The CORE-MATH tanh is correctly rounded

Cyprien Peignier and Guillaume Melquiond and Paul Zimmermann
August 2026

We consider the CORE-MATH file `tanh.c` from commit `72edf106` [1]. Let $\circ$ denote any of the four IEEE754 rounding modes: round to nearest, towards zero, up or down.

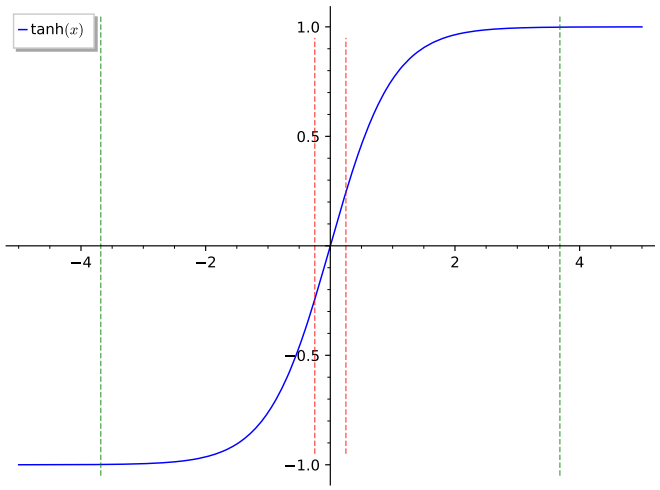


Fig. 1: Evaluation of tanh is done along multiple branches, borders are shown in dashed lines.

I. The tiny case $|x| \in [0, x_0]$

If $x = 0$ then the code returns 0 explicitly, and if $x \neq 0$ it returns $\mathrm{FMA}(x, -2^{-55}, x) = \circ(x - 2^{-55}x)$.

Theorem 1: For $|x| \in (0, x_0]$, the result is correctly rounded, i.e., $\circ(\tanh(x)) = \mathrm{FMA}(x, -2^{-55}, x)$.

Proof: Let $x_1 = \sqrt{3} \cdot 2^{-27} \approx$ `0x1.bb67ae8584caap-27` and let x be a double in $(0, x_1]$.
Starting with:

$$x > \tanh(x) > x - \frac{1}{3}x^3,$$

which is easily obtained knowing: $\tanh' = 1 - \tanh^2$.
Then by using $x \leq x_1$:

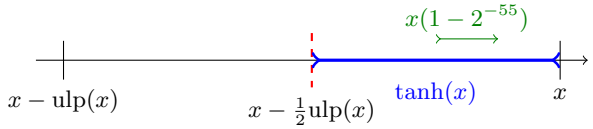
$$x - \frac{1}{3}x^3 \geq x - 2^{-54}x$$

Which means we have:

$$x > \tanh(x) > x(1 - 2^{-54})$$

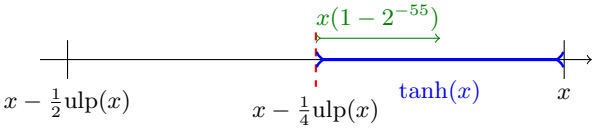
Thus if x is not a power of 2:

$$x > \tanh(x) > x - \frac{1}{2}\,\mathrm{ulp}(x)$$



And if x is a power of 2:

$$x > \tanh(x) > x - \frac{1}{4}\,\mathrm{ulp}(x)$$



By using this reasoning for $|x|$, since tanh is odd and ulp is even, we have proved that the returned value is correctly rounded for $|x| \leq x_1$. Now let x_0 be the smallest binary64 number greater than $\sqrt[3]{\frac{3}{4}} \cdot 2^{-26}$, i.e., $x_0 =$ `0x1.d12ed0af1a27fp-27` and let $x \in (x_1, x_0)$.
We get by using $x < x_0$:

$$x - \frac{1}{3}x^3 > x - 2^{-80}$$

And since $\frac{3}{2} \cdot 2^{-27} < x < 2^{-26}$,

$$\mathrm{ulp}(\tanh(x)) = \mathrm{ulp}(x) = 2^{-79},$$

thus:

$$x > \tanh(x) > x - \frac{1}{2}\,\mathrm{ulp}(x).$$

Multiplying the inequality by -1 gives the result for negative numbers.

Since $\mathrm{ulp}(x) = 2^{-79}$ and $x < 2^{-26}$ for $x \in (x_0, x_1)$, we have $2^{-55}x < 2^{-81} < 2^{-80} = \frac{1}{2}\mathrm{ulp}(x)$, which means $\tanh(x)$ and $(1 - 2^{-55}) \cdot x$ are strictly between two rounding boundaries. Therefore for $|x| \in (0, x_0)$, we have the correct rounding in every rounding mode:

$$\mathrm{FMA}(x, -2^{-55}, x) = \circ(\tanh(x)).$$

Moreover, we can check that for $|x| = x_0$, this holds too. ∎

II. Fast path for $|x| \in (x_0, 1/4)$

The code evaluates a degree-17 polynomial $Q = x + x^3R$ which coefficients were computed using Sollya (version 7.0) to minimize $|(\tanh(x) - Q(x))/x^3|$ with the following script:

```
prec = 128;
d = [0x1p-30,0.25];
R = fpminimax((tanh(x)-x)/x^3,
```

Algorithm 1 FastPath

Require: x a binary64 number such that $|x| \in (x_0, 1/4)$
Ensure: if not FAIL, returns $\circ(\tanh(x))$
1: $c \leftarrow$ coefficients[8]
2: $x_2 \leftarrow \circ(x \cdot x)$
3: $x_3 \leftarrow \circ(x_2 \cdot x)$
4: $x_4 \leftarrow \circ(x_2 \cdot x_2)$
5: $x_8 \leftarrow \circ(x_4 \cdot x_4)$
6: $p_1 \leftarrow \circ(\circ(c[4]+\circ(x_2 \cdot c[5]))+\circ(x_4 \cdot \circ(c[6]+\circ(x_2 \cdot c[7]))))$
7: $p_0 \leftarrow \circ(\circ(c[0]+\circ(x_2 \cdot c[1]))+\circ(x_4 \cdot \circ(c[2]+\circ(x_2 \cdot c[3]))))$
8: $p_0 \leftarrow \circ(p_0 + \circ(x_8 \cdot p_1))$
9: $p_0 \leftarrow \circ(p_0 \cdot x_3)$
10: $r_h, r_\ell \leftarrow \text{fast\_two\_sum}(x, p_0)$
11: $e \leftarrow \circ(x_3 \cdot \texttt{0x1.c0p-52})$
12: $\ell_b \leftarrow r_h + \circ(r_\ell - e)$
13: $u_b \leftarrow r_h + \circ(r_\ell + e)$
14: **if** $\circ(\ell_b) = \circ(u_b)$ **then return** $\circ(\ell_b)$
15: **else return** FAIL
16: **end if**

Algorithm 2 (fast_two_sum)

Require: a, b two binary64 numbers such that $|a| \geq |b|$
Output: $h + \ell$ approximating $a + b$
1: $h \leftarrow \circ(a + b)$
2: $\ell \leftarrow \circ(b - \circ(h - a))$
3: **return** h, ℓ

```
    [|0,2,4,6,8,10,12,14|],
    [|53...|], d, absolute);
```

For $[\texttt{Xmin}, \texttt{Xmax}] \subseteq [x_0, 1/4]$, let s_ℓ and s_u be two numbers such that for $x \in [\texttt{Xmin}, \texttt{Xmax}]$:

$$s_\ell x^3 \leq Q(x) - \tanh(x) \leq s_u x^3. \tag{1}$$

If we are able to prove for all x in $[\texttt{Xmin}, \texttt{Xmax}]$, where $\ell_b(x)$ and $u_b(x)$ are the values computed at lines 12 and 13:

$$\ell_b(x) \leq Q(x) - s_u x^3 \tag{2}$$
$$u_b(x) \geq Q(x) - s_\ell x^3, \tag{3}$$

we get $\ell_b(x) \leq \tanh(x) \leq u_b(x)$ and obtain the result:

$$\circ(\ell_b(x)) = \circ(u_b(x)) \implies \circ(\ell_b(x)) = \circ(\tanh(x)).$$

This means that if the rounding test succeeds, then the returned value is correctly rounded. This Gappa script (we used Gappa 1.8.2) aims at proving Eq. (2):

```
@rnd = float<ieee_64,RND>;

c0 = -0x1.5555555555555p-2;
c1 = 0x1.1111111110f33p-3;
c2 =  -0x1.ba1ba1b9b8ea6p-5;
c3 =  0x1.664f4838e0a43p-6;
c4 =  -0x1.226e17d1bc09bp-7;
c5 =  0x1.d6c64dfba2565p-9;
c6 =  -0x1.7bdd094d327afp-10;
c7 =  0x1.1535ad0c31d0ep-11;

x2 =      x*x;
x2r rnd = x*x;
x4 =      x2 *x2;
x4r rnd = x2r*x2r;
x3 =      x2 *x;
x3r rnd = x2r*x;
x8 =      x4 *x4;
x8r rnd = x4r*x4r;

p1 =      (c4+x2 *c5) + x4 *(c6+x2 *c7);
p1r rnd = (c4+x2r*c5) + x4r*(c6+x2r*c7);

p0 =      (c0+x2 *c1) + x4 *(c2+x2 *c3);
p0r rnd = (c0+x2r*c1) + x4r*(c2+x2r*c3);

p01 =      p0  + x8 *p1;
p01r rnd = p0r + x8r*p1r;

p =      p01 *x3;
pr rnd = p01r*x3r;

q = x + p;

eps = EPS;
er rnd= eps*x3r;

#fast_two_sum
s rnd = x + pr;
l = rnd (-(s-(x+pr)));

lb = s + rnd(l - er);
ub = s + rnd(l + er);

{x in [XMIN, XMAX] /\ (
  |s-(x+pr)| <= 1b-78 /\
  @FIX(s-(x+pr),-132) ->
  |l-(-(s-(x+pr)))| <= 1b-132
  ) ->
(lb-q)+(x3*B) <= 0 /\ pr // x in [-1,1]}

lb - q + x3 * B ->
(rnd(l-er)-(l-er)) + (l-(-(s-(x+pr))))
+ (pr-p) - (er - B * x3);
```

It is used by replacing `EPS` with `0x1.c0p-52` and `B` with s_u on the interval [`XMIN` , `XMAX`].
For Eq. (3), we do the same but replace `B` with s_ℓ and the last few lines with:

```
{x in [XMIN, XMAX] /\ (
  |s-(x+pr)| <= 1b-78 /\
  @FIX(s-(x+pr),-132) ->
  |l-(-(s-(x+pr)))| <= 1b-132
  ) ->
(ub-q)+(x3*B) >= 0 /\ pr // x in [-1,1]}
```

```
ub - q + x3*B ->
(rnd(l+er)-(l+er)) + (l-(-(s-(x+pr))))
+ (pr-p) + (er - ((-B) * x3));
```

We used the fact that the second step of `fast_two_sum` is exact, given the first argument is larger in absolute value than the second, which we have verified by adding the goal `pr // x in [-1,1]`.

The second assumption in the goal is an additional theorem to help Gappa prove the property:

Lemma 1: We use variables defined in the script. If $|s - (x + p_r)| \leq 2^{-78}$ and $s - (x + p_r) \in 2^{-132}\mathbb{Z}$, then

$$|\ell - (-(s - (x + p_r)))| \leq 2^{-132}.$$

Proof: We will use the notation $X = -(s - (x + p_r))$. We start by using $|X| \leq 2^{-78}$, which gives us

$$|\circ(X) - X| < 2^{-131} = 2 \cdot 2^{-132}.$$

Indeed, if $|X| = 2^{-78}$, then the rounding error is 0, else $\mathrm{ulp}(X) \leq 2^{-131}$ and we use the fact that $|\circ(X) - X| < \mathrm{ulp}(X)$ to conclude.

Since X is an integer multiple of 2^{-132}, so is $\circ(X)$ and thus so is $\circ(X) - X$ which we can write as:

$$\circ(X) - X = K \cdot 2^{-132}$$

for K an integer. Using the previous result we get $|K| < 2$, which means:

$$|\circ(X) - X| \leq 2^{132}.$$

And since $\ell = \circ(X)$, we have proved the lemma. ∎

We now need some constants s_ℓ and s_u that both satisfy Eq. (1) and enable Gappa to prove properties Eq. (2) and Eq. (3). They were obtained using Sollya and were checked using the Interval library for Rocq. These bounds can be found in the following table.

[Xmin, Xmax] s_ℓ | s_u
[x0, 0x1.35p-26] 0x1.5555p-56 | 0x1.5556p-56
[0x1.35p-26, 0x1p-15] 0x1.5555p-56 | 0x1.5556p-56
[0x1p-15 , 0x1p-10] 0x1.5519p-56 | 0x1.5556p-56
[0x1p-10 , 0x1p-5] 0x1.2687p-57 | 0x1.551ap-56
[0x1p-5 , 0x1p-2] -0x1.222ep-58 | 0x1.2688p-57

We have successfully used Gappa to prove the correctness of the fast path on $[x_0, 1/4] \setminus I$, where $I = (\texttt{0x1p-26}, \texttt{0x1.0cp-26})$. On the remaining interval I, Gappa fails to prove the property.

Theorem 2: The fast path is also accurate on I.

Proof: Using Gappa we can prove the following for x a binary64 number in I:

$$|\ell_b - Q(x)| < 2^{-127}.$$

And using SageMath we obtain:

$$2^{-80} < x - \tanh(x) < 2^{-79} = \mathrm{ulp}(x)/2,$$

as well as

$$2^{-80} + 2^{-127} < x - Q(x) < 2^{-79} - 2^{-127}.$$

Which means both $\tanh(x)$ and ℓ_b are in the interval $(x - \mathrm{ulp}(x)/2, x)$. Thus for $x > 2^{-26}$,

$$\circ(\ell_b) = \circ(\tanh(x)).$$

We have shown a stronger property than required: on I, ℓ_b is always correctly rounded. ∎

We therefore have proved that the fast path is accurate on $(x_0, 1/4)$. Negative numbers are treated similarly. Sollya's bounds were computed for $(Q(x) - \tanh(x))/x^3$ which is even, and thus remain valid for negative x. We get on $[-\texttt{Xmax}, -\texttt{Xmin}]$:

$$s_u x^3 \leq Q(x) - \tanh(x) \leq s_\ell x^3,$$

i.e., s_u and s_ℓ are swapped. Also the value of e is now negative which means the roles of ℓ_b and u_b are swapped too. Therefore in this case we need to prove:

$$u_b(x) \leq Q(x) - s_\ell x^3$$

and

$$\ell_b(x) \geq Q(x) - s_u x^3.$$

It is done in the exact same manner as the positive case, using a similar Gappa script and the same Sollya bounds on opposite intervals. For $-I$, the Gappa bound: $|\ell_b - Q| < 2^{-127}$ is still valid. Mutiplying the inequalities from the positive case by -1 concludes the proof.

III. Accurate path for $x_0 < |x| < 1/4$

The accurate path is taken whenever the rounding test of the fast path fails in algorithm FastPath. The code evaluates a degree-29 polynomial $P = x + x^3 R(x^2)$ in `double-double` precision: let y_0, y_1 be the result of this computation. We then test if the last 52 bits of the mantissa of y_1 are zero, because that would potentially place $y_0 + y_1$ on a rounding boundary. In this case we adjust y_1 by an ulp using the sign of y_2 after line 10. If x matches a special case—limited to a single value and its opposite—then the return value is hardcoded depending on the rounding mode.

We will see later that we enter the branch at line 11 only when x is a hard-to-round case, and we checked on all known hard-to-round cases that y_1 is never zero in this case.

The coefficients of R are a mix of `double-double` and `double` precision:

```
c0=-0x1.5555555555555p-2-0x1.5555555554cc4p-56;
c1=0x1.1111111111111p-3+0x1.111110f8c0178p-59;
c2=-0x1.ba1ba1ba1ba1cp-5+0x1.7917c1d676ff5p-59;
```

Algorithm 3 AccuratePath

Input: x binary64 in $(x_0, 1/4]$
Output: $y_0 + y_1$ approximating $P(x)$
1: $c_\ell \leftarrow$ coefficients_double[5]
2: $c \leftarrow$ coefficients_double-double[9][2]
3: $x_2 \leftarrow \circ(x \cdot x)$
4: $x_{2\ell} \leftarrow \mathrm{FMA}(x, x, -x_2)$
5: $y_2 \leftarrow x_2 \cdot (c_\ell[0] + x_2 \cdot (c_\ell[1] + x_2 \cdot (c_\ell[2] + x_2 \cdot (c_\ell[3] + x_2 \cdot c_\ell[4]))))$
6: $y_1, y_2 \leftarrow$ polydd$(x_2, x_{2\ell}, 9, c, y_2)$ ▷ $y_1 + y_2 \approx R(x^2)$
7: $y_1, y_2 \leftarrow$ mulddd_accurate(y_1, y_2, x)
8: $y_1, y_2 \leftarrow$ muldd_accurate$(y_1, y_2, x_2, x_{2\ell})$
9: $y_0, y_1 \leftarrow$ fast_two_sum(x, y_1)
10: $y_1, y_2 \leftarrow$ fast_two_sum(y_1, y_2)
11: **if** the last 52 bits of the significand of y_1 are zero **then**
12: **if** $y_2 y_1 \geq 0$ **then**
13: $y_1 \leftarrow \mathrm{next\_away}(y_1)$
14: **else**
15: $y_1 \leftarrow \mathrm{next\_towards\_zero}(y_1)$
16: **end if**
17: **if** $|x|$ is a special case **then**
18: **return** expected value
19: **end if**
20: **end if**
21: **return** $\circ(y_0 + y_1)$

Algorithm 4 (polydd)

Input: $x = x_h + x_\ell$ double-double, n integer, c array of double-double coefficients, r binary64
Output: h, ℓ approximating $\sum_{i=0}^{n-1} c[i]x^i + rx^{n-1}$
1: $h, t \leftarrow$ fast_two_sum$(c[n-1]_h, r)$
2: $\ell \leftarrow \circ(t + c[n-1]_\ell)$
3: for i from $n-2$ downto 0
4: $h, \ell \leftarrow$ muldd_accurate(x_h, x_ℓ, h, ℓ)
5: $h, t \leftarrow$ fast_two_sum$(c[i]_h, h)$
6: $\ell \leftarrow \circ(\circ(\ell + c[i]_\ell) + t)$
7: **return** h, ℓ

Algorithm 5 (muldd_accurate)

Input: x_h, x_ℓ, c_h, c_ℓ binary64 numbers
Output: h, ℓ approximating $(x_h + x_\ell)(c_h + c_\ell)$
1: $h \leftarrow \circ(x_h c_h)$
2: $\ell_1 \leftarrow \mathrm{FMA}(x_h, c_h, -h)$
3: $\ell_2 \leftarrow \circ(x_h c_\ell)$
4: $\ell_3 \leftarrow \circ(x_\ell c_h)$
5: $\ell \leftarrow \circ(\circ(\ell_3 + \ell_2) + \ell_1)$
6: **return** fast_two_sum(h, ℓ)

Algorithm 6 (mulddd_accurate)

Input: x_h, x_ℓ, t binary64 numbers
Output: h, ℓ approximating $(x_h + x_\ell)t$
1: $h \leftarrow \circ(x_h t)$
2: $\ell_1 \leftarrow \mathrm{FMA}(x_h, t, -h)$
3: $\ell_2 \leftarrow \circ(x_\ell t)$
4: $\ell \leftarrow \circ(\ell_1 + \ell_2)$
5: **return** fast_two_sum(h, ℓ)

```
c3=0x1.664f4882c10fap-6-0x1.9d5cb27c0af28p-63;
c4=-0x1.226e355e6c23cp-7-0x1.c9674586913f3p-61;
c5=0x1.d6d3d0e157db3p-9-0x1.71376fa06ce94p-65;
c6=-0x1.7da36452b5e46p-10-0x1.aba8d51bd9cp-65;
c7=0x1.3558247faa32dp-11-0x1.e0cfb423aedfdp-65;
c8=-0x1.f57d76ea30928p-13-0x1.c30601213cae9p-67;
cl0=0x1.967e0a63ca836p-14;
cl1=-0x1.497b99d2a77d1p-15;
cl2=0x1.0ae346258cbdep-16;
cl3=-0x1.aade68fb2f076p-18;
cl4=0x1.22e609bf8671fp-19;
```

They were computed to minimize the relative error $|P(x)/\tanh(x) - 1|$ using the following Sollya script:

```
prec=340;
d=[0x1.d12ed0af1a27fp-27, 0.25];
p=fpminimax(tanh(x),
[| 3, 5, 7, 9,11,13,15,17,19,21,23,25,27,29|],
[|DD,DD,DD,DD,DD,DD,DD,DD,DD, D, D, D, D, D|],
d, relative, x);
supnorm(p, tanh(x), d, relative, 1b-40);
d2 = [-0.25, -0x1.d12ed0af1a27fp-27];
supnorm(p, tanh(x), d2, relative, 1b-40);
```

which also proves for $|x| \in [x_0, 1/4]$:

$$|P(x)/\tanh(x) - 1| < 2^{-109}. \tag{4}$$

Now let Y be $y_0 + y_1$ before its final rounding, as it is just after line 10 of the accurate path. We have used Gappa to prove the following bound for $|x| \in [x_0, 1/4]$:

$$|(Y - P(x))/P(x)| < 2^{-103.3}. \tag{5}$$

To achieve this result we used the fact that the second step of fast_two_sum(a, b) is exact when $|a| \geq |b|$, as well as the fact that ℓ_1 is exactly $x_h c_h - h$ in muldd_accurate and exactly $x_h t - h$ in mulddd_accurate, given $x_h c_h$ and $x_h t$ are integer multiples of 2^{-1074}. These conditions were checked by adding additional Gappa goals.

With the triangle inequality:

$$\begin{aligned}
|Y - \tanh(x)| &\leq |Y - P(x)| + |P(x) - \tanh(x)| \\
&< 2^{-103.3}|P(x)| + 2^{-109}|\tanh(x)| \quad (6)\\
&= 2^{-103.3}P(|x|) + 2^{-109}\tanh(|x|) \quad (7)\\
&= 2^{-103.3}(P(|x|) + 2^{-6}\tanh(|x|)) \\
&< 2^{-103.3}\tanh(|x|)(1 + 2^{-109} + 2^{-6}) \quad (8)\\
&< 2^{-103.2}\tanh(|x|).
\end{aligned}$$

Inequality (6) is obtained using (4) and (5). Equation (7) is valid because both $\tanh$ and P are odd. For (8) we use (4) to

have $P(|x|) < \tanh(|x|)(1+2^{-109})$, and we use the fact that $2^{-103.3}(1+2^{-109}+2^{-6}) \le 2^{-103.2}$.

We have thus proved for $|x| \in (x_0, 1/4)$:

$$|Y - \tanh(x)| < 2^{-103.2}|\tanh(x)|. \tag{9}$$

Theorem 3: Let Y be the value of $y_0 + y_1$ before its final rounding, i.e., after line 10 in Algorithm AccuratePath. For $|x| \in (x_0, 1/4)$, if $y = \tanh(x)$ has less than 48 identical bits after the round bit, then the accurate path is correct, i.e., $\circ(Y) = \circ(\tanh(x))$.

Proof: First let us recall the round bit is the next bit after the unit in last place. The hypothesis thus implies y is at distance at least $2^{-48} \cdot (1/2)\,\mathrm{ulp}(y)$ from any rounding boundary. And using $\mathrm{ulp}(y) > 2^{-53}|y|$, we obtain that the distance between y and any rounding boundary is at least $2^{-102}|y|$. Thus if z is a rounding boundary, using (9):

$$|Y - y| < |y - z|.$$

Therefore Y is closer to $y = \tanh(x)$ than any rounding boundary, which means Y and $\tanh(x)$ are on the same side of any boundary: $\circ(Y) = \circ(\tanh(x))$.

Let Y_f be the value of $y_0 + y_1$ before line 21. If the branch at line 11 has not been taken, then $Y_f = Y$ and the result is correctly rounded by the above argument. Now if the last 52 bits of the significand of y_1 are zero at line 11, then Y_f might slightly differ from Y as there are two cases:

- If $|y_1| \ge (1/2)\,\mathrm{ulp}(y_0)$, then Y has infinitely many zeroes after the round bit, and using (9), y has 49 or more identical bits after the round bit: this case cannot happen under the hypothesis.
- If $|y_1| < (1/2)\,\mathrm{ulp}(y_0)$, then $\mathrm{ulp}(y_1) < 2^{-53}\,\mathrm{ulp}(y_0)$ and because of the properties of the ulp function, $\mathrm{ulp}(y_1) \le 2^{-54}\,\mathrm{ulp}(y_0) \le 2^{-106}|y_0|$. Thus, since $|Y| \ge |y_0| - |y_1| \ge |y_0|/2$, we have $\mathrm{ulp}(y_1) \le 2^{-105}|Y|$. Therefore we have:

$$\begin{aligned}
|Y_f - y| &\le |Y_f - Y| + |Y - y| \\
&< |\,\mathrm{ulp}(y_1)| + 2^{-103.2}|y| && (10)\\
&\le 2^{-105}|Y| + 2^{-103.2}|y| \\
&\le 2^{-105}(1+2^{-103.2})|y| + 2^{-103.2}|y| && (11)\\
&\le 2^{-102}|y|. && (12)
\end{aligned}$$

Here, (10) comes from the adjusting of y_1 of at most $\mathrm{ulp}(y_1)$ like (9), (11) comes from (9) and (12) comes from the fact that $2^{-105}(1+2^{-103.2}) + 2^{-103.2} \le 2^{-102}$.

And thus we have shown $|Y_f - y| < |y - z|$ for every rounding boundary z, which means that adjusting y_1 will not change the rounded result.

We have proved $\circ(Y_f) = \circ(\tanh(x))$, thus the accurate path is correctly rounded if $\tanh(x)$ has less than 48 identical bits after the round bit. ∎

Theorem 4: The accurate path on $(-1/4, -x_0) \cup (x_0, 1/4)$ is correctly rounded.

Proof: Let x be a binary64 number such that $|x| \in (x_0, 1/4)$. If x has 48 or more identical bits after the round bit, then it is present is the test file `tanh.wc` gathering all worst cases. This value of x has therefore been checked in this case. If x has less than 48 identical bits after the round bit, then Theorem 3 proves the correct rounding for x. ∎

IV. Branch $1/4 \le |x| < x_2$

Let $x_2 = \texttt{0x1.d76c8b439581p+1} \approx 3.683$. This branch was tested exhaustively for $x > 0$ and all four rounding modes with and without fma contraction. During this search, a failure was found in the fast path with the original error bound and $x = \pm\texttt{0x1.a0112a16e9318p+1}$ for rounding upward (also for $x = \pm\texttt{-0x1.a0bd10af4ac2bp+1}$ and rounding upward). The error bound was increased to ensure these values are correctly rounded. This exhaustive search ensures both the fast path and the accurate path are correct for this branch.

Since the rounding modes to nearest and toward zero are "symmetric", and the approximation used are either even or odd, the result for $x < 0$ are the same or the opposite respectively. Thus for $x < 0$ we only have to check for rounding upward or downward, which are not "symmetric".

V. Branch $x_2 \le |x| < x_3$

Let $x_3 = \texttt{0x1.30fc1931f09cap+4} \approx 19.06$. For $|x| \ge x_3$, $\tanh(x)$ rounds to ± 1 to nearest. Indeed, let x be a double precision floating-point number greater or equal to x_3. Then $\tanh(x) \ge \tanh(x_3) = (1-e^{-2x_3})/(1+e^{-2x_3})$. Now we can verify using SageMath that $e^{-2x_3} < 2^{-55}$. We therefore have $(1-2^{-55})/(1+2^{-55}) < \tanh(x)$ and it is easy to check (either by hand or using SageMath) that $(1-2^{-55})/(1+2^{-55}) > 1-2^{-54}$, which amounts to

$$1 - \mathrm{ulp}(1)/4 < \tanh(x) < 1.$$

This means that for rounding to nearest : $\circ(\tanh(x)) = 1$, and using the fact that both $\tanh$ is odd and that $\circ(-x) = -\circ(x)$, we have the result for $|x| \ge x_3$.

This branch was tested exhaustively for $x > 0$ and all four rounding modes with and without fma contraction. This exhaustive search ensures both the fast path and the accurate path are correct for this branch.

As with the previous branch, for $x < 0$ it has to be tested exhaustively for rounding upward and downward.

References

[1] Sibidanov, A., Zimmermann, P., and Glondu, S. The CORE-MATH Project. In *ARITH 2022 - 29th IEEE Symposium on Computer Arithmetic* (virtual, France, 2022).