The CORE-MATH sin is correctly rounded

Paul Zimmermann
September 25, 2026

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

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

We consider the CORE-MATH `sin.c` file from commit `6b84457`, and the four rounding modes from the C23 standard: to nearest (ties to even), toward zero, toward $-\infty$, toward $+\infty$.

I. Branch $0 \le |x| \le x_0$

Let $x_0 =$ `0x1.7137449123ef6p-26`. For $x = \pm 0$, the CORE-MATH function returns x, which is the correct rounding; and for $0 < |x| \le x_0$, it returns $\mathrm{fma}(x, -2^{-54}, x)$. This special branch is only needed to avoid a spurious underflow in the polynomial evaluation used in the other branches.

Lemma 1: For $0 < |x| \le x_0$, $\mathrm{fma}(x, -2^{-54}, x)$ yields the correct rounding of $\sin(x)$, whatever the rounding mode.

Proof: Since the formula $\mathrm{fma}(x, -2^{-54}, x)$ is an odd function, like $\sin(x)$, it suffices to consider $0 < x \le x_0$. First, for $0 < x < 1$, we have:

$$x - \frac{x^3}{6} < \sin(x) < x. \tag{1}$$

First consider $0 < x \le 2^{-26}$. Then if follows from Eq. (1) $x - 2^{-54}x < \sin(x) < x$. Since $\mathrm{ulp}(x) > 2^{-53}x$ (even in the subnormal range), we deduce $x - \frac{1}{2}\mathrm{ulp}(x) < \sin(x) < x$, whence $\sin(x)$ and $x - 2^{-54}x$ are in $(\mathrm{nextdown}(x), x)$, thus round to the same value, except maybe when x is a power of two, since then $\mathrm{nextdown}(x) = x - \frac{1}{2}\mathrm{ulp}(x)$. The case $x = \pm 2^k$, $-1074 \le k \le -26$, is checked exhaustively.

Now consider $2^{-26} < x \le x_0$. We have $\mathrm{ulp}(x) = 2^{-78}$, and we can check that $x^3/3 < 2^{-79} = \frac{1}{2}\mathrm{ulp}(x)$. Again, both $\sin(x)$ and $x - 2^{-54}x$ lie in $(\mathrm{nextdown}(x), x)$, and the proof proceeds as above. ∎

The bound x_0 is optimal for a formula like $\mathrm{fma}(x, -2^{-54}, x)$. Indeed, for $x = \mathrm{nextup}(x_0)$, $\sin(x)$ rounds to $\mathrm{nextdown}(x)$ to nearest, whereas $\mathrm{fma}(x, -2^{-54}, x)$ rounds to x. And if we use $\mathrm{fma}(x, m, x)$ with a different value of m such that it rounds to $\mathrm{nextdown}(x)$ for $x = \mathrm{nextup}(x_0)$, then by linearity of the formula, it would round to $\mathrm{nextdown}(x/2)$ for $x = \mathrm{nextup}(x_0/2)$, which would be wrong.

II. Branch $x_0 < |x| < 2^{31}$

In this branch, Algorithm ModerateFastPath is used. The constant i approximates $2^{14}/\pi$:

$$|i - \frac{2^{14}}{\pi}| < 2^{-41.496},$$

Algorithm 1 (ModerateFastPath)

Input: x a binary64 number, $x_0 < |x| < 2^{31}$
Output: the correct rounding $\circ(\sin x)$ or FAIL

1: $s \leftarrow \mathrm{sign}(x)$
2: $i =$ `0x1.45f306dc9c883p+12`
3: $p_h =$ `-0x1.921fb54442d18p-13`
4: $p_\ell =$ `-0x1.1a62633145c07p-67`
5: $k \leftarrow \mathrm{round}(\circ(i \cdot |x|))$
6: $r_h \leftarrow \mathrm{fma}(k, p_h, |x|), \quad r_\ell \leftarrow \circ(k \cdot p_\ell)$
7: $r = \circ(r_h + r_\ell)$
8: $r_2 \leftarrow \circ(r \cdot r)$
9: write $k \bmod 2^{15} = 2^{14}t + 2^7 i_1 + i_2$ with $t \in \{0, 1\}$ and $0 \le i_1, i_2 < 2^7$
10: if $t = 1$ then $s \leftarrow -s$
11: $s_{1h}, s_{1\ell} \leftarrow \mathrm{muldd}(S_{1h}[i_1], S_{1\ell}[i_1], C_{2h}[i_2], C_{2\ell}[i_2])$
12: $s_{2h}, s_{2\ell} \leftarrow \mathrm{muldd}(S_{2h}[i_2], S_{2\ell}[i_2], C_{1h}[i_1], C_{1\ell}[i_1])$
13: $S_h, S_\ell \leftarrow \mathrm{fastsum}(s_{1h}, s_{1\ell}, s_{2h}, s_{2\ell})$
14: $C_h \leftarrow \circ(\circ(C_{1h}[i_1]C_{2h}[i_2]) - \circ(S_{1h}[i_1]S_{2h}[i_2]))$
15: $s_1 = 1.0, \quad s_3 =$ `-0x1.55555553068fp-3`
16: $c_2 = -0.5, \quad c_4 =$ `0x1.55555553bfd3p-5`
17: $s_h \leftarrow r \cdot (s_1 + s_3 \cdot r_2)$
18: $c_h \leftarrow r_2 \cdot (c_2 + c_4 \cdot r_2)$
19: $f_h \leftarrow S_h, \quad f_\ell \leftarrow S_\ell + S_h c_h + C_h s_h$
20: $f_h \leftarrow s \cdot f_h, \quad f_\ell \leftarrow s \cdot f_\ell$
21: $\epsilon \leftarrow$ `0x1.dep-64`
22: $\ell_b \leftarrow f_h + \circ(f_\ell - \epsilon), \quad u_b \leftarrow f_h + \circ(f_\ell + \epsilon)$
23: if $\circ(\ell_b) = \circ(u_b)$ return $\circ(\ell_b)$ else return FAIL

while the constants p_h and p_ℓ approximate $-\pi/2^{14}$:

$$|p_h + p_\ell + \frac{\pi}{2^{14}}| < 2^{-122.041},$$

and $|p_h + \pi/2^{14}| < 2^{-66.858}$. At lines 17-19, we omitted roundings for clarity, where in line 19, the terms in f_ℓ are accumulated from left to right, as prescribed by the C language. In Algorithm ModerateFastPath, $S_h + S_\ell$ approximates $\sin(z)$ and C_h approximates $\cos(z)$, where $z = t_1 + t_2$, $t_1 = i_1\pi/2^7$ and $t_2 = i_2\pi/2^{14}$.

The tables S_{1h}, $S_{1\ell}$, C_{1h}, $C_{1\ell}$, S_{2h}, $S_{2\ell}$, C_{2h}, $C_{2\ell}$ are defined as follows:

- for $0 \le i_1 < 2^7$, $S_{1h} + S_{1\ell}$ is a double-double approximation of $\sin(t_1)$, and $C_{1h} + C_{1\ell}$ is a double-double approximation of $\cos(t_1)$, where $t_1 = i_1\pi/2^7$;
- for $0 \le i_2 < 2^7$, $S_{2h} + S_{2\ell}$ is a double-double approximation of $\sin(t_2)$, and $C_{2h} + C_{2\ell}$ is a double-double approximation of $\cos(t_2)$, where $t_2 = i_2\pi/2^{14}$).

Algorithm 2 (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: $h \leftarrow \circ(x_h c_h)$
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: $\ell \leftarrow \circ(\circ(\ell_1 + \ell_2) + \ell_3)$

Algorithm 3 (fasttwosum) from [1]

Input: a, b binary64 numbers
Output: x, y such that $x + y$ approximates $a + b$
1: $x \leftarrow \circ(a + b)$
2: $z \leftarrow \circ(x - a)$
3: $y \leftarrow \circ(b - z)$

Lemma 2: The integer k computed at line 5 of Algorithm ModerateFastPath satisfies:

$$|x| = \frac{k\pi}{2^{14}} + y,$$

with $k \le 2^{44}$ and $|y| < 2^{-13.340}$.
Proof: Since $|x| < 2^{31}$, and $i < 2^{13}$, $|ix| < 2^{44}$, thus $i \cdot |x|$ does not overflow at line 5, and k approximates $2^{14}|x|/\pi$. Let $u = \circ(i \cdot |x|)$.

$$\begin{aligned} |k - \frac{2^{14}|x|}{\pi}| &\le |k - u| + |u - i|x|| + |x||i - \frac{2^{14}}{\pi}| \\ &\le \frac{1}{2} + \mathrm{ulp}(u) + 2^{-41.496}|x|, \end{aligned}$$

since $|i - 2^{14}/\pi| < 2^{-41.496}$. Since $|ix| < 2^{44}$, $\mathrm{ulp}(u) \le \mathrm{ulp}(2^{43}) = 2^{-9}$. The term $2^{-41.496}|x|$ is bounded by $2^{-41.496} \cdot 2^{31} < 2^{-10.496}$. We thus get:

$$|k - \frac{2^{14}|x|}{\pi}| < \frac{1}{2} + 2^{-8.562}.$$

Multiplying by $\pi/2^{14}$, we obtain:

$$|x| = \frac{k\pi}{2^{14}} + y,$$

with $|y| < (1/2 + 2^{-8.562})\frac{\pi}{2^{14}} < 2^{-13.340}$. ∎

Lemma 3: At line 6, r_h is exact, i.e., $r_h = k \cdot p_h + |x|$, and after line 7:

$$|r - y| < 2^{-65.999},$$

with $|r|, |r_h| < 2^{-13.339}$.
Proof: If $|x| < 2^{-14}$, then $|\circ(i \cdot |x|)| < 1/2$ thus $k = 0$ and $r_h = |x|$ is exact. Now assume $|x| \ge 2^{-14}$. Then $|x|$ is an integer multiple of $\mathrm{ulp}(2^{-14}) = 2^{-66}$. Now p_h is an integer multiple of 2^{-62}, and since k is an integer, $k \cdot p_h + |x|$ is an integer multiple of 2^{-66}. Now let us bound $k \cdot p_h + |x|$ using $|x| = k\pi/2^{14} + y$:

$$\begin{aligned} |kp_h + |x|| &= k|p_h + \frac{\pi}{2^{14}}| + |y| \\ &\le k \cdot 2^{-66.858} + 2^{-13.340} < 2^{-13.339}, \end{aligned}$$

Algorithm 4 (fastsum)

Input: x_h, x_ℓ, y_h, y_ℓ binary64 numbers
Output: h, ℓ such that $h + \ell$ approximates $x_h + x_\ell + c_h + c_\ell$
1: $h, t \leftarrow \mathrm{fasttwosum}(x_h, y_h)$
2: $\ell \leftarrow \circ(\circ(x_\ell + y_\ell) + t)$

since $k < 2^{44}$. Now $k \cdot p_h + |x|$ can be written $m \cdot 2^{-66}$ with m integer, and $|m| < 2^{-13.339}/2^{-66} < 2^{53}$, thus $k \cdot p_h + |x|$ is exact.

Since $k < 2^{44}$ and $|p_\ell| < 2^{-66}$, we have $|r_\ell| < 2^{-22}$ and the rounding error on r_ℓ is bounded by $\mathrm{ulp}(2^{-23}) = 2^{-75}$. Now recall that $|r_h| := |kp_h + |x|| < 2^{-13.339}$, thus the rounding error on $r_h + r_\ell$ at line 7 is bounded by $\mathrm{ulp}(2^{-13.339} + 2^{-22}) = 2^{-66}$. In summary:

$$\begin{aligned} |r - y| &\le |r - (r_h + r_\ell)| + |r_\ell - kp_\ell| + |kp_h + |x| + kp_\ell - y| \\ &\le 2^{-66} + 2^{-75} + k|p_h + p_\ell + \frac{\pi}{2^{14}}| \\ &\le 2^{-66} + 2^{-75} + 2^{44} \cdot 2^{-122.041} < 2^{-65.996}. \end{aligned}$$

It follows $|r| < |y| + 2^{-65.996} < 2^{-13.339}$. ∎

The bound $|r| < 2^{-13.339}$ is tight, since for $x =$ `0x1.fe8cd2d79106cp+30` and rounding upward, we get $r \approx 2^{-13.3417}$.

Theorem 1: For $x_0 < |x| < 2^{31}$, and any rounding mode, Algorithm ModerateFastPath is correct, i.e., if not FAIL, it delivers $\circ(\sin(x))$.
Proof: Since s_{1h} and s_{2h} only depend on i_1, i_2 and the rounding mode, we can check by exhaustive search whether the FastTwoSum condition is fulfilled in $\mathrm{fastsum}(s_{1h}, s_{1\ell}, s_{2h}, s_{2\ell})$, which calls $\mathrm{fasttwosum}(s_{1h}, s_{2h})$: this is the case. Similarly, since S_h, S_ℓ, C_h only depend on i_1, i_2 and the rounding mode, we can determine by exhaustive search a bound on the difference between $S_h + S_\ell$ and $\sin(t_1 + t_2)$, and between C_h and $\cos(t_1 + t_2)$, where $t_1 = i_1\pi/2^7$ and $t_2 = i_2\pi/2^{14}$. The largest error for $\sin(t_1 + t_2)$ is obtained for $i_1 = 47$, $i_2 = 36$ and rounding upward:

$$|S_h + S_\ell - \sin(t_1 + t_2)| < 2^{-103.342}, \tag{2}$$

while the largest error for $\cos(t_1 + t_2)$ is obtained for $i_1 = 106$, $i_2 = 73$ and rounding downward:

$$|C_h - \cos(t_1 + t_2)| < 2^{-51.569}.$$

This exhaustive search also shows that $|S_h| \le 1$, $|S_\ell| < 2^{-51.569}$ and $|C_h| \le 1$.

For $|r| < 2^{-13.339}$, Sollya proves that the difference between $\sin(r)$ and $s_1 r + s_3 r^3$ is bounded by $2^{-76.494}$, and that between $\cos(r) - 1$ and $c_2 r^2 + c_4 r^4$ is bounded by $2^{-92.723}$. Gappa (we used version 1.8.2 everywhere in this document) proves that the rounding error on s_h is bounded by $2^{-65.159}$, and that on c_h is bounded by $2^{-79.299}$. We deduce:

$$\begin{aligned} |s_h - \sin(r)| &< 2^{-76.494} + 2^{-65.159} < 2^{-65.158}, \\ |c_h - (\cos(r) - 1)| &< 2^{-92.723} + 2^{-79.299} < 2^{-79.298}. \end{aligned}$$

Now since $|r-y| < 2^{-65.999}$, $\sin(r)-\sin(y) = (r-y)\cos(\xi_1)$ and $\cos(r)-\cos(y) = (y-r)\sin(\xi_2)$ for $\xi_1, \xi_2 \in (r, y)$, thus:

$$\begin{aligned} |s_h - \sin(y)| &< 2^{-65.158} + 2^{-65.999} < 2^{-64.518}, \\ |c_h - (\cos(y) - 1)| &< 2^{-79.298} + 2^{-65.999} \cdot 2^{-13.339} \\ &< 2^{-78.317}. \end{aligned}$$

Let $x' = x \bmod \pi$. Then $x' = z + y$:

$$\begin{aligned} &|S_h + S_\ell + S_h c_h + C_h s_h - \sin(x')| \\ &\le |S_h + S_\ell - \sin(z)| + |S_h c_h - \sin(z)(\cos(y) - 1)| \\ &+ |C_h s_h - \cos(z)\sin(y)|. \end{aligned}$$

The first term of the right-hand side $|S_h + S_\ell - \sin(z)|$ is bounded by $2^{-103.342}$ from Eq. (2). The second term is bounded as follows:

$$\begin{aligned} &|S_h c_h - \sin(z)(\cos(y) - 1)| \\ &\le |S_h||c_h - (\cos(y) - 1)| + |S_h - \sin(z)||\cos(y) - 1| \\ &\le 2^{-78.317} + (2^{-51.569} + 2^{-103.342}) \cdot 2^{-27.680} < 2^{-77.709}, \end{aligned}$$

since $|S_h| \le 1$, $|S_h - \sin(z)| \le |S_\ell| + 2^{-103.342}$, and $|y| \le 2^{-13.340}$. Finally, the last term is bounded by:

$$\begin{aligned} &|C_h s_h - \cos(z)\sin(y)| \\ &\le |C_h||s_h - \sin(y)| + |C_h - \cos(z)||\sin(y)| \\ &\le 2^{-64.518} + 2^{-51.569} \cdot 2^{-13.340} < 2^{-63.700}. \end{aligned}$$

Summing up, we obtain:

$$\begin{aligned} &|S_h + S_\ell + S_h c_h + C_h s_h - \sin(x')| \\ &< 2^{-103.342} + 2^{-77.709} + 2^{-63.700} < 2^{-63.699}. \end{aligned}$$

We still have to take into account the rounding errors while computing f_ℓ in line 19, and that in $\circ(f_\ell \pm \epsilon)$ in line 22. Since $|S_\ell| < 2^{-51.569}$, $|S_h| \le 1$, $|c_h| \le |\cos(y) - 1| + 2^{-78.320} < 2^{-27.680}$, the rounding error on $S_h c_h$ is bounded by $\mathrm{ulp}(2^{-27.680}) = 2^{-80}$, and that on $S_\ell + \circ(S_h c_h)$ is bounded by $\mathrm{ulp}(2^{-26.679}) = 2^{-80}$ too. Now since $|C_h| \le 1$ and $|s_h| \le |\sin(y)| + 2^{-64.518} < 2^{-13.340}$, the rounding error on $C_h s_h$ is bounded by $\mathrm{ulp}(2^{-14}) = 2^{-66}$. Let $\alpha = \circ(S_\ell + \circ(S_h c_h))$ and $\beta = \circ(C_h s_h)$. The sum $\alpha + \beta$ is thus bounded in absolute value by:

$$|\cos(y) - 1| + |\sin(y)| + 2 \cdot 2^{-80} + 2^{-66} < 2^{-13.339},$$

thus the rounding error on $f_\ell = \circ(\alpha + \beta)$ is bounded by $\mathrm{ulp}(2^{-14}) = 2^{-66}$ too. Finally adding $\pm\epsilon$ to f_ℓ does not make it exceed 2^{-13} in absolute value either, thus the rounding error in $\circ(f_\ell \pm \epsilon)$ is bounded by 2^{-66} too. In fact, since $\epsilon = 7.46875 \cdot \mathrm{ulp}(2^{-14})$, the rounding error on $f_\ell \pm \epsilon$ is bounded by $0.53125 \cdot \mathrm{ulp}(2^{-14}) < 2^{-66.912}$, which is the maximal distance from ϵ to a multiple of $\mathrm{ulp}(f_\ell)$ (if f_ℓ or $f_\ell \pm \epsilon$ are in a smaller binade, this bound also holds). The accumulated rounding errors in lines 19 and 22 are thus bounded by:

$$2 \cdot 2^{-80} + 2 \cdot 2^{-66} + 2^{-66.912} < 2^{-64.659}.$$

We deduce (after fixing the sign of $\sin(x)$):

$$\ell_b - \sin(x) < -\epsilon + 2^{-63.699} + 2^{-64.659} < 0,$$

and similary $u_b - \sin(x) > 0$, thus ℓ_b and u_b enclose $\sin(x)$, and if they round to the same value, so does $\sin(x)$. ∎

III. Branch $|x| \ge 2^{31}$

In this branch, the fast path first reduces the argument using Payne-Hanek's algorithm [3], as detailed in Algorithm ReduceLarge.

Algorithm 5 (ReduceLarge)

Input: x a binary64 number, $x \ge 2^{31}$
Output: integer k and binary64 number r such that $x/(2\pi) \bmod 1 \approx k/2^{15} + r$

1: use a precomputed table T such that $\sum_i T_i/2^{64i}$ is the 64-bit expansion of $1/(2\pi)$
2: write x as $m \cdot 2^e$ with $2^{52} \le m < 2^{53}$
3: $i \leftarrow \lceil (e+1)/64 \rceil$
4: $f \leftarrow e \bmod 64$
5: $U_0 \leftarrow (T_i, T_{i+1})$ shifted by f bits to the left
6: $U_1 \leftarrow (T_{i+1}, T_{i+2})$ shifted by f bits to the left
7: $u \leftarrow U_0 \cdot 2^{64} + U_1$ ▷ 128-bit variable
8: $v \leftarrow m \cdot u$ ▷ 128-bit multiplication
9: $v \leftarrow v + (2^{112} + 2^{59} + 2^{52}) \bmod 2^{128}$
10: write $v = v_h \cdot 2^{113} + v_\ell$ with $0 \le v_\ell < 2^{113}$
11: $k \leftarrow v_h, \quad r \leftarrow 2^{-68}\lfloor 2^{-60} v_\ell \rfloor - 2^{-16}$

Lemma 4: Let x be a binary64 number, $x \ge 2^{31}$. Then the values k and r returned by Algorithm ReduceLarge satisfy:

$$\frac{x}{2\pi} \mod 1 = \frac{k}{2^{15}} + r + s, \tag{3}$$

with $|r| \le 2^{-16}$ and $|s| < 2^{-68.988}$.

Proof: Since m defined at line 2 is an integer, and so is T_i ($0 \le T_i < 2^{64}$), the contribution of T_i to the fractional part of $x/(2\pi)$ is $mT_i 2^{e-64i}$. Thus if $e - 64i \ge 0$, $mT_i 2^{e-64i}$ is an integer, and its contribution is zero. Conversely, if $e - 64i$ is small, its contribution is negligible. This is the heart of the Payne-Hanek algorithm: only a few values of i are useful. The value i computed at line 3 is such that $mT_i 2^{e-64i}$ is the first to contribute to the fractional part. The value f computed at line 4 is such that the lower $64-f$ bits from T_i do contribute. Lines 5 and 6 shift the bits of the table T, so that mU_0 contributes to exactly 64 bits to the fractional part: the product mU_0 has $53 + 64 = 117$ bits at most, its upper bits contributing to the integer part of $x/(2\pi)$. Similarly, the product mU_1 has at most 117 bits: it contributes to bits of weight 2^{-128} to 2^{-12} of the fractional part, thus at most to 2^{-11}, where $-11 = 53 - 64$. The ignored part $m(U_2 + 2^{-64}U_3 + \cdots)$ contributes at most to 2^{-75} to the fractional part, where $-75 = 53 - 2 \cdot 64$. We thus have with u the 128-bit variable defined at line 7:

$$\mathrm{frac}(2^{-128}mu) \le \mathrm{frac}(\frac{x}{2\pi}) < \mathrm{frac}(2^{-128}mu) + 2^{-75}.$$

The variable k captures the upper 15 bits from the product mu, while r captures the following 53 bits (with possibly upper bits being zero). Without line 9, the truncation error on r at line 11 would be less than 2^{-68}, thus at the end we would have:

$$\frac{x}{2\pi} \mod 1 = \frac{k}{2^{15}} + r + s,$$

with $0 \le s < 2^{-68} + 2^{-75} < 2^{-67.988}$, with $0 \le r < 2^{-15}$. The addition of 2^{112} to v in line 9 and the subtraction of 2^{-16} to r in line 11 round r between -2^{-16} and 2^{-16} instead of between 0 and 2^{-15}. Similarly, the addition of $2^{59} + 2^{52}$ takes into account half of the maximal value of s. In case $v + 2^{112} + 2^{59} + 2^{52} \ge 2^{128}$, the value of v wraps around; this is not a problem since the ignored carry does not change $x/(2\pi) \bmod 1$. ∎

The fast path for this branch is detailed in Algorithm LargePath.

Algorithm 6 (LargePath)

Input: x a binary64 number, $|x| \ge 2^{31}$
Output: the correct rounding $\circ(\sin x)$ or FAIL

1: $s \leftarrow \text{sign}(x), \quad k, r \leftarrow \text{ReduceLarge}(|x|)$
2: write $k = 2^{14}t + 2^7 i_1 + i_2$ with $t \in \{0, 1\}$, $0 \le i_1, i_2 < 2^7$
3: if $t = 1$ then $s \leftarrow -s$
4: $s_{1h}, s_{1\ell} \leftarrow \text{muldd}(S_{1h}[i_1], S_{1\ell}[i_1], C_{2h}[i_2], C_{2\ell}[i_2])$
5: $s_{2h}, s_{2\ell} \leftarrow \text{muldd}(S_{2h}[i_2], S_{2\ell}[i_2], C_{1h}[i_1], C_{1\ell}[i_1])$
6: $S_h, S_\ell \leftarrow \text{fastsum}(s_{1h}, s_{1\ell}, s_{2h}, s_{2\ell})$
7: $C_h \leftarrow \circ(\circ(C_{1h}[i_1]C_{2h}[i_2]) - \circ(S_{1h}[i_1]S_{2h}[i_2]))$
8: $s_1 =$ `0x1.921fb54442d18p2`
9: $s_3 =$ `-0x1.4abbcdb6b26d1p5`
10: $c_2 =$ `-0x1.3bd3cc9be45dep4`
11: $c_4 =$ `0x1.03c1eee483083p6`
12: $r_2 \leftarrow \circ(r \cdot r)$
13: $s_h \leftarrow r \cdot (s_1 + s_3 \cdot r_2)$
14: $c_h \leftarrow r_2 \cdot (c_2 + c_4 \cdot r_2)$
15: $f_h \leftarrow S_h, \quad f_\ell \leftarrow S_\ell + S_h c_h + C_h s_h$
16: $f_h \leftarrow s \cdot f_h, \quad f_\ell \leftarrow s \cdot f_\ell$
17: $\epsilon \leftarrow$ `0x1.41p-63`
18: $\ell_b \leftarrow f_h + \circ(f_\ell - \epsilon), \quad u_b \leftarrow f_h + \circ(f_\ell + \epsilon)$
19: if $\circ(\ell_b) = \circ(u_b)$ return $\circ(\ell_b)$ else return FAIL

Theorem 2: For $|x| \ge 2^{31}$, and any rounding mode, Algorithm LargePath is correct, i.e., if not FAIL, it delivers $\circ(\sin(x))$.

Proof: From Lemma 4, we have

$$\frac{x}{2\pi} \mod 1 = \frac{k}{2^{15}} + r + s,$$

with $|r| \le 2^{-16}$ and $|s| < 2^{-68.988}$. Let $x' = (-1)^t x$, with t defined in line 2. We deduce since $k = 2^{14}t + 2^7 i_1 + i_2$:

$$\sin(x') = \sin(z + y),$$

with $z = t_1 + t_2$, $t_1 = \pi i_1/2^7$, $t_2 = \pi i_2/2^{14}$, and $y = 2\pi(r + s)$.

For $|r| \le 2^{-16}$, Sollya proves that the difference between $\sin(2\pi r)$ and $s_1 r + s_3 r^3$ is bounded by $2^{-69.373}$, and that between $\cos(2\pi r) - 1$ and $c_2 r^2 + c_4 r^4$ is bounded by $2^{-83.695}$. Gappa proves that the rounding error on s_h is bounded by $2^{-63.999}$, and that on c_h is bounded by $2^{-76.999}$. We deduce:

$$\begin{aligned} |s_h - \sin(2\pi r)| &< 2^{-69.373} + 2^{-63.999} < 2^{-63.964}, \\ |c_h - (\cos(2\pi r) - 1)| &< 2^{-83.695} + 2^{-76.999} < 2^{-76.985}. \end{aligned}$$

Now since $|2\pi r - y| = |2\pi s| < 2\pi 2^{-68.988} < 2^{-66.336}$, $\sin(2\pi r) - \sin(y) = (2\pi r - y)\cos(\xi_1)$ and $\cos(2\pi r) - \cos(y) = (y - 2\pi r)\sin(\xi_2)$ for $\xi_1, \xi_2 \in (2\pi r, y)$, thus:

$$\begin{aligned} |s_h - \sin(y)| &< 2^{-63.964} + 2^{-66.336} < 2^{-63.709}, \\ |c_h - (\cos(y) - 1)| &< 2^{-76.985} + 2^{-66.336} \cdot 2^{-13.348} \\ &< 2^{-76.778}, \end{aligned}$$

since $|y| < 2\pi r + 2^{-66.336} < 2^{-13.348}$, and $|\sin(y)| \le |y|$.

Then since $x' = z + y$:

$$\begin{aligned} &|S_h + S_\ell + S_h c_h + C_h s_h - \sin(x')| \\ &\le |S_h + S_\ell - \sin(z)| + |S_h c_h - \sin(z)(\cos(y) - 1)| \\ &+ |C_h s_h - \cos(z)\sin(y)|. \end{aligned}$$

The first term of the right-hand side $|S_h + S_\ell - \sin(z)|$ is bounded by $2^{-103.342}$ from Eq. (2). Indeed, it only depends on the precomputed values, which are the same as in Algorithm ModerateFastPath. The second term is bounded as follows:

$$\begin{aligned} &|S_h c_h - \sin(z)(\cos(y) - 1)| \\ &\le |S_h||c_h - (\cos(y) - 1)| + |S_h - \sin(z)||\cos(y) - 1| \\ &\le 2^{-76.778} + (2^{-51.569} + 2^{-103.342}) \cdot 2^{-27.696} < 2^{-76.541}, \end{aligned}$$

since $|S_h| \le 1$, $|S_h - \sin(z)| \le |S_\ell| + 2^{-103.342}$, and $|y| \le 2^{-13.348}$. Finally, the last term is bounded by:

$$\begin{aligned} &|C_h s_h - \cos(z)\sin(y)| \\ &\le |C_h||s_h - \sin(y)| + |C_h - \cos(z)||\sin(y)| \\ &\le 2^{-63.709} + 2^{-51.569} \cdot 2^{-13.348} < 2^{-63.190}. \end{aligned}$$

Summing up, we obtain:

$$\begin{aligned} &|S_h + S_\ell + S_h c_h + C_h s_h - \sin(x')| \\ &< 2^{-103.342} + 2^{-76.541} + 2^{-63.190} < 2^{-63.189}. \end{aligned}$$

We still have to take into account the rounding errors while computing f_ℓ in line 15, and that in $\circ(f_\ell \pm \epsilon)$ in line 18. Since $|S_\ell| < 2^{-51.569}$, $|S_h| \le 1$, $|c_h| \le |\cos(y) - 1| + 2^{-76.778} < 2^{-27.696}$, the rounding error on $S_h c_h$ is bounded by $\text{ulp}(2^{-27.696}) = 2^{-80}$, and that on $S_\ell + \circ(S_h c_h)$ is bounded by $\text{ulp}(2^{-27.695}) = 2^{-80}$ too. Now since $|C_h| \le 1$ and $|s_h| \le |\sin(y)| + 2^{-63.709} < 2^{-13.348}$, the rounding error on $C_h s_h$ is bounded by $\text{ulp}(2^{-14}) = 2^{-66}$. Let $\alpha = \circ(S_\ell + \circ(S_h c_h))$ and $\beta = \circ(C_h s_h)$. The sum $\alpha + \beta$ is thus bounded in absolute value by:

$$|\cos(y) - 1| + |\sin(y)| + 2 \cdot 2^{-80} + 2^{-66} < 2^{-13.347},$$

thus the rounding error on $f_\ell = \circ(\alpha + \beta)$ is bounded by $\text{ulp}(2^{-14}) = 2^{-66}$ too. Finally adding $\pm\epsilon$ to f_ℓ does not make it exceed 2^{-13} in absolute value either, thus the rounding error in $\circ(f_\ell \pm \epsilon)$ is bounded by 2^{-66} too. The accumulated rounding errors in lines 15 and 18 are thus bounded by:

$$2 \cdot 2^{-80} + 3 \cdot 2^{-66} < 2^{-64.414}.$$

We deduce (after fixing the sign of $\sin(x)$):

$$\ell_b - \sin(x) < -\epsilon + 2^{-63.189} + 2^{-64.414} < 0,$$

and similary $u_b - \sin(x) > 0$, thus ℓ_b and u_b enclose $\sin(x)$, and if they round to the same value, so does $\sin(x)$. ∎

IV. Accurate path

There are two algorithms for the accurate path. For $x_0 < |x| < 2^{-16}$, Algorithm SmallAccurate is used, otherwise for $|x| \geq 2^{-16}$, Algorithm LargeAccurate is used.

A. Branch $x_0 < |x| < 2^{-16}$

In this branch, we use a very simple algorithm (SmallAccurate).

Algorithm 7 (SmallAccurate)

Input: x a binary64 number, $x_0 < |x| \leq 2^{-16}$
Output: the correct rounding of $\sin x$

1: $c_{3h} =$ `-0x1.5555555555555p-3`
2: $c_{3\ell} =$ `-0x1.55554b00de7e8p-57`
3: $c_5 =$ `0x1.111111110848p-7`
4: $x_{2h} \leftarrow \circ(x \cdot x)$
5: $x_{2\ell} \leftarrow \mathrm{fma}(x, x, -x_{2h})$
6: $h_0 \leftarrow \circ(c_5 \cdot x_{2h})$
7: $h_1 \leftarrow \circ(c_{3\ell} + h_0)$
8: $h_2, \ell_2 \leftarrow \mathrm{fasttwosum}(c_{3h}, h_1)$
9: $h_3, \ell_3 \leftarrow \mathrm{muldd}(h_2, \ell_2, x_{2h}, x_{2\ell})$
10: $h_4, \ell_4 \leftarrow \mathrm{muldd}(h_3, \ell_3, x, 0)$
11: $h, t \leftarrow \mathrm{fasttwosum}(x, h_4)$
12: $\ell \leftarrow \circ(\ell_4 + t)$
13: return $\circ(h + \ell)$

Theorem 3: Given a binary64 number x, $x_0 < |x| < 2^{-16}$, Algorithm SmallAccurate returns the correct rounding of $\sin x$, whatever the rounding mode.

Proof: Let $S(x) = x + (c_{3h} + c_{3\ell})x^3 + c_5x^5$. Sollya proves the following bound for $x_0 < |x| < 2^{-16}$:

$$|S(x) - \sin(x)| \leq 2^{-112.743}|\sin x|.$$

We know that $x_{2h} + x_{2\ell} = x^2$, whatever the rounding mode.

We analyzed the rounding errors with a Gappa script, which proves the following bound for the final values h, ℓ of Algorithm SmallAccurate (before the return instruction):

$$|h + \ell - S(x)| \leq 2^{-103}|S(x)|.$$

We used in this script the fact that the error in the two FastTwoSum instructions is bounded respectively by $\mathrm{ulp}_{106}(h_2) \leq 2^{-105}|h_2|$ and $\mathrm{ulp}_{106}(h) \leq 2^{-105}|h|$.

Combining these two bounds yields:

$$|h + \ell - \sin(x)| \leq 2^{-102.998}|\sin x|.$$

If $y := \sin x$ has less than 48 identical bits after the round bit, then y is at distance at least $2^{-49}\mathrm{ulp}(y) \geq 2^{-102}|y|$ from a rounding boundary, since $\mathrm{ulp}(y) \geq 2^{-53}|y|$. Thus $h + \ell$ is closer from $\sin x$ than any rounding boundary, and as a consequence $h + \ell$ correctly rounds to $\circ(\sin x)$.

If $\sin x$ has 48 or more identical bits after the round bit, it is contained in the `sin.wc` file, and correct rounding is ensured since the CORE-MATH test suite reports no failure. ∎

B. Branch $|x| \geq 2^{-16}$

In this branch, the accurate path first performs an argument reduction (ReduceAccLarge) using Payne-Hanek's algorithm, then calls Algorithm AccurateLarge.

Algorithm 8 (ReduceAccLarge)

Input: x a binary64 number, $x \geq 2^{31}$
Output: integer k, sign bit σ and 128-bit integer r such that $x/(2\pi) \bmod 1 \approx k/2^{13} + (-1)^\sigma \cdot r/2^{128}$

1: use a precomputed table T such that $\sum_i T_i/2^{64i}$ is the 64-bit expansion of $1/(2\pi)$
2: write x as $m \cdot 2^e$ with $2^{52} \leq m < 2^{53}$
3: $i \leftarrow \lceil (e+1)/64 \rceil$
4: $f \leftarrow e \bmod 64$
5: $U_0 \leftarrow (T_i, T_{i+1})$ shifted by f bits to the left
6: $U_1 \leftarrow (T_{i+1}, T_{i+2})$ shifted by f bits to the left
7: $U_2 \leftarrow (T_{i+2}, T_{i+3})$ shifted by f bits to the left
8: $u \leftarrow U_0 \cdot 2^{64} + U_1$ ▷ 128-bit variable
9: $v \leftarrow (m \cdot u + \lfloor (m \cdot U_2)/2^{64} \rfloor) \bmod 2^{128}$
10: write $v = k \cdot 2^{115} + R$ with $0 \leq R < 2^{115}$
11: if $r < 2^{114}$ then $\sigma \leftarrow 0$
12: else $k \leftarrow (k+1) \bmod 2^{13}$, $\sigma \leftarrow 1$, $r \leftarrow 2^{115} - r$

Lemma 5: Let x be a binary64 number, $x \geq 2^{31}$. Let k, σ, r be the values returned by Algorithm ReduceAccLarge, and denote $R = (-1)^\sigma r/2^{128}$, then:

$$\frac{x}{2\pi} \mod 1 = \frac{k}{2^{13}} + R + s,$$

with $|R| \leq 2^{-14}$ and $0 \leq s < 2^{-127.999}$.

Proof: The beginning of the proof is the same as in Lemma 4: the product mU_0 contributes to exactly 64 bits to the fractional part of $x/(2\pi)$, of weights 2^{-64} to 2^{-1}; the product mU_1 contributes to bits of weight 2^{-128} to at most 2^{-12} of the fractional part (since m has 53 bits, mU_1 has 117 bits at most); and the additional product mU_2 contributes to bits of weight 2^{-192} to at most 2^{-76} of the fractional part. If $U_3, U_4, ...$ are the next (ignored) shifted words, their contribution is less than $m(U_3 + U_4 2^{-64} + U_5 2^{-128} + ...)2^{-256} < 2^{-139}$. Let $mU_2 = h2^{64} + \ell$ with $0 \leq h, \ell < 2^{64}$. Then:

$$\frac{x}{2\pi} = \frac{m(U_0 2^{64} + U_1) + h}{2^{128}} + \frac{\ell}{2^{192}} + t \bmod 1,$$

with $0 \leq t < 2^{-139}$. The value of v at line 9 is exactly $m(U_0 2^{64} + U_1) + h$ modulo 2^{128}, thus after this line we have $x/(2\pi) \bmod 1 = v/2^{128} + s$, with $0 \leq s < \ell/2^{192} + 2^{-139} < 2^{-127.999}$. The final lines 11 and 12 round k to nearest, ensuring $-2^{-14} \leq R < 2^{-14}$ instead of $0 \leq R < 2^{-13}$, leaving the sum $k/2^{13} + R$ invariant. ∎

The main algorithm for the accurate path in this branch is Algorithm AccurateLarge, which uses auxiliary algorithms mhUU, evalPS and evalPC described below. Some remarks about Algorithm AccurateLarge:

- this algorithm mainly deals with 128-bit unsigned integers representing values in $[0, 1)$. In the proof we use

Algorithm 9 (AccurateLarge)

Input: x a binary64 number, $|x| \geq 2^{-16}$
Output: the correct rounding of $\sin x$

1: $k, r, \sigma_r \leftarrow \text{ReduceAccLarge}(x)$
2: $\sigma \leftarrow \text{sign}(x)$ ▷ $\sigma = \pm 1$
3: if $k \geq 2^{12}$ then $\sigma \leftarrow -\sigma$
4: write $k \bmod 2^{12} = i_1 2^6 + i_2$ with $0 \leq i_1, i_2 < 2^6$
5: $s_{t_1}, c_{t_1} \leftarrow s_1[i_1], c_1[i_1]$
6: $s_{t_2}, c_{t_2} \leftarrow s_2[i_2], c_2[i_2]$
7: $\lambda \leftarrow \text{mhUU}(s_{t_1}, c_{t_2})$
8: $\mu \leftarrow \text{mhUU}(c_{t_1}, s_{t_2})$
9: if $i_1 < 32$ then $s_z \leftarrow \lambda + \mu$ else $s_z \leftarrow \lambda - \mu$
10: $\lambda' \leftarrow \text{mhUU}(c_{t_1}, c_{t_2})$
11: $\mu' \leftarrow \text{mhUU}(s_{t_1}, s_{t_2})$
12: if $i_1 < 32$ then $c_z \leftarrow \lambda' - \mu'$ else $c_z \leftarrow \lambda' + \mu'$
13: $u_{2h} \leftarrow \lfloor u_2/2^{64} \rfloor$
14: $s_r \leftarrow \text{evalPS}(u, u_2, u_{2h}, u_4)$
15: $c_r \leftarrow \text{evalPC}(u_2, u_{2h}, u_4)$
16: $w_1 \leftarrow \text{mhUU}(s_z, c_r)$
17: $w_2 \leftarrow \text{mhUU}(c_z, s_r)$
18: if ($i_1 < 32$ and $\sigma_r = 0$) or ($i_1 \geq 32$ and $\sigma_r = 1$) then $s \leftarrow w_1 + w_2$
19: elif $w_1 \leq w_2$ then $s \leftarrow w_2 - w_1$, $\sigma = -\sigma$
20: else $s \leftarrow w_1 - w_2$
21: return $\text{u128tod}(s, \sigma)$

small letters to denote these 128-bit unsigned integers, for example u at line 6, and capital letters to denote the corresponding rational value, for example $U = u/2^{128}$;

- the constant π_2 approximates $2^{126}\pi$, thus $U = u/2^{128}$ at line 6 approximates $2\pi R$, U_2 approximates $(2\pi R)^2$, and U_4 approximates $(2\pi R)^4$;
- the precomputed values $S_1[i_1] := s_1[i_1]/2^{128}$ and $C_1[i_1] := c_1[i_1]/2^{128}$ approximate $\sin(t_1)$ and $|\cos(t_1)|$ where $t_1 = \pi i_1/2^6$. We have $\sin(t_1) \geq 0$, $\cos(t_1) \geq 0$ for $i_1 \leq 32$, and $\cos(t_1) < 0$ for $32 < i < 64$;
- the precomputed values $S_2[i_2] := s_2[i_2]/2^{128}$ and $C_2[i_2] := c_2[i_2]/2^{128}$ approximate $\sin(t_2)$ and $|\cos(t_2)|$ where $t_2 = \pi i_2/2^{12}$. We have $\sin(t_2), \cos(t_2) \geq 0$;
- since all computations are done with integers (except the final rounding done by routine u128tod at line 21), the error analysis does not depend on the rounding mode;
- line 9 uses the relation $\sin(t_1 + t_2) = \sin(t_1)\cos(t_2) + \cos(t_1)\sin(t_2)$, adjusting the sign when $i_1 \geq 32$ since then $\cos(t_1) \leq 0$ but the value stored in c_{t_1} is nonnegative. Then s_z approximates $\sin(z)$ with $z = t_1 + t_2$;
- line 12 uses the relation $\cos(t_1 + t_2) = \cos(t_1)\cos(t_2) - \sin(t_1)\sin(t_2)$, adjusting the sign when $i_1 \geq 32$ since then $\cos(t_1) \leq 0$ but the value stored in c_{t_1} is nonnegative, and c_z approximates $|\cos(z)|$.

Algorithm mhUU approximates the high part of the product of two 128-bit values, by splitting each one into two 64-bit parts, ignoring the product of the lower parts, and truncating the middle products. If A, B are the corresponding rational values $a/2^{128}$ and $b/2^{128}$, then $C = c/2^{128}$ approximates $A \cdot B$.

Algorithm 10 (mhUU)

Input: a, b two 128-bit integers
Output: c approximating $ab/2^{128}$

1: write $a = a_h 2^{64} + a_\ell$ with $0 \leq a_h, a_\ell < 2^{64}$
2: write $b = b_h 2^{64} + b_\ell$ with $0 \leq b_h, b_\ell < 2^{64}$
3: $h \leftarrow a_h b_h$
4: $m_1 \leftarrow a_h b_\ell, \quad m_2 \leftarrow a_\ell b_h$
5: $c \leftarrow h + \lfloor m_1/2^{64} \rfloor + \lfloor m_2/2^{64} \rfloor$

Algorithm evalPS approximates $\sin(2\pi R)$ for $0 \leq R \leq 2^{-14}$.

Algorithm 11 (evalPS)

Input: a 128-bit integer r, $0 \leq r \leq 2^{114}$
Output: s_r approximating $2^{128}\sin(2\pi r/2^{128})$

1: $c_1 = \texttt{0xfffffffffffffffffffffffffffffc396}$
2: $c_3 = \texttt{0x2aaaaaaaaaaaaaaaaaaaa9646e8f872e}$
3: $c_5 = \texttt{0x22222222222220467b898367fb1249}$
4: $c_7 = \texttt{0xd00d00bfff5ee}$
5: $\pi_2 = \texttt{0xc90fdaa22168c234c4c6628b80dc1cd1}$
6: $u \leftarrow \text{mhUU}(\pi_2, 8r)$
7: $u_2 \leftarrow \text{mhUU}(u, u)$
8: $u_4 \leftarrow \text{mhUU}(u_2, u_2)$
9: $u_{2h} \leftarrow \lfloor u_2/2^{64} \rfloor$
10: $s_5 \leftarrow c_5 - c_7 u_{2h}$
11: $s_1 \leftarrow c_1 - \text{mhUU}(c_3, u_2)$
12: $s_2 \leftarrow s_1 + \text{mhUU}(s_5, u_4)$
13: $s_r \leftarrow \text{mhUU}(s_2, u)$

Lemma 6: Given as input $0 \leq r \leq 2^{114}$, the value s_r returned by Algorithm evalPS satisfies:

$$|S_r - \sin(2\pi R)| < 2^{-125.314}.$$

Proof: Let $S(x) = C_1 x - C_3 x^3 + C_5 x^5 - C_7 x^7$, where $C_i = c_i/2^{128}$. Sollya proves that for $|R| \leq 2\pi \cdot 2^{-14} < 2^{-11.348}$:

$$|S(R) - \sin(2\pi R)| < 2^{-128.601}.$$

Since $r \leq 2^{114}$, $8r$ does not overflow at line 6, and is therefore exact. For $0 \leq r/2^{128} \leq 2^{-14}$, our Gappa script proves $2^{-7} \leq S_5 := s_5/2^{128} < 2^{-6}$, thus s_5 does not underflow. It also proves $0.999999 < S_1 \leq 1$, thus s_1 does not underflow either (since $c_1 < 1$ and $\text{mhUU}(c_3, u_2) \geq 0$, we cannot have $s_1 = 1$). It also proves that s_2 does not overflow. Finally it proves:

$$|S_r - S(2\pi R)| < 2^{-125.471}.$$

Adding the Sollya bound $2^{-128.601}$ concludes the proof. ■

Similarly, Algorithm evalPS approximates $\cos(2\pi R)$. In the real code, the computations of u, u_2, u_4 and u_{2h} are shared between evalPS and evalPC.

Lemma 7: Given as input $0 \leq r \leq 2^{114}$, the value c_r returned by Algorithm evalPC satisfies:

$$|C_r - \cos(2\pi R)| < 2^{-125.539}.$$

Algorithm 12 (evalPC)

Input: a 128-bit integer r, $0 \le r \le 2^{114}$
Output: c_r approximating $2^{128}\cos(u/2^{128})$

1: $v_0 = \texttt{0xffffffffffffffffffffffffffffffff}$
2: $v_2 = \texttt{0x7fffffffffffffffffffffffa998bf0}$
3: $v_4 = \texttt{0xaaaaaaaaaaaaaaaaaa2caecc94962c7}$
4: $v_6 = \texttt{0x5b05b05b05ac1a642013bc289028b6}$
5: $v_8 = \texttt{0x1a0193d9a550c}$
6: $\pi_2 = \texttt{0xc90fdaa22168c234c4c6628b80dc1cd1}$
7: $u \leftarrow \mathrm{mhUU}(\pi_2, 8r)$
8: $u_2 \leftarrow \mathrm{mhUU}(u, u)$
9: $u_4 \leftarrow \mathrm{mhUU}(u_2, u_2)$
10: $u_{2h} \leftarrow \lfloor u_2/2^{64} \rfloor$
11: $c_6 \leftarrow v_6 - v_8 u_{2h}$
12: $c_4 \leftarrow v_4 - \mathrm{mhUU}(c_6, u_2)$
13: $c_0 \leftarrow v_0 - \mathrm{mhUU}(v_2, u_2)$
14: $c_r \leftarrow c_0 + \mathrm{mhUU}(c_4, u_4)$

Proof: Let $C(x) = C_0 x - C_2 x^2 + C_4 x^4 - C_6 x^6 + C_8 x^8$, where $C_i = v_i/2^{128}$. Sollya proves that for $|R| \le 2\pi \cdot 2^{-14} < 2^{-11.348}$:

$$|C(R) - \cos(2\pi R)| < 2^{-128}.$$

For $0 \le r/2^{128} \le 2^{-14}$, our Gappa script proves $2^{-10} \le C_6 < 2^{-9}$, thus c_6 does not underflow. It also proves $2^{-5} \le C_4 < 2^{-4}$, thus c_4 does not underflow either; and it proves $0.999999 \le C_0 \le 1-2^{-128}$, thus c_0 does not underflow either. It also proves $0.999999 \le C_r \le 1 - 2^{-128}$, thus c_r does not overflow. Finally it proves:

$$|C_r - C(2\pi R)| < 2^{-125.829}.$$

Adding the Sollya bound 2^{-128} concludes. ∎

Theorem 4: Given a binary64 number x, $|x| \ge 2^{-16}$, Algorithm AccurateLarge returns the correct rounding of $\sin x$, whatever the rounding mode.

Proof: First we know from Lemma 5 that, denoting $z = k/2^{13}$:

$$\frac{x}{2\pi} \mod 1 = z + R + s,$$

with $|s| < 2^{-127.999}$. Since $\sin x$ is contracting, it yields:

$$|\sin(2\pi(z+R)) - \sin(x)| < 2^{-127.999}. \tag{4}$$

The rational $\pi_2/2^{128}$ approximates $2\pi/8$ downward:

$$\Pi_2 := \frac{\pi_2}{2^{128}} = \frac{2\pi}{8} - \epsilon_1,$$

with $0 \le \epsilon_1 < 2^{-130.642}$.

An exhaustive search over all 2^{12} possible values of i_1, i_2 bounds the maximal error for s_z at line 9, with $z = t_1 + t_2$:

$$|S_z - \sin(2\pi z)| < 2^{-125.691},$$

where the maximal error is obtained for $i_1 = 12$ and $i_2 = 53$. It also shows $0 \le s_z \le 2^{128} - 3$, where the minimal value is obtained for $i_1 = i_2 = 0$, and the maximal one for $i_1 = 32$, $i_2 = 0$. A similar exhaustive search bounds the maximal error for c_z at line 12:

$$|C_z - |\cos(2\pi z)|| < 2^{-125.691},$$

where the maximal error is obtained for $i_1 = 44$ and $i_2 = 53$. It also shows $0 \le c_z \le 2^{128} - 3$, where the minimal value is obtained for $i_1 = 32$, $i_2 = 0$, and the maximal one for $i_1 = i_2 = 0$.

After the computation of w_1 and w_2 at lines 16-17, a Gappa script proves:

$$\begin{aligned} |W_1 - \sin(2\pi z)\cos(2\pi R)| &< 2^{-124.530}, \\ |W_2 - |\cos(2\pi z)\sin(2\pi R)|| &< 2^{-124.867}. \end{aligned}$$

If $i_1 < 32$ and $\sigma_r = 0$, then $\cos(2\pi z), \sin(2\pi R) \ge 0$, thus W_2 approximates $\cos(2\pi z)\sin(2\pi R)$ and $w_1 + w_2$ approximates $2^{128}\sin(2\pi(z+R))$; if $i_1 \ge 32$ and $\sigma_r = 1$, then $\cos(2\pi z), \sin(2\pi R) \le 0$, thus W_2 approximates $\cos(2\pi z)\sin(2\pi R)$ too, and $w_1 + w_2$ approximates $2^{128}\sin(2\pi(z+R))$. Otherwise, $\cos(2\pi z)\sin(2\pi R) \le 0$, thus we have to subtract w_2 from w_1, and adjust the sign, which is what the algorithm does. In all cases, the total error is bounded as follows:

$$|(-1)^\sigma S - \sin(2\pi(z+R))| < 2^{-124.530} + 2^{-124.867} < 2^{-123.688}.$$

Combined with the bound from Eq. (4), we obtain:

$$|(-1)^\sigma S - \sin(x)| < 2^{-123.688} + 2^{-127.999} < 2^{-123.617}. \tag{5}$$

Now assume $|\sin x| \ge 2^{-27}$ and $y = \sin x$ has less than 43 identical bits after the round bit. Then if z is a rounding boundary, $|y - z| \ge 2^{-44}\mathrm{ulp}(y) \ge 2^{-123}$, since $|y| \ge 2^{-27}$ implies $\mathrm{ulp}(y) \ge 2^{-79}$. Thus Eq. (5) shows that $(-1)^\sigma S$ is closer from $\sin x$ than from any rounding boundary, thus rounding $(-1)^\sigma S$ yields the correct rounding of $\sin x$.

All inputs such that $\sin x$ has 43 or more identical bits after the round bit were computed in [2], and are included in the `sin.wc` file. The CORE-MATH test suite checked they are correctly rounded, for all rounding modes, and with or without FMA contraction.

It remains to prove correct rounding when $|\sin x| < 2^{-27}$. In this case, necessarily $k = 0$ in Algorithm ReduceAccLarge, since otherwise from Eq. (3), $|z + R| \ge 2^{-14} - 2^{-127.999}$, which from Eq. (4) is incompatible with $|\sin x| < 2^{-27}$. If $2^{-16} \le |x| < 2\pi - 2^{-27}$, this cannot happen. Thus necessarily we have $|x| > 6$. Using Lefèvre's algorithm, in each binade $[2^{e-1}, 2^e)$, for $3 \le e \le 1024$, we generated all binary64 inputs such that $|R| < 2^{-27} + 2^{-127.999} < 2^{-26.999}$ in Algorithm ReduceAccLarge, and kept those for which $\sin x$ is at distance less than 2^{-106} from a rounding boundary (the bound 2^{-123} would have been enough, but we wanted to find a non-empty set). We added these inputs to `sin.wc`, and checked they are correctly rounded for all rounding modes, with and without FMA. For example, in the binade $[2^{1023}, 2^{1024})$, we found 15 such inputs, from `0x1.28fe30a8ce991p+1023` to `0x1.f7c1520fe316p+1023`. Among these inputs, we found $\sin x$ is closest to a rounding boundary (in absolute

value) for $x = \texttt{0x1.0951808d5c44dp+765}$, with $\sin x$ at distance about $2^{-119.457}$ from a rounding boundary, which leaves 4 guard bits with respect to the bound $2^{-123.617}$. ■

V. Efficiency

Figure 1 compares the efficiency of several implementations of the binary64 sine function for $1 \le x \le 2^{100}$. We see all implementations have two regimes: one for small inputs with an ad-hoc argument reduction, and one for large inputs, with Payne-Hanek's argument reduction. The threshold between the two regimes differs between implementations: around 2^{15} for LLVM libc, around 2^{19} for the Intel library, around 2^{21} for AMD libm (with a spike around 2^{14}), around 2^{25} for GNU libc, and around 2^{31} for CORE-MATH. Note that only CORE-MATH and LLVM libc are correctly rounded; they also have the fastest Payne-Hanek reductions. Extending the graph up to 2^{1024} gives a similar behaviour.

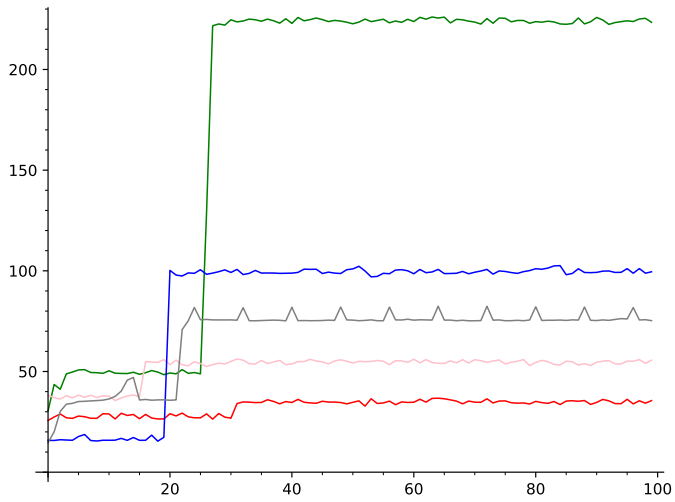

Fig. 1. Reciprocal throughput of several implementations of the binary64 function around 2^e for $0 \le e \le 100$, on an Intel Xeon Silver 4214. The blue curve is the Intel math library 2026.1.0, the red curve is CORE-MATH compiled with gcc 16.2.0, the pink curve is LLVM libc (cd6dab0) compiled with clang 21.1.8, the grey curve is AMD libm 5.3,

Acknowledgements. The author thanks Eric Hennenhoefer who gave some feedback (with the help of AI) on an earlier version of this paper.

References

[1] 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).

[2] Lefèvre, V., Ly, T., and Zimmermann, P. Computing hard-to-round cases of sin, cos, tan in double precision. In *ARITH 2026 - 33rd IEEE International Symposium on Computer Arithmetic* (Fulda, Germany, June 2026).

[3] Payne, M., and Hanek, R. Radian reduction for trigonometric functions. *SIGNUM Newsletter 18* (1983), 19–24.