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

Jean-Michel Muller and whoever is willing to collaborate

March 29, 2001

1 Introductory discussion

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 motonicity,

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 $\sin(\infty)$. An exception is 0^0: it is not uniquely defined (for any positive number y, one can build a sequence a_n and a sequence b_n, both going to zero, such

that $a_n^{b_n}$ goes to y), but the convention "$0^0 = 1$" has the advantage of preserving several mathematical formulas, hence it is better to keep it.

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

2 Draft proposal

2.1 Functions being considered here

ln, log2, log10, exp, 2^x, 10^x, sin, cos, tan, asin, acos, atan, power(x,y)= x^y, sinh, cosh, tanh, asinh, acosh, atanh

2.2 Exceptions

2.2.1 NaNs (as input or output values)

all functions having at least one NaN as input value should return a Nan, with the exception power(NaN,0)=NaN$^0 = 1$ (note that 0^{NaN} is NaN. The reason for these choices, suggested by Kahan, is that x^0 is 1 for ANY x, whereas 0^y is 1 if $y = 0$ and 0 otherwise)

$\sin(\pm\infty)$, $\cos(\pm\infty)$, $\tan(\pm\infty)$ are NaNs. ln, log2, log10 of a negative number, or power(negative,noninteger) are NaNs. asin, acos, atanh of a number outside $[-1, +1]$ are NaNs. power(1,±∞) is NaN (was suggested by Kahan).

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

2.2.2 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 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 closed 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 more 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.

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).

Level 2 correct rounding in the whole domain where the function is mathematically defined. We might suggest a level "2bis" (which is not to be considered higher or lower in quality) for which correct rounding is provided unless this prevents preservation of output range (i.e., output range has priority).

Some comments:

- We consider Level 0 as the minimum acceptable level.
- Level 2 is attainable at reasonable cost for at least some functions, in single and double precisions. Tables 1 and 2 give the hardest-to-round cases for double precision exponentials and logarithms. These tables are extracted from a paper by Lefèvre and Muller to appear in the proceedings of the 15th IEEE Symposium on Computer Arithmetic. The hardest-to-round cases for the full-double precision range are also already known for 2^x and $\log_2(x)$.
- finding the hardest-to-round cases for the trigonometric functions in double and extended precisions might be difficult. Moreover, even if the hardest-to-round cases are known, functions implementing sines, cosines and tangents with correct rounding might be quite slow. This is the reason for level 1.

Table 1: *Worst cases for the exponential function in the full double-precision range. Exponentials of numbers less than $\ln(2^{-1074})$ are underflows (a routine should return 0 or the smallest non zero positive representable number, depending on the rounding mode). Exponentials of numbers larger than $\ln(2^{1024})$ are overflows.*

Interval	worst case (binary)
$[\ln(2^{-1074}), -2^{-30}]$	$\exp(-1.1110110100110001100011101111101101100010011111101010 \times 2^{-27})$ $= 1.1111111111111111111111111100001001011001110011100 0100 \quad 1 \quad 1^{59}0001... \times 2^{-1}$
$[-2^{-30}, 0)$	$\exp(-1.0000000000000000000000000000000000000000000000000001 \times 2^{-51})$ $= 1.1111111111111111111111111111111111111111111111111100 \quad 0 \quad 0^{100}1010... \times 2^{-1}$
$(0, +2^{-30}]$	$\exp(1.1111111111111111111111111111111111111111111111111111 \times 2^{-53})$ $= 1.0000000000000000000000000000000000000000000000000000 \quad 1 \quad 1^{104}0101...$
$[2^{-30}, \ln(2^{1024})]$	$\exp(1.0111111111111110011111111111111011100000000000100100 \times 2^{-32})$ $= 1.0000000000000000000000000000000010111111111111101000 \quad 0 \quad 0^{57}1101...$
	$\exp(1.1000000000000001011111111111111011011111111111011100 \times 2^{-32})$ $= 1.0000000000000000000000000000000011000000000000010111 \quad 1 \quad 1^{57}0010...$
	$\exp(1.1001111010011100101110111111110101100000010000001011 \times 2^{-31})$ $= 1.0000000000000000000000000000000110011110100111001011 1 \quad 1 \quad 0^{57}1010...$
	$\exp(110.00001111010100101111001101111010111011001111110100)$ $= 110101100.01010000101101000000100111001000101011101110 \quad 0 \quad 0^{57}1000...$

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

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

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. $\ln(0) = -\infty$;
  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), $\log_2(1.0 \times 2^p) = p$;
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, $2^p = 1.0 \times 2^p$.

- 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. $\tanh^{-1}(1) = +\infty$.