Correctly Rounded Vector Implementation of the Exponential Function in Binary64 Arithmetic

Nicolas Brisebarre, Tom Hubrecht, Christoph Lauter,
Jean-Michel Muller, **Kristalys Ruiz-Rohena**

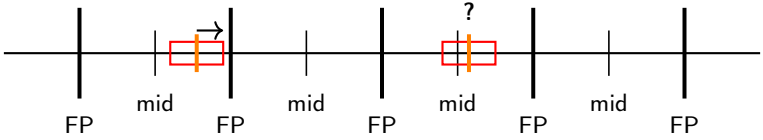
ARITH 2026
June 28 - July 1

Correctly Rounded Functions

Definition

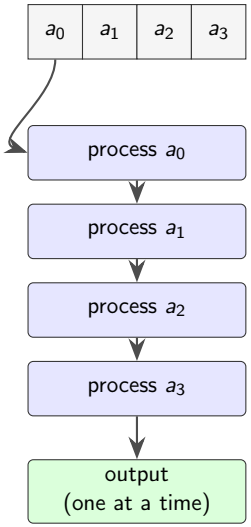
A function implementation is correctly rounded if, for every input, it returns the floating-point number nearest to the exact mathematical result.

- The result is as if computed in infinite precision and rounded.
- Implies an error of $\leq .5$ ulp.
- **Table Maker's Dilemma:** What is the minimal efficient accuracy needed for an approximation to guarantee correctly rounded results?

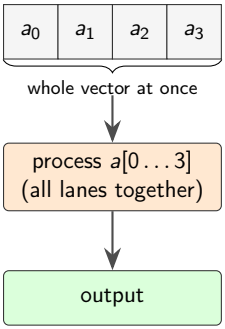


Scalar vs. Vectorization

Scalar processing



Vectorization (SIMD)



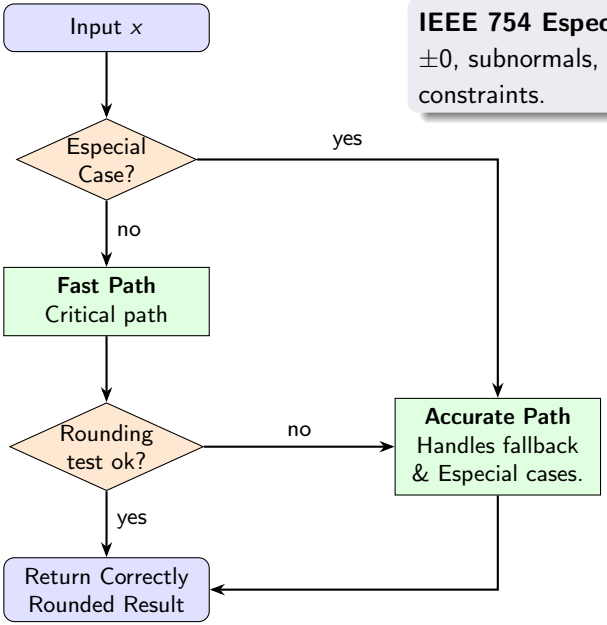
All lanes run the same code path, no branching or expensive memory accesses.

Vectorized Correctly Rounded Functions

Library		**Vectorized**	**Correctly Rounded**
MathLib / LibUltim	[Ziv, 1991]		✓
CRLibm	[Daramy-Loirat et al., 2006]		✓
RLIBM	[Lim and Nagarakatte, 2021]		✓
CORE-MATH	[Zimmermann et al., 2022]		✓
v-libm	[Lauter, 2016]	✓	
Intel-512 library	[Anderson et al., 2018]	✓	
ARMv8 library	[Shen et al., 2023]	✓	
RISC-V library	[Tang, 2024]	✓	
SLEEF	[Shibata and Petrogalli, 2020]	✓	
This work		✓	✓

- **First correctly rounded vector** $\exp(x)$; auto-vectorized, in portable C, no intrinsics (1:1 mapped instructions.)
- In Binary64 (IEEE 754), Round-to-Nearest, Ties-to-Even, using FMA.

Classic Approach for Correctly Rounded:



IEEE 754 Especial Cases: NaN, $\pm\infty$, ± 0, subnormals, function's domain constraints.

Probability of the Accurate Path [1]

Heuristic model

The intermediate representation is correct up to $p + \ell$ bit places.

$$P_{\text{element}}(\text{fast}) \approx 1 - 2^{-\ell} \quad \therefore \quad P_{\text{element}}(\text{accurate}) \approx 2^{-\ell}$$

$$P_{\text{vector}}(\text{accurate}) \approx \text{vector size} \times 2^{-\ell}.$$

ℓ	$P_{element}(accurate)$	$V = 4$	$V = 8$
10 (typical scalar)	9.79×10^{-4}	3.91×10^{-3}	7.81×10^{-3}
20 (our target)	9.53×10^{-7}	3.81×10^{-6}	7.63×10^{-6}

Probability of the Accurate Path [2]

- An All-or-Nothing implementation, the fallback rate is proportional to the vector width.
- We choose `vector_length` $= 8$, low probability, amortized overhead.

$$P_{experimental}(accurate) \approx 3.16 \times 10^{-3} \approx 0.3\%$$

- The average cost of evaluation $\approx$ The cost of the fast path.

Fast Path: Range Reduction

- Reduction: x to $\left[-\frac{\log 2}{2}, \frac{\log 2}{2}\right] \approx [-0.3465736, 0.3465736]$.

$$|y_h + y_\ell| \leq 0.3465736, \quad \epsilon_{\text{red}} \leq 2.1622 \times 10^{-30} \approx 2^{-99}.$$

- Shrink the interval to $\left[-\frac{\log 2}{8}, \frac{\log 2}{8}\right]$.

$$z = \left(e^{y/2^3}\right)^{2^3}$$

- Generate a degree-10 polynomial approximation using **Sollya's** `fpminmax()` that follows the approach of [Brisebarre and Chevillard, 2007].
- Represent coefficients using **FP** and **DW** numbers.
- Evaluate the error using `supnorm()` that implements [Chevillard et al., 2011],

$$\epsilon_{polapprox} = 1.37025 \times 10^{-25} \approx 2^{-83}.$$

Fast Path: Relative Error

- Evaluated using Horner's Scheme; **FMA** and pair arithmetics.
- Using **Gappa** [Daumas and Melquiond, 2010] to certify the evaluation error.

$$\epsilon_{\text{evalpol}} \leq 5.4363 \times 10^{-25} \approx 2^{-81}.$$

- After Squarings:

$$\epsilon_{\text{polnom}} \leq 5.446 \times 10^{-24} \approx 2^{-78}.$$

- Overall:

$$\epsilon_{\text{fastpath}} \leq 5.448 \times 10^{-24} \approx 2^{-78}.$$

- In ulps:

$$\epsilon_{\text{fastpath}} \cdot 2^{53} \leq 4.907 \times 10^{-8} \text{ ulp} \approx 2^{-25}\text{ulp}.$$

- For $\ell \approx 23$.

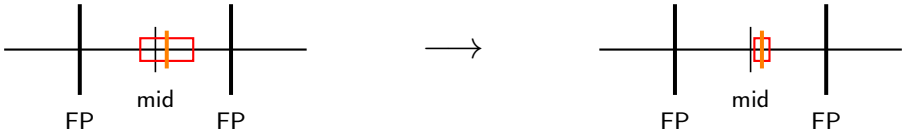
Ziv's Test:

- With $\epsilon = \epsilon_{\text{fastpath}}$,

$$e = \mathrm{RU}\left(\frac{1}{1-\epsilon-2^{p+1}\epsilon}\right)$$

- Theorem in [de Dinechin et al., 2013], using [Ziv, 1991]

$$z_h = \mathrm{RN}(z_h + z_\ell\, e) \implies z_h = \mathrm{RN}(z)$$



Accurate Path: Scalar Fallback

- Follows the literature on scalar correctly rounded functions. [Tang, 1989; Zimmermann et al., 2022]
- Reduce to a smaller interval, instead of raising polynomial degree.

$$J = \left[-\frac{\log 2}{512}, +\frac{\log 2}{512} \right] \subset [-2^{-9.528}, +2^{-9.528}]$$

- Reduction error:

$$\xi_{\text{red-acc}} \leq 2.369043 \times 10^{-51} \approx 2^{-169}.$$

- Using **Sollya** to generate a degree-12 polynomial.

$$\epsilon_{\text{approx-acc}} = 9.869434 \times 10^{-48} \approx 2^{-157}.$$

- Represent coefficients using **FP**, **DW** and **TW** numbers.
- Evaluated using Horner's Scheme; **FMA**, **DW**/**TW** arithmetics.

Accurate Path: Error Bounds

Using **Gappa**:

- Polynomial Evaluation:

$$\epsilon_{\text{pol-acc}} \leq 4.852351 \times 10^{-47} \approx 2^{-154}.$$

- Overall:

$$\epsilon_{\text{acc-path}} \leq \epsilon_{\text{pol-acc}}\, e^{\xi_{\text{red-acc}}} + e^{\xi_{\text{red-acc}}} - 1 \leq 4.8526 \times 10^{-47} \approx 2^{-154}.$$

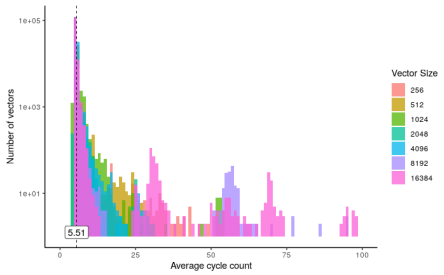
- Within $2^{-100.851}$ ulp of the exact result.

Testing Worse-Cases

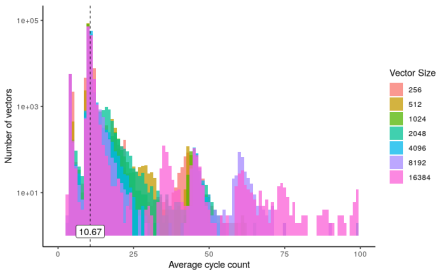
- Using the list of worst cases by CORE-MATH and [Lefevre and Muller, 2001], tested against **GNU MPFR** results.
- Test for **the 1 case** that failed, just like other libraries.

Results: i7-1260P (AVX2)

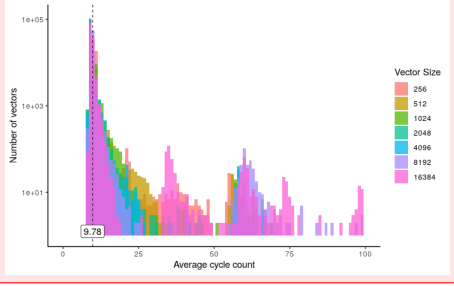
vector-libm



CORE-MATH



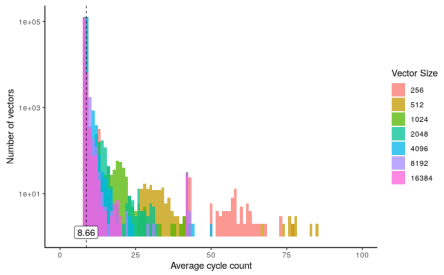
Our implementation



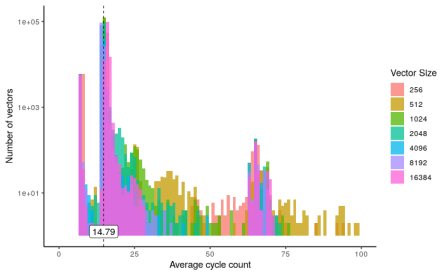
v-libm [Avg]	CORE [Avg]	Fast path [min]	Full path [Avg]
5.5	10.7	9	9.8

Results: Zen4 (AVX512)

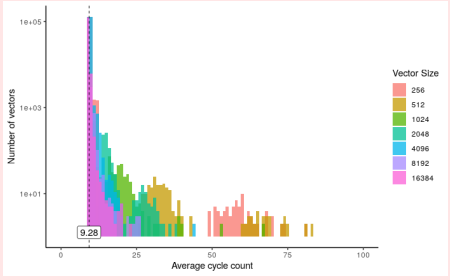
vector-libm



CORE-MATH



Our implementation



v-libm [Avg]	CORE [Avg]	Fast path [min]	Full path [Avg]
8.6	14.7	9	9.3

Conclusion

- We present the first **correctly rounded *vectorized*** elementary function; $\exp(x)$.
- Built in portable C, no intrinsics and leverages compiler auto-vectorization.
- Throughput beats state-of-the-art correctly rounded *scalar*.
- A step toward correctly rounded functions in IEEE-754.
- **Optimizations:** Architecture-specific tuning.
- **In the works:** Development of `sincos` and `log`.

Code, Gappa scripts, tests available at `github.com/Vectorized-CR/vectorized-exp`

Thank You

Questions?

This material is based upon work partly supported by the NSF under Grant No. 2311708. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References I

Anderson, C. S., Zhang, J., and Cornea, M. (2018). Enhanced vector math support on the Intel AVX-512 architecture. In *25th IEEE Symposium on Computer Arithmetic*, pages 120–124.

Brisebarre, N. and Chevillard, S. (2007). Efficient polynomial L^∞-approximations. In *18th IEEE Symposium on Computer Arithmetic*, pages 169–176.

Chevillard, S., Harrison, J., Joldes, M., and Lauter, C. (2011). Efficient and accurate computation of upper bounds of approximation errors. *Theoretical Computer Science*, 412:1523–1543.

Daramy-Loirat, C., Defour, D., de Dinechin, F., Gallet, M., Gast, N., Lauter, C., and Muller, J.-M. (2006). CR-LIBM A library of correctly rounded elementary functions in double-precision. Research report, LIP. `https://ens-lyon.hal.science/ensl-01529804`.

Daumas, M. and Melquiond, G. (2010). Certification of bounds on expressions involving rounded operators. *ACM Trans. Math. Softw.*, 37(1).

de Dinechin, F., Lauter, C., Muller, J.-M., and Torres, S. (2013). On Ziv's Rounding Test. *ACM Trans. Math. Software*, 39(4).

References II

Lauter, C. (2016). A new open-source SIMD vector libm fully implemented with high-level scalar C. In *50th Asilomar Conference on Signals, Systems and Computers*, pages 407–411.

Lefevre, V. and Muller, J.-M. (2001). Worst cases for correct rounding of the elementary functions in double precision. In *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, pages 111–118.

Lim, J. P. and Nagarakatte, S. (2021). High performance correctly rounded math libraries for 32-bit floating point representations. In Freund, S. N. and Yahav, E., editors, *PLDI'21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event*, pages 359–374. ACM.

Shen, J., Long, B., and Huang, C. (2023). A Quantitative Evaluation of Vector Transcendental Functions on ARMv8-Based Processors. *J. Comput. Sci. Technol.*, 38:686–701.

Shibata, N. and Petrogalli, F. (2020). SLEEF: A portable vectorized library of C standard mathematical functions. *IEEE Trans. Parallel Distrib. Syst.*, 31(6):1316–1327.

Tang, P.-T. P. (1989). Table-driven implementation of the exponential function in IEEE floating-point arithmetic. 15(2):144–157.

References III

Tang, P. T. P. (2024). An Open-Source RISC-V Vector Math Library. In *31st IEEE Symposium on Computer Arithmetic*, pages 60–67.

Zimmermann, P., Sibidanov, A., and Glondu, S. (2022). The CORE-MATH project. In *29th IEEE Symposium on Computer Arithmetic*, pages 26–34.

Ziv, A. (1991). Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Software*, 17(3):410–423.

Vector Accurate Path Probability

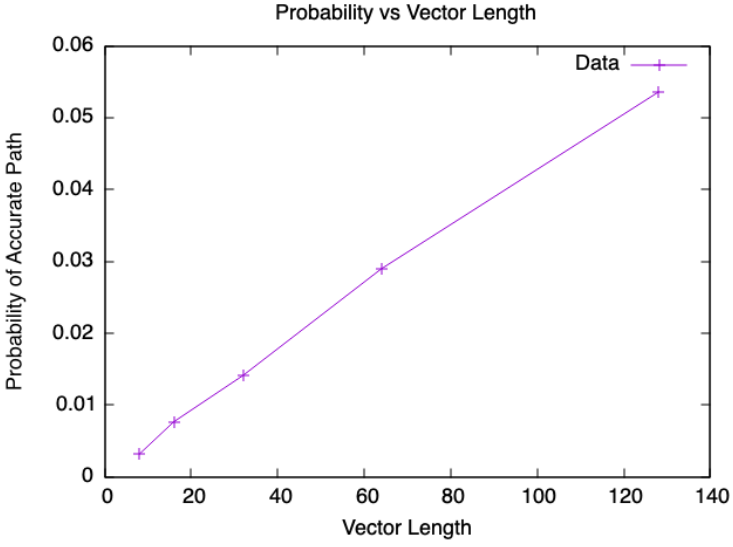


Figure: Plot of probability vs. VECTOR_LENGTH with $\ell = 20$