Calculating neutrino oscillations probabilities on GPU

2nd Computing Challenges Workshop

J. Dalseno
jeremypeter.dalseno {at} usc.es

02 October 2024

Introduction

Calculation speed of neutrino oscillations probability a bottleneck

Ultimately limits sensitivity to fundamental physics parameters

- eg. CP violation in atmospheric neutrino oscillations

For experimental observables x and model parameters θ

Binned-χ^2 seems to be the approach for now

$$\chi^2(\theta) = \sum_{i=1}^{N_{\text{bins}}} \left(\frac{N_i - N(x|\theta)}{\sqrt{N_i}} \right)^2$$

- Bin data, sum over bins, fast

Unbinned maximum-likelihood (ML) approach far more sensitive

$$-2 \log \mathcal{L}(\theta) = -2 \sum_{i=1}^{N_{\text{events}}} \log(\mathcal{P}(x_i|\theta))$$

- Sum over events, slow

Goal 1: Improve oscillations calculation speed to increase sensitivity

Neutrino oscillations

PMNS neutrino mixing matrix

$$U_{\text{PMNS}} = \begin{pmatrix} +c_{12}c_{13} & +s_{12}c_{13} & +s_{13}e^{-i\delta_{CP}} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{+i\delta_{CP}} & +c_{12}c_{23} - s_{12}s_{23}s_{13}e^{+i\delta_{CP}} & +s_{23}c_{13} \\ +s_{12}s_{23} - c_{12}c_{23}s_{13}e^{+i\delta_{CP}} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{+i\delta_{CP}} & +c_{23}c_{13} \end{pmatrix}$$

Hamiltonian in matter with constant density ρ

$$\hat{H} = U_{\text{PMNS}} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Delta m_{21}^2 & 0 \\ 0 & 0 & \Delta m_{31}^2 \end{pmatrix} U_{\text{PMNS}}^{\dagger}/(2E) \pm \begin{pmatrix} \sqrt{2}G_F N_A \rho Y_e & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Solve time-dependent Schrödinger equation for oscillation probabilities

$$i\frac{d\Psi}{dt} = \hat{H}\Psi$$

Finally, propagate through Earth with several layers of different density

$$\mathcal{P}(\nu_a \to \nu_b) \equiv \prod_{i=1}^{N_{\text{layers}}} \mathcal{P}_i(\nu_a \to \nu_b)$$

Computation time is adding up

Neutrino oscillations calculation

Two approaches to solving the time-dependent Schrödinger equation

$$i\frac{d\Psi}{dt} = \hat{H}\Psi$$

Analytical

Diagonalise the Hamiltonian

Obtain eigenvalues, eigenvectors

Exact

Fast

Difficult to increase N_ν

Numerical

Use the matrix exponential

$\Psi \equiv A(\nu_a \to \nu_b) = e^{-i\hat{H}t} = e^{-iL\hat{H}}$

Padé approximation

Slow

Generalisation to arbitrary N_ν trivial

For broader physics studies, the numerical solution would be ideal

Goal 2: Obtain an exact solution with the matrix exponential

Accelerating neutrino oscillations calculation

For numerical solutions to be viable, compete with analytical solutions

State-of-the-art: `NuFast`

- arXiv:2405.02400 [hep-ph] (2024)

Benchmark: ~ 100 ns per mixing probability calculation (Laptop Intel i7)

Implement matrix exponential in `Eigen` (since 2006)

- C++ template header-only library, no dependencies, easy to include
- Fastest free solution on the market, many algorithms vectorised
- Backend for several industry-grade software packages *eg*. `TensorFlow`
- Intuitive interface, resembles expressions on paper, easy to use

```
Eigen::Matrix3cd
AmpMatter( const Eigen::Matrix3cd& Hmatter, const double& L )
{ return (std::complex<double>(0.0, -L*invhbarc)*Hmatter).exp(); }
```

11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz

- Laptop-grade CPU from 2021

~ 700 ns per mixing probability calculation

- Almost 10 times slower than `NuFast`

Accelerating neutrino oscillations calculation

Every event is independent

Execute calculation in parallel on CPU

AMD Ryzen Threadripper 3990X 64-Core Processor

- 64 cores, 128 threads

> 20 times faster at $\sim 30\ \mathrm{ns}$ per mixing probability calculation

- Numerical solution now 3 times faster than `NuFast`

There are some issues here

1. Did you just pay 10 000 EUR to beat a laptop by a factor of 3 ???
2. If you can multithread, so can they. The factor 10 penalty still applies.

Obviously, there is no silver bullet on the CPU

Accelerating neutrino oscillations calculation

Port to GPU (`CUDA C++`)

`Eigen` already compatible with `CUDA` but with heavy restrictions

- Maximum of 4 neutrino generations possible
- No native matrix exponential

`Eigen` CPU matrix exponential manually copied over to `CUDA` code

`NuFast` (CPU)	`CUDA` (RTX4090)
$\sim$100 ns	< 5 ns

Numerical solutions on GPU competitive with analytical approach

As an aside, more sophisticated media also becomes possible

PMNS matrix

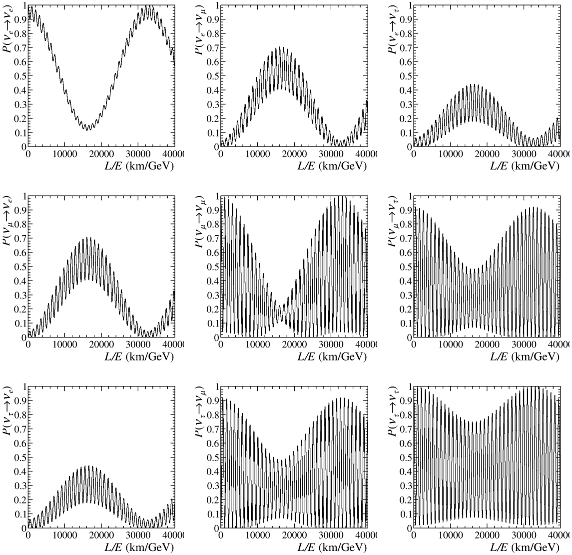
Mass heirarchy

(That 1 subscriber is not me)

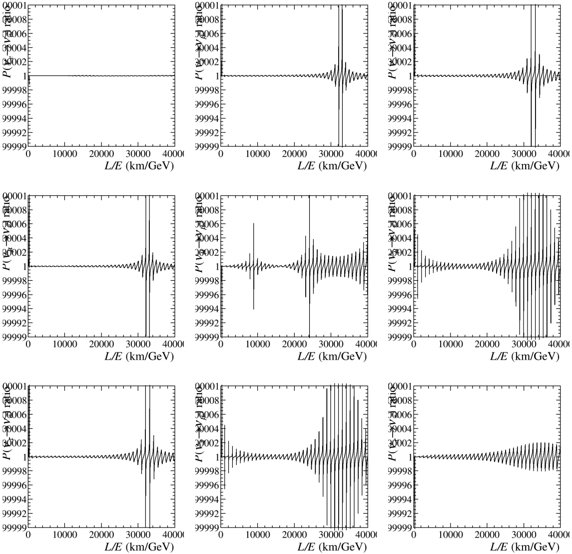
Plan survival probability videos in $\cos\theta_z$ vs E_ν eventually

Neutrino mixing probabilities

NuFast and Eigen agree well on the 9 mixing probabilities



Ratio of neutrino mixing probabilities



Small difference in implementation of the matrix exponential

ν_e: Excellent, ν_μ: Good to within 10^{-6}, ν_τ: Good to within 10^{-5}

Exact numerical approach

- Software implementations of matrix exponential favour generality
  - Leads to approximate solutions, eg. Padé in `Eigen` and `SciPy`
- However, generality from the neutrino perspective looks quite different
  - The Hamiltonian is diagonalisable
  - The dimensionality is small at worst
- For these cases, exact and practical numerical solutions are feasible
  - eg. Eigenvalue decomposition
- Express Hamiltonian as $\hat{H} = V \Lambda V^{-1}$
- V the matrix of eigenvectors, Λ the diagonal matrix of eigenvalues
- Then $e^{\hat{H}} = V e^{\Lambda} V^{-1}$
  - Looks suspiciously like solving the Schrödinger equation
- No explicit software implementation as of yet
  - In `Eigen` on CPU, this amounts to massive code bloat from 1 to 4 lines
- Simplicity cannot be replicated on GPU at this time
  - Code to generate eigenvalues and eigenvectors needs to be written

Summary

Decreasing neutrino oscillations calculation time is investigated

Analytical	Numerical
Exact	Padé approximation
Fast	Fast
Difficult to increase N_ν	Generalisation to arbitrary N_ν trivial
Threadripper 3990X: < 5 ns	RTX4090: < 5 ns

Replacing Padé approximation with Exact solution on GPU underway

GPU performance figures could go either way, but probably not by much

All things being equal, numerical solutions always slower

- Nothing prevents analytical solutions from being ported to GPU

Numerical solutions gain their advantage from versatility

- Trivial to add sterile neutrino

For general purpose software, inclusion of numerical approach justified

Intend to provide header to link/package with `PyNu`