# Accuracy of Mathematical Functions in Single, Double, Double Extended, and Quadruple Precision

Brian Gladman, Vincenzo Innocente, John Mather, Paul Zimmermann

# Accuracy of Mathematical Functions in Single, Double, Double Extended, and Quadruple Precision

Brian Gladman, Vincenzo Innocente[*], John Mather[†], Paul Zimmermann[‡]

February 2025

*Computer users, most of whom assume they are working with reliable routines, unwittingly accept results from functions where the accuracies vary significantly from one mathematical library to another, from one library function to another, and even over different argument intervals of the same function. [...] Users are not likely to demand an improved situation because most of them, having neither the time nor the inclination to test manufacturer-supplied software, do not know the problem exists. This paper contains the results of such tests of elementary functions from several computer companies. The data [...] demonstrate that the industry does not satisfy the needs of those who require accurate and efficient mathematical software.*

These lines, written nearly four decades ago in 1984 by Black, Burton and Miller [8], are unfortunately still very true today.

The IEEE 754 standard, even in its latest 2019 revision [1], does not *require* correctly rounded mathematical functions, it only *recommends* them. In turn, current mathematical libraries do not provide correct rounding, which is the best possible result. Thus, users might get different results with different libraries, or different versions of the same library. This can have dramatic consequences: for example missed collisions in the Large Hadron Collider [7] or reproducibility issues in neuroimaging [18].

This document compares the accuracy of several mathematical libraries for the evaluation of mathematical functions, in single, double and quadruple precision (respectively `binary32`, `binary64`, and `binary128` in the IEEE 754 standard), and also in the extended double format. For single precision, an exhaustive search is possible for univariate functions, thus the given values are upper bounds. For larger precisions or bivariate functions, since an exhaustive search is not possible with academic resources, we use a black-box algorithm that tries to locate the values with the largest error; the given values are only lower bounds, but comparing them can give an idea of the relative accuracy of different libraries. An interesting fact is that, for several functions, different libraries yield the same largest known error, for the exact same input value, which probably means they use the same code base. While GCC clearly says that the accuracy of mathematical functions is unknown [17], some libraries document the largest known errors[1] [11, 14, 28]. Also, in [22], the

[*]CERN

[†]Side Effects Software Inc.

[‡]Université de Lorraine, CNRS, Inria, LORIA

[1]For the GNU libc, the values given in the "Known Maximum Errors in Math Functions" section are however not always accurate, since larger values can be found in GNU libc Bugzilla (and in this document).

authors have proven rigorous upper bounds for some double-precision functions (exp, log, sin, tan) of the Intel Math Library, in some domains (however the code might have changed since then).

Today, at least for single precision and most double precision functions, it is known how to get correct rounding (for all rounding modes, not only for rounding to nearest) at very low cost, and reference implementations exist that outperform current libraries (see references in [9]).

# 1 Introduction

In this document we compare the accuracy of the following mathematical libraries: GNU libc 2.41 [19], the Intel Math Library shipped with the Intel oneAPI DPC++ Compiler 2025.0.0 (IML) [20], AMD LibM 5.0 [3], Newlib 4.5.0 [31], OpenLibm 0.8.5 [29], Musl 1.2.5 [26], the Apple Math Library 15.1.1 available under Darwin 24.1.0 [4], the LLVM libc 19.1.7 [23], the Microsoft library from Visual Studio 2022 (MSVC) [24], the mathematical library from FreeBSD 14.2 [16], libamath from the Arm Performance Libraries 24.04 [5], the CUDA mathematical library 12.8 [27], and the ROCm mathematical library 6.2.4 [12]. We only consider rounding to nearest-even; with other rounding modes one did get disastrous results with some libraries in the past [21], as reported already in 2002 by Vincent Lefèvre.[2] We do not compare to the x87 instructions `fsin` and others, which are known to have bad accuracy [13].

For each function, assuming $y$ is the value returned by the library, and $z$ is the exact result (as with infinite precision), we denote by $e$ the absolute difference between $y$ and $z$ in terms of units-in-last-place of $z$ (i.e., when $z$ is in the subnormal range, in terms of multiples of the smallest positive subnormal). The value $z$ is approximated with the GNU MPFR library [15], using a larger precision. Our definition of ulp (unit-in-last-place) is the following: for $2^{e-1} \leq |x| < 2^e$, and precision $p$, we define $\mathrm{ulp}(x) = 2^{e-p}$. i.e., the distance between two consecutive $p$-bit floating-point numbers in the binade $[2^{e-1}, 2^e)$, see [25]. When one of $y$ and $z$ is NaN or $\pm\infty$, we use the following rules:

- if both $y$ and $z$ are NaN, we consider the error is 0;
- if only one of $y$ and $z$ is NaN, we consider the error is infinite;
- if both $y$ and $z$ are $\pm\infty$, we consider the error is 0 if they have the same sign, and infinite otherwise;
- if the library result is $\infty$, and the MPFR result is finite, we assume the library result is the next floating-point number after the largest finite number in the format (assuming unbounded exponent);
- if the library result is finite, and the MPFR result is $\infty$, we recompute the MPFR result using unbounded exponent, and compute the difference with this value.

The results for GNU libc, AMD LibM, Newlib, OpenLibm, and Musl were obtained on an Intel Core i5-4590 or i7-8700 with GCC 14.2.0 under Debian. Newlib was configured with default flags (in particular, without use of hardware FMA), and with the default configuration.[3] Those for LLVM

[2] `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=153022`

[3] For binary32, by default, the old SunPro functions are used; with `OBSOLETE_MATH_DEFAULT=0`, Newlib will use instead a new set of mathematical functions provided by Arm, that use binary64 for intermediate computations.

libc were obtained on an Intel(R) Xeon(R) Silver 4214; an AMD EPYC2 7352 was used for the Intel Math Library (IML in short), with the Intel compiler version 2025.0.0, using `-fp-model=strict`, including `mathimf.h` instead of `math.h` (see [10] for more details about the Intel compiler). Those for the Apple math library were obtained on a Mac M1 (arm64) with clang 15.0.0. The results with the Microsoft library were obtained using the Universal C Runtime Library (UCRT) distributed with Windows SDK version 10.0.22621 for Windows 11, and the code was run on a Dell Precision 5540 with an Intel Core i9 9880H running Windows 11 and on a Dell PowerEdge R7525 with two AMD EPYC2 7352 running Windows Server 2019. The FreeBSD results were obtained with a libvirt image running on a Intel Xeon Silver 4214. The results for libamath from the Arm Performance Libraries were obtained on an AWS Graviton3 c7g.metal instance running Ubuntu 22.04. The CUDA library was tested on a NVidia Grace-Hopper system running CUDA 12.8. The tests were also run on a GTX1060 GPU, hosted on an AMD Ryzen 7 1800X, obtaining identical results. The ROCm library was tested on an AMD Radeon Instinct MI50 running ROCm 6.2.4 hosted on a Intel Xeon Silver 4114. Tests were also run on a "Radeon Pro WX 9100", hosted on a AMD Ryzen 9 5900X, obtaining identical results.

In all tables, values of $e$ are given with 3 decimal digits, rounded up; thus for example $e = 2.17$ for a univariate single-precision function means that the relative error is bounded by 2.17 ulps for all `binary32` inputs, and in all other cases (larger formats or bivariate functions) it means the largest *known* error is bounded by 2.17 ulps, with at least one case giving an error of more than 2.16 ulps. Note that correct rounding implies an entry 0.5, but the converse is false: when the "round-to-even" rule applies, if the library returns the "odd" value, the error will still be 0.5 ulp.

It should be noted that one might get different results for a given library on different hardware, for at least two reasons. Firstly some libraries have a runtime dispatcher which invokes different source code for different cpus, for example using the fused multiply-add, or extensions SSE, AVX, AVX2 or AVX512. This is the case of the GNU libc and of the Intel Math library. Secondly the very same binary might produce different results on different hardware due to the use of some assembly instructions that are implemented differently. This is for example the case with the `rsqrt` and `rcp` instructions that differ on Intel and AMD hardware, see §3.2.

# 2 Single Precision

## 2.1 Univariate Functions

The IEEE 754 single-precision format (`binary32`) has $2^{32} - 2^{24}$ values, not counting +Inf, -Inf, and NaN. For a function with a single input, it is thus possible to check all values by exhaustive search.

Table 1 summarizes the largest known ulp error for each function and each library. For univariate functions, the corresponding input can easily be found by exhaustive search, while Table 2 gives the corresponding inputs for bivariate functions.

In all tables, empty cells mean the corresponding function is not available.

We see that for all libraries, the sqrt function is correctly rounded for all binary32 inputs, as required by IEEE 754.[4] The following functions are correctly rounded: `rsqrtf` in the Intel Math Library, `cbrtf` in OpenLibm, Musl, the Apple library and FreeBSD, `coshf` in MSVC, `sinhf` and `tanhf` in AMD LibM (which uses the CORE-MATH implementation [32]), 21 functions from

[4] As noticed by Hugues de Lassus, correct rounding implies a maximal error of 0.5 ulp, but the converse is not necessarily true. However, we also checked the results agree with GNU MPFR.

| library | GNU libc | IML | AMD | Newlib | OpenLibm | Musl | Apple | LLVM | MSVC | FreeBSD | ArmPL | CUDA | ROCm |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | 2.41 | 2025.0.0 | 5.0 | 4.5.0 | 0.8.5 | 1.2.5 | 15.1.1 | 19.1.7 | 2022 | 14.2 | 24.04 | 12.8 | 6.2.4 |
| acos | **0.500** | 0.528 | 0.897 | 0.899 | 0.918 | 0.918 | 0.634 | **0.500** | 0.669 | 0.895 | 1.32 | 1.34 | 1.47 |
| acosh | **0.500** | 0.501 | 0.504 | 2.01 | 2.01 | 2.01 | 0.502 | **0.500** | 2.89 | 2.01 | 2.79 | 2.18 | 0.564 |
| asin | **0.500** | 0.528 | 0.781 | 0.926 | 0.743 | 0.743 | 0.634 | **0.500** | 0.861 | 0.672 | 2.41 | 1.36 | 2.54 |
| asinh | **0.500** | 0.527 | 0.518 | 1.78 | 1.78 | 1.78 | 0.515 | **0.500** | 1.99 | 1.78 | 3.57 | 1.78 | 0.573 |
| atan | **0.500** | 0.541 | 0.501 | 0.853 | 0.853 | 0.853 | 0.722 | **0.500** | 0.501 | 0.853 | 2.88 | 1.21 | 2.10 |
| atanh | **0.500** | 0.507 | 0.547 | 1.73 | 1.73 | 1.73 | 0.511 | **0.500** | 2.35 | 1.73 | 3.09 | 3.16 | 0.574 |
| cbrt | **0.500** | 0.520 | 0.548 | 3.56 | **0.500** | **0.500** | **0.500** | **0.500** | 1.83 | **0.500** | 1.53 | 1.17 | 1.14 |
| cos | 0.561 | 0.548 | 0.729 | 2.91 | 0.501 | 0.501 | 0.846 | **0.500** | 0.530 | 0.501 | 0.561 | 1.52 | 1.61 |
| cosh | **0.500** | 0.506 | 1.03 | 2.51 | 1.36 | 1.03 | 0.579 | **0.500** | **0.500** | 1.36 | 1.89 | 2.34 | 0.567 |
| erf | **0.500** | 0.780 | 0.531 | 0.968 | 0.943 | 0.968 | 0.501 | **0.500** | 3.99 | 0.890 | 1.93 | 1.04 | 1.51 |
| erfc | **0.500** | 0.934 | | 63.9 | 3.17 | 3.13 | 0.750 | | 6.66 | 3.18 | 1.64 | 4.49 | 3.33 |
| exp | 0.502 | 0.506 | 0.501 | 0.911 | 0.911 | 0.502 | 0.514 | **0.500** | 0.501 | 0.911 | 0.502 | 1.94 | 1.00 |
| exp10 | 0.502 | 0.507 | 0.501 | 1.00 | | 3.88 | 0.514 | **0.500** | | | | 2.07 | 1.00 |
| exp2 | 0.502 | 0.519 | 0.501 | 1.00 | 0.501 | 0.502 | 0.514 | **0.500** | 2.14 | 0.501 | 0.502 | 2.39 | 0.871 |
| expm1 | **0.500** | 0.544 | Inf | 0.813 | 0.813 | 0.813 | 0.687 | **0.500** | 3.02 | 0.813 | 1.51 | 1.45 | 1.45 |
| j0 | 9.00 | **0.678** | | 6.18e6 | 3.66e6 | 3.66e6 | | | | 3.66e6 | | 3.78e10 | 7.60e7 |
| j1 | 9.00 | **1.69** | | 1.68e7 | 2.25e6 | 2.25e6 | | | | 2.25e6 | | 7.48e9 | 7.53e7 |
| lgamma | **0.500** | 0.510 | | 7.50e6 | 7.50e6 | 7.50e6 | 0.501 | | 2.92e5 | 7.50e6 | | 1.35e7 | 7.50e6 |
| log | 0.818 | 0.519 | 0.577 | 0.888 | 0.888 | 0.818 | 0.511 | **0.500** | 0.562 | 0.888 | 0.818 | 0.865 | 1.89 |
| log10 | **0.500** | 0.516 | 1.40 | 2.10 | 0.832 | 0.832 | 0.502 | **0.500** | 0.626 | 0.832 | 0.82 | 2.09 | 1.71 |
| log1p | **0.500** | 0.525 | 0.501 | 1.30 | 0.839 | 0.835 | 0.513 | **0.500** | 1.44 | 0.839 | 2.02 | 0.887 | 0.579 |
| log2 | 0.752 | 0.508 | 0.766 | 1.65 | 0.865 | 0.752 | 0.502 | **0.500** | 2.04 | 0.865 | 0.752 | 0.919 | 1.00 |
| sin | 0.561 | 0.546 | 0.530 | 1.37 | 0.501 | 0.501 | 0.846 | **0.500** | 0.530 | 0.501 | 0.561 | 1.50 | 1.61 |
| sinh | 0.501 | 0.538 | **0.500** | 2.51 | 1.83 | 1.83 | 0.579 | **0.500** | 0.501 | 1.83 | 2.26 | 2.94 | 0.922 |
| sqrt | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | | **0.500** | **0.500** |
| tan | **0.500** | 0.520 | 0.509 | 3.48 | 0.800 | 0.800 | 0.746 | **0.500** | 0.502 | 0.800 | 3.30 | 3.10 | 2.33 |
| tanh | **0.500** | 0.514 | **0.500** | 2.19 | 2.19 | 2.19 | 0.817 | **0.500** | 1.27 | 2.19 | 2.59 | 1.82 | 1.41 |
| tgamma | **0.500** | 0.510 | | 239 | 0.501 | 0.501 | 0.501 | | 3.58e5 | 0.501 | | 4.34 | 1.68e7 |
| y0 | 8.98 | **3.40** | | 4.84e6 | 4.84e6 | 4.84e6 | | | | 4.84e6 | | 2.36e10 | 7.53e7 |
| y1 | 9.00 | **2.07** | | 6.18e6 | 4.17e6 | 3.66e6 | | | | 4.17e6 | | 4.96e10 | 9.35e7 |
| acospi | 1.94 | **0.504** | | | | | | | | | | | |
| asinpi | 1.86 | **0.506** | | | | | | | | | | | |
| atanpi | 1.93 | **0.545** | | | | | | | | | | | |
| cospi | 1.98 | **0.501** | 1.95 | | | | | **0.500** | | **0.501** | 2.65 | 0.966 | 0.966 |
| exp10m1 | **0.500** | | | | | | | | | | | | |
| exp2m1 | **0.500** | | | | | | | **0.500** | | | | | |
| log10p1 | **0.833** | | | | | | | | | | | | |
| log2p1 | **0.500** | | | | | | | | | | | | |
| sinpi | 1.98 | **0.501** | 1.96 | | | | | **0.500** | | 0.755 | 2.49 | 0.967 | 0.967 |
| tanpi | 2.93 | 1.00 | 2.22 | | | | | | | **0.800** | | | |
| rsqrt | | **0.500** | | | | | | | | | | 1.52 | 0.864 |
| atan2 | **0.500** | 0.550 | 0.584 | 1.52 | 1.55 | 1.55 | 0.722 | **0.500** | 0.584 | 1.55 | 2.93 | 2.18 | 2.01 |
| atan2pi | 1.61 | **0.841** | | | | | | | | | | | |
| compound | | **0.501** | | | | | | | | | | | |
| hypot | 0.501 | Inf | 0.501 | 1.21 | 1.21 | 0.927 | 0.501 | **0.500** | 0.501 | 1.21 | | 1.03 | 1.57 |
| pow | 0.817 | 0.515 | 1.56 | 169. | 0.970 | 0.817 | 0.514 | **0.500** | 0.568 | 0.970 | 0.817 | 2.60 | 1.40 |

Table 1: Single precision: largest value of $e$ (for univariate functions), and largest *known* value of $e$ (for bivariate functions). Empty cells mean the corresponding function is not available.

| | GNU libc 2.41 | | IML 2025.0.0 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | 0x1.960176p-63 | 0x1.f88dp+80 | -0x1.58a7ecp-118 | 0x1.58a7bep-123 |
| atan2pi | -0x1.baca2ap-96 | -0x1.a6c0c4p-82 | -0x1.461adep+23 | -0x1.74fbc8p+22 |
| compound | | | 0x1.46674cp+50 | 0x1.d5738ap-30 |
| hypot | -0x1.003222p-20 | -0x1.6a2d58p-32 | -0x1.527182p+127 | -0x1.803006p+127 |
| pow | 0x1.025736p+0 | 0x1.309f94p+13 | 0x1.fe7782p-1 | -0x1.c361cap+14 |

| | AMD LibM 5.0 | | Newlib 4.5.0 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | 0x1.fffe24p+59 | 0x1.000adcp+73 | -0x1.f9cf48p+49 | 0x1.f60598p+51 |
| hypot | -0x1.0554acp+44 | -0x1.6dc9e6p+32 | -0x1.6b05c4p-127 | 0x1.6b3146p-126 |
| pow | 0x1.10fff4p+0 | 0x1.58fd76p+10 | 0x1.d6411cp-102 | 0x1.793482p+0 |

| | OpenLibm 0.8.5 | | Musl 1.2.5 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | 0x1.a10104p+123 | 0x1.99f182p+125 | 0x1.a10104p+123 | 0x1.99f182p+125 |
| hypot | -0x1.6b05c4p-127 | 0x1.6b3146p-126 | 0x1.26b188p-127 | -0x1.a4f2fp-128 |
| pow | 0x1.343e4ep+0 | 0x1.af3c4p+8 | 1.025736p+0 | 1.309f94p+13 |

| | Apple 15.1.1 | | LLVM 19.1.7 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | -0x1.ce62cep-116 | 0x1.cbf9bp-113 | 0x1.960176p-63 | 0x1.f88dp+80 |
| hypot | -0x1.003222p-20 | -0x1.6a2d58p-32 | 0x1.65e2cp+124 | 0x1.ac1p+112 |
| pow | 0x1.95e42cp-27 | 0x1.369972p-14 | -0x1.8p-49 | 0x1.8p+1 |

| | MSVC 2022 | | FreeBSD 14.2 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | 0x1.fffe24p+59 | 0x1.000adcp+73 | 0x1.a10104p+123 | 0x1.99f182p+125 |
| hypot | -0x1.003222p-20 | -0x1.6a2d58p-32 | -0x1.6b05c4p-127 | 0x1.6b3146p-126 |
| pow | 0x1.107fe4p+0 | 0x1.631e6cp+10 | 0x1.343e4ep+0 | 0x1.af3c4p+8 |

| | ArmPL 24.04 | |
|---|---|---|
| | $x$ | $y$ |
| atan2 | -0x1.9be82ap-101 | 0x1.92cfb2p-101 |
| pow | 0x1.025736p+0 | 0x1.309f94p+13 |

| | CUDA 12.8 | | ROCm 6.2.4 | |
|---|---|---|---|---|
| | $x$ | $y$ | $x$ | $y$ |
| atan2 | 0x1.0e5beap+6 | 0x1.016188p+6 | 0x1.e9d044p-37 | 0x1.dfb95ep-35 |
| hypot | 0x1.007594p+1 | -0x1.003512p+1 | -0x1.ad2d0ap+111 | -0x1.7f456ap+118 |
| pow | 0x1.6794b6p+0 | 0x1.f9f1bap+7 | 0x1.25f64ep-5 | 0x1.a47bc2p+4 |

Table 2: Inputs for single precision bivariate functions giving the largest known error.

GNU libc (also using the CORE-MATH code), and all functions available in the LLVM library. However, with rounding towards zero, AMD's `tanhf` yields `0x1p+0` instead of `0x1.fffffep-1` for $x$ = `0x1.205968p+3` (this issue was already in AMD LibM 4.2).

We get large errors for `j0f`, `j1f`, `y0f`, `y1f` and `lgammaf` for all libraries where these functions are available, except for GNU libc, IML, and the Apple library.

Nevertheless, for almost all single precision functions, it is now possible to choose a library which provides presumably correct rounding (exceptions are the Bessel functions, `acospi`, `asinpi`, `atanpi`, `log10p1`, `tanpi`, `atan2pi` and `compound`).

## 2.2 Bivariate Functions

For single precision bivariate functions, it is not possible to perform an exhaustive search with academic resources, since there are up to $2^{64}$ possible pairs of inputs. For example, for the power function $x^y$, there are about $2^{61}$ input pairs $x, y > 0$ that do not yield underflow nor overflow. We thus used the algorithm described in §3.1 to obtain the values for bivariate functions in Table 1, which are *lower bounds* for the largest error.

**Notes about GNU libc.** GNU libc 2.41 implements new C23 functions for all formats: `acospi`, `asinpi`, `atanpi`, `cospi`, `sinpi`, `tanpi` and `atan2pi`.

**Notes about the Intel Math Library.** For $x$ = `-0x1.527182p+127` and $y$ = `-0x1.803006p+127`, the `hypotf` function yields NaN instead of +Inf. The `rsqrt` function, which is not yet standardized in C99, is called `invsqrt` in the Intel Math Library; in single precision, it is correctly rounded for all rounding modes.

**Notes about AMD LibM.** AMD LibM provides new functions: `cospi`, `sinpi`, `tanpi`. We noticed a regression: for $x$ = `-0x1.154244p+4`, AMD LibM 5.0 `expm1f` yields NaN instead of `-0x1.fffffep-1`.

**Notes about Newlib.** For negative integers, Newlib `tgammaf` returns an infinite value, whereas other libraries return NaN. We use the `lgammaf_r` function, since we were unable to compile the `lgammaf` function (`undefined reference to '_impure_ptr'`).

**Notes about the Apple Math Library.** The `erff`, `lgammaf` and `tgammaf` functions seem to call the corresponding double function, which explains the very good accuracy and the very small number of incorrectly-rounded results. The single precision `exp10` function is available as `__exp10f`.

**Notes about LLVM-libc.** Although the maximum observed distance is 0.5 ulp for `powf` and rounding to nearest, some inputs are not correctly rounded for other rounding modes, for example for $x$ = `0x1.9a1p-20` and $y$ = `0x1.ep+0`, LLVM 19.1.7 `powf` yields `0x1.b5e4d4p-37` for rounding towards zero instead of `0x1.b5e4d6p-37` ($x^y$ is exact).

**Notes about the Microsoft mathematical library.** The Bessel functions are not available in single precision (they are only available in double precision).

**Notes about FreeBSD.** We use the generic x86_64 binary from the FreeBSD distribution. Some slighly different results might be obtained if you recompile FreeBSD using fused-multiply-add.

# 3 Double Precision

For double precision it is not possible to perform an exhaustive search with academic resources. We thus use a black-box algorithm—described in §3.1—that tries to find large errors. Therefore, the values in the double-precision tables are only lower bounds of the largest error.
REMARK: We did not want to analyze the code of each library, since this approach would need more human work, and would require to start again from scratch for each new version of the library. We did not want either to design code specific to each function: for example for the double precision sine or cosine, one could test inputs near $2^{1024}$, to check the argument reduction is correctly done. We expect our search algorithm automatically detects such issues.

## 3.1 Search Algorithm

The idea of the algorithm is to subdivide recursively the set of values to search for. We describe it for a univariate double precision function, but it works for any IEEE format, as long as there is a corresponding integer type with the same bit-width, and it also works for bivariate functions.

Assume $f(x)$ is a univariate double precision function. The number of possible inputs of $f$ is less than $2^{64}$, thus each one can be mapped to a 64-bit integer. Assume we have a conversion function `to_double` from `uint64_t` to `double`. The algorithm takes as input a range $[a, b]$ of `uint64_t` values, and a threshold $t$. If $b - a < t$, it checks exhaustively all double precision values `x=to_double(i)` for $a \le i < b$. This means for each $x$, we compute the ulp-error $e$ between the value $y \approx f(x)$ returned by the corresponding library, and the exact result $z$ (as with infinite precision), as described in §1, and record the largest error.

If $b - a \ge t$, we subdivide the interval $[a, b]$ into two (almost) equal intervals, in each one we generate $t$ random values and compute the corresponding errors. We then recurse in the interval where we found the largest error.

For example with $t = 10^6$, the initial interval has $2^{64}$ values, thus we compute $f(x)$ on $2t$ random inputs $x$ ($t$ in each sub-range of $2^{63}$ values), and so on... The recursion stops when the width of the current interval is less than the total number of evaluations done so far in the recursion tree, thus an exhaustive search will at most double the search time.

In practice we use a variant of this algorithm suggested by Eric Schneider: instead of recursing only in the sub-interval giving the largest error on the random sample, we keep at each level of the search tree a list of say 20 intervals with the largest sample errors. Then we subdivide each of these intervals, which yields 40 smaller intervals (or 80 for bivariate functions), and keep again the 20 most promising ones.

We tried three variants of this algorithm, depending on how we choose the "best" sub-interval. The first strategy—described above—keeps the sub-interval with the largest ulp-error. A second strategy keeps the sub-interval with the maximal *average* ulp-error (considering only inputs which yield a meaningful ulp-error, discarding those giving NaN, zero or $\pm\infty$). A third strategy keeps the sub-interval with the largest expected ulp-error; for this, we estimate the mean and standard deviation of the ulp-error on each sub-interval, from which we deduce an estimate of the largest ulp-error for the number of points in the sub-interval [30]. In practice we found the first strategy

to be more effective, with the second and third strategies finding sometimes larger errors. Thus when the search program is run on a machine with $n$ cores, we assign one core to the second and third strategies, and $n-2$ cores to the first one.

The program also keeps track of the worst cases found for each library, and tries these input values for the other libraries. This helps determining the libraries using the same code base. The search programs (`check_sample.c` for univariate functions, and `check_sample2.c` for bivariate functions), the exhaustive search program for `binary32` univariate functions (`check_exhaustive.c`) and the source code of this article (containing in comment the $x$-values yielding the largest errors for `binary32`) are available from `https://gitlab.inria.fr/zimmerma/math_accuracy`.

We have also used the worst cases found by Vincent Lefèvre, publicly available at `https://www.vinc17.net/research/testlibm/`.

### 3.2 Results

We used a threshold of at least $t = 10^6$ for all libraries, often on processors with at least 32 cores, and the search program was run multiple times, cycling over all libraries, to detect common large errors.

Table 3 summarizes the largest known errors found using the above algorithm, for example the 0.531 entry for `acos` and IML means that for all inputs tried by the above algorithm, the error for the arc-cosine function with the Intel Math Library was bounded by 0.531 ulp. On each line, bold-face entries correspond to the best largest known error. Detailed tables (Tables 4-11) give the input values (in hexadecimal) yielding the corresponding ulp-error $e$, which enables the reader to reproduce our results.

In double precision, the Intel Math Library gives the best results in most cases. However, it was observed that the Intel Math Library gives better results on AMD hardware than on Intel hardware for `acosh`, `asin`, `asinh` and `atan2`; a possible explanation is that these functions use the `rsqrt` instruction, which is known to be more accurate on AMD hardware [6]. The square root function seems to be correctly rounded for all libraries, as required by IEEE 754. All LLVM functions except `cbrt` seem to be correctly rounded. Large errors occur for the AMD `cbrt` function, for the `j0`, `j1`, `y0` and `y1` functions for all libraries that provide them except the Intel Math Library, for the `lgamma` function from all libraries but the GNU and Intel libraries, for the `tgamma` function from Newlib, OpenLibm, the Apple library, MSVC and FreeBSD, for the power function from OpenLibm, MSVC and FreeBSD, and for `compound` from IML.

**Notes about AMD LibM.** An issue noticed in 4.1 (maybe it was there earlier) is still in 5.0: for subnormal numbers, `cbrt` gives huge errors, for example for $x$ = `-0x0.01fffffffffffp-1022`, it yields `-0x1.9d23e0bb9777ap-323` instead of the expected value `-0x1.fffffffffffabp-344`.

**Notes about Newlib.** For negative integers, Newlib `tgamma` returns an infinite value, whereas other libraries return NaN. The C standard says that a domain error or pole error may occur for a negative integer, but due to some non-standard dealing of Newlib with errno, we were unable to check whether it is the case. We however excluded negative integers for Newlib `tgamma` in our tests.

**Notes about the Microsoft mathematical library.** The `_y1` function yields NaN for $x = +0$ instead of -Inf, which explains the Inf value in Table 3.

| library | GNU libc | IML | AMD | Newlib | OpenLibm | Musl | Apple | LLVM | MSVC | FreeBSD | ArmPL | CUDA | ROCm |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | 2.41 | 2025.0.0 | 5.0 | 4.5.0 | 0.8.5 | 1.2.5 | 15.1.1 | 19.1.7 | 2022 | 14.2 | 24.04 | 12.8 | 6.2.4 |
| acos | **0.523** | 0.531 | 1.36 | 0.930 | 0.930 | 0.930 | 1.06 | | 0.934 | 0.930 | 1.52 | 1.53 | 0.772 |
| acosh | 2.25 | **0.509** | 1.32 | 2.25 | 2.25 | 2.25 | 2.25 | | 3.22 | 2.25 | 2.66 | 2.52 | 0.662 |
| asin | **0.516** | 0.531 | 1.06 | 0.981 | 0.981 | 0.981 | 0.709 | | 1.05 | 0.981 | 2.69 | 1.99 | 0.710 |
| asinh | 1.92 | **0.507** | 1.65 | 1.92 | 1.92 | 1.92 | 1.58 | | 2.05 | 1.92 | 2.04 | 2.57 | 0.661 |
| atan | **0.523** | 0.528 | 0.863 | 0.861 | 0.861 | 0.861 | 0.876 | | 0.863 | 0.861 | 2.24 | 1.77 | 1.73 |
| atanh | 1.78 | **0.507** | 1.04 | 1.81 | 1.81 | 1.80 | 2.01 | | 2.50 | 1.81 | 3.00 | 2.50 | 0.664 |
| cbrt | 3.67 | 0.523 | 1.53e22 | 0.670 | 0.668 | 0.668 | 0.729 | 4.51e15 | 1.86 | 0.668 | 1.79 | **0.501** | **0.501** |
| cos | **0.516** | 0.518 | 0.919 | 0.887 | 0.834 | 0.834 | 0.948 | **0.500** | 0.897 | 0.834 | | 1.52 | 0.798 |
| cosh | 1.93 | **0.516** | 1.85 | 2.67 | 1.47 | 1.04 | 0.523 | | 1.91 | 1.47 | 1.93 | 1.40 | 0.563 |
| erf | 1.43 | **0.773** | 1.00 | 1.02 | 1.02 | 1.02 | 6.41 | | 4.62 | 1.02 | 2.29 | 1.50 | 1.12 |
| erfc | 5.19 | **0.827** | | 4.08 | 4.08 | 3.72 | 10.7 | | 8.46 | 4.08 | 1.71 | 4.51 | 4.08 |
| exp | 0.511 | 0.530 | 1.01 | 0.949 | 0.949 | 0.511 | 0.521 | **0.500** | 1.50 | 0.949 | 0.511 | 0.928 | 0.929 |
| exp10 | 0.513 | 0.538 | 1.02 | 0.896 | | 4.14 | 0.521 | **0.500** | | | 0.510 | 1.11 | 1.11 |
| exp2 | 0.511 | 0.535 | 1.05 | 0.896 | 0.751 | 0.511 | 0.521 | **0.500** | 2.23 | 0.751 | 0.509 | 0.948 | 0.947 |
| expm1 | 0.911 | 0.512 | 0.670 | 0.909 | 0.909 | 0.909 | 0.706 | **0.500** | 3.06 | 0.909 | 2.18 | 1.18 | 1.91 |
| j0 | 4.51e14 | **0.600** | | 9.01e15 | 4.51e14 | 4.51e14 | 3.83e14 | | 1.88e26 | 4.51e14 | | 3.08e20 | 1.25e13 |
| j1 | 4.47e14 | **0.615** | | 9.01e15 | 1.10e15 | 1.10e15 | 1.10e15 | | 3.85e26 | 1.10e15 | | 1.73e21 | 4.80e13 |
| lgamma | 11.1 | **0.515** | | 4.45e15 | 4.45e15 | 4.45e15 | 2.33e16 | | 5.10e13 | 4.45e15 | | 5.11e15 | 4.45e15 |
| log | 0.520 | 0.518 | 0.610 | 0.946 | 0.946 | 0.520 | 0.508 | **0.500** | 0.577 | 0.946 | 0.520 | 0.564 | 0.663 |
| log10 | 1.62 | 0.532 | 1.09 | 2.08 | 0.814 | 0.814 | 0.514 | **0.500** | 0.633 | 0.814 | 1.62 | 1.43 | 0.785 |
| log1p | 0.899 | 0.521 | 0.636 | 0.896 | 0.896 | 0.900 | 0.667 | **0.500** | 1.44 | 0.896 | 1.74 | 1.50 | 1.00 |
| log2 | 0.548 | **0.500** | 1.72 | 2.06 | 0.921 | 0.555 | 0.515 | **0.500** | 0.812 | 0.921 | 0.554 | 1.31 | 0.734 |
| sin | **0.516** | 0.518 | 0.895 | 0.888 | 0.831 | 0.831 | 0.944 | **0.500** | 0.799 | 0.831 | | 1.52 | 0.800 |
| sinh | 1.93 | **0.521** | 1.49 | 2.67 | 1.88 | 1.88 | 0.539 | | 1.51 | 1.88 | 2.59 | 1.51 | 0.868 |
| sqrt | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** | | **0.500** | **0.500** |
| tan | 0.619 | **0.550** | 1.38 | 1.02 | 1.02 | 1.02 | 3.53 | **0.500** | 1.32 | 1.02 | | 2.09 | 1.30 |
| tanh | 2.21 | **0.556** | 1.40 | 2.22 | 2.22 | 2.22 | 0.613 | | 1.44 | 2.22 | 2.76 | 1.48 | 0.866 |
| tgamma | 8.68 | **0.519** | | 2.27e3 | 1.03e3 | 16.0 | 1.03e3 | | 9.01e15 | 1.03e3 | | 10.1 | 13.7 |
| y0 | 5.93e15 | **1.14** | | 1.42e15 | 1.42e15 | 1.42e15 | 1.42e15 | | 3.30e25 | 1.42e15 | | 1.18e21 | 1.95e13 |
| y1 | 5.56e15 | **1.25** | | 5.56e15 | 5.56e15 | 5.56e15 | 5.56e15 | | Inf | 5.56e15 | | 1.17e21 | 6.14e13 |
| acospi | 1.51 | **0.541** | | | | | | | | | | | |
| asinpi | 1.50 | **0.521** | | | | | | | | | | | |
| atanpi | 1.51 | **0.942** | | | | | | | | | | | |
| cospi | 1.86 | **0.503** | 2.47 | | | | | | | 0.812 | 3.18 | 1.52 | 0.894 |
| exp10m1 | **3.54** | | | | | | | | | | | | |
| exp2m1 | **1.93** | | | | | | | | | | | | |
| log10p1 | **1.95** | | | | | | | | | | | | |
| log2p1 | **1.88** | | | | | | | | | | | | |
| sinpi | 1.86 | **0.532** | 2.56 | | | | | | | 0.811 | 3.16 | 1.52 | 0.892 |
| tanpi | 2.85 | 1.00 | 2.56 | | | | | | | **0.969** | | | |
| rsqrt | | **0.501** | | | | | | | | | | 0.510 | 1.00 |
| atan2 | **0.524** | 0.548 | 1.51 | 1.55 | 1.55 | 1.55 | 0.747 | | 0.750 | 1.55 | 2.23 | 1.76 | 1.82 |
| atan2pi | 1.49 | **1.02** | | | | | | | | | | | |
| compound | | **1.13e15** | | | | | | | | | | | |
| hypot | 0.792 | 0.751 | 1.03 | 1.21 | 1.21 | 1.04 | 1.21 | **0.500** | 1.22 | 1.21 | | 1.89 | 1.21 |
| pow | **0.523** | 1.73 | 1.00 | 0.892 | 636. | 0.525 | 0.757 | | 91.3 | 636. | **0.523** | 1.84 | 1.40 |

Table 3: Double precision: Largest known error.

| function | GNU libc 2.41<br>$x$ | IML 2025.0.0<br>$x$ |
|---|---|---|
| acos | 0x1.dffffb3488a4p-1 | 0x1.6c05eb219ec46p-1 |
| acosh | 0x1.0001ff6afc4bap+0 | 0x1.01825ca7da7e5p+0 |
| asin | -0x1.0000045b2c904p-3 | 0x1.6c042a6378102p-1 |
| asinh | -0x1.02657ff36d5f3p-2 | -0x1.000276d9cf31ap-4 |
| atan | 0x1.f9004c4fef9eap-4 | -0x1.ffff8020d3d1dp-7 |
| atanh | -0x1.ebb5133a9d9a4p-4 | -0x1.e2cfb2667f17ep-9 |
| cbrt | 0x1.7a337e1ba1ec2p-257 | -0x1.f7af4893d1d51p-616 |
| cos | -0x1.7120161c92674p+0 | -0x1.d19ebc5567dcdp+311 |
| cosh | -0x1.633c654fee2bap+9 | -0x1.5a364e6b98134p+9 |
| erf | 0x1.c332bde7ca515p-5 | -0x1.c5bcc48b2d463p-21 |
| erfc | 0x1.3ff2d63705b29p+0 | 0x1.a8cf81fa2a28fp+4 |
| exp | -0x1.49f33ad2c1c58p+9 | 0x1.fce66609f7428p+5 |
| exp10 | -0x1.57449153f316ep-7 | -0x1.5cd9d94d49a85p+1 |
| exp2 | -0x1.1a4ce073ea908p-5 | -0x1.8002f29666b99p-6 |
| expm1 | 0x1.63be411e096ep-2 | -0x1.62fe464c64f65p-8 |
| j0 | 0x1.33d152e971b4p+1 | 0x1.aff859518c846p+7 |
| j1 | -0x1.ea75575af6f09p+1 | -0x1.67b5541c7d8b7p+7 |
| lgamma | -0x1.f613ab0969f81p+1 | -0x1.3f62c60e23b31p+2 |
| log | 0x1.1211bef8f68e9p+0 | 0x1.008000db2e8bep+0 |
| log10 | 0x1.de02157073b31p-1 | 0x1.feda7b62c1033p-1 |
| log1p | -0x1.2bf183e0344b2p-2 | 0x1.000aee2a2757fp-9 |
| log2 | 0x1.1406d79e1b574p+0 | 0x1.b4ebe40c95a01p+0 |
| sin | -0x1.f8b791cafcdefp+4 | -0x1.0e16eb809a35dp+944 |
| sinh | -0x1.633c654fee2bap+9 | -0x1.adc135eb544c1p-2 |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | -0x1.317cd745dd37cp+9 | 0x1.49adfd996a81dp+18 |
| tanh | 0x1.e126eee514cbcp-3 | 0x1.002629fd74484p+0 |
| tgamma | -0x1.c18caecc00f7bp+2 | -0x1.3e0001ad3bee3p+6 |
| y0 | 0x1.c982eb8d417eap-1 | 0x1.4cdee58a47eddp-31 |
| y1 | 0x1.193bed4dff243p+1 | 0x1.c513c569fe78ep+0 |
| acospi | 0x1.da0ea62dd7231p-1 | 0x1.6a18f7dda5343p-1 |
| asinpi | -0x1.8805060cb885cp-3 | 0x1.921fabd2a2b25p-750 |
| atanpi | 0x1.9601b055fdf97p-3 | 0x1.a7fb417ac774ap-2 |
| cospi | -0x1.1a0a2fa299b92p+6 | -0x1.f01d619f61e5bp-8 |
| exp10m1 | 0x1.e880c5bafbd41p-2 | |
| exp2m1 | 0x1.d047583a6c6dp-1 | |
| log10p1 | -0x1.4c2971893052fp-1 | |
| log2p1 | 0x1.a7b725780ff2cp-2 | |
| sinpi | -0x1.45f3e53e1d707p-7 | 0x0.07f16ec91c164p-1022 |
| tanpi | -0x1.fae7d0ef22d4ep-2 | 0x1.49b79692667bp+46 |
| rsqrt | | 0x1.00018f5913816p-458 |

Table 4: Double precision: inputs giving the largest known error in GNU libc and the Intel Math Library (univariate functions).

| function | GNU libc 2.41<br>$x, y$ | IML 2025.0.0<br>$x, y$ |
|---|---|---|
| atan2 | 0x1.ed6060626eecfp-429<br>0x1.f42ebb62994dcp-426 | 0x1.b77ade79a36d5p-326<br>0x1.ff6a37b72b52bp-319 |
| atan2pi | -0x1.fe856e7997f8p+381<br>0x1.90ece816f9a7cp+343 | -0x1.026462f302171p-391<br>0x1.39b157b1210a4p-390 |
| compound | | -0x1.0000000000001p-1<br>0x1p+10 |
| hypot | 0x0.603e52daf0bfdp-1022<br>-0x0.a622d0a9a433bp-1022 | 0x0.19deaac345ffap-1022<br>0x0.92c8727c389b6p-1022 |
| pow | 0x1.010e2e7ee71aep+0<br>0x1.44bf0047427f6p+17 | 0x1.fffff9c61ce4p-1<br>0x1.c4e304ed4c734p+31 |

Table 5: Double precision: inputs giving the largest known error in GNU libc and the Intel Math Library (bivariate functions).

### 3.3 Other Functions

For other functions from the C standard like `ldexp`, correct rounding might at first glance seem easier to implement. However, as reported by Fred Tydeman, cutting the smallest subnormal number in half does not always return zero. We noticed that for rounding upwards, for $x$ = `0x1p-1074`, Newlib 4.5.0 (already reported for 4.3.0) returns 0 for `ldexp(x,-1)` instead of $x$. There is a similar issue for `ldexpf` and `ldexpl`.

## 4 Double Extended Precision

This format corresponds to the C type `long double` on `x86_64` processors. Some libraries do not provide double extended precision, or do not provide mathematical functions for this format. The results are summarized in Table 12, and detailed in Tables 13-16. We see that in this format, the Intel Math library is better than all other libraries for all functions it provides, both univariate and bivariate, except for the `hypot` function.

**Notes about GNU libc.** We noticed that for extended double precision, the GNU libc can give different results on x86_64 hardware, whether it is Intel or AMD hardware. For example for $x$ = `0xb.1722e675ab2f4c1p-5`, `expm1l` yields `0xd.413ec8a7dda564ap-5` on AMD cpu, and `0xd.413ec8a7dda5648p-5` on Intel cpu.

**Notes about the Intel Math Library.** The `j0l`, `j1l`, `y0l`, and `y1l` functions call the corresponding quadruple precision function, which explains why the largest error is near 0.5 ulp in our experiments (for the `j1l`, `y0l` and `y1l` functions, we found inputs that are not correctly rounded thanks to the BaCSeL software tool).

**Notes about Newlib.** Newlib only provides long double functions for platforms where `long double` is the same as `double` (which is not the case of the `x86_64` processor) with two exceptions:

| function | AMD LibM 5.0<br>$x$ | Newlib 4.5.0<br>$x$ |
|---|---|---|
| acos | 0x1.35b03e336a82bp-1 | -0x1.0068b067c6feep-1 |
| acosh | 0x1.209fae707a0edp+0 | 0x1.0001ff6afc4bap+0 |
| asin | -0x1.00d44cccfa99p-1 | -0x1.004d1c5a9400bp-1 |
| asinh | 0x1.005ae8d126f7ep+0 | -0x1.02657ff36d5f3p-2 |
| atan | -0x1.60370d15396b7p-1 | 0x1.62ff6a1682c25p-1 |
| atanh | -0x1.d8fb311a52173p-2 | -0x1.f97fabc0650c4p-4 |
| cbrt | 0x0.7fffffffffffp-1022 | -0x1.00ddafe7d9deep-885 |
| cos | 0x1.91e60af551108p-1 | -0x1.4ae182c1ab422p+21 |
| cosh | 0x1.ff76fb3f476d5p+0 | 0x1.633cc2ae1c934p+9 |
| erf | 0x1.11642f2eab9edp+0 | -0x1.c57541b55c8ebp-16 |
| erfc | | 0x1.5182d8799b84bp+0 |
| exp | 0x1.b97dc8345c55p+5 | 0x1.2e8f20cf3cbe7p+8 |
| exp10 | -0x1.285d82b75258fp+2 | 0x1.ce7ef793d4b0ap-2 |
| exp2 | 0x1.fffbff4152bafp+9 | -0x1.ff95ecb4e6331p-2 |
| expm1 | 0x1.9a57c1ec06a74p-2 | 0x1.62ff47a01658fp-2 |
| j0 | | 0x1.45f3067a0f4b2p+847 |
| j1 | | 0x1.45f3066f80258p+325 |
| lgamma | | -0x1.3a7fc9600f86cp+1 |
| log | 0x1.10803885617a6p+0 | 0x1.48ae5a67204f5p+0 |
| log10 | 0x1.10fdf4211fd45p+0 | 0x1.55535a0140a21p+0 |
| log1p | 0x1.e0013fd35cbbp-4 | -0x1.2bf1de6b04a8ap-2 |
| log2 | 0x1.0b541b6746bd1p+0 | 0x1.68d778f076021p+0 |
| sin | -0x1.85e624577c23ep-1 | -0x1.842d8ec8f752fp+21 |
| sinh | 0x1.1feb2a79f307p+3 | -0x1.633cae1335f26p+9 |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | 0x1.371a47b7e4eb2p+11 | 0x1.3f9605aaeb51bp+21 |
| tanh | -0x1.fde5bd2769a01p-1 | -0x1.e134557098e37p-3 |
| tgamma | | -0x1.535175475cc8dp+7 |
| y0 | | 0x1.c982eb8d417eap-1 |
| y1 | | 0x1.193bed4dff243p+1 |
| cospi | -0x1.fffffffffffffp-28 | |
| sinpi | -0x1.ffffed20ca4dep-14 | |
| tanpi | 0x1.ffffce32e4a7cp-15 | |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | -0x1.ec2003fe89e36p-433<br>0x1.ec173ed5fd8f5p-427 | -0x1.358bb5eb25bdcp+813<br>0x1.2f86b82481a0ap+815 |
| hypot | 0x1.ea74e4b1da32bp+1002<br>-0x1.0a9efc37c242cp+1003 | 0x1.6a0a41410b1abp-1004<br>-0x0.a24afe71b539fp-1022 |
| pow | 0x1.3bbe002a257a7p-771<br>0x1.64bf51fe77808p+0 | 0x1.b44c681a51345p-822<br>0x1.3e262867f583fp-11 |

Table 6: Double precision: inputs giving the largest known error in AMD LibM and Newlib.

| function | OpenLibm 0.8.5<br>$x$ | Musl 1.2.5<br>$x$ |
|---|---|---|
| acos | -0x1.0068b067c6feep-1 | -0x1.0068b067c6feep-1 |
| acosh | 0x1.0001ff6afc4bap+0 | 0x1.0001ff6afc4bap+0 |
| asin | -0x1.004d1c5a9400bp-1 | -0x1.004d1c5a9400bp-1 |
| asinh | -0x1.02657ff36d5f3p-2 | -0x1.0240f2bdb3f25p-2 |
| atan | 0x1.62ff6a1682c25p-1 | 0x1.62ff6a1682c25p-1 |
| atanh | -0x1.f97fabc0650c4p-4 | -0x1.f8a404597baf4p-4 |
| cbrt | -0x1.13a5ccd87c9bbp+1008 | -0x1.13a5ccd87c9bbp+1008 |
| cos | -0x1.34e729fd08086p+21 | -0x1.34e729fd08086p+21 |
| cosh | -0x1.6310ab92794a8p+9 | -0x1.502bf5ad80729p+0 |
| erf | -0x1.c57541b55c8ebp-16 | -0x1.c57541b55c8ebp-16 |
| erfc | 0x1.5182d8799b84bp+0 | 0x1.527f4fb0d9331p+0 |
| exp | 0x1.2e8f20cf3cbe7p+8 | -0x1.18209ecd19a8cp+6 |
| exp10 | | -0x1.fe8c27141c94ap+3 |
| exp2 | -0x1.ff1eb5acee46bp+9 | -0x1.1a4ce073ea908p-5 |
| expm1 | 0x1.62ff47a01658fp-2 | 0x1.62ff47a01658fp-2 |
| j0 | 0x1.33d152e971b4p+1 | -0x1.33d152e971b4p+1 |
| j1 | -0x1.ea75575af6f09p+1 | 0x1.ea75575af6f09p+1 |
| lgamma | -0x1.3a7fc9600f86cp+1 | -0x1.3a7fc9600f86cp+1 |
| log | 0x1.48ae5a67204f5p+0 | 0x1.dc0b586f2b26p-1 |
| log10 | 0x1.553e1cb579ee9p+0 | 0x1.553e1cb579ee9p+0 |
| log1p | -0x1.2bf1de6b04a8ap-2 | -0x1.2bf32aaf122e2p-2 |
| log2 | 0x1.67eaf07ce24d1p+0 | 0x1.0b53197bd66c8p+0 |
| sin | 0x1.4d84db080b9fdp+21 | 0x1.4d84db080b9fdp+21 |
| sinh | -0x1.63324af2fb5b7p-1 | -0x1.63324af2fb5b7p-1 |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | 0x1.3f9605aaeb51bp+21 | 0x1.3f9605aaeb51bp+21 |
| tanh | -0x1.e134557098e37p-3 | -0x1.e134557098e37p-3 |
| tgamma | -0x1.540b170c4e65ep+7 | -0x1.fc4b534c8eccp+2 |
| y0 | 0x1.c982eb8d417eap-1 | 0x1.c982eb8d417eap-1 |
| y1 | 0x1.193bed4dff243p+1 | 0x1.193bed4dff243p+1 |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | -0x1.358bb5eb25bdcp+813<br>0x1.2f86b82481a0ap+815 | -0x1.358bb5eb25bdcp+813<br>0x1.2f86b82481a0ap+815 |
| hypot | 0x1.6a0a41410b1abp-1004<br>-0x0.a24afe71b539fp-1022 | 0x1.00014d4b1c6b9p-1015<br>-0x1.000105ba9bf4p-1015 |
| pow | 0x1.000002c5e2e99p+0<br>0x1.c9eee35374af6p+31 | 0x1.010e2e7ec0c83p+0<br>0x1.44bf00479249dp+17 |

Table 7: Double precision: inputs giving the largest known error in OpenLibm and Musl.

| function | Apple 15.1.1<br>$x$ | LLVM 19.1.7<br>$x$ |
|---|---|---|
| acos | -0x1.8d313198a2e03p-53 | |
| acosh | 0x1.00007fb3703ddp+0 | |
| asin | 0x1.eae75e3d82b6fp-2 | |
| asinh | -0x1.fdefd03df4cd7p-3 | |
| atan | 0x1.01e0be37af68fp+1 | |
| atanh | 0x1.ffd834a270fp-10 | |
| cbrt | 0x1.fed1fe9f1122dp+11 | |
| cos | 0x1.2f29eb4e99fa2p+7 | -0x1.13a5ccd87c9bbp+1008 |
| cosh | -0x1.62dabd4848dc4p-2 | |
| erf | -0x1.e057e7a0e494cp-2 | |
| erfc | 0x1.bba14dc3507ccp+1 | |
| exp | -0x1.4133f4fd79c1cp-13 | 0x1p-53 |
| exp10 | -0x1.c37443e446523p-16 | 0x1.bcb7b1526e50dp-55 |
| exp2 | -0x1.b3d9b47ad1b2fp-13 | 0x1.71547652b82fep-53 |
| expm1 | 0x1.e7f93188565ecp-5 | 0x1p-52 |
| j0 | 0x1.6148f5b2c2e45p+2 | |
| j1 | -0x1.ea75575af6f09p+1 | |
| lgamma | -0x1.bffcbf76b86fp+2 | |
| log | 0x1.490af72a25a81p-1 | 0x1.5b6e7e4e96f86p+2 |
| log10 | 0x1.2501ee5628b08p-1 | 0x1.e12d66744ff81p+429 |
| log1p | -0x1.ffffff3fffffdp-28 | 0x1p-53 |
| log2 | 0x1.6b015f8d9a784p-1 | 0x1.b4ebe40c95a01p+0 |
| sin | -0x1.07e4c92b5349dp+4 | -0x1.13a5ccd87c9bbp+1008 |
| sinh | 0x1.d7131e11fc6b3p-2 | |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | -0x1.a81d98fc58537p+6 | -0x1.13a5ccd87c9bbp+1008 |
| tanh | 0x1.00cf9f273d84p+1 | |
| tgamma | -0x1.5456e56919a19p+7 | |
| y0 | 0x1.c982eb8d417eap-1 | |
| y1 | 0x1.193bed4dff243p+1 | |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | -0x1.6a539153430d8p-416<br>0x1.d2b5b9dc716d8p-415 | |
| hypot | 0x1.6a0a41410b1abp-1004<br>-0x1.4495fce36a73ep-1023 | 0x1.a308e1455f447p+0<br>0x1.9d931a83ef879p+0 |
| pow | 0x1.111616f835fb1p-72<br>0x1.c6cfa07925d49p+3 | |

Table 8: Double precision: inputs giving the largest known error in the Apple and LLVM Math Libraries.

| function | MSVC 2022 $x$ | FreeBSD 14.2 $x$ |
|---|---|---|
| acos | -0x1.010fd0ad6aa41p-1 | -0x1.0068b067c6feep-1 |
| acosh | 0x1.0007fd4307b75p+0 | 0x1.0001ff6afc4bap+0 |
| asin | -0x1.0239000439deep-1 | -0x1.004d1c5a9400bp-1 |
| asinh | -0x1.00212bb59c31ep-4 | -0x1.02657ff36d5f3p-2 |
| atan | -0x1.60370d15396b7p-1 | 0x1.62ff6a1682c25p-1 |
| atanh | -0x1.ffbe8dd88527fp-9 | -0x1.f97fabc0650c4p-4 |
| cbrt | -0x1.55cd285f321f6p-64 | -0x1.13a5ccd87c9bbp+1008 |
| cos | -0x1.9200634d4471fp-1 | -0x1.34e729fd08086p+21 |
| cosh | -0x1.1ff088806d82ep+3 | -0x1.6310ab92794a8p+9 |
| erf | 0x1.755dca4d8b458p+0 | -0x1.c57541b55c8ebp-16 |
| erfc | 0x1.f6094003e85d6p+1 | 0x1.5182d8799b84bp+0 |
| exp | -0x1.74046dfefd9d1p+9 | 0x1.2e8f20cf3cbe7p+8 |
| exp2 | -0x1.72286b6f94510p-2 | -0x1.ff1eb5acee46bp+9 |
| expm1 | -0x1.62d7c116d2e32p-1 | 0x1.62ff47a01658fp-2 |
| j0 | 0x1.bbbd6a9201265p+657 | 0x1.33d152e971b4p+1 |
| j1 | 0x1.a635d8219ad13p+157 | -0x1.ea75575af6f09p+1 |
| lgamma | -0x1.bffe071eea741p+2 | -0x1.3a7fc9600f86cp+1 |
| log | 0x1.0ffc349469a2fp+0 | 0x1.48ae5a67204f5p+0 |
| log10 | 0x1.e005e1e891807p-1 | 0x1.553e1cb579ee9p+0 |
| log1p | -0x1.8000000000000p-53 | -0x1.2bf1de6b04a8ap-2 |
| log2 | 0x1.160732376ad7fp+0 | 0x1.67eaf07ce24d1p+0 |
| sin | -0x1.11b624b546894p+9 | 0x1.4d84db080b9fdp+21 |
| sinh | -0x1.aff899f6fcad6p+4 | -0x1.63324af2fb5b7p-1 |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | -0x1.4d7c8b8320237p+11 | 0x1.3f9605aaeb51bp+21 |
| tanh | -0x1.fb52ec8460d82p-1 | -0x1.e134557098e37p-3 |
| tgamma | -0x1.5c00000003c15p+7 | -0x1.547cf3ddaaddap+7 |
| y0 | 0x1.c1d741dc52512p+744 | 0x1.c982eb8d417eap-1 |
| y1 | +0 | 0x1.193bed4dff243p+1 |
| cospi | | -0x1.fe3bb5207682dp-3 |
| sinpi | | 0x1.0806840ac80ap+3 |
| tanpi | | 0x1.c9542a6e8f18fp+0 |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | -0x1.f1037a6756bfep-881<br>0x1.959f99be632e6p+142 | -0x1.358bb5eb25bdcp+813<br>0x1.2f86b82481a0ap+815 |
| hypot | -0x1.6a5a0ce661358p+890<br>-0x1.0151c108425b1p+890 | 0x1.6a0a41410b1abp-1004<br>-0x1.4495fce36a73ep-1023 |
| pow | 0x1.fffff9c61ce40p-1<br>0x1.c4e304ed4c734p+31 | 0x1.000002c5e2e99p+0<br>0x1.c9eee35374af6p+31 |

Table 9: Double precision: inputs giving the largest known error in the Microsoft and FreeBSD Math Libraries.

| function | ArmPL 24.04<br>$x$ |
|---|---|
| acos | 0x1.251869c3f7881p-1 |
| acosh | 0x1.071334daf83adp+0 |
| asin | 0x1.0479b37d95e5cp-1 |
| asinh | -0x1.000eeed78380ap+0 |
| atan | 0x1.032b4811f3dc5p+0 |
| atanh | -0x1.e7c1f36602014p-4 |
| cbrt | 0x1.fffad1cec59fep-332 |
| cosh | -0x1.628af341989dap+9 |
| erf | 0x1.0000187085e56p-8 |
| erfc | 0x1.46cffdf330b13p+4 |
| exp | -0x1.49f33ad2c1c58p+9 |
| exp10 | -0x1.5acf96f42165bp-7 |
| exp2 | -0x1.f7087fb1cf9e8p+9 |
| expm1 | 0x1.633f993a730c9p-2 |
| log | 0x1.1211bef8f68e9p+0 |
| log10 | 0x1.de02157073b31p-1 |
| log1p | -0x1.2e496d25897ecp-2 |
| log2 | 0x1.0b53f741fb8c4p+0 |
| sinh | 0x1.9fa32b1149d35p-2 |
| tanh | -0x1.c416448380debp-3 |
| cospi | 0x1.5a33cc258p-22 |
| sinpi | 0x1.ffffe4b85e77ap-2 |

| function | $x, y$ |
|---|---|
| atan2 | 0x1.d5de7a294d493p-935<br>0x1.d030d7b608be1p-935 |
| pow | 0x1.010e2e7ee71aep+0<br>0x1.44bf0047427f6p+17 |

Table 10: Double precision: inputs giving the largest known error in the Arm Performance Library.

| function | CUDA 12.8<br>$x$ | ROCm 6.2.4<br>$x$ |
|---|---|---|
| acos | 0x1.266637a3d2bbcp-1 | -0x1.36b1482765f6dp-1 |
| acosh | 0x1.1d7bc19163966p+0 | 0x1.0aaade11e199ap+0 |
| asin | -0x1.2ef2481799c7cp-1 | 0x1.df27e1c764802p-2 |
| asinh | 0x1.0ab3fc30267c2p-1 | 0x1.2aae7abf26ce3p-2 |
| atan | 0x1.52184b1b9bd9bp+0 | -0x1.0684fa9fa7481p+0 |
| atanh | -0x1.f586714622a66p-3 | -0x1.2493fecd07e5p-3 |
| cbrt | -0x1.588a24f7a97d3p+535 | 0x1.1e0ef6faa076p+175 |
| cos | 0x1.25133ca3904dfp+20 | 0x1.2a33ae1a620b1p+1 |
| cosh | 0x1.e7ffe229fe99ep+1 | -0x1.e7fa36b6eb43p+1 |
| erf | 0x1.340ff534d52bfp-2 | -0x1.10c4c3d3b6cdbp+0 |
| erfc | 0x1.8659a03b35abcp-7 | 0x1.f27ea468cbe91p-19 |
| exp | -0x1.625f1b359729ep+9 | -0x1.625f1c27780c8p+9 |
| exp10 | -0x1.a7d980016dc5ap+0 | 0x1.5c1ece7fea4bep+0 |
| exp2 | -0x1.ff40169bd093bp+9 | -0x1.ff3e3278714d8p+9 |
| expm1 | 0x1.a0e95d59498e9p-2 | 0x1.632cfb1033275p-2 |
| j0 | -0x1.0e126bbcb3e65p+25 | 0x1.ddca13ef271d2p+3 |
| j1 | -0x1.635ab5a8baf45p+26 | 0x1.aa5baf310e5a2p+3 |
| lgamma | -0x1.fa471547c2fe5p+1 | -0x1.3a7fc9600f86cp+1 |
| log | 0x1.69e7aa6da2df5p-1 | 0x1.5556123e8a2bp-1 |
| log10 | 0x1.803dea263187fp-1 | 0x1.55558196a2cbp+0 |
| log1p | -0x1.ffffffbaefe27p-2 | -0x1.5efad5491a79bp-1022 |
| log2 | 0x1.670c5aa6680abp+0 | 0x1.55562d11c546fp+0 |
| sin | -0x1.1c49ad613ff3bp+19 | -0x1.f05e952d81b89p+5 |
| sinh | 0x1.be64384e3ac1ep+0 | -0x1.ff9faf9b69235p-5 |
| sqrt | 0x1.fffffffffffffp-1 | 0x1.fffffffffffffp-1 |
| tan | 0x1.da7a85a88bbecp+11 | -0x1.66af736e8555p+18 |
| tanh | -0x1.19398a9a24319p-1 | 0x1.00433533940cdp-4 |
| tgamma | -0x1.2baa17692a3f2p+7 | -0x1.201a11d80c13dp+2 |
| y0 | 0x1.16bad92479879p+25 | 0x1.ab8e1c4a1e74ap+3 |
| y1 | 0x1.2391e4c8faa6p+26 | 0x1.e9e480605283cp+4 |
| cospi | -0x1.ae7b6f6da3747p-2 | -0x1.ffb154d307aa1p-3 |
| sinpi | -0x1.4778e04770c45p-4 | 0x1.0092557af9571p-2 |
| rsqrt | 0x1.f81ab000eb7d9p+479 | 0x1.0000000000002p+484 |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | 0x1.9cde4ff190e45p+931<br>0x1.37d91467e558bp+931 | 0x1.401ec07d65549p+888<br>0x1.3c3976605bb0cp+888 |
| hypot | -0x1.41fcfeeb2e246p+420<br>-0x1.8d4d41eacdeccp+420 | 0x1.afa7134ad6d8p-403<br>0x1.6a0ff6e086067p-384 |
| pow | 0x1.6b2d4fdb85ba1p-1<br>-0x1.f0d1d713b0262p+10 | 0x1.17efb14831458p-421<br>0x1.f8c34d6504b2p-7 |

Table 11: Double precision: inputs giving the largest known error in CUDA and ROCm.

sqrt and hypot. However, in Newlib 4.5.0, the hypotl function does not work properly: for $x \geq 2^{8192}$, the call hypotl(x,0) gives infinity.

**Notes about OpenLibm.** The powl function does not seem to be thread-safe, the tgammal function yields +Inf for x=-0x6.db747ae147ae148p+8 instead of 0x0.01dbd551da54538p-16385, and expm1l yields NaN instead of 0xf.ffffcfce79e56d5p+16380 for x=0x2.c5c85fdf170c604cp+12.

**Notes about the Apple Math Library.** The Apple Darwin ABI for ARM processors maps the C long double type to double, thus there is no real "double extended" format. The same holds for the Microsoft math library.

**Notes about LLVM-libc.** The LLVM-libc library only implements the square root function in double-extended precision, and for this function we could not find any error larger than 0.5 ulp (for rounding to nearest). Since a single function is implemented, we don't mention LLVM-libc in Tables 12-16.

**Notes about FreeBSD.** For x=-0x6.e000000000000008p+8 which is very near a negative integer, the tgammal function yields -0 instead of -0x4.b40cdf839d0bbp-16392.

# 5 Quadruple Precision

Only the GNU libc and the Intel Math Library support quadruple precision, through the _Float128 type in GNU libc, and _Quad in the Intel Math Library (using the option of the Intel C compiler -Qoption,cpp,--extended_float_types). The results are summarized in Table 17, and detailed in Table 18. Only the square root function is correctly rounded (or at least seems to be). The Intel Math Library gives better results than the GNU libc for all functions it provides, except for lgamma and tgamma. Apart from these two functions, and from the Bessel functions j0, j1, y0, y1, the observed error for the Intel Math Library is at most 1.4 ulps. The GNU libc has large errors for j0, j1, y0, y1, and pow.

**Acknowledgements.** The authors thank Claude-Pierre Jeannerod, Vincent Lefèvre and Terje Mathisen who helped improving that article, Alexei Sibidanov who helped compiling Newlib, Eric Schneider, Nick Timmons, Hugues de Lassus, Fred J. Tydeman and Sélène Corbineau for interesting discussions. Joseph Myers suggested to included the double extended format. Experiments presented in this article were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work was also supported by the French "Ministère de l'Enseignement Supérieur et de la Recherche", by the "Conseil Régional de Lorraine", and by the European Union, through the "Cyber-Entreprises" project. Experiments on GPU were performed on hardware made available by CERN. Experiments with the Microsoft library were possible thanks to Brice Goglin and the TADaaM project-team from Inria.

| library | GNU libc | Intel Math Library | OpenLibm | Musl | FreeBSD |
|---|---|---|---|---|---|
| version | 2.41 | IML 2025.0.0 | 0.8.5 | 1.2.5 | 14.2 |
| acos | 1.75 | **0.505** | 0.938 | 1.75 | 0.938 |
| acosh | 2.99 | **0.502** | 3.14 | 2.99 | 2.24 |
| asin | 1.15 | **0.506** | 1.03 | 2.00 | 1.03 |
| asinh | 2.96 | **0.506** | 3.19 | 2.96 | 1.63 |
| atan | 0.640 | **0.501** | 1.10 | 0.640 | 1.10 |
| atanh | 2.88 | **0.501** | 85.4 | 3.19 | 1.52 |
| cbrt | 0.824 | **0.503** | 0.890 | 0.890 | 0.890 |
| cos | 1.51 | **0.502** | 0.799 | 0.799 | 0.799 |
| cosh | 3.40 | **0.502** | 4.86 | 3.73 | 0.936 |
| erf | 1.17 | **0.518** | 1.17 | 1.17 | 0.995 |
| erfc | 4.73 | **0.527** | 5.77 | 5.12 | 1.38e8 |
| exp | 1.27 | **0.501** | 2.00 | 1.54 | 0.752 |
| exp10 | 1.50 | **0.501** | | 40.1 | |
| exp2 | 0.788 | **0.501** | 2.18 | 0.788 | 0.753 |
| expm1 | 3.08 | **0.502** | Inf | 9.71e3 | 0.517 |
| j0 | 9.79e17 | **0.501** | | | |
| j1 | 3.38e18 | **0.501** | | | |
| lgamma | 12.2 | **0.549** | 9.08e19 | 9.08e19 | 1.65e20 |
| log | 0.998 | **0.501** | 1.22 | 0.998 | 0.512 |
| log10 | 1.36 | **0.502** | 1.22 | 1.36 | 0.511 |
| log1p | 2.49 | **0.501** | 2.60 | 2.49 | 0.516 |
| log2 | 0.995 | **0.502** | 1.64 | 0.995 | 0.509 |
| sin | 1.51 | **0.502** | 0.799 | 0.799 | 0.799 |
| sinh | 3.40 | **0.503** | 4.85 | 9.71e3 | 0.803 |
| sqrt | **0.500** | **0.500** | **0.500** | **0.500** | **0.500** |
| tan | 1.76 | **0.504** | 1.02 | 1.02 | 1.02 |
| tanh | 3.22 | **0.506** | 2.56 | 2.95 | 0.640 |
| tgamma | 9.77 | **0.550** | Inf | 3.69e19 | 4.24e16 |
| y0 | 1.38e18 | **0.501** | | | |
| y1 | 4.61e18 | **0.501** | | | |
| acospi | **2.86** | | | | |
| asinpi | **2.09** | | | | |
| atanpi | **1.56** | | | | |
| cospi | 1.80 | | | | **0.797** |
| exp10m1 | **3.57** | **3.57** | | | |
| exp2m1 | **3.20** | **3.20** | | | |
| log10p1 | **4.17** | **4.17** | | | |
| log2p1 | **3.95** | **3.95** | | | |
| sinpi | 1.80 | | | | **0.797** |
| tanpi | 3.02 | | | | **1.50** |
| rsqrt | | **0.501** | | | |
| atan2 | 0.751 | **0.501** | 1.69 | 0.751 | 1.69 |
| atan2pi | **1.60** | | | | |
| compound | | **0.502** | | | |
| hypot | **0.584** | 0.751 | 0.981 | 1.08 | 0.981 |
| pow | 0.914 | **0.501** | 3.77e4 | 3.77e4 | 3.77e4 |

Table 12: Double extended precision: Largest known error.

| function | GNU libc 2.41<br>$x$ | IML 2025.0.0<br>$x$ |
|---|---|---|
| acos | 0xf.fe002cabd608585p-4 | 0x8.af256cd27462348p-4 |
| acosh | 0x1.1ecdb5b8f0c5d79p+0 | 0x1.1f9c4feedfe4f2cp+0 |
| asin | 0x8.171fd358c4cb27bp-4 | -0x8.018aef8787e5a6bp-4 |
| asinh | -0x8.0bb656992eac437p-4 | 0x7.ff15da44c3651abp-4 |
| atan | -0x1.0411ae010d4c5b1ep+0 | -0x8.00f60592e42d79p+8 |
| atanh | -0x3.337ceaccc9025258p-4 | 0x3.e7be418257523408p-4 |
| cbrt | -0xc.f4fd71a450e6a0bp-14732 | -0x2.320375fd33ed311cp-13376 |
| cos | -0x3.d067a048093bdf94p+9160 | -0x4.b0df0d7d55044918p+8 |
| cosh | 0x2.c5d375f827733ac4p+12 | -0x7.f6a09874512cf768p-4 |
| erf | 0xd.7fe64ab05cf75e8p-4 | -0x1.c55160e785ee1cbap-4 |
| erfc | 0x1.59723d7ee47e3034p+0 | 0x3.03c7b9f943690558p-4 |
| exp | 0x5.8b9111182b4467ep-4 | 0x2.c590e6ab0d71c77p+12 |
| exp10 | 0x1.2da9675e95849c3ep+12 | -0x1.2ab76ac25255a1aap+12 |
| exp2 | -0x7.3f819acf048f1678p-4 | -0x3.fe9a346527a75d98p-16 |
| expm1 | 0x5.8b910bbe3c26818p-4 | -0x1.0040016b56008656p-8 |
| j0 | -0x2.67a2a5d2e367f784p+0 | -0x1.6a09e667f3bd238cp-32 |
| j1 | 0x3.d4eaaeb5ede115p+0 | 0x8.001819d5fc886dap-4 |
| lgamma | -0x3.ec9403f23a1f21cp+0 | -0x4.07fe15510b6a28p+0 |
| log | 0x1.20dad075f537ae56p+0 | 0x1.1001246349edf00cp+0 |
| log10 | 0x1.272b7c3bbb08ae12p+0 | 0x1.010141e1049fce68p+0 |
| log1p | -0x6.451f6c3fd0d4a218p-4 | -0xe.fefa23913fa3eb7p-8 |
| log2 | 0x1.058f12b8b3ac44bep+0 | 0x1.01004bfffe4316bep+0 |
| sin | -0x6.e2368c0ed74e5698p+16 | -0xc.141cf155623856bp+8 |
| sinh | 0x2.c5d375f827733ac4p+12 | 0x7.b0af44fc25df3efp-4 |
| sqrt | 0xf.ffffffffffffffp-4 | 0xf.ffffffffffffffp-4 |
| tan | 0x1.974cd2181086913p+8 | 0xc.845cb771b06f4c5p+0 |
| tanh | 0x3.b9979a543d0fbfa8p-4 | 0x7.fb808a1ef99076ep-4 |
| tgamma | -0x1.70a55b2628a7cb68p+4 | -0x6.9c7abe03c485dbbp+8 |
| y0 | 0xe.4c175c6a0bf51e8p-4 | 0x1.000213a50d97fd8ep+0 |
| y1 | 0xb.bfc89c6a1903022p+0 | 0x4.002362c1b67ad6cp+0 |
| acospi | 0xf.fec85670a4b0761p-4 | |
| asinpi | 0x8.14d7e32b5c44642p-4 | |
| atanpi | 0x8.14d7e32b5c44642p-4 | |
| cospi | 0x7.ae7df669ffba0068p-4 | |
| exp10m1 | 0x2.c7fd02fd98797bf4p-4 | |
| exp2m1 | 0x9.3c1d13fc7c58944p-4 | |
| log10p1 | 0x4.a486e7fa771f839p-4 | |
| log2p1 | 0x5.843d01be597f38fp-4 | |
| sinpi | 0x2.8bea50886e293398p-8 | |
| tanpi | -0x3.a8c5dc49265fd13p+0 | |
| rsqrt | | 0x1.61e30a1ac16221eap+12600 |

Table 13: Double extended precision: inputs giving the largest known error in GNU libc and the Intel Math Library (univariate functions).

| function | GNU libc 2.41<br>$x, y$ | IML 2025.0.0<br>$x, y$ |
|---|---|---|
| atan2 | -0x7.9301460b8463cbp+15368<br>0xf.25cd5eb1280b4d1p+15372 | -0x5.c0c9cc5a59632f88p+16340<br>0x5.db7810fba1ce4908p+16348 |
| atan2pi | 0x4.8010e21a5d13ad38p+212<br>0x5.bb4879ec6325337p+10940 | |
| compound | | 0xa.613dc3f1daa2566p+5432<br>-0x4.fcf34863ee4e7a38p-20 |
| hypot | -0x2.97b86706043d619p+7240<br>0x1.8256bdd12d2e163ep+7240 | -0x3.00bad8a56d87a0cp-16384<br>-0xe.6d794db04791398p-16388 |
| pow | 0x2.21dda4bcec55b158p-3616<br>0x7.ef1ef5fbe3df50dp-16 | 0xc.b80572af668bb57p+152<br>-0x6.8a6d3d7b442f3c18p+4 |

Table 14: Double extended precision: inputs giving the largest known error in GNU libc and the Intel Math Library (bivariate functions).

## References


[1] IEEE Std 754-2019. 2019. *IEEE Standard for Floating-Point Arithmetic*. Institute of Electrical and Electronics Engineers, New York, NY, USA. 84 pages. https://doi.org/10.1109/IEEESTD.2019.8766229

[2] Ramesh C. Agarwal, James W. Cooley, Fred G. Gustavson, James B. Shearer, Gordon Slishman, and Bryant Tuckerman. 1986. New scalar and vector elementary functions for the IBM System/370. *IBM J. Res. Develop.* 30, 2 (1986), 126–144.

[3] AMD. 2024. AMD LibM version 5.0. https://developer.amd.com/amd-aocl/amd-math-library-libm/.

[4] Apple. 2025. Apple Math Library (MacOS 15.1.1, Apple M1).

[5] ARM. 2024. Arm Performance Libraries version 24.04. https://developer.arm.com/downloads/-/arm-performance-libraries.

[6] Jeff M. Arnold. 2016. A Study of the `rsqrt` and `rcp` Instructions on Intel and AMD Platforms. https://github.com/jeff-arnold/math_routines.git. 22 pages.

[7] David H. Bailey. 2020. Variable precision computing: Applications and challenges. Slides presented at the ICERM workshop on Variable Precision in Mathematical and Scientific Computing. https://www.davidhbailey.com/dhbtalks/dhb-icerm-2020.pdf.

[8] Cheryl M. Black, Robert P. Burton, and Thomas H. Miller. 1984. The Need for an Industry Standard of Accuracy for Elementary-Function Programs. *ACM Trans. Math. Software* 10, 4 (1984), 361–366.

[9] Nicolas Brisebarre, Guillaume Hanrot, Jean-Michel Muller, and Paul Zimmermann. 2024. Correctly-rounded evaluation of a function: why, how, and at what cost? (May 2024). https://hal.science/hal-04474530

| function | OpenLibm 0.8.5<br>$x$ | Musl 1.2.5<br>$x$ |
|---|---|---|
| acos | -0x8.040541d0054d89p-4 | 0xf.fe002cabd608585p-4 |
| acosh | 0x1.10384b24aec007fcp+0 | 0x1.1ecdb5b8f0c5d79p+0 |
| asin | 0x8.0519515d1e15a6bp-4 | -0x3.fff0a397b8dea17cp-8 |
| asinh | -0x5.c9866cb231f2c7c8p-4 | -0x8.0bb656992eac437p-4 |
| atan | 0x6.fffde214a06fb5f8p-4 | -0x1.0411ae010d4c5b1ep+0 |
| atanh | -0xf.fffffffffffe78p-32 | 0x3.344a915e34e5e6b8p-4 |
| cbrt | -0x3.fffffffa5623708p+4588 | -0x3.fffffffa5623708p+4588 |
| cos | 0x3.e0db6fa4b23541ap+4 | 0x3.e0db6fa4b23541ap+4 |
| cosh | 0x2.c5d374f9436efd1p+12 | 0x2.c5d37484e4c162bp+12 |
| erf | 0xd.7fe64ab05cf75e8p-4 | 0xd.7fe64ab05cf75e8p-4 |
| erfc | 0x1.5cc0e1cc32a3dc98p+0 | 0x1.5c9262fa4210902p+0 |
| exp | 0x8.aa2253c0d601dedp+0 | -0x2.c5a1073a0f38b61cp+12 |
| exp10 | | 0xd.41cfea690e121b5p+8 |
| exp2 | -0xf.ffffd9f32ee1e06p-12 | -0x7.3f819acf048f1678p-4 |
| expm1 | 0x2.c5c85fdf170c604cp+12 | 0x2.c5c85fdf170c604cp+12 |
| lgamma | -0x2.74ff92c01f0d82acp+0 | -0x2.74ff92c01f0d82acp+0 |
| log | 0xb.504a14384e9b137p-4 | 0x1.20dad075f537ae56p+0 |
| log10 | 0xb.ffac4b4c47e00c3p-4 | 0x1.272b7c3bbb08ae12p+0 |
| log1p | -0x4.c669bd1813ec8bd8p-4 | -0x6.451f6c3fd0d4a218p-4 |
| log2 | 0x1.6646b082fd1065cep+0 | 0x1.058f12b8b3ac44bep+0 |
| sin | -0x2.a2a4a117ff34b034p+8 | -0x2.a2a4a117ff34b034p+81 |
| sinh | -0x2.c5d375cbe7e4a81cp+12 | 0x2.c5c85fdbc1ccc354p+12 |
| sqrt | 0xf.ffffffffffffffp-4 | 0xf.ffffffffffffffp-4 |
| tan | -0x6.fae45244b0bc104p+8 | -0x6.fae45244b0bc104p+8 |
| tanh | 0x3.8b2602d43bdf4c28p-4 | 0x4.024182351388d15p-4 |
| tgamma | -0x6.db747ae147ae148p+8 | -0x2.8d19fd20f3aa62cp+4 |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | 0x3.d34c9d81dcd29354p+5568<br>0xf.3afc4f6c9f5c4a2p+5568 | -0x7.9301460b8463cbp+15368<br>0xf.25cd5eb1280b4d1p+15372 |
| hypot | 0x1.73f339f61eda21dp-16384<br>0x2.e45f9f9500877e2p-16384 | 0x2.00007da75fd5903cp-8960<br>0x2.d42207352184bff4p-8960 |
| pow | 0xf.a795000b7dae5b4p-4<br>-0x7.e4a42355b11a8098p+16 | 0xf.a795000b7dae5b4p-4<br>-0x7.e4a42355b11a8098p+16 |

Table 15: Double extended precision: inputs giving the largest known error in OpenLibm and Musl.

| function | FreeBSD 14.2<br>$x$ |
|---|---|
| acos | -0x8.040541d0054d89p-4 |
| acosh | 0x1.0001fe8811e16ee2p+0 |
| asin | 0x8.0519515d1e15a6bp-4 |
| asinh | -0x8.4d30725ad98215ap-4 |
| atan | 0x6.fffde214a06fb5f8p-4 |
| atanh | -0x7.ff5cbca0e5646c78p-12 |
| cbrt | -0x3.fffffffa5623708p+4588 |
| cos | 0x3.e0db6fa4b23541ap+4 |
| cosh | -0xf.c52a6a14a334e77p-4 |
| erf | 0x1.c5a9ec676d7e551ep-24 |
| erfc | 0x3.fffffff7fffffc34p-36 |
| exp | -0x2.c5b2c28ca01620dcp+12 |
| exp2 | -0x3.ffe0d002661900a8p+12 |
| expm1 | 0x3.80e4f0677c553158p-4 |
| lgamma | -0x2.74ff92c01f0d82acp+0 |
| log | 0x1.01007581714f057ap+0 |
| log10 | 0x1.0101ff27167c589ap+0 |
| log1p | 0x1.0071fc209b87c88p-8 |
| log2 | 0x1.0101e3c5cde465ccp+0 |
| sin | -0x2.a2a4a117ff34b034p+8 |
| sinh | 0xd.ae04a309c0411f3p-4 |
| sqrt | 0xf.ffffffffffffffp-4 |
| tan | -0x6.fae45244b0bc104p+8 |
| tanh | 0x1.80371de2d031a66ep+0 |
| tgamma | -0x6.e000000000000008p+8 |
| cospi | -0x3.f25066fd5ea7265p-4 |
| sinpi | -0x4.019c9d05624aa028p-4 |
| tanpi | -0x1.7fffffffd731f108p+0 |

| function | $x, y$ |
|---|---|
| atan2 | 0x3.d34c9d81dcd29354p+5568<br>0xf.3afc4f6c9f5c4a2p+5568 |
| hypot | 0x1.73f339f61eda21dp-16384<br>0x2.e45f9f9500877e2p-16384 |
| pow | 0xf.a795000b7dae5b4p-4<br>-0x7.e4a42355b11a8098p+16 |

Table 16: Double extended precision: inputs giving the largest known error in FreeBSD.

| library<br>version | GNU libc<br>2.41 | Intel Math Library<br>IML 2025.0.0 |
|---|---|---|
| acos | 1.28 | **0.502** |
| acosh | 4.00 | **0.501** |
| asin | 1.20 | **0.502** |
| asinh | 3.95 | **0.501** |
| atan | 1.41 | **0.501** |
| atanh | 3.89 | **0.501** |
| cbrt | 0.736 | **0.501** |
| cos | 1.52 | **0.501** |
| cosh | 1.92 | **0.501** |
| erf | 1.42 | **0.501** |
| erfc | 4.38 | **0.504** |
| exp | 0.751 | **0.501** |
| exp10 | 2.00 | **0.501** |
| exp2 | 1.08 | **0.501** |
| expm1 | 1.64 | **0.501** |
| j0 | 4.10e32 | **2.90e28** |
| j1 | 3.57e33 | **3.33e31** |
| lgamma | **13.0** | 2.79e30 |
| log | 1.05 | **0.501** |
| log10 | 2.01 | **0.501** |
| log1p | 3.51 | **0.501** |
| log2 | 3.31 | **0.501** |
| sin | 1.52 | **0.501** |
| sinh | 2.07 | **0.501** |
| sqrt | **0.500** | **0.500** |
| tan | 1.06 | **0.502** |
| tanh | 2.39 | **0.501** |
| tgamma | **10.7** | 8.20e3 |
| y0 | 1.69e33 | **4.79e27** |
| y1 | 3.47e33 | **1.45e30** |
| acospi | **1.70** | |
| asinpi | **1.86** | |
| atanpi | **2.32** | |
| cospi | **1.79** | |
| exp10m1 | **3.25** | |
| exp2m1 | **2.23** | |
| log10p1 | **3.47** | |
| log2p1 | **3.35** | |
| rsqrt | | **0.501** |
| sinpi | **1.79** | |
| tanpi | **3.03** | |
| atan2 | 1.89 | **0.501** |
| atan2pi | **2.81** | |
| compound | | **0.501** |
| hypot | 0.792 | **0.501** |
| pow | 30.3 | **1.40** |

Table 17: Quadruple precision: Largest known error.

| function | GNU libc 2.41<br>$x$ | IML 2025.0.0<br>$x$ |
|---|---|---|
| acos | 0x9.fdbe71e81d65064f0f24b2602998p-4 | 0xf.f80616c2416bf63c33a739ae3a08p-4 |
| acosh | 0x1.0f97586eba090200118df0902f99p+0 | 0x1.004ae7a1e9d7b621b12baeda616dp+0 |
| asin | 0x7.79659a0b568bad280c8ec7eb8278p-4 | 0x7.ff86cc20db4e6f7fd33ce212282cp-8 |
| asinh | 0x5.a924236647ffb723576b172b52fcp-4 | 0x1.0000f6bea05a0cafd1e775e627d3p-4 |
| atan | 0x3.7ff864717fc99760d470d1a994cp-4 | -0x1.15eb4e54ee6ca35bf8b1764f30d4p+0 |
| atanh | 0x2.c02a24f3472c7840afbd8cfb68bap-4 | -0xd.9fe29c463116c87fa567e436489p-8 |
| cbrt | -0x5.a837d1198a72e5a89695db79896cp-13792 | -0x2.10d29fbb2036d1d7ffdd8bf63184p+10912 |
| cos | -0x3.08db9df46e0cd142071fdec7eb6p+64 | -0x6.081f6e15f81d27ac2a6038eed3bp+2232 |
| cosh | -0x2.c5d376fd225ce5739bef59cb0e16p+12 | -0x2.ba5adc2ddaf3f5466db2cd018394p+4 |
| erf | 0xd.f3a140b19b0e7d0fafae7eec5ebp-4 | 0x5.a5182e2e3fce6963a492839ebb3cp-8 |
| erfc | 0x1.517e84504890cba9f9f65ff93206p+0 | 0x6.0a5ca72c4efcd78f90acc0aefbbp+0 |
| exp | -0x2.c5b323ac8f24d66ed41ee61ab6bap+12 | -0x5.6622c128e27c6a8c991743947adcp-8 |
| exp10 | 0x3.e9d3cc7e0cbdc5bc7fdfc1932fd6p+0 | 0x1.1e2a2ef09a4f66e4d3648a85045bp+12 |
| exp2 | 0x1.fffe69758fd951b5213a6d47be1ap+0 | -0x7.cab667376a3dd98217d7b028adccp-8 |
| expm1 | 0x5.a1195b05aec378d0b236943f4a18p-4 | 0x8.ca3ec068eee81b45c0adcae049ap+4 |
| j0 | -0x8.a75ab6666f64eae68f8eb383dad8p+0 | 0x3.7c3f883498c0d5e0dab7e54a98b2p+4 |
| j1 | -0x1.7059c8d303730c6b82b12d9941b9p+8 | -0x1.7059c8d303730c6b82b12d9941b9p+8 |
| lgamma | -0x3.ec2152452b5eaf0f070d215b3418p+0 | -0x3.24c1b793cb35efb8be699ad3d9bap+0 |
| log | 0xf.d016f49074a9c4fe793af2394278p-4 | 0xc.4806c5e4877bbeb4b44ed03d9f18p-5364 |
| log10 | 0x1.6a291ea0aa11fb374f1df8b3ac6bp+0 | 0x1.9b621e77f399e4a8c1a85a964e94p-12364 |
| log1p | 0x6.a0aed5f6dad05d6ff33ecd883dc8p-4 | -0x6.2611e37be5cf4388865319f859b4p-12 |
| log2 | 0xb.54170d5cfa8fd72a47d6bda19068p-4 | 0xf.f63cee8e97ac6783532625273eap-4 |
| sin | 0x5.6a5005df151cc2274e119666a9c8p+64 | 0x4.246e3c1f1094e4159999f13cff24p+5604 |
| sinh | 0x6.7e79f3aada38698b910c300b19b8p-4 | -0x1.6606d9c89bc66d481844a8589dcbp+0 |
| sqrt | 0xf.ffffffffffffffffffffffffffff8p-4 | 0xf.ffffffffffffffffffffffffffff8p-4 |
| tan | -0x3.832b771f9462df46117b6a863fa2p+8 | 0xb.eb95e948d6f2a74a1d3a7694bd88p+3816 |
| tanh | -0x3.c26abeca541298cca288adbd1e12p-4 | -0x2.01d7bf6773e2b04acd388c84cd4ep-4 |
| tgamma | -0x1.62ab0823decc5cf957d9a218cf27p+4 | 0x2.00003274fc8659f8ed68e96e0378p-16224 |
| y0 | 0x6.b99c822052e965e1754eb5ffeb08p+4 | 0x3.9561432d16442ec543c74876d1c8p+4 |
| y1 | 0x2.3277da9bfe485c85c35e5bcc806p+0 | 0x2.80bc307275f6a6a3feb2ab211838p+4 |
| acospi | 0xb.50b04f0a7917332097372387f018p-4 | |
| asinpi | -0xa.7ca6c96caefe80b9d757de58a578p-4 | |
| atanpi | 0x3.2d7177f38d6ee175de059ee6394p-4 | |
| cospi | -0x2.5845a017ae769ea1c20e04c7e47cp+28 | |
| exp10m1 | 0xb.2ee9062818e91afe8e80fae1818p-4 | |
| exp2m1 | 0xb.2ee9062818e91afe8e80fae1818p-4 | |
| log10p1 | -0x6.eec527c6a8d6e31ca9f0dcdd747cp-4 | |
| log2p1 | -0x7.fff3a57fdd2666bcecba33bd89f4p-4 | |
| rsqrt | | 0x1.00db76159f986d3a3614199fd36fp-188 |
| sinpi | -0x1.45f52c8be3999ba924f1f3e5a7b7p-8 | |
| tanpi | -0x1.a8c117800f4a648c213cd0b6db1fp+0 | |

| function | $x, y$ | $x, y$ |
|---|---|---|
| atan2 | 0x1.41df5aa214612c7e019fa6ade88p-13316<br>0x5.e53b26a270a29eb9f77ef8ef7af8p-13316 | -0x1.fb41ff205f5ade930a9fcbba8ea8p-16384<br>0x2.23f098fd6b8799dbeb03219bfa08p-10520 |
| atan2pi | -0x2.c6994d7f40fae88117550b428404p+16040<br>0xe.181a2c7696ba5cf23a6eab680b2p+16040 | |
| compound | | 0x2.660c17a54ded3960195d40db8ee4p-11652<br>0xd.45349a19be26b49582eb8c9bb588p+11660 |
| hypot | -0x1.80e7403e1b344c4a78edeced92e4p-16384<br>-0x2.986c750d01c32e4c807c12ad685p-16384 | 0x8.79ec30b61f9b839fe507bbdf414p-11908<br>0xb.94f6832f64d0729ebd68035ed7a8p-11908 |
| pow | 0x1.364dcbbad0512d7bacaae2a8d56bp+0<br>-0xe.68759219434c37725fdf30d17d2p+12 | 0x4p-16496<br>0x3.ffffff39c102f0aa11bb2c8a91dp-128 |

Table 18: Quadruple precision: inputs giving the largest known error in GNU libc and the Intel Math Library.

[10] Dr. Martyn J. Corden and David Kreitzer. 2010. Consistency of Floating-Point Results using the Intel(r) Compiler or Why doesn't my application always give the same answer? https://rc.dartmouth.edu/wp-content/uploads/2018/03/FP_Consistency_12.pdf.

[11] NVIDIA Corporation. 2025. CUDA C Programming Guide v12.8, Section H Mathematical Functions. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#mathematical-functions-appendix

[12] Advanced Micro Devices, Inc. 2025. AMD ROCm Software and Libraries. https://github.com/RadeonOpenCompute/ROCm. https://github.com/ROCm/llvm-project/blob/amd-staging/amd/device-libs/doc/OCML.md.

[13] Warren Ferguson, Marius Cornea, Cristina Anderson, and Eric Schneider. 2015. The Difference Between x87 Instructions FSIN, FCOS, FSINCOS, and FPTAN and Mathematical Functions sin, cos, sincos, and tan. https://software.intel.com/content/dam/develop/external/us/en/documents/x87trigonometricinstructionsvsmathfunctions.pdf.

[14] Free Software Foundation, Inc. 2024. GNU libc: Known Maximum Errors in Math Functions. http://www.gnu.org/software/libc/manual/html_node/Errors-in-Math-Functions.html

[15] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. 2007. MPFR: A Multiple-Precision Binary Floating-Point Library With Correct Rounding. *ACM Trans. Math. Software* 33, 2 (2007), article 13.

[16] FreeBSD. 2024. FreeBSD libc version 14.2. https://www.freebsd.org/releases/14.1R/announce.

[17] GCC. 2025. GCC Online Manual. https://gcc.gnu.org/onlinedocs/gcc/Floating-point-implementation.html.

[18] Tristan Glatard, Lindsay B. Lewis, Rafael Ferreira da Silva, Reza Adalat, Natacha Beck, Claude Lepage, Pierre Rioux, Marc-Etienne Rousseau, Tarek Sherif, Ewa Deelman, Najmeh Khalili-Mahani, and Alan C. Evans. 2014. Reproducibility of neuroimaging analyses across operating systems. *Frontiers Neuroinformatics* 9 (2014), 12. https://doi.org/10.3389/fninf.2015.00012

[19] GLIBC. 2025. GNU libc version 2.41. https://www.gnu.org/software/libc/.

[20] Intel. 2025. Intel Math Library. Distributed with the Intel oneAPI DPC++ Compiler 2025.0.0.

[21] Soonho Kong, Sicun Gao, and Edmund M. Clarke. 2013. *Floating-point Bugs in Embedded GNU C Library.* Technical Report CMU-CS-13-130. Carnegie Mellon University. Available at http://reports-archive.adm.cs.cmu.edu/anon/2013/CMU-CS-13-130.pdf.

[22] Wonyeol Lee, Rahul Sharma, and Alex Aiken. 2017. On automatically proving the correctness of math.h implementations. In *Proceedings of the ACM on Programming Languages (POPL).* 41:1–47:32. https://doi.org/10.1145/3158135.

[23] LLVM. 2025. LLVM-libc C Standard Library 19.1.7. https://github.com/llvm/llvm-project/releases.

[24] Microsoft. 2022. Microsoft Visual Studio 2022.

[25] Jean-Michel Muller. 2005. *On the definition of ulp(x)*. Research Report RR-5504, LIP RR-2005-09. INRIA, LIP. 16 pages. https://hal.inria.fr/inria-00070503

[26] MUSL. 2024. Musl version 1.2.5. https://musl.libc.org/.

[27] Nvidia. 2025. CUDA Math Library. https://developer.nvidia.com/cuda-math-library.

[28] OpenCL. 2024. OpenCL Relative Error as ULPs. https://registry.khronos.org/OpenCL/specs/2.2/html/OpenCL_C.html#relative-error-as-ulps.

[29] OpenLibm. 2025. OpenLibm version 0.8.5. https://openlibm.org/.

[30] Max Petzold. 2000. *A note on the first moment of extreme order statistics from the normal distribution*. Technical Report. Göteborg University. School of Business, Economics and Law. 6 pages, https://gupea.ub.gu.se/handle/2077/3092.

[31] RedHat. 2024. RedHat Newlib version 4.5.0. https://sourceware.org/newlib/.

[32] Alexei Sibidanov, Paul Zimmermann, and Stéphane Glondu. 2022. The CORE-MATH Project. In *ARITH 2022 - 29th IEEE Symposium on Computer Arithmetic*. virtual, France, 26–34. https://hal.inria.fr/hal-03721525