# Gal's Accurate Tables Method Revisited

Damien Stehlé
UHP/LORIA
615 rue du jardin botanique
F-54602 Villers-lès-Nancy Cedex
stehle@loria.fr

Paul Zimmermann
INRIA Lorraine/LORIA
615 rue du jardin botanique
F-54602 Villers-lès-Nancy Cedex
zimmerma@loria.fr

## Abstract

*Gal's accurate tables algorithm aims at providing an efficient implementation of mathematical functions with correct rounding as often as possible. This method requires an expensive pre-computation 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 accuracy of the function implementation, 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 demonstrate how the whole method can be turned into practice for* $2^x$ *and* $\sin x$ *for* $x \in [\frac{1}{2}, 1[$*, in double precision.*

**Introduction.** The IEEE-754 standard for floating-point arithmetic [10] requires the four basic arithmetic operations and the square root to be correctly rounded, but does not require a correct rounding for any other mathematical function, like trigonometric, exponential and logarithmic functions. For this reason, there are libraries which produce incorrectly rounded results, and this causes serious difficulties for the portability and reproducibility of scientific calculations. The reticence to extend the standard to the mathematical functions may come from the fact that an implementation guaranteeing a correct rounding is too expensive. The March 2001 meeting of the IEEE-754 revision committee concluded with the following note:

> *Transcendentals — these are too hard to standardize now because nobody knows how to tradeoff precision vs performance. If less than correct rounding is specified [. . . ] then 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 mathematical functions, namely Gal's accurate tables method [6, 7, 13], in the hope it will shrink the efficiency gap between mathematical functions libraries in use and those guaranteeing a correct rounding [1, 20, 22], and give more support for proposals of standardization of these functions [5].

The implementation of a mathematical function over its full domain for some given precision usually requires two phases (see [4] 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 uses a few more bits in a very few steps), while the accurate phase is often based on Ziv's strategy [21] which extends the working precision until the result can be guaranteed correctly rounded (eventually, one may also use a sharp bound on the required precision if such a bound is known [11]). 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 $2^x$ itself), 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 small degree polynomial approximating the function (or the related functions) over the restricted range.

There are very satisfactory answers for the first [16] and last sub-phases [2]. Gal's method addresses the second subphase. The original technique is based on a table of "almost regularly spaced" distinguished points whose images by the function (or the related functions) are unusually close to machine numbers. The table, of size a few kilobytes, is obtained through 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) when rounding towards $0$ or $\pm\infty$: this problem is therefore closely related to the Table Maker's Dilemma (TMD for short). A direct consequence is that we can use the methods finding the worst cases of a one-variable function [11, 18] 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 extend the lattice-based algorithm of [18] to find simultaneously bad cases for two functions.

Although the method can be easily generalized to other functions and to arbitrary precision, we focus here 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 probability that the quick phase fails decreases drastically 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 [11]. It requires only quadruple precision calculations.

The rest of the paper is organized as follows. In §1 we recall some basic facts about the TMD and Gal's accurate tables method. In §2 we describe what is the best set of distinguished values in the table and how this choice improves the quick and accurate phases. In §3 we explain how to obtain these tables. Finally, in §4 we demonstrate the practicability of the method for $2^x$ and $\sin x$, in double precision.

*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 double-precision floating-point numbers in $[\frac{1}{2}, 1[$. For any reals $x$ and $c$, $x \bmod c$ denotes $x - \lfloor \frac{x}{c} \rceil$ — if $\frac{x}{c}$ is half an odd integer, we choose any of the possibilities. In particular, $x \bmod 1$ is the "centered" fractional part of $x$. We denote by $\diamond(x)$ 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 $\|(x_1, \ldots, x_n)\|_1 = \sum_{i=1}^n |x_i|$.

## 1. 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.

**The Random Model.** We consider that a function behaves as a random black-box when the first bits of its output are disregarded. Such an assumption is very strong and completely heuristic. Nevertheless experiments tend to validate this hypothesis and in particular the experiments of §4 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 a run of $p$ consecutive identical bits — starting from some given position — in the binary expansion of $f(x)$ is $2^{1-p}$.

- 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 such consecutive $x$'s is less than $n \cdot 2^{n-p}$ (see the Appendix).

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 derive some easy properties:

- The probability of having 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 $2^{2-2p}$.
- The set of the $x \in [\frac{1}{2}, 1[_n$ such that both $f_1(x)$ and $f_2(x)$ have $p$ 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 $\approx n \cdot 2^{n+1-2p}$.

These assumptions may seem surprising in the case of $\sin$ and $\cos$ since we have $\sin^2 + \cos^2 = 1$, but for our purpose this seems reasonable since it is not contradicted by the experiments of §4.

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

**Gal's Accurate Tables Method.** Gal's accurate tables method [6, 7] 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 use the equation:

$$2^x = 2^{x_0} \cdot 2^h, \text{ or } \sin x = \sin x_0 \cdot \cos h + \cos x_0 \cdot \sin h,$$

where $x_0$ is in a table of distinguished points and is stored with an approximation of its corresponding $\diamond(2^{x_0})$ (resp. $\diamond(\sin x_0)$ and $\diamond(\cos x_0)$); $h = x - x_0$ 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. Finally, it is interesting to have tables with approximately $2^{10}$ distinguished elements, for cache optimization reasons.

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:

$$|2^{52} \cdot 2^{x_0^{(i)}} \bmod 1| \leq 2^{-10}, \text{ or}$$

$$|2^{53+e} \cdot \sin x_0^{(i)} \bmod 1|, |2^{53} \cdot \cos x_0^{(i)} \bmod 1| \leq 2^{-10},$$

where $e = 1$ if $|x_0^{(i)}| \leq \frac{\pi}{6}$ and 0 otherwise. According to the random model, $x_0^{(i)}$ is very close to $\frac{1}{2} + \frac{i}{2^{11}}$, implying a negligible increase of the value that can be taken by $|h|$. 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 $\approx 63$ 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.

**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. An *$m$-bad case* for $f$ is a $n$-bit machine number $x$ 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 an $m$-bad case for $2^x$ if $|2^{52+x} \bmod 1| \leq 2^{-m}$, an $m$-bad case for $\cos x$ if $|2^{53} \cdot \cos x \bmod 1| \leq 2^{-m}$, an $m$-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 an $m$-bad case for the given format, 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 unreasonably expensive because of its $\approx 2^n$ complexity. Lefèvre and Muller proposed a $\approx 2^{2n/3}$ algorithm [11] , which was sufficient to perform some systematic work in double precision. Stehlé, Lefèvre and Zimmermann gave a generalization of this algorithm by using lattice-based Coppersmith's work to find small roots of modular polynomials. They obtained an algorithm with a heuristic $\approx 2^{5n/7}$ complexity [18]. Although these complexity bounds are better than the one of Lefèvre's algorithm, it seems that the practical cutoff between both methods is around the double extended precision, i.e. $n = 64$.

## 2. 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).

### 2.1. A Table Made of Bad Cases

Under the random model assumption, it is possible to strengthen Gal's requirements on the run lengths 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 §4. 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 the error made while evaluating the polynomial approximating $2^h$ for $h$ close to 0 is roughly the same: if the coefficients of the polynomial are doubles, 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 $\approx 2^{12}$ of them, and the average distance between two consecutive ones is below $2^{-9.977}$ as shown by the experiments of §4. 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

$$S(h) = h - a_3h^3 + a_5h^5 \text{ and } C(h) = 1 - \frac{1}{2}h^2 + a_4h^4,$$

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:

$$\begin{aligned}
&|\sin h - S(h)| \\
&\leq |\sin h - (h - \tfrac{h^3}{6} + \tfrac{h^5}{5!})| + |a_3 - \tfrac{1}{6}||h|^3 + |a_5 - \tfrac{1}{5!}||h^5| \\
&\leq 2^{-76.839-12.299} + 2^{-56.584-32.931} + 2^{-62.906-54.885} \\
&\leq 2^{-88.314}, \\
&|\cos h - C(h)| \\
&\leq |\cos h - (1 - \tfrac{h^2}{2} + \tfrac{h^4}{24})| + |a_4 - \tfrac{1}{24}||h|^4 \\
&\leq 2^{-65.862-9.491} + 2^{-58.584-43.908} \leq 2^{-75.352}.
\end{aligned}$$

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:

| formula | error bound | bound |
|---|---|---|
| $k \leftarrow \diamond(h^2)$ | $2^{-75}$ | $2^{-21.953}$ |
| $k' \leftarrow \diamond(h \cdot k)$ | $2^{-84.988}$ | $2^{-32.930}$ |
| $S^* \leftarrow \diamond(a_5 \cdot k)$ | $2^{-80.952}$ | $2^{-28.859}$ |
| $S^* \leftarrow \diamond(S^* - a_3)$ | $2^{-55.999}$ | $2^{-2.584}$ |
| $S^* \leftarrow \diamond(k' \cdot S^*)$ | $2^{-86.754}$ | $2^{-35.513}$ |
| $C^* \leftarrow \diamond(a_4 \cdot k)$ | $2^{-78.777}$ | $2^{-26.536}$ |
| $C^* \leftarrow \diamond(C^* - \frac{1}{2})$ | $2^{-55.999}$ | $2^{-1.000}$ |
| $C^* \leftarrow \diamond(k \cdot C^*)$ | $2^{-74.824}$ | $2^{-22.951}$. |

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 fused multiply-add (fma). For example with a fma:

- $h' \leftarrow \diamond_{\text{fma}}(s_0 + c_0 \cdot h)$,
- $t \leftarrow \diamond(h' - s_0)$,
- $l \leftarrow \diamond_{\text{fma}}(t - c_0 \cdot h)$.

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 $h' + l = s_0 + c_0 \cdot h$ and

$$|(h' + l) - (\sin x_0 + \cos x_0 \cdot h)| \leq 2^{-73.999}.$$

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

| formula | error bound | bound |
|---|---|---|
| $S^* \leftarrow \diamond(c_0 \cdot S^*)$ | $2^{-86.631}$ | $2^{-35.700}$ |
| $C^* \leftarrow \diamond(s_0 \cdot C^*)$ | $2^{-74.610}$ | $2^{-23.199}$ |
| $C^* \leftarrow \diamond(S^* + C^*)$ | $2^{-74.609}$ | $2^{-23.198}$ |
| $C^* \leftarrow \diamond(l + C^*)$ | $2^{-73.271}$ | $2^{-23.197}$. |

When the distance between $2^{53+e} \cdot \sin x$ and $\mathbb{Z}$ is larger than $2^{(53+e)-73.271} = 2^{-19.271+e}$ — 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 $\approx 99.9998\%$.

### 2.2. An 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 $2^{-94}$ 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 §4). We do this by evaluating the polynomial $P(h) = 1 + a_1 h + \ldots + a_4 h^4$, where $a_i = \diamond\left(\frac{(\ln 2)^i}{i!}\right)$. 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 $(x_0, \diamond(2^{x_0}))$ 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 $(x_0, \diamond(2^{x_0}))$ 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 $Q(h)$. We finally approximate $2^x$ by $\diamond(2^{x_0})\bar{Q}(h)$ in quadruple precision, the error being bounded by:

$$2^{-106.821+1} + 2^{-94} \cdot 1.000360 + 2^{-114+1} \leq 2^{-93.999}.$$

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 $\texttt{173f930a6f9c23}/2^{53}$ 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: if $x$ is not a $42$-bad case, the error bound implies that the value returned for $2^x$ is correctly rounded; if $x$ is a $42.000072$-bad case, the output is stored in the table; otherwise $x$ must be $\texttt{173f930a6f9c23}/2^{53}$.

## 3. 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 far from trivial: 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. $\sin x$ and $\cos x$) 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 [11] or the lattice-based algorithm of Stehlé, Lefèvre and Zimmermann [18]. 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 has 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.

**A few Reminders on Lattices.** 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 $L = \left\{\sum_{i=1}^{\ell} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\right\}$. Notice that the rank $\ell$ 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. 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 is independent 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 [9] 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 algo-

rithm [12, 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 $O(\log^2 M)$ a basis $[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4]$ of $L$ 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 necessarily Minkowski-reduced, and therefore the vectors may not be as short as possible. The greedy algorithm of [14] outputs a basis reaching the first four minima.

### 3.1. 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 algorithm tries to find all the machine numbers $x \in [-\frac{T}{2^n}, \frac{T}{2^n}]$ with:

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

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]. 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 §4: we followed the strategy of halving $T$ until the algorithm does not fail, and we found an average $T$ corresponding to the theory.

In the algorithm we use a routine LatticeReduce, which can be for example one of [12, 14, 15]. The input functions are made independent of $N$: $F_i(t) = Nf_i(t/N)$.

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 $C = O(N^{1/2})$.

**Theorem 2** *In case* `SimultaneousBadCaseSearch` *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 $i \in \{1, 2\}$.*

---

**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 $\varepsilon$ such that $|P_i(t) - F_i(t)| < \varepsilon$ for $|t| \leq T$ and $i \in \{1, 2\}$.
3. Compute $M' = \lfloor \frac{1/2}{1/M + \varepsilon} \rfloor$, $C = 3M'$, and[1] $\tilde{P}_i(\tau) = \lfloor C \cdot P_i(T\tau) \rceil$ 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 $g_1 = C, g_2 = CT \cdot \tau, g_3 = \tilde{P}_1(\tau) + 3\upsilon, g_4 = \tilde{P}_2(\tau) + 3\phi$.
6. Create the $4 \times 5$ integral matrix $L$ where $L_{k,l}$ is the coefficient of the monomial $e_l$ in $g_k$.
7. $V \leftarrow \mathrm{LatticeReduce}(L)$
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 $\upsilon$ and $\phi$. We have $\deg Q \leq 1$.
11. Let $q(t) = Q(\frac{t}{T})$. For each $t_0$ in $\mathrm{IntegerRoots}(q, [-T, T])$ do
    if $|F_i(t_0) \bmod 1| < \frac{1}{M}$ for all $i \in \{1, 2\}$, then output $t_0$.

---

**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 modulo $C$, namely $(\frac{t_0}{T}, \upsilon_0, \phi_0) \in [-1, 1]^3$. 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. ∎

### 3.2. Complexity Analysis of the Algorithm

We now bound the complexity of the algorithm. Let $a_0^{(i)}, a_1^{(i)}, \ldots$ be the Taylor coefficients of $f_i$ for $i \in \{1, 2\}$. 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 $\approx \frac{a_3^{(i)} T^3}{N^2}$. 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.

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

It is easy to compute the determinant of this lattice:

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

**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:** Two bounds must be satisfied:

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

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 the present algorithm resembles very closely the algorithm of [18], 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.

## 4. Experimental Results

We give in [19] the values of the tables for $2^x$ and $\sin x$.

**The Table for $2^x$.** For $2^x$, we give the values $t_0 \in [|2^{52}, 2^{53}[|$ satisfying:

$$\left|2^{52+\frac{t_0}{2^{53}}} \bmod 1\right| \leq 2^{-41}.$$

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 $m = 52$.

**The Table for $\sin x$.** For $\sin x$, we give the values $t_0 \in [|2^{52}, 2^{53}[|$ satisfying:

$$\left|2^{53+e} \cdot \sin\frac{t_0}{2^{53}} \bmod 1\right|, \left|2^{53} \cdot \cos\frac{t_0}{2^{53}} \bmod 1\right| \leq 2^{-21}.$$

with $e = 1$ if $\frac{t_0}{2^{53}} \leq \frac{\pi}{6}$ and $e = 0$ otherwise. The table was computed by an implementation of the algorithm described in §3 using GNU-MP [8] and MPFR [17], in 9 hours on a 2.4 Ghz Opteron, instead of about 850 years with the naive algorithm.

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 the random model predicts.

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:

$$\begin{array}{rcl} |2^{64} \sin x_0 \bmod 1|, |2^{64} \cos x_0 \bmod 1| & \leq & 2^{-34}, \\ |2^{114} \sin x_1 \bmod 1|, |2^{113} \cos x_1 \bmod 1| & \leq & 2^{-40}. \end{array}$$

**Conclusion.** We presented a faster method to compute entries for Gal's accurate tables method, using lattice reduction. As a consequence, it becomes feasible to compute the best possible entries, which greatly improves the accuracy of Gal's method. A possible improvement for unary functions would be to search for simultaneous worst cases of the first two coefficients in the Taylor expansion, i.e. $f(x_0)$ and $f'(x_0)$. If $f(x_0)$ and $f'(x_0)$ are not correlated, the algorithm of §3.1 should work, maybe after slight modifications to unbalance the extra bits between $f(x_0)$ and $f'(x_0)$. Consider for example $x_0 = 4503599642142214/2^{53}$ in double precision: $x_0$ is 20-bad for $2^x$, and 13-bad for $2^x \cdot \log 2$, which means that $\diamond(2^{x_0}) + h \diamond (2^{x_0} \cdot \log 2) + P(h)$ yields about 73 bits of accuracy of $2^{x_0+h}$ for $|h| \leq 2^{-10}$, where $P(h)$ containing the Taylor coefficients of degree $\geq 2$ can be evaluated in double-precision,

**Acknowledgements.** We thank Vincent Lefèvre who provided the $2^x$-table; Florent de Dinechin, Nicolas Brisebarre, David Defour and the members from the DALI team for helpful discussions; and both anonymous referees for their comments.

## 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.

[3] 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.

[4] 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.

[5] D. Defour, G. Hanrot, V. Lefèvre, J.-M. Muller, N. Revol, and P. Zimmermann. Proposal for a Standardization of Mathematical Function Implementation in Floating-Point Arithmetic. *Numerical Algorithms*, 37(1-4):367–375, 2004.

[6] 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.

[7] 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.

[8] T. Granlund. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 4.1.4 edition, 2004. Available at http://www.swox.se/gmp/.

[9] P. M. Gruber and C. G. Lekkerkerker. Geometry of Numbers, second edition. Amsterdam, North-Holland, 1987.

[10] IEEE standard for binary floating-point arithmetic. Technical Report ANSI-IEEE Standard 754-1985, New York, 1985.

[11] 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.

[12] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Math. Annalen*, 261:515–534, 1982.

[13] J.-M. Muller. Elementary Functions, Algorithms and Implementation. Birkhauser Boston, 1997.

[14] 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.

[15] P. Nguyen and D. Stehlé. Floating-Point LLL Revisited. In *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer-Verlag, 2005.

[16] M. Payne and R. Hanek. Radian reduction for trigonometric functions. *SIGNUM Newsletter*, 18:19–24, 1983.

[17] THE SPACES PROJECT. The MPFR library, version 2.1.1, 2005. http://www.mpfr.org/.

[18] D. Stehlé, V. Lefèvre, and P. Zimmermann. Searching Worst Cases of a One-Variable Function. *IEEE Transactions on Computers*, pages 340–346. IEEE Computer Society, 2005.

[19] D. Stehlé, and P. Zimmermann. Gal's Accurate Tables Method Revisited. http://www.loria.fr/~stehle/IMPROVEDGAL.html, 2004.

[20] Sun Microsystems. Libmcr 0.9 beta: A reference correctly-rounded library of basic double-precision transcendental elementary functions. http://www.sun.com/download/products.xml?id=41797765, 2004.

[21] A. Ziv. Fast Evaluation of Elementary Mathematical Functions with Correctly Rounded last Bit. *ACM Transactions on Mathematical Software*, 17(3):410–423, 1991.

[22] A. Ziv, M. Olshansky, E. Henis, A. Reitman. MathLib, version 12.20.2001. ftp://www-126.ibm.com/pub/mathlib/.


**Appendix.** We study here the expected maximum distance between two bad cases of a function $f$ under the random model assumption of §1.

**Lemma 1** *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:*

$$E[M] \leq n \cdot 2^{p-n}.$$

(This is not the best possible bound, but it is sufficient here.)

**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. Let $k = \lambda \log 2 \cdot 2^{p-1}$. Using the fact that $\log(1-x) \leq -x$ for any $x \in [0, 1[$, we can bound the probability above by:

$$2^{n-1} \cdot e^{-\lambda \log 2} = 2^{n-1} \cdot 2^{-\lambda},$$

from which we obtain that:

$$E[M] \leq 1 \cdot (k2^{-n}) + 2^{n-1-\lambda} \cdot \frac{1}{2}.$$

Taking $\lambda = 2n - p - 1$ ends the proof. ∎