GPU 上的中微子探测

Berrysoft(王宇逸)

清华大学工程物理系

2022 年 3 月 26 日

何为中微子

在一个典型的 β 衰变中，原子核放出一个电子。这个电子的能量服从右图的分布。
由于能量动量守恒，泡利假设存在另一种粒子分走了部分能量动量。费米将其命名为**中微子**（neutrino）。

$$n \to p + e^- + \bar{\nu}_e$$

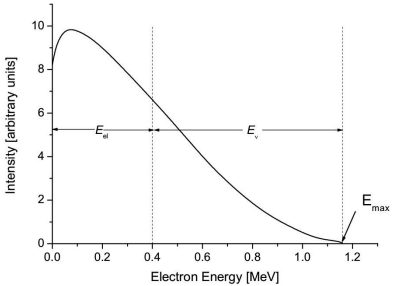


图: 图片来源：维基百科

江门中微子实验（JUNO）

图: JUNO 的地理位置

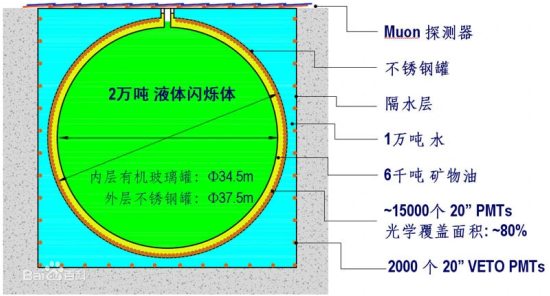


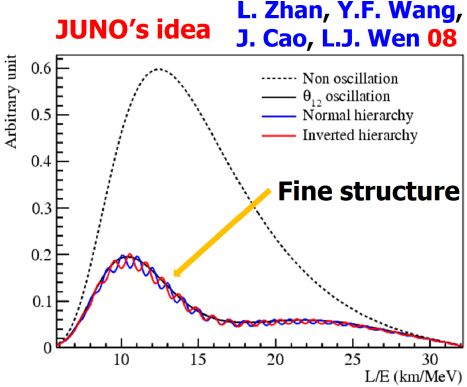
图: JUNO 的探测器结构

JUNO 的物理目标

中微子振荡的发现确认了中微子存在质量，这是一个超出现有标准模型的结论。确定三个质量本征态的质量顺序，能够帮助我们排除一些物理模型。

如今我们能够确定 $m_2 > m_1, \Delta m^2_{31} \gg \Delta m^2_{21}$。因此中微子质量仅存在两种排序可能：$m_3 > m_2 > m_1$（正序）或 $m_2 > m_1 > m_3$（反序）。

电子反中微子 $\bar{\nu}_e$ 的能谱根据中微子质量顺序的不同而有着微弱的差异，如右图。



逆 β 衰变

液体闪烁体中含有大量的质子，中微子主要和质子反应：

$$\bar{\nu}_e + p \to n + e^+$$

产生的正电子会与液闪分子相互作用，产生大量光子。正电子会与电子湮灭：

$$e^+ + e^- \to \gamma + \gamma$$

放出的 γ 主要与电子发生康普顿散射，被散射的电子也会与液闪分子相互作用产生光子。

光电倍增管

光电倍增管（PMT）是一个信号放大器。它利用光电效应，将光子信号转变为单电子信号。再将一个电子变成许多电子，从而形成可以观测的波形。

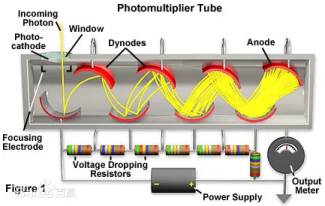
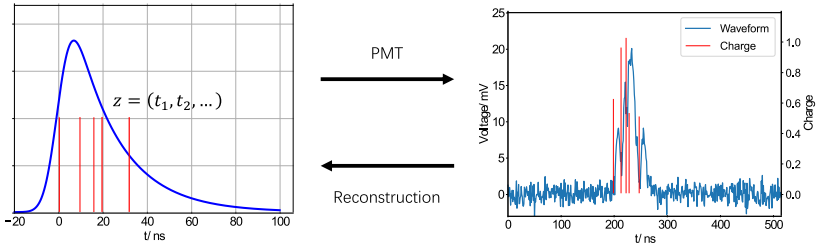


图: PMT 的放大原理示意图

光电子的泊松过程

光电子击中 PMT 的过程是一个泊松过程。下图是发光曲线与波形。



Sample from Poisson process

Convolution with single PE waveform

重建算法需要光强 μ 与发光曲线的时间偏移 t_0。

Fast Stochastic Matching Pursuit (FSMP)

我们使用马尔科夫链蒙特卡罗（MCMC）采样这一过程：

波形 光强 光电子序列 时间偏移

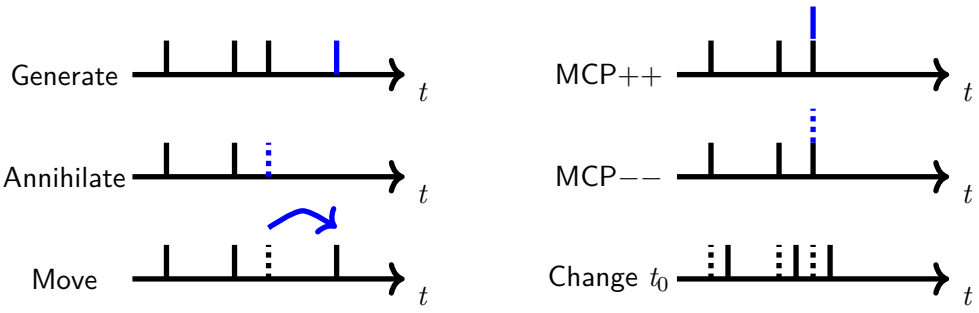
$$p(w|\mu) = \sum_{z,t_0} p(w|z,t_0)\,q(z,t_0)\,\frac{p(z,t_0|\mu)}{q(z,t_0)} \approx C\frac{1}{M}\sum_{i=1}^{M}\frac{p(z_i|t_{0i},\mu)}{p(z_i|t_{0i},\mu_0)} \tag{1}$$

MCMC

并估计 t_0 与 μ：

$$\hat{t}_0 = \frac{1}{M}\sum_{i=1}^{M} t_{0i}$$
$$\hat{\mu} = \arg\min_{\mu}\ p(w|\mu) \tag{2}$$

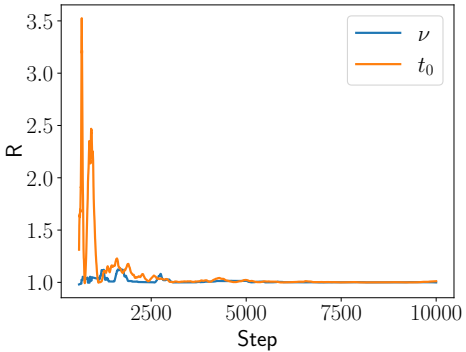
MCMC 的步骤



这些操作需要满足一定的接受率才会被接受，例如 $\min\left\{1, \frac{p(t_0'|\boldsymbol{z}_i,\mu)g(t_0' \to t_0)}{p(t_0|\boldsymbol{z}_i,\mu)g(t_0 \to t_0')}\right\}$

收敛性

使用 Gelman 和 Rubin 的方法 (1992) 可以计算混合 MCMC 的收敛性。
通常，这个链会在 $3000 \sim 5000$ 步左右收敛。

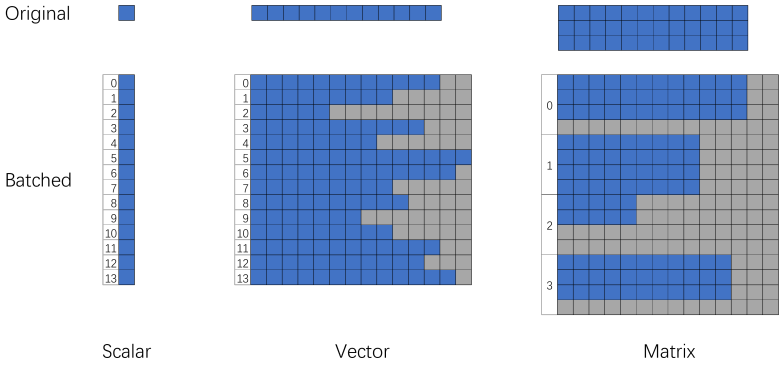


手写 CUDA（失败）

尝试令一个 block 处理一个波形，由于 CUDA 的同步机制，恰好可以实现对波形的并行处理。
我们写了 1350 行 CUDA 代码（包括单元测试），但是很不幸，由于对同步机制的理解并不深刻，代码至今没有跑通。
但是这些代码**非常壮观**，因此在这里给一点样例。

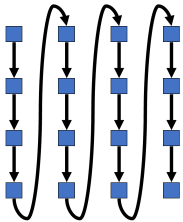
```
__global__ void flow(
    const std::size_t NW,
    const std::size_t mnw, // max nw
    const std::size_t mnl, // max nl
    const std::uint32_t* pnw, // NW
    const std::uint32_t* pnl, // NW
    float* __restrict__ cx, // NW x mnw x mnl
    const float* __restrict__ tlist, // NW x mnl
    float* __restrict__ z, // NW x mnw
    const float* __restrict__ mus,
    const float* __restrict__ sig2s,
    const float* __restrict__ sig2w,
    const float* __restrict__ A, // NW x mnw x mnl
    const float* __restrict__ p_cha, // NW x mnl
    const float* __restrict__ c_cha, // NW x (mnl + 1)
    const float* __restrict__ mu_t, // NW
    std::uint32_t* __restrict__ s0_history, // NW x TRIALS
    float* __restrict__ delta_nu_history, // NW x TRIALS
    float* __restrict__ t0_history, // NW x TRIALS
    // Temp buffers
    float* __restrict__ istar, // NW x TRIALS
    float* __restrict__ home_s, // NW x TRIALS
    float* __restrict__ A_vec, // NW x mnw
    float* __restrict__ c_vec, // NW x mnw
    float* __restrict__ delta_cx, // NW x mnw x mnl
    float* __restrict__ delta_z, // NW x mnw
    float* __restrict__ temp_cx, // NW x mnw x mnl
    float* __restrict__ temp_z, // NW x mnw
    float* __restrict__ wanders, // NW x TRIALS
    float* __restrict__ wts, // NW x TRIALS
    float* __restrict__ accepts, // NW x TRIALS
    float* __restrict__ accts, // NW x TRIALS
    fsmp_step* __restrict__ flip // NW x TRIALS
)
```

批量处理 I


Original
Batched
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Scalar
Vector
Matrix

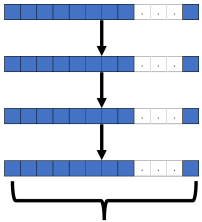
批量处理 II

Original algorithm



: one waveform

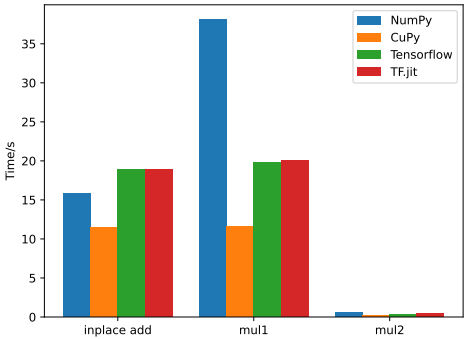
Batched algorithm



5000 waveforms

在 A100 上，可以达到 10ms/波形。

CuPy 与 Tensorflow 的速度对比 I



CuPy 与 Tensorflow 的速度对比 II

CuPy	Tensorflow

`sel_add = cp.ElementwiseKernel(`
`  "float32 delta, bool sel",`
`  "float32 dest",`
`  "if(sel) dest += delta",`
`  "sel_add"`
`)`
`def cp_inplace_add(a, b, c):`
`  sel_add(c, b, a)`
	`def tf_inplace_add(a, b, c):`
	`  a = tf.where(b, a + c, a)`

表: 并不知道 Tensorflow 有没有什么高效的 inplace/masked add 方法

致谢

- 续本达（@heroxbd）老师贡献了大量的代码与优化。
- 徐大成（@thexdc）同学的波形分析论文：arXiv:2112.06913
- 杰哥（@jiegec）的多次讨论与 GPU 计算方面的指导。
- 哈利教教写 CUDA！
- 感谢其他提供点子的同学！
- 部分图片来源杂糅，无法完全表明出处。

资源

- NeutrinoFSMP：讲稿，与 benchmark。
- TurboSMP：失败的 CUDA 尝试。

Q&A