CORE-MATH: 2024 progress report

Paul Zimmermann

MPFR/MPC/MPFI/ARB Developers Meeting Bordeaux 2024, June 17

CORE-MATH Mission

CORE-MATH Mission: provide on-the-shelf open-source mathematical functions with **correct rounding** that can be integrated into current mathematical libraries (GNU libc, Intel Math Library, AMD Libm, Newlib, OpenLibm, Musl, Apple Libm, llvm-libc, CUDA libm, ROCm)

Cf `https://core-math.gitlabpages.inria.fr/`

CORE-MATH license: MIT to make integration easier.

Correct rounding

The best possible result for a given rounding mode.

Already required by IEEE 754 for basic arithmetic (addition, subtraction, multiplication, division, square root, fused multiply-add).

CORE-MATH secret goal: implement correct rounding for mathematical functions, as efficient (or even faster) than current mathematical libraries, so that these libraries integrate the CORE-MATH code, and users of these libraries get their computations correctly rounded (thus reproducible).

Current mathematical libraries

	`binary32`	`binary64`	`binary80`	`binary128`
GNU libc	√	√	√	√
Intel Math Library	√	√	√	√
AMD Libm	√	√		
Redhat Newlib	√	√		
OpenLibm	√	√	√	
Musl	√	√	√	
Apple libm	√	√		
LLVM libm	√	√		
Microsoft libm	√	√		
FreeBSD	√	√	√	
ARM PL	√	√		
CUDA	√	√		
ROCm	√	√		

Largest Known Errors (binary32)

Excluding the gamma, lgamma and Bessel functions, and infinite values:

GNU libc 2.39	3.13 (erfc)
Intel Math Library 2024.0.2	1.00 (tanpi)
AMD Libm 4.1	1.56 (pow)
Redhat Newlib 4.4.0	169 (pow)
OpenLibm 0.8.1	3.17 (erfc)
Musl 1.2.4	3.88 (exp10)
Apple libm 14.0	0.862 (cos)
LLVM libm 17.0.6	0.5
Microsoft libm 2022	6.66 (erfc)
FreeBSD 14.0	3.18 (erfc)
ArmPL 23.10	3.57 (asinh)
CUDA 12.2.1	4.49 (erfc)
ROCm 5.7.0	3.33 (erfc)

Source: https://members.loria.fr/PZimmermann/papers/accuracy.pdf

Largest Known Errors (binary64)

Excluding the gamma, lgamma and Bessel functions, and infinite values:

GNU libc 2.39	5.19 (erfc)
Intel Math Library 2024.0.0	1.73 (pow)
AMD Libm 4.1	1.85 (cosh)
Redhat Newlib 4.4.0	4.08 (erfc)
OpenLibm 0.8.1	636 (pow)
Musl 1.2.4	4.14 (exp10)
Apple libm 14.0	10.7 (erfc)
LLVM libm 17.0.6	0.5
Microsoft libm 2022	91.3 (pow)
FreeBSD 14.0	636 (pow)
ArmPL 23.10	3.00 (atanh)
CUDA 12.2.1	4.51 (erfc)
ROCm 5.7.0	4.08 (erfc)

Source: https://members.loria.fr/PZimmermann/papers/accuracy.pdf

Which functions?

Target the 39 functions defined in IEEE 754-2019 (C99 + C23 functions):

- `exp`, `expm1`, `exp2`, `exp2m1`, `exp10`, `exp10m1`;
- `log`, `log2`, `log10`, `logp1`, `log2p1`, `log10p1`;
- `hypot`;
- `rSqrt`;
- `compound`;
- `rootn`, `pown`, `pow`, `powr`;
- `sin`, `cos`, `tan`, `sinPi`, `cosPi`, `tanPi`;
- `asin`, `acos`, `atan`, `asinPi`, `acosPi`, `atanPi`;
- `atan2`, `atan2Pi`;
- `sinh`, `cosh`, `tanh`;
- `asinh`, `acosh`, `atanh`.

Which target formats

- single precision (`binary32`);
- double precision (`binary64`);
- extended double precision (`long double` on x86_64);
- quadruple precision (`binary128`).

Year 2024 status

C99 and C23 binary32 and binary64 functions implemented: `acos`, `acosh`, `acospi`, `asin`, `asinh`, `asinpi`, `atan`, `atan2`, `atan2pi`, `atanh`, `atanpi`, `cbrt`, `cos`, `cosh`, `cospi`, `erf`, `erfc`, `exp`, `exp10`, `exp10m1`, `exp2`, `exp2m1`, `expm1`, `hypot`, `log`, `log10`, `log10p1`, `log1p`, `log2`, `log2p1`, `pow`, `rsqrt`, `sin`, `sinh`, `sinpi`, `tan`, `tanh`, `tanpi`.

Plus two binary32 functions: lgammaf, tgammaf.

And two long double functions: cbrtl, exp2l (log2l in review, expl to be implemented by Sélène Corbineau).

For some C23 functions, CORE-MATH is the only library providing them so far (partial support in Intel library, FreeBSD, CUDA and RoCM).

Inexact exception

Recently, we added support for the inexact exception.

When $f(x)$ is inexact, the inexact flag is set.

When $f(x)$ is exact, the inexact flag is unchanged (no "spurious" exception). Example: $x = 10^k$ for `log10`, several inputs for `cbrt`, `pow`, `hypot`, ...

And what about speed?

Reciprocal throughput (in cycles of Intel Xeon Silver 4214) of some CORE-MATH routines (cf `https://core-math.gitlabpages.inria.fr/`):

function	CORE-MATH 2024	GNU libc 2.37	Intel Math Library 2023.2.0	LLVM d099dbb
`acosf`	**15**	*25*	*20*	23
`asinf`	**17**	*26*	*19*	19
`asinhf`	17	*33*	***15***	18
`erfcf`	**23**	*54*	*36*	
`exp`	16	*13*	***10***	24
`pow`	53	*42*	***34***	
`cbrtl`	**54**	*140*		
`expll`	**63**	*82*		

Timings produced in our test environment (might be different in other conditions).

And what about bugs?

Univariate binary32 functions are tested by exhaustive search on all possible inputs, with all 4 rounding modes.

The error analysis of most binary64 functions is detailed in the code, with support of Sage functions.

The generation of worst-cases is done using BaCSel (modulo bugs of BaCSel).

Laurence Rideau and Laurent Théry (Inria Sophia-Antipolis) did a full formal proof of the fast path of the binary64 power function with Coq.

Guillaume Melquiond and Paul Geneau de Lamarlière (Inria Saclay) have formally proven a part of the argument reduction of the CORE-MATH binary64 logarithm.

How can I contribute?

- find a bug in the published functions and report it to the CORE-MATH mailing list
- submit a faster correctly-rounded implementation
- find hard-to-round cases for binary64, binary80 or binary128
- make a formal proof of some implementation
- work on the integration in some mathematical library

References

`https://core-math.gitlabpages.inria.fr/`: main page

`https://gitlab.inria.fr/core-math/core-math/`: source code

`https://sympa.inria.fr/sympa/arc/core-math/`: mailing list