阶段二 实验报告

—— Gamma 在液体闪烁体内的能量沉积

致理-数理 1　刘苏青* 2021013371

致理-物 12　卢一鸣† 2021013384

致理-数理 1　马腾跃‡ 2021013389

2023 / 08 / 24

摘　要

本文首先大致概括了大作业阶段二的整体思路，然后详细阐述了 git 上各文件的用法，最后提出大作业实现过程中遇到的问题，并给出相应的解决方法，便于读者快速了解我们组的项目成果。

关键词：极大似然、能量沉积、相互作用点、液体闪烁体

*liu-sq21@mails.tsinghua.edu.cn

†luym21@mails.tsinghua.edu.cn

‡mty21@mails.tsinghua.edu.cn

目录

1 整体思路

大作业阶段二主要通过 rank.cpp 程序，实现了对一系列相互作用子事件的排序。我们的核心思路是以相互作用子事件的顺序作为参数，考虑每个事件中所有相互作用子事件的全排列，然后根据部分观测量构造似然函数，并找到其最大似然对应的排序作为预测排序。

首先，我们考虑每种排列下沉积能量对应的概率 p_E。由于电子的运动尺度 $\sim 1\text{mm}$，光子的平均自由程 (即相邻相互作用子事件的平均距离)$\sim 10^2\text{mm}$，因此我们在不知道具体的电子运动轨迹时，可以忽略其沉积过程，直接用能量沉积重心来代替相互作用点。选择任意相邻的三个相互作用子事件可以确定散射角度，根据康普顿散射的原理，在已知入射光子能量和散射角度时，就能够计算出沉积能量的理论值 E。根据能量分辨率 σ_E/E 的计算公式可计算出沉积能量的观测标准差 σ_E。假设沉积能量的观测值服从正态分布，则在 $E^* \sim \mathcal{N}(E, \sigma_E)$ 下计算 $p(E^* = E_{dep})$ 即为每个相互作用子事件沉积能量对应的概率，将全部相互作用子事件的概率求积可得该排列下沉积能量对应的概率 p_E。

然后，我们考虑每种排列下时间对应的概率 p_t。选择任意相邻的两个相互作用子事件可以确定相互作用距离，除以光速可以得到时间间隔的理论值，进而可以计算每个相互作用点处的时间的理论值 t(初始时间为 0)。假设时间间隔的观测值服从正态分布，则在 $t^* \sim \mathcal{N}(t, 1.5)$ 下计算 $p(t^* = time)$ 即为每个相互作用子事件的时间对应的概率，将全部相互作用子事件的概率求积可得该排列下时间对应的概率 p_t。

此外，我们考虑每种排列下相互作用距离的概率 p_{inter}。选择任意相邻的两个相互作用子事件可以确定相互作用距离 x，根据相互作用点的定义，在已知入射光子能量时，就能够计算出平均自由程的理论值 λ。假设相互距离的观测值服从指数分布，则在 $x^* \sim \mathcal{E}(\lambda)$ 下计算 $p(x^* = x)$ 即为每个相互作用距离对应的概率，将全部相互作用距离的概率求积可得该排列下时间对应的概率 p_{inter}。

最后，计算三个概率之积 $\mathcal{L} = p_E * p_t * p_{inter}$ 作为似然函数，由于若干个 0 和 1 之间的概率之积非常小，可能会超出计算机的计算精度，因此我们将所有概率取对数后再求和来处理。

2 实现方式

2.1 数据预处理

此 R 程序命名为 data_preprocessing.R，用于实现数据预处理的功能。通过引入 rhdf5 包，该程序可读取 deposit_test.h5 文件，并将第 i 个事件对应的数据拆分出来保存在 data/i.txt 中。文件 i.txt 的第一行为该事件对应的相互作用子事件的个数，第二行往后为第 i 个事件对应的数据 (deposit_test.h5) 文件的第二至七列。

2.2 单个事例计算

此 cpp 程序命名为 rank_90/140/150.cpp，且已经编译成名为 rank_90/140/150 的可执行文件。你可以直接运行该文件，并在输入框中粘贴标准格式的输入内容 (格式见 data_preprocessing.R 处理后的形式)，按下回车计算单个事例最有可能的作用点排列顺序。

- Sigma **类**: 该类包括两个向量 E_gamma_values 和 LS_sigma_values。此类的实例可以从文件加载数据，并提供线性插值搜索的方法，以实现对每个入射光子能量 E_gamma 给出一个液闪分子单位体积总散射截面 LS_sigma 的作用。

```
class Sigma \{
private:
    vector E_gamma_values;
    vector LS_sigma_values;
    ...
\};
```

- **全局** sigma **实例**: Sigma 类的实例，从文件"LS_sigma_data.dat" 加载数据。

```
Sigma sigma("LS_sigma_data.dat");
```

- vec_dot **函数**: 一个函数，它接受两个向量并计算它们的点积。

```
double vec_dot(vec vec1, vec vec2) {...}
```

- calculate_theta **函数**: 一个函数，它接受三个整数作为参数并计算两个向量之间的夹角。

```
double calculate_theta(int pre, int now, int suf)
    {...}
```

- calculate_prob **函数**: 一个函数，根据正态分布公式计算某个值在此分布下的概率。

```
double calculate_prob(double x_std, double sig_std,
    double x_test) {...}
```

- calculate_energy_sigma **函数**: 一个根据题目所给公式计算能量标准偏差的函数。

```
double calculate_energy_sigma(double E_std) {...}
```

- dfs **函数**: 一个函数，利用深度优先搜索来寻找最有可能的物理路径。

  如果直接枚举所有可能的全排列来计算概率密度，复杂度为 $\Theta(n!)$，在 n 较大时不能完成任务

  我们采用搜索的方式枚举所有的全排列，也就是采用“依次确定每个位置上放置的数”的方法。这样我们可以每次试探一个电子沉积点的顺序，计算“若此电子沉积点是处于正确位置时”对应的相互作用概率、光子运动时间概率以及能量沉积概率。而若这三个概率中其中任意一个极小，则可以直接舍弃剩余可能的放置顺序 (剩余的电子沉积点任意排列，都不能弥补这一极小概率)。

  值得注意的是，能量观测器所收集到的能量信息是较为准确的 (标准差较小)。这意味着如果在某一个点，计算得到的能量与观测器收集到的能量存在一定差距，就会得到极小的概率密度。因此，我们上述的**剪枝**操作能够极大地减少枚举不可行排列的时间复杂度。。

  剩余计算相互作用概率、光子运动时间概率以及能量沉积概率的部分是简单的。

```
void dfs(int k, double fval, double tval, double eval)
    {...}
```

2.3 批量计算

为了最大程度利用服务器 CPU 多核性能，我们采用如下方法进行并行计算：即先将 2000 个事件的数据划分成指定的若干组，再通过 create_run_bash.sh 生成同等数量的 bash 脚本，而后通过 parallel.sh 同时运行，从而实现对每个组的数据进行并行计算。

下面逐步介绍实现这一功能的所有脚本的主要组成部分和功能：

- create_run_bash.sh：自动生成指定数目的计算脚本，存储到 script 文件夹中，并给予它们运行权限。
- run{i}.sh：对特定组别的事例输入文件，调用可执行文件 rank 进行计算，把标准输出保存至 output/{i}_ans.txt 文件中。
- parallel.sh：运行所有的 run{i}.sh 脚本。

2.4 输出结果合并

此 R 程序命名为 output_merging.R，用于实现输出结果合并的功能。通过引入 rhdf5 和 data.table 包，该程序可读取第 i 个事件对应的输出结果 data/i_ans.txt，并将所有输出结果汇总整理成题目所需的格式，然后输出为 order_test.h5 文件。

3 小组分工

- **刘苏青**：编写 data_preprocessing.R、output_merging.R 程序和 parallel.sh 脚本，在服务器上运行测试程序以完成最后的衔接工作，整理仓库，并编辑写作实验报告的对应部分。
- **卢一鸣**：提出思路，编写 rank.cpp 程序，整理仓库，并编辑写作实验报告的对应部分。
- **马腾跃**：编写 create_run_bash.sh 脚本、is_empty.sh 脚本和 Makefile 文件，调整各文件接口，在服务器上运行测试程序，整理仓库，并编辑写作实验报告的主要框架及对应部分。

4 成果复现说明

为确保整个流程的完整性、可复现性和一次性，我们使用 GNU make 来组织所有的程序文件。执行特定任务需要直接在命令行中输入对应的命令，比如 make all 来执行所有任务。根据所使用的计算平台的能力，可以在 Makefile 开头设置计算脚本的数量 script_count 和每个脚本计算的事例数 data_per_script。(数值设置参考：CPU：11th Gen Intel(R) core(TM) i7-11800H；内存：16GB 。script_count 设置为十，基本刚好达到 CPU 最大功耗并维持系统正常运转。若要使用服务器来加速计算，则可以考虑将 Makefile 中的 data_per_script 设为 1、script_count 设为 2000)

Makefile 各指令具体调用方法如下：

1. make all **或** make：执行所有任务，包括运行所有脚本、生成所有结果。
2. make data：运行 data_processing.R 脚本处理数据 deposit_test.h5，生成 2000 个事例所对应的.txt 输入文件。
3. make script：运行 create_run_bash.sh 脚本，生成指定数量的 bash 脚本，用于分段排序处理两千个事例。
4. make output：运行 parallel.sh 脚本，并行运行所有的计算 bash 脚本。
5. make output_merging.R：运行 output_merging.R 脚本剪贴并处理所有输出的 txt 文件，生成 order_test.h5 文件。
6. make clean：执行清理任务，删除所有的输出文件。

此外，为了保证对 linux 平台的适用性，我们把程序的编译过程写入了 Makefile 中，如果需要单独编译 rank.cpp，可以单独使用 make rank_90/140/150 命令。

5 问题及解决方案

5.1 计算的主程序运行精度问题

在 rank.cpp 文件编写过程中，我们设置了 exit_limit 这一变量，以决定某三个连续的相互作用子事件的排列的概率是否过于小以至于可以忽略这种全排列。将 exit_limit 设置为较大的值，可以放宽这一剪枝操作的范围，加速程序运行，但会忽略一些情况，造成精度缺失。反之，则会提高结果的精度，但会极大地放慢程序运行的速度。因此，我们根据多次运行的经验对个别事件设置了稍高的 exit_limit，以缩短程序的运行时间。

5.2 个别事件运行不出结果的问题

在运行程序的过程中，我们发现个别事件 (event_id=671、741、1472、1647 和 1984) 在我们设置的参数范围内得不到任何结果 (此时运行时长已经较长，不宜再修改参数范围了)。因此，我们选择跳过这五个事件，并根据 deposit_test.h5 中的 time 列的数据对该事件的所有相互作用子事件进行粗略排序，然后和其他大多数事件一并赋值到 order_test.h5 文件中。

5.3 相互作用距离对应的概率 p_{inter}

在运行程序的过程中，我们发现考虑相互作用距离对应的概率 p_{inter} 并未对准确率带来较为明显的提升，反而会降低程序的运行速度。因此，我们在最终版的程序里移除了这部分概率带来的影响。