



浙江大学

本科实验报告

课题题目： μ 子测量实验

姓名学号： 陈书扬 3210106335

陈科 3210104844

指导老师： 陈星

2022 年 12 月 31 日

	目录
1 背景知识	4
1.1 背景相关	4
1.2 课题概述	4
2 原理	4
2.1 基本原理	4
2.2 实验原理	4
2.3 公式推导	4
3 仪器系统	4
3.1 实验器件	4
3.2 实验仪器	5
3.3 实验软件架构	6
3.3.1 示波器的软件架构	6
3.3.2 主要算法的软件架构	7
3.3.3 Sensor CASSY-2 的软件架构	7
3.3.4 图形界面相关软件结构	7
4 实验	9
4.1 实验方法	9
4.1.1 原本的凸包算法	9
4.1.2 修改后凸包算法	10
4.1.3 维护区间最小值索引的方法	13
4.1.4 低采样率方法与峰值修正策略	14
4.2 实验步骤	14
4.2.1 采集步骤	14
4.3 实验数据	16
4.4 实验结果	17
4.4.1 测试结果	17
4.4.2 实际采样结果	17
5 分析与讨论	18
5.1 误差来源	18
5.2 误差分析	19
5.2.1 寻峰误差	19
5.2.2 峰值修正误差	19
5.2.3 最终测量结果误差分析	19
5.3 实验分析	19
5.3.1 关于 Sensor-CASSY 2 的可用性	19
6 实验总结	20

7 课题总结	20
7.1 仪器系统照片	20
7.2 课题所用器材清单	21
7.3 花絮和留言	21

1 背景知识

1.1 背景相关

μ 子是一种轻子, 由宇宙中的介子衰变产生, 是宇宙射线到达地面时的一种主要成分。 μ 子会很快衰变成电子、反电子中微子和 μ 子中微子, 寿命很短, 仅有 $2.2\mu\text{s}$ 。由于相对论效应, μ 子的寿命会被延长, 从而有几率到达地球表面, 被捕捉到。在应用上, μ 子质量较大, 不像电子一样只能在直线加速器中加速, 能够在圆环加速器中加速, 在对撞实验中具有重要的意义; μ 子可以进行透射和散射成像, 考古上曾经通过 μ 子成像探测胡夫金字塔的内部。此外, μ 子也可以被用于催化核聚变反应, 对于低温核聚变具有一定的研究价值。[1]

1.2 课题概述

本课题分为实际采集和数值模拟两个部分, 实际采集为直接在实验室中捕获 μ 子并将 μ 子捕获事件转换为数值信号, 通过数据分析测算 μ 子的寿命和能量分布等; 数值模拟为通过随机方法模拟 μ 子进入采集区域的频率, 并对捕获事件进行处理, 从而得到 μ 子寿命和能量分布的模拟结果。本报告将主要描述其中的实际采集部分。

2 原理

2.1 基本原理

使用一块塑料晶体采集 μ 子(具体原理见下), 塑料晶体的一侧置光电倍增管, 光电倍增管接示波器。若有 μ 子进入塑料晶体, 无论是残留在内还是穿透, 都会产生一个脉冲信号; 如果残留在内并发生衰变, 则又会产生一个脉冲信号(具体原理见下)。这样, 如果发现一对发生时间相近的脉冲, 其相距时间在可接受的 μ 子寿命范围之内(实验中取 $1\sim10\mu\text{s}$), 那么便可以判断这一对脉冲属于同一个 μ 子。示波器收集到脉冲信号, 并传输给电脑; 电脑上检索并匹配脉冲信号, 如果判断为 μ 子, 那么计算两者的时间差作为 μ 子寿命, 通过脉冲峰的高度判断出能量值的相对大小。如此, 即可得到 μ 子的寿命分布与能量分布。

2.2 实验原理

实验装置之间的连接关系如图示。塑料晶体的尺寸为 $40 \times 20 \times 10(\text{cm}^3)$, 内部为闪烁体材料。高能粒子(如 μ 子)进入闪烁体时, 动能发生衰减, 减少的动能传递给闪烁体, 使得闪烁体中的电子受到激发。之后电子退激, 能量以光的形式散发出来, 然后被光电倍增管捕捉。

光电倍增管是借用光电效应将微弱光信号转化成可观测电信号的仪器。电子退激所发的光照射在光电倍增管的光阴极上, 产生光电效应, 发射出光电子。在光阴极和光阳极之间有多个打拿极, 电子通过打拿极时不断地被倍增, 最后达到可观测的程度。

之后, 这些电信号被传输到示波器上, 示波器再将数据转移到电脑上, 从而获得能够进行处理的时序信号。在这之后存在的问题为:

- (1) 如何从时序信号中找到峰?
- (2) 如何从峰中获得较为准确的 μ 子衰变时间和能量分布?

这部分为我们所做的主要工作。由于篇幅较长, 该部分在“实验方法”中作描述 [4.1], 在此不作赘述。

2.3 公式推导

3 仪器系统

3.1 实验器件

- (1) 塑料晶体。在 [2.2] 中已经叙述。

(2) μ 子。 μ 子在实验室中自然采集, 理论上的通量为 $1.2 \text{ cm}^{-2} \cdot \text{min}^{-1}$ [3].

3.2 实验仪器

(1) 光电倍增管。

(2) 示波器。示波器的型号为 Tektronix TDS 1002C-EDU. 该示波器为双通道示波器, 存储深度为 2500, 极限采样率为 1.0 GS/s . 后续会对该示波器的性能进行进一步分析。

(3) 电脑。系统为 Windows 8.1, 主频为 2.9GHz。由于这一部分实验对电脑本身性能要求不高, 剩余参数不再给出。

(4) Sensor-CASSY 2 及其配套的 MCA Box。Sensor-CASSY 是 Leybold 公司生产的用作教学目的的传感器, Sensor-CASSY 2 是 Sensor-CASSY 的第二代。它能够像示波器一样采集电信号。Sensor-CASSY 2 本身不像示波器一样具有处理数据的能力, 它只能将数据传输回电脑。但另一方面, Sensor-CASSY 2 传输数据的能力比示波器要强得多, 这使得它在该实验上比示波器更适合用于采集数据。其结构如下图所示。

虽然 Sensor-CASSY 2 在该实验中之前的主要用途是搭配 MCA Box 进行多道分析, 但实际上选择合适的连线之后, Sensor-CASSY 2 可以直接从光电倍增管中接线到 INPUT B 模块的两极, 从而直接取代示波器的采集功能; 并且 Sensor-CASSY 2 没有显式的单次采样时间限制, 这也是认为其数据采集能力要强于示波器的原因。



图 1: Sensor-CASSY 2 的结构

Sensor-CASSY 2 本身无数据处理功能。但从上图中可以看到, Sensor-CASSY 2 在面板上具有三个模块, 其中 INPUT A 和 INPUT B 上都带有一个串口。Leybold 另外提供 Sensor-CASSY 2 的一些可选配件, 这些配件上也带有串口, 可以接在 Sensor-CASSY 2 的串口上。输入信号先通过附加件, 经过处理后再传输给 Sensor-CASSY 2(如下图所示)。这样, Sensor-CASSY 2 就具有了一定的数据处理能力。

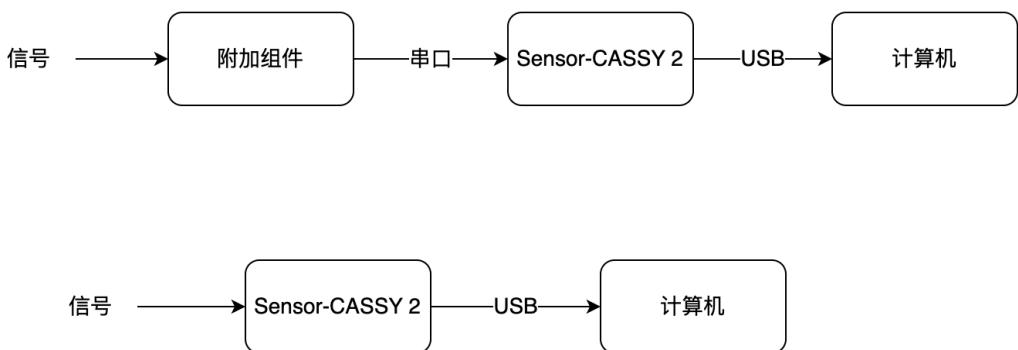


图 2: Sensor-CASSY 2 的两种运行模式

MCA Box(Multiple Channel Analyzer Box, 多道分析仪) 就是其中的一种配件, 结构如下图。信号进入 MCA Box 时, 其中的脉冲信号会被统计, MCA Box 通过脉冲的幅度放入相应的道, 最后形成多道分析的能量谱。



图 3: MCA Box

虽然 Sensor-CASSY 2 采集能力优秀, 但本实验中最后的解决方案并没有用到这两个仪器, 在后续部分中将会陈述原因。[\[5.3.1\]](#) 在实验过程中, 我们完成了对多道分析仪的编程操作。我们认为该仪器还具有相当的探索空间, 因此也将其使用方法记录在该报告中。

Sensor CASSY-2 与 MCA Box 的官方文档见于[此处](#) 和[此处](#)。

3.3 实验软件架构

3.3.1 示波器的软件架构

VISA(Virtual Instrument Software Architecture, 虚拟仪器软件架构), 是通用的仪器编程 I/O 标准与规范的总称, 由 VXI plug&play 制定。本实验中使用的 Tek 示波器支持使用 VISA 标准进行编程。VISA 标准有

许多实现, Tektronix 自身也提供一套 VISA 的实现 TekVISA。本实验中使用的是一种更常用的实现 NI-VISA, 该实现由 NI(National Instrument, 美国国家仪器有限公司) 完成。下图为 VISA 标准实现与仪器通信的框架。

NI-VISA 的网站见[NI-VISA 官网](#)。

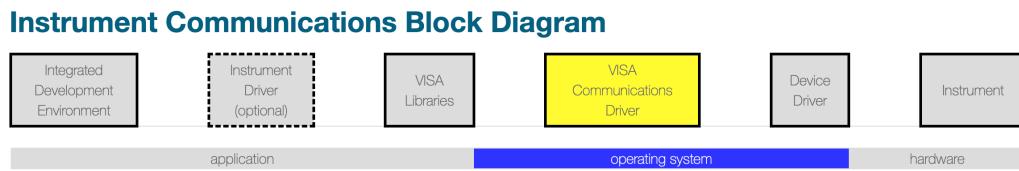


图 4: VISA 框架

NI-VISA 驱动附带 C/C++、VB 的库, 因此可以直接使用 C/C++ 进行编程。在 20 级同学所完成的实验中, 采用的即是 C++ 的 VISA 接口。经过阅读, 我们认为 C++ 的 VISA 接口容易造成较差的可读性, 不利于后续继续开发, 同时观察到该实验对计算性能的要求并不高, 因此采用了 PyVISA。PyVISA 是一个 VISA 的前端, 提供较为简单的 Python 接口, 能够较大程度上简化代码组成。PyVISA 仅仅是一个前端, 需要有一个 VISA 后端 (如 NI-VISA) 才能运行。

实验室的电脑上, 已经安装有 NI-VISA 和 PyVISA。

3.3.2 主要算法的软件架构

由于通信接口上选择了 PyVISA 这样基于 Python 的前端, 我们的算法也全部使用 Python 实现。主要涉及 Python 的科学计算相关包环境, 如 Numpy、Scipy 等。这类包较为常用, 且不是实验重点, 在此不作赘述。

3.3.3 Sensor CASSY-2 的软件架构

Sensor CASSY-2 官方提供了两种可供读写的方式。一种是通过官方的 CASSY LAB 2 软件, 在 CASSY LAB 上可以比较直观地操作; 另一种是通过编程的方式。Leybold 官方提供开发者工具包 (SDK)(见于[此处](#)列表中的 CASSY SDK)。CASSY SDK 基于 .NET Framework 框架编写, 编程语言为 C#。其中的 Api/LD.Api.dll 类库提供了对 CASSY 所有相关仪器的接口。由于涉及.NET 开发, 该部分在进行编程时需要使用 Visual Studio。由于使用了.NET Framework 中有关串口通信的模块, CASSY SDK 对.NET Framework 的版本也有一定的要求; 实测.NET Framework 4.8 可以在实验室的电脑上满足要求。

3.3.4 图形界面相关软件结构

我们主要使用了 Python 中基于 tkinter 的 ttkbootstrap, 其是一个的界面美化库, 使用这个工具可以发出类似前端 bootstrap 风格的 tkinter 桌面程序。接下来我们大致说明一下图形界面编写架构, 如果后人想要修改代码可以来这里寻找一些可乘之机。

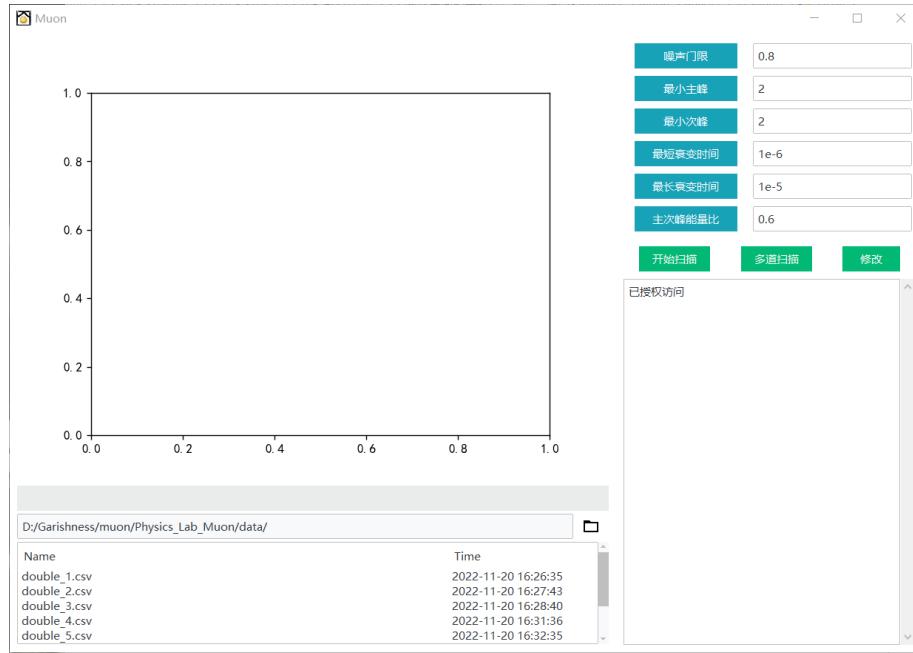


图 5: UI

我们将图形界面的大致分为四个区域，左上角为数据展示区，左下角为数据选择区，右上角为按键功能区，右下角为日志输出区。

- (1) **数据展示区:** 使用 FigureCanvasTkAgg 函数，支持了 plt 的绘图，下面的按钮可以切换显示，是使用一个循环自增的全局变量控制显示图像的。
- (2) **数据选择区:** 这里又分为文件选择与文件树
 - a. **文件选择:** 默认为同目录下“/data”文件，这也是扫描数据写入与多道分析的目录。
 - b. **文件树:** 展示目录下的文件，新扫描的文件也会加入文件树，依据“_x.csv”的 x 排序，同时也会做命名格式筛选，忽略不符合命名格式的文件。左键双击单个数据可以选择展示这个数据。
- (3) **按键功能区:** 这部分集成了扫描，多道与超参数输入的按钮。多道与扫描严格来说并不冲突，但设置了不能同时使用。
 - a. **超参数输入:** 这里可以修改超参数，只要能被 float() 转化都可以，理论上而言可以再任何时候修改参数，不过不推荐这么做。
 - b. **多道:** 会遍历数据目录中的所有文件进行多道统计，统计时间长达几分钟，统计完毕会在数据展示区展示统计结果。使用了另一个线程进行处理防止统计时的卡顿。
 - c. **扫描:** 若有示波器可以进行扫描，使用了切换式的开关，重复按下按钮可以停止扫描，停止标志是在末尾进行停止标志的探测，探测成功即停止。也使用了另一个线程进行处理防止统计时的卡顿。
- (4) **日志输出区:** 可以输出日志，日志禁止了修改。

4 实验

4.1 实验方法

对于如何寻找最低采样点, 我们修改了凸包算法来寻找绝对值较大的最低采样点, 然后用维护区间最小值索引的方法拾遗一些凸包无法寻出的绝对值较小的最低采样点。

4.1.1 原本的凸包算法

引入几个概念: [4]

- **凸多边形** 凸多边形是指所有内角大小都在 $[0, \pi]$ 范围内的简单多边形。
- **凸包** 在平面上能包含所有给定点的最小凸多边形叫做凸包。实际上可以理解为用一个橡皮筋包含住所有给定点的形态。

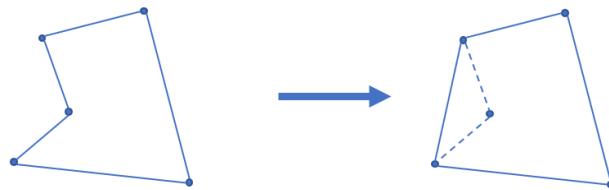


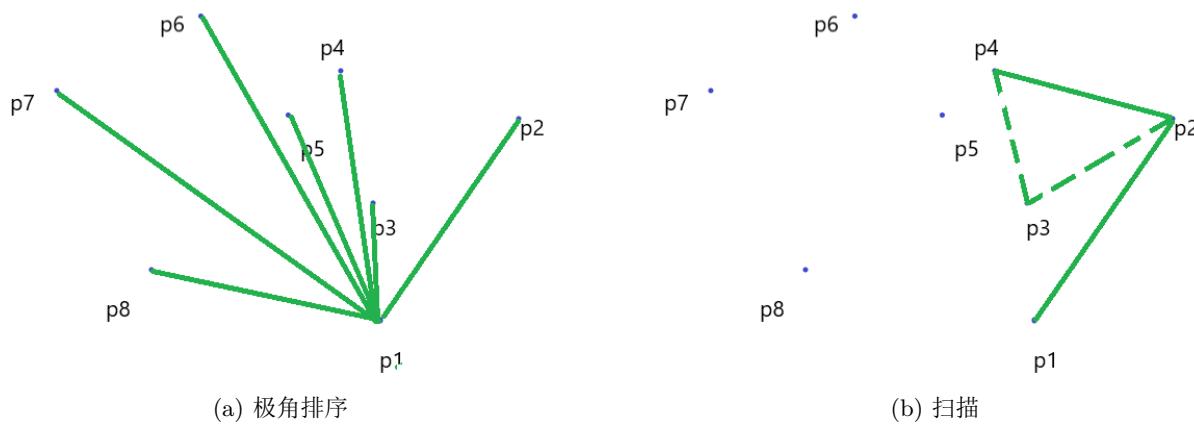
图 6: 凸包

- **凸壳** 凸包的一部分就是凸壳。

这里使用寻找凸包的算法是 Graham 算法。其原理是维护一个凸壳, 通过不断在凸壳中加入新的点和去除影响凸性的点, 最后形成凸包。接下来是对 Graham 算法的详细介绍: [5]

(1) 排序: Graham 算法的第一步就是对点集进行排序, 即选定一个原点, 然后对其他点进行极角的大小逆时针排序。我们示波器扫描出的点可以看作以正无穷远的点为原点所已经极角排序后的结果, 虽不完全符合 Graham 算法有序的定义, 但也有足够用来寻找凸包了。

(2) 扫描: 扫描基本上就是根据排序的顺序进行出入栈, 假设现在 p_4 要入栈, 判断 $\overrightarrow{p_2p_3} \times \overrightarrow{p_3p_4}$ 的正负, 如果是负的说明 p_3 不在凸壳之内, 将 p_3 弹出, 如果有多个这样的点就将他们都弹出, 然后将 p_4 入栈。Graham 算法就是重复这个流程直到扫描过所有点。



Graham

```

1      # p[] 存的是 (x,y) 坐标, np.cross 是叉乘函数, st[]存的是凸壳上的
2          点在 p[]中的索引, top 是栈顶
3
4      top=0
5      st[top]=i-1
6      top+=1
7      st[top]=i
8      while i<n:
9          while top>1 and np.cross(p[st[top]]-p[st[top-1]],p[i]-p[st[
10             top]])<=0:
11              top=top-1
12      top+=1
13      st[top]=i
14      i+=1

```

(3) 正确性概述: 在栈内的所有点所构成的所求多边形, 满足内角属于 $[0, \pi]$ 的凸包性质。且被筛去的点都包含在所求多边形之内。所以 Graham 算法扫描出的点 \Leftrightarrow 这些点的凸包。

(4) 复杂度分析: Graham 算法的复杂度主要是在排序上, 但我们默认点已经是有序的省去了排序的操作, 所以复杂度只剩下扫描的 $O(n)$, 是比较优秀的算法。

4.1.2 修改后凸包算法

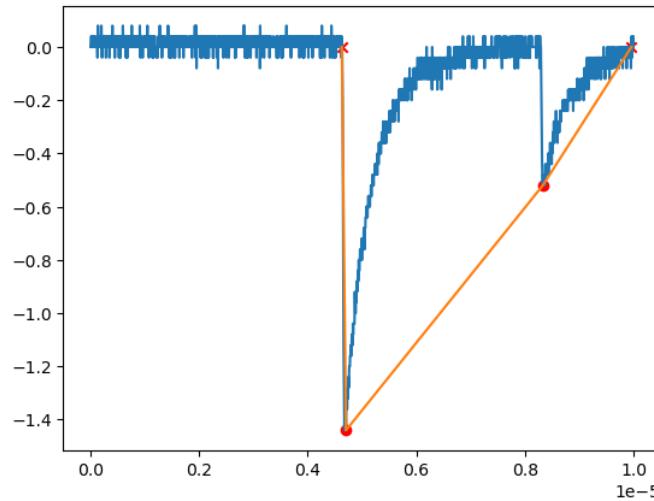


图 8: 例子

上图是一个算法扫描的例子, 我们选择了第一个超过噪声门限的点作为起点, 然后开始 Graham 算法的扫描过程, 直到扫描到的点低于噪声门限或者达到数据尽头。可以看到在这个例子中我们非常好的把主峰与次峰的最低点都找到了, 然而现实中多是并没有那么的美好, 真实的数据中会有各种花里胡哨的情况要去处理, 所以我们对凸包算法进行了些修改, 在维持其主体的情况下增添了许多细节, 来保证所寻之点是我们想要的

峰的最小值。同时可以看出，若有多组峰存在，我们会依据噪声门限进行分割，进行多次 Graham 算法的扫描过程。

(1) **噪声门限**: 我们选择了 0.8V 作为噪声的门限，因为拥有这个门限，所以保证我们在进入 Graham 算法时都是在峰的下降沿，因为有时这个下降沿可能也会构成凸壳，所以我们额外增加了一个转折回升判断来确保寻到的第一个是峰顶，同时根据凸包性质，后面所寻的点也应该是峰顶。

主峰判断

```

1 #fg为找到主峰的标记
2 if (not fg) and top>1 and p[i][1]-p[st[top]][1]>=0:
3     fg=top+1#找到了峰
4     ans[0]=ans[0]+1
5     ans[ans[0]]=i

```

但同时因为有些次峰与主峰之间的扫描点可能回落到噪声门限以下，所以可能会被算法分割开来，我们就在扫描之后进行了二次匹配，以保证是成对的双峰。(此功能在程序外实现)

因为主峰一般会比较高，我们也尝试提高了进入的门限，即最小主峰大小，来减少 Graham 算法空跑的次数，以提高算法运行效率。

(2) **峰顶降噪**: 下图可见在峰顶较小范围之内噪声影响十分巨大，所以需要进行降噪，因为寻峰只需要最小值点，所以我们在保证算法正确性的前提下，选择了 $tesp = 100$ 对采样率 $4ns$ 与 $tesp = 4$ 对采样率 $0.1\mu s$ ，这个 $tesp$ 区间的意义是对每个关键点(需要判断其是否是峰的点)，我们直接视这个区间内的最小值为其峰值，如果一个区间内有多个关键点，就把这个几个关键点合并，粗略跑了几组数据后发现效果较好。

这种行为本质其实还是滤波，不过和普通取加权平均值滤波的区别是其使用的是区间最小值，而且只是关键点周围区间的最小值，这样不仅可以最大程度保留原有图像的特性，也能同样达到一定滤波的效果，可以说一举两得。

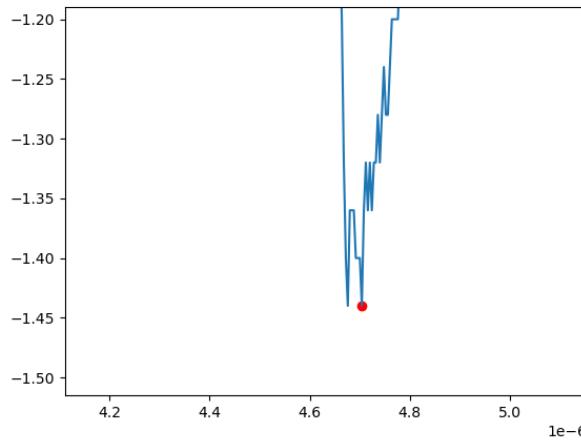


图 9: 峰顶

(3) **最小峰宽**: 这个主要是为了处理一些突增超过噪声门限，但又并不是衰变峰的噪声，在实际实验中这样的噪声还是有较多存在的，无法将其看作误差忽略。

(4) 平顶处理: 判断出平顶后在平顶的最前与最后进行标记。判断平顶依据是连续三个点相等, 在未降低 0.4ns 采样率之前这个判据会更加复杂, 因为点更密集所以连续三点情况很多也不是平顶, 但降低采样率到 $0.1\mu\text{s}$ 之后连续三个点基本只有平顶才会如此了。如果遇到极端情况只有少于 t_{esp} 点的平顶会被下步最后的统一再处理合并为一个点。

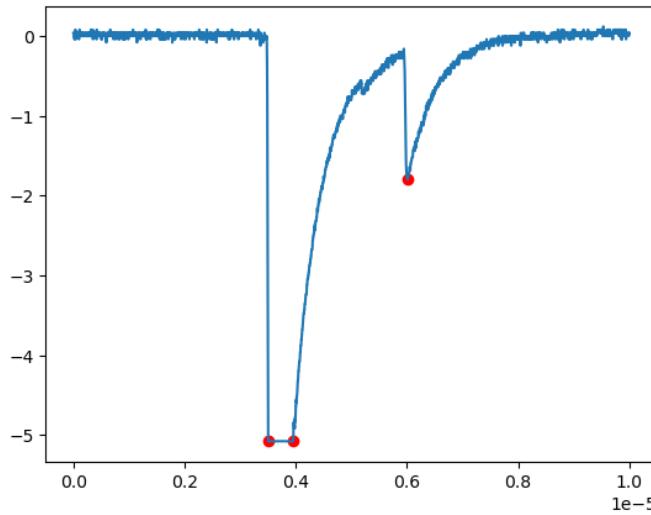


图 10: 平顶

```

1   if p[i][1]==p[i-1][1] and i<n-1 and p[i][1]==p[i+1][1]:
2       ans[0] += 1
3       ans[ans[0]]=i
4       while p[i][1]==p[i+1][1]:
5           i += 1
6       st[top]=i-1

```

(5) 整合处理: 上述一系列处理都是夹杂在 Graham 算法主体之中实现的, 这不可避免的会导致一些遗漏情况, 于是我们在 Graham 算法主体完成之后进行了最小峰宽的判断, 同时顺便对比较重要的峰顶降噪进行了二次判断, 这里主要是判断降噪的关键点是否会被同一个 t_{esp} 包含, 若包含要将之合并。如此对峰的初步处理就完成了。

```

1   if fg and cnt>=dur:#间隔判据
2       # print(fg,st[:top+1])
3       # print(ans[:ans[0]+1])
4       fg=fg+1
5       while fg<=top and p[ans[ans[0]]][0]+tesp>=p[st[fg]][0]:
6           fg=fg+1
7       while fg<=top and (p[st[fg]][0]<p[i][0]-tesp if i<n else st[fg]
8           ]!=i-1):
9           ans[0]=ans[0]+1
          ans[ans[0]]=st[fg]

```

```

10         bias=fg
11         # print(fg)
12         maxn=p[st[fg]][1]
13         while fg<=top and p[st[fg]][0]< p[ans[ans[0]]][0]+tesp:
14             if maxn>p[st[fg]][1]:
15                 maxn=p[st[fg]][1]
16                 bias=fg
17                 # print(maxn,bias)
18                 fg=fg+1
19                 ans[ans[0]]=st[bias]
20                 if p[st[bias]][0]+tesp>p[n-1][0]:
21                     ans[0]=ans[0]-1
22                 # print(ans[:ans[0]+1])
23                 aans=np.vstack((aans,ans))#结果存于此

```

4.1.3 维护区间最小值索引的方法

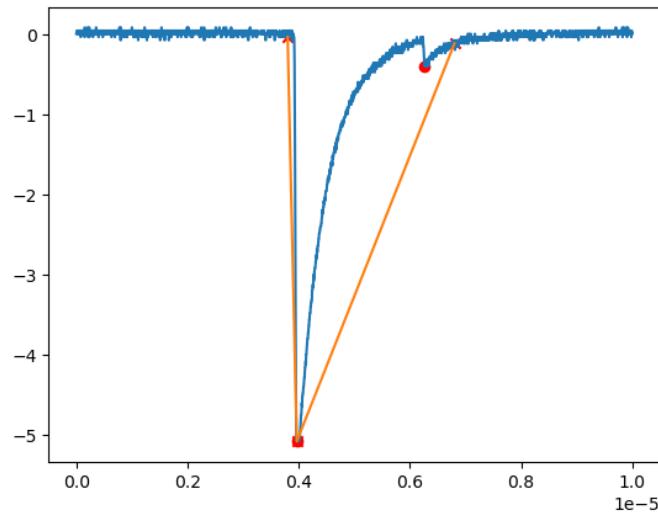


图 11: 次峰

但是这个算法还有问题存在，上图所示较小的次峰因为我们无法很好的确定结束点导致可能会被遗漏，对此我们使用初步处理的结果对数据进行二次处理，目标就是将这些遗漏点重拾。在这里，我们使用维护区间最小值索引的方法。

(1) 算法流程: 在绝大多数情况之下，这种小次峰只可能出现在一次 Graham 算法结束的边缘，所以我们取每次的 Graham 算法为间隔，使用这一次的最后一个峰顶与下一次的第一个峰顶构成一个区间，维护这个区间最小值的索引，若没有下一次的扫描，则以结束边缘作为区间端点。

对这个最小值的索引我们分两种情况讨论:

- 最小值的索引在区间的端点附近: 这会是常见的情况，因为我们所取的区间本就是以峰的最小值为端点，若我们想要找到这两个“大峰”之间的“小峰”(以形状来看)，我们选择逐渐缩小整个区间，更

具体的, 当索引出现在左端点附近时我们将区间左端缩小, 右端同理, 如此反复, 直到索引不再出现在端点, 或者区间长度小于了一个峰可能的长度。

b. 最小值的索引在区间内, 远离端点: 索引不再出现在端点且区间长度仍大于一个峰的长度, 那么索引点也必然是一个峰的最小点了, 我们将这个点添加到输出即可。

但是这个索引点是有概率误判突增的噪声为次峰的, 所以我们提高了判断这个索引点的噪声门限, 即程序中最小次峰大小, 降低误判概率。

(2) 正确性概述: 一个只有一个峰的给定区间的最小值一定是这个峰的峰顶, 这是维护区间最小值索引的基本思想。其主要的难点是如何确定这个区间, 我们通过初步处理后的结果可以快速选定有限个的区间进行扫描, 同时也尽量保证了一个区间只有一个峰, 且区间最小值就是这个峰。

其实这里还有另一个以这个算法为基的寻峰算法: 先对整个区间寻最小值索引, 再通过索引把区间分开, 对每个子区间维护最小值寻峰, 寻到峰就再把子区间分开, 如此往复直到所有峰被找到。这个算法的效果与效率应该也都是比较好的, 但没时间验证了, 交给后人了。

(3) 复杂度分析: 我们只可能进行常数级的 Graham 算法扫描, 然后对每次扫描我们会进行一个维护区间最小值的寻峰, 因为缩小区间最坏最坏只可能遍历一次原数据, 所以复杂度是 $O(n)$ 的, 其主要复杂度瓶颈是在维护最小值上。我们是使用 numpy 库自带的 argmin 去求的最小值, 我并不清楚其具体实现, 最坏估计其是 $O(n)$ 的, 于是算法复杂度上限就为 $O(n^2)$ 。但是维护区间最小值可以使用众所周知的堆来维护, 这样寻求最小值复杂度就降到了 $O(\log n)$, 总复杂度就是 $O(n \log n)$, 不过实现太过繁琐, 就不去作此优化了。

4.1.4 低采样率方法与峰值修正策略

先前的方法中, 由于单次采样时间长度只有 $10\mu s$, 故一次采样最多只能找到一个 μ 子。同时, 我们观察到, 在先前的采样率下, 在单个峰上就会采样非常多的点, 但事实上确定一个峰只需要能采集到峰顶的点就可以了。综合判断, 我们认为先前的采样率是过高的。通过降低采样率, 在固定的存储深度下, 就能相应地延长单次采样时间。在采样时间足够长的时候, 虽然不能做到连续采集, 但是就会有一次采样捕捉到多个 μ 子的可能, 从而提高采集效率。

这就是我们采用的低采样率方法, 将采样率降低至了 $0.1\mu s$, 并将采样后的峰修正以补足降低的采样率。具体的说明详见陈书扬同学的论文。

4.2 实验步骤

此部分主要说明我们所编写的采集程序的使用方法。

4.2.1 采集步骤

打开我们编写的采集软件界面, 其功能已经在软件说明部分简略介绍过了。[\[3.3.4\]](#) 可以通过下方的窗口滑动查看数据, 双击文件名, 可以在上方图形窗口中显示图像。图像中, 散点标识的是识别出峰的位置, 峰的位置已经经过修正。红色表示的成对点是经过匹配得到的双峰, 橙色表示没有匹配的单峰。

由于在低采样率下峰很窄, 未必能够看清, 我们实现了局部放大功能。点击“缩放”, 图片进入放大模式, 此时每次点击“缩放”, 电脑会依次放大显示图中每一个峰周围的详细图像。在放大显示模式下, 我们会画出相关的辅助线。辅助线太多会不利于观察, 我们只保留上升沿的拟合曲线, 用红色虚线表示。

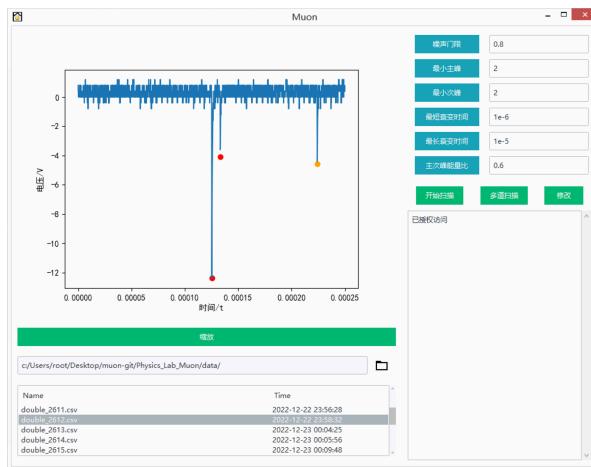


图 12: 浏览图像



图 13: 放大显示

在选定的数据文件夹内，点击右侧的“多道扫描”，可以进行多道分析。多道分析的道数目前只支持在程序内修改，且运行速度较慢，约 6000 份的数据需要运行 2~3 分钟。

多道分析运行结束后，绘图区会显示电子能量分布和 μ 子能量分布的结果。初始状态默认两个分布绘制在一张图上，点击下方的“切换”可以依次显示单个分布。同时，右侧的日志框中会显示所探测到 μ 子的数量和分析得到的平均衰变时间。

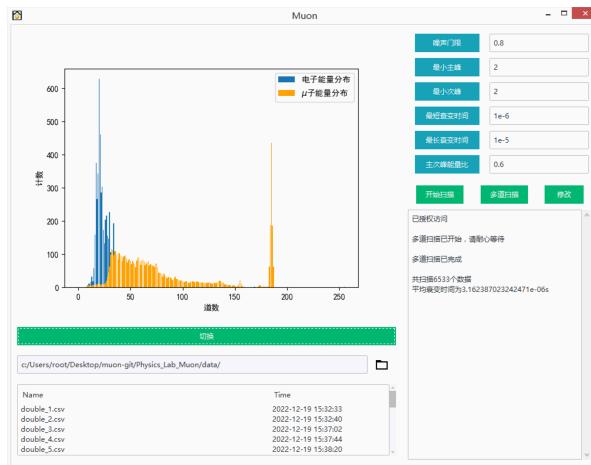


图 14: 多道分析结果

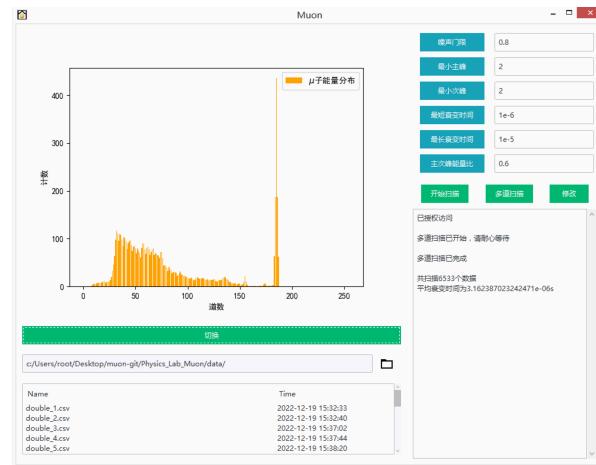


图 15: 单幅展示分析结果

在使用采集功能时，右侧提供了六个允许修改的数据处理参数。

噪声门限 在指定值以下的电压会被视为噪音。

最小主峰 若峰的高度小于指定值，则该峰不会被识别为主峰。

最小次峰 若峰的高度小于指定值，则该峰不会被识别为次峰。

最短衰变时间 若两个峰之间的距离小于指定值，则不认为这两个峰可能是衰变的关系，因为太近了。

最长衰变时间 若两个峰之间的距离大于指定值，则不认为这两个峰可能是衰变的关系，因为太远了。

主次峰能量比 如果后峰高度与前峰高度之比大于指定值, 则不认为这两个峰可能是衰变的关系, 因为次峰不可能和主峰差不多高。

参数需要在采集开始之前预先设定, 建议使用默认值。

在选定好存储数据的目录后, 点击“开始采集”。该程序只存储有衰变信号的数据, 默认展示最新采集到的数据。右侧的日志框会显示当前的记录情况。

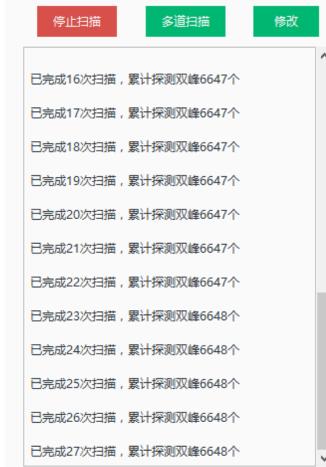


图 16: 采集中的情形

再点击上方的“停止扫描”, 即可停止采集。此时点击“多道分析”, 可以进行数据分析。

4.3 实验数据

该实验为了能顺利进行多道分析, 最终需要采集到的数据量比较大。在经过我们尝试之后, 发现要进行 128 道及以上的数据采集往往需要进行 3 天及以上的采集时间。因此每次采集都会得到较为庞大的数据。示波器每次捕捉的数据都保存为一个 csv 格式的文件, 文件的内容大致为:

```

data > double_15.csv
1 8.000000119209289551e-01
2 4.000000059604644775e-01
3 4.000000059604644775e-01
4 -4.000000059604644775e-01
5 0.0000000000000000e+00
6 4.000000059604644775e-01
7 0.0000000000000000e+00
8 4.000000059604644775e-01
9 -4.000000059604644775e-01
10 -8.000000119209289551e-01
11 0.0000000000000000e+00
12 4.000000059604644775e-01
13 0.0000000000000000e+00
14 0.0000000000000000e+00
15 4.000000059604644775e-01
16 4.000000059604644775e-01
17 4.000000059604644775e-01
18 0.0000000000000000e+00
19 0.0000000000000000e+00
20 0.0000000000000000e+00
21 4.000000059604644775e-01
22 8.000000119209289551e-01
23 4.000000059604644775e-01
24 0.0000000000000000e+00
25 0.0000000000000000e+00
26 4.000000059604644775e-01
27 4.000000059604644775e-01

```

图 17: 数据记录格式概览

共 2500 行, 1 列, 记录的一列数据为 2500 个点分别对应的纵坐标。横坐标是时间数据, 无论在哪一组数据中总是相同, 为了节省存储空间略去不记。

在采集的过程中, 可以将示波器产生的每一个波形都记录下来, 也可以在检测到衰变时再记录波形。这在下一部分会进行讨论。[4.4.2]

4.4 实验结果

4.4.1 测试结果

使用了前几届所留的 38805 份单峰与 5824 份双峰进行测试。最后对双峰有 269 组数据扫描失败, 准确率有 95.38%, 对单峰有 1221 组数据扫描失败, 准确率有 96.85%。这说明我们的寻峰算法相比以前的至少有 95% 的可信度, 是可以使用的。但从后面实验看来, 寻峰算法仍存在不少问题, 在之后具体分析。[5.1]

4.4.2 实际采样结果

在最后一次长时间的数据采集中, 我们在低采样率下进行了总时长 205.4h 的采集。在采集时, 我们只记录了检测到衰变信号的数据, 后续的数据分析也只使用这些带有衰变的数据进行。在往届同学留下的程序中, 所有数据都记录下来 (包括单峰的数据) 并参与了分析的过程。当然, 这两种方式在最后处理数据时都不会构成错误。大部分的单峰数据虽然不能找到衰变的信号, 但是肉眼基本可以判断如此明显的峰基本就是来自于 μ 子, 因此单峰参与能量分布计算是合理的。如果单峰的数据也记录下来, 最终形成的数据量就会相当大。保险起见, 我们最后只收集了检测到衰变信号的数据, 以确保获取到的都是 μ 子。

在上述采集时间内, 我们共获取到 6536 个 μ 子, 平均每个小时 31.82 个。我们也使用上一届编写的程序进行过长时间的采集; 在一次总时长为 239.5h 的采集中, 获取到的 μ 子数量为 2259 个, 平均每个小时 9.43 个。在其他次的采集中, 往届所编写的程序每小时平均采集 μ 子数最多不超过 15 个; 以此估计, 我们的采样方法大致可以将 μ 子的采集效率提高至原来的 2 倍左右。

这样的比较并不完全严谨。首先, 我们的程序和往届的程序并不是同时运行, 因为不能同时有两个程序读取示波器的信息。这就不能保证实验时的真实 μ 子数是一致的。其次, 我们所采用的寻峰算法不同, 可能两个程序都会有一定的误识别。不过, 粗略地可以认为, 我们的程序效率大约是以往的两倍。

如果以收集到的全部峰 (包括单峰) 数据作为比较, 我们的程序效率大约比以往提高 30%。这个指标与上面所说并不是一个指标, 不过这个指标计算起来要更准确。因为往届的算法一次采样必定只能获取一个峰, 最后获取的总峰数就是采样次数。利用这一点, 只运行我们的程序, 同时统计采样次数就可以计算。

只保留衰变数据尽管存储空间占用小, 但多道分析时就不能划分太大的道数。我们所获取的数据最多支持 512 道分析, 1024 道时图像质量会比较差。下面给出的都是 256 道下的结果。

注: 因为没有能量标定, 这里每一道的能量值无实际意义。事实上, 之前的所有图中峰的高度也只有相对大小意义。

μ 子能量分布中, 右侧异常突出的线在往届所提供的能量分布图上也有出现, 并且在我们的图像上这条线还要更高。我们猜测这是平顶所造成的, 有可能虽然平顶经过修复, 但是修复后的值也是集中在这一高度。我们可以对 μ 子的能量分布作一个局部放大, 截掉上面过高的部分, 这样就能看出来轮廓符合 μ 子能量分布的外形。

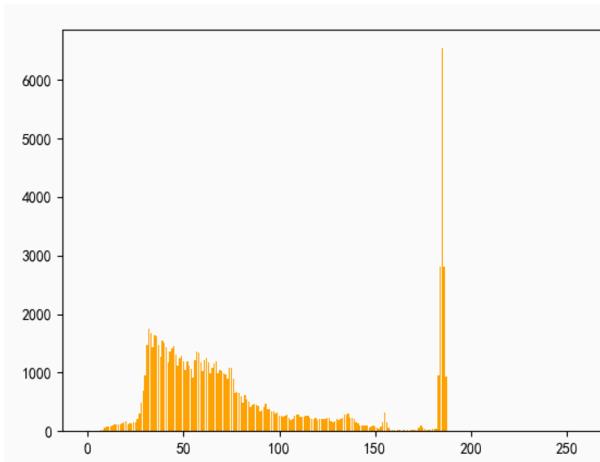
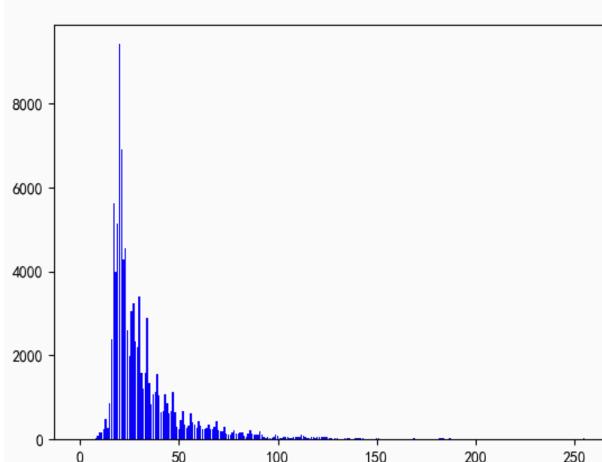
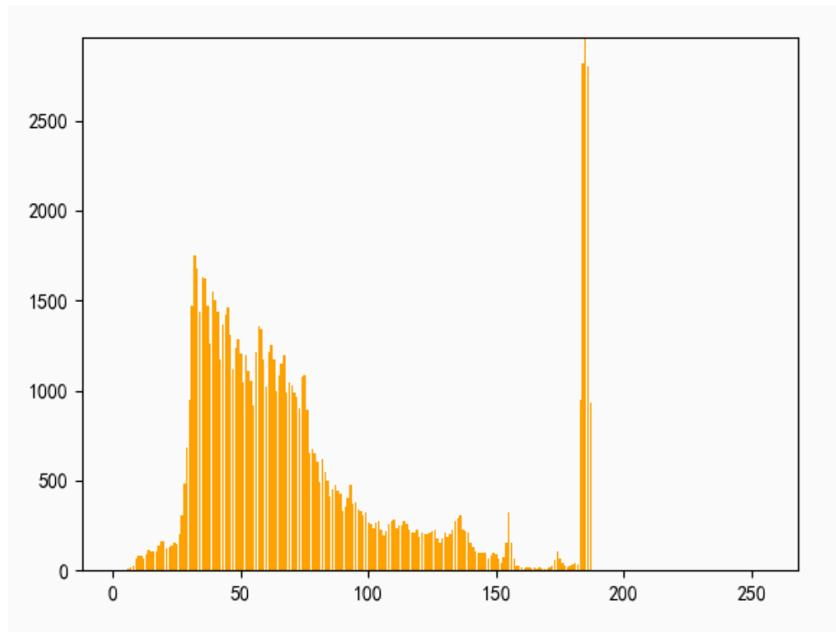
图 18: μ 子能量分布

图 19: 电子能量分布

图 20: 放大的 μ 子能量分布

我们利用这些 μ 子计算得的平均寿命为 $3.16\mu s$. 与理论值相比稍微偏长。在处理时，认为主次峰间隔在 $1\mu s \sim 10\mu s$ 范围之外的都不是 μ 子。

5 分析与讨论

5.1 误差来源

我们的方案可能产生的误差比往届的要多。一方面，在寻峰中会产生误差，即会漏掉峰或者计入一些并不是峰的东西；另一方面，峰值的修正也会有一定的偏差。

因为寻峰程序编写时主要参考的是往届已有并做过标记的数据，并未在新采样率的数据上经过细致的修正，且寻峰算法更偏向于找到更多的衰变峰，对于双峰的情况会出现误判，以上都导致了寻峰误差的产生。

5.2 误差分析

5.2.1 寻峰误差

除非主峰极其罕见的几乎叠加出现, 否则对于主峰的搜寻理论上而言是没有误判的, 主要的误判来源是来自次峰的搜寻。

对于次峰, 若是我们在凸包过程中搜寻到的, 那大概率也的确会是次峰。但若是在维护区间最小值过程中找到的, 就很可能会是噪声峰, 这种噪声峰难以剔除, 因为他与次峰十分的相似, 很难找出他们之前不同的某个特殊特征来进行筛选判断。实验结果也说明我们的次峰标准寻找过于宽松, 导致 μ 子的寿命长于平均值。对于这个现在也没有很好的解决思路, 但可以把他看作拟合匹配问题, 这是有可能被转化为一个优化问题, 运行一些迭代的算法进行计算与判断的。

5.2.2 峰值修正误差

对于下降沿采不到点的情况, 理论上就会产生较大的误差, 只能尽量减小。只要维持低的采样率, 就必然会有这种情况出现。因此, 我们认为想要解决该种情况的问题, 不如研究其他的采样方法来得有效。[\[5.3.1\]](#)

对于真实波形不符合预测模型的情况, 我们也观察到多次。但在理论上, 峰的顶部应该是完全尖锐的。观察到这种情况下波峰位置的抖动往往比较大, 我们初步认为这是噪声或产生峰时的其他干扰所导致。

对于次峰的修正, 如果要修正次峰上的误差, 就需要有确定次峰下降沿起点的方法。但这是相当麻烦的, 因为次峰很有可能在主峰的上升沿上出现, 其左侧不一定是水平线, 确定起来比较复杂。另外, 次峰的下降沿很短, 采到点的概率也相对小。最后, 从修正的价值上来看, 次峰其实没有太大修正的必要, 因为次峰是电子峰, 所产生的是电子的能量分布。我们主要想要得到的是 μ 子的能量分布, 电子的能量分布可以看作一个副产品。

5.2.3 最终测量结果误差分析

平均衰变时间 $t = 3.16\mu s$, 相对于理论值其实偏高不少。时间上的度量与峰高度修正的算法显然无关, 我们选择的采样率也不至于在采集时把峰漏掉, 因此我们认为主要的误差来自寻峰算法。进一步推测, 寻峰算法最主要的误差应该不是漏峰, 而是把一些并不是次峰的波形当做了次峰。这些被当作次峰的杂波往往距离主峰有一定距离, 从而拉长了平均衰变时间。我们曾经将 μ 子的寿命上限调整到 $20\mu s$ 进行实验, 结果 μ 子的平均寿命计算结果达到了 $6\mu s$ 以上, 显然在距离主峰 $10 \sim 20\mu s$ 的区间内又有误被当作次峰的点。这也是我们作出此分析的主要原因。

对于 μ 子和电子的能量分布图, 我们得到的图像外观上与往届得到的结果大致相同。由于能量没有标定, 因此不进行定量的分析。

5.3 实验分析

5.3.1 关于 Sensor-CASSY 2 的可用性

在上述报告中, 我们多次提到 Sensor-CASSY 2 具有优秀的连续采集能力, 但是我们没有采用。原因在于, Sensor-CASSY 2 的极限采样频率只有 $1\mu s^{-1}$, 即每 $1\mu s$ 采样一个点。这个采样率是该实验无法接受的。在采样率分析中, 我们所确定并最终使用的采样率已经是可能进行有效修复的最小采样率, 而这个采样率恰好是 $0.1\mu s^{-1}$, 为 Sensor-CASSY 2 极限采样率的 10 倍。我们也专门在 $1\mu s^{-1}$ 的采样率下进行观察, 发现该采样率下甚至有可能会把整个峰给漏掉, 这显然是不可接受的。

当然, 这并不是说 Sensor-CASSY 2 完全没有用于采样的可能性。我们有两种思路: 其一, Sensor-CASSY 2 在极限采样率下并不是所有采集到的数据都不可用。如果能使用 [\[4.1.4\]](#) 进行修正, 应该还是能还原相当精度的结果的。这样, 采集到的信号将会只有一部分可用。但是, 由于 Sensor-CASSY 2 的采样时间很长, 两者权衡之下, 最终平均探测到的 μ 子数量可能会比原来更多。在我们的实际测试中, Sensor-CASSY 2 至少可以连续采样 1s(可能伴有少许数据延迟), 即有效采样时间比示波器要高 2~3 个数量级, 因此在理论上还是可行的。但是, 这样的做法是否会影响到最终平均寿命和能量分布的计算, 目前还没有讨论。

其二, 可以考虑 MCA Box。该器件在官网上给出的实验示例中就包括 μ 子的探测(见[此处](#)), 因此该附件应该是具有足够的分辨能力的; 并且该器件应该也不是简单的计数器件, 一定能够记录峰的高度数据, 否则无法绘制出能量分布。目前不清楚其是否能够记录时序数据。文档在 CASSY SDK 的说明文档中给出了链接, 但是 Leybold 的编程手册全部为德文, 阅读起来也有一定的困难。

6 实验总结

在本次实验中, 我们更新了往届遗留的采样方式, 创建了峰值修正算法, 使得 80% 以上的峰值复原至参考值的 5% 误差以内, 并解决了平顶修正问题; 同时, 更新了寻峰算法, 使得算法能够在一组信号中具有多个峰的场景下工作。

经过我们的改造, 现在的采样方式同等时间下能够找到的 μ 子数量大约是以往的两倍。不过, 测得 μ 子的平均寿命 $3.16\mu s$ 有些偏大, 寻峰算法还有待改正; Sensor-CASSY 2 的潜力也还没有成功开发出来。

7 课题总结

7.1 仪器系统照片

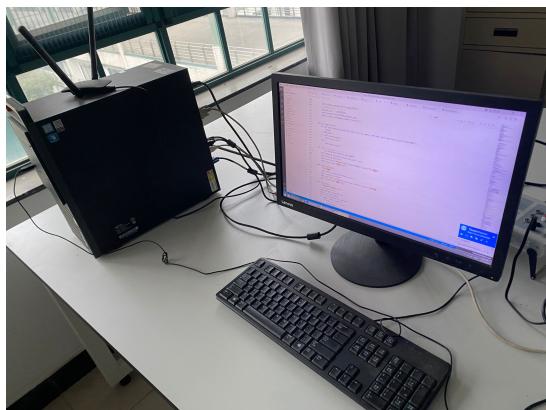


图 21: 电脑



图 22: 塑料晶体

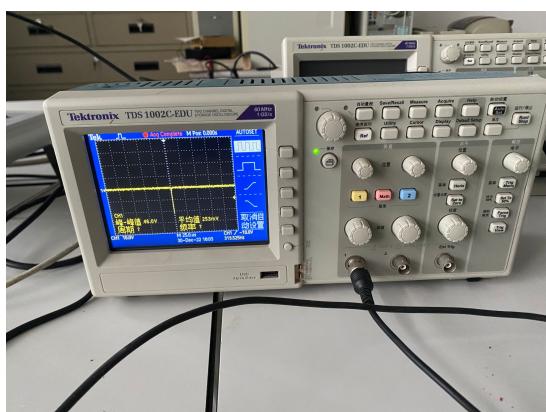


图 23: 示波器



图 24: CASSY-Sensor(老款)



图 25: 连接系统全貌

7.2 课题所用器材清单

Tektronix TDS 1002C-EDU 示波器两台, Windows8 电脑一台, Sensor-CASSY 2 一个, 塑料晶体一个, 光电倍增管一个, 三通头一个, 各种型号线缆若干。

7.3 花絮和留言

ck: 我就只写了寻峰和 GUI, 还被 csy 吐槽代码写得丑, 最后寻峰算法也还要很多问题, 转移到采样率降低的情况也没有经过细致的测试, GUI 也是赶工赶出来的, 不过功能倒是都有了。这个 GUI 写的是真的久, 从头开始搭还是过于耗费时间了, 不过这也和我比较摸鱼有关, 一般除实验课也不会主动去写代码, 都是最后补的天, 下次还是要在平时努力。其他所有的工作都是抱 csy 的大腿完成的, 被带飞真的爽。

下面请队友 c · 实验主导大腿 · sy 发表完结感言:

csy: 这个实验总体感觉工作量还是挺大的。当时提出降低采样率, 觉得只是一个很小的话题, 没想到真就做了一个学期……可能和中间有些时候没思路摸鱼也有关系吧 (x). 另外感觉分工还是挺重要的, 我自己做可能得累死。寻峰算法比较接近 OI, ck 全包了。GUI 也是。不过我自己由于这一次没有查询多少相关文献, 完全就事论事自己想办法, 也还是挺耗脑筋的。最后没能把 Sensor-CASSY 2 用上, 模拟那边也没来得及看, 算是一点小遗憾。

参考文献

- [1] 吕治严. 宇宙线 μ 子寿命测量实验及电子学设计 [D]. 中国科学技术大学, 2009.
- [2] 宇宙射线 μ 子的探测与测量, 刘畅.(前几届同学的工作)
- [3] 宇宙射线 μ 子的探测与测量, 杨凯, 陈星, 鲍德松.
- [4] [OI wiki](#) 计算几何: 凸包
- [5] [洛谷题解 \(需要注册账号\)](#) 【模板】二维凸包题解