

Computing hard-to-round cases of gamma and lgamma for binary64

Paul Zimmermann

December 2025

This note explains how we computed the hard-to-round cases of the gamma function for the binary64 format. We are looking for hard-to-round cases with at least 42 identical bits after the round bit.

1 Case $x \geq 1$

For $x \geq x_0 := 0x1.573fae561f648p+7$, $\Gamma(x) > 2^{1024}$, thus we can restrict to $x < x_0$. We first tried using BaCSeL with a degree-1 approximation of the gamma function, but it was too slow, especially for large x . For example, on a 48-core machine, it took 52 minutes of real time to check the interval [23.598, 23.611]. To use a degree-2 approximation in BaCSeL, we had to implement the trigamma function in GNU MPFR, which reduces the above time to less than 8 minutes. We completed this search using the EXPLOR computing center, and found 14,988 hard-to-round cases, including 23 exact cases (the integers 1...23). From the non-exact values, the worst case in this range is $x = 0x1.676921a72fecfp+6$, with 55 identical bits after the round bit.

2 Case $0 < x < 1$

Lemma 1 *For $0 < x \leq 2^{-106}$, $\Gamma(x)$ rounds to the same value as $1/x$ in the binary64 format (to nearest).*

Proof: We first notice that for $0 < x < 1$, we have

$$1/x - 0.578 < \Gamma(x) < 1/x.$$

If x is a power of 2, then $y = \text{RN}(1/x) = 1/x$, thus $|y - \Gamma(x)| < 1 < \frac{1}{2}\text{ulp}(y)$, and $\Gamma(x)$ rounds to y to nearest. Now assume x is not a power of 2, with $0 < x < 2^{-106}$, and let $y = \text{RN}(1/x)$. We know from [2] that if x is a n -bit number, the longest runs of zeros or ones in $1/x$ have length $n - 1$ (if $1/x$ is not exact). Since $y = \text{RN}(1/x)$, we have $|y - 1/x| \leq \frac{1}{2}\text{ulp}(1/x)$. Since we can have at most 52 identical bits after the upper 54 bits of $1/x$, we deduce

that $1/x$ is at distance at least $2^{-54}\text{ulp}(1/x) \geq 1$ from a rounding boundary. This implies $|y - 1/x| \leq \frac{1}{2}\text{ulp}(1/x) - 1$, and in turn $|y - \Gamma(x)| < \frac{1}{2}\text{ulp}(1/x) \leq \frac{1}{2}\text{ulp}(y)$, thus y is the rounding to nearest of $\Gamma(x)$. ■

The proof of this lemma extends to an n -bit format: for $0 < x \leq 2^{-2n}$, $\Gamma(x)$ rounds to the same value as $1/x$. For the binary64 number $x_1 := 0x1.fffffffffffffp-106$, which is about twice larger than the lemma threshold, $\Gamma(x_1)$ does not round to the same value as $1/x_1$. Computing with BaCSeL hard-to-round cases with at least 51 identical bits after the round bit in the binade $[2^{-106}, 2^{-105})$, we obtain 120 inputs; $\Gamma(x)$ and $1/x$ round to the same value (to nearest) for all of them, except for $0x1.fffffffffffffp-106$. For inputs not found by BaCSeL, we have less than 51 identical bits after the round bit, thus with the notations from the above proof, $1/x$ is at distance at least $2^{-52}\text{ulp}(1/x) \geq 2$ from a rounding boundary, which implies $|y - 1/x| \leq \frac{1}{2}\text{ulp}(1/x) - 2$, and in turn $|y - \Gamma(x)| < \frac{1}{2}\text{ulp}(y)$. Thus the smallest positive value for which $\Gamma(x)$ and $1/x$ round to a different value (to nearest) is $0x1.fffffffffffffp-106$. Thanks to Lemma 1, we can restrict the search with BaCSeL in this case to $2^{-106} \leq x < 1$, which corresponds to 106 binades.

Corollary 1 *For x a binary64 number, $0 < x < 0x1.fffffffffffffp-106$, $\Gamma(x)$ and $1/x$ round to the same value to nearest and with unbounded exponent range, where $\Gamma(x) > 2^{1024}(1 - 2^{-54})$ iff $x \leq 2^{-1024}$.*

We performed a search with BaCSeL in the range $[2^{-106}, 1)$, which found 256,898 hard-to-round cases, with a maximum of 62 identical bits after the round bit, obtained for $x = 0x1.bb2278c9b2f97p-105$ and $x = 0x1.e1d6dce30863ap-32$.

3 Case $-1 < x < 0$

For $-0.3 < x < 0$, we have $1/x - 1 < \Gamma(x) < 1/x$, thus using the same argument as in Lemma 1, we see that for $|x| < 2^{-106}$, $\Gamma(x)$ rounds to the same value as $1/x$ to nearest. Like for $0 < x < 1$, it thus remains 106 binades to check in this case: $-1 < x \leq -2^{-106}$. We performed a search with BaCSeL in the range $(-1, -2^{-106}]$, which found 251,955 hard-to-round cases, with a maximum of 63 identical bits after the round bit, obtained for $x = -0x1.86624b284baf5p-16$.

4 Case $-2^{52} < x < -184$

First when $x < -2^{52}$, $\text{ulp}(x) \geq 1$, thus x is an integer (which also holds for $x = -2^{52}$). Since $\Gamma(x)$ is undefined for x integer, this case is easy.

Now assume $-2^{52} < x < -184$. Let n be a negative integer. In the range $(n, n+1)$, when n is even, $\Gamma(x)$ decreases from $+\infty$ to a positive value, then increases to $+\infty$; when n is odd, $\Gamma(x)$ increases from $-\infty$ to a negative value, then decreases to $-\infty$. Since $\Gamma(x)$ is undefined for n and $n+1$, the largest absolute values of $\Gamma(x)$ are thus obtained for $\text{nextabove}(n)$ and $\text{nextbelow}(n+1)$. Moreover, these largest absolute values are increasing with $|n|$. For

$n = -185$, we find that the largest absolute values are respectively $\approx 8.53 \cdot 10^{-328}$ and $\approx 1.58 \cdot 10^{-325}$, both smaller than 2^{-1075} .

Corollary 2 *For any non-integer $x < -184$, $\Gamma(x)$ underflows.*

5 Case $-184 < x < -1$

For $-184 \leq n \leq -178$, the smallest absolute value of $\Gamma(x)$ over $(n, n + 1)$ is smaller than 2^{-1075} , thus we split the search into $(n, x_1]$ and $[x_2, n + 1)$, where x_1 (resp. x_2) is the largest (resp. smallest) binary64 number such that $|\Gamma(x)| \geq 2^{-1075}$ over $(n, x_1]$ (resp. $[x_2, n + 1)$).

In this range, the SLZ algorithm used by BaCSeL fails near integers. The reason is that the higher derivatives of Γ are much larger in absolute value than Γ itself. Thus if we use the degree- d approximation given by the Taylor expansion around some x_0 , the error term of degree $d + 1$ is large, and only allows small intervals to be checked at once (see Fig. 1).

d	1	2	3	4	5	6
T	1	2,547	91,804	788,649	3,307,983	9,211,582

Figure 1: For Taylor approximations of degree d around $x_0 = -178 - 2^{-13}$, largest interval T in term of $\text{ulp}(x_0)$ such that the mathematical error is bounded by $2^{-10}\text{ulp}(\Gamma(x_0))$.

By the explicit Lagrange theorem, we have for a binary64 number x_0 and an integer i , $0 \leq i < t$:

$$\Gamma(x_0 + iu) = a + ib + i^2c + i^3d,$$

where $u = \text{ulp}(x_0)$, $a = \Gamma(x_0)$, $b = u\Gamma'(x_0)$, $c = u^2\Gamma''(x_0)/2$, and $d = u^3\Gamma^{(3)}(\xi)/6$ for $\xi \in [x_0, x_0 + tu]$. From a bound on $|\Gamma^{(3)}|$ in $[x_0, x_0 + tu]$, we deduce a bound ε_0 on the term $|i^3d|$. If we want that $\Gamma(x_0 + iu)$ is within say at most 2^{-16} ulps from a rounding boundary, this gives another bound ε_1 . With $\varepsilon = \varepsilon_0 + \varepsilon_1$, using the table of difference method already used in [1], we can update the approximation of $\Gamma(x_0 + iu)$ from i to $i + 1$ in only two additions. Only when this approximation is within ε of a rounding boundary, which happens with probability about 2^{-15} , we perform a full check. We have implemented this algorithm using GNU MPFR to compute the coefficients a, b, c , and the Pari/GP library to compute a bound on d (MPFR does not provide the 3rd derivative of the Γ function, and the 2nd derivative is so far implemented in the `trigamma` branch). We found 15,144 hard-to-round cases in that range, with a maximal of 54 identical bits after the round bit, obtained for $x = -0x1.2c0358d14dacep+6$, $x = -0x1.d97de88bda2dfp+5$, and $x = -0x1.5ac06a291806bp+3$.

6 Subnormal output

In the above sections, we did not consider gradual underflow, thus the values produced with either BaCSeL or the algorithm described in §5 might not be real hard-to-round cases for the binary64 format.

For $2^{e-1} \leq |\Gamma(x)| < 2^e$, $-1074 \leq e \leq -1022$, we should consider an output precision of $1021 + 53 + e$ instead of 53, thus ranging from 0 to 52, or from 1 to 53 if we take into account the round bit.

For each value of e , we have only a few ranges $(k, k+1)$ where $|\Gamma(x)|$ crosses $[2^{e-1}, 2^e)$. In total we found 300 sub-ranges to check depending on k and e . We found 26 hard-to-round cases with at least 42 identical bits after the round bit (which has weight 2^{-1075} in the subnormal range). The largest number of identical bits after the round bit is 46, obtained for $x = -0x1.63fd90fb983d2p+7$ in the range $(-178, -177)$, $x = -0x1.5fd732a09e52cp+7 \in (-176, -175)$, and $x = -0x1.57f7b9e290bc3p+7 \in (-172, -171)$. Since this search produced only few inputs, we extended it to numbers with at least 37 identical bits after the round bit. We found 760 additional values, thus a total of 786 inputs with output in the subnormal range.

7 Conclusion

In total we found 539,771 hard-to-round cases, some of which with an unbounded exponent range, and some of which with less than 42 identical bits after the round bit.

8 Hard-to-round cases for lamma

8.1 Case $x > 0$

Contrary to gamma, the lgamma function increases much more slowly, and is defined on almost the entire positive binary64 range. For $x \geq x_1 := 0x1.754d9278b51a8p+1014$, $\text{lgamma}(x) < 2^{1024}$, and for $x < x_1$, $\text{lgamma}(x)$ is smaller than the largest binary64 number.

Near zero, for $0 < x < x_0 := 0x1.762d86356be3fp+0$, $\text{lgamma}(x)$ is decreasing, and for $x > x_0$, $\text{lgamma}(x)$ is increasing. In this range we used BaCSeL with degree 2, using the Pari/GP library like for gamma. We found a total of 4,170,965 hard to round cases with at least 42 identical bits after the round bit. Since this file was quite large to store in the CORE-MATH git repository, we restricted to inputs with at least 44 identical bits after the round bit, this gave 1,042,741 inputs, including the exact cases $x = 1$ and $x = 2$. The worst case in this range is $0x1.129b17eed6bebp+579$ with 64 identical bits after the round bit.

8.2 Case $x < 0$

For $x \leq -2^{52}$, x is an integer, thus $\text{lgamma}(x)$ is not defined. It thus suffices to consider the range $-2^{52} < x < 0$. In this range, we have to consider separately each interval $(k, k+1)$ with k integer. For example, for $-2^{52} < x < -2^{51}$, a binary64 number is either an integer or half an integer: we only have to consider the later case.

Since BaCSeL is not efficient in this case where we have only very few numbers per interval to check, we have to use another method. We use the recurrence formula $\Gamma(x+1) = x\Gamma(x)$, which translates into $\log \Gamma(x+1) = \log \Gamma(x) + \log |x|$ for negative x . Using the fact that

for $|t| < 1/2$, $\log(1+t) = t + \mu t^2$ with $|\mu| < 1$, we obtain for k a positive integer such that $x+k < 0$:

$$\begin{aligned}\log \Gamma(x+k) &= \log \Gamma(x) + \log |x| + \cdots + \log |x+k-1| \\ &= \log \Gamma(x) + \sum_{i=0}^{k-1} (\log |x| + \log(1+i/x)) \\ &= \log \Gamma(x) + k \log |x| + \frac{k(k-1)}{2x} + \varepsilon,\end{aligned}$$

with $|\varepsilon| < k^3/(3x^2)$. We thus get

$$\log \Gamma(x+k) = y_0 + ky_1 + k^2y_2 + \varepsilon,$$

with $y_0 = \log \Gamma(x)$, $y_1 = \log |x| - 1/(2x)$, $y_2 = 1/(2x)$, and $|\varepsilon| < k^3/(3x^2)$. For example, if $z = -2^{52} + 1/2$ and $k = 2^{28}$, the ε term corresponds to a relative error less than 2^{-78} . Within each binade $[-2^{53-k}, -2^{52-k}]$ for $k < 0$, we have $2^k - 1$ arithmetic progressions $z, z+1, z+2, \dots$ that do not hit integer values. For each such arithmetic progression, we get a polynomial approximation as in §5, and similarly we use the table of difference method.

We can use one more term in the Taylor expansion:

$$\begin{aligned}\log \Gamma(x+k) &= \log \Gamma(x) + \log |x| + \cdots + \log |x+k-1| \\ &= \log \Gamma(x) + \sum_{i=0}^{k-1} (\log |x| + \log(1+i/x)) \\ &= \log \Gamma(x) + k \log |x| + \frac{k(k-1)}{2x} - \frac{k(k-1)(2k-1)}{12x^2} + \varepsilon,\end{aligned}$$

with $|\varepsilon| < k^4/(4x^3)$. It follows

$$\log \Gamma(x+k) = y_0 + ky_1 + k^2y_2 + k^3y_3 + \varepsilon,$$

with $y_0 = \log \Gamma(x)$, $y_1 = \log |x| - 1/(2x) - 1/(12x^2)$, $y_2 = 1/(2x) + 1/(4x^2)$, $y_3 = -1/(6x^2)$, and $|\varepsilon| < k^4/(4x^3)$.

However, the above method does not work well for $-2^{23} \leq x < -2^{14}$. In this range, we used a degree-2 Taylor approximation with explicit bound for the remainder, and the table-of-differences method to evaluate this approximation at successive points.

Acknowledgements. Many thanks to Jean-Michel Muller for fruitful discussions and who suggested several nice ideas, and to Bruno Salvy for discussions on how to fill the remaining gap $-2^{23} \leq x < -2^{14}$. This work was made possible by France 2030 project number 22-PECY-0010, and by the use of the EXPLOR computing center from University of Lorraine.

References

- [1] DE DINECHIN, F., MULLER, J., PASCA, B., AND PLESCO, A. An FPGA architecture for solving the table maker's dilemma. In *22nd IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2011, Santa Monica, CA, USA, Sept. 11-14, 2011* (2011), J. R. Cavallaro, M. D. Ercegovic, F. Hannig, P. Ienne, E. E. S. Jr., and A. F. Tenca, Eds., IEEE Computer Society, pp. 187–194.
- [2] LANG, T., AND MULLER, J.-M. Bounds on runs of zeros and ones for algebraic functions. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic* (2001), IEEE Computer Society, pp. 13–20.