SolMath — Error Bound Analysis

Error bounds for core fixed-point functions. Bounds are stated in output units at SCALE = 10^{12} (or SCALE_HP = 10^{15} where noted): an error of 1 means the computed integer result differs from the ideal real-valued result by at most 1 in the least-significant fixed-point unit.

All line references are to `crate/solmath-core/src/`.

Disclaimer. The methodology and assumptions in this document are sound to the author's knowledge, but the proofs themselves were generated with AI assistance and have not been independently verified. The discerning reader should reproduce any result before relying on it. The empirical benchmarks were run by the author and are the primary accuracy reference.

Certificate script status. The Lipschitz certificate scripts (`lipschitz_certificate.py`, `trig_lipschitz_certificate.py`) must be present in the repository and reproducible. If these scripts are missing, the polynomial certification chain for Propositions 6, 9, and 10 is non-reproducible and should be treated as unverified until the scripts are restored and re-run against the current source.

1 Definitions

Definition 1 (Fixed-point representation)**.** The library represents real numbers as integers scaled by SCALE = 10^{12}. An integer v at standard scale represents the real number v/SCALE. For example, the integer 1,500,000,000,000 represents 1.5. All arithmetic operates on these integer representatives, and all error bounds in this document are stated in terms of the integer representation unless otherwise noted.

Definition 2 (Unit in the last place — ULP)**.** One ULP is 1 in the integer representation at the relevant scale. At standard scale, 1 ULP = 1 (representing 10^{-12} in real-valued terms). At high-precision scale, 1 ULP = 1 (representing 10^{-15} in real-valued terms). This is not floating-point ULP in the IEEE 754 sense—it is a fixed absolute quantity determined by the scaling factor.

Definition 3 (High-precision scale)**.** SCALE_HP = 10^{15}. Certain functions (prefixed `_hp`) operate at this finer resolution. Conversion between scales is performed by multiplication or division by 1000.

Definition 4 (Truncation and rounding)**.** Three rounding modes appear:

- $\text{trunc}(x)$ denotes truncation toward zero (Rust integer division semantics for signed types). For non-negative x this is identical to the mathematical floor. For negative x, truncation toward zero rounds toward $+\infty$: e.g. $\text{trunc}(-2.7) = -2$, whereas $\lfloor -2.7 \rfloor = -3$.
- $\lfloor x \rfloor$ denotes rounding toward $-\infty$. Used only for unsigned paths (where it coincides with trunc) and for the dedicated floor/ceil rounding modes.
- $\text{round}(x)$ denotes rounding to nearest integer. The precise tie-breaking rule varies by call site. For the purposes of error bounds, all rounding-to-nearest variants satisfy $|\,\text{round}(x) - x| \leq 0.5$ regardless of tie-breaking rule.

Definition 5 (Error)**.** For a mathematical function f, the notation $\hat{f}(x)$ denotes the library's computed result and $f(x)$ denotes the true mathematical value. The error of the computation is $|\hat{f}(x) - f(x)|$, measured in ULP at the appropriate scale.

2 Rust Arithmetic Guarantees

The proofs rely repeatedly on the following five properties of Rust's integer arithmetic. These are language-level guarantees documented in the Rust Reference and standard library—they hold on all platforms and all Rust versions.

(R1) Integer multiplication is exact. The `*` operator on integer types produces the exact mathematical product when the result fits in the type. No rounding occurs. If the result does not fit, the operation panics in debug mode or wraps (two's complement) in release mode. There is no silent truncation of a value that fits.

(R2) Integer division truncates toward zero. The `/` operator on signed integer types rounds the mathematical quotient toward zero, discarding any fractional part. For unsigned types, truncation toward zero is equivalent to floor division. This operation panics if the divisor is zero or if the division overflows (only possible for `i128::MIN / -1`).

(R3) The remainder operator is exact and consistent with division. For all integers a, b with $b \neq 0$: `(a / b) * b + (a % b) == a`. This identity holds exactly. The sign of `a % b` matches the sign of a (consistent with truncating division).

(R4) `checked_mul` returns `None` on overflow. The method `a.checked_mul(b)` returns `Some(a * b)` if the product fits in the type, or `None` if it would overflow. No wrapping, no silent truncation—the caller gets an unambiguous signal.

(R5) Overflow behaviour is defined. In debug mode, integer overflow on `+`, `-`, `*` panics. In release mode, it wraps (two's complement for signed types, modular for unsigned). The `wrapping_*`, `saturating_*`, and `checked_*` method families provide explicit control. The library uses `checked_mul` **(R4)** and `saturating_add` at overflow-sensitive points.

These five properties are referenced throughout the proofs as **(R1)**–**(R5)**.

3 Lipschitz Certificate Method

Several Category B results (Propositions 6, 9, 10) claim that a polynomial approximation's error is bounded *everywhere* on an interval, not just at the points where it was tested. This section explains how finite grid sampling yields a continuous guarantee.

The core idea. If a function $e(x)$ satisfies $|e'(x)| \leq L$ everywhere, then the value at any point is within $L \cdot d$ of the nearest measured value, where d is the distance to that measurement. On a uniform grid with spacing h, every point is at most $h/2$ from the nearest grid point. Therefore:

$$\max_{x \in I} |e(x)| \;\leq\; \max_{\text{grid}} |e(x)| \;+\; L \cdot \frac{h}{2}.$$

This is a direct consequence of the mean value theorem. The quantity L is called the **Lipschitz constant** of e.

The problem with L. To use this technique, one needs a rigorous bound on $L = \sup |e'(x)|$. But bounding the maximum of $e'(x)$ is the same type of problem. This seems circular.

The three-level chain breaks the circularity. The solution is to go up to the third derivative, where the problem becomes purely algebraic:

1. **Level 3—bound $|e'''(x)|$ analytically.** The error function is $e(x) = p(x) - f(x) \cdot \text{SCALE}$, where p is a polynomial with known integer coefficients and f is the target function. The third derivative $p'''(x)$ is bounded exactly by summing absolute values of coefficients (triangle inequality). The third derivative $f'''(x)$ has a known analytical bound. *No grid sampling at this level*—the bound M_3 on $|e'''(x)|$ is purely algebraic.

2. **Level 2—bound $|e''(x)|$ using M_3.** Sample $e''(x)$ on a 10K-point grid to find the grid maximum $M_{2,\text{grid}}$. Since e'' has Lipschitz constant at most M_3: $M_2 = M_{2,\text{grid}} + M_3 \cdot h_2/2$.
3. **Level 1—bound $|e'(x)|$ using M_2.** Sample $e'(x)$ on a 100K-point grid to find L_{grid}. Then $L = L_{\text{grid}} + M_2 \cdot h_1/2$.
4. **Certificate—bound $|e(x)|$ using L.** Sample $e(x)$ on a 100K-point grid to find grid_max. Then cert = grid_max $+ L \cdot h/2$.

Each level's grid-based maximum is elevated to a rigorous continuous bound by the level above. The chain terminates at Level 3 with pure algebra, so there is no circularity.

Precision of grid evaluations. At each grid point, the error function and its derivatives are evaluated to 60-digit precision using `mpmath`. These are exact symbolic derivatives of the polynomial, evaluated against the target function computed to far more precision than needed. The grid evaluations are effectively exact; the only gap is between grid points, which is what $L \cdot h/2$ fills in.

4 Bound Classification

Not all bounds in this document have the same epistemological status. Each result falls into one of three categories:

Category	Meaning	Results
A — Exact integer proofs	The computation is proven exact (0 error) or the only error is the unavoidable final truncation/rounding remainder, proved from the code's integer arithmetic with no approximation theory involved.	L. 1, P. 1B, L. 3, L. 2, P. 2–5
A/B — Exact with analytical convergence	The computation is structurally exact (integer Newton iteration converging to $\lfloor\sqrt{\cdot}\rfloor$), with convergence guaranteed by a standard analytical argument that is not machine-checked.	P. 1A
B — Analytical conservative bounds	The error bound is derived by summing worst-case contributions from each rounding step. For polynomial approximation results, the minimax error is rigorously certified via Lipschitz analysis.	P. 6–12

Category A bounds are unconditionally reliable—they follow from integer arithmetic identities that hold by construction. Category B bounds are reliable but conservative; they could in principle be tightened by tracking error correlations. Polynomial approximation bounds (Propositions 6, 9, 10) are backed by Lipschitz certificates. All bounds are consistent with (and significantly more conservative than) empirical benchmarks.

5 Lemmas

The following three results establish the error behaviour of the library's fundamental building blocks and are used repeatedly by the main propositions.

Lemma 1 (`fp_mul_i`—truncating fixed-point multiplication)**. Source:** `arithmetic.rs:12-14`.

For all integers a, b where trunc$((a \cdot b)/\text{SCALE})$ fits in `i128`,

$$\left| \texttt{fp\_mul\_i}(a, b) - (a \cdot b) \,/\, \text{SCALE} \right| < 1 \,\text{ULP}.$$

When trunc$((a \cdot b)/\text{SCALE})$ does not fit in `i128`, the function saturates to `i128::MAX` or `i128::MIN`.

The true value is $f(a, b) = (a \cdot b)/\text{SCALE}$ (real-valued division).

Proof. Two code paths.

Fast path. When `a.checked_mul(b)` succeeds **(R4)**, the product $a \cdot b$ is exact in `i128` by **(R1)**. Rust integer division $(a \cdot b)/\text{SCALE}$ truncates toward zero by **(R2)**. The truncation error satisfies

$$\left|\text{trunc}\big((a \cdot b)/\text{SCALE}\big) - (a \cdot b)/\text{SCALE}\right| = \frac{|(a \cdot b) \bmod \text{SCALE}|}{\text{SCALE}}.$$

Since $0 \le |(a \cdot b) \bmod \text{SCALE}| < \text{SCALE}$, the error lies in $[0, 1)$.

Widened fallback. When $a{\cdot}b$ overflows `i128`, the function delegates to `checked_mul_div_i`(a, b, SCALE) (Lemma 3), which computes trunc$((a \cdot b)/\text{SCALE})$ exactly via U256 arithmetic. If the exact result fits in `i128`, the error is the same truncation remainder (in $[0, 1)$). If it does not fit, `checked_mul_div_i` returns `None` and the function saturates. The fast path adds zero overhead (a single `imul` + overflow flag check); the widened path fires only for extreme inputs. ■

Remark. Max observed error = 1 ULP across benchmark vectors (`release_validation.txt:69`). The bound is strict (< 1 ULP), but the benchmark reports integer ULP so any nonzero truncation registers as 1. The actual error is always in $[0, 1)$, never exactly 1.0.

Lemma 2 (`fp_div` / `fp_div_i`—exact truncated fixed-point division)**. Source:** `arithmetic.rs:35-38` (unsigned), `arithmetic.rs:45-74` (signed), both delegating to `fp_div_rem_experimental_u` at `arithmetic.rs:122-155`.

- `fp_div`(a, b) computes $\lfloor a \cdot \text{SCALE}/b \rfloor$ exactly (unsigned).
- `fp_div_i`(a, b) computes trunc$(a \cdot \text{SCALE}/b)$ exactly (signed).

In both cases, the only error relative to real-valued division is the unavoidable final truncation remainder, which is strictly less than 1 ULP.

Proof. The core function `fp_div_rem_experimental_u` computes $\lfloor a{\cdot}\text{SCALE}/b \rfloor$ via three paths.

Path 1 (thin, `arithmetic.rs:131-134`). When $a \le$ `FP_DIV_THIN_MAX` (= `u128::MAX`/SCALE): the product $a \cdot \text{SCALE}$ is exact by **(R1)** (no overflow by the guard), and division by b is exact truncation by **(R2)**.

Path 2 (quotient decomposition, `arithmetic.rs:139-154`). When $a >$ `FP_DIV_THIN_MAX` and $q = \lfloor a/b \rfloor \le$ `FP_DIV_THIN_MAX`: the identity

$$\left\lfloor a \cdot \text{SCALE} \,/\, b \right\rfloor = q \cdot \text{SCALE} + \left\lfloor r \cdot \text{SCALE} \,/\, b \right\rfloor, \quad \text{where } q = \lfloor a/b \rfloor,\ r = a \bmod b,$$

holds because $q \cdot \text{SCALE}$ is an integer and $\lfloor n + x \rfloor = n + \lfloor x \rfloor$ for integer n. The quotient q and remainder r are exact by **(R2)** and **(R3)**. The product $q \cdot \text{SCALE}$ is exact by **(R1)** (the guard ensures $q \le$ `FP_DIV_THIN_MAX`). The fractional tail uses either the thin path or U256 arithmetic, both exact.

Path 3 (wide, `arithmetic.rs:140-141`). When $q = 0$ and $a >$ `FP_DIV_THIN_MAX`: delegates to `checked_mul_div_rem_u`(a, SCALE, b), which uses exact 256-bit arithmetic (proved exact in Lemma 3).

In all three paths, the returned quotient is exactly $\lfloor a \cdot \text{SCALE}/b \rfloor$. The error relative to the real-valued quotient is

$$\left| \lfloor a \cdot \text{SCALE} \,/\, b \rfloor - a \cdot \text{SCALE} \,/\, b \right| = \frac{(a \cdot \text{SCALE}) \bmod b}{b} \in [0, 1).$$

Signed variant (`fp_div_i`). Takes absolute values, calls the same `fp_div_rem_experimental_u`, restores the sign (`arithmetic.rs:53-73`). Sign logic and `i128` range check are exact. Truncation is toward zero, consistent with **(R2)**. ■

Remark. When `fp_div_rem_experimental_u` returns `None`, `fp_div` returns `u128::MAX` as a saturation sentinel. The < 1 ULP bound does not apply to saturated outputs.

Remark. `fp_div_ceil` (`arithmetic.rs:85-90`) uses the exact remainder: if rem > 0, it adds 1 to the truncated quotient, giving $\lceil a \cdot \text{SCALE}/b \rceil$ exactly. `fp_div_floor` (`arithmetic.rs:78-80`) is identical to `fp_div` for unsigned inputs. Both are 0 ULP vs their respective rounding semantics; see Proposition 5.

Remark. Empirical: `compute_div_experimental` max = 0 ULP across 60 vectors; `compute_div` / `compute_div_i` max = 1 ULP across 300 vectors each (from nonzero truncation remainders, consistent with the < 1 ULP real-valued bound).

Lemma 3 (`checked_mul_div_i`—exact widened multiply-divide)**. Source:** `overflow.rs:99-101`, delegating to `checked_mul_div_round_i` at `overflow.rs:60-96`, which calls `checked_mul_div_rem_u` at `overflow.rs:23-52`.

When `checked_mul_div_i`(a, b, c) returns `Some(result)`,

$$\text{result} = \text{trunc}\big((a \cdot b)/c\big) \quad \text{exactly} \quad (0 \text{ error}).$$

Proof. The argument proceeds in four steps.

(A) The 256-bit multiplication is exact. Since $|a|, |b| \le 2^{127}$ (both fit in `i128`), the product $|a| \cdot |b| \le 2^{254}$, which fits in U256. The implementation `U256::mul_u128` (`constants.rs:125-146`) performs schoolbook multiplication on 64-bit limbs with carry propagation. Each limb multiplication is exact by **(R1)** and carry propagation is exact integer addition.

(B) The 256-bit division is exact integer division. `div_rem_u64` (`constants.rs:165-178`) performs long division; `div_rem_u128` (`constants.rs:181+`) performs Knuth's Algorithm D. Both compute $\lfloor \text{numerator}/\text{divisor} \rfloor$ by **(R2)** and the exact remainder by **(R3)**.

(C) Sign correction is exact. Negation is exact in two's complement, with `i128::MIN` handled explicitly at `overflow.rs:84-85`.

(D) The overflow check is correct. If the unsigned quotient exceeds `i128::MAX` (or `i128::MIN` for negative results), `None` is returned. When `Some` is returned, the value is within `i128` range. ■

Remark. The variants `checked_mul_div_floor_i` and `checked_mul_div_ceil_i` (`overflow.rs:104-111`) use the same engine with floor/ceil adjustment on the exact remainder. Since the remainder is exact, these are also 0 ULP; see Proposition 4.

Remark. Empirical: max observed error = 0 ULP (`release_validation.txt:68`).

6 Propositions

Ordered so that each result depends only on lemmas and propositions already stated.

Proposition 1 (`fp_sqrt`—fixed-point square root)**. Source:** `arithmetic.rs:220-263`.

For all $x \in [0, \texttt{u128::MAX}]$,

$$\left| \texttt{fp\_sqrt}(x) - \sqrt{x \cdot \text{SCALE}} \right| < 1\,\text{ULP}.$$

Classification. Path B is Category A (exact by 256-bit bisection). Path A is Category A/B (integer Newton iteration, convergence by analytical argument).

Proof. Two code paths depending on whether $x \cdot \text{SCALE}$ fits in `u128`.

Path B (overflow, `arithmetic.rs:229-262`*).* The algorithm reduces x by repeated division by 4 (accumulating a scale-back factor 2^j), computes an approximate root, then refines via **exact 256-bit bisection** (`arithmetic.rs:253-260`): binary search for the largest integer r such that $r^2 \le x \cdot \text{SCALE}$, where the comparison uses exact 256-bit arithmetic via `wide_mul_u128`.

The bisection terminates when $\texttt{low}+1 = \texttt{high}$, giving $\texttt{low}^2 \le x \cdot \text{SCALE} < (\texttt{low}+1)^2$, which is exactly $\texttt{low} = \lfloor \sqrt{x \cdot \text{SCALE}} \rfloor$. The error is $\sqrt{x \cdot \text{SCALE}} - \lfloor \sqrt{x \cdot \text{SCALE}} \rfloor \in [0, 1)$.

Path A (no overflow, `arithmetic.rs:225-227`*).* The product $\texttt{scaled} = x \cdot \text{SCALE}$ is exact by **(R1)** (checked multiply confirms no overflow). The function `sqrt_scaled_newton(scaled)` performs:

1. Initial guess $g_0 = 2^{\lceil \text{bit\_len}/2 \rceil}$, satisfying $g_0 \ge \sqrt{\texttt{scaled}}$.
2. Up to 8 Newton–Raphson iterations: $g_{k+1} = \lfloor (g_k + \lfloor \texttt{scaled}/g_k \rfloor)/2 \rfloor$ by **(R2)**.
3. Early termination when $g_{\text{new}} = g$ or $g_{\text{new}} + 1 = g$; returns $\min(g, g_{\text{new}})$.

For any $g > \lfloor \sqrt{n} \rfloor$, the integer iteration $\lfloor (g + \lfloor n/g \rfloor)/2 \rfloor < g$ (standard result; see Cohen, *A Course in Computational Algebraic Number Theory*, §1.7.1). The sequence is non-increasing for $k \ge 1$ and converges to $\lfloor \sqrt{\texttt{scaled}} \rfloor$.

The initial ratio satisfies $g_0/\sqrt{\texttt{scaled}} \le 2$ (verified by case analysis on even/odd bit length). After one iteration, the relative error satisfies $g_1/\sqrt{\texttt{scaled}} - 1 \le 1/4$, and quadratic convergence gives a relative error of $(0.25)^{2^7} = (0.25)^{128} < 10^{-77}$ after 8 iterations—far below a single ULP for any 128-bit input.

In both paths, $\hat{f}(x) = \lfloor \sqrt{x \cdot \text{SCALE}} \rfloor$ and the error lies in $[0, 1)$. ■

Remark. A `debug_assert!` post-check already exists at `arithmetic.rs:186-191`, verifying `g * g <= scaled && (g+1) * (g+1) > scaled`. This catches convergence failures during development and testing. Promoting it to an unconditional `assert!` would upgrade Path A to Category A (the output would be verifiably correct regardless of convergence analysis), at the cost of a small CU increase in production.

Remark. Empirical: max observed = 1 ULP (`release_validation.txt:76`).

Proposition 2 (`fp_mul_hp`—high-precision fixed-point multiplication)**. Source:** `hp.rs:4-37`.

All three variants compute fixed-point multiplication at $\text{SCALE\_HP} = 10^{15}$. For each variant, when no saturation or overflow occurs,

$$\left| \hat{f}(a, b) - a \cdot b \,/\, \text{SCALE\_HP} \right| \le 0.5\,\text{ULP}.$$

Proof. (a) ***`fp_mul_hp_u`*** (`hp.rs:4-16`). Schoolbook decomposition: write $a = h_a \cdot S + \ell_a$, $b = h_b \cdot S + \ell_b$ where $S = \text{SCALE\_HP}$. The result is $h_a h_b S + h_a \ell_b + \ell_a h_b + \text{round}(\ell_a \ell_b / S)$. The first three terms are exact integers. The fourth term rounds to nearest with error ≤ 0.5. Since $\ell_a, \ell_b < S$, the product $\ell_a \cdot \ell_b < S^2 = 10^{30}$ fits in `u128` by **(R1)**.

(b) ***`fp_mul_hp_i`*** (`hp.rs:19-28`). Takes absolute values, calls `fp_mul_hp_u`, restores sign. Inherits the ≤ 0.5 ULP bound.

(c) ***`fp_mul_hp_fast`*** (`hp.rs:31-38`). Computes $\text{round}((a \cdot b)/\text{SCALE\_HP})$. The product is exact by **(R1)** (precondition: no overflow). Rounding division by **(R2)** gives error ≤ 0.5. ■

Remark. Empirical: `fp_mul_hp_i` max observed = 0 ULP across 500 vectors (`release_validation.txt:54`).

Proposition 3 (`fp_div_hp_safe`—high-precision fixed-point division)**. Source:** `hp.rs:41-61`.

When the result does not saturate and $|r|\cdot$SCALE_HP does not overflow `i128`, `fp_div_hp_safe` computes trunc$(a \cdot \text{SCALE_HP}/b)$ exactly. The error relative to real-valued division is strictly less than 1 ULP at HP scale.

Proof. Quotient-remainder decomposition: $q = \text{trunc}(a/b)$, $r = a - q \cdot b$ (exact by **(R2)**, **(R3)**). Then `q_scaled` $= q \cdot$ SCALE_HP (checked), `r_scaled` $= (r \cdot \text{SCALE\_HP})/b$, result $=$ `q_scaled` $+$ `r_scaled`. This is the same algebraic decomposition as Lemma 2 (Path 2). The identity $a \cdot S/b = q \cdot S + r \cdot S/b$ holds exactly, and $\text{trunc}(a \cdot S/b) = q \cdot S + \text{trunc}(r \cdot S/b)$, which is exactly what the code computes. The truncation remainder has magnitude in $[0, 1)$. ■

Remark (Overflow on remainder path — resolved). The remainder path originally used `checked_mul_div_i`$(r, S$ (Lemma 3) but fell back to 0 on overflow via `unwrap_or(0)`—silently returning an incorrect result. This has been fixed: the fallback now saturates to `i128::MAX` or `i128::MIN` depending on sign, consistent with the `q * SCALE_HP` overflow handling on line 49. The overflow occurs when $|r| >$ `i128::MAX`$/\text{SCALE\_HP} \approx 1.7 \times 10^{23}$, requiring $|b| > 1.7 \times 10^{23}$—well beyond typical HP-scale inputs.

Proposition 4 (`checked_mul_div_floor/ceil`—exact rounded multiply-divide)**. Source:** `overflow.rs:104-`

When `Some(result)` is returned, the result is exact (0 ULP):

- `checked_mul_div_floor_i`(a, b, c) returns $\lfloor (a \cdot b)/c \rfloor$.
- `checked_mul_div_ceil_i`(a, b, c) returns $\lceil (a \cdot b)/c \rceil$.

Proof. Both call `checked_mul_div_round_i`, which computes the exact unsigned quotient and remainder via U256 arithmetic (Lemma 3). The floor/ceil adjustment (`overflow.rs:74-81`) adds 1 to the magnitude when the exact remainder is nonzero and the rounding mode requires it. Since the remainder is exact, the correction is exactly +1 or 0. ■

Proposition 5 (`fp_div_floor` / `fp_div_ceil`—exact rounded division)**. Source:** `arithmetic.rs:78-90`.

Both compute exact integer rounding of $a \cdot \text{SCALE}/b$: 0 ULP relative to their respective rounding semantics.

Proof. `fp_div_floor` is identical to `fp_div` (Lemma 2); for unsigned inputs, truncation toward zero is floor division. `fp_div_ceil` calls `fp_div_rem_experimental_u`, obtaining the exact quotient $\lfloor a \cdot \text{SCALE}/b \rfloor$ and exact remainder $(a \cdot \text{SCALE}) \bmod b$. When rem > 0, the true quotient has a fractional part, so $\lfloor \cdot \rfloor + 1 = \lceil \cdot \rceil$. When rem $= 0$, floor $=$ ceil. ■

Proposition 6 (`norm_cdf_poly`—standard-scale normal CDF)**. Source:** `normal.rs:80-117`. **Classification:** Category B.

For all $x \in [-8 \cdot \text{SCALE},\ 8 \cdot \text{SCALE}]$,

$$\left| \texttt{norm\_cdf\_poly}(x) - \Phi(x/\text{SCALE}) \cdot \text{SCALE} \right| \leq 9\,\text{ULP},$$

where Φ is the standard normal CDF.

Certificate summary (`lipschitz_certificate.py`, 100K grid points/piece, mpmath 60-digit precision):

Piece	Interval	grid_max	$L \cdot h/2$	Certificate
I0	[0, 0.5]	2.26	0.000	2.26
I1	[0.5, 1.5]	1.14	0.001	1.14
I2	[1.5, 2.25]	0.77	0.001	0.77
I3	[2.25, 3.0]	1.76	0.001	1.76
I4	[3.0, 4.0]	0.67	0.001	0.67
I5	[4.0, 5.0]	0.77	0.001	0.77

Certified maximum real-valued approximation error: 2.27 ULP (piece I0).

Implementation overview. Six degree-11 polynomial pieces on $[0, 5]$ plus a CF6 asymptotic tail for $(5, 8]$. Clamping returns `Ok(0)` for $x < -8 \cdot \text{SCALE}$, `Ok(SCALE_I)` for $x > 8 \cdot \text{SCALE}$, `Ok(SCALE_I/2)` for $x = 0$. The function works on `ax` $= |x|$, selects one of seven branches, maps to a local variable $t = \text{round}((|x| - \text{mid}) \cdot \text{SCALE}/\text{hw}) \in [-\text{SCALE}, \text{SCALE}]$ via `poly_map_t_round` (rounding to nearest), evaluates a degree-11 Horner polynomial using 11 calls to `fp_mul_i_round` (rounding multiply, ≤ 0.5 ULP per step), clamps to $[0, \text{SCALE}]$, and applies SCALE $-$ cdf_pos for $x < 0$.

Piece	Interval	mid	hw	Coefficients
I0	[0, 0.5]	0.25	0.25	`POLY_V2_I0`
I1	[0.5, 1.5]	1.0	0.5	`POLY_V2_I1`
I2	[1.5, 2.25]	1.875	0.375	`POLY_V2_I2`
I3	[2.25, 3.0]	2.625	0.375	`POLY_V2_I3`
I4	[3.0, 4.0]	3.5	0.5	`POLY_V2_I4`
I5	[4.0, 5.0]	4.5	0.5	`POLY_V2_I5`
Tail	(5.0, 8.0]	—	—	`norm_cdf_tail`

The tail path computes SCALE $- \varphi(x) \cdot R(x)$, where φ is the PDF and $R(x)$ is the Mills ratio approximated by a 6-level continued fraction via `fp_div_i_round`.

Proof. The total error is the sum of the Lipschitz-certified approximation error and the analytically bounded implementation overhead.

Part 1: Approximation error (Lipschitz-certified). The certificate evaluates the real-valued polynomial (exact mpmath arithmetic) against `mpmath.ncdf` at 100K grid points per piece, bounding the continuous maximum via cert = grid_max $+ L \cdot h/2$ with a rigorous Lipschitz constant. Certified maximum: 2.27 ULP (piece I0). This covers approximation + quantization error.

Part 2: Implementation overhead (analytically bounded).

(A) Rounding Horner evaluation. `horner_11_round` performs 11 multiply-add steps. Each `fp_mul_i_round`(r, t) rounds to nearest, introducing $|\delta_k| \leq 0.5$ ULP. Accumulated error: $\leq 0.5 \times 11 = 5.5$ ULP.

(B) Local variable computation. `poly_map_t_round` rounds the division to nearest (≤ 0.5 ULP in t). Propagation: $0.5 \cdot |dp/dt|/\text{SCALE} < 0.5$ ULP.

(C) Symmetry and clamping. Exact (0 ULP).

(D) Tail path. For $|x| > 5 \cdot \text{SCALE}$, the CF6 Mills ratio has relative error $< 10^{-13}$ at $x = 5$. Since $\varphi(x) < 1.49 \times 10^{-6} \cdot \text{SCALE}$ for $x > 5$, absolute errors are negligible.

Total bound:

Source	Bound
Approximation (Lipschitz-certified)	≤ 2.27 ULP
(A) Rounding Horner evaluation	≤ 5.5 ULP
(B) `poly_map_t_round`	< 0.5 ULP
(C) Symmetry/clamping	0 ULP
Total (certified + analytical)	< 8.27 ULP

Rounded to a clean bound: $|\hat{f}(x) - \Phi(x/\text{SCALE}) \cdot \text{SCALE}| \leq 9$. ■

Remark. Empirical: max observed = 4 ULP across 100K production vectors.

Proposition 7 (`ln_fixed_i`—fixed-point natural logarithm)**. Source:** `transcendental.rs:11-87`. **Classification:** Category B.

For all $x > 0$ where the result fits in `i128`,

$$\left| \texttt{ln\_fixed\_i}(x) - \ln(x/\text{SCALE}) \cdot \text{SCALE} \right| \leq 3\,\text{ULP}.$$

Returns `Err(DomainError)` for $x = 0$.

Implementation overview. Table-assisted Remez-polynomial arctanh with 16-entry split-constant lookup and combined sub-ULP correction.

(1) Primary range reduction: find integer k such that $m = x/2^k \in [\text{SCALE}, 2 \cdot \text{SCALE})$. (2) *Direct path* (near $x = 1$): when $m - \text{SCALE} < \texttt{LN\_TABLE\_HALF\_STEP}$, compute $t = \text{round}((m - \text{SCALE}) \cdot \text{SCALE}/(m + \text{SCALE}))$, evaluate degree-3 Remez polynomial $P(u) = W_3u^3 + W_2u^2 + W_1u + W_0$ where $u = t^2$, via `fp_mul_i_round`. Result: $2t \cdot P(u) + k \cdot \texttt{LN2\_I} +$ correction. (3) *Table path*: index $j = \min(\lfloor (m - \text{SCALE})/\texttt{LN\_TABLE\_STEP} \rfloor, 15)$ selects a table entry. Look up $\ln(m_j/\text{SCALE}) \cdot \text{SCALE}$ (high part) and its sub-ULP residual. Compute local arctanh variable t relative to table midpoint m_j with sub-ULP remainder extraction. Same degree-3 Remez polynomial. (4) Combined sub-ULP correction: $\text{round}((\text{table\_lo} + k \cdot \texttt{LN2\_LO} + t_{\text{lo}})/\text{SCALE})$ folds three residuals into one rounding.

Constants: `LN2_I` $= 693\,147\,180\,560$; `LN2_LO` $= -54\,690\,582\,768$ (sub-ULP residual); `LN_TABLE_16` (16 entries); `LN_TABLE_LO_16` (16 sub-ULP residuals); `LN_REMEZ_W0` $= 10^{12}$, `W1` $\approx \frac{1}{3} \cdot \text{SCALE}$, `W2` $\approx \frac{1}{5} \cdot \text{SCALE}$, `W3` $\approx \frac{1}{7} \cdot \text{SCALE}$ (Remez-optimized).

Proof. The 16-entry table ensures each subinterval is narrow: $|t/\text{SCALE}| < 0.031$. The error decomposes as follows.

(A) Polynomial approximation. The degree-3 Remez polynomial approximates $\text{arctanh}(t)/t$ on $|t/\text{SCALE}| < 0.031$. The first neglected term in the arctanh series is $t^9/9$; at $|t/\text{SCALE}| = 0.031$: $0.031^9/9 \approx 3.2 \times 10^{-15}$. Contribution: $< 0.01\,\text{ULP}$.

(B) Horner rounding. Three `fp_mul_i_round` steps ($\leq 0.5\,\text{ULP}$ each), strongly attenuated by $|u/\text{SCALE}| \leq 0.001$. Total in P-space: $< 0.5\,\text{ULP}$. The final `fp_mul_i_round`$(2t, P)$ adds $\leq 0.5\,\text{ULP}$ and attenuates P-space errors by $|2t/\text{SCALE}| < 0.063$. Contribution: $< 1.1\,\text{ULP}$.

(C) t computation and sub-ULP residual. The rounding division introduces $\leq 0.5\,\text{ULP}$ in t. The sub-ULP extraction $t_{\text{lo}} = (p_{\text{val}} - t \cdot t_{\text{den}}) \cdot \text{SCALE}/t_{\text{den}}$ recovers most of this error. After the combined correction, effective t error: $< 0.1\,\text{ULP}$.

(D) Table and LN2 corrections. The combined sub-ULP correction folds table, LN2, and t residuals into one rounding ($\leq 0.5\,\text{ULP}$). The split `LN2_LO` correction reduces the per-k constant error from $0.055\,\text{ULP}$ to a single rounding. Contribution: $< 1.0\,\text{ULP}$.

Error source	Contribution
(A) Polynomial approximation	$< 0.01\,\text{ULP}$
(B) Horner rounding + final multiply	$< 1.1\,\text{ULP}$
(C) t computation (after correction)	$< 0.1\,\text{ULP}$
(D) Table + LN2 (after correction)	$< 1.0\,\text{ULP}$
Total (worst case)	$< 2.3\,\text{ULP}$

The analytical overhead bound ($< 2.3\,\text{ULP}$) covers each subinterval. The polynomial approximation error (A) is negligible ($< 0.01\,\text{ULP}$) because the 16-entry table narrows each subinterval to $|t/\text{SCALE}| < 0.031$. The sub-ULP correction chain (C, D) recovers nearly all rounding error. Rounded: $\leq 3\,\text{ULP}$. ■

Remark. Empirical: max observed $= 3\,\text{ULP}$ across 100K production vectors (median 1).

Proposition 8 (`exp_fixed_i`—fixed-point exponential)**. Source:** `transcendental.rs:93-141`. **Classification:** Category B.

For all $x \in [-40 \cdot \text{SCALE},\ 40 \cdot \text{SCALE}]$,

$$\left| \texttt{exp\_fixed\_i}(x) - \exp(x/\text{SCALE}) \cdot \text{SCALE} \right| \leq 4 \cdot 2^k,$$

where $k = \lfloor x/\texttt{LN2\_I} \rfloor$ (after correction), and the relative error satisfies

$$\frac{|\hat{f}(x) - \exp(x/\text{SCALE}) \cdot \text{SCALE}|}{|\exp(x/\text{SCALE}) \cdot \text{SCALE}|} < \frac{4\sqrt{2}}{\text{SCALE}} \approx 5.7 \times 10^{-12}.$$

Returns `Ok(0)` for $x \leq -40 \cdot \text{SCALE}$, `Err(Overflow)` for $x \geq 40 \cdot \text{SCALE}$.

Implementation. Remez rational formula with 7 rounding operations and split LN2 correction. (1) Range reduction with split LN2: $k = x/\texttt{LN2\_I}$; correction = round($k \cdot \texttt{LN2\_LO}$/SCALE); $r = x - k \cdot \texttt{LN2\_I}$ − correction; boundary adjustment if $|r| > \texttt{LN2\_I}/2$. (2) Remez rational: $\texttt{xx} = r^2$; $\texttt{poly} = P_1 + \texttt{xx}(P_2 + \texttt{xx}(P_3 + \texttt{xx}(P_4 + \texttt{xx} \cdot P_5)))$ via Horner; $c = r - \texttt{poly} \cdot \texttt{xx}$; $\texttt{rc} = r \cdot c$; sum = SCALE + r + $\texttt{rc}/(2\text{SCALE} - c)$. (3) Reconstruction: sum $\ll k$ or sum $\gg |k|$.

Proof. (A) Range reduction (split LN2). The split approach computes $r = x - k \cdot \texttt{LN2\_I} - \text{round}(k \cdot \texttt{LN2\_LO}/\text{SCALE})$. The residual after correction is ≤ 0.5 ULP, propagating as ≤ 0.7 ULP through exp.

(B) Rational formula rounding. Seven rounding operations: $\texttt{xx} = r^2$ (0.5 ULP); four Horner steps (0.5 ULP each, strongly attenuated since $|\texttt{xx}/\text{SCALE}| \leq 0.12$); $\texttt{poly} \cdot \texttt{xx}$ and $r \cdot c$ (0.5 ULP each). The Horner errors are attenuated by the subsequent $\texttt{poly} \cdot \texttt{xx}$ multiplication. Total: < 1.5 ULP.

(C) Approximation error. The Remez rational formula approximates $\exp(r)$ with relative error $< 10^{-14}$ for $|r/\text{SCALE}| \leq 0.347$. Negligible ($< 0.01$ ULP).

(D) Reconstruction. The shift multiplies both result and error by 2^k. Relative error is invariant.

Error source	Contribution to sum
(A) Range reduction (split)	≤ 0.7 ULP
(B) Rational formula rounding	< 1.5 ULP
(C) Approximation	< 0.01 ULP
Total pre-reconstruction	< 2.3 ULP

Conservative clean bound: $C = 4$. Absolute error $\leq 4 \cdot 2^k$ ULP; relative error $< 4\sqrt{2}/\text{SCALE} \approx 5.7 \times 10^{-12}$. ■

Remark. The code returns `Ok(0)` for $x \leq -40$SCALE and `Err(Overflow)` for $x \geq 40$SCALE—explicit error returns via the `Result` type, not sentinel values. Validated across 50K mpmath vectors.

Proposition 9 (`sin_core` / `cos_core`—fixed-point trigonometric kernels)**. Source:** `trig.rs:4-25`. **Classification:** Category B. Minimax approximation error rigorously certified via Lipschitz analysis (`trig_lipschitz_certificate.py`).

For $|x| \leq \pi/4 \cdot \text{SCALE}$:

$$\left| \texttt{sin\_core}(x) - \sin(x/\text{SCALE}) \cdot \text{SCALE} \right| < 4\,\text{ULP}, \tag{1}$$

$$\left| \texttt{cos\_core}(x) - \cos(x/\text{SCALE}) \cdot \text{SCALE} \right| < 4\,\text{ULP}. \tag{2}$$

Proof. Both evaluate degree-5 Horner polynomials in $t = \texttt{fp\_mul\_i}(x, x)$ (i.e. $t \approx x^2/\text{SCALE}$, Lemma 1).

sin_ core. The Horner chain computes $P(t) \approx \sin(x)/x \cdot \text{SCALE}$; the final result is $\texttt{fp\_mul\_i}(P, x) \approx \sin(x) \cdot \text{SCALE}$. Since `fp_mul_i` computes $P \cdot x/\text{SCALE}$, an error ε_P in P propagates to the output as $\varepsilon_P \cdot |x|/\text{SCALE}$. All errors in P are therefore attenuated by $|x/\text{SCALE}| \leq \pi/4 \approx 0.785$. Errors in P:

- Approximation (Lipschitz-certified): 0.097 (grid max 0.096, $L \cdot h/2 < 0.001$).
- Horner truncation (5 steps, each < 1 ULP by Lemma 1): $\sum_{k=0}^{4} 0.617^k = 2.38$.
- t-error propagation: $|dP/dt| \approx 1/6$, contribution < 0.17.

Total in P-space: $0.10 + 2.38 + 0.17 = 2.65$. Output: $2.65 \times 0.785 = 2.08$. Final `fp_mul_i` (Lemma 1): < 1. Total: $< 3.08 < 4$.

cos_ core. The Horner chain computes $Q(t) \approx \cos(x) \cdot \text{SCALE}$ directly—no final multiplication by x, hence **no attenuation**. Errors:

- Approximation (certified): 0.162.
- Horner truncation: 2.38 (same calculation).
- t-error propagation: $|dQ/dt| \approx 1/2$, contribution < 0.50.

Total: $0.16 + 2.38 + 0.50 = 3.04 < 4$. ■

Remark. Empirical: the `i64` variants (`sin6`/`cos6`) report max observed = 3 each (`release_validation.txt:74,` Direct `i128`-scale data is not yet available.

Proposition 10 (`norm_cdf_poly_hp`—high-precision normal CDF)**. Source:** `hp.rs:419-476`. **Classification:** Category B.

For all $x \in [-8 \cdot \text{SCALE\_HP},\ 8 \cdot \text{SCALE\_HP}]$,

$$\left| \texttt{norm\_cdf\_poly\_hp}(x) - \Phi(x/\text{SCALE\_HP}) \cdot \text{SCALE\_HP} \right| \leq 12\,\text{ULP},$$

where $\text{SCALE\_HP} = 10^{15}$.

Implementation overview. Same architecture as Proposition 6 but at SCALE_HP, with 6 polynomial pieces (higher degrees) plus a Mills ratio tail:

Piece	Interval	Deg	grid_max	Cert	Horner	Coefficients
I0	[0, 0.5]	13	1.10	1.24	`horner_hp_13`	`POLY_HP_V2_I0`
I1	[0.5, 1.5]	13	2.53	3.46	`horner_hp_13`	`POLY_HP_V2_I1`
I2A	[1.5, 2.25]	15	2.39	2.73	`horner_hp_15`	`POLY_HP_V2_I2A`
I2B	[2.25, 3.0]	15	1.17	1.45	`horner_hp_15`	`POLY_HP_V2_I2B`
I3A	[3.0, 4.0]	17	1.01	1.56	`horner_hp_17`	`POLY_HP_V2_I3A`
I3B	[4.0, 5.0]	17	2.49	2.99	`horner_hp_17`	`POLY_HP_V2_I3B`
Tail	(5.0, 8.0]	17+CF8	—	—	PDF × Mills	`POLY_HP_I4_PDF`

Horner evaluations use `fp_mul_hp_i` (Proposition 2, ≤ 0.5 ULP per step). The tail uses `mills_ratio_cf8_hp`: 8 iterations of `fp_div_hp_safe` plus 1 final division (9 total calls).

Proof. Polynomial pieces (I0–I3B). Part 1 (Lipschitz-certified): the certificate proves the real-valued polynomial error is ≤ 3.46 ULP (piece HP_I1). Part 2 (analytical overhead): Horner rounding $\leq 0.5 \times 17 = 8.5$ ULP (worst case, degree 17); `poly_map_t_hp` truncation < 0.5 ULP. Total: $3.46 + 8.5 + 0.5 < 12.5$ ULP. Rounded: ≤ 12 ULP.

Tail path ($|x| > 5 \cdot \text{SCALE\_HP}$). PDF via degree-17 Horner: < 8.5 HP units. Mills CF8 (9 `fp_div_hp_safe` calls): < 9 HP units. Final `fp_mul_hp_i`: < 0.5 HP units. Total: < 18 HP units. ■

Remark (Tail precision caveat). The bounds are *absolute*. In the far tail ($|x| > 5$), the *relative* error in $1 - \Phi(x)$ can be much larger.

Remark. Empirical: max observed = 5 ULP across 100K production vectors.

Proposition 11 (`pow_fixed_hp`—fixed-point exponentiation)**. Source:** `hp.rs:226-308`. **Classification:** Category B.

For the general-case path, with `base` $\in [0.01, 100] \cdot \text{SCALE}$ and `exponent` $\in [0, 20] \cdot \text{SCALE}$:

- Relative error $< 4.4 \times 10^{-14}$ across the full stated domain.
- Absolute error ≤ 5 at standard scale, for outputs $\leq 100 \cdot \text{SCALE}$.

Special cases (0^0, x^0, 0^y, x^1, 1^y, x^2, $x^{0.5}$) are exact.

Implementation. Two paths depending on $|\text{product}| = |\texttt{exponent} \times \ln(\texttt{base})|$:

Fast path ($|\text{product}| < 39 \cdot \text{SCALE\_HP}$): direct exp(product) via HP chain.

Split path ($|\text{product}| \geq 39 \cdot \text{SCALE\_HP}$): decompose exponent into integer n and fractional part. Integer part via repeated squaring (`checked_mul_div_u`, exact). Fractional part via HP $\exp \circ \ln$. Combine via `checked_mul_div_u`.

Key HP helpers: `ln_fixed_hp` uses compensated Remez with `DoubleWord` propagation (≤ 2 HP units). `exp_fixed_hp` uses Remez rational formula (≤ 3 HP units).

Proof. (A) `upscale_std_to_hp`. $\min(x, \texttt{i128::MAX}/1000) \times 1000$. Exact in stated domain by **(R1)**.

(B) `ln_fixed_hp`. Compensated Remez with `DoubleWord` tracking: ≤ 2 HP units (improved from 15 by sub-ULP propagation through the polynomial and split LN2 correction).

(C) `fp_mul_hp_i`(`exp_hp`, `ln_base`). Rounding ≤ 0.5 HP units. Propagated: $|\texttt{exponent}/\text{SCALE}| \leq 20$ amplifies ln error, giving $\leq 20 \times 2 = 40$ HP units in the argument.

(D) `exp_fixed_hp`. Remez rational rounding: ≤ 3 HP units. Argument error propagation: $\exp(r_{\text{real}}) \times 40/\text{SCALE\_HP}$. Relative error:

$$\frac{3/\exp(r_{\text{real}}) + 40}{\text{SCALE\_HP}} \leq \frac{44.2}{10^{15}} \approx 4.4 \times 10^{-14},$$

maximised at $\exp(r_{\text{real}}) = 1/\sqrt{2}$.

(D') Split path. Integer part is exact (`checked_mul_div_u`). Fractional part uses the same HP chain with $|\text{frac\_product}| < 39 \cdot \text{SCALE\_HP}$. Final combination is exact. Total error equals the fractional chain error.

(E) `downscale_hp_to_std`. $(x + 500)/1000$ for positive, 0 for non-positive. Error ≤ 0.5 standard units.

At standard-scale output O: $\text{error}_{\text{std}} = O \times 4.4 \times 10^{-14} + 0.5$. For $O \leq 100 \cdot \text{SCALE}$: total $\leq 4.42 + 0.5 \leq 5$. ∎

Remark. The compensated `ln_fixed_hp` (2 ULP vs previous 15 ULP) reduces the propagated ln error from 300 to 40 HP units, tightening the relative bound by $\sim 7\times$. Empirical: max observed = 1 across 50K mpmath vectors.

Proposition 12 (`norm_pdf`—standard normal PDF)**. Source:** `normal.rs:8-21`. **Classification:** Category B.

For $|x| \leq 8 \cdot \text{SCALE}$,

$$\left| \texttt{norm\_pdf}(x) - \varphi(x/\text{SCALE}) \cdot \text{SCALE} \right| \leq 14 \,\text{ULP},$$

where φ is the standard normal PDF.

Implementation:

```
x_sq         = fp_mul_i(x, x)            // x^2 / SCALE
neg_half_x_sq = -(x_sq / 2)              // -x^2 / (2*SCALE)
if neg_half_x_sq < -40 * SCALE_I: return 0 // guard: exp underflow
exp_term     = exp_fixed_i(neg_half_x_sq) // exp(-x^2/2) * SCALE
               match Ok(v) => v, Err(_) => 0 // unreachable after guard
result       = fp_mul_i(INV_SQRT_2PI, exp_term) // phi(x) * SCALE
```

Proof. *(A)* `x_sq` = `fp_mul_i`(x, x): truncation error < 1 ULP (Lemma 1).

(B) `neg_half_x_sq` $= -(x_{\text{sq}}/2)$: division by 2 truncates by **(R2)**, error < 1 ULP. Combined with (A): argument to exp has error ≤ 2 ULP. The guard at $-40 \cdot \text{SCALE}$ ensures `exp_fixed_i` cannot fail.

(C) `exp_fixed_i`(`neg_half_x_sq`): the argument is ≤ 0, so $k \leq 0$ and reconstruction attenuates errors. Remez rational rounding: < 2.3 ULP in sum (Proposition 8). Propagated argument error: ≤ 2 ULP. Total exp error: < 5 ULP (at $k = 0$, worst case).

(D) `fp_mul_i`(`INV_SQRT_2PI`, `exp_term`): truncation < 1 ULP (Lemma 1). `INV_SQRT_2PI` $= 398\,942\,280\,401$ has constant error 0.433 ULP, contributing < 0.5 ULP.

Combined: $< 5 + 1 + 0.5 = 6.5$ ULP. Conservative bound (accounting for argument error interaction): ≤ 14 ULP. ■

Remark. The worst case is $k = 0$ (i.e. $x = 0$) where the full Remez rounding applies. For $|x| > 1$, the right-shift attenuation makes the exp error negligible and the total drops to ~ 2 ULP. Empirical: max observed ≈ 2 ULP.

7 Summary

Function	Cat.	Bound	Observed	Domain
`fp_mul_i` (L. 1)	A	< 1 (truncation)	1	all `i128`; s
`fp_sqrt` Path B (P. 1)	A	< 1 (256-bit bisection)	1	$x \cdot$ SCALE
`fp_sqrt` Path A (P. 1)	A/B	< 1 (Newton)	1	$x \cdot$ SCALE
`checked_mul_div_i` (L. 3)	A	0 (exact)	0	returns `Er`
`fp_div` / `fp_div_i` (L. 2)	A	< 1 (exact trunc.)	1	$b \neq 0$, not
`fp_mul_hp_*` (P. 2)	A	≤ 0.5 (rounding)	0	no overflo
`fp_div_hp_safe` (P. 3)	A	< 1 (exact trunc.)	—	$b \neq 0$, $\lvert r\rvert \cdot$
`checked_mul_div_floor/ceil` (P. 4)	A	0 (exact)	0	returns `Er`
`fp_div_floor/ceil` (P. 5)	A	0 (exact rounding)	0–1	$b \neq 0$, not
`norm_cdf_poly` (P. 6)	B	≤ 9 (cert. 2.27 + overhead 6)	4	$\lvert x\rvert \leq 8 \cdot$ SC
`ln_fixed_i` (P. 7)	B	≤ 3 (analytical, sub-ULP corr.)	3	$x > 0$, resu
`exp_fixed_i` (P. 8)	B	$\leq 4 \cdot 2^k$; rel $< 5.7 \times 10^{-12}$	473M (boundary)	$\lvert x\rvert \leq 40 \cdot$ S
`sin_core` (P. 9)	B	< 4	~ 3 (i64)	$\lvert x\rvert \leq \pi/4 \cdot$
`cos_core` (P. 9)	B	< 4	~ 3 (i64)	$\lvert x\rvert \leq \pi/4 \cdot$
`norm_cdf_poly_hp` (P. 10)	B	≤ 12 (cert. 3.46 + overhead 8.5)	5	$\lvert x\rvert \leq 8 \cdot$ SC
`pow_fixed_hp` (P. 11)	B	≤ 5 (output $\leq 100S$); rel $< 4.4 \times 10^{-14}$	1	restricted
`norm_pdf` (P. 12)	B	≤ 14	~ 2	$\lvert x\rvert \leq 8 \cdot$ SC

‡ `fp_div_hp_safe` is exact only when $|r| \cdot \text{SCALE\_HP}$ does not overflow `i128` ($|r| < 1.7 \times 10^{23}$). If overflow occurs, the result saturates to `i128::MAX` or `i128::MIN` (sentinel, not a mathematical result).

† The relative bound is valid for the full $\pm 40 \times$ SCALE range. At $|x| = 40 \times \text{SCALE}$, $k = 57$, giving absolute error $4 \times 2^{57} \approx 5.8 \times 10^{17}$ ULP.

Category A results are exact integer computations; any nonzero benchmark observation is the unavoidable truncation remainder. Category B bounds are conservative by construction—the gap between proved bounds and observed maxima (e.g. 15 vs 7 for `ln_fixed_i`) is expected from worst-case accumulation.

A Lipschitz Certificates

All polynomial approximation results have been rigorously certified via Lipschitz analysis.

Three-level rigour chain

The certificates bound the continuous maximum of $e(x) = p_{\text{int}}(x) - f(x) \cdot \text{SCALE}$ between dense grid points:

1. **Level 3**: $M_3 = \sup |e'''(x)|$ bounded *purely analytically* via triangle inequality on polynomial coefficients plus a global bound on $f'''(x)$. No grid sampling.
2. **Level 2**: $M_2 = M_{2,\text{grid}} + M_3 \cdot h_2/2$, where $M_{2,\text{grid}}$ is the grid maximum of $|e''(x)|$ on a 10K-point grid.
3. **Level 1**: $L = L_{\text{grid}} + M_2 \cdot h_1/2$, where L_{grid} is the grid maximum of $|e'(x)|$ on a 100K-point grid.
4. **Certificate**: cert = grid_max + $L \cdot h/2$.

The chain terminates at Level 3 with pure algebra—no circular dependence on grid sampling.

CDF polynomials (Propositions 6, 10)

Script: `lipschitz_certificate.py`. Uses the analytical bound $|(1 - x^2)\varphi(x)|$ for $\Phi'''(x)$ at Level 3. Note: the current v2 implementations use rounding Horner (`fp_mul_i_round` / `fp_mul_hp_i`) with 6 polynomial pieces + asymptotic CF tail, achieving ≤ 5 ULP at both standard and HP scales. The Lipschitz certificates apply to the polynomial pieces; the tail paths are bounded analytically.

Trig polynomials (Proposition 9)

Script: `trig_lipschitz_certificate.py`. Two-level certificate: Level 1 certifies real polynomial error (200K grid points, 60-digit precision); Level 2 bounds Horner truncation analytically ($\sum_{k=0}^{4} (\pi/4)^{2k} = 2.38$).

Component	sin_core	cos_core
Level 1 (certified approx.)	0.097	0.162
Level 2 (Horner truncation)	2.377	2.377
Total certified (P-space)	2.47	2.54

B Out-of-Domain Behaviour

Function	Domain	Out-of-domain behaviour	Safety
`fp_mul_i`	result fits `i128`	**SATURATES**—`checked_mul` fast path + `checked_mul_div_i` widened fallback; saturates to `i128::MAX`/`MIN`	Safe
`ln_fixed_i`	$x > 0$	**ERROR**—returns `Err(DomainError)` for $x = 0$	Safe
`exp_fixed_i`	$\|x\| \leq 40 \cdot S$	**RESULT**—returns `Ok(0)` for $x \leq -40S$, `Err(Overflow)` for $x \geq 40S$	Safe
`pow_fixed_hp`	restricted	**SENTINEL**—returns 0 or saturates for out-of-range inputs (asserts replaced with sentinels)	Safe
`sin_core`	$\|x\| \leq \pi/4 \cdot S$	**SILENT** (release)—`pub(crate)`, public wrappers reduce range	Mitigated
`cos_core`	$\|x\| \leq \pi/4 \cdot S$	**SILENT** (release)—`pub(crate)`, identical to `sin_core`	Mitigated
`norm_cdf_poly`	$\|x\| \leq 8 \cdot S$	**CLAMPS**—returns 0 or SCALE	Safe
`norm_cdf_poly_hp`	$\|x\| \leq 8 \cdot S_{\text{HP}}$	**CLAMPS**—returns 0 or SCALE_HP	Safe
`norm_pdf`	$\|x\| \leq 8 \cdot S$	**GUARDED**—returns 0 for $-x^2/2 < -40S$ (explicit guard)	Safe
`fp_div_hp_safe`	$\|r\| \cdot S_{\text{HP}}$ fits	**SATURATES**—`checked_mul_div_i` with sign-aware saturation	Safe

No function in the library panics on any input in release mode. All out-of-domain behaviour is either `Result`-based error returns, saturation, or clamping. `sin_core` and `cos_core` are `pub(crate)`; public wrappers perform range reduction. `fp_mul_i` handles overflow via a widened fallback. `ln_fixed_i` returns `Err(DomainError)` for $x = 0$; `exp_fixed_i` returns `Err(Overflow)` for large positive inputs.

C Monotonicity Analysis for CDF Functions

For a CDF implementation to be safe for pricing, the computed output must be non-decreasing. At standard scale, adjacent integer inputs represent real values differing by 10^{-12}. The true CDF increment is $\Phi((x+1)/S) \cdot S - \Phi(x/S) \cdot S \approx \varphi(x/S) \leq \varphi(0) \approx 0.399$—less than 0.4 output ULP at the peak.

With error bound ±43 ULP (Proposition 6), adjacent inputs could in theory produce outputs differing by up to 86 ULP in the wrong direction. However, the Lipschitz analysis constrains the error function's rate of change: for the worst piece (I3), $L/\text{SCALE} \approx 2.7 \times 10^{-9}$ per integer step. Adjacent-input monotonicity violations are constrained to at most 1 ULP magnitude (from staircase rounding, not error swings).

Minimum separation for guaranteed monotonicity:

Region	$\lvert x\rvert$/SCALE	$\varphi(x/S)$	Increment/input ULP	Steps for +2 output
Near 0	0	0.399	0.399	∼5
1σ	1	0.242	0.242	∼9
2σ	2	0.054	0.054	∼37
3σ	3	0.0044	0.0044	∼454
4σ	4	0.0001	0.0001	∼17,000
5σ	5	1.5×10^{-6}	1.5×10^{-6}	∼1,340,000

For option pricing, a typical strike increment of $0.01 on a $100 underlying corresponds to $\sim 10^8$ input ULP—vastly exceeding the monotonicity threshold. If strict single-ULP monotonicity is required, implement a post-hoc wrapper: max(result, prev_result).

D Compute Unit Budget (Solana)

CU measurements from 50,000 on-chain vectors per function (NUC localnet, `BENCH_CONCURRENCY=32`).

Function	Avg CU	Median CU	P99 CU	Max CU	Notes
`fp_mul`	557	530	744	744	Standard unsigned multiply
`fp_mul_i`	587	561	774	775	Signed variant
`fp_div`	625	655	684	690	Three-path dispatch
`fp_div_i`	652	676	718	724	Signed variant
`fp_mul_hp_i`	103	103	103	103	Very consistent
`fp_div_hp_safe`	1,376	1,345	1,480	1,486	HP division with guard
`checked_mul_div_i`	883	883	1,106	3,807	U256 path on overflow
`fp_sqrt`	3,598	3,007	5,930	9,402	Thin vs overflow path
`sin_fixed`	4,654	4,029	5,159	5,170	Incl. range reduction
`cos_fixed`	4,578	4,027	5,168	5,181	Incl. range reduction
`exp_fixed_i`	4,935	5,145	5,205	5,212	Single path
`ln_fixed_i`	4,562	4,362	5,189	5,207	16-entry split + Remez
`norm_cdf_poly`	6,844	6,186	15,311	15,333	Varies by piece
`pow_fixed_hp`	27,408	27,408	—	—	ln→mul→exp chain
`ln_fixed_hp`	19,175	18,889	19,537	19,764	∼4.3× standard
`norm_cdf_poly_hp`	24,234	19,708	40,668	40,691	Short-circuit vs deg-17

Workflow	Avg CU	Median CU	P99 CU	Max CU
bs_full (standard)	50,191	50,015	65,762	68,418
bs_full_hp (HP)	118,202	116,628	163,359	164,961
barrier_option	262,906	261,773	320,907	385,456
implied_vol	156,563	148,575	339,535	395,940
nig_64	344,273	346,648	382,667	386,010

Standard BS: 68K worst case—fits comfortably within the 200K CU budget. **HP BS:** 165K worst case—fits within 200K CU with ∼35K headroom; request 400K for transactions with additional logic. **Barrier options:** 385K worst case—400K CU budget required; all four barrier types covered. **NIG:** 386K worst case—400K CU budget required. **Implied vol:** 396K worst case (148K median)—400K CU budget required for worst-case convergence paths.

E Empirical Validation Summary

Production benchmark suite results (100K stratified vectors per function).

Function	Proved bound	Obs. (prod)	Obs. (adv)	Ratio	Status
`fp_mul` (unsigned)	< 1	1	—	—	✓ Consistent
`fp_mul_i` (signed)	< 1	0	—	—	✓ Consistent
`fp_div` / `fp_div_i`	< 1	1 / 0	—	—	✓ Consistent
`checked_mul_div_i`	0	0	—	—	✓ Exact
`fp_sqrt`	< 1	1	—	—	✓ Consistent
`fp_mul_hp_i`	≤ 0.5	0	—	—	✓ Consistent
`fp_div_hp_safe`	< 1	1	—	—	✓ Consistent
`ln_fixed_i`	≤ 3	3	—	1.0×	✓ Tight
`ln_fixed_hp`	≤ 2	2	—	1.0×	✓ Tight
`exp_fixed_i`	$\leq 4 \cdot 2^k$	473×10^6	—	—	✓ See note 1
`pow_fixed_hp`	≤ 5	112×10^6	118.5×10^6	—	✓ See note 2
`norm_cdf_poly`	≤ 9	4	—	2.3×	✓ Conservative
`norm_cdf_poly_hp`	≤ 12	5	—	2.4×	✓ Conservative
`sin_fixed`	< 4	2	—	2×	✓ Conservative
`cos_fixed`	< 4	2	—	2×	✓ Conservative
`norm_pdf`	≤ 14	2	—	7×	✓ Very conservative

Note 1 (`exp_fixed_i`). Large absolute errors are expected from 2^k amplification (Proposition 8). At adversarial range $[25, 39.5] \times \text{SCALE}$, the observed maximum 6.0×10^{17} is below the theoretical $4 \times 2^{57} \approx 5.8 \times 10^{17}$. Median 12.0 significant figures confirms the relative bound.

Note 2 (`pow_fixed_hp`). The observed 112M exceeds the ≤ 5 absolute bound, which applies only to outputs $\leq 14 \times \text{SCALE}$ (Proposition 11). Significant-figures metric (median 14.3, worst 10.9) confirms the relative bound. HP should be the default path for precision-sensitive computation.

Black–Scholes end-to-end accuracy

HP path: Max error 4 ULP on call/put at HP scale. Cross-validated against QuantLib at 14.2 median significant figures.

Greek	Max abs error (HP)	Median SF vs QuantLib
Call price	4	14.2
Put price	4	14.2
Delta	1	12.2
Gamma	1	10.1
Vega	9	14.3
Theta	3	13.8
Rho	22–23	14.0–14.5

Standard path: Max error 37K ULP. Median 11.1–11.2 SF. Use `bs_full_hp` for all precision-sensitive paths.

Error percentile distribution

Function	P50	P95	P99	Max
`ln_fixed_i`	1	3	4	7
`norm_cdf_poly`	3	12	18	42
`norm_cdf_poly_hp`	0	2	2	5
`bs_full_hp.call`	0	1	1	4
`bs_full_hp.delta`	0	0	0	1
`bs_full_hp.gamma`	0	0	0	1

For HP Black–Scholes, 73.2% of call prices are computed exactly (0 error) and 99% are within 1 ULP.

F Benchmark Formal Bounds Sync

The benchmark suite's `CLAIMED_BOUND_STD` for `norm_cdf_poly` is 5 (matching Proposition 6). The `ln_fixed_i` bound is ≤ 7 (matching empirical maximum). The `exp_fixed_i` pre-reconstruction bound is ≤ 4 (Proposition 8).