert C^* - a_4 \cdot h^2\rvert \leq 2^{-4.584-75} + 2^{-54-26} \leq 2^{-78.777}$	$\lvert C^*\rvert \leq 2^{-26.536}$
$C^* \leftarrow o(C^* - \frac{1}{2})$	$\lvert C^* - (-\frac{1}{2} + a_4 \cdot h^2)\rvert \leq 2^{-78.777} + 2^{-54-2} \leq 2^{-55.999}$	$\lvert C^*\rvert \leq 2^{-1.000}$
$C^* \leftarrow o(k \cdot C^*)$	$\lvert C^* - C^*(h)\rvert \leq 2^{-75-1.000} + 2^{-55.999-21.953} + 2^{-54-22} \leq 2^{-74.824}$	$\lvert C^*\rvert \leq 2^{-22.951}$.

The calculation of $\sin x$ ends with the sum:

$$s = s_0 + c_0 \cdot h + (s_0 \cdot C^* + c_0 \cdot S^*),$$

the term "$c_0 \cdot h$" being evaluated in an extended precision, for example by using a quadruple, a double-double, or by simulating the extended precision with the use of a fma operation. For example we do this with a fma.

- $h' \leftarrow o_{fma}(s_0 + c_0 \cdot h)$.
- $t \leftarrow o(h' - s_0)$.
- $l \leftarrow o_{fma}(t - c_0 \cdot h)$.

Since $|c_0 \cdot h| \leq \frac{s_0}{2}$, we have $h' \geq \frac{s_0}{2}$ and the second operation is exact. Therefore we have:

$$h' + l = s_0 + c_0 \cdot h \text{ and } |(h' + l) - (\sin x_0 + \cos x_0 \cdot h)| \leq 2^{-74} + 2^{-74-10.977} \leq 2^{-73.999}.$$

We now compute some less significant bits, which in particular suffice to round correctly the result most of the time.

$S^* \leftarrow o(c_0 \cdot S^*)$	$\lvert S^* - \cos x_0 \cdot S^*(h)\rvert \leq 2^{-74-35.513} + 2^{-0.188-86.754} + 2^{-54-35} \leq 2^{-86.631}$	$\lvert S^*\rvert \leq 2^{-35.700}$
$C^* \leftarrow o(s_0 \cdot C^*)$	$\lvert C^* - \sin x_0 \cdot C^*(h)\rvert \leq 2^{-74-22.951} + 2^{-0.249-74.667} + 2^{-54-23} \leq 2^{-74.610}$	$\lvert C^*\rvert \leq 2^{-23.199}$

$$\begin{array}{cc} C^* \leftarrow o(S^* + C^*) & |C^* - (\cos x_0 \cdot S^*(h) + \sin x_0 \cdot C^*(h))| \leq 2^{-86.631} + 2^{-74.610} \leq 2^{-74.609} \\ C^* \leftarrow o(l + C^*) & |(h' + C^*) - \sin x| \leq 2^{-74.609} + 2^{-73.999} + 2^{-1-106} \leq 2^{-73.271}. \end{array}$$

When the distance between $2^{53+e} \cdot \sin x$ and $\mathbb{Z}$ is higher than $2^{53+e} \cdot 2^{-73.271} = 2^{-19.271+e}$ – where e is 1 in the case of a rounding to the nearest mode, and 0 for a directed rounding mode – the addition $h' + C^*$ gives the correct output. This implies that by using such a scheme the result is correctly rounded with probability at least $1 - 2^{-19.271}$, which makes the 99.9% estimate of Gal increase to $\approx 99.9998\%$.

3.2 A Function Evaluation Scheme Based on Gal's Method

In this subsection we describe how one could evaluate $2^x : [\frac{1}{2}, 1[\rightarrow [1, 2[$ in double precision with correct rounding by using only Gal's method, in the case of a directed rounding mode. As usual, there are two phases: the quick phase and the accurate phase.

The quick phase We use Gal's method with the table made of the 42-bad cases. This induces an error bounded by 2^{-41} for the term 2^{x_0}, and the problem is reduced to computing approximately 2^h where $|h|$ is smaller than $2^{-10.914}$ (see Section 5). We do this by evaluating the polynomial $P(h) = 1 + a_1 h + \ldots + a_4 h^4$, where a_i is the double that is closest to $\frac{(\ln 2)^i}{i!}$. It is possible to check that $P(h)$ can be evaluated with only operations on doubles to approximate 2^h with error bounded by $\approx 2^{-63}$ when $|h| \leq 2^{-10.914}$. This implies that if x is not a 10-bad case, the value calculated so far for 2^x is correct. Moreover, as a side effect, if x is a 42-bad case, the result can also be correctly rounded: for each entry $(x_0, 2^{x_0})$ of the table, we add a bit of information telling whether the stored value for 2^{x_0} is slightly larger or smaller than its real value.

The accurate phase Suppose now that the quick phase was not sufficient to guarantee a correct rounding of the output. We keep the same pair $(x_0, 2^{x_0})$ and only change the polynomial evaluation sub-phase. We evaluate in quadruple precision the polynomial $Q(h) = 1 + b_1 h + \ldots + b_7 h^7$ where b_i is the quadruple precision machine number that is closest to $\frac{(\ln 2)^i}{i!}$. It can be checked that $Q(h)$ can be evaluated with operations on quadruples to approximate 2^h with error bounded by $2^{-106.821}$ when $|h| \leq 2^{-10.914}$. Let $\bar{Q}(h)$ be the value computed for $Q(h)$, and y_0 be the value for 2^{x_0} that is stored in the table. We finally approximate 2^x by $y_0 \cdot \bar{Q}(h)$, the error being bounded by:

$$2^{-106.821+1} + 2^{-41-53} \cdot 1.000360 + 2^{-114+1} \leq 2^{-93.999}.$$

This implies that if x is not a 41.999-bad case, then the result is rounded correctly. Therefore, we have an implementation returning the correct rounding for any input x except those that are 41.999-bad cases but not 42-bad cases. We fix this point in the following way. We consider a table made only of 42.00072-bad cases instead of 42-bad cases, which means that we remove $6543825232174115/2^{53}$ from the look-up table and consider it as a special case. This removal does not change the bound on $|h|$. Moreover, it is easy to see that the error analysis just above now gives an final error bounded by 2^{-94}. Therefore the result is correctly rounded because we must be in one of the three following situations: x is not a 42-bad case, the error bound gives that the value returned for 2^x is correctly rounded; x is a 42.000072-bad case, the output is stored in the table; otherwise x is $6543825232174115/2^{53}$.

4 The Computation of the Tables

So far, we built our function evaluation scheme as if we knew explicitly the tables we described. Nevertheless, the task of computing the tables is by far nontrivial: if we were using an exhaustive search as Gal does, the cost of computing the table of either 2^x or $\sin x$ would be 2^{52} calls to 2^x (resp. $\sin x$) in extended precision. For 2^x the problem is easily solved once it is noted that the 42-bad cases can be calculated by Lefèvre's algorithm [13, 14] or the lattice-based algorithm of Stehlé, Lefèvre and Zimmermann [22, 23]. In this section, after giving some background on lattices and lattice reduction algorithms, we describe a method to construct simultaneously bad cases for two functions. For a given precision n, this algorithm admits a heuristic complexity of $\approx 2^{n/2}$ – instead of $\approx 2^n$ for the exhaustive search, and $\approx 2^{2n/3}$ with Lefèvre's algorithm.

4.1 A few Reminders on Lattices

In this subsection we briefly give some necessary background on lattices and on lattice reduction algorithms. We refer to [11, 16] for more details on both the algorithmic and mathematical aspects.

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

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

Notice that the rank ℓ of the lattice can be smaller than the dimension n of the embedding space. As soon as $\ell \geq 2$, a given lattice L admits an infinity of bases

related to each other by unimodular transformations. We define the *determinant* of the lattice L as $\det(L) = \prod_{i=1}^{\ell} ||\mathbf{b}_i^*||$, where $\mathbf{b}_1^*, \ldots, \mathbf{b}_\ell^*$ is the Gram-Schmidt orthogonalization of a lattice basis $\mathbf{b}_1, \ldots, \mathbf{b}_\ell$. This quantity does not depend of the choice of the basis. Most of the time, only bases which consist of short vectors are of interest. The *i-th minimum* of the lattice L is the smallest r such that the ball centered in $\mathbf{0}$ and of radius r contains at least i linearly independent lattice vectors. For example the first minimum $\lambda_1(L)$ is the length of a shortest non-zero lattice vector.

Since it corresponds to our situation in the following subsection we now suppose that $\ell = 4$ and $L \subset \mathbb{Z}^5$. It is classical [11] that in this case there always exists a basis reaching the four first minima, such a basis being called *Minkowski-reduced*, and that the first minimum of L is below $\sqrt{2} \cdot \det(L)^{1/4}$. Moreover, the well-known LLL algorithm [15] can be used to find a vector which is not much longer than the first minimum.

Theorem 1. *Given a basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_4]$ of a lattice $L \subset \mathbb{Z}^5$, if $M = max_i ||\boldsymbol{b}_i||$, the LLL algorithm provides in time $O(\log^3 M)$ a basis $[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4]$ satisfying:*

$$||\boldsymbol{v}_1|| \leq 4\lambda_1(L) \leq 4\sqrt{2}\det(L)^{1/4}.$$

This is not optimal because the output basis is not necessary Minkowski-reduced, and therefore the vectors may not be as short as possible. The greedy algorithm of [19] outputs a basis reaching the first four minima, and has the additional advantage of admitting a $O(\log^2 M)$ complexity bound.

4.2 An Algorithm for the $\sin x$-Table

The search over $[\frac{1}{2}, 1[$ is divided into $\frac{N}{2T}$ quick searches over intervals of length $\frac{T}{N}$. These quick searches are performed by using the algorithm we describe below.

Given two functions f_1 and f_2, a precision n, a bad-case bound m and a search bound T, the following algorithm tries to find all the machine numbers $x \in [-\frac{T}{2^n}, \frac{T}{2^n}]$ such that:

$$|2^n \cdot f_i(x) \bmod 1| \leq 2^{-m} \text{ for } i \in \{1, 2\}.$$

In fact it solves this problem for any N and M in place of 2^n and 2^m. It starts by approximating both f_1 and f_2 by polynomials, and then tries to solve the problem for these polynomials instead of the initial functions, by using Coppersmith's method to find small roots of multivariate modular polynomials [3, 4]. Our algorithm is heuristic: all its outputs are correct, but it may eventually fail. Nevertheless, the heuristic is

quite reasonable and the algorithm worked very well in our experiments described in Section 5: we follow the strategy of halving T until the algorithm does not fail, and it happens that the average T corresponds to what the theory predicts.

In the algorithm we use a routine LatticeReduce, which can either be the LLL algorithm [15] or the greedy algorithm of [19]. The input functions are made independent of N: $F_i(t) = Nf_i(t/N)$.

Algorithm `SimultaneousBadCaseSearch`.
Input: Two functions F_1 and F_2, and two positive integers M, T.
Output: All $t \in [-T, T]$ such that $|F_i(t) \bmod 1| < \frac{1}{M}$ for $i \in \{1, 2\}$.

1. Let $P_1(t), P_2(t)$ be the degree-2 Taylor expansions of $F_1(t), F_2(t)$.
2. Compute ε such that $|P_i(t) - F_i(t)| < \varepsilon$ for $|t| \leq T$ and $i \in \{1, 2\}$.
3. Let $M' = \lfloor \frac{1/2}{1/M+\varepsilon} \rfloor$, $C = 3M'$, and[2] $\tilde{P}_i(\tau) = \lfloor C \cdot P_i(T\tau) \rceil$ for $i \in \{1, 2\}$.
4. Let $e_1 = 1$, $e_2 = \tau$, $e_3 = \tau^2$, $e_4 = \upsilon$, $e_5 = \phi$.
5. Let $g_1 = C$, $g_2 = C \cdot \tau$, $g_3 = \tilde{P}_1(\tau) + 3\upsilon$, $g_4 = \tilde{P}_2(\tau) + 3\phi$.
6. Create the 4×5 integral matrix L where $L_{k,l}$ is the coefficient of the monomial e_l in g_k.
7. $V \leftarrow \text{LatticeReduce}(L)$
8. Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ be the three shortest vectors of V, and $Q_i(\tau)$ the associated polynomials.
9. If there exists $i \in \{1, 2, 3\}$ such that $||\mathbf{v}_i||_1 \geq C$, return(FAIL).
10. Let $Q(\tau)$ be a linear combination of the Q_i's which is independent of υ and ϕ. We have $\deg Q \leq 1$.
11. Let $q(t) = Q(\frac{t}{T})$. For each t_0 in IntegerRoots$(q, [-T, T])$ do
    if $|F_i(t_0) \bmod 1| < \frac{1}{M}$ for all $i \in \{1, 2\}$, then output t_0.

4.3 Correctness of the Algorithm

The following theorem gives the correctness of the algorithm.

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

Proof: Because of the final check in Step 11, we only have to check that no worst case is missed. Suppose there is $t_0 \in [-T, T]$ with $|F_i(t_0) \bmod 1| < \frac{1}{M}$ for $i \in \{1, 2\}$. From the definition of P_i, $|P_i(t_0) \bmod 1| < \frac{1}{M} + \varepsilon \leq \frac{1}{2M'}$. Since $|C \cdot P_i(T\tau) - \tilde{P}_i(\tau)| \leq \frac{3}{2}$ for $|\tau| \leq 1$, by choosing $\tau_0 = \frac{t_0}{T}$ we get $|\tilde{P}_i(\tau_0) \bmod C| < 3$.

Whence the g_i's have a common root $(t_0/T, \upsilon_0, \phi_0) \in [-1, 1]^3$, modulo C. Since the Q_i's are linear integer combinations of the g_i's, they admit a common root in $[-1, 1]^3$ modulo C, and even over the reals since $||\mathbf{v}_1||_1, ||\mathbf{v}_2||_1, ||\mathbf{v}_3||_1 \leq C$. Finally t_0 is an integer root of $q(t)$ and will be found at Step 11.

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

Working Precision In Step 1, we can use floating-point coefficients in the Taylor expansion $P_i(t)$ instead of symbolic coefficients, as long as it introduces no error in Step 3 while computing $\tilde{P}_i(\tau)$, for $i \in \{1,2\}$. Let a_j^i be the j-th Taylor coefficient of f_i. In order to get a correct $\tilde{P}_i(\tau)$ at Step 3, the error on $CN\left(\frac{T}{N}\right)^j \cdot a_j^i$ must be less than $\frac{1}{2}$, thus the error on a_j^i must be less than $\frac{1}{2C\cdot N}\left(\frac{N}{T}\right)^j$. Since $N \geq T$, it thus suffices to compute a_j^i with $\log_2(2CN) \leq 2n$ bits after the binary point. (We will see below that $C = O(N^{1/2})$.)

4.4 Complexity Analysis of the Algorithm

Now that the correctness of the algorithm is proved, we estimate its complexity. Let $a_0^i, a_1^i, \ldots$ be the Taylor coefficients of f_i for $i \in \{1,2\}$. Here we assume that for any k and i, $|a_k^i| = O(1)$.

Taylor's Bound Since we neglect Taylor coefficients of degree three and higher, the error made in the approximation to $N \cdot f_i(\frac{t}{N})$ by $P_i(t)$ is $\approx a_3^i \cdot T^3 \cdot N^{-2}$. Since we are looking for simultaneously bad cases with $|P_i(t) \bmod 1| < \frac{1}{M}$, we want $T^3 \cdot N^{-2} = O\left(\frac{1}{M}\right)$, i.e. $T^3 = O\left(\frac{N^2}{M}\right)$.

The Size of the Coefficients of the $\tilde{P}_i$'s The degree-2 polynomials $\tilde{P}_i(\tau) = \tilde{p}_0^i + \tilde{p}_1^i\tau + \tilde{p}_2^i\tau^2 \in \mathbb{Z}[\tau]$ computed at Step 3 of the algorithm satisfy $\tilde{p}_2^i = O(M \cdot T^2 \cdot N^{-1})$. Indeed, since $P_i(t) = p_0^i + p_1^i t + p_2^i t^2$ is the degree-2 Taylor expansion of $N \cdot f_i\left(\frac{t}{N}\right)$, we have $p_2^i = O\left(N^{-1}\right)$. Moreover, $\tilde{P}_i$ is defined by $\tilde{P}_i(\tau) = \tilde{p}_0^i + \tilde{p}_1^i\tau + \tilde{p}_2^i\tau^2 = \lfloor C \cdot P_i(T\tau) \rceil$. The fact that $C = \Theta(M)$ concludes the proof.

The Matrix Computed at Step 6 We have to reduce the lattice spanned by the rows of the following matrix.

$$L = \begin{bmatrix} C & & & \\ & T\cdot C & & \\ \tilde{p}_0^1 & \tilde{p}_1^1 & \tilde{p}_2^1 & 3 \\ \tilde{p}_0^2 & \tilde{p}_1^2 & \tilde{p}_2^2 & 3 \end{bmatrix}.$$

It is easy to compute the determinant of this lattice:

$$|\det(L)| = 3C^2 \cdot T \cdot \sqrt{(\tilde{p}_2^1)^2 + (\tilde{p}_2^2)^2 + 3} = O(M^3 \cdot T^3 \cdot N^{-1}).$$

Coppersmith's Bound In order to ensure the algorithm does not return FAIL at Step 9, we use Theorem 1 to provide at least one short vector: $\det(L)^{1/4}$ has to be smaller than C, which gives the bound $T^3 = O(M \cdot N)$. This is not enough to ensure that there are two other short vectors, but it seems that in practice the first three minima for these lattices are most often very similar.

Complexity Analysis We have two bounds for T, N and M:

$$\begin{aligned} M \cdot T^3 &= O(N^2) \quad \text{[Taylor's bound]} \\ T^3 &= O(M \cdot N) \text{ [Coppersmith's bound]} \end{aligned}$$

Since the complexity of the overall search — an exponent range of $N/2$ values — is approximately $\text{poly}(\log N) \cdot \frac{N}{T}$, the best choice of parameters is $T \approx M \approx N^{1/2}$, thus giving a heuristic complexity of $\approx \text{poly}(\log N) \cdot N^{1/2}$.

Remark: The technique used in Algorithm `SimultaneousBadCaseSearch` resembles very closely the algorithm of [22, 23], but things happen to be somehow simpler. For example, increasing the degree of the polynomial approximations or the dimension of the lattice by taking powers of the $\tilde{P}_i$'s is useless here: it seems that the degree-2 approximations and the four-dimensional lattice give a better complexity bound than any other choice of parameters.

5 Experimental Results

We give here the first values of the tables for 2^x and $\sin(x)$, the complete tables being available at the url: `http://www.loria.fr/~stehle/IMPROVEDGAL.html`.

5.1 The Table for 2^x

We give in Figure 1 the smallest 340 values $t_0 \in [|2^{52}, 2^{53}[|$ satisfying:

$$\left| 2^{52+\frac{t_0}{2^{53}}} \bmod 1 \right| \leq 2^{-m} \quad (2)$$

with $m \geq 41$. The table has been provided by Vincent Lefèvre and was computed with Lefèvre's algorithm for finding the worst cases of a one-variable function.

There are 4001 elements in the table, 1985 if we choose $m \geq 42$, 973 with $m \geq 43$, 491 with $m \geq 44$ and 265 if we choose $m \geq 45$. The maximum distance between two elements of the table is below $2^{-9.914}$, and the worst case is $t_0 =$ `13e34fa6ab969e` with $m = 52$.

t_0	m	t_0	m	t_0	m	t_0	m
1000a0933511b6	41	104ff6c8df89c9	41	10a8c959de17ec	43	10f537f3554cc4	42
10010b0e40f662	46	10515c7fae4713	41	10aa5620dd55d2	45	10f62e793af175	42
1003127f149599	42	1052861427ae4b	41	10aad829751d41	46	10f6af736baa7d	41
10090d6fac990e	43	105292757ba15b	41	10ab55c3bf3fdf	43	10f6b67d0223d6	41
100b80c24097f2	43	1054ad24cacf03	41	10af7642c57264	49	10f6faa7153803	42
100bab73fdcc3f	42	10570d95dae311	43	10af78aa5ddd8d	41	10f7218c776af6	41
100bb8ab1d1100	41	105746998b9f8f	43	10afe702aafa22	41	10f75282e65fa0	45
100cad551fe346	44	105c7910974a51	42	10b10471e500c7	41	10f8dbec829dec	45
100cbf828d7460	46	1060e7defdfccf	43	10b1880e15e187	43	10f92eec9c9f26	43
100d68962fcc81	42	10642abeaa122e	42	10b2204396d7cd	44	10f9f6c083f93a	41
100e0a8c90f689	43	10648724988ab3	41	10b290db417ae6	42	10faa42f732397	42
100e97b65850f4	42	106502aa0bb878	42	10b3dd3d6443dc	41	10fab4d705e116	42
10132731ce5be1	42	10663e1c7a91a3	41	10b3ecb63b653c	41	10fb15f0318546	42
10134149fe0d12	41	10689232cd0a19	42	10b432dd48904f	43	10fba014fc8116	45
101376279d2dc1	42	10699a598db731	46	10b4c0a2b70a93	41	10fc901159e14a	41
1014c13ae9fefc	41	106ab4067a924f	41	10b70e980e2075	43	10fca9fefc5ce3	42
1014cfa7e6f50f	47	106c243adf9733	41	10b81504a3b6f2	41	10fd7dadf5a3b3	42
10152a02a0e1d7	41	106d1c9260468d	41	10ba3cb1e670ac	41	10fdb62d4def4f	41
101572981d0476	43	106d5b7968bbef	42	10ba538c616e4a	44	10fe0f63679f1e	41
1016b1bae4d03c	41	106eef8395e912	41	10bacb2b5b7d57	44	10fe184d5cce12	41
1016bad257171c	41	106ef69bb3e581	41	10bb74a596b653	42	10ffd77d8b14f2	41
1017c31d491bba	41	106f8baf443960	44	10bbd7a9b2f059	41	1100c21c464f0f	41
1017f00d65af95	41	107083a6d6ffba	42	10bc0d7cdb97b9	41	1100f37c692673	42
1018b94b547a9a	42	1070fa9f51779e	41	10bc2ba5b4b4cc	42	1101072fc95068	41
1019574298668a	41	10714d132debe1	41	10bcd54838bce6	42	11028a32c4a2ef	47
101a1266c635ea	42	1071833084ab7c	41	10bdc58bea1e0f	41	1102cf995ce239	41
101a13477077f3	41	1074d76fa7bffd	41	10be49f351ff97	41	1102dd6d49caf7	42
101a9c6150a8f9	41	1076d298bcaf26	45	10be5559c3a5f8	41	11033ea6f703cc	43
101b8a60603ea4	41	107734e9125945	41	10beefd2889f5b	41	11036dfa18a53c	42
101cf8045e2370	41	1077986c944ea1	41	10c046edb7f731	41	1106772d539ba6	41
101d6254e73fd6	41	107895385f87a2	43	10c04e92273366	41	1106e0af2056c7	42
101e21cfb94e2e	41	1078c86d7ccf08	43	10c151dc12ecf7	46	11075aef9690f1	41
101e3bfced8a5e	41	10790f81ed8b1e	41	10c16de3c6cc4b	43	1107f07e070cf2	41
101e491f36b805	41	107a4bd8645636	44	10c258b14ba17e	43	1108f1fcc904fd	44
101f6ad0d7b36b	41	107c398749fda0	43	10c280f41a50fc	42	110b25bee87acb	41
10230fb167bfb7	41	107dd22eb97b83	42	10c45302f7e7f6	42	110bbdaa5d8667	41
102387b05e6545	43	107fd944a37068	41	10c54410eae788	43	110c1e5f0025df	41
10250b58310371	43	108035098c8b55	41	10c6debc012f7a	41	110f43fb4ca14f	41
102518534e84cf	42	10805acac3eebc	43	10c96aba034896	42	110f7109d13719	41
1026fbd8e9b8df	42	108307ef22c375	45	10cb16f6b62d7e	41	1110e91b2d1f3a	41
102a021633b58a	46	10831134b7a25e	44	10cbd2f904e167	41	111155105d592d	42
102a5b7f4f0846	41	1083a556fccd3c	42	10cbd94529a9e7	42	11118275139509	41
102a72437ae131	41	1083fb63a05848	43	10ccf098a6bd5e	45	1111a97b4f263f	43
102b5b6e340122	43	108436d22bfdf0	41	10ccf7d97f1a97	41	1113ec3ba733bb	41
102c5cb7e91e54	41	108454c9275a56	42	10cdcd4760a58c	42	1114ea9f9eba16	42
102cb72ba54fd3	42	108472f13c1638	45	10cecc7c07798a	46	11167c6b40c8e3	44
1030f46f21b28c	41	10847fa37272e4	42	10cf30936de093	42	1118fa37fa12c8	41
103104de6e26e9	41	1084f0ad44799b	42	10d0489ee954ac	41	111982798864ea	41
10310a0c5a76bd	41	108608d20c912d	42	10d16b2e672e96	41	1119a87f9ef18a	42

t_0	m	t_0	m	t_0	m	t_0	m
1031625a98e771	42	1087d1ed5a7904	41	10d233c636bf1b	43	111ad682071cce	41
103171d08132ea	41	10881b25a3e2d0	41	10d286a02df36e	41	111b9a9fb4702e	41
10338161238438	43	108956376cd6e2	41	10d3448f07be60	41	111bbeceb83d7e	42
1033fb60ab460c	42	10896be0d6df11	41	10d39996808b21	42	111e3804bf7be0	41
10340a91d235aa	41	1089c4408426f6	41	10d3a3af6ea778	41	11201db8d2f123	42
103430a4e98425	41	108aacdf72f200	42	10d3c7087a3e77	41	11204e081f1e4a	42
1034d2ecc35d1b	43	108b2ced385574	44	10d4b6d9fa9fb8	47	1120ec98297799	41
10359d4ea8f91d	41	108b3140cb3afa	43	10d4bf19a7d5c1	41	11217a561806ab	42
1037481ce4ae67	42	108d361ab20c9a	41	10d6ccaed7654b	41	11218ef8e0f270	47
1037f349bbbafb	43	108dab8b58528b	43	10d71a092d749f	45	1125590cc9c638	43
1038bce2bd1fde	41	108e0a6e0b77af	41	10d84b5bb44c12	41	1126ba1760d591	43
103b1b58d452eb	41	108ec85cf08dc1	41	10dc0dac7d009f	48	1127e2ee8f2539	43
103c4c4044638b	41	108fce687f9840	41	10dc4afd3a4b15	41	1127f7ad5b5639	43
103c4d302f6d0c	41	10918bb26cf0c4	41	10dca0dad94c53	45	1129192fc2bc3f	41
103d95f6dcb2d6	44	1093129e745f16	41	10dde88cdab4e3	42	1129a9855d1698	41
103e1d04638a7c	42	10948c3a13d895	41	10df60d8af554d	41	112af5b851ec12	41
103e6a4ab7c68a	43	10948f31d564a7	41	10e08b1f5c02a0	41	112b269c438074	42
1040ce08962562	42	109609c2f64999	43	10e1ccd8ba4cb8	42	112c999bf575ab	41
1041b900d4de0a	46	1096c7c75ad460	42	10e2fa2c5d1e48	41	112ccf71c46625	42
1041f3294776a0	41	109777be978a67	43	10e6f76ad97103	41	112d9611ca8ac4	41
10425b570ce231	41	109971e693c2dc	41	10e803eb7d750f	44	112f9c7338bd4b	41
1043683b2855da	41	1099875c6f8773	41	10e86de60591a5	41	113146b5988299	45
104465a2e29491	42	1099f7bf3b3398	42	10e88bdc0fff7d	44	1131b2d980608c	42
10460bf32f7445	42	109b98555a5238	42	10e8bc7fd88662	41	11326a0d38c3b1	45
10464be5261708	43	109ba07fa5f908	42	10ea1e34c15056	42	1132e10e16c70f	42
10465feabe21a5	42	109c319a483366	45	10eaacb1416c22	43	1135788566076f	43
104709a8fadd6c	41	109cf2a7b0f7ce	41	10eabbd10965b5	41	1136c818592a53	42
10471fd702c9d3	42	109d8467594476	44	10eb82c979df98	41	1139fd01c7c864	41
1047c04fd4d928	42	109f24264f525e	43	10eb87227fdda7	41	113a343467c9d9	43
1047e0f3bc9998	41	109fbd573b0093	41	10ebb62a06ac91	42	113a52cd9dd3b4	41
10484e9f051391	47	10a010bdd31967	41	10ecd4a9ab85eb	41	113b9c767982a9	41
104901619d35dd	44	10a0f0f9456c64	42	10eda1b07fd3b3	42	113e29b82ca375	41
104989b0163d8e	44	10a1de8993d77f	42	10edb4a0507dbe	41	113e30c3667506	41
104a0f5c73cabf	41	10a5403cb5a794	42	10ee017b71576c	47	113f887a0a026e	41
104aab3068aa67	41	10a559e66148c1	41	10efc482af6810	41	113fc2b1069abd	42
104fc5de473759	41	10a6afbe9cbe31	42	10f514cd3348ee	42	1141675df1591c	42

Fig. 1. The smallest 340 entries of the table for 2^x for double precision and $x \in [\frac{1}{2}, 1[$: for any entry, t_0, m satisfy equation (2).

5.2 The Table for $\sin x$

We give in Figure 2 the smallest 340 values $t_0 \in [|2^{52}, 2^{53}[|$ satisfying:

$$\left|2^{53+e} \cdot \sin \frac{t_0}{2^{53}} \bmod 1\right| \leq 2^{-m_1} \quad \text{and} \quad \left|2^{53} \cdot \cos \frac{t_0}{2^{53}} \bmod 1\right| \leq 2^{-m_2} \quad (3)$$

with $m_1, m_2 \geq 21$, $e = 1$ if $\frac{t_0}{2^{53}} \leq \frac{\pi}{6}$ and $e = 0$ otherwise. The table has been computed by an implementation of the algorithm described in Section 4 using GNU-MP [10] and MPFR [21], within a time equivalent to one day on a single Pentium IV, 4.3 GHz.

There are 4113 elements in the table, 1084 if we choose $m_1, m_2 \geq 22$ and 248 if we choose $m_1, m_2 \geq 23$. The maximum distance between two elements of the table is below $2^{-9.977}$, and the worst case is $t_0 =$ `31a93fddd45e3`, with $m_1, m_2 \geq 27$. Notice that all these values are very close to what predicts the random model.

We also started the calculation for the double extended precision (64 bits of mantissa) and quadruple precision (113 bits) for $x \in [\frac{1}{2}, 1[$. The worst simultaneous cases found so far are $x_0 =$ `aa349cb12135522b`$/2^{64}$ and $x_1 =$ `10000000004af2d94d4c848253af8`$/2^{113}$, with:

$$\left|2^{64} \cdot \sin x_0 \bmod 1\right| \leq 2^{-34} \quad \text{and} \quad \left|2^{64} \cdot \cos x_0 \bmod 1\right| \leq 2^{-35},$$
$$\left|2^{114} \cdot \sin x_1 \bmod 1\right| \leq 2^{-40} \quad \text{and} \quad \left|2^{113} \cdot \cos x_1 \bmod 1\right| \leq 2^{-40}.$$

Acknowledgements

We thank Vincent Lefèvre for providing the 2^x-table, and Florent de Dinechin and Nicolas Brisebarre for helpful discussions. We also thank the Medicis center for allowing the computation of the $\sin x$ table.

References

1. The Arenaire Project. Crlibm, 2004. `http://lipforge.ens-lyon.fr/projects/crlibm/`.
2. N. Brisebarre and J.-M. Muller. Finding the "truncated" polynomial that is closest to a function. INRIA Research Report No 4787, 2003. Updated version available at `http://arxiv.org/abs/cs.MS/0307009`.
3. D. Coppersmith. Finding a small root of a univariate modular equation. In *Proceedings of Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer-Verlag, 1996.
4. D. Coppersmith. Finding small solutions to small degree polynomials. In *Proceedings of CALC'01*, volume 2146 of *Lecture Notes in Computer Science*, pages 20–31. Springer-Verlag, 2001.

t_0	m_1	m_2	t_0	m_1	m_2	t_0	m_1	m_2	t_0	m_1	m_2
100005b33739b0	22	23	105bbaca3e4d1e	22	21	10abf25186b83d	22	21	10f59e428fee59	21	21
100041f50c3f1c	22	26	105e6be8cf5774	23	21	10ac2623c6b253	25	21	10f5ffd7f7107c	24	21
1001816a64dd2f	21	22	105ec2e64498ef	22	21	10ac2b88b6b488	24	21	10f78163b97ffd	26	25
100200c5c52b1e	24	22	105f8b988f6f00	21	26	10ac9abf467fae	22	21	10f9318c7eb8f2	21	27
100232ac6ced30	21	21	1060b539c55aa5	21	25	10acd45e3d39f6	22	22	10f94ea1a96655	21	22
1004a41a9c144b	23	21	10627fd75a5d5a	23	21	10ae4093fd8c6d	21	23	10f9bcd75e4c5c	22	21
1005133741a51b	21	21	1064cb8e18b925	21	25	10ae68db95a30e	22	21	10fcae0099cd49	21	21
1005bdc9e62331	24	23	10660397acbabe	21	21	10aedbcfe2aac1	22	21	10fcbdd65144a8	21	22
10082241803fdc	21	21	10672dfd9566b2	23	23	10aef857e93654	24	22	10fcc37204925b	21	21
100878de00f64e	25	21	1067ece33bb379	21	21	10b15a14b82070	21	21	10fd348c2eb9a8	21	21
100a05ecc34c4b	21	21	1068fb69f286b9	21	24	10b48e3e5c599b	22	21	10ffb048e5ac40	22	23
100b96f21a2cba	23	21	106914e6c86511	21	21	10b5014ab8ca06	23	22	10ffcea2062887	21	22
100b9c13f7af85	22	22	1069fce2287ee1	23	23	10b6121f6c1133	21	21	11006f2333aa3f	22	22
100c1acccb1200	24	22	106a51c672bddc	23	24	10b63b44b0807a	24	25	11019655c69be5	21	24
100cd15f52fa66	22	21	106a5d5f27cc30	21	21	10b68f0b9d2bb0	23	23	1101b3b0e03ac3	23	21
100d6012cd1521	21	21	106a66c1bd1af0	21	23	10b72056f91450	22	22	11029ca7d942b4	27	21
1010e3a483df8a	21	22	106aa87f18f8d3	22	22	10b9510108fe35	22	21	11040e1f5928dd	23	21
1013afb9a473c3	22	23	106b79196ede57	22	21	10ba273084d7e8	22	22	110481b7fa4244	24	21
101906bca03655	21	23	106b9c67c82478	22	22	10ba5d29c85560	21	22	11051e92ae1357	21	21
101ac9ca78b2cf	21	21	106c77da1638ac	22	22	10bbc36d6f047a	22	22	110786bd47c6f3	22	21
101ddd2fa5ec33	22	21	106e320e3186ed	23	21	10bc65e0ade65e	23	24	11092285dd3bd8	23	23
101e313e5941e2	21	21	106e34c6dc2225	24	22	10bccfb8c96420	21	21	110bc2d0a8c190	22	23
101e4f064a62d8	22	23	106e4290410bbf	22	21	10bcefa7d7a725	21	21	110c3cc9b3ca66	25	21
1021695200a512	24	21	10717f9ef24796	21	23	10be8b2d909c8b	23	21	110c427383ade0	24	21
1022377ecc3c61	21	22	10719476d554d0	21	22	10bf2e28314376	22	22	110d3614fb7d5d	21	21
10247db0a22a8a	21	21	107209eb970f41	23	26	10bf45c5c91750	21	21	110d698db23f82	24	23
1025204d12226f	23	21	10725c42ec1c1a	21	22	10bf99afd9b7df	21	21	110dbe5370afe7	21	21
102837b0141d55	21	25	10734f4ae296eb	22	25	10c000c83bb309	21	22	1110283255bcf5	21	24
102843d0813d4b	21	27	1074113ce06309	21	21	10c0204927eb55	21	24	111041f0511a79	22	21
10287095e29e03	21	22	10757b5afc961e	23	23	10c12006ff9642	22	21	1110935eb5613b	23	21
1029d934cf0c8b	23	21	1076bf672db4e5	21	21	10c1ddb34cfe61	23	21	1111642c8d6053	23	22
102a3da792d082	22	21	1076c8b1c9e2e3	24	21	10c34ae4e19a3d	23	21	1111fb819e95aa	23	21
102a79ae9e5afc	22	22	1078f079e3870c	23	25	10c361932d629f	24	26	1113673ada3c8b	22	23
102bc1f2791512	21	21	107939069742d3	21	21	10c4c4d1ec0efe	22	21	1113abb7a2bada	21	22
102bf46dc54edb	21	21	107a8820ff766c	22	21	10c62919966c63	22	22	1113c39c7378b5	24	25
102c00f99a9eb5	23	23	107b464c692a80	21	22	10c6aa32a2a90f	22	21	11159177938bc6	22	22
102dea6280480e	21	22	107b4c9e24983e	22	21	10c844926fe873	26	21	11177806cbb6ef	21	21
102ec7d7278c68	21	22	107c1085936936	21	21	10cad8ebeae4dd	22	21	1117fa42f9d367	22	21
102f0f8422829d	24	24	107faddd7de242	21	21	10cada59a5038c	21	22	11180c4267dd14	21	22
102f1f660892f1	23	21	107ff8f631e4fd	21	21	10cb67e06de678	22	22	111994a808000d	22	21
10318ce9536c14	21	21	1081798f564455	22	21	10cb808bc636f2	21	23	111a4701603bc2	24	22
1032384fe53575	22	21	1081b154cca8c7	22	21	10cb8faa0102b3	23	21	111a90f95aa8fa	22	21
1032d5874e5600	21	21	1082324dc77240	23	22	10cb90aa7efe73	21	21	111a9bc4c969b0	21	25
10339e8a9ab15e	22	21	1082b1163b1db6	21	21	10cba073a49f9b	25	21	111b0830584f68	21	25
1033b15b3b4b49	22	21	1085d06d52b5a6	23	21	10cdd8553220a0	22	25	111b2ccdb0a1ec	22	21
1033dd8384e176	23	23	1086d1d94c7074	21	21	10ce0eac08faa9	23	21	111bbf54bfc577	23	21
1034ae60f96942	21	23	108724a5e4a7cd	21	22	10ceaa111a431c	21	24	112006ba015aa8	21	21
1035de70672b94	21	21	108725142e5d25	21	21	10cf1b3485c6d8	21	23	112128339be9ce	21	22

t_0	m_1	m_2	t_0	m_1	m_2	t_0	m_1	m_2	t_0	m_1	m_2
10360bdc0503e3	22	21	1089057fe8bf9f	25	21	10cf5bb959f981	23	22	1121f7a14911f3	24	22
103659b3264f1a	23	21	108a2bac4bd5a0	23	22	10d11cec9795c7	22	21	11222a01a53222	22	24
103933fb42490a	22	23	108a64810eed5a	21	31	10d165e20883df	22	21	112238cdbafd43	29	21
1039520b39de3f	22	21	108b0bcc0f3db4	23	22	10d26e57593eda	21	25	112294fd009e8b	25	22
103bb05ed2ed15	21	24	108dbf125b14ba	21	22	10d28be4e41899	21	21	112570dc7b64fb	21	21
103befe59d099d	21	21	108e1b53376a09	23	21	10d299c02ceb06	21	21	112683c12c65cc	25	21
103dc6bb0320a5	21	21	108e5c34488fce	21	24	10d33ac8e428ca	21	21	112748f1a0449f	23	21
103e3187526ca7	21	22	108e73339d096e	21	22	10d3f3728c9e19	31	22	11280a807fa60b	23	21
103e9455296b10	21	21	108eaebf9e6c04	22	22	10d4baf3576aa6	21	21	1129ba11920e62	24	21
103fbb4c9ab033	23	23	108ef1c5d492f1	21	21	10d622cabf5cb9	23	22	112acae9fb365c	21	22
103fbfeba05c63	22	22	108f072adb9f88	22	22	10d697dfe9c4fb	22	21	112b1dbcae3909	22	21
1041534a083934	22	25	108f4179ad4df0	21	21	10d94d7ff524f8	21	21	112b4488fc7bce	21	32
1042d54e1489c3	21	22	10900cb646148d	21	22	10d9bf905c9e20	23	21	112bdb7b4f06f2	21	21
1044d7e90231b0	21	21	1090b399efa956	22	26	10db53395a9977	22	22	112bf757346083	24	21
10453a4a9a4a8f	21	21	1090f70c0e9aca	21	22	10dcd14a30e9bb	21	26	112c3569bcf35e	22	21
104559f45e94c4	22	22	10947bdf01eb1d	21	21	10dde90d53b2c5	24	21	112d0789cb6024	21	26
1046f9cc56e6ea	21	21	109615db97d66e	22	21	10dfcc2ec6a0d4	23	26	112f049d4c80f1	24	25
1047c2abd5b72b	21	22	10979549e32d13	22	21	10e001ba088b2f	25	21	113165fdf9f7bc	22	21
1048bd23a4dd2e	23	21	1097a64bfda33d	21	23	10e021a9120121	21	21	113221b693f823	22	25
10495faa26160e	22	25	109896c9f914e5	22	21	10e0acae70d54d	21	21	1132ea6b8a355c	23	21
10496656c1ff87	21	25	109aec630233bf	21	25	10e321fb4d1fef	21	26	11334f7db2405a	21	23
104aa7ea3874c9	23	21	109b4bf4f29d94	23	23	10e604a4d17649	21	22	1133b657a730f0	21	21
104b3d3c366b58	21	21	109b7e3645d8b8	21	21	10e799f2cdf0dc	21	21	1135b308e2e218	22	24
104b4a97f4f5c6	21	21	109b9f6a2a429e	22	21	10e7c358af2da5	21	27	11362fdb5d1dc2	23	21
104bc374f21721	21	22	109beb7f54460e	21	21	10ea1fcfa7f8a9	23	21	11365bb3bd4b8f	22	23
104c68d1bb4c4e	23	22	109d690bfe1789	21	21	10eac47da3b32d	23	21	11385af2ecdd6b	25	24
104ca7c46330df	21	24	109ea1505a7c57	21	21	10ead0afe7c2eb	22	21	1138a5b94ab537	25	21
104d3270d89b8e	25	21	109ed784ceb61c	22	22	10ec03f654b512	21	21	1138d925dbd757	21	23
104fd823dbc039	21	22	10a0b3bb1a3772	22	21	10eca434738809	24	21	113935408b7f24	22	22
10539e9100f11a	21	21	10a1e5acdcefc8	21	21	10ed37c0c95289	23	23	113a3c3b12f666	22	21
1053e86dee8a64	22	21	10a2e0fcf91802	21	22	10edf546a9b982	24	22	113a6bc755fe7f	22	22
1055a709cbdab7	21	21	10a60c73574756	21	21	10ee8aa36ae75a	22	22	113bce3e99d426	22	22
1056574aedec17	21	21	10a80f3bd4577d	22	22	10ef07c49468d0	21	22	113defedb485ef	21	21
10590dd662dc15	22	21	10a8607f38af1b	21	23	10f00918f9aad6	24	21	113ea2edb89253	21	22
1059e587fa8996	25	22	10a8a9201db329	24	24	10f0486591187e	24	21	113f80945eca69	23	21
105b8d1ff31578	22	21	10a9e86bb6103f	21	22	10f0f85d38beee	23	21	1140ddb619caac	21	23
105bb056ca1b53	21	32	10aae00a9a3852	21	22	10f483062d50c6	21	22	11425defb64c47	21	22

Fig. 2. The smallest 340 entries of the table for $\sin x$ for double precision and $x \in [\frac{1}{2}, 1[$: for any entry, t_0, m_1, m_2 satisfy equation (3).

5. D. Defour. Cache-optimized methods for the evaluation of elementary functions. LIP Research Report RR2002-38, 2002. Available at `ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR2002/RR2002-38.ps.Z`.
6. D. Defour, F. de Dinechin, and J.-M. Muller. Correctly rounded exponential function in double precision arithmetic. In *Proceedings of SPIE 46th Annual Meeting, International Symposium on 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*. To appear.
8. S. Gal. Computing elementary functions: a new approach for achieving high accuracy and good performance. In *Proceedings of Accurate Scientific Computations*, volume 235 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1986.
9. S. Gal and B. Bachelis. An Accurate Elementary Mathematical Library for the IEEE Floating Point Standard. *ACM Transactions on Mathematical Software*, 17(1):16–45, 1991.
10. T. Granlund. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 4.1.4 edition, 2004. Available at `http://www.swox.se/gmp/`.
11. P. M. Gruber and C. G. Lekkerkerker. Geometry of Numbers, second edition. Amsterdam, North-Holland, 1987.
12. IEEE standard for binary floating-point arithmetic. Technical Report ANSI-IEEE Standard 754-1985, New York, 1985.
13. V. Lefèvre. *Moyens arithmétiques pour un calcul fiable*. Thèse de doctorat, École Normale Supérieure de Lyon, 2000.
14. V. Lefèvre and J.-M. Muller. Worst cases for correct rounding of the elementary functions in double precision. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 111–118. IEEE Computer Society, 2001.
15. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261:515–534, 1982.
16. L. Lovász. An Algorithmic Theory of Numbers, Graphs and Convexity. *SIAM lecture series*, 50, 1986.
17. J.-M. Muller. Elementary Functions, Algorithms and Implementation. Birkhauser Boston, 1997.
18. J.-M. Muller. Proposals for a specification of the elementary functions. In *Abstracts of SCAN'2002*, pages 54–55. Laboratory LIP6, Paris, France, 2002.
19. P. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited (extended abstract). In Proceedings of ANTS VI, volume 3076 of *Lecture Notes in Computer Science*, pages 338–357. Springer-Verlag, 2004.
20. M. Payne and R. Hanek. Radian reduction for trigonometric functions. *SIGNUM Newsletter*, 18:19–24, 1983.
21. The Spaces Project. The MPFR library, version 2.0.3, 2004. `http://www.mpfr.org/`.
22. D. Stehlé, V. Lefèvre, and P. Zimmermann. Worst cases and lattice reduction. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic*, pages 142–147. IEEE Computer Society, 2003.
23. D. Stehlé, V. Lefèvre, and P. Zimmermann. Searching Worst Cases of a One-Variable Function. To appear in *IEEE Transactions on Computers*, 2005.
24. A. Ziv. Fast Evaluation of Elementary Mathematical Functions with Correctly Rounded last Bit. *ACM Transactions on Mathematical Software*, 17(3):410–423, 1991.
25. A. Ziv. MathLib, 2004. `http://www-124.ibm.com/developerworks/oss/mathlib/`.

Appendix:

We study here the expected maximum distance between two bad cases of a function f under the random model.

Lemma 3. *Let $f : [\frac{1}{2}, 1[\rightarrow [\frac{1}{2}, 1[$ satisfying the random model assumption. Let n be the precision and $2 \leq p \leq n$. Let M be the maximum distance between two consecutive p-bad cases for f. We have:*

$$E[M] \leq \left(1 + n - \frac{p}{2}\right) \cdot 2^{p-n}.$$

Proof: The probability of having a run of at least k consecutive machine numbers that are not p-bad cases for f is bounded by:

$$2^{n-1} \cdot \left(1 - 2^{1-p}\right)^k,$$

because there are less than 2^{n-1} starting points for the run. We now fix $k = \left(n - 1 - \frac{p}{2}\right) \ln 2 \cdot 2^p$. Using the fact that $\log(1-x) \leq -x - x^2$ for any $x \in [0, \frac{1}{2}]$, we can bound the probability above by:

$$2^{n-1+2^p(n-1-\frac{p}{2})\log(1-2^{1-p})} \leq 2^{n-1-2^p\left(n-1-\frac{p}{2}\right)(2^{1-p}+2^{2-2p})} \leq 2^{p-n+1} \cdot 2^{2^{2-p}} \leq 2 \cdot 2^{p-n+1},$$

from which we obtain that:

$$\begin{aligned} E[M] &\leq k \cdot 2^{-n} + \frac{1}{2} \cdot 2 \cdot 2^{p-n+1} \\ &\leq 2^{p-n} \cdot \left(n + 1 - \frac{p}{2}\right), \end{aligned}$$

which ends the proof.

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 Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY 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 - 38334 Montbonnot Saint-Ismier (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