# NuFast: Efficient Three-Flavor Neutrino Oscillation Probabilities in Rust

Baalateja Kataru

*Planckeon Labs*
baalateja@planckeon.org

February 3, 2026

**Abstract**

We present `nufast`, a Rust implementation of the NuFast algorithm for computing three-flavor neutrino oscillation probabilities in vacuum and constant-density matter. Our implementation achieves performance competitive with optimized C++ code: $\sim 61$ ns for vacuum and $\sim 95$ ns for matter calculations per energy point —approximately **27% faster** than C++ for matter effects. This performance advantage stems from LLVM's aggressive optimization of Rust's ownership-based memory model during Newton-Raphson iterations. The crate is published on crates.io and includes WebAssembly bindings enabling browser-based applications. We also provide `VacuumBatch`, an optimized API for batch calculations that pre-computes mixing matrix elements, achieving 45% speedup for energy spectrum computations. This work enables high-performance neutrino physics calculations in modern software ecosystems while maintaining memory safety guarantees.

***Keywords:*** *neutrino oscillations, matter effects, MSW effect, NuFast algorithm, Rust, WebAssembly, high-performance computing*

## 1. Introduction

Neutrino oscillation is a quantum mechanical phenomenon where neutrinos change flavor as they propagate through space. The discovery of neutrino oscillations—implying nonzero neutrino masses—represents physics beyond the Standard Model and was recognized with the 2015 Nobel Prize in Physics. Accurate and efficient computation of oscillation probabilities is essential for analyzing data from current and next-generation experiments including DUNE, Hyper-Kamiokande, and JUNO.

The transition probability from flavor $\alpha$ to flavor $\beta$ after propagating distance $L$ with energy $E$ is given by:

$$P_{\alpha\beta} = |\langle \nu_\beta \mid \nu_\alpha(L) \rangle|^2 = \sum_{i,j} U^*_{\alpha i} U_{\beta i} U_{\alpha j} U^*_{\beta j} e^{-i\Delta m^2_{ij} L/2E} \tag{1}$$

where $U$ is the Pontecorvo-Maki-Nakagawa-Sakata (PMNS) mixing matrix and $\Delta m^2_{ij} = m^2_i - m^2_j$ are mass-squared differences.

In matter, the Mikheyev-Smirnov-Wolfenstein (MSW) effect modifies the effective mixing parameters through coherent forward scattering of electron neutrinos on electrons. Computing these modified probabilities efficiently has been a longstanding challenge in neutrino phenomenology.

### 1.1. The NuFast Algorithm

The NuFast algorithm, developed by Denton and Parke [1], provides a computationally optimal method for three-flavor oscillation probabilities. Key innovations include:

1. **Eigenvalue-Eigenvector Identity (EEI)**: Avoids cubic equation solving by using only 2×2 matrix diagonalization (quadratic equations)
2. **Square root elimination**: The quadratic's discriminant square root cancels in probability expressions
3. **Optimal eigenvalue ordering**: Initial DMP approximation propagates to other eigenvalues efficiently
4. **Newton-Raphson refinement**: Optional iterations for arbitrary precision

> *Remark.* NuFast has been adopted by major collaborations: it is implemented in MaCH3 (the primary reweighting framework for T2K and other US/Japan experiments) and JUNO analysis pipelines, achieving "dramatic speed ups—close to an order of magnitude—over other 'optimized' algorithms" [1].

## 1.2. Motivation and Historical Context

This implementation represents the culmination of several years of work on neutrino oscillation phenomenology.

### 1.2.1. Undergraduate Research (2022–2023)

The author's undergraduate capstone thesis at Krea University, supervised by Dr. Sushant Raut, explored the *Interplay between Neutrino Oscillations and Linear Algebra*. The research investigated applications of the Eigenvalue-Eigenvector Identity (also called the Rosetta identity) [2] and the Adjugate Identity [3] to streamline symbolic calculations of oscillation probabilities.

The goal was to derive novel series expansions of oscillation probabilities in matter up to second order in the mass hierarchy parameter $\alpha \equiv \Delta m^2_{21}/\Delta m^2_{31}$ only—as opposed to second order in both $\alpha$ and $\sin\theta_{13}$ as in Akhmedov et al. [4]. Using Mathematica and the Cayley-Hamilton formalism, the author explored whether these linear algebra identities could simplify the analytic calculation. While the symbolic expansions proved computationally intractable (repeatedly exhausting available memory), numerical implementations were successful, resulting in `pytrino`—a Python/Cython library published on PyPI.

### 1.2.2. Postgraduate Research (2023–2024)

The author continued this work during a postgraduate program, pivoting to investigate neutrino oscillations on quantum computers using Hamiltonian simulation (Trotter-Suzuki decomposition) and quantum machine learning approaches. This work reproduced published results [5], [6] using IBM's Qiskit framework.

### 1.2.3. Correspondence with Dr. Denton (October 2024)

In October 2024, the author consulted Dr. Peter Denton regarding research directions in neutrino physics, describing prior work on the EEI and challenges with 3+1 sterile neutrino extensions. Dr. Denton explained that while the EEI is powerful for three flavors (2×2 diagonalization), four-flavor oscillations require cubic eigenvector equations—"analytically much much worse, and also numerically somewhat unstable."

Dr. Denton recommended the NuFast algorithm, noting approximately 100 ns per probability calculation on his laptop. This benchmark became a target for our Rust implementation.

# 2. Algorithm Details

## 2.1. Vacuum Oscillations

In vacuum, the oscillation probability depends on:

- **Mixing angles**: $\theta_{12}, \theta_{13}, \theta_{23}$
- **CP-violating phase**: $\delta_{\text{CP}}$
- **Mass-squared differences**: $\Delta m^2_{21}, \Delta m^2_{31}$
- **Baseline and energy**: $L$ (km), $E$ (GeV)

**Definition (PMNS Matrix).** The PMNS matrix in the standard parameterization is:

$$U = \begin{pmatrix} c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta} & c_{23}c_{13} \end{pmatrix} \tag{2}$$

where $c_{ij} = \cos\theta_{ij}$ and $s_{ij} = \sin\theta_{ij}$.

The vacuum algorithm computes exact probabilities using trigonometric identities without matrix exponentiation or diagonalization.

### 2.2. Matter Effects

For propagation through matter with constant electron density $N_e$, the effective Hamiltonian acquires a matter potential:

$$H = H_{\text{vacuum}} + \text{diag}(a, 0, 0), \quad a = \sqrt{2}G_F N_e = 7.63 \times 10^{-5}(\rho Y_e)[\text{eV}^2 \ / \ \text{GeV}] \tag{3}$$

where $\rho$ is the matter density in g/cm³ and $Y_e$ is the electron fraction.

NuFast uses:
1. **DMP approximation**: Initial eigenvalue estimate from Denton-Minakata-Parke [7]
2. **Newton-Raphson refinement**: $N_{\text{Newton}}$ iterations for improved precision

> *Remark.* For long-baseline experiments like DUNE (1300 km, 2.5 GeV, $\rho \approx 2.8$ g/cm³), $N_{\text{Newton}} = 0$ provides sub-percent accuracy. Higher precision is available with $N_{\text{Newton}} = 1$ or 2.

## 3. Implementation

### 3.1. Core API

Our Rust implementation provides an ergonomic API:

```rust
use nufast::{VacuumParameters, MatterParameters};
use nufast::{probability_vacuum_lbl, probability_matter_lbl};

// Vacuum oscillation with NuFIT 5.2 parameters
let params = VacuumParameters::nufit52_no(1300.0, 2.5);
let probs = probability_vacuum_lbl(&params);
println!("P(νμ → νe) = {:.4}", probs.Pme);

// Matter oscillation
let params = MatterParameters::nufit52_no(
    1300.0,  // L (km)
    2.5,     // E (GeV)
    2.8,     // ρ (g/cm³)
    0.5,     // Ye
    0        // N_Newton
);
let probs = probability_matter_lbl(&params);
```

### 3.2. VacuumBatch Optimization

For batch calculations (e.g., computing energy spectra), we provide `VacuumBatch` which pre-computes mixing matrix elements:

```rust
use nufast::VacuumBatch;

let batch = VacuumBatch::nufit52_no();
let spectrum = batch.spectrum(1300.0, 0.5, 5.0, 1000);
// 1000-point spectrum in ~72 µs
```

**Key Result.** `VacuumBatch` achieves **45% speedup** over repeated single-point calls by pre-computing all nine $|U_{\alpha i}|^2$ elements and the Jarlskog invariant once, then reusing them across all energy/baseline points.

### 3.3. WebAssembly Support

The `nufast-wasm` crate compiles the physics engine to WebAssembly:

```javascript
import init, { wasmCalculateEnergySpectrum } from 'nufast-wasm';

await init();
const spectrum = wasmCalculateEnergySpectrum({
  theta12_deg: 33.44,
  theta13_deg: 8.57,
  // ... other parameters
}, 1300, 0.5, 5.0, 200);
```

The compiled WASM module is approximately **32 KB gzipped**, enabling browser-based neutrino physics with near-native performance.

## 4. Benchmark Methodology

All benchmarks were performed on AMD Ryzen (WSL2/Windows). Methodology:

- **Iterations**: 10 million ($10^7$) calculations per measurement
- **Energy range**: 0.5–5.0 GeV (DUNE-like parameters)
- **Parameters**: NuFIT 5.2 best-fit values [8]
- **Anti-optimization**: Sink variables prevent dead code elimination

Rust benchmarks use Criterion with statistical analysis. C++, Fortran, and Python use high-resolution timing with standard deviation over 10 runs.

# 5. Results

| Language | Vacuum | N=0 | N=1 | N=2 | N=3 |
|---|---|---|---|---|---|
| **Rust** | 61 ns | **95 ns** | 106 ns | 113 ns | 117 ns |
| C++ | **49 ns** | 130 ns | 143 ns | 154 ns | 164 ns |
| Fortran | 51 ns | 107 ns | 123 ns | 146 ns | 167 ns |
| Python | 14,700 ns | 21,900 ns | 21,200 ns | 18,500 ns | 16,300 ns |

Table 1: Single-point oscillation probability timing (ns/call). N = Newton-Raphson iterations. Bold indicates fastest per column.

| Comparison | Vacuum | Matter (N=0) | Interpretation |
|---|---|---|---|
| Rust vs C++ | +24% | **−27%** | Rust faster for matter |
| Rust vs Fortran | +20% | −11% | Rust competitive |
| Rust vs Python | ×241 | ×230 | Compiled advantage |

Table 2: Relative performance (negative = Rust faster).

## 5.1. Key Findings

### 5.1.1. Rust Outperforms C++ for Matter Calculations

**Key Result.** Rust achieves **27% speedup** over C++ for matter oscillations with $N_{\text{Newton}} = 0$. This exceeds Dr. Denton's quoted  100 ns benchmark.

This advantage likely stems from:
1. **Ownership-enabled optimization**: Rust's strict aliasing rules allow LLVM to optimize more aggressively
2. **Loop vectorization**: Newton iteration inner loops optimize well in LLVM
3. **Zero-cost abstractions**: Rust idioms compile to efficient machine code

### 5.1.2. Vacuum Performance

For vacuum (simpler computation), C++ and Fortran are  20% faster. This is expected for compute kernels where Fortran excels.

### 5.1.3. Throughput

| Language | Vacuum (M/s) | Matter (M/s) |
|---|---|---|
| Rust | 17.5 | **10.5** |
| C++ | **20.3** | 7.7 |
| Fortran | 19.7 | 9.4 |
| Python | 0.07 | 0.05 |

Table 3: Throughput in millions of calculations per second.

# 6. Applications

## 6.1. Interactive Visualization: Imagining the Neutrino

The `nufast-wasm` module powers *Imagining the Neutrino*, an interactive web-based educational tool:

https://planckeon.github.io/itn/

Features include:

- Real-time oscillation probability animation
- PMNS matrix visualization (2D and 3D representations)
- Energy spectrum and baseline scan plots
- CP asymmetry visualization: $A_{\mathrm{CP}} = P(\nu_\mu \to \nu_e) - P(\overline{\nu}_\mu \to \overline{\nu}_e)$
- PREM Earth density model with automatic density calculation
- 11 experimental presets (DUNE, T2K, NOvA, etc.)
- Internationalization (7 languages)

With WASM, the visualization computes 400-point energy spectra in real-time ( 72 µs with VacuumBatch) as users adjust parameters.

## 6.2. Future Directions

- **PyO3 bindings**: Python interface via Rust FFI
- **SIMD optimization**: Explicit vectorization for batch computations
- **Variable density**: Piecewise-constant matter profiles for Earth models

# 7. Conclusion

We have demonstrated that Rust provides an excellent platform for computational neutrino physics:

| | |
|---|---|
| **Performance** | 27% faster than C++ for matter effects |
| **Throughput** | 10.5 M calculations/s (matter, N=0) |
| **Memory safety** | Compile-time guarantees, zero runtime overhead |
| **WASM support** | 32 KB gzipped for browser deployment |
| **Batch API** | VacuumBatch for 45% faster spectra |
| **Distribution** | Published on crates.io |

Table 4: Summary of nufast capabilities.

The combination of performance, safety, and modern tooling makes Rust an attractive choice for future neutrino physics software development.

# Acknowledgments

The author thanks Dr. Peter B. Denton (Brookhaven National Laboratory) for recommending the NuFast algorithm and providing guidance on neutrino oscillation calculation approaches. This work builds on the NuFast algorithm developed by Denton and Parke, with original implementations available at https://github.com/PeterDenton/NuFast.

The author acknowledges the neutrino physics education received during undergraduate research at Krea University under Dr. Sushant Raut, which provided the theoretical foundation for this implementation.

# Code Availability

All code is open source under the MIT license:

| | |
|---|---|
| **Crates.io** | https://crates.io/crates/nufast |
| **GitHub** | https://github.com/planckeon/nufast |
| **Docs** | https://docs.rs/nufast |
| **Visualization** | https://planckeon.github.io/itn/ |

Benchmark implementations (Rust, C++, Fortran, Python) are included in `benchmarks/`.

# References

## Bibliography

[1] P. B. Denton, "Neutrino Oscillation Probabilities: A Compact Multi-algorithm Approach," 2024.

[2] P. B. Denton, S. J. Parke, and X. Zhang, "Eigenvector-based approach to neutrino oscillation probabilities in matter," *Physical Review D*, vol. 101, p. 93001, 2020.

[3] A. Abdullahi and S. J. Parke, "Eigenvalue-independent eigenvectors and the adjugate of a matrix," 2022.

[4] E. K. Akhmedov, R. Johansson, M. Lindner, T. Ohlsson, and T. Schwetz, "Series expansions for three-flavor neutrino oscillation probabilities in matter," *JHEP*, vol. 4, p. 78, 2004.

[5] C. A. Argüelles and B. J. P. Jones, "Neutrino Oscillations in a Quantum Computer," *Physical Review D*, vol. 99, p. 96005, 2019.

[6] J. Turro and others, "A Quantum simulation of neutrino-matter collective oscillations," 2021.

[7] P. B. Denton, H. Minakata, and S. J. Parke, "Compact Perturbative Expressions for Neutrino Oscillations in Matter." 2016.

[8] I. Esteban, M. C. González-García, M. Maltoni, T. Schwetz, and A. Zhou, "The fate of hints: updated global analysis of three-flavor neutrino oscillations." 2020.