ECHO

Math and Proofs

Version v0.1-draft
Commit: HEAD
November 30, 2025

James Ross & Echo Contributors

Foreword

Echo is a deterministic, multiverse-aware engine. This booklet walks you in with progressive layers: orient yourself, learn the core building blocks, then dive into math and operations. Each shelf can stand alone; together they form the full Echo field guide.

If you are new, start with the onboarding roadmap and glossary. If you build or extend Echo, keep the determinism contract and scheduler flow in view. Future work will deepen each part and add more diagrams as Echo evolves.

Contents

Part I

Math and Proofs

0.1 Deterministic Mathematics

The cornerstone of Echo's reliability is its deterministic math module. Standard IEEE 754 floating-point arithmetic is famously non-deterministic across different architectures and compiler optimizations.

0.1.1 The Hazards of Hardware Floats

- **NaN Payloads:** The bit pattern of ‘NaN‘ (Not a Number) is not fully specified. ‘0.0 / 0.0‘ might produce different bits on x86 vs ARM.
- **Signed Zero:** ‘-0.0‘ and ‘+0.0‘ compare equal but hash differently.
- **FMA (Fused Multiply-Add):** Some CPUs combine multiply and add into one step, changing rounding results.
- **Transcendental Functions:** ‘sin()‘, ‘cos()‘, ‘pow()‘ are often implemented in libc or hardware microcode, with varying precision.

0.1.2 Echo's Current Float Wrapper: `F32Scalar`

Echo wraps ‘f32‘ in a strict type, ‘F32Scalar‘, with the policies implemented today:

1. **Canonical Zero:** The constructor maps ‘-0.0‘ to ‘+0.0‘ (see Listing 1).
2. **NaN passthrough (for now):** NaNs are preserved as-is; canonical NaN collapsing is a planned follow-up.
3. **Transcendentals:** ‘sin‘/‘cos‘ currently call ‘f32::sin‘/‘f32::cos‘. Deterministic LUT or polynomial replacements are planned to remove platform variance.

Listing 1: F32Scalar canonicalizes -0.0 and derives ordering

```rust
#[derive(Debug, Copy, Clone)]
pub struct F32Scalar {
    value: f32, // invariant: never stores -0.0
}

impl F32Scalar {
    pub const fn new(num: f32) -> Self {
        Self { value: num + 0.0 } // -0.0 -> +0.0
    }
}

impl Ord for F32Scalar {
    fn cmp(&self, other: &Self) -> Ordering {
        self.value.total_cmp(&other.value)
    }
}
```

0.1.3 Geometric Types

Built on top of ‘F32Scalar‘ are the standard geometric types:

- `Vec2`, `Vec3`, `Vec4`
- `Mat3`, `Mat4` (Column-major)
- `Quat` (Quaternions for rotation)

These types ensure that vector operations (dot product, cross product, matrix multiplication) propagate determinism.

0.1.4 Fixed Point Option

For environments where floating-point hardware is completely unreliable or missing, Echo supports a ‘DFix64‘ (Deterministic Fixed Point) backend. This uses ‘i64‘ to represent decimal numbers (e.g., 32 bits for integer, 32 bits for fraction).