The CORE-MATH acosh is correctly rounded

Paul Zimmermann
June 17, 2026

***Abstract*—We prove that the CORE-MATH binary64 hyperbolic arc-cosine function is correctly rounded.**

***Index Terms*—IEEE 754, binary64 format, correct rounding.**

We consider the CORE-MATH `acosh.c` file from commit `ba60363`. The function $\mathrm{acosh}(x)$ is only defined for $x \geq 1$ (see Figure 1). It is mathematically defined as follows:

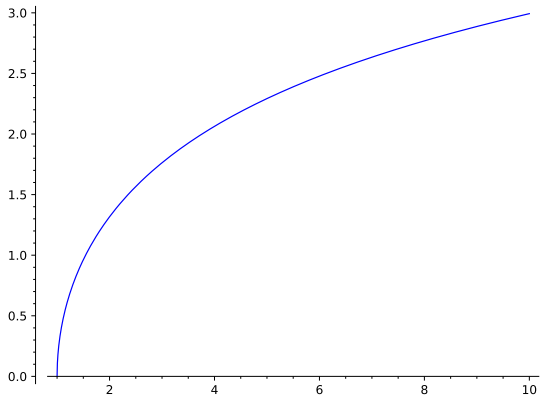

Fig. 1. Graph of $\mathrm{acosh}(x)$ on $[1, 10]$.

$$\mathrm{acosh}(x) = \log(x + \sqrt{x^2 - 1}).$$

This shows that asymptotically, $\mathrm{acos}(x)$ behaves like $\log(2x)$.

The case $x = 1$ is checked separately.

I. Branch $1 < x < x_0$

Let $x_0 =$ `0x1.1e83e425aee63p+0`. For $1 < x < x_0$, CORE-MATH uses a polynomial approximation from the reduced argument $z = x - 1$ (which is exact), and using $\sqrt{2z}$. Since this branch contains only about $2^{48.9}$ inputs, an exhaustive search is possible. During this search, we found that a previous error bound was too small, for example $x =$ `0x1.00a800422847ap+0` was not correctly rounded with rounding toward zero, with/without FMA contraction.

II. Branch $x_0 \leq x < 111.75$

This branch first computes a double-double approximation $t_h + t_\ell$ of $x + \sqrt{x^2 - 1}$, then uses Algorithm FastPath (see §V), with $2x$ at line 3 replaced by t_h, and g at line 17 replaced by $\circ(t_\ell/t_h)$. The main idea why this works is that $\log(t_h + t_\ell) \approx \log(t_h) + t_\ell/t_h$, where Algorithm FastPath precisely approximates $\log(t_h) + g$. We now perform a rigorous error analysis.

Since the error analysis of the approximation of $\log(t_h)$ is already done in §V, it suffices to bound the difference between $\mathrm{acosh}(x)$ and $\log(t_h) + g$. For this, we need to detail how t_h and g are computed. This is done in Algorithm ComputeThG. Note: at lines 5-6, instead of computing i_{sh}

Algorithm 1 (ComputeThG)

Input: x a binary64 number, $x_0 \leq x < 111.75$
Output: binary64 numbers t_h and g such that $\log(t_h) + g$ approximates $\mathrm{acosh}(x)$

1: $x_{2h} \leftarrow \circ(x \cdot x)$
2: $w_h \leftarrow \circ(x_{2h} - 1)$ ▷ exact
3: $w_\ell \leftarrow \mathrm{fma}(x, x, -x_{2h})$ ▷ exact
4: $s_h \leftarrow \circ(\sqrt{w_h})$
5: $i_{sh} \leftarrow \circ(0.5/w_h)$
6: $s_\ell \leftarrow \circ(\circ(w_\ell - \mathrm{fma}(s_h, s_h, -w_h)) \cdot \circ(s_h \cdot i_{sh}))$
7: $t_h, u_\ell \leftarrow \mathrm{fasttwosum}(x, s_h)$
8: $t_\ell \leftarrow \circ(u_\ell + s_\ell)$
9: $g \leftarrow \circ(t_\ell/t_h)$

which approximates $1/(2w_h)$ and multiplying by $s_h i_{sh}$ which approximates $1/(2\sqrt{w_h})$, one could divide by $2s_h$, but this yields a worse reciprocal throughput.

Firstly, since $1 \leq x_{2h} < 2^{14}$, $w_h = \circ(x_{2h} - 1)$ is exact, since x_{2h} can be written $m \cdot 2^e$ with m, e integers, $2^{52} \leq m < 2^{53}$, and $-52 \leq e \leq -39$. Thus $x_{2h} - 1 = m' \cdot 2^e$, where $m' = m - 2^{-e}$ is an integer, and $0 \leq m' < 2^{53}$.

Secondly, $w_\ell = \mathrm{fma}(x, x, -x_{2h})$ is also exact at line 3, since the rounding error of a product is always exactly representable, whatever the rounding mode (error-free transformation or EFT). Thus we have $w_h + w_\ell = x^2 - 1$. Moreover, since w_ℓ represents the rounding error in $x_{2h} = \circ(x^2)$, we have $|w_\ell| < \mathrm{ulp}(x_{2h})$. Since $\mathrm{ulp}(w_h)/\mathrm{ulp}(x_{2h}) \geq 0.25$ (the minimum being attained near x_0), it follows $|w_\ell| < 4\mathrm{ulp}(w_h)$ thus $|w_\ell| < 2^{-50} w_h$.

For rounding to nearest, it is known that $s = \circ(\sqrt{x})$ followed by $t = -\mathrm{fma}(s, s, -x)$ is an error-free transformation (EFT) [2]. However this is no longer true with directed roundings. Consider for example a precision of 3 bits, and $x = 10$. Then $s = \mathrm{RU}(\sqrt{x}) = 3.5$ and $s^2 - x = 2.25$ is not exactly representable on 3 bits. Thus if we denote $t = -\mathrm{fma}(s_h, s_h, -w_h)$ the quantity computed at line 6, we don't always have $s_h^2 + t = w_h$.

Lemma 1: Let x a positive p-bit number, $s = \circ(\sqrt{x})$ and $t = -\mathrm{fma}(s, s, -x)$. Then whatever the IEEE-754 rounding mode:

$$|s^2 + t - x| \leq 2^{1-p}\mathrm{ulp}(x).$$

Proof: Without loss of generality, we can assume $1 \leq x < 4$. Let $u = 2^{-p}$. We first deal separately with the case where x is

the largest binary64 number below 4 and rounding upwards, i.e., $x = 4 - 4u$. Then $s = 2$, $t = -4u$, and $s^2 + t - x = 0$.

Otherwise, we always have $s < 2$, thus $\mathrm{ulp}(s) = 2u$, and $s - 2u < \sqrt{x} < s + 2u$. On the one hand, noting $r = x - s^2$, we have:

$$|r| = |\sqrt{x} + s| \cdot |\sqrt{x} - s| < (2s + 2u) \cdot (2u) = 4su + 4u^2.$$

Since $s \leq 2 - 2u$, this yields $|r| < 8u$. On the other hand, r is a integer multiple of $\mathrm{ulp}(s)^2 = 4u^2$. It follows $r = m \cdot (4u^2)$ with m integer, $|m| < (8u)/(4u^2) = 2^{p+1}$. If m is even, then r is representable on p bits, $t = r$, thus $s^2 + t - x = 0$. If m is odd, then r is not representable on p bits, but $|t - r| = 4u^2 = 2^{1-p} \cdot (2u) \leq 2^{1-p}\mathrm{ulp}(x)$. ∎

This bound is tight and attained. For example in binary32, consider $x =$ `0x1.00962ep+0` and rounding toward zero. Then $s =$ `0x1.004b0ap+0`, $t =$ `0x1.0048e6p-22`, and $|s^2 + t - x| = 2^{-46} = 2^{1-p}\mathrm{ulp}(x)$.

Next, we analyze the difference between $s_h + s_\ell$ and $\sqrt{x^2 - 1}$.

Lemma 2: In Algorithm ComputeThG, let t_h the value computed at line 7, and t_ℓ the value computed at line 8. Then whatever the IEEE 754 rounding mode, we have:

$$|t_h + t_\ell - (x + \sqrt{x^2 - 1})| \leq 2^{-92.939}.$$

Proof: Denote $t = -\mathrm{fma}(s_h, s_h, -w_h)$, and $t' = w_\ell + t$. Let us first bound the difference between $s_h + t'/(2s_h)$ and $\sqrt{x^2 - 1} = \sqrt{w_h + w_\ell}$:

$$\begin{aligned}
&\left| s_h + \frac{t'}{2s_h} - \sqrt{x^2 - 1} \right| = \left| s_h + \frac{t'}{2s_h} - \sqrt{w_h} \cdot \sqrt{1 + \frac{w_\ell}{w_h}} \right| \\
&\leq \left| s_h + \frac{t'}{2s_h} - \left(\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}} \right) \right| \\
&+ \sqrt{w_h} \cdot \left| \sqrt{1 + \frac{w_\ell}{w_h}} - \left(1 + \frac{w_\ell}{2w_h} \right) \right|. \qquad (1)
\end{aligned}$$

Since $|w_\ell| < 4\mathrm{ulp}(w_h)$ as shown previously, the term $|w_\ell/w_h|$ is bounded by $4\mathrm{ulp}(w_h)/w_h$ thus by 2^{-50}. Since $|\sqrt{1+u} - (1 + u/2)| < 0.126 \cdot u^2$ for $|u| \leq 2^{-8}$, using $u = w_\ell/w_h$, the term $|\sqrt{1 + w_\ell/w_h} - (1 + w_\ell/(2w_h))|$ from Eq. (1) is bounded by $0.126(w_\ell/w_h)^2 < 2^{-102.988}$.

Remembering $t' = w_\ell + t$, the first term from the right-hand side of Eq. (1) is bounded as follows:

$$\begin{aligned}
&\left| s_h + \frac{t'}{2s_h} - \left(\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}} \right) \right| \\
&\leq \left| s_h + \frac{t}{2s_h} - \sqrt{w_h} \right| + \frac{|w_\ell|}{2} \cdot \left| \frac{1}{s_h} - \frac{1}{\sqrt{w_h}} \right|. \qquad (2)
\end{aligned}$$

For the first term of the right-hand side from Eq. (2), we have:

$$\left| s_h + \frac{t}{2s_h} - \sqrt{w_h} \right| \leq \frac{|(s_h + t/(2s_h))^2 - w_h|}{s_h + t/(2s_h) + \sqrt{w_h}}. \qquad (3)$$

Applying Lemma 1 with $x = w_h$, $s = s_h$ and $t = t$, the numerator of the right-hand side from Eq. (3) is bounded by $2^{-52}\mathrm{ulp}(w_h) + t^2/(4s_h^2)$. With the notation from the proof of Lemma 1, we see that $t = -\circ(-r)$, thus since $|r| < 8u = 2^{-50}$, we have $|t| \leq 2^{-50}$, and $|t|/s^2 \leq 2^{-50}$ since $1 \leq s \leq 2$. When x in Lemma 1 (thus w_h here) is scaled by 2^{2k}, $s = \circ(\sqrt{x})$ is scaled by 2^k, and $t = -\mathrm{fma}(s, s, -x)$ is scaled by 2^{2k}, thus t/s^2 is invariant, and $t^2/(4s_h^2) \leq 2^{-52}|t|$.

We split the analysis according to x. For $x_0 \leq x < 16$, we have $w_h < 2^8$, $|t| \leq 2^{-44}$, $s_h > 0.502$, thus $2^{-52}\mathrm{ulp}(w_h) + t^2/(4s_h^2) < 2^{-95.415}$, and $s_h + t/(2s_h) + \sqrt{w_h} > 1.005$, thus the ratio is bounded by $2^{-95.415}/1.005 < 2^{-95.422}$. For $16 \leq x < 111.75$, we have $w_h < 12488$, $|t| < 2^{-38}$, $s_h > 15.968$, thus $2^{-52}\mathrm{ulp}(w_h) + t^2/(4s_h^2) < 2^{-89.415}$, and $s_h + t/(2s_h) + \sqrt{w_h} > 31.937$, thus the ratio is bounded by $2^{-89.415}/31.937 < 2^{-94.412}$. We conclude for the bound of Eq. (3):

$$\left| s_h + \frac{t}{2s_h} - \sqrt{w_h} \right| < 2^{-94.412}. \qquad (4)$$

Now:

$$\left| \frac{1}{s_h} - \frac{1}{\sqrt{w_h}} \right| = \left| \frac{\sqrt{w_h} - s_h}{s_h\sqrt{w_h}} \right|.$$

The numerator is bounded by $\mathrm{ulp}(s_h)$ by definition, and since $s_h \geq 0.999\sqrt{w_h}$, the denominator is lower bounded by $0.999w_h$. Since $|w_\ell| < 2^{-50}w_h$ and $s_h < 111.75$, this yields

$$\frac{|w_\ell|}{2} \cdot \left| \frac{1}{s_h} - \frac{1}{\sqrt{w_h}} \right| \leq \frac{2^{-50}}{0.999}\mathrm{ulp}(s_h) < 2^{-95.998}.$$

Together with Eq. (4), using $2^{-94.412} + 2^{-95.998} < 2^{-93.997}$, we find for the bound of Eq. (2):

$$\left| s_h + \frac{t'}{2s_h} - \left(\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}} \right) \right| < 2^{-93.997}.$$

Now since $w_h < 12488$, the bound $\sqrt{w_h} \cdot 2^{-102.988}$ for the last term from the right-hand side of Eq. (1) is bounded by

$$\sqrt{12488} \cdot 2^{-102.988} < 2^{-96.183}.$$

Adding all together, and using $2^{-93.997} + 2^{-96.183} < 2^{-93.710}$, we obtain:

$$\left| s_h + \frac{t'}{2s_h} - \sqrt{x^2 - 1} \right| < 2^{-93.710}.$$

It remains to add the rounding error in Algorithm ComputeThG. A Gappa script proves that the total rounding error for $x_0 \leq x < 111.75$ is bounded by $2^{-94.212}$ (see Appendix A). Using $2^{-93.710} + 2^{-94.212} < 2^{-92.939}$, this concludes the proof. ∎

This branch uses the same algorithm (FastPath) as in branch $32896 \leq x < x_1$ (§V), except $\mathrm{acosh}(x)$ is approximated by $\log(t_h) + g$ where g is the value computed at line 9 of ComputeThG, instead of $\log(2x) + g(x)$ as in Eq. (6). Concretely, $2x$ at line 3 of FastPath is replaced by t_h, and g at line 17 is the value computed by ComputeThG. In the error analysis of §V, we thus simply have to replace the bound $2^{-66.437}$ for the difference between $\log(2x) + g(x)$ and

$\operatorname{acosh}(x)$ in Eq. (8) by a bound for the difference between $\log(t_h) + g$ and $\operatorname{acosh}(x)$:

$$\begin{aligned}
&|\log(t_h) + g - \operatorname{acosh}(x)| \\
&= |\log(t_h) + g - \log(x + \sqrt{x^2-1})| \\
&\le |\log(t_h) + g - \log(t_h + t_\ell)| \\
&+ |\log(t_h + t_\ell) - \log(x + \sqrt{x^2-1})|. \qquad (5)
\end{aligned}$$

The last term of the right-hand side of Eq. (5) is bounded by $2^{-92.939}$ from Lemma 2, multiplied by a bound on the derivative of the logarithm on $(t_h + t_\ell, x + \sqrt{x^2-1})$. The derivative of $\log(t)$ is $1/t$, and both $t_h + t_\ell$ and $x + \sqrt{x^2-1}$ are lower bounded by 1.621 at $x = x_0$. This last term is thus bounded by $2^{-92.939}/1.621 < 2^{-93.635}$.

The first term of the right-hand side of Eq. (5) is:

$$|\log(t_h) + g - \log(t_h(1 + \frac{t_\ell}{t_h}))| = |g - \log(1 + \frac{t_\ell}{t_h})|.$$

Since $|t_\ell| < 2^{-37.541}$ and $t_h > 1.621$, we have $|g|, |t_\ell/t_h| < 2^{-37.541}/1.621 < 2^{-38.237}$. Since $|\log(1+u) - u| < 0.502u^2$ for $|u| \le 2^{-8}$, it follows:

$$|g - \log(1 + \frac{t_\ell}{t_h})| \le |g - \frac{t_\ell}{t_h}| + 0.502 \cdot 2^{-76.474}.$$

The term $|g - t_\ell/t_h|$ is the rounding error on $g = \circ(t_\ell/t_h)$, which is bounded by $\operatorname{ulp}(2^{-38.237}) = 2^{-91}$. Since $2^{-91} + 0.502 \cdot 2^{-76.474} < 2^{-77.468}$, we finally get for the bound of Eq. (5):

$$|\log(t_h) + g - \operatorname{acosh}(x)| < 2^{-77.468} + 2^{-93.635} < 2^{-77.467}.$$

Replacing the term $2^{-66.437}$ in Eq. (8) by $2^{-77.467}$, we get for the values ℓ_h, ℓ_ℓ at the end of Algorithm FastPath:

$$\begin{aligned}
|\ell_h + \ell_\ell - \operatorname{acosh}(x)| &< 2^{-77.467} + 2^{-73.339} + 2^{-96.018} \\
&+ 2^{-62.999} + 2^{-96.049} + 2^{-85.878} \\
&< 2^{-62.997}.
\end{aligned}$$

The minimal 9-bit value of the error bound ϵ that works at line 20 of Algorithm FastPath is thus `0x1.81p-63` $> 2^{-62.997} + 2^{-64}$.

III. Branch $111.75 \le x < 738$

The fast path algorithm for this branch is the same as in branch $32896 \le x < x_1$ (§V). The only difference is that the function $g(x)$, which models the difference between $\operatorname{acosh}(x)$ and $\log(2x)$, is now of the form:

$$g(x) = g_0 + g_1 x^{-2} + g_2 x^{-4} + g_3 x^{-6},$$

with:

$$\begin{aligned}
g_0 &= \texttt{0x1.5c4b6148816e2p-66}, \\
g_1 &= \texttt{-0x1.000000000005cp-2}, \\
g_2 &= \texttt{-0x1.7fffffebf3e6cp-4}, \\
g_3 &= \texttt{-0x1.aab6691f2bae7p-5}.
\end{aligned}$$

See Figure 2. Sollya command `dirtyinfnorm` shows that:

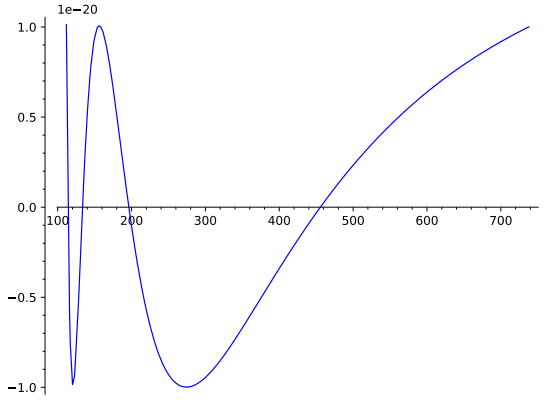

Fig. 2. Graph of $\log(2x) + g(x) - \operatorname{acosh}(x)$ on $[111.75, 738]$.

$$|\log(2x) + g(x) - \operatorname{acosh}(x)| < 2^{-66.419}.$$

The analysis for the total error in the variable f computed at line 11 of Algorithm FastPath is the same as in the proof of Lemma 3, since f only depends on the significand t', and not on the exponent e. For the total error on the final value of ℓ_ℓ, a modified Gappa script proves it is bounded by $2^{-62.778}$, taking into account the rounding error on d_x, and $|\ell_\ell| < 2^{-11.2}$. The total error is thus bounded by:

$$\begin{aligned}
|\ell_h + \ell_\ell - \operatorname{acosh}(x)| &< 2^{-66.419} + 2^{-73.339} + 2^{-96.018} \\
&+ 2^{-62.778} + 2^{-96.049} + 2^{-85.878} \\
&< 2^{-62.665}.
\end{aligned}$$

Since $2^{-62.665} + 2^{-64} \le \epsilon$ as soon as $\epsilon \ge$ `0x1.c3p-63`, the end of the proof of Lemma 3 applies.

IV. Branch $738 \le x < 32896$

The fast path algorithm for this branch is the same as in branch $32896 \le x < x_1$ (§V). The only difference is that the function $g(x)$, which models the difference between $\operatorname{acosh}(x)$ and $\log(2x)$, is now of the form:

$$g(x) = g_0 + g_1 x^{-2} + g_2 x^{-4},$$

with:

$$\begin{aligned}
g_0 &= \texttt{-0x1.7f77c8429c6c6p-67}, \\
g_1 &= \texttt{-0x1.fffffffff214p-3}, \\
g_2 &= \texttt{-0x1.8000268641bfep-4}.
\end{aligned}$$

See Figure 3. Sollya command `dirtyinfnorm` shows that:

$$|\log(2x) + g(x) - \operatorname{acosh}(x)| < 2^{-66.428}.$$

The analysis for the total error in f is the same as in the proof of Lemma 3, since f only depends on the significand t', and not on the exponent e. For the total error on the final value of ℓ_ℓ, a modified Gappa script proves it is bounded by $2^{-62.994}$, taking into account the rounding error on d_x, and $|\ell_\ell| < 2^{-11.2}$. The total error is thus bounded by:

$$\begin{aligned}
|\ell_h + \ell_\ell - \operatorname{acosh}(x)| &< 2^{-66.428} + 2^{-73.339} + 2^{-96.018} \\
&+ 2^{-62.994} + 2^{-96.049} + 2^{-85.878} \\
&< 2^{-62.865}.
\end{aligned}$$

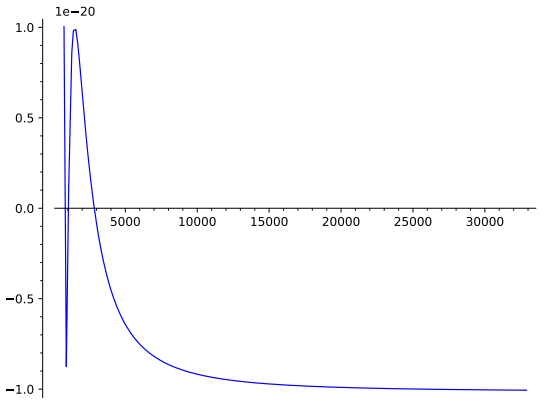

Fig. 3. Graph of $\log(2x) + g(x) - \operatorname{acosh}(x)$ on $[738, 32896]$.

Since $2^{-62.865} + 2^{-64} \leq \epsilon$ as soon as $\epsilon \geq$ `0x1.9ap-63`, the end of the proof of Lemma 3 applies.

V. Branch $32896 \leq x < x_1$

Let $x_1 =$ `0x1.ap+31`. The algorithm used in this branch is given in Algorithm 2 (FastPath). It uses the following approximation:

$$\operatorname{acosh}(x) \approx \log(2x) + g(x), \tag{6}$$

where $g(x) = g_0 + g_1/x^2$. Figure 4 shows how good this approximation is, and Sollya `dirtyinfnorm` command gives a maximal absolute difference bounded by $2^{-66.437}$.

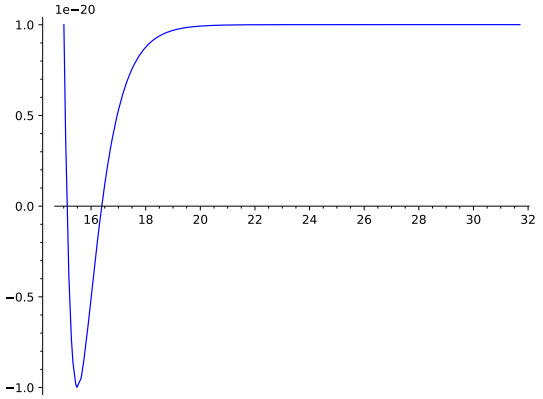

Fig. 4. Graph of $\log(2x) + g(x) - \operatorname{acosh}(x)$ on $[32896, x_1]$, where the abscissa denotes $\log_2 x$.

Comments about Algorithm FastPath:

- the integer j computed at line 5 is such that $2^{j/2^{10}}$ approximates t'. Then at line 8, r approximates $2^{-j/2^{10}}$, thus rt' is close to 1, thus d_x is small at line 9;
- at line 11, we omit roundings for clarity, and the binary64 coefficients $f_0, ..., f_4$ are such that (proven by Sollya):

$$|u + f_0u^2 + f_1u^3 + \cdots + f_4u^6 - \log(1+u)| < 2^{-85.878}, \tag{7}$$

for $|u| < 2^{-11.296}$;
- at lines 12-13, $L_h + L_\ell$ is a double-double approximation of $\log 2$ with absolute error bounded by $2^{-102.018}$. Since $16 \leq e \leq 32$ in this branch, and L_h is representable on 42 bits, the product L_he at line 14 is exact. Moreover since L_h is an integer multiple of 2^{-42}, so is L_he;
- the tables ℓ_{1h} and $\ell_{1\ell}$ are such that $\ell_{1h}[i_1] + \ell_{1\ell}[i_1]$ approximates $-\log r_1[i_1]$ for $0 \leq i_1 \leq 2^5$. Moreover $\ell_{1h}[i_1]$ is an integer multiple of 2^{-42}, and the maximal absolute error $|\ell_{1h}[i_1] + \ell_{1\ell}[i_1] + \log r_1[i_1]|$ is bounded by $2^{-97.066}$, $|\ell_{1h}[i_1]| < 0.70$, and $|\ell_{1\ell}[i_1]| < 2^{-43}$;
- the tables ℓ_{2h} and $\ell_{2\ell}$ are such that $\ell_{2h}[i_2] + \ell_{2\ell}[i_2]$ approximates $-\log r_2[i_2]$ for $0 \leq i_2 < 2^5$. Moreover $\ell_{2h}[i_2]$ is an integer multiple of 2^{-42}, the maximal absolute error $|\ell_{2h}[i_2] + \ell_{2\ell}[i_2] + \log r_2[i_2]|$ is bounded by $2^{-97.033}$, $|\ell_{2h}[i_2]| < 0.03$, and $|\ell_{2\ell}[i_2]| < 2^{-43}$;
- since L_he, $\ell_{1h}[i_1]$ and $\ell_{2h}[i_2]$ are integer multiples of 2^{-42}, so is their sum at line 14, and this sum is bounded by $32 \cdot 0.70 + 0.70 + 0.03 < 24 < 2^{47} \cdot 2^{-42}$, thus ℓ_h at line 14 is exact (and the order of summation does not matter);
- at lines 15-19, we accumulate several small quantities to d_x, to form the lower part of the approximation of $\operatorname{acosh}(x)$, the upper part being ℓ_h computed at line 14. We accumulate these quantities from the smaller one to the larger one (in absolute value) to minimize the rounding errors.

Algorithm 2 (FastPath)

Input: x a binary64 number, $32896 \leq x \leq x_1$
Output: the correct rounding $\circ(\operatorname{acosh}(x))$ or FAIL

1: $g_0 =$ `0x1.7a0ed2effdd1p-67`
2: $g_1 =$ `-0x1.000000017d048p-2`
3: write $2x = 2^e \cdot (1 + 2^{-52}t)$ with $0 \leq t < 2^{52}$, t integer
4: write $t = i \cdot 2^{47} + d$ with $0 \leq i < 2^5$ and $0 \leq d < 2^{47}$
5: compute an integer $j \in [0, 1024]$ from t, d and tables indexed by i
6: write $j = i_1 2^5 + i_2$ with $0 \leq i_1 \leq 2^5$ and $0 \leq i_2 < 2^5$
7: $t' \leftarrow 1 + 2^{-52}t$
8: $r \leftarrow r_1[i_1] \cdot r_2[i_2]$ ▷ exact
9: $d_x \leftarrow \operatorname{fma}(r, t', -1)$
10: $d_{x2} \leftarrow \circ(d_x^2)$
11: $f \leftarrow d_{x2}((f_0 + d_xf_1) + d_{x2}((f_2 + d_xf_3) + d_{x2}f_4))$
12: $L_h =$ `0x1.62e42fefa38p-1`
13: $L_\ell =$ `0x1.ef35793c7673p-45`
14: $\ell_h \leftarrow \ell_{1h}[i_1] + \ell_{2h}[i_2] + L_he$ ▷ exact
15: $\ell_1 \leftarrow \circ(\circ(L_\ell e) + \circ(\ell_{1\ell}[i_1] + \ell_{2\ell}[i_2]))$
16: $\ell_2 \leftarrow \circ(f + \ell_1)$
17: $z \leftarrow \circ(1/\circ(x^2))$, $g \leftarrow \circ(g_0 + \circ(zg_1))$
18: $\ell_3 \leftarrow \circ(g + \ell_2)$
19: $\ell_\ell \leftarrow \circ(d_x + \ell_3)$
20: $\epsilon =$ `0x1.4dp-62`
21: $\ell_b \leftarrow \ell_h + \circ(\ell_\ell - \epsilon)$, $u_b \leftarrow \ell_h + \circ(\ell_\ell + \epsilon)$
22: if $\circ(\ell_b) = \circ(u_b)$ then return $\circ(\ell_b)$ else return FAIL

Lemma 3: For $32896 \leq x < x_1$, Algorithm FastPath is correct, i.e., if it does not return FAIL, the returned value is the correct rounding of $\operatorname{acosh}(x)$.

Proof: We know from Sollya that:

$$|\log(2x) + g(x) - \operatorname{acosh}(x)| < 2^{-66.437}.$$

Algorithm FastPath approximates $\log(2x) + g(x)$ by $\ell_h + \ell_\ell$. Let us analyze the different rounding errors line per line.

At line 8, $r_1[i_1]$ is a 20-bit approximation of $2^{-i_1/2^5}$, and $r_2[i_2]$ is a 20-bit approximation of $2^{-i_2/2^{10}}$, thus the product $r_1[i_1] \cdot r_2[i_2]$ is exact, and it approximates $2^{-j/2^{10}}$.

By checking the extremal values of t' for a given j, for all $j \in [0, 1024]$, taking into account the way j is computed (we refer to the source code for this), we can prove that at line 9, we have $|d_x| < 2^{-11.296}$, where the maximal value is obtained for $j = 46$.

Using a small Gappa script, we are able to show that the total rounding error on f at line 11, taking into account the rounding error on the FMA at line 9, and the rounding error at line 10, is bounded by $2^{-73.339}$, and that $|f| < 2^{-23.575}$.

At lines 12-13, the difference between $L_h + L_\ell$ and $\log 2$ is bounded by $2^{-102.018}$, and will be multiplied at lines 14-15 by e which is bounded by 32, yielding an error of at most $2^{-96.018}$.

We have seen above that the computation of ℓ_h at line 14 is exact.

Another Gappa script shows that the cumulated rounding errors at lines 15-19 is bounded by $2^{-62.999}$, taking also into account the rounding error in the FMA computing d_x at line 9, and shows that $|\ell_\ell| < 2^{-11.287}$ after line 19.

Since the difference between $\ell_{1h}[i_1] + \ell_{1\ell}[i_1]$ and $-\log r_1[i_1]$ is bounded by $2^{-97.066}$, and that between $\ell_{2h}[i_2] + \ell_{2\ell}[i_2]$ and $-\log r_2[i_2]$ by $2^{-97.033}$, this yields an additional error of at most $2^{-97.066} + 2^{-97.033} < 2^{-96.049}$.

Taking also into account the error from Eq. (7) on approximating $\log(1+u)$, the total absolute error is bounded by:

$$\begin{aligned} |\ell_h + \ell_\ell - \operatorname{acosh}(x)| &< 2^{-66.437} + 2^{-73.339} + 2^{-96.018} \\ &+ 2^{-62.999} + 2^{-96.049} + 2^{-85.878} \\ &< 2^{-62.870}. \end{aligned} \quad (8)$$

Now since $|\ell_\ell| < 2^{-11.287}$, we have $|\ell_\ell \pm \epsilon| < 2^{-11.286}$, thus the rounding error on $\ell_\ell \pm \epsilon$ is bounded by 2^{-64}. Thus $\ell_b - \operatorname{acosh}(x) < -\epsilon + 2^{-62.870} + 2^{-64} \leq 0$. Similarly, $u_b - \operatorname{acosh}(x) > \epsilon - 2^{-62.870} - 2^{-64} \geq 0$. Thus $\ell_b \leq \operatorname{acosh}(x) \leq u_b$, whence if ℓ_b and u_b round to the same value, so does $\operatorname{acosh}(x)$. ∎

The minimal 9-bit value that works for this branch is $\epsilon =$ `0x1.99p-63`.

VI. Branch $x_1 \leq x$

The fast path algorithm for this branch is the same as in branch $32896 \leq x < x_1$ (§V). The only difference is that the function $g(x)$, which models the difference between $\operatorname{acosh}(x)$ and $\log(2x)$, is now $g(x) = 0$. As seen on Figure 5, the difference $|\log(2x) - \operatorname{acosh}(x)|$ decreases on $[x_1, 2^{40}]$ (and further decreases for larger values), thus it is bounded by its value at x_1:

$$|\log(2x) - \operatorname{acosh}(x)| < 2^{-65.4}.$$

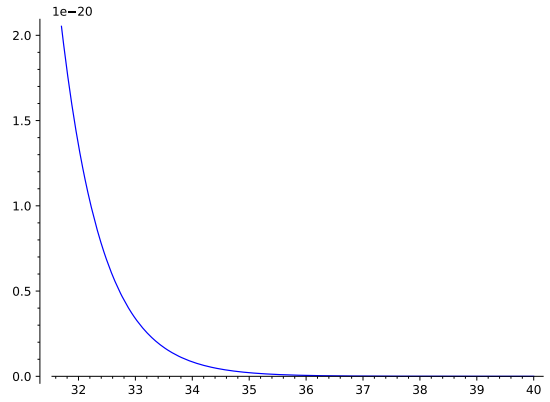

Fig. 5. Graph of $\log(2x) - \operatorname{acosh}(x)$ on $[x_1, 2^{40}]$, where the abscissa denotes $\log_2 x$.

In Algorithm FastPath, since $g = 0$, we simply ignore lines 16-17. A modified Gappa script shows that the cumulated rounding errors at lines 15-19, also taking into account the rounding error on d_x, is bounded by $2^{-62.999}$. This yields the bound:

$$\begin{aligned} |\ell_h + \ell_\ell - \operatorname{acosh}(x)| &< 2^{-65.4} + 2^{-73.339} + 2^{-96.018} \\ &+ 2^{-62.999} + 2^{-96.049} + 2^{-85.878} \\ &< 2^{-62.747}. \end{aligned}$$

Since $2^{-62.747} + 2^{-64} < \epsilon$ as soon as $\epsilon \geq$ `0x1.b2p-63`, the end of the proof of Lemma 3 applies.

VII. Accurate path

Since we did an exhaustive test of branch $1 < x < x_0$, we only have to prove the accurate path for $x \geq x_0$. The accurate path is detailed in Algorithm AccuratePath.

Remarks: the approximation a at input is computed as $\circ$(`0x1.71547652b82fep+0` $\cdot \ell_b$), where ℓ_b is the (rounded) approximation from the fast path. For $x \geq 2^{1023}$, $z_h = 2x$ overflows at line 11, the actual code is using a trick to avoid this overflow, that we don't detail here for simplicity. At line 16, $t_1[i_1]$ is a 26-bit approximation of $2^{-i_1/2^4}$, $t_2[i_2]$ a 26-bit approximation of $2^{-i_2/2^8}$, $t_3[i_3]$ a 26-bit approximation of $2^{-i_3/2^{12}}$, and $t_4[i_4]$ a 26-bit approximation of $2^{-i_4/2^{16}}$, thus the products $t_1[i_1] \cdot t_2[i_2]$ and $t_3[i_3] \cdot t_4[i_4]$ are exact.

From [1, Theorem 4.1], we know that Algorithm TwoSum satisfies, if all the roundings are faithful:

$$|s + t - (a + b)| \leq 2^{-52}\operatorname{ulp}(s).$$

After line 11 of Algorithm AccuratePath, $z_h + z_\ell$ is a double-double approximation of $x + \sqrt{x^2 - 1}$. We analyze in Lemma 4 how accurate this approximation is.

Lemma 4: After line 11 of Algorithm AccuratePath, we have:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| < 2^{-98.143} x.$$

Proof: For $x_0 \leq x < 2^{26}$, $x_{2\ell}$ is exact in line 2 (EFT for multiplication), thus $x_{2h} + x_{2\ell} = x^2$. At line 3, t_h is exact too (same argument as w_h in Algorithm ComputeThG). Assume

Algorithm 3 (AccuratePath)

Input: $x \geq x_0$ a binary64 number, $a \approx \log_2(x + \sqrt{x^2-1})$
Output: $y_h + y_\ell$ approximating $\operatorname{acosh}(x)/2$

1: **if** $x < 2^{26}$ **then**
2: $\quad x_{2h} = \circ(x \cdot x), \quad x_{2\ell} = \mathrm{fma}(x, x, -x_{2h})$
3: $\quad t_h = \circ(x_{2h} - 1), \quad w_h, w_\ell = \mathrm{fasttwosum}(t_h, x_{2l})$
4: $\quad s_h = \circ(\sqrt{w_h})$
5: $\quad s_\ell = \circ(\circ(w_\ell - \mathrm{fma}(s_h, s_h, -w_h))/(2s_h))$
6: $\quad z_h, z_\ell = \mathrm{fasttwosum}(x, s_h), \quad z_\ell = \circ(z_\ell + s_\ell)$
7: $\quad z_h, z_\ell = \mathrm{fasttwosum}(z_h, z_\ell)$
8: **else if** $x < 2^{52}$ **then**
9: $\quad z_h = 2x,\ z_\ell = \circ(-0.5/x)$
10: **else**
11: $\quad z_h = 2x,\ z_\ell = 0$
12: write $z_h = t \cdot 2^e$ with $1 \leq t < 2$
13: $v \leftarrow \circ(a - e + \texttt{0x1.00008p+0})$
14: $i \leftarrow \lfloor 2^{16}(v-1) \rfloor$
15: write $i = 2^{12}i_1 + 2^8 i_2 + 2^4 i_3 + i_4$ with $0 \leq i_1 \leq 16$ and $0 \leq i_2, i_3, i_4 < 16$
16: $t_{12} \leftarrow t_1[i_1] \cdot t_2[i_2], \quad t_{34} \leftarrow t_3[i_3] \cdot t_4[i_4]$ ▷ exact
17: $t_h \leftarrow \circ(t_{12} \cdot t_{34}), \quad t_\ell \leftarrow \mathrm{fma}(t_{12}, t_{34}, -t_h)$
18: $d_h \leftarrow \circ(t_h \cdot t), \quad d_\ell \leftarrow \mathrm{fma}(t_h, t, -d_h)$
19: $s_h \leftarrow \circ(t_\ell \cdot t), \quad s_\ell \leftarrow \mathrm{fma}(t_\ell, t, -s_h)$
20: $x_h, x_\ell \leftarrow \mathrm{fasttwosum}(d_h - 1, d_\ell)$
21: $x_\ell \leftarrow \circ(x_\ell + z_\ell 2^{-e})$
22: $x_h, x_\ell \leftarrow \mathrm{adddd}(x_h, x_\ell, s_h, s_\ell)$
23: $e_{\ell 0} \leftarrow \ell_{20} \cdot e$ where $\ell_{20} = \texttt{0x1.62e42fefa38p-2}$
24: $e_{\ell 1} \leftarrow \ell_{21} \cdot e$ where $\ell_{21} = \texttt{0x1.ef35793c768p-46}$
25: $e_{\ell 2} \leftarrow \circ(\ell_{22} \cdot e)$ where $\ell_{22} = \texttt{-0x1.9ff0342542fc3p-91}$
26: $H \leftarrow H_1[i_1] + H_2[i_2] + H_3[i_3] + H_4[i_4]$ ▷ exact
27: $M \leftarrow M_1[i_1] + M_2[i_2] + M_3[i_3] + M_4[i_4]$ ▷ exact
28: $L \leftarrow L_1[i_1] + L_2[i_2] + L_3[i_3] + L_4[i_4]$
29: $H \leftarrow H + e_{\ell 0}$ ▷ exact
30: $p_h, p_\ell \leftarrow \mathrm{PolyEval}(x_h, x_\ell)$
31: $q_h, q_\ell \leftarrow \mathrm{adddd}(p_h, p_\ell, e_{\ell 1}, e_{\ell 2})$
32: $r_h, r_\ell \leftarrow \mathrm{adddd}(q_h, q_\ell, M, L)$
33: $y_h, z_\ell \leftarrow \mathrm{fasttwosum}(H, r_h), \quad y_\ell \leftarrow \circ(z_\ell + r_\ell)$

Algorithm 4 (adddd)

Input: double numbers x_h, x_ℓ, s_h, s_ℓ
Output: double-double approximation $s+\ell$ of $x_h + x_\ell + s_h + s_\ell$

1: $s, \ell \leftarrow \mathrm{TwoSum}(x_h, s_h)$
2: $\ell \leftarrow \circ(\ell + \circ(x_\ell + s_\ell))$

$2^{e-1} \leq x < 2^e$ for $1 \leq e \leq 26$. Then $t_h < 2^{2e}$, and t_h and $x_{2\ell}$ are integer multiples of $\mathrm{ulp}^2(x) = 2^{2e-106}$. Thus the significant bits of t_h and $x_{2\ell}$ cover at most a range of 106 bits, thus their exponent difference cannot exceed 53. By Lemma 1 from [3], the FastTwoSum at line 3 is exact: $w_h + w_\ell = t_h + x_{2l} = x^2 - 1$. We thus have:

$$\begin{aligned}|s_h + s_\ell - \sqrt{x^2-1}| &= |s_h + s_\ell - \sqrt{w_h + w_\ell}| \\ &\leq |s_h + s_\ell - (\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}})| + |\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}} - \sqrt{w_h + w_\ell}|.\end{aligned}$$

Algorithm 5 (TwoSum)

Input: a and b binary64 numbers
Output: $s + t$ approximating $a + b$

1: $s \leftarrow \circ(a + b)$
2: $a' \leftarrow \circ(s - b)$
3: $b' \leftarrow \circ(s - a')$
4: $\delta_a \leftarrow \circ(a - a')$
5: $\delta_b \leftarrow \circ(b - b')$
6: $t \leftarrow \circ(\delta_a + \delta_b)$

The first term is the rounding error when computing s_h and s_ℓ, while the second term is the mathematical error when approximating $\sqrt{1+u}$ by $1 + u/2$ for u small (here $u = w_\ell/w_h$). Since the FastTwoSum at line 3 is exact, w_ℓ is the rounding error when computing $w_h = \circ(t_h + x_{2\ell})$, thus $|w_\ell| < \mathrm{ulp}(w_h)$, and $|w_\ell/w_h| < 2^{-52}$. For $|u| \leq 2^{-8}$, we have $|\sqrt{1+u} - (1 + u/2)| < 0.126u^2$, and since $0.126 \cdot 2^{-104} < 2^{-106.988}$, the second term is bounded as follows:

$$|\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}} - \sqrt{w_h + w_\ell}| < 2^{-106.988}\sqrt{w_h}. \tag{9}$$

For the first term, denoting $t = -\mathrm{fma}(s_h, s_h, -w_h)$:

$$|s_h + s_\ell - (\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}})| \tag{10}$$

$$\leq |s_h + \frac{t}{2s_h} - \sqrt{w_h}| + |\frac{w_\ell}{2s_h} - \frac{w_\ell}{2\sqrt{w_h}}| + |s_\ell - \frac{w_\ell + t}{2s_h}|. \tag{11}$$

For the first term of the right bound of Eq. (11):

$$\begin{aligned}|s_h + \frac{t}{2s_h} - \sqrt{w_h}| &\leq \frac{|(s_h + t/(2s_h))^2 - w_h|}{s_h + t/(2s_h) + \sqrt{w_h}} \\ &\leq \frac{|s_h^2 + t + t^2/(4s_h^2) - w_h|}{s_h + t/(2s_h) + \sqrt{w_h}}.\end{aligned}$$

Let ϵ_1 be the rounding error in the computation of t, i.e., $t = w_h - s_h^2 + \epsilon_1$, then:

$$|s_h + \frac{t}{2s_h} - \sqrt{w_h}| \leq \frac{|\epsilon_1 + t^2/(4s_h^2)|}{s_h + t/(2s_h) + \sqrt{w_h}}.$$

From Lemma 1, we know $|\epsilon_1| \leq 2^{-52}\mathrm{ulp}(w_h)$, and from the proof of Lemma 1, we see that $|t| \leq 8u \leq 4\mathrm{ulp}(x)$, thus $t^2/(4s_h^2) \leq 4\mathrm{ulp}(x)^2/s_h^2$. Using $s_h > 0.449x$ (s_h/x is minimal near x_0), it follows $t^2/(4s_h^2) < 4/0.449^2(\mathrm{ulp}(x)/x)^2 < 2^{-99.689}$. Using $s_h + t/(2s_h) \geq 0.999\sqrt{w_h}$ and $\mathrm{ulp}(w_h) \leq 2^{-52}w_h$, it follows:

$$|s_h + \frac{t}{2s_h} - \sqrt{w_h}| \leq \frac{2^{-104}w_h + 2^{-99.689}}{1.999\sqrt{w_h}}.$$

Since $w_h > 0.252$, $2^{-99.689} < 2^{-97.700}w_h$, thus

$$|s_h + \frac{t}{2s_h} - \sqrt{w_h}| \leq 2^{-98.681}\sqrt{w_h} < 2^{-98.681}x. \tag{12}$$

For the second term of the right bound of Eq. (11):

$$|\frac{w_\ell}{2s_h} - \frac{w_\ell}{2\sqrt{w_h}}| = \frac{|w_\ell|}{2s_h\sqrt{w_h}} \cdot |\sqrt{w_h} - s_h|.$$

Since $|\sqrt{w_h} - s_h| < \mathrm{ulp}(s_h)$ by definition, and $|w_\ell/w_h| \leq 2^{-52}$, it follows:

$$|\frac{w_\ell}{2s_h} - \frac{w_\ell}{2\sqrt{w_h}}| \leq 2^{-105}\sqrt{w_h} < 2^{-105}x. \tag{13}$$

For the third term of the right bound of Eq. (11), it comes from the error on $w_\ell - t$ and that of the division by $2s_h$:

$$\begin{aligned}&|s_\ell - \frac{w_\ell + t}{2s_h}| \\ &\leq |s_\ell - \frac{\circ(w_\ell + t)}{2s_h}| + |\frac{\circ(w_\ell + t)}{2s_h} - \frac{w_\ell + t}{2s_h}|.\end{aligned}$$

The second term is bounded by $\mathrm{ulp}(u)/(2s_h)$ where $u = w_\ell + t$. Since $|t| \leq 4\mathrm{ulp}(x)$ and $|w_\ell| < \mathrm{ulp}(w_h) \leq \mathrm{ulp}(x^2)$, we have $|u| < \mathrm{ulp}(x^2) + 4\mathrm{ulp}(x)$. It can be easily shown that $|u| < 2^{-50.003}x^2$ in this range, thus reusing $s_h > 0.449x$: $\mathrm{ulp}(u)/(2s_h) < 2^{101.847}x$. The first term is bounded by $\mathrm{ulp}(s_\ell)$ and has the same bound:

$$\mathrm{ulp}(s_\ell) \leq 2^{-52}|s_\ell| \leq 2^{-52}\frac{|\circ(u)|}{2s_h} \leq 2^{101.847}x.$$

In summary:

$$|s_\ell - \frac{w_\ell + t}{2s_h}| \leq 2^{100.847}x. \tag{14}$$

Plugging Eq. (12), Eq. (13) and Eq. (14) into Eq. (11), we obtain:

$$|s_h + s_\ell - (\sqrt{w_h} + \frac{w_\ell}{2\sqrt{w_h}})| < 2^{-98.376}x.$$

Adding the mathematical error from Eq. (9), we get:

$$|s_h + s_\ell - \sqrt{x^2 - 1}| < 2^{-98.372}x.$$

Now after $z_h, z_\ell = \mathrm{fasttwosum}(x, s_h)$, the rounding error of the FastTwoSum is either zero for rounding to nearest, or bounded by $\mathrm{ulp}_{106}(z_h)$ using Theorems 9-11 from [3], where $z_h \leq 2x$. This rounding error is thus bounded by $\mathrm{ulp}_{106}(2x) \leq 2^{-104}x$. And since $|z_\ell| \leq \mathrm{ulp}(z_h)$ after line 6 of Accuratepath, and $|s_\ell| < \circ(u/(2s_h)) \leq 2^{-49.847}x$, we have $|z_\ell + s_\ell| < 2^{-52}|2x| + 2^{-49.847}x < 2^{-49.311}x$, thus the rounding error on $\circ(z_\ell + s_\ell)$ is bounded by $\mathrm{ulp}(2^{-49.311}x) \leq 2^{-101.311}x$. Thus after line 6, using $2^{-98.372} + 2^{-104} + 2^{-101.311} < 2^{-98.169}$:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| < 2^{-98.169}x.$$

Now after line 7, we have another potential rounding error bounded by $\mathrm{ulp}_{106}(z_h) \leq 2^{-104}x$, which yields:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| < 2^{-98.143}x.$$

For $2^{26} \leq x < 2^{52}$, we have $z_h = 2x$ (which is exact) and $z_\ell = \circ(-0.5/x)$, thus:

$$\begin{aligned}&|z_h + z_\ell - (x + \sqrt{x^2 - 1})| = |x + z_\ell - x\sqrt{1 - 1/x^2}| \\ &\leq |x - \frac{0.5}{x} - x\sqrt{1 - 1/x^2}| + |-\frac{0.5}{x} - z_\ell|.\end{aligned}$$

For $|u| \leq 2^{-8}$, we have $|\sqrt{1+u} - (1 + u/2)| < 0.126u^2$, thus $|\sqrt{1 - 1/x^2} - (1 - 1/(2x^2))| < 0.126/x^4$. The first term above is thus bounded by $0.126/x^4 \cdot x < 2^{-106.988}x$; since $0.5/x \leq 2^{-27}$, the second term is the rounding error while computing z_ℓ, which is bounded by $\mathrm{ulp}(0.5/x) \leq 2^{-53}/x \leq 2^{-105}x$ since $x \geq 2^{26}$. We conclude in this case:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| < 2^{-104.675}x.$$

For $x \geq 2^{52}$, we have $z_h = 2x$ (which is exact) and $z_\ell = 0$. Then:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| = |x - x\sqrt{1 - 1/x^2}|.$$

For $|u| \leq 2^{-8}$, we have $|\sqrt{1+u} - 1| < 0.501|u|$, thus $|\sqrt{1 - 1/x^2} - 1| < 0.501/x^2$. We conclude in this case:

$$|z_h + z_\ell - (x + \sqrt{x^2 - 1})| < 2^{-104.997}x.$$

■

Lemma 5: After line 22 of Algorithm AccuratePath, denoting $r_1 = t_1[i_1]$, $r_2 = t_2[i_2]$, $r_3 = t_3[i_3]$, $r_4 = t_4[i_4]$, and $R = r_1r_2r_3r_4$, we have:

$$|x_h + x_\ell - (R \cdot T - 1)| < 2^{-103.677},$$

where $T = 2^{-e}(z_h + z_\ell)$.

Proof: We have seen t_{12} and t_{34} are exact, and $t_h + t_\ell = t_{12} \cdot t_{34}$ at line 17 (EFT for multiplication), thus:

$$t_h + t_\ell = R.$$

Similarly, $d_h + d_\ell = t_h \cdot t$ after line 18, and $s_h + s_\ell = t_\ell \cdot t$ after line 19, thus:

$$d_h + d_\ell + s_h + s_\ell = R \cdot t.$$

We wrote a Sage program that, for all possible values of i_1, i_2, i_3, i_4, finds extremal values of x giving these values, on subranges where i_1, i_2, i_3, i_4 remain constant. This program proves $|d_h - 1| < 2^{-17.520}$, thus $d_h - 1$ is exact at line 20 and the rounding error of the FastTwoSum call is bounded by $\mathrm{ulp}_{106}(2^{-17.520}) = 2^{-123}$ (Theorems 9-11 from [3]). This program also proves $|x_\ell| \leq 2^{-53}$ after line 21, thus the rounding error on x_ℓ at line 21 is bounded by $\mathrm{ulp}(2^{-54}) = 2^{-106}$. Finally before line 22, $|x_h| < 2^{-17.520}$, $|x_\ell| \leq 2^{-53}$, $|s_h| < 2^{-53}$, $|s_\ell| < 2^{-107}$.

It follows from the analysis of adddd and TwoSum that after line 22, $|x_h| < 2^{-17.519}$. Since $|s + \ell - (x_h + s_h)| \leq 2^{-52}\mathrm{ulp}(s)$ after line 1 of adddd, and $|s - (x_h + s_h)| < \mathrm{ulp}(s)$, we deduce $|\ell| < (1 + 2^{-52})\mathrm{ulp}(2^{-17.519}) < 2^{-69.999}$. Since $|x_\ell + s_\ell| < 2^{-53} + 2^{-107}$, the rounding error in $\circ(x_\ell + s_\ell)$ at line 2 of adddd is bounded by $\mathrm{ulp}(2^{-53}) = 2^{-105}$. Then the rounding error in $\circ(\ell + \cdots)$ at the same line is bounded by 2^{-105} too. The total rounding error of adddd is thus bounded by (since s from adddd is x_h after line 22 of AccuratePath):

$$2^{-52}\mathrm{ulp}(x_h) + 2 \cdot 2^{-105} < 2^{-103.999}.$$

The total rounding error of lines 16-22 of AccuratePath is thus bounded by:

$$2^{-123} + 2^{-106} + 2^{-103.999} < 2^{-103.677}.$$

This concludes the proof. ■

Since $RT \approx 1 + x_h + x_\ell$, we now have:

$$\operatorname{acosh}(x) \approx \log(z_h + z_\ell) \approx e\log 2 - \log R + \log(1 + x_h + x_\ell).$$

Since $e \leq 1024$ and ℓ_{20}, ℓ_{21} are representable on 42 bits, $e_{\ell 0}$ and $e_{\ell 1}$ are exact. The sum $\ell_{20} + \ell_{21} + \ell_{22}$ approximates $\log(2)/2$ with error less than $2^{-145.438}$, thus after multiplication by e this induces an error less than $2^{-135.438}$, and taking into account the rouding error in $e_{\ell 2}$:

$$|e_{\ell 0} + e_{\ell 1} + e_{\ell 2} - e\frac{\log(2)}{2}| < 2^{-135.438} + 2^{-133} < 2^{-132.755}. \tag{15}$$

At lines 26-28, $H_1[i_1] + M_1[i_1] + L_1[i_1]$ is a triple-double approximation of $-\log(t_1[i_1])/2$ with error bounded by $2^{-146.016}$, where $H_1[i_1]$ is a multiple of 2^{-40} and $|H_1[i_1]| < 0.347$, $M_1[i_1]$ is a multiple of 2^{-91} and $|M_1[i_1]| < 2^{-41}$, and $|L_1[i_1]| < 2^{-92}$. Similarly for $H_2[i_2] + M_2[i_2] + L_2[i_2]$ which approximates $-\log(t_2[i_2])/2$ with error bounded by $2^{-146.200}$ and $|H_2[i_2]| < 0.021$ (other properties unchanged), for $H_3[i_3] + M_3[i_3] + L_3[i_3]$ which approximates $-\log(t_3[i_3])/2$ with error bounded by $2^{-146.143}$ and $|H_3[i_3]| < 0.002$ (other properties unchanged), and for $H_4[i_4] + M_4[i_4] + L_4[i_4]$ which approximates $-\log(t_4[i_4])/2$ with error bounded by $2^{-146.376}$ and $|H_4[i_4]| < 0.001$ (other properties unchanged). The sum H at line 26 is thus exact, a multiple of 2^{-40}, and $|H| < 0.371$. Similarly, the sum M at line 27 is exact, a multiple of 2^{-91}, and $|M| \leq 2^{-39}$. Finally, the sum L at line 28 is bounded by 2^{-90} in absolute value, and the rounding error on L is bounded by 2^{-142}. We thus have:

$$\begin{aligned}|H + M + L + \log(R)/2| &< 2^{-146.016} + 2^{-146.200} \\ &+ 2^{-146.143} + 2^{-146.376} + 2^{-142} < 2^{-141.711}.\end{aligned}$$

Before line 29, since H is a multiple of 2^{-40}, $|H| < 0.371$, ℓ_{20} thus $e_{\ell 0}$ is a multiple of 2^{-43}, $|e_{\ell 0}| < 355$, we have $H + e_{\ell 0}$ multiple of 2^{-43} with $|H + e_{\ell 0}| < 356$, thus $H + e_{\ell 0}$ is exact.

To approximate $\log(1 + x_h + x_\ell)/2$, the code uses a degree-6 polynomial approximation $q(x)$ of $\log(1+x)/2$ for $|x| < 2^{-17.519}$, with absolute error bounded by $2^{-128.752}$. The coefficients of degree 1 to 3 of $q(x)$ are double-doubles, while the coefficients of degree 4 to 6 are doubles (see Algorithm PolyEval).

Algorithm muldd is exactly Algorithm 10 from [4]. In [4], when the inputs are normalized, i.e., $x_h = \mathrm{RN}(x_h + x_\ell)$, similarly for $c_h + c_\ell$, and the precision is at least 4, the authors give a relative error bound of $7u^2$ for rounding to nearest-even, where $u = 2^{-53}$ is the unit roundoff. (A tight relative error bound of $5u^2$ is given for rounding to nearest-even in [5, Theorem 2.7].)

Lemma 6: Let x_h, x_ℓ two binary64 values, with $|x_h| < 2^{-17.519}$ and $|x_\ell| \leq 2^{-53}$, then the output values p_h, p_ℓ of Algorithm PolyEval satisfy:

$$|p_h + p_\ell - \frac{1}{2}\log(1 + x_h + x_\ell)| < 2^{-103.939}.$$

Proof: First Sollya proves for $|x| < 2^{-17.519} + 2^{-53} < 2^{-17.518}$:

$$|q(x) - \frac{1}{2}\log(1+x)| < 2^{-128.752}.$$

Algorithm 6 (PolyEval)

Input: x_h, x_ℓ two binary64 values
Output: $p_h + p_\ell$ approximating $\log(1 + x_h + x_\ell)/2$

1: $d_5 \leftarrow \circ(x_h q_6)$, $\quad c_5 \leftarrow \circ(q_5 + d_5)$
2: $d_4 \leftarrow \circ(x_h c_5)$, $\quad c_4 \leftarrow \circ(q_4 + d_4)$
3: $d_3 \leftarrow \circ(x_h c_4)$
4: $c_{3h}, t_{3\ell} \leftarrow \mathrm{fasttwosum}(q_{3h}, d_3)$
5: $c_{3\ell} \leftarrow \circ(t_{3\ell} + q_{3\ell})$
6: $d_{2h}, d_{2\ell} \leftarrow \mathrm{muldd}(x_h, x_\ell, c_{3h}, c_{3\ell})$
7: $c_{2h}, t_{2\ell} \leftarrow \mathrm{fasttwosum}(q_{2h}, d_{2h})$
8: $c_{2\ell} \leftarrow \circ(d_{2\ell} + \circ(t_{2\ell} + q_{2\ell}))$
9: $d_{1h}, d_{1\ell} \leftarrow \mathrm{muldd}(x_h, x_\ell, c_{2h}, c_{2\ell})$
10: $c_{1h}, t_{1\ell} \leftarrow \mathrm{fasttwosum}(q_{1h}, d_{1h})$
11: $c_{1\ell} \leftarrow \circ(d_{1\ell} + \circ(t_{1\ell} + q_{1\ell}))$
12: $p_h, p_\ell \leftarrow \mathrm{muldd}(x_h, x_\ell, c_{1h}, c_{1\ell})$

Algorithm 7 (muldd)

Input: x_h, x_ℓ, c_h, c_ℓ binary64 numbers
Output: h, ℓ approximating $(x_h + x_\ell)(c_h + c_\ell)$

1: $\ell_1 \leftarrow x_h c_\ell$, $\ell_2 \leftarrow x_\ell c_h$
2: $h_1 \leftarrow x_h c_h$, $\ell_3 \leftarrow \mathrm{fma}(x_h, c_h, -h_1)$
3: $\ell_4 \leftarrow \ell_3 + (\ell_1 + \ell_2)$
4: $h, \ell \leftarrow \mathrm{fasttwosum}(h_1, \ell_4)$

Then a Gappa script proves

$$|p_h + p_\ell - q(x_h + x_\ell)| < 2^{-103.94}.$$

This concludes since $2^{-128.752} + 2^{-103.94} < 2^{-103.939}$. This Gappa script also proves $|p_h| < 2^{-18.518}$ and $|p_\ell| \leq 2^{-71}$. ■

Theorem 1: Let $x \geq x_0$ a binary64 number, then the output y_h, y_ℓ of Algorithm AccuratePath satisfies:

$$|(2y_h + 2y_\ell) - \operatorname{acosh}(x)| < 2^{-94.347}.$$

Proof: From the proof of Lemma 6, we know $|p_h| < 2^{-18.518}$ and $|p_\ell| \leq 2^{-71}$. Since $\ell_{21} < 2^{-45}$, $|\ell_{22}| < 2^{-90}$ and $e \leq 1024$, $|e_{\ell 1}| \leq 2^{-35}$ and $|e_{\ell 2}| \leq 2^{-80}$. Thus in the adddd call from line 31 of AccuratePath, we have $|s| < 2^{-18.517}$, $|\ell| \leq 2^{-71}$ after TwoSum, and $|\ell| < 2^{-69.998}$ finally, thus the rounding error of this adddd call is bounded by:

$$2^{-52}\mathrm{ulp}(s) + 2^{-123} + \mathrm{ulp}(2^{-72}) < 2^{-121.678},$$

where 2^{-123} accounts for the rounding error in $\circ(x_\ell + s_\ell)$, here $\circ(p_\ell + e_{\ell 2})$.

For the adddd call from line 32, we have $|q_h| < 2^{-18.517}$, $|q_\ell| < 2^{-69.998}$, $|M| < 2^{-39}$, $|L| \leq 2^{-90}$, thus inside adddd we have $|s| < 2^{-18.516}$, $|\ell| \leq 2^{-71}$ after TwoSum, and $|\ell| < 2^{-69.413}$ finally, thus the rounding error of this adddd call is bounded by:

$$2^{-52}\mathrm{ulp}(s) + 2^{-122} + \mathrm{ulp}(2^{-72}) < 2^{-121.192},$$

where 2^{-122} accounts for the rounding error in $\circ(x_\ell + s_\ell)$, here $\circ(q_\ell + L)$.

At line 33, the FastTwoSum error is bounded by $2^{-106}\mathrm{ulp}(y_h)$ using Theorems 9-11 from [3]. Since $|H| <$

356 and $|r_h| < 2^{-18.516}$, we have $|y_h| < 357$, thus the FastTwoSum error is bounded by 2^{-97}. Since $|z_\ell| \le \mathrm{ulp}(357) \le 2^{-44}$ and $|r_\ell| < 2^{-69.413}$ from above, we have $|y_\ell| < 2^{-43.999}$ thus the rounding error on y_ℓ is bounded by 2^{-96}. The total rounding error at line 33 is thus bounded by $2^{-97} + 2^{-96} < 2^{-95.415}$.

Lemma 4 states with $z := z_h + z_\ell$:

$$|z - (x + \sqrt{x^2 - 1})| < 2^{-98.143} x,$$

from which we deduce:

$$|\log(z) - \mathrm{acosh}(x)| < 2^{-98.143} \frac{x}{t},$$

where t lies between z and $x + \sqrt{x^2 - 1}$. Since $t > 1.621x$ for $x \ge x_0$, we deduce:

$$|\log(z) - \mathrm{acosh}(x)| < \frac{2^{-98.143}}{1.621} < 2^{-98.839}.$$

Lemma 5 states that $RT = 1 + x_h + x_\ell + \epsilon$ with $|\epsilon| < 2^{-103.677}$. Thus:

$$\log(z_h + z_\ell) = e\log(2) - \log(R) + \log(1 + x_h + x_\ell + \epsilon),$$

where $e\log(2)/2$ is approximated by the sum $e_{\ell 0} + e_{\ell 1} + e_{\ell 2}$, $-\log(R)/2$ by $H + M + L$ (before the addition of $e_{\ell 0}$ to H), and $\frac{1}{2}\log(1 + x_h + x_\ell)$ by $p_h + p_\ell$. Letting $x := x_h + x_\ell$, we have:

$$\begin{aligned} &|\log(1 + x + \epsilon) - \log(1 + x)| = |\log(1 + \frac{\epsilon}{1 + x})|, \\ &\le 0.502 \frac{|\epsilon|}{1 + x} < 2^{-104.671}, \end{aligned}$$

using again $|\log(1 + u) - u| < 0.502u^2$ for $|u| \le 2^{-8}$.

Wrapping up, the total error between $y_h + y_\ell$ and $\mathrm{acosh}(x)/2$ is bounded by:

$$\begin{aligned} &2^{-99.939} + 2^{-104.671} + 2^{-132.755} + 2^{-141.711} + 2^{-103.939} \\ &+ 2^{-121.678} + 2^{-121.192} + 2^{-95.415} < 2^{-95.347}, \end{aligned}$$

where $2^{-99.939}$ is from the bound for $\log(z) - \mathrm{acosh}(x)$ divided by 2, $2^{-104.671}$ is the error from the approximation $x_h + x_\ell$ (Lemma 5 and application of $\log(1 + u)$), $2^{-132.755}$ bounds the difference between $e_{\ell 0} + e_{\ell 1} + e_{\ell 2}$ and $e\log(2)/2$ (Eq. (15)), $2^{-141.711}$ is the error bound for the computation of $H + M + L$, $2^{-103.939}$ is the error bound for PolyEval, the terms $2^{-121.678}$ and $2^{-121.192}$ account for the adddd error in lines 31 and 32, and $2^{-95.415}$ bounds the rounding error of line 33. Multiplying by 2 yields the bound of the theorem. ∎

Theorem 2: For any binary64 number x, $x \ge 270$, such that $\mathrm{acosh}(x)$ has less than 43 identical bits after the round bit, if $2y_h + 2y_\ell$ is the double-word approximation of $\mathrm{acosh}(x)$ computed by Algorithm AccuratePath, then $\circ(2y_h + 2y_\ell)$ is the correct rounding of $\mathrm{acosh}(x)$.

Proof: Let $y = \mathrm{acosh}(x)$. If y has less than 43 identical bits after the round bit, then y is at distance at least $2^{-44}\mathrm{ulp}(y)$ from a rounding boundary z: $|y - z| \ge 2^{-44}\mathrm{ulp}(y)$. Since $\mathrm{ulp}(y) > 2^{-53}|y|$, this yields $|y - z| \ge 2^{-97}|y|$, and since $x \ge 270$, $y > 6.291$, thus $|y - z| > 2^{-94.347}$. From Theorem 1, there is no rounding boundary between $2y_h + 2y_\ell$ and $\mathrm{acosh}(x)$, which concludes the proof. ∎

Since CORE-MATH contains all hard-to-round inputs with at least 43 identical bits after the round bit, and these inputs yield no failure, we conclude the accurate path is correct for $x \ge 270$.

For $x_0 \le x < 270$, we had to compute more hard-to-round inputs. More precisely, for $12 \le x < 270$, we had to compute hard-to-round inputs with at least 42 identical bits after the round bit, which implies $|y - z| \ge 2^{-96}|y|$ in the proof of Theorem 2, and the lower bound 12 satisfies $2^{-96} \cdot \mathrm{acosh}(12) > 2^{-94.347}$; for $2.6 \le x < 12$, we had to compute hard-to-round inputs with at least 41 identical bits after the round bit, where $2^{-95} \cdot \mathrm{acosh}(2.6) > 2^{-94.347}$; for $1.5 \le x < 2.6$, we had to compute hard-to-round inputs with at least 40 identical bits after the round bit, where $2^{-94} \cdot \mathrm{acosh}(1.5) > 2^{-94.347}$; and for $x_0 \le x < 1.5$, we had to compute hard-to-round inputs with at least 39 identical bits after the round bit, where $2^{-93} \cdot \mathrm{acosh}(x_0) > 2^{-94.347}$. Using BaCSeL, we found 22922 new hard-to-round inputs. All these new inputs are correctly rounded, which proves the accurate path is correct for $x \ge x_0$.

Acknowledgements. Many thanks to Claude-Pierre Jeannerod who checked the correctness of Lemma 1 and pointed out a typo in the corresponding binary32 example.

REFERENCES

[1] BOLDO, S., GRAILLAT, S., AND MULLER, J. On the robustness of the 2sum and fast2sum algorithms. *ACM Trans. Math. Softw. 44*, 1 (2017), 4:1–4:14.

[2] BORGES, C. F., JEANNEROD, C.-P., AND MULLER, J.-M. High-level algorithms for correctly-rounded reciprocal square roots. In *Proceedings of the 29th International Symposium on Computer Arithmetic* (Lyon (virtual meeting due to the COVID pandemic), France, Sept. 2022), Proceedings of the 29th International Symposium on Computer Arithmetic.

[3] JEANNEROD, C.-P., AND ZIMMERMANN, P. FastTwoSum revisited. In *32nd IEEE Symposium on Computer Arithmetic, ARITH 2025, El Paso, TX, USA, May 4-7* (2025).

[4] JOLDES, M., MULLER, J., AND POPESCU, V. Tight and rigorous error bounds for basic building blocks of double-word arithmetic. *ACM Trans. Math. Softw. 44*, 2 (2017).

[5] MULLER, J., AND RIDEAU, L. Formalization of double-word arithmetic, and comments on "tight and rigorous error bounds for basic building blocks of double-word arithmetic". *ACM Trans. Math. Softw. 48*, 1 (2022), 9:1–9:24.

APPENDIX

A. Rounding error in $t_h + t_\ell$

We analyze here the total rounding error in the computation of t_h, t_ℓ in Algorithm ComputeThG, where $t_h + t_\ell$ approximates $x + \sqrt{x^2 - 1}$. We know that $w_h + w_\ell$ is exact, i.e., $w_h + w_\ell = x_{2h} - 1$. The rounding errors are that on i_{sh} at line 5, on s_ℓ at line 6, on the FastTwoSum at line 7, and on t_ℓ at line 8.

We use the following Gappa script, with `RND` replaced by all four rounding modes:

```
@rnd = float<ieee_64,RND>;
X2 = x*x;
x2h rnd= x*x;
```

```
W = X2-1;
wh = x2h-1; # exact
wl = x*x-x2h; # exact
S = sqrt(W);
Sh = sqrt(wh);
sh rnd= sqrt(wh);
Ish = 0.5/wh;
ish rnd= 0.5/wh;
t = -rnd(sh*sh-wh);
U = wl+t;
u rnd= wl+t;
V = 0.5*(1/sh); # V=1/(2sh)
v = sh*ish;
vr rnd= sh*ish;
UV = U*V;
uv rnd= u*v; # uv is sl in the code
# th, ul = fasttwosum(x,sh)
th rnd= x+sh;
z = th-x; # exact
ul = -rnd(th-(x+sh));
tl rnd= ul+uv;
tr = th+tl;
T = x+sh+UV;

{ x in [0x1.1e83e425aee63p+0,111.75]
  -> |tr-T| in ? }

x2h+wl-X2 -> 0;
wh+wl-W -> 0;
v-V ->
  sh*(ish - Ish) + (sh*Ish - 0.5/sh) {sh <> 0};
sh*Ish - 0.5/sh ->
  0.5/(wh*sh) * (sh*sh-wh) {sh <> 0};
#@-Wno-hint-difference
Sh*Sh - wh -> 0;
#@-Whint-difference
th+ul-(x+sh) -> ul-(-(th-(x+sh)));
th+tl-(x+sh+uv) -> (th+ul-(x+sh)) + (tl-(ul+uv));
th+tl-(x+sh+UV) -> (uv-UV) + (th+ul-(x+sh))
  + (tl-(ul+uv)) {sh <> 0};

$ x in 64;
```

This script proves that the rounding error in $t_h + t_\ell$ is bounded by $2^{-94.212}$. This bound is obtained for RZ.