An Implementation of Sin and Cos Using Gal's Accurate Tables

Pascal Leroy (phl)

2025-02-02

This document describes the implementation of functions `Sin` and `Cos` in Principia. The goals of that implementation are to be portable (including to machines that do not have a fused multiply-add instruction), achieve good performance, and ensure correct rounding.

Overview

The implementation follows the ideas described by [GB91] and uses accurate tables produced by the method presented in [SZ05]. It guarantees correct rounding with a high probability. In circumstances where it cannot guarantee correct rounding, it falls back to the (slower but correct) implementation provided by the CORE-MATH project [SZG22] [ZSG+24]. More precisely, the algorithm proceeds through the following steps:

- perform argument reduction using Cody and Waite's algorithm in double precision (see [Mul+10, p. 379]);
- if argument reduction loses too many bits (i.e., the argument is close to a multiple of $\frac{\pi}{2}$), fall back to `cr_sin` or `cr_cos`;
- otherwise, uses accurate tables and a polynomial approximation to compute `Sin` or `Cos` with extra accuracy;
- if the result has a "dangerous rounding configuration" (as defined by [GB91]), fall back to `cr_sin` or `cr_cos`;
- otherwise return the rounded result of the preceding computation.

In this document we assume a base-2 floating-point number system with M significand bits[1] similar to the IEEE formats. We define a real function $\mathfrak{m}$ and an integer function $\mathfrak{e}$ denoting the *significand* and *exponent* of a real number, respectively:

$$x = \pm\,\mathfrak{m}(x) \times 2^{\mathfrak{e}(x)} \qquad \text{with} \qquad 2^{M-1} \leq \mathfrak{m}(x) \leq 2^{M} - 1$$

Note that this representation is unique. Furthermore, if x is a floating-point number, $\mathfrak{m}(x)$ is an integer.

The distance between 1 and the next larger floating-point number is:

$$\epsilon_M \coloneqq 2^{1-M}$$

and the distance between 1 and the next smaller floating-point number is $\frac{\epsilon_M}{2}$. The *unit of the last place* of x is defined as:

$$\mathfrak{u}(x) \coloneqq 2^{\mathfrak{e}(x)}$$

In particular, $\mathfrak{u}(1) = \epsilon_M$.

We ignore the exponent bias, overflow and underflow as they play no role in this discussion.

[1] In `binary64`, $M = 53$.

Argument Reduction

Given an argument x, the purpose of argument reduction is to compute a pair of floating-point numbers $(\hat{x}, \delta\hat{x})$ such that:

$$\begin{cases} \hat{x} + \delta\hat{x} \cong x \pmod{\frac{\pi}{2}} \\ \hat{x} \text{ is approximately in } \left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \\ |\delta\hat{x}| < \mathfrak{u}(\hat{x}) \end{cases}$$

Approximation of π

We approximate $\frac{\pi}{2}$ as the sum of two floating-point numbers:

$$\frac{\pi}{2} \cong C + \delta C$$

where C is obtained by truncating $\frac{\pi}{2}$ to κ_1 significand bits:

$$C \coloneqq \left\lfloor 2^{-\kappa_1}\, \mathfrak{m}\!\left(\frac{\pi}{2}\right)\right\rfloor 2^{\kappa_1}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)$$

and δC is defined as $\left[\!\left[\frac{\pi}{2} - C\right]\!\right]$. Obviously we have:

$$0 < \frac{\pi}{2} - C < 2^{\kappa_1}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)$$

but if κ_1 is chosen to cut the significand of $\frac{\pi}{2}$ at a place where it has zeroes, we can actually have a stricter bound:

$$\frac{\pi}{2} - C < 2^{\kappa_2}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right) \qquad \text{with} \qquad \kappa_2 \leq \kappa_1$$

and therefore:

$$\mathfrak{u}\!\left(\frac{\pi}{2} - C\right) < \frac{2^{\kappa_2}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)}{\mathfrak{m}\!\left(\frac{\pi}{2} - C\right)} \leq 2^{\kappa_2 - M + 1}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)$$

Since the function $\mathfrak{u}$ is always a power of 2 this implies:

$$\mathfrak{u}\!\left(\frac{\pi}{2} - C\right) = 2^{\kappa_2 - M}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)$$

and:

$$\left|\frac{\pi}{2} - C - \delta C\right| \leq \frac{1}{2}\, \mathfrak{u}\!\left(\frac{\pi}{2} - C\right) = 2^{\kappa_2 - M - 1}\, \mathfrak{u}\!\left(\frac{\pi}{2}\right)$$

In other words, we have a representation with a significand that has effectively $2M - \kappa_2$ bits and is such that multiplying C by an integer less than or equal to 2^{κ_1} is exact. The representation of $\frac{\pi}{2}$ has three zeroes after the 18th bit of its significand, so by taking $\kappa_1 = 18$ we have $\kappa_2 = 14$.

Argument Reduction for Small Angles

If $|x| < \left[\!\left[\frac{\pi}{4}\right]\!\right]$ then $\hat{x} = x$ and $\delta\hat{x} = 0$.

Argument Reduction for Medium Angles

If $|x| \leq 2^{\kappa_1}\left[\!\left[\frac{\pi}{2}\right]\!\right]$ then we compute:

$$\begin{cases} n &= \left[\!\left[\left[\!\left[x\left[\!\left[\frac{2}{\pi}\right]\!\right]\right]\!\right]\right]\!\right] \\ y &= x - n\, C \\ \delta y &= [\![n\, \delta C]\!] \\ \hat{x} &= [\![y - \delta y]\!] \\ \delta\hat{x} &= (y - \hat{x}) - \delta y \end{cases}$$

First, note that $|n| \leq 2^{\kappa_1}$. Using the accuracy model of [Higo2], equation (2.4), we have[2]:

$$\begin{aligned} |x| &\leq 2^{\kappa_1}\frac{\pi}{2}(1+\delta_1) \\ |n| &\leq \left\lceil 2^{\kappa_1}\frac{\pi}{2}(1+\delta_1)\frac{2}{\pi}(1+\delta_2)(1+\delta_3)\right\rfloor \\ &\leq \left\lceil 2^{\kappa_1}(1+\gamma_3)\right\rfloor \end{aligned}$$

where the notation follows [Higo2], lemma 3.1. Because $2^{\kappa_1}\gamma_3$ is very small (less that 2^{-33}), the rounding cannot cause n to exceed 2^{κ_1}.

The product $n\ C$ is exact thanks to the κ_1 trailing zeroes of C. The subtraction $x - n\ C$ is exact by Sterbenz's Lemma. Finally, the last two steps form a compensated summation so that $\hat{x} + \delta\hat{x} = y + \delta y$.

The overall error on $\hat{x} + \delta\hat{x}$ comes from the error on the computation of δy:

$$\delta y = n\delta C(1+\delta_4)$$

Argument Reduction for Large Angles

Accurate Tables and Their Generation

Computation of the Functions

References

[GB91] S. Gal and B. Bachelis. "An Accurate Elementary Mathematical Library for the IEEE Floating Point Standard". In: *ACM Transactions on Mathematical Software* 17.1 (Mar. 1991), pp. 26–45.

[Higo2] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, 2002.

[Mul+10] J.-M. Muller, N. Brisebarre, F. De Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic.* Birkhäuser, 2010.

[SZ05] D. Stehlé and P. Zimmermann. "Gal's accurate tables method revisited". In: *17th IEEE Symposium on Computer Arithmetic (ARITH'05)* (Cape Cod, MA, USA, June 27–29, 2005). Ed. by P. Montuschi and E. Schwarz. IEEE Computer Society, June 2005, pp. 257–264.
DOI: `10.1109/ARITH.2005.24`.

[SZG22] A. Sibidanov, P. Zimmermann, and S. Glondu. "The CORE-MATH Project". In: *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH).* IEEE, Sept. 2022, pp. 26–34.
DOI: `10.1109/ARITH54963.2022.00014`.
eprint: `https://inria.hal.science/hal-03721525v3/file/core-math-final.pdf`.

[ZSG+24] P. Zimmermann, A. Sibidanov, S. Glondu, et al. *The CORE-MATH Project.* Software. Apr. 2024.

[2] We note that in Higham's notation $u = \epsilon_M/2$, see pages 37 and 38.