Synthesizing complex exp() and complex log() for a Fine Musical Instrument

A. D. Levine
dl@acm.org

Abstract

Presented is a novel, extensible, purely mechanical basis for coding complex functions exp() and log(). This method dynamically generates all utilized fractional constants (from the integers zero through five) and implements recursive vector rotation requiring no corrective scaling, maintaining precision via the conservative properties of symplectic integration in both the circular domain and the hyperbolic domain, a significant improvement over CORDIC, with which the author has complete implementation experience.

1 Method

Conversion between rectangular and polar, and between linear and logarithmic or exponential, is a naturally occurring construct within the context of basic digital computation, in the form of a simple recursive pattern. The technique described creates implicit tracing of a dual set of curves: *the unit circle* and *the unit hyperbola*.

Circle Tracing

The Symplectic Oscillator[1] is designed for DSP (digital signal processing) hardware in musical application, generating cosine and sine waves, which graphed as orthogonal coordinates trace a circle, evolving state variables u and v that are *in quadrature*: u (*cosine*) is advanced in phase by $\frac{\pi}{2}$ radians relative to v (*sine*). This evolution is via symplectic integration: of u as the derivative of v, and of $-v$ as the derivative of u. The signed computation sequence to evolve the state variables via symplectic integration is:

$$\pm\Omega_{uv}^{wz} \triangleq \left[\begin{cases} u{+}{=}zv,\ v{+}{=}2wu,\ u{+}{=}zv & \text{forward,} \\ u{-}{=}zv,\ v{-}{=}2wu,\ u{-}{=}zv & \text{reverse.} \end{cases} \right] \tag{1.1}$$

With the parameter α expressing the *fraction of revolution* per integration, the coefficients w and z are:

$$w \triangleq +\sin(\pi\alpha)\cos(\pi\alpha)^{+1} \tag{1.2}$$
$$z \triangleq -\sin(\pi\alpha)\cos(\pi\alpha)^{-1} \tag{1.3}$$

which given to $+\Omega_{uv}^{wz}$ is algebraically equivalent to multiplication of a vector by a rotation matrix:

$$\begin{bmatrix} \cos(2\pi\alpha) & -\sin(2\pi\alpha) \\ +\sin(2\pi\alpha) & \cos(2\pi\alpha) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{1.4}$$

Matrix rotation degrades with repetition, however. Rotation by symplectic integration is numerically stable, generally, though precision-stress error arises as the $\cos(\pi\alpha)$ term approaches zero, where a singularity exists. This method constrains integration measures to $|\alpha| \leq \frac{1}{4}$, and the precision utilized yields bit-accurate results.

Hyperbola Tracing

With the parameter χ expressing the *fraction of quadrupling* per integration, the coefficients w and z are:

$$w \triangleq +\sinh(\ln(2)\chi)\cosh(\ln(2)\chi)^{+1} \tag{1.5}$$
$$z \triangleq +\sinh(\ln(2)\chi)\cosh(\ln(2)\chi)^{-1} \tag{1.6}$$

which given to $+\Omega_{uv}^{wz}$ is algebraically equivalent to multiplication of a vector by an exponentiation matrix:

$$\begin{bmatrix} \cosh(2\ln(2)\chi) & +\sinh(2\ln(2)\chi) \\ +\sinh(2\ln(2)\chi) & \cosh(2\ln(2)\chi) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{1.7}$$

2 Implementation

This method is not focused on speed, it has been meticulously authored for extreme mathematical precision, design integrity, and engineering elegance, attributes demanded in the crafting of a *fine musical instrument*. The application runs on a powerful DSP chip, which like FPGAs does not do native floating-point divide. Functions coded in C yielding bit-exact results for an instance of the floating-point format (`'fp_t'`) include:

$$\mathrm{div}(n,d),\ \mathrm{sqrt}(x),\ \mathrm{exp2}([r,i]),\ \mathrm{log2}([r,i]),\ \exp([r,i]),\ \log([r,i]) \tag{2.1}$$

Base and Angle Measure

Constants π and $\ln(2)$ are applied as divisors to input values in exp() and multipliers to output values in log(), wrapping exp2() and log2(), which use $base_r$=2, and $angle\_measure_i$=2. Both π and $\ln(2)$ are operational logarithms initially generated bit-exact with a variant[2] of the function $\mathrm{agm}(a,g)$, arithmetic geometric mean:

$$\pi \equiv \mathrm{div}(\mathbf{4},\mathrm{agm}(\mathbf{0},\mathbf{2})) \tag{2.2}$$

$$\ln(2) \equiv \mathrm{div}(\mathbf{1},\mathrm{agm}(\mathrm{div}(\mathbf{5},\mathbf{3}),\mathrm{div}(\mathbf{4},\mathbf{3})) \tag{2.3}$$

Coefficient Synthesis

Two tables of integration coefficients are initially generated, each with depth $m = (\texttt{mantissa\_width(fp\_t)}+2)$ yielding bit-exactness, as exp2(), log2(), and the tables utilize multi-precision floating-point,[3] beginning with axiomatic positing of exact integration coefficient terms for measures double the largest that will be applied, then a signed computation sequence derives the integration coefficients by identity and recursive subdivision:

$$\left[\begin{cases} x{=}\mathbf{0},\ y{=}\mathbf{1}, & \text{circular} \implies \alpha \equiv \frac{1}{2} \wedge x \equiv \cos(\pi\alpha) \wedge y \equiv \sin(\pi\alpha), \\ x{=}\mathrm{div}(\mathbf{5},\mathbf{4}),\ y{=}\mathrm{div}(\mathbf{3},\mathbf{4}), & \text{hyperbolic} \implies \chi \equiv 1 \wedge x \equiv \cosh(\ln(2)\chi) \wedge y \equiv \sinh(\ln(2)\chi). \end{cases}\right] \tag{2.4}$$

$$\left[x{+}{=}\mathbf{1},\ z_n{=}\mathrm{div}\left(\begin{cases} -y & \text{circular,} \\ +y & \text{hyperbolic.} \end{cases}, x\right),\ y{=}\mathrm{div}(y,\mathbf{2}),\ w_n{=}y,\ x{=}\mathrm{sqrt}(\mathrm{div}(x,\mathbf{2})),\ y{=}\mathrm{div}(y,x)\right]_{n=1}^{m} \tag{2.5}$$

Curve Measuring

The tables facilitate precise traversal between points in the collective interval of the coefficients, by convergent binary search via recursive application of signed symplectic integration. Composing this with scaling of the hyperbola by linear bias of angular measure, and folding of symmetry of the circle, maps all curve points to a bijective pair, comprising products of integrations from $\langle u,v\rangle = \langle 1,0\rangle$, and sums of angular measures, thus creating an exponential $\leftrightarrow$ logarithmic *recursive binary mapping* of the circular and the hyperbolic domains:

$$\left\{\langle u,v\rangle,\ \prod_{n=1}^{\{1,\ldots,m\}}\left\{\pm\Omega_{uv}^{w_n z_n}\right\}\right\} \overset{1:1}{\longleftrightarrow} \left\{0,\ \sum_{n=1}^{\{1,\ldots,m\}}\left\{\pm\left(2^{-n}\right)\right\}\right\} \tag{2.6}$$

3 Application

To exponentiate, $\langle 1,0\rangle$ is directionally integrated to converge angular measure with input component in both domains, and resulting circular components are then scaled with the sum of resulting hyperbolic components. To find a logarithm, $\langle 1,0\rangle$ is directionally integrated to converge the sum of resulting hyperbolic components with input magnitude, and a complex input is directionally integrated to converge angularly with $\langle 1,0\rangle$ in the circular, yielding both measures. As an example, a computation sequence to find $\ln(x)$ for $x > 0 \in \mathbb{R}$ is:

$$\left[\begin{array}{ll} \text{the hyperbola scales by powers of } 2^2\text{:} & scale{=}\mathrm{floating\_point\_exponent}(x),\ scale{-}{=}scale \bmod 2, \\ \text{map } x \text{ into hyperbolic table range:} & x^*{=}x\,(2^{-scale}), \\ \text{initialize product, sum, } \log_2(2)\text{:} & \langle u,v\rangle{=}\langle 1,0\rangle,\ measure{=}0,\ radix{=}1, \\ \text{iterate hyperbolic table for } \{w_n,z_n\}\text{:} & \left[\begin{cases} +\Omega_{uv}^{w_n z_n},\ measure{+}{=}2^{radix-n}, & \text{if } u+v < x^* \\ -\Omega_{uv}^{w_n z_n},\ measure{-}{=}2^{radix-n}, & \text{if } u+v > x^* \\ \text{stop iteration}, & \text{if } u+v \equiv x^* \end{cases}\right]_{n=1}^{m} \\ \text{convert (unmap } \log_2(x^*)\text{) to natural:} & \ln(x) \equiv (measure{+}scale)\ln(2) \end{array}\right] \tag{3.1}$$

---

[1] A.D. Levine, The Symplectic Oscillator. https://muzemazer.com/SymplecticOscillator.pdf

[2] `fp_t agm(fp_t a, fp_t g) {int n = mantissa_width(fp_t) >> 1; while (a = div(a + g, 2), --n) g = sqrt(a * g); return a;}`

[3] T.J. Dekker, A Floating-Point Technique for Extending the Available Precision. https://ir.cwi.nl/pub/9159/9159D.pdf