Accuracy of Mathematical Functions in Single Precision

Paul Zimmermann

February 3rd, 2020 (revised February 4, May 10 and May 26)

The IEEE 754 single precision (`binary32`) format has $2^{32} - 2^{24} = 4278190080$ values, not counting +Inf, -Inf, and NaN values. For a function with a single input—i.e., excluding the `pow` function for example—it is possible to check all values by exhaustive search, and to compare them to the GNU MPFR library, which guarantees correct rounding (we only consider rounding to nearest here). We have checked the accuracy of all single-precision functions from the GNU libc 2.31, and for the Intel Math Library shipped with the Intel compiler (icc) 19.0.4.243. For each function, assuming e is the difference in ulps (of the correctly-rounded result) against the correctly-rounded result obtained with GNU MPFR, we indicate the number of inputs with $e \geq 1$, with $e \geq 2$, and the maximum value of e.

Note: while doing this comparison, we noticed that we can get different results on two different x86_64 processors, both with GNU libc 2.31 and the same compiler (GCC 9.2.1) and operating system (Debian 9.2.1-25). For example, the `expf` function for $x = 3.256463242e+01$ gives $1.388801499e+14$ on an Intel Core i5-4590 or Intel Xeon E7-4850, but gives $x = 1.388801415e+14$ on a Xeon E7540 (which is the correctly rounded result). The reason seems to be that the Intel Core i5-4590 uses the fma-version of `expf`, whereas the Intel Xeon E7-4850 uses the sse2-version.

The results for the GNU libc were obtained on a Xeon E7-4850, with GCC 8.3.0 under Debian 8.3.0-6. Those for the Intel Math Library were obtained on an AMD EPYC 7702, with icc version 19.0.4.243 (gcc version 9.2.0 compatibility).

We see the sqrt function is correctly rounded for all binary32 inputs, for both the GNU libc and the Intel Math Library. The Intel Math Library gives more accurate results than the GNU libc, except for exp10, where the GNU libc returns an incorrectly rounded result for one input only, namely $x = -10.43461704$, and for the exp function.

The j0, j1, y0, and y1 functions give huge errors for the GNU libc, this is known, see bugs #14469, #14470, #14471, and #14472 on bugzilla.

	GNU libc 3.21			icc 19.0.4.243		
function	$e \geq 1$	$e \geq 2$	max e	$e \geq 1$	$e \geq 2$	max e
acos	5422146	0	1	65950	0	1
acosh	243413455	2698	2	283	0	1
asin	4581700	0	1	469988	0	1
asinh	619608176	2748	2	963558	0	1
atan	21089464	0	1	505178	0	1
atanh	52062348	5790	2	240154	0	1
cbrt	453492162	0	1	15279372	0	1
cos	28209642	0	1	15732888	0	1
cosh	17868534	3558	2	139708	0	1
erf	126805016	0	1	908674	0	1
erfc	20494449	302363	3	1761	0	1
exp	170648	0	1	250299	0	1
exp10	1	0	1	386017	0	1
exp2	168362	0	1	717434	0	1
expm1	12920601	0	1	539655	0	1
j0	1353797232	310998452	4760682496	11960	0	1
j1	1369557306	337817680	714456064	11628	2	2
lgamma	500354453	8246657	7	100287	0	1
log	416908	0	1	1045	0	1
log10	29787060	62225	2	151074	0	1
log1p	11534111	0	1	255701	0	1
log2	313550	0	1	276	0	1
sin	29362812	0	1	12374252	0	1
sinh	71328448	34776	2	247226	0	1
sqrt	0	0	0	0	0	0
tan	83411250	0	1	694770	0	1
tanh	118674314	729782	2	164068	0	1
tgamma	209259574	20924067	8	3971282	0	1
y0	1314115311	207848357	15117099008	6785	1	3
y1	1213975420	177569092	464033280	10384	1	2