The CORE-MATH acos is correctly rounded

Paul Zimmermann
April 20, 2026

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

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

We consider the CORE-MATH `acos.c` file from commit `7457819`. The function $\mathrm{acos}(x)$ is only defined for $|x| \le 1$ (for $|x| > 1$ it returns NaN), and it decreases from π to 0 on $[-1, 1]$ (see Figure 1). We consider all four rounding modes

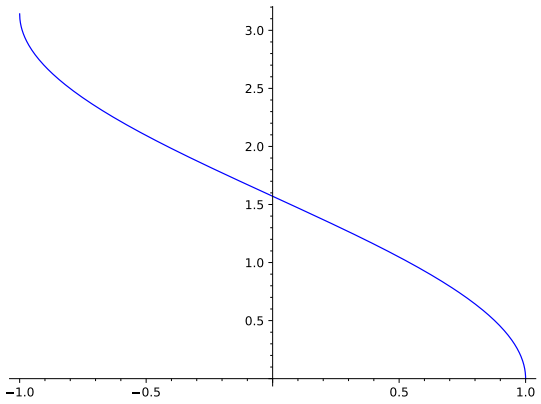

Fig. 1. Graph of $\mathrm{acos}(x)$ on $[-1, 1]$.

from the C23 standard: to nearest-even, toward zero, toward $+\infty$, and toward $-\infty$.

I. Branch $|x| < 2^{-15}$

In this case we use Algorithm FastPathTiny, which approximates $\mathrm{acos}(x)$ by $p - (x + x^3/6)$, where $p = p_h + p_\ell$ is a double-double approximation of $\pi/2$. The fasttwosum routine

Algorithm 1 (FastPathTiny)

Input: x a binary64 number, $|x| \le 2^{-15}$
Output: the correct rounding $\circ(\mathrm{acos}(x))$ or FAIL

1: $p_h = \texttt{0x1.921fb54442d18p+0}$
2: $p_\ell = \texttt{0x1.1a62633145c07p-54}$
3: $c = \texttt{-0x1.5555555555555p-3}$ ▷ approximates $-1/6$
4: $t = \circ(x^2), \quad u = \circ(c \cdot x), \quad v = \circ(t \cdot u)$
5: $h, w = \mathrm{fasttwosum}(p_h, -x)$
6: $\ell = \circ(v + \circ(w + p_\ell)), \quad \epsilon = \texttt{0x1.34p-79}$
7: $\ell_b = h + \circ(\ell - \epsilon), \quad u_b = h + \circ(\ell + \epsilon)$
8: if $\circ(\ell_b) = \circ(u_b)$ return $\circ(\ell_b)$ else return FAIL

is defined in Algorithm 2.

Lemma 1: For x a binary64 number, $|x| < 2^{-15}$, and any rounding mode, if Algorithm FastPathTiny does not return FAIL, then it yields the correct rounding of $\mathrm{acos}(x)$.

Algorithm 2 (fasttwosum)

Input: a, b binary64 numbers
Output: h, ℓ such that $h + \ell$ approximates $a + b$

1: $h \leftarrow \circ(a + b)$
2: $t \leftarrow \circ(h - a)$
3: $\ell \leftarrow \circ(b - t)$

Proof: Sollya [2] proves that the absolute error is bounded by $2^{-78.736}$:

```
prec=64;
ph=0x1.921fb54442d18p+0;
pl=0x1.1a62633145c07p-54;
c=0x1.5555555555555p-3;
q=x+c*x^3;
y=ph+pl-q;
d=[-2^-15],2^-15];
dirtyinfnorm(y-acos(x),d);
1.9852337356788177956e-24
```

We thus have:

$$|[p - (x + cx^3)] - \mathrm{acos}(x)| \le 2^{-78.736}. \tag{1}$$

We then use the following Gappa script to bound the rounding error in Algorithm FastPathTiny, with `RND` substituted by all rounding modes:

```
@rnd = float<ieee_64,RND>;
ph = 0x1.921fb54442d18p+0;
pl = 0x1.1a62633145c07p-54;
c = -0x1.5555555555555p-3;
t = x*x;
tr rnd= x * x;
u = c*x;
ur rnd=c*x;
v = t*u;
vr rnd= tr*ur;
y = -x;
# h, w = fasttwosum(ph, y)
h rnd= ph + y;
w = rnd(-(h-(ph+y)));
l1 rnd= w+pl;
l rnd= vr+l1;

{ x in [-1b-15,1b-15] -> x // ph in [-1,1]
  /\ |(h+l)-((ph+pl)+(y+v))| in ? }

(h+w) - (ph+y) -> w - (-(h-(ph+y)));
```

```
(h+l)-((ph+pl)+(y+v)) -> ((h+w) - (ph+y))
  + (l1 - (w+pl)) + (l - (vr+l1)) + (vr-v);
```

This script shows that the absolute error is bounded by $2^{-98.0891}$:

$$|h + \ell - [p - (x + cx^3)]| \leq 2^{-98.0891}. \quad (2)$$

Combining Eq. (1) with Eq. (2), we get:

$$|h + \ell - \mathrm{acos}(x)| < 2^{-78.735}.$$

With the goal `|l| in ?`, this script also proves $|\ell| \leq 2^{-47.5012}$. Thus the rounding error in $\circ(\ell \pm \epsilon)$ for ℓ_b and u_b at line 7 of Algorithm FastPathTiny is bounded by $\mathrm{ulp}(2^{-47.5}) = 2^{-100}$. Thus since $\epsilon > 2^{-78.735} + 2^{-100}$, we have $\ell_b < \mathrm{acos}(x) < u_b$. By monotonicity of rounding, if $\circ(\ell_b) = \circ(u_b)$, this concludes the proof. ∎

REMARK: in the actual code, to avoid a spurious underflow in the computation of v, we set $v = 0$ when $|x| \leq x_0 :=$ `0x1.cb3b3869747f4p-55`, since then $\mathrm{acos}(x)$ rounds to the same value as $\mathrm{acos}(0)$. The proof remains correct since this adds an absolute error less than $|cx^3| < 2^{-165}$. Moreover, for $|x| \leq x_0$, we always have $\ell_b = u_b$, thus the accurate path is never called (Lemma 2).

Lemma 2: For $|x| \leq x_0$, and any C23 rounding mode, Algorithm FastPathTiny never returns FAIL.

Proof: Since $x_0 < \mathrm{ulp}(p_h)$, at line 5, h is either p_h, $p_h + \mathrm{ulp}(p_h)$ or $p_h - \mathrm{ulp}(p_h)$, depending on the rounding mode and the sign of x. We know from [Theorem 9-11] [3] that $h + w$ either equals $p_h - x$, or equals its rounding in double precision $\circ_{106}(p_h - x)$. Thus $h + w$ is decreasing with x, and since for a given rounding mode and sign of x, h is fixed, then w is decreasing with x. For each rounding mode, it thus suffices to check the property is satisfied for $x = \pm x_0$. And indeed, for $x = \pm x_0$, we have $\circ(\ell_b) = \circ(u_b)$ for all rounding modes. ∎

A. Accurate path

This branch uses the same routine for the accurate path as in §II, which is proven correct in §IV.

II. Branch $2^{-15} \leq |x| < 2^{-4}$

For $|x| < 0.5$, the code computes the integer $j = \lfloor 2^7 \circ(x^2) \rceil$, and a polynomial approximation of $\mathrm{asin}(x)$ indexed by j is used:

$$q(x) = x \cdot (c_0 + c_1 + t(c_2 + c_3 t + \cdots + c_7 t^5)),$$

where $t = \mathrm{fma}(x, x, -2^{-7} j)$, and then $\mathrm{acos}(x)$ is approximated by:

$$p_h + p_\ell - q(x), \quad (3)$$

where $p_h + p_\ell$ is a double-double approximation of $\pi/2$. The integer j goes from 0 for $2^{-15} \leq |x| \leq 2^{-4}$ to 32 for $|x|$ near 0.5. We only study the case $|x| < 2^{-4}$ here. For $2^{-4} \leq |x| < 0.5$, the correctness will be proven by exhaustive tests in Section III.

In the case $|x| < 2^{-4}$, we have $\circ(x^2) < 2^{-8}$, thus $j = 0$ (it is easy to check that for $x = \mathrm{nextdown}(2^{-4})$, $\mathrm{RU}(x^2) < 2^{-8}$). In this $j = 0$ case, Algorithm FastPathSmall is used, with $c_0, ..., c_7$ be the coefficients for index $j = 0$, and t simplifies to $\circ(x^2)$.

Algorithm 3 (FastPathSmall)

Input: x a binary64 number, $2^{-15} \leq |x| < 2^{-4}$
Output: the correct rounding $\circ(\mathrm{acos}(x))$ or FAIL

1: $p_h =$ `0x1.921fb54442d18p+0`
2: $p_\ell =$ `0x1.1a62633145c07p-54`
3: $t = \circ(x^2), \quad z = -x, \quad z_\ell = 0$
4: $t_2 = \circ(t \cdot t)$
5: $g_2 = \circ(c_2 + \circ(tc_3)), \quad h_4 = \circ(c_4 + \circ(tc_5))$
6: $h_6 = \circ(c_6 + \circ(tc_7)), \quad g_4 = \circ(h_4 + \circ(t_2 h_6))$
7: $e_2 = \circ(g_2 + \circ(t_2 g_4)), \quad d = \circ(te_2)$
8: $f_h = c_0, \quad f_\ell = \circ(c_1 + d)$
9: $f_h, f_\ell = \mathrm{muldd}(z, z_\ell, f_h, f_\ell)$
10: $h, \ell = \mathrm{fastsum}(p_h, p_\ell, f_h, f_\ell)$
11: $\epsilon = \circ(\circ(zt),$ `0x1.81p-52`$)$
12: $\ell_b = h + \circ(\ell - \epsilon), \quad u_b = h + \circ(\ell + \epsilon)$
13: if $\circ(\ell_b) = \circ(u_b)$ return $\circ(\ell_b)$ else return FAIL

REMARK: in line 11 of FastPathSmall, ϵ is negative for $x > 0$. In this case, ℓ_b is an upper bound and u_b a lower bound, but the rounding test remains correct. In the following we assume ϵ represents $|\epsilon|$.

Algorithm fasttwosum is the same as in §I, and fastsum and muldd are given in Algorithms 4 and 5.

Algorithm 4 (fastsum)

Input: $a := a_h + a_\ell, b := b_h + b_\ell$ double-double numbers
Output: h, ℓ such that $h + \ell$ approximates $a + b$

1: $h, s \leftarrow \mathrm{fasttwosum}(a_h, b_h)$
2: $u \leftarrow \circ(a_\ell + b_\ell)$
3: $\ell \leftarrow \circ(u + s)$

Algorithm 5 (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: $\ell \leftarrow \circ(\circ(\ell_2 + \ell_3) + \ell_1)$

Lemma 3: For x a binary64 number, $2^{-15} \leq |x| < 2^{-4}$, and any rounding mode, if Algorithm FastPathSmall does not return FAIL, then it yields the correct rounding of $\mathrm{acos}(x)$.

Proof: In $[2^{-15}, 2^{-4}]$, we can prove with Sollya, for $y = p_h + p_\ell - q(x)$ (see Figure 2):

$$10219 \cdot 2^{-70} \leq (y - \mathrm{acos}(x))/x^3 \leq 10985 \cdot 2^{-70}. \quad (4)$$

We then use the following Gappa script, with Gappa option `-Echange-threshold=0`, with `RND` substituted by all rounding modes, `EPS` substituted by `0x1.81p-52`, `XMIN`

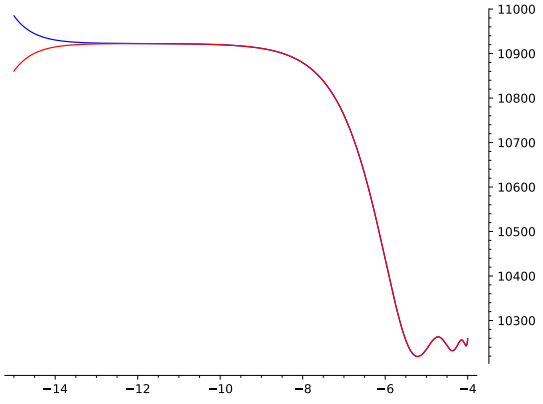

Fig. 2. Graph of $((p_h + p_\ell - q(x)) - \mathrm{acos}(x))/x^3$ scaled by 2^{70}, for $2^{-15} \le |x| \le 2^{-4}$, where the x-axis gives $\log_2 |x|$. The blue curve is for $x > 0$, the red curve for $x < 0$.

and XMAX substituted by 2^{-15} and 2^{-4} for positive x, by -2^{-4} and -2^{-15} for negative x, and BND substituted by the upper bound $10985 \cdot 2^{-70}$:

```
@rnd = float<ieee_64,RND>;
ph = 0x1.921fb54442d18p+0;
pl = 0x1.1a62633145c07p-54;
p = ph+pl; # approximates pi/2
x = rnd(x_);
x2 = x*x;
t rnd= x*x;
z = -x;
c0 = 1;
c1 = 0;
c2 = 0x1.5555555555555p-3;
c3 = 0x1.33333333333e4p-4;
c4 = 0x1.6db6db6d31f82p-5;
c5 = 0x1.f1c71f6889397p-6;
c6 = 0x1.6e874b7045b46p-6;
c7 = 0x1.1f753132271e2p-6;
x4 = x2*x2;
t2 rnd= t*t;
g2 = c2+x2*c3;
g2r rnd= c2+t*c3;
h4 = c4+x2*c5;
h4r rnd= c4+t*c5;
h6 = c6+x2*c7;
h6r rnd= c6+t*c7;
g4 = h4+x4*h6;
g4r rnd= h4r+t2*h6r;
e2 = g2+x4*g4;
e2r rnd= g2r+t2*g4r;
d = x2*e2;
dr rnd= t*e2r;
fl = c1+d;
flr rnd= c1+dr;
# fh1, fl1 = muldd(z, 0, c0, flr)
q = z*(c0+fl);
fh1 rnd= z*c0;
l1 = -(fh1-(z*c0));
l2 rnd= z*flr;
# l3=0
fl1 rnd= l2+l1;
# fh1+fl1 approximates q
# h, l = fastsum(ph, pl, fh1, fl1)
# h, s = fasttwosum(ph, fh1)
h rnd= ph+fh1;
s = rnd(-(h-(ph+fh1)));
u rnd= pl+fl1;
l rnd= u+s;
eps rnd= |z*t|*EPS;
lb = h + rnd(l - eps);
# h+l approximates ph+pl+q
y = ph+pl+q;
x3 = x2*x;

{ x in [XMIN,XMAX] -> fh1 // ph in [-1,1]
  /\ (lb-y)+x3*BND <= 0 }

fh1+l1 -> z*c0;
(fh1+fl1)-q -> ((fh1+l1) - z*c0)
  + (fl1 - (l2+l1)) + (l2 - z*flr) + z*(flr-fl);
(h+s) - (ph+fh1) -> s - (-(h-(ph+fh1)));
(h+l)-(p+(fh1+fl1)) -> ((h+s) - (ph+fh1))
  + (u - (pl+fl1)) + (l - (u+s));
```

This script proves:

$$\frac{\ell_b - y}{x^3} + 10985 \cdot 2^{-70} \le 0.$$

Combined with Eq. (4), this proves $\ell_b \le \mathrm{acos}(x)$.

A similar Gappa script which differs only by the following lines proves $\mathrm{acos}(x) \le u_b$:

```
ub = h + rnd(l + eps);
{ x in [XMIN,XMAX] -> fh1 // ph in [-1,1]
  /\ (ub-y)+x3*BND >= 0 }
```

We thus have $\ell_b \le \mathrm{acos}(x) \le u_b$, whence since rounding is monotonic $\circ(\ell_b) \circ (\mathrm{acos}(x)) \le \circ(u_b)$, if $\circ(\ell_b) = \circ(u_b)$, they equal the correct rounding of $\mathrm{acos}(x)$. ■

If we reduce the error bound 0x1.81p-52 to 0x1.d3p-53, the fast path fails for x = 0x1.7cb54339263fbp-12 without FMA. On the other hand, the largest error bound (as a 9-bit number) we found that defeats the rounding test is 0x1.00p-52 with x = 0x1.d12b3716d66e4p-7. The smallest error bound we could prove with Gappa for $2^{-15} \le |x| < 2^{-4}$ is 0x1.5fp-52, but it took about 4 hours on our computer, and this branch of the code shares the same error bound than $2^{-4} \le |x| < 0.5$, for which 0x1.81p-52 was proven to be enough by exhaustive tests (§III). It might be that an error bound smaller than 0x1.81p-52 works too, while defeating the condition $\ell_b \le \mathrm{acos}(x) \le u_b$ of the rounding test. Indeed, it can be that cases where this condition is not satisfied are

far from rounding boundaries, and thus $\circ(\ell_b)$ is still the correct rounding. However, finding the smallest "working" error bound is only possible with exhaustive search, which would be quite expensive.

A. Accurate path

The accurate path is analyzed in §IV.

III. Case $2^{-4} \le |x| \le 1$

For $2^{-4} \le |x| \le 1$, we performed an exhaustive search for all rounding modes, with and without FMA. This exhaustive search revealed that a previous error bound was too small.

IV. Accurate path

In this section we prove the correctness of the accurate path (routine `as_acos_refine` in the code) for $|x| < 0.5$. In theory, we only have to prove it for $|x| < 2^{-4}$ since for $2^{-4} \le |x| < 1$ we performed an exhaustive search (§III), which checks the accurate path is correct when it is called. Moreover we know from Lemma 2 that the accurate path is never called for $|x| \le x_0 := \texttt{0x1.cb3b3869747f4p-55}$, thus we assume $x_0 < |x| < 0.5$ in this section.

The accurate path takes as input the binary64 number x and the approximation ϕ of $\mathrm{acos}(x)$ computed by FastPathSmall ($\phi = \ell_b$). From ϕ, an integer $j \in [0, 32]$ is deduced by $j \approx \lfloor 64 \cdot |\phi - \pi/2|/\pi \rceil$. For our domain of interest $|x| < 0.5$, we have $0 \le j \le 11$. Since $\phi \le \pi/2$ for $x > 0$ and $\phi \ge \pi/2$ for $x < 0$, we have:

$$\mathrm{acos}(x) = \frac{\pi}{2} \mp j\frac{\pi}{64} - \delta, \tag{5}$$

where $\mp$ is a minus sign for $x > 0$, a plus sign for $x < 0$, and $|\delta|$ is bounded by $\approx \pi/128$. Indeed, due to approximations and rounding errors, $|\delta|$ might slightly exceed $\pi/128$. For a given j, sign of x, and rounding mode, $|\delta|$ is maximal at the boundaries of the range of x with given sign yielding this value of j. We found that at all these boundaries, $|\delta| < 0.0246$. Eq. (5) translates to:

$$\delta = \frac{\pi}{2} \mp j\frac{\pi}{64} - \mathrm{acos}(x), \tag{6}$$

and to:

$$\sin(\delta) = \cos(j\frac{\pi}{64})x \mp \sin(j\frac{\pi}{64})\sqrt{1-x^2}. \tag{7}$$

Double-double approximations of $\cos(j\frac{\pi}{64})$ and $\sin(j\frac{\pi}{64})$ are tabulated, which yield double-double approximations of $\sqrt{1-x^2}$ and $\sin(\delta)$. Using a polynomial approximation of $\mathrm{asin}(t)$, we deduce a double-double approximation of δ. Then it suffices to plug this approximation into Eq. (5). Five hard-to-round cases are not rounded correctly by this algorithm, and need to be dealt with separately.

For simplicity, we omit roundings in lines 20 and 26 of Algorithm AccuratePath. At lines 15-16, we split $S_h d_{ch}$ into $S_c + d_{Sc}$ (up to rounding), where S_c is a multiple of $\mathrm{ulp}(M) = 2^{-56}$, and $|d_{Sc}| < 2^{-56}$. The same holds for $C_h d_{sh}$ which is split into $C_s + d_{Cs}$ at lines 17-18. This ensures that the subtraction $C_s - S_c$ at line 19 is exact.

Algorithm 6 (AccuratePath)

Input: x, ϕ binary64 numbers, with $\phi \approx \mathrm{acos}(x)$
Output: $p_h + p_\ell$ approximating $\mathrm{acos}(x)$

1: $s_2 = \circ(x^2), \quad d_2 = \mathrm{fma}(x, x, -s_2)$
2: $c_{2h}, c_{2\ell} = \mathrm{fasttwosum}(1, -s_2)$
3: $c_{2\ell} = \circ(c_{2\ell} - d_2)$
4: $c_{2h}, c_{2\ell} = \mathrm{fasttwosum}(c_{2h}, c_{2\ell})$
5: $c_h = \circ(\sqrt{c_{2h}})$
6: $c_\ell = \circ(\circ(c_{2\ell} - \mathrm{fma}(c_h, c_h, -c_{2h})) \cdot \circ(0.5/c_h))$
7: $P = \texttt{0x1.921fb54442d18p+0}$ ▷ $P \approx \pi/2$
8: $I = \texttt{0x1.45f306dc9c883p+4}$ ▷ $I \approx 64/\pi$
9: $j = \lfloor \circ(|\circ(\phi - P)| \cdot I) \rceil$
10: $C_h + C_\ell \approx \cos(j\pi/64)$ ▷ tabulated
11: $S_h + S_\ell \approx \sin(j\pi/64)$ ▷ tabulated
12: $d_{sh} = \circ(|x| - S_h), \quad d_{s\ell} = -S_\ell$
13: $d_{ch} = \circ(c_h - C_h), \quad d_{c\ell} = \circ(c_\ell - C_\ell)$
14: $M = \texttt{0x1.8p-4}$
15: $S_c = \circ(\mathrm{fma}(S_h, d_{ch}, M) - M)$
16: $d_{Sc} = \mathrm{fma}(S_h, d_{ch}, -S_c)$
17: $C_s = \circ(\mathrm{fma}(C_h, d_{sh}, M) - M)$
18: $d_{Cs} = \mathrm{fma}(C_h, d_{sh}, -C_s)$
19: $v = \circ(C_s - S_c)$
20: $d_v = (C_h d_{s\ell} + C_\ell d_{sh}) - (S_h d_{c\ell} + S_\ell d_{ch}) - (d_{Sc} - d_{Cs})$
21: $v, d_v = \mathrm{fasttwosum}(v, d_v)$
22: $j' = 32 - j$ if $x > 0$, $j' = 32 + j$ if $x < 0$
23: $v_2, d_{v2} = \mathrm{muldd}(v, d_v, v, d_v)$
24: $v = -\mathrm{sign}(x)v, \quad d_v = -\mathrm{sign}(x)d_v$
25: let $c_0 t + c_1 t^3 + \cdots + c_7 t^{15}$ approximating $\mathrm{asin}(x)$ where $c_i = c_{ih} + c_{i\ell}$ for $i \le 4$ ▷ tabulated
26: $r = v_2(c_5 + v_2(c_6 + v_2 c_7))$
27: $f_h, f_\ell = \mathrm{polydd}(v_2, d_{v2}, 5, r)$
28: $f_h, f_\ell = \mathrm{muldd}(v, d_v, f_h, f_\ell)$
29: $p_h = \circ(\texttt{0x1.921fb54442dp-5} \cdot j')$
30: $p_\ell = \circ(\texttt{0x1.8469898cc518p-53} \cdot j')$
31: $p_s = \circ(\texttt{-0x1.fc8f8cbb5bf6cp-102} \cdot j')$
32: $p_\ell, p_s = \mathrm{Sum}(f_h, f_\ell, p_\ell, p_s)$
33: $p_h, p_\ell = \mathrm{fasttwosum}(p_h, p_\ell)$
34: $p_\ell = \circ(p_\ell + p_s)$

Algorithm TwoSum is Algorithm 4.4 from [4]: From Theorem 4.1 from [1], Algorithm TwoSum satisfies, if all the rounding are faithful:

$$|s + t - (a + b)| \le 2^{-52}\mathrm{ulp}(s). \tag{8}$$

We split the analysis of Algorithm AccuratePath in three parts: in §IV-A we analyze the approximation of $\sqrt{1-x^2}$, in §IV-B we analyze the approximation of $\sin(\delta)$, from which in §IV-C we deduce an approximation of δ and thus of $\mathrm{acos}(x)$ using Eq. (5).

A. Approximation of $\sqrt{1-x^2}$

The double-double approximation $c_h + c_\ell$ of $\sqrt{1-x^2}$ is computed at lines 1-6.

Algorithm 7 (polydd)

Input: $x = x_h + x_\ell$ double-double, n integer, a double r
Output: $h+\ell$ approximating $c_0+c_1\cdot x^2+\cdots+(c_{n-1}+r)x^{2n-2}$

1: $h, t \leftarrow \text{fasttwosum}(c_{n-1,h}, r)$
2: $\ell \leftarrow \circ(t + c_{n-1,\ell})$
3: for i from $n-2$ downto 0
4: $\quad h, \ell \leftarrow \text{muldd}(x_h, x_\ell, h, \ell)$
5: $\quad h, e \leftarrow \text{fasttwosum}(c_{i,h}, h)$
6: $\quad \ell \leftarrow \circ(\circ(\ell + c_{i,\ell}) + e)$

Algorithm 8 (Sum)

Input: $x_h + x_\ell$ and $c_h + c_\ell$ two double-double pairs
Output: $s_h + s_\ell$ approximating $(x_h + x_\ell) + (c_h + c_\ell)$

1: $s_h, \ell \leftarrow \text{TwoSum}(x_h, c_h)$
2: $s_\ell \leftarrow \circ(\circ(x_\ell + c_\ell) + \ell)$

Lemma 4: For $|t| \le 2^{-7}$:

$$|\sqrt{1+t} - (1 + \frac{t}{2})| < 0.126t^2.$$

Proof: We can see graphically that the function $f(t) = (\sqrt{1+t} - (1 + t/2))/t^2$ is negative and increasing on $[-2^{-7}, 2^{-7}]$, thus reaches its maximum in absolute value at $t = -2^{-7}$, for which it is about 0.125491. ∎

Lemma 5: If $|\ell/h| \le 2^{-7}$, then:

$$\left|\sqrt{h} + \frac{\ell}{2\sqrt{h}} - \sqrt{h+\ell}\right| < 0.126\frac{\ell^2}{h^2}\sqrt{h}.$$

Proof: Apply Lemma 4 with $t = \ell/h$. ∎

We apply Lemma 5 with $h = c_h^2$ and $\ell = c_{2h} + c_{2\ell} - h$, with $c_{2h}, c_{2\ell}$ the values computed at line 4. Our Gappa script proves $|\ell/h| \le 2^{-50.415}$, so the condition of the lemma is fulfilled, thus we deduce from Lemma 5 and the fact that $c_h \le 1$:

$$\left|c_h + \frac{\ell}{2c_h} - \sqrt{c_{2h} + c_{2\ell}}\right| < 2^{-103.818}. \quad (9)$$

At line 6, the lower term $c_{2\ell}$ is added to $-\text{fma}(c_h, c_h, -c_{2h})$ which takes into account the rounding error in $c_h = \circ(\sqrt{c_{2h}})$. The Gappa script proves:

$$\left|c_\ell - \frac{\ell}{2c_h}\right| < 2^{-102.677}. \quad (10)$$

Combining Eq. (9) and Eq. (10) we obtain for the values of $c_{2h}, c_{2\ell}$ at line 4, and those of c_h, c_ℓ at lines 5-6:

$$\left|c_h + c_\ell - \sqrt{c_{2h} + c_{2\ell}}\right| < 2^{-102.137}. \quad (11)$$

Now we have to bound $|c_h + c_\ell - \sqrt{1-x^2}|$:

$$\begin{aligned}\left|c_h + c_\ell - \sqrt{1-x^2}\right| &\le \left|c_h + c_\ell - \sqrt{c_{2h} + c_{2\ell}}\right| \\ &+ \left|\sqrt{c_{2h} + c_{2\ell}} - \sqrt{1-x^2}\right|. \quad (12)\end{aligned}$$

Gappa proves that after line 4, $|c_{2h} + c_{2\ell} - (1-x^2)| \le 2^{-104}$. Let $\alpha = \sqrt{c_{2h} + c_{2\ell}}$ and $\beta = \sqrt{1-x^2}$. We thus have $|\alpha^2 - \beta^2| \le 2^{-104}$, thus $|\alpha - \beta| \le 2^{-104}/(\alpha+\beta)$. Gappa proves $c_{2h} + c_{2\ell} \ge 0.75$, and since $|x| \le 0.5$, $1 - x^2 \ge 0.75$ too, thus $\alpha, \beta \ge \sqrt{0.75} \ge 0.866$, and $|\alpha - \beta| \le 2^{-104}/(2 \cdot 0.866) < 2^{-104.792}$. Plugging into Eq. (12) and using Eq. (11), we obtain:

$$\left|c_h + c_\ell - \sqrt{1-x^2}\right| < 2^{-101.924}. \quad (13)$$

Algorithm 9 (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: $t \leftarrow \circ(\delta_a + \delta_b)$

B. Approximation of $\sin(\delta)$

A double-double approximation $v + d_v$ of $\sin(\delta)$ is computed at lines 7-21. As seen above, for $|x| < 0.5$, the integer j at line 9 satisfies $0 \le j \le 11$. The analysis depends on the precomputed constants C_h, C_ℓ, S_h, S_ℓ at lines 10 and 11. We analyze separately the case $j = 0$ since Gappa does not like variables which are zero.

Case $j = 0$**.** This case may happen when $x_0 < |x| \le$ `0x1.92155f7a366b7p-6`. We then have $C_h = 1$, and $C_\ell = S_h = S_\ell = 0$. It follows $d_{sh} = |x|$, $d_{s\ell} = 0$, Gappa proves $0.999 < c_h < 1.001$ thus $d_{ch} = c_h - C_h$ is exact, $d_{c\ell} = c_\ell$, $S_c = d_{Sc} = 0$, $C_s = \circ(\circ(|x| + M) - M)$ is the upper part of $|x|$ (rounded to a multiple of 2^{-56}), d_{Cs} is the lower part of $|x|$ such that $|x| = C_s + d_{Cs}$, $v = C_s$, $d_v = d_{Cs}$, thus $v + d_v = |x|$. We thus have exactly $\sin|\delta| = v + d_v$ in Eq. (7).

Let $V = (C_h + C_\ell)|x| - (S_h + S_\ell)(c_h + c_\ell)$, and $\delta' = \text{sign}(x)\delta$. For $1 \le j \le 11$, Gappa finds the bound (obtained for $j = 11$):

$$|v + d_v - V| \le 2^{-103.064}.$$

The approximations $C_h + C_\ell$ of $\cos(j\pi/64)$ and $S_h + S_\ell$ of $\sin(j\pi/64)$ have an absolute error bounded by $2^{-108.574}$. It follows:

$$\begin{aligned}|V - \sin(\delta')| &\le |C_h + C_\ell - \cos(j\pi/64)| \cdot |x| \\ &+ |S_h + S_\ell| \cdot |c_h + c_\ell - \sqrt{1-x^2}| \\ &+ |S_h + S_\ell - \sin(j\pi/64)| \cdot \sqrt{1-x^2}.\end{aligned}$$

Since $|x| < 0.5$, the first term above is bounded by $2^{-109.574}$. Since $|S_h + S_\ell| \le 1$, and $|c_h + c_\ell - \sqrt{1-x^2}| < 2^{-101.924}$ by Eq. (13), their product is bounded by $2^{-101.924}$. The last term is bounded by $2^{-108.574}$, thus:

$$|V - \sin(\delta')| \le 2^{-109.574} + 2^{-101.924} + 2^{-108.574} < 2^{-101.902}.$$

Combined with the above bound on $|v + d_v - V|$, this yields:

$$|v + d_v - \sin(\delta')| < 2^{-103.064} + 2^{-101.902} < 2^{-101.369}. \quad (14)$$

C. Approximation of δ

Lines 22-28 compute a double-double approximation f_h+f_ℓ of δ, from the approximation $v+d_v$ of $\sin(\delta')$. First, since $|\delta| < 0.0246$ and with Eq.(14), we have $|v+d_v| < 0.0247$, thus after line 21, we have $|v| < 0.025$ and $|d_v| \le 2^{-58}$.

Let us denote $q(t) = c_0 t + \cdots + c_7 t^{15}$ the polynomial approximation of $\mathrm{asin}(t)$ used in Algorithm AccuratePath. After the muldd call at line 28, Gappa proves for $t = v + d_v$:

$$|f_h + f_\ell - q(t)| < 2^{-106.759}.$$

We now have:

$$|f_h+f_\ell-\delta| \le |f_h+f_\ell-q(t)|+|q(t)-\mathrm{asin}(t)|+|\mathrm{asin}(t)-\delta|.$$

The first term in the right-hand side is bounded by $2^{-106.759}$ from just above. Sollya bounds the second term over $[-0.025, 0.025]$ by $|q(t) - \mathrm{asin}(t)| < 2^{-108.871}$. Since $|t - \sin(\delta)| < 2^{-101.369}$, the third term is bounded by $2^{-101.369}$ multiplied by a bound for the derivative of $\mathrm{asin}(t)$ over $[-0.025, 0.025]$. This derivative is maximal at $t = \pm 0.025$, where it is less than 1.0004, this yields a bound of $2^{-101.369} \cdot 1.0004 < 2^{-101.368}$ for the third term. Combining, we get:

$$|f_h+f_\ell-\delta| \le 2^{-106.759}+2^{-108.871}+2^{-101.368} < 2^{-101.326}.$$

At lines 29-31, we multiply the integer j', which lies in $[0, 64]$ (remember $0 \le j \le 32$) by a triple-word approximation of $\pi/64$, with error less than 2^{-155}. Since $j \le 11$ we have $21 \le j' \le 43$. The constants involved in p_h and p_ℓ are such that the multiplication by j' is exact. Since $|j'| \le 64$, we have $|p_s| < 2^{-95}$, thus the rounding error on p_s is bounded by $\mathrm{ulp}(2^{-96}) = 2^{-148}$. The error from the approximation of $\pi/64$ multiplied by $j' \le 2^6$ is bounded by 2^{-149}, thus after line 31:

$$|p_h + p_\ell + p_s - j'\frac{\pi}{4}| < 2^{-148} + 2^{-149} < 2^{-147.415}.$$

Gappa proves that the rounding error of calling Algorithm Sum is bounded by $2^{-105.752}$, thus after line 32:

$$\begin{aligned}&|p_h + p_\ell + p_s - \mathrm{acos}(x)|\\ &< 2^{-101.326} + 2^{-147.415} + 2^{-105.752} < 2^{-101.260}.\end{aligned}$$

On line 33, Gappa proves that the precondition $|p_\ell| \le |p_h|$ of $\mathrm{fasttwosum}$ is satisfied, and the rounding error due to $\mathrm{fasttwosum}$ is bounded by 2^{-104}. Finally, on line 34, Gappa proves the rounding error is bounded by 2^{-103}. We thus have for the final values p_h and p_ℓ:

$$|p_h+p_\ell-\mathrm{acos}(x)| \le 2^{-101.260}+2^{-104}+2^{-103} < 2^{-100.724}. \quad (15)$$

Theorem 1: For any binary64 number x, $|x| < 0.5$, such that $\mathrm{acos}(x)$ has less than 46 identical bits after the round bit, if $p_h + p_\ell$ is the double-word approximation of $\mathrm{acos}(x)$ computed by Algorithm AccuratePath, then $\circ(p_h + p_\ell)$ is the correct rounding of $\mathrm{acos}(x)$.

Proof: Let $y = \mathrm{acos}(x)$. If y has less than 46 identical bits after the round bit, then y is at distance at least $2^{-47}\mathrm{ulp}(y)$ from a rounding boundary z: $|y - z| \ge 2^{-47}\mathrm{ulp}(y)$. Since $\mathrm{ulp}(y) > 2^{-53}|y|$, this yields $|y - z| \ge 2^{-100}|y| \ge 2^{-100}$, since for $|x| < 0.5$, $\mathrm{acos}(x) \ge 1$. Thus Eq. (15) shows that $|p_h+p_\ell-y| < |y-z|$, i.e., p_h+p_ℓ is closer from $\mathrm{acos}(x)$ than any rounding boundary. It follows $\circ(p_h + p_\ell) = \circ(\mathrm{acos}(x))$. ■

V. Hard-to-round cases

During this proof, we realized that hard-to-round cases for $x < 0$ were missing in the CORE-MATH test suite. Indeed, since $\mathrm{acos}(-x)$ differs from $\pm\mathrm{acos}(x)$, it does not suffice to compute hard-to-round cases for $x > 0$.

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] Chevillard, S., Joldeş, M., and Lauter, C. Sollya: An environment for the development of numerical codes. In *Mathematical Software - ICMS 2010* (Heidelberg, Germany, September 2010), K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, Eds., vol. 6327 of *Lecture Notes in Computer Science*, Springer, pp. 28–31.

[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] Muller, J.-M., Brunie, N., de Dinechin, F., Jeannerod, C.-P., Joldes, M., Lefèvre, V., Melquiond, G., Revol, N., and Torres, S. *Handbook of Floating-point Arithmetic (2nd edition)*. Birkhäuser Basel, July 2018.