

# Some notes for a proposal for elementary function implementation in floating-point arithmetic

Guillaume Hanrot    Vincent Lefèvre    Jean-Michel Muller  
Nathalie Revol    Paul Zimmermann

June 8, 2001

## Abstract

In the present document, we present some aspects of what a standard for the implementation of the elementary functions could be. We do not claim that we always suggest the right choices, or that we have thought about all relevant issues. The mere goal of this paper is to raise questions and to launch the discussion towards a standard.

## 1 Introductory discussion

We take the opportunity of the current discussion on the revision of the IEEE-754 Standard for Floating-Point Arithmetic to discuss the possibility of standardizing (some of) the elementary functions. The IEEE-754 Standard [1] does not deal with these functions. This is due to several reasons, the most serious one being the Table Maker's Dilemma (TMD for the sake of brevity), which is the problem of providing correctly rounded transcendentals. To quote Kahan [4],

*Committing an approximation function to hardware is unlikely to be an unqualified success. Delivering any result other than the mathematically correct result rounded to the destination format is open to criticism. Delivering a result more slowly than a software implementation can raise questions of why the function is in hardware. Dedicating significant amounts of chip area to support transcendental functions is usually better spent improving the speed of vector multiplication. In short, the silicon implementation should be fast, accurate, and cost nothing.*

In the same document, Kahan says that in the absence of correctly rounded results, the most important property to maintain is monotonicity.

An unfortunate consequence of that lack of standardization is that extremely poor libraries are still in use (see [10] or [9] for some examples). Nevertheless, the quality of most elementary function libraries has greatly improved during the last decade.

Our suggestions are based on the reading of some of the works of Kahan [3, 4, 5, 6], our own experience on studying elementary function implementation, and recent progress of some of the authors of this paper concerning the Table Maker's Dilemma [8, 7].

We suggest three levels of quality, the lowest one (level 0) being regarded as the minimum acceptable level for a library, and the highest one representing the best quality that can be reached (on the whole, correct rounding in all the input range).

Several properties would be desirable. Among them:

1. correct rounding (for the 4 rounding modes)
2. preservation of the output range: e.g., one would like a sine to be between  $-1$  and  $1$ , an arctangent to be between  $-\pi/2$  and  $+\pi/2$ , etc. Not satisfying this preservation could have nonnegligible consequences. If the output range of function  $f$  is  $[a, b]$ , a programmer might successively compute:  $y = f(x)$ ,  $z = g(y)$ , with  $g$  defined in a subset of  $[a, b]$  only. Hence, if the software for  $f$  returns a value of  $y$  out of that interval (even slightly), important errors might occur.
3. preservation of monotonicity,
4. preservation of symmetries (e.g.  $\sin(-x) = -\sin(x)$ ),
5. to our opinion, when a directed rounding mode is selected, even if correct rounding cannot be satisfied, the direction of rounding **MUST** be preserved.
6. correct handling of exceptions and denormal numbers.
7. each time a function cannot be uniquely defined using continuity, a NaN should be returned: examples are  $1^{\pm\infty}$  or  $\sin(\infty)$ . A possible exception is  $0^0$ : it is not uniquely defined, but the convention " $0^0 = 1$ " has the advantage of preserving some mathematical formulas, hence many authors suggest to keep it.

It is worth being noticed that these desirable properties are sometimes not compatible. For instance, in single precision and round-to-nearest mode, a correctly rounded routine for arctangent would return values larger than  $\pi/2$  for input values large enough. The single precision number which is closest to  $\arctan(2^{30})$  is

$$\frac{13176795}{8388608} = 1.57079637050628662109375 > \frac{\pi}{2}.$$

Therefore, if the arctangent function is implemented in round-to-nearest mode, we get an arctangent larger<sup>1</sup> than  $\pi/2$ . A consequence of this is that in such a system,

$$\tan\left(\arctan\left(2^{30}\right)\right) = -2.2877\dots \times 10^7.$$

A more obvious example is the fact that with rounding modes towards  $\pm\infty$ , correct rounding and preservation of symmetries are not compatible.

## 2 What could be included in a proposal

### 2.1 Functions being considered here

$\ln$ ,  $\log_2$ ,  $\log_{10}$ ,  $\exp$ ,  $2^x$ ,  $10^x$ ,  $\sin$ ,  $\cos$ ,  $\tan$ ,  $\operatorname{asin}$ ,  $\operatorname{acos}$ ,  $\operatorname{atan}$ ,  $\operatorname{atan}(x, y)$ ,  $\operatorname{power}(x, y) = x^y$ ,  $\sinh$ ,  $\cosh$ ,  $\tanh$ ,  $\operatorname{asinh}$ ,  $\operatorname{acosh}$ ,  $\operatorname{atanh}$ .

But most of what is said here could apply to special functions (gamma, erf, erfc, Bessel functions, etc.) and some algebraic functions such as cube root.

### 2.2 Exceptions

#### 2.2.1 Values that cannot be defined using continuity

Sometimes, different choices are legitimate. What is important is consistency. Consider three examples:

- the case of  $0^0$  is important. On the one hand, as said above, there is no way of defining  $0^0$  using continuity, but on the other hand, many important properties remain satisfied if we chose  $0^0 = 1$  (which is frequently adopted, as a definition, by mathematicians). Kahan suggests to choose  $0^0 = 1$ . A consequence of that (also mentioned by Kahan) is that it implies that  $\operatorname{power}(\operatorname{NaN}, 0) = \operatorname{NaN}^0 = 1$  whereas  $0^{\operatorname{NaN}} = \operatorname{NaN}$ ,

---

<sup>1</sup>But *equal* to the machine representation of  $\pi/2$ .

since  $x^0$  is 1 for ANY  $x$ , whereas  $0^y$  is 1 if  $y = 0$  and 0 otherwise. If we happen to choose that  $0^0$  is NaN (which is perfectly legitimate), then  $power(NaN, 0)$  is NaN.

- another example is  $\log(-0)$ . On the one hand, as suggested by Goldberg [2],  $-0$  may be thought as a small negative number that has underflowed to zero. This would favor the choice  $\log(-0) = NaN$ . On the other hand, such a choice would first imply that we can have  $x = y$  and  $\log(x) \neq \log(y)$  (with  $x = +0$  and  $y = -0$ , since the IEEE Standard requires that the test  $-0 == +0$  should return *true*). Moreover, in a similar<sup>2</sup> case, the IEEE standard seems to favor the second choice, since  $\sqrt{-0}$  is defined as  $-0$ . For these reasons of consistency (and only for these reasons), we would prefer the choice  $\log(-0) = -\infty$ , but we of course recognize this choice has drawbacks.
- $Power(1, \pm\infty)$  is similar to  $0^0$ . One can build  $u_n \rightarrow 1$  and  $v_n \rightarrow +\infty$  such that  $u_n^{v_n}$  goes to anything you desire (or nothing at all). Kahan suggests  $Power(1, \pm\infty) = NaN$ , which implies (for reasons of consistency)  $Power(1, NaN) = NaN$ .

### 2.2.2 NaNs (as input or output values)

All functions having at least one NaN as input value should return a NaN, with the possible exception  $power(NaN, 0) = NaN^0 = 1$  (if  $0^0$  is defined as 1). We tend to prefer  $power(NaN, 0) = NaN$ .

$\sin(\pm\infty)$ ,  $\cos(\pm\infty)$ ,  $\tan(\pm\infty)$  are NaNs.  $\ln$ ,  $\log_2$ ,  $\log_{10}$  of a negative number, or  $power(\text{negative}, \text{noninteger})$  are NaNs.  $\text{asin}$ ,  $\text{acos}$ ,  $\text{atanh}$  of a number outside  $[-1, +1]$  are NaNs.  $Power(1, NaN)$  is NaN if  $power(1, \infty)$  is NaN.

It is **not allowed** to return a NaN when the exact result is mathematically defined (e.g., sine of a huge number).

### 2.2.3 infinities

- infinities as result of over/underflow: proceed as for the arithmetic operations (each time the result is mathematically defined but larger or smaller than the largest representable number)
- “exact infinities”:  $\log(+0)$ ,  $\log_2(+0)$ ,  $\log_{10}(+0)$  are  $-\infty$ , with flag “zero divide” being raised. We suggest the same for  $-0$ , although it

---

<sup>2</sup>Unless we consider complex arithmetic.

might seem counterintuitive (but keeping  $x = y \Rightarrow \log(x) = \log(y)$  is important).

It is worth being noticed that tangents never return infinities (no Single Precision, Double Precision, or Double extended FP number is close enough to a multiple of  $\pi/2$ ).

### 2.3 Roundings

We suggest to allow three levels of quality. Which level is actually provided should appear clearly in the documentation of the elementary function library (or hardware). It is of course allowable to provide all levels, the programmer being then able to select a tradeoff between quality and speed. In such a case, the default should be the highest available level.

**Level 0** in **round-to-nearest** mode, the returned result should always be one of the two floating-point numbers that surround the exact result (if the exact result is a floating-point value – which is rare with the transcendental functions: see the section devoted to the “inexact” flag), the system should return that value). In the **round towards  $-\infty$  mode**, the returned result should always be less than or equal to the exact result. No error greater than 1.5 ulps is allowed. In the **round towards  $+\infty$  mode**, the returned result should always be larger than or equal to the exact result. No error more than 1.5 ulps is allowed. The **round towards zero mode** behaves as the round towards  $+\infty$  mode for positive values and the round towards  $-\infty$  mode for negative values. In all cases, where the “exact” function is monotonous, **the implementation must be monotonous too**. In **round-to-nearest** mode, the symmetries of the function around 0 (properties of the kind  $f(-x) = \pm f(x)$ ) should be preserved<sup>3</sup>.

**Level 1** there is a domain (usually around zero) where the implemented function is correctly rounded. Outside this domain, the implementation should satisfy the criteria of Level 0. We suggest the domain should at least contain  $[-2\pi, +2\pi]$  for sin, cos and tan; and  $[-1, 1]$  for exp, cosh, sinh,  $2^x$  and  $10^x$  (other functions: to be discussed. Compromise involving facility of implementation and usefulness of requirement).

---

<sup>3</sup>In practice this requirement is not a problem: function implementers will use these symmetries for simplifying their programs.



Table 2: Worst cases for the natural (radix  $e$ ) logarithm in the full double precision range.

Interval	worst case (binary)
[ $2^{-1074}, 1$ )	$\log(1.1001010001110110111000110000010011001101011111000111 \times 2^{-384})$ $= -100001001.10110110000011001010111101000111101100110101 \quad 1 \quad 0^{60}1010\dots$
	$\log(1.1110101001110001110110000101110011101110000000100000 \times 2^{-509})$ $= -101100000.00101001011010100110011010110100001011111111 \quad 1 \quad 1^{60}0000\dots$
	$\log(1.0010011011101001110001001101001100100111100101100000 \times 2^{-232})$ $= -10100000.101010110010110000100101111001101000010000100 \quad 0 \quad 0^{60}1001\dots$
( $1, 2^{1024}$ ]	$\log(1.0110001010101000100001100001001101100010100110110110 \times 2^{678})$ $= 111010110.01000111100111101011101001111100100101110001 \quad 0 \quad 0^{64}1110\dots$

## 2.4 “Inexact result” flag

The following is extracted from Muller’s book *Elementary functions, algorithms and implementation* (Birkhauser, 1997).

It is difficult to know when functions such as  $x^y$  give an exact result. However, using a theorem due to Lindemann, one can show that the sine, cosine, tangent, exponential, or arctangent of a nonzero finite machine number, or the logarithm of a finite machine number different from 1 is not a machine number, so that its computation is always inexact. With the most common functions, the only “exact” operations are:

- for the radix- $e$  logarithm and exponential functions:
  1.  $\ln(\pm 0) = -\infty$  ;
  2.  $\ln(+\infty) = +\infty$  ;
  3.  $\ln(1) = +0$  ;
  4.  $e^0 = 1$  ;
  5.  $e^{-\infty} = +0$  ;
  6.  $e^{+\infty} = +\infty$  ;
- for the radix-2 logarithm and exponential functions (assuming a radix-2 representation):
  1.  $\log_2(\pm 0) = -\infty$  ;
  2.  $\log_2(+\infty) = +\infty$  ;

3.  $\log_2(1) = +0$  ;
  4. for any integer  $p$  such that  $2^p$  is exactly representable (i.e.,  $p$  is between the smallest possible exponent — unless denormal numbers are allowed — and the largest one),  $\log_2(2^p) = p$  ;
  5.  $2^0 = 1$  ;
  6.  $2^{-\infty} = +0$  ;
  7.  $2^{+\infty} = +\infty$  ;
  8. for any integer  $p$  between the smallest possible exponent minus the number of mantissa bits and the largest one,  $exp2(p) = 2^p$ .
- for the sine, cosine, tangent, and arctangent functions:
    1.  $\sin(0) = 0$  (with same sign as the input zero) ;
    2.  $\cos(0) = 1$  ;
    3.  $\tan(0) = 0$  (with same sign as the input zero) ;
    4.  $\arctan(0) = 0$  (with same sign as the input zero) ;
  - for the hyperbolic sine, cosine, tangent, and arctangent functions:
    1.  $\sinh(0) = 0$  (with same sign as the input zero) ;
    2.  $\sinh(-\infty) = -\infty$  ;
    3.  $\sinh(+\infty) = +\infty$  ;
    4.  $\cosh(0) = 1$  ;
    5.  $\cosh(-\infty) = +\infty$  ;
    6.  $\cosh(+\infty) = +\infty$  ;
    7.  $\tanh(0) = 0$  (with same sign as the input zero) ;
    8.  $\tanh(-\infty) = -1$  ;
    9.  $\tanh(+\infty) = 1$  ;
    10.  $\tanh^{-1}(0) = 0$  (with same sign as the input zero) ;
    11.  $\tanh^{-1}(-1) = -\infty$  ;
    12.  $\tanh^{-1}(1) = +\infty$ .

### 3 Now, it's up to you

We are eagerly looking for comments from the computer arithmetic and numerical analysis communities.

## References

- [1] American National Standards Institute and Institute of Electrical and Electronic Engineers. IEEE standard for binary floating-point arithmetic. *ANSI/IEEE Standard, Std 754-1985*, New York, 1985.
- [2] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–47, March 1991.
- [3] W. Kahan. Minimizing  $q^*m-n$ , text accessible electronically at <http://http.cs.berkeley.edu/~wkahan/>. At the beginning of the file "nearpi.c", 1983.
- [4] W. Kahan. Computer system support for scientific and engineering computation. Lecture notes, available at <http://www.validlab.com/fp-1988/lectures/>, 1988.
- [5] W. Kahan. Lecture notes on the status of IEEE-754. Postscript file accessible electronically through the Internet at the address <http://http.cs.berkeley.edu/~wkahan/ieee754status/ieee754.ps>, 1996.
- [6] W. Kahan. What can you learn about floating-point arithmetic in one hour? Postscript version accessible electronically at <http://http.cs.berkeley.edu/~wkahan/ieee754status/>, 1996.
- [7] 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*, Vail, USA, 2001. IEEE Computer Society Press, Los Alamitos, CA.
- [8] V. Lefèvre, J.-M. Muller, and A. Tisserand. Toward correctly rounded transcendentals. *IEEE Transactions on Computers*, 47(11), November 1998.
- [9] J.M. Muller. *Elementary Functions, Algorithms and Implementation*. Birkhauser, Boston, 1997.
- [10] K. C. Ng. Argument reduction for huge arguments: Good to the last bit (can be obtained by sending an e-mail to the author: [kwok.ng@eng.sun.com](mailto:kwok.ng@eng.sun.com)). Technical report, SunPro, 1992.