ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ФИЗИКИ ЭЛЕМЕНТАРНЫХ ЧАСТИЦ

БАКАЛАВРСКАЯ РАБОТА

«ОПРЕДЕЛЕНИЕ НАПРАВЛЕНИЯ ДВИЖЕНИЯ МЮОНОВ В ЭКСПЕРИМЕНТЕ JUNO С ПОМОЩЬЮ ВОДНОГО ЧЕРЕНКОВСКОГО ДЕТЕКТОРА»

Выполнил студент:

409 группы

Дьяков Роман Игоревич

_____

Научный консультант:

Ст.н.с., к.ф.-м.н.

Чуканов Артем Владиславович

_____

Научный руководитель:

проф., д.ф.-м.н.

Наумов Дмитрий Вадимович

_____

Допущен к защите

Зав. кафедрой, д.ф.-м.н.,

академик РАН В. А. Матвеев

_____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

10 января 2015 года в городе Кайпин, Цзянмэнь, на юге Китая, было начато строительство подземной нейтринной обсерватории Цзянмэнь (Jiangmen Underground Neutrino Observatory, JUNO). Возведение объекта начато в рамках одноименного эксперимента и направлено на определение иерархии масс нейтрино в качестве основной цели.

Строение детектора JUNO изображено на Рисунок 1. Регистрация реакторных антинейтрино происходит в центральном детекторе, представляющего из себя акриловую сферу, наполненную жидким сцинтиллятором, и массив больших и малых фотоэлектронных умножителей (ФЭУ), а также промежуточный буфер воды. Электронные антинейтрино, испускаемые реакторами, фиксируются в детекторе JUNO, благодаря процессу обратного бета-распада $\bar{\nu}_e + p \longrightarrow e^+ + n$, где p – протон из жидкого сцинтиллятора. Регистрируемый сигнал состоит из двух компонент: энерговыделение и аннигиляция позитрона (от 0,7 до 12 МэВ) и запаздывающий сигнал (до 400 мкс), испускаемый в результате захвата нейтрона водородом (ровно 2,2 МэВ). По зарегистрированному сигналу определяется точка взаимодействия антинейтрино и его энергия.

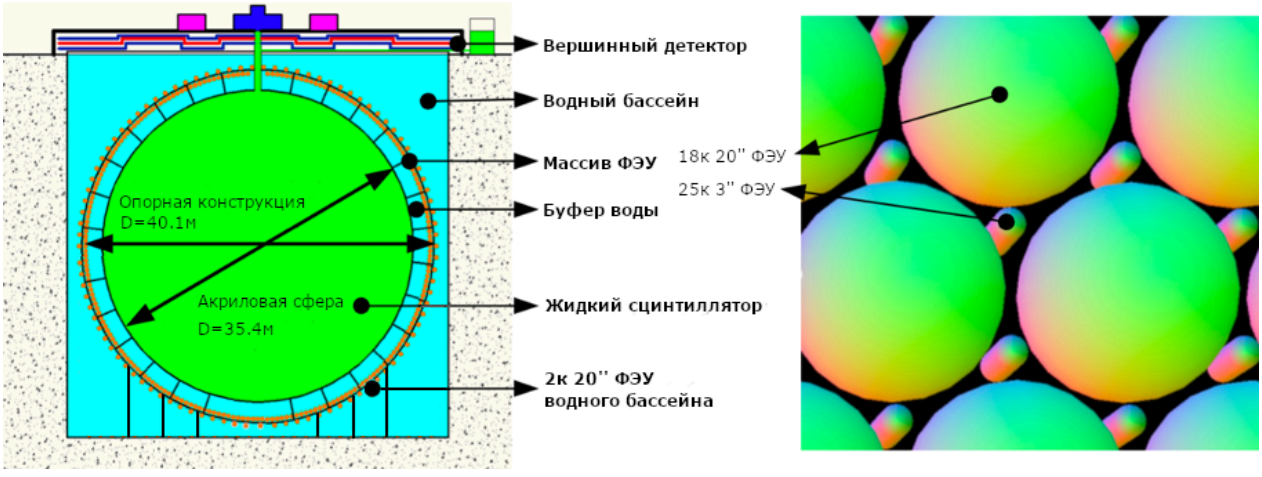

Рисунок 1 – Схематический вид детектора.

Справа показано размещение 3-дюймовых ФЭУ между 20-дюймовыми ФЭУ.

Как и любой детектор, JUNO подвержен воздействию радиоактивного фона, возникающего из компонентов самого детектора, от близлежащий горных пород и от космического излучения. Последний источник вносит наибольший вклад в фоновые события, возникающие в детекторе: симуляция методом Монте-Карло показала, что частота прохождения космических мюонов сквозь центральный детектор составляет порядка 3 событий в секунду со средней энергией 215 ГэВ. Высокоэнергетические космические мюоны и последующие ливни могут взаимодействовать с веществом сцинтиллятора и производить радиоактивные изотопы ^{9}Li и ^{8}He, которые могут распадаться, испуская бета-частицу и нейтрон, имитируя сигнал антинейтрино.

В целях более эффективного поиска и изучения нейтрино необходимо исключить область детектора через которую пролетел мюон на время, много большее времени жизни родившихся изотопов. Для этого в программном обеспечении детектора устанавливается вето на рассмотрение области пролета.

В отличие от нейтрино, мюонные треки также можно фиксировать в водном бассейне (Рисунок 1) в целях последующего использования точек входа-выхода мюона для определения направления полета частицы и уточнения уже найденных треков в центральном детекторе. Соответственно, целью данной работы является определение направления движения мюонов в эксперименте JUNO с помощью водного черенковского детектора. Для достижения поставленной цели необходимо провести предварительное исследование конструкции JUNO, затем разработать алгоритм поиска мюонных треков и, как итог, написать программу для определения направления движения мюонов.

В ходе работы будут рассмотрены различные алгоритмы поиска мюонов, их сравнение и выбор наилучшего варианта, реализация алгоритма и возникшие в ходе реализации проблемы и статистика по точности определения мюонных треков.

1. Детектор JUNO

Подземная нейтринная обсерватория JUNO – многоцелевой сцинтилляционный детектор массой 20 килотонн, строительство которого было предложено для определения иерархии масс нейтрино как основной цели в физике элементарных частиц. Превосходное энергетическое разрешение и большой доверительный объем, ожидаемые для детектора JUNO, предлагают широкие возможности для решения многих важных тем в физике нейтрино и астрофизике частиц. Предполагается, что за 6 лет работы детектора накопленная статистика позволит определить иерархию масс нейтрино с доверительной вероятностью $3-4\,\sigma$. Измерение спектра антинейтрино с отличным энергетическим разрешением также приведет к точному определению параметров осцилляций нейтрино $\sin^2\theta_{12}$, Δm^2_{21} и $|\Delta m^2_{ee}|$ с погрешностью менее 1%, что сыграет решающую роль в будущем испытании на унитарность матрицы MNSP.

В первую очередь JUNO построен для наблюдения потока антинейтрино от атомных электростанций Yangjiang и Taishan на побережье Южно-Китайского моря. Однако, обсерватория способна также наблюдать потоки нейтрино/антинейтрино из других наземных и внеземных источников, в том числе нейтрино от вспышек сверхновых, диффузный фон сверхновых звезд, геонейтрино, атмосферные нейтрино и солнечные нейтрино.

Предлагаемая конструкция детектора JUNO обеспечит уникальную возможность своевременно и экономически эффективно решать многие важные вопросы в области элементарных частиц и астрофизики. Он обладает огромным потенциалом для дальнейшего продвижения в нашем стремлении понять фундаментальные свойства нейтрино, одного из строительных блоков нашей Вселенной.

Детектор JUNO состоит из центрального детектора, водного черенковского детектора и мюонного трекера. Центральный детектор состоит из жидкого сцинтиллятора с доверительной массой с расчетным энергетическим разрешением $3\%/\sqrt{E(MeV)}$. Центральный детектор погружен в

цилиндрический бассейн с водой. Не менее 2 м воды в любом направлении защищает центральный детектор от радиоактивности окружающей породы. Около 1600 20-дюймовых ФЭУ установлены в бассейне с водой для обнаружения черенковского света от внешнего фона, выступающие в качестве вето-детектора. Ожидается, что эффективность обнаружения мюонов будет такой же, как и у водного черенковского детектора эксперимента Daya Bay, что составляет 99,8%.

Основной вклад в радиоактивный фон вносят естественная радиоактивность близлежащих горных пород (U, Th, K), воды и атмосферы (U, Th, K, Rn) и материалов самого детектора (U, Th, K, Co, Kr, Ar), а также высокоэнергетические космические лучи. Высокоэнергетические космические мюоны и последующие ливни могут взаимодействовать с веществом сцинтиллятора (а именно ^{12}C) и производить радиоактивные изотопы с $Z \leq 6$ в результате электромагнитных или адронных процессов. Среди них ^{9}Li и ^{8}He с периодами полураспада 0,178 с и 0,119 с, соответственно. Они являются наиболее серьезным фоновым источником для реакторных антинейтрино, поскольку они могут распадаться, испуская бета-частицу и нейтрон, мимикрируя под сигнал антинейтрино. Предположительная частота рождения ^{9}Li и ^{8}He в детекторе JUNO составляет 150 и 50 событий в сутки, соответственно.

Фон ^{9}Li и ^{8}He коррелирует с родительским мюоном во времени и пространстве. Расстояние между мюон-индуцированными изотопами и траекторией родительского мюона в среднем составляет от 2 до 6 (при большом электромагнитном ливне) метров. Наиболее эффективный подход для учета фона ^{9}Li и ^{8}He состоит в том, чтобы наложить вето на достаточный объем детектора вдоль траектории мюона в течение относительно длительного времени, например, в несколько раз больше времени жизни изотопа. С помощью мюонной симуляции для JUNO было установлено, что скорость ливня мюонов составляет порядка 1 раза в 2 секунды. Моделирование также предсказывает, что после создания ливня мюон все еще выживает, и его направление изменяется

незначительно. Таким образом, критический вопрос заключается в том, насколько хороша реконструкция мюонного трека.

Мюон в JUNO проделывает следующий путь:

- Перед попаданием в центральный детектор частица проходит через водный бассейн и создает там черенковское излучение. Из-за оптической природы этого света, его зафиксируют только ФЭУ, находящиеся непосредственно в водном бассейне. Несмотря на то, пересечет ли мюон центральный детектор или нет, мы зафиксируем его, т.к. потери на ионизацию составляют $1{,}43 MeV/cm$, что на 1 метр трека создаст не менее $1{,}5 \cdot 10^5$ фотоэлектронов.
- Для учета фона наиболее важными являются мюоны, проходящие через центральный детектор. При входе в центральный детектор мюон сначала проходит через водяной буфер между стальной поддержкой ФЭУ и акриловой сферой. На этом пути он создаст черенковский свет, как в водном бассейне.
- После этого мюон пересекает акриловую сферу и весь объем жидкого сцинтиллятора.
- При выходе из детектора мюон будет пересекать буфер воды еще раз. При этом мы снова фиксируем черенковский свет, но уже отраженный от стенок камеры ввиду конструкции детектора.

Поскольку механизм появления фотонов в воде и жидком сцинтилляторе сильно различается, длина трека в каждой среде оказывает большое влияние на характеристики события. Углы ориентации трека θ и φ оказывают, в свою очередь, меньшее влияние ввиду сферической формы центрального детектора (Рисунок 2). Мюонные треки - самые интенсивные события, ожидаемые в JUNO, что дает им четкую сигнатуру. Тем не менее подробные характеристики зависят в основном от длины трека мюона в жидком сцинтилляторе и в воде.

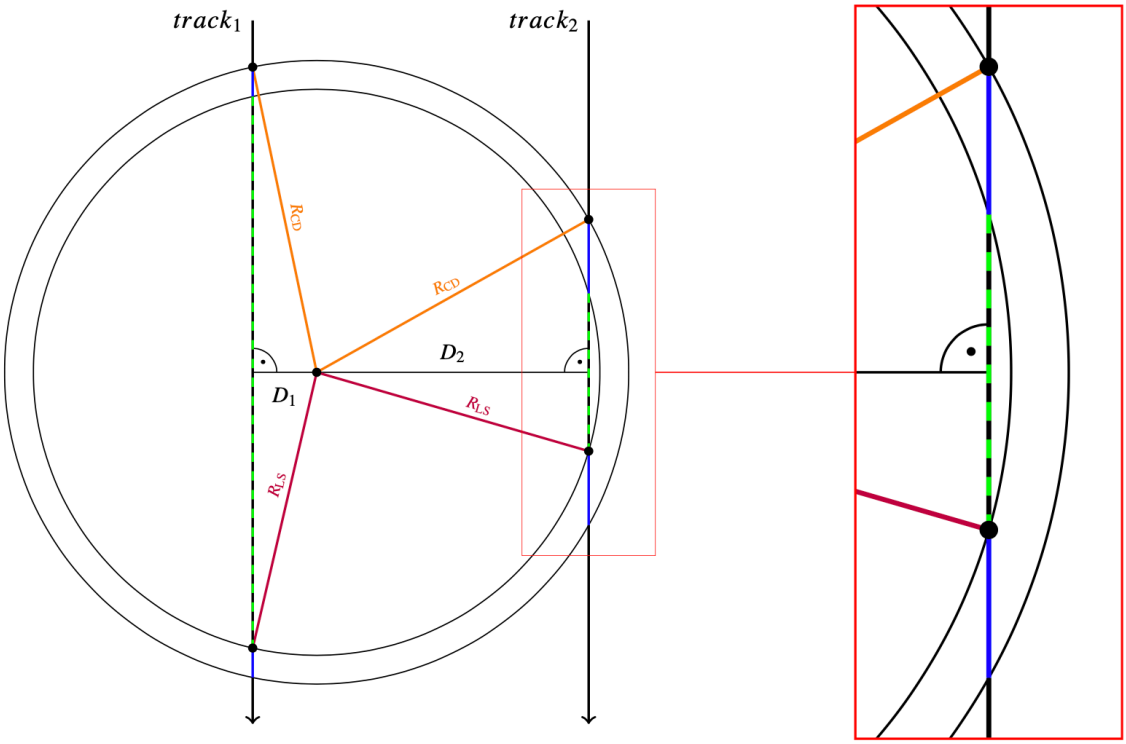

Рисунок 2 – Расположение мюонного трека относительно сферы детектора. Для обоих треков длина в водяном буфере отмечена синим цветом, а длина трека в центральном детекторе указана зеленой пунктирной линией. Длина трека в центральном детекторе неодинакова и зависит от расстояния до центра D. Ближе к краю детектора различить максимумы энерговыделения в центральном детекторе невозможно, однако в водном буфере это два различимых максимума.

Для реконструкции одиночных мюонных треков в центральном детекторе сейчас используется следующий алгоритм:

- моделирование калибровочных мюонов: вертикальных треков с шагом 0,5 м вдоль оси OX;
- представление отклика детектора на калибровочные мюоны в виде суммы конечного числа сферических функций: $f(x) \approx u_0 + u_1 + \cdots + u_m \approx \sum_{n=0}^{m} u_n(\theta, \lambda) = \sum_{n=0}^{m} \sum_{k=0}^{n} P_n^{(k)}(\cos\theta)(c_{nk}\cos k\lambda + s_{nk}\sin k\lambda)$, т.е. поиск коэффициентов c_{nk} и $s_n k$;

- для реконструкции мюонного трека производится вращение полученной функции отклика и подбор расстояния до тех пор, пока она не будет полностью совпадать с откликом детектора на искомый мюонный трек;
- найденные углы вращения и расстояние до центра детектора и будут параметрами направления движения мюонного трека.

Для оптимизации поиска углов вращения и расстояния до центра используется следующий алгоритм:

- сфера была разделена на 3 части: вид сверху, вид сбоку и вид снизу;
- в каждой из частей ищутся максимумы энерговыделения (Рисунок 3);
- из найденных точек составляются различные комбинации мюонов, которые могли пересечь детектор (в комбинацию входят мюоны, максимальный угол влета которых составляет 80 градусов; для двойных и тройных мюонов так же используется условие, что угол между ними должен быть меньше 0,25 радиан, так как в подавляющем большинстве случаев эти мюоны идут параллельно друг другу);
- реконструкция каждой возможной комбинации и выбор той, для которой хи-квадрат минимальный.

Время реконструкции одного события с двумя мюонами составляет примерно одну минуту на 4-х процессорах. Для трех мюонов из-за большого количества перебираемых вариантов на одно событие (около 26), время обработки одного события составляет уже 4 минуты на 4-х процессорах. Это значительное время, которое необходимо уменьшить, для чего требуется оптимизация алгоритма.

Также есть нерешенные вопросы о реконструкции направления неполного трека и реконструкции направления в случае возникновения электромагнитного ливня.

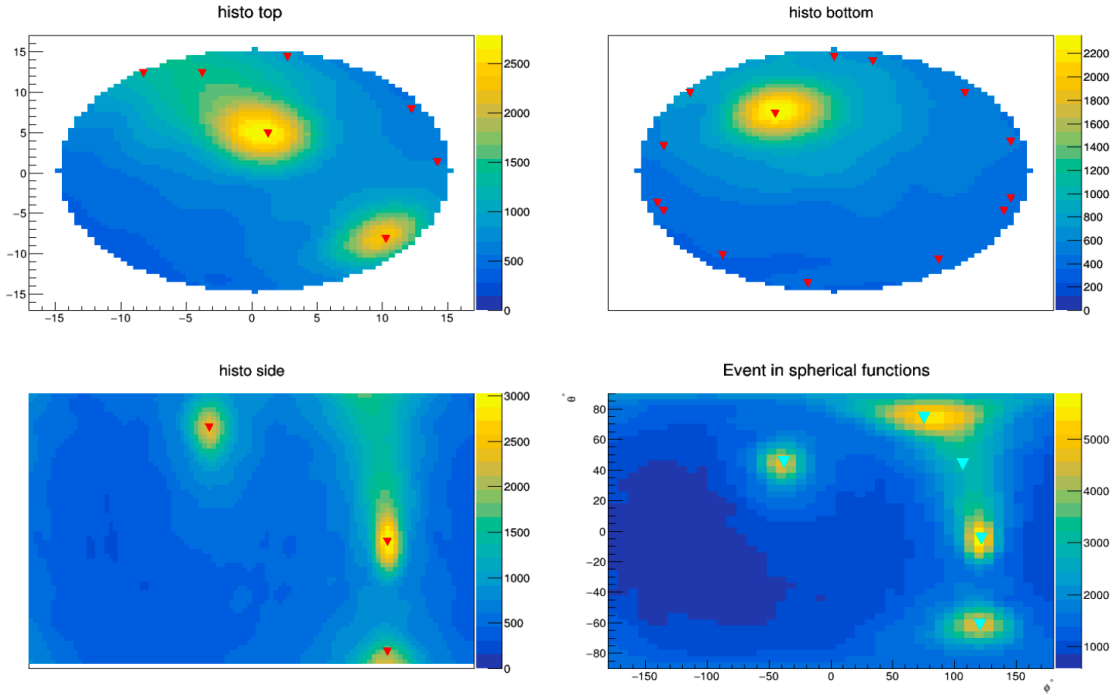

Рисунок 3 – Разделение сферы на части
для независимого поиска максимумов на полюсах

Эти задачи отчасти решаются получением дополнительных данных о пролетающих мюонах в окружающем центральный детектор водном бассейне. Если в бассейне найти максимумы энерговыделения, то мы получим дополнительные опорные точки для реконструкции треков. Это позволит уменьшить количество конфигураций в случаях 2-3 мюонов, что, при оптимальном алгоритме поиска, уменьшит время на реконструкцию заведомо неверных вариантов, а также даст вторую необходимую опорную точку в случаях неполного трека и возникновения электромагнитного ливня.

Для разработки алгоритма поиска мюонов в водном бассейне было произведено моделирование и выгружены данные для событий с одним и двумя мюонами. Далее было начато исследование вариантов поиска максимумов на сфере с использованием Python 3. Основные требования к алгоритму: точность определения точки влета/вылета не более трех метров, высокая скорость работы. Наилучший вариант по результатам тестирования планируется интегрировать в уже работающий код по поиску мюонных треков.

2. Предварительное исследование

В ходе работы над проектом было разработано несколько вариантов решения задачи поиска максимума на сфере. Для проверки этих вариантов было проведено прототипирование на языке Python3 и запуск получившихся алгоритмов на тестовых данных, предварительно выгруженных из программы моделирования.

Формат входных данных следующий: в первой строке указывается количество мюонных треков n, в последующих n строках записаны номер мюона `i`, точные координаты `x, y, z` точек пересечения акриловой сферы, все следующие строки имеют формат `x, y, z, c` и кодируют координаты ФЭУ в водном бассейне и количество фотоэлектронов, зафиксированное этим ФЭУ. Пример входного файла:

```
1
0 12708.5 1389.32 12240.7 12538.7 10564.2 -6668.56
2638.43 459.53 459.53 2
3203.15 1469.22 1469.22 1
5139.96 540.231 540.231 2
...
```

Чтение этих данных организовано в многомерные массивы библиотеки numpy. Массив `io_dec` используется для точек входа-выхода и имеет формат `(x, y, z)` для каждой записи (нас не интересует, какая из точек является точкой входа, а какая точкой выхода, так как нет четких критериев для их разделения), а `points_dec` для фотоумножителей, имеет формат `(x, y, z, c)`. Далее эти массивы преобразуются к видам полярных координат `(theta, phi)` и цилиндрических координат `(theta, z)` посредством процедур:

```
to_rad(dec):
    return np.array([
        np.arctan2(pow(pow(dec[:, 0], 2)+pow(dec[:, 1], 2
), 0.5), dec[:, 2]),
        np.arctan2(dec[:, 1], dec[:, 0]),
    ]).T

to_cyl(dec):
    return np.array([
        np.arctan2(dec[:, 1], dec[:, 0]),
        dec[:, 2]
    ]).T
```

Получившиеся массивы будут иметь суффиксы `_rad` и `_cyl` соответственно (например, `io_cyl`). Эти массивы используются далее для проверки гипотез.

В качестве основных метрик были выбраны время работы алгоритма, количество верных отработок алгоритма (количество верно найденных максимумов энерговыделения) и количество неверных отработок алгоритма (количество ненайденных максимумов, количество неверно найденных максимумов). Исходя из этих метрик выбирается самый удачный алгоритм для последующей реализации.

Время в прототипах замерялось встроенной в jupyter notebook – среды программирования на Python – “магической” функцией `%%time`. Правильность выбора максимума определялась автоматически по вхождению найденной точки в шар диаметра 2 метра вокруг точки смоделированного максимума. Время на сравнение максимума и получение изображения не учитывается в времени работы алгоритма, тесты проводились раздельно.

2.1. Поиск с помощью дифференцирования по сетке

Необходимое условие экстремума: пусть точка x_0 является точкой экстремума функции f, определенной в некоторой окрестности точки x_0. Тогда либо производная $f'(x_0)$ не существует, либо $f'(x_0) = 0$. По этому условию мы можем найти точки локального максимума численно, введя сетку на развертке сферы и проведя численное дифференцирование любым стандартным методом (Рисунок 4).

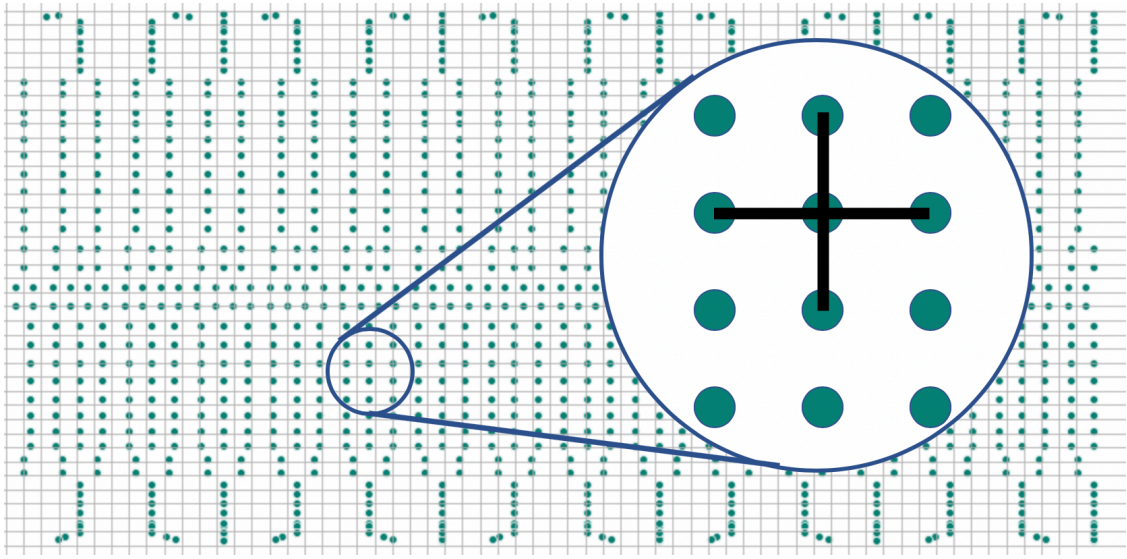

Рисунок 4 – Пример введения сетки для численного дифференцирования

Однако этот вариант сопряжен с трудностями при введении сетки. Как видно на развертке сферы (Рисунок 4), ФЭУ расположены по площади неравномерно, поэтому введение сетки будет сложной процедурой, которая потребует значительного времени и ресурсов. Также она будет зависеть от каждого ФЭУ в отдельности, поэтому небольшое смещение каждого ФЭУ или выход его из строя скажется на работоспособности кода.

В виду сложности реализации и неэффективности от этой идеи сразу отказались и ее прототип реализован не был.

2.2. Поиск с помощью гистограмм

Для приближенного поиска максимума можно ввести равномерную сетку в каждом узле этой сетки посчитать среднее или суммарное значение зарегистрированных фотоэлектронов. Сравнивая значение в каждом узле с соседними, это позволит определить максимумы с произвольной точностью в зависимости от шага сетки.

Этот способ особенно удобен тем, что есть готовые библиотеки, реализующие так называемые гистограммы (по аналогии с методом графического представления данных), которые оптимально реализуют метод построения необходимой структуры данных для дальнейшего поиска. Такая конструкция есть как в библиотеке numpy для Python3, так и в CERN ROOT.

В прототипе использовалась библиотека numpy и объект histogram2d. В самом простом случае использование выглядит следующим образом:

```
H, xedges, yedges = np.histogram2d(points_cyl2[:, 0], poi
nts_cyl2[:, 1], bins=(60, 30))
```

В данном примере мы строим гистограмму для всех ФЭУ в цилиндрических координатах, вводя сетку по полярному углу и аппликанте 60 на 30 соответственно. Массив `points_cyl2 (theta, z)` – массив всех ФЭУ в цилиндрической системе координат, в котором каждая координата повторяется столько раз, сколько фотоэлектронов было зарегистрировано.

Графическая иллюстрация получившейся двухмерной гистограммы изображена на Рисунок 5. На ней невооруженным глазом видно две точки максимума в верхней части рисунка, соответствующие точкам входа мюонов в центральный детектор. В нижней части рисунка имеется два менее выраженных максимума, которые можно найти программным способом, они соответствуют точкам выхода мюонов за пределы акриловой сферы.

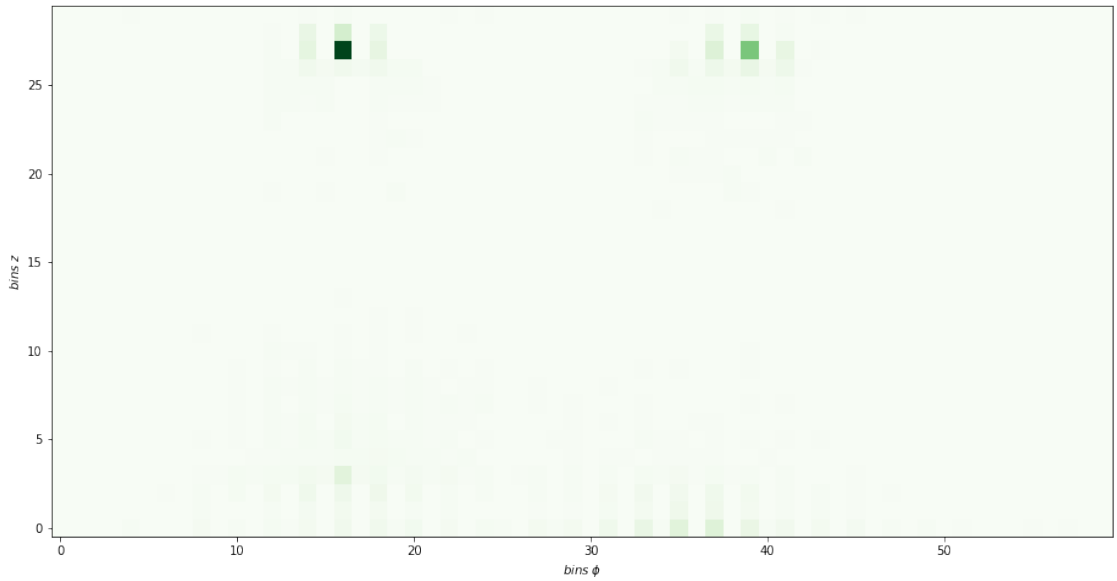

Рисунок 5 – Визуализация гистограммы на наборе ФЭУ в цилиндрических координатах

Как можно заметить на Рисунок 5, максимумы в верхней и нижней частях сферы выглядят как набор полос более яркого цвета. Так происходит из-за того, что в цилиндрических и сферических координатах плотность расположения ФЭУ вблизи полюсов меньше, чем вблизи “экватора” сферы. С этим связана первая проблема данного метода: полюса сферы и ее центральную часть нужно рассматривать по-отдельности, а потом “склеивать” их в одно целое. Также метод имеет довольно высокую погрешность и часто не находит максимумы.

При поиске данным методом большой проблемой является низкая “контрастность” рисунка в его нижней части: ФЭУ в нижней части улавливают отраженный свет из-за чего световое пятно оказывается размазанным и неярким, поэтому определить это пятно не удается. Количество неверно найденных максимумов высокое из-за большого расстояния между ФЭУ на “полюсах” детектора: полосы гистограммы в которые не попало ни одного ФЭУ оставались пустыми из-за чего находилось по несколько максимумов там, где на самом деле был один.

В таблице 1 приведены метрики для дальнейшего сравнительного анализа данного метода с размерами сетки гистограммы 60 по горизонтали на 30 по вертикали. При этих размерах достигается максимальная точность.

Таблица 1 – сравнительные характеристики для метода гистограмм

Характеристика	**Результат**	**Единица измерения**
Время работы	5,13	с/40 изображений
Количество верно найденных максимумов	26	шт. из 60
Количество ненайденных максимумов	34	шт. из 60
Количество неверно найденных максимумов	20	шт.

2.3. Поиск максимума методом интегрирования

Для увеличения точности предыдущего метода был доработан алгоритм работы: теперь при прохождении по сетке шаг делается не в одно деление, а в половину (то есть количество шагов в каждом направлении увеличивается в два раза), полюса рассматриваются по-отдельности, а потом максимумы склеиваются вместе, если расстояние между ними достаточно маленькое.

Данная функциональность не реализована в стандартных библиотеках, поэтому ее реализация заняла значительное время. Составление необходимых структур данных для разделения верхней, нижней и центральной частей представлено в листинге далее:

```
step_dec = 750

_, _, _, max_t = sph_max(points_dec)
points = np.array(sph_cond(points_dec, max_t/6))

top = points[(points[:, 2]) > 8000]
bottom = points[(points[:, 2]) < -8000]
center = points_cyl[abs(points_cyl[:, 2]) < 10000]

def get_square(points, x, y, step_x=2*step_dec, step_y=2*
step_dec):
    return points[(points[:, 0] > x) & (points[:, 0] < x+
step_x) & (points[:, 1] > y) & (points[:, 1] < y+step_y)]

for x in range(-20000, +20000, step_dec):
    for y in range(-20000, +20000, step_dec):
        gs = get_square(top, x, y)[:, 3].sum()
        top_grid.append((x, y, gs))
```

```
for x in range(-20000, +20000, step_dec):
    for y in range(-20000, +20000, step_dec):
        gs = get_square(bottom, x, y)[:, 3].sum()
        bot_grid.append((x, y, gs))

for x in np.arange(-np.pi, +np.pi, np.pi/15):
    for y in np.arange(-10000, +10000, step_dec):
        gs = get_square(center, x, y, np.pi/30)[:, 2].sum
()
        cen_grid.append((x, y, gs))
```

После выполнения данной процедуры получаем похожую на гистограмму структуру, по которой обычным сравнением с соседями находятся максимумы энерговыделения. Графическое представление данных структур изображено на Рисунок 6.

Похожий метод используется для определения максимумов внутри центрального детектора и алгоритм работы хорошо себя показал для подобного поиска. Однако в случае центрального детектора имеется большее количество точек, поэтому в каждый квадрат того же размера попадает большее количество ФЭУ, а следовательно события видны ярче, что позволяет избежать случайных срабатываний.

Метрики для дальнейшего сравнительного анализа приведены в таблице 2. Как видно из таблицы, точность метода возросла по сравнению с предыдущим, но также возросло и количество ложных срабатываний, и время работы алгоритма.

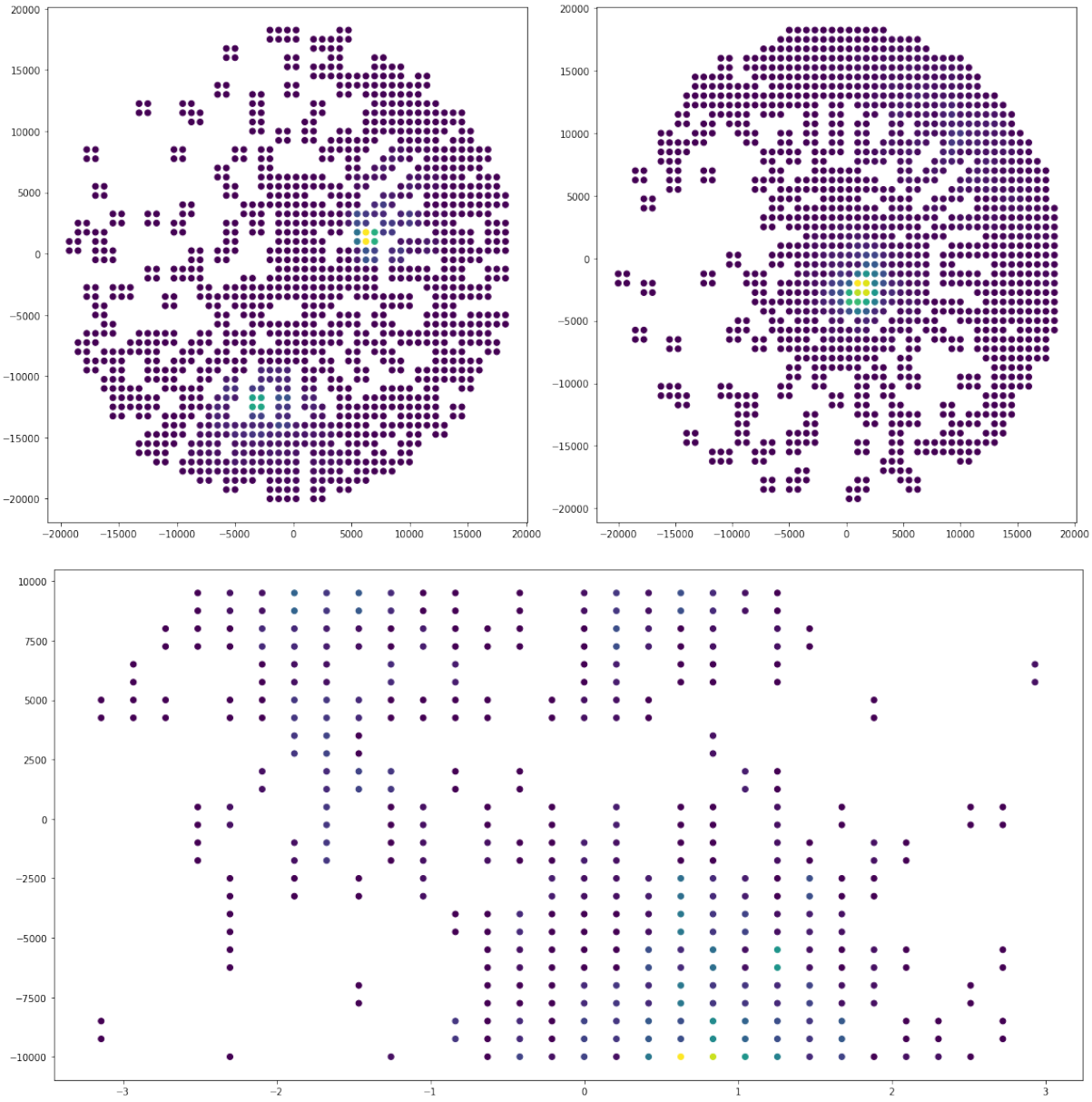

Рисунок 6 – Разделение сферы ФЭУ водного бассейна на части для независимого поиска максимумов на полюсах

Таблица 2 – сравнительные характеристики для метода интегрирования

Характеристика	**Результат**	**Единица измерения**
Время работы	19,62	с/40 изображений
Количество верно найденных максимумов	43	шт. из 60
Количество ненайденных максимумов	17	шт. из 60
Количество неверно найденных максимумов	25	шт.

2.4. Поиск максимума по определению

После получения неудовлетворительных результатов было решено проводить поиск максимума по определению: M_0 называется точкой локального максимума функции f, если существует проколотая окрестность точки M_0 такая, что для любого M $f(M) \leq f(M_0)$.

Алгоритм работы, в данном случае, следующий:

1. Отбрасываются все ФЭУ, в которых зарегистрировано слишком мало фотонов. Это позволит не тратить лишнее время на проверку заведомо неверных ФЭУ.
2. Для каждого из оставшихся ФЭУ проверяем условие, что на небольшом расстоянии от текущего ФЭУ, все зарегистрировали не больше фотонов. Если данное условие выполнено, то без дополнительных проверок можем считать, что нашли максимум энерговыделения.

```
def sph_max(points):
    """Функция получения ФЭУ с максимальным количеством з
арегистрированных частиц"""
    sph_max = (0, 0, 0, 0)
    for x, y, z, t in points:
        if t > sph_max[3]:
            sph_max = (x, y, z, t)
    return sph_max

def sph_cond(points, cond):
    """Функция получения ФЭУ по условию на количество
    зарегистрированных частиц"""
    sph_cond = []
    for x, y, z, t in points:
        if t > cond:
            sph_cond.append((x, y, z, t))
    return sph_cond
```

```
def dist(x1, y1, z1, x2, y2, z2):
    """Дистанция между точками"""
    return pow(pow(x1 - x2, 2)
               +pow(y1 - y2, 2)
               +pow(z1 - z2, 2), 1/2)

def sph_coord(points, x1, y1, z1):
    """Находит все точки, лежащие в радиусе 3м
    от точки с координатами x1, y1, z1"""
    sph_cond = []
    for x2, y2, z2, t2 in points:
        if dist(x1, y1, z1, x2, y2, z2) < 3000:
            sph_cond.append((x2, y2, z2, t2))
    return sph_cond

_, _, _, max_t = sph_max(points_dec)
points = sph_cond(points, max_t/6)

for x, y, z, t in points:
    if sph_max(sph_coord(points, x, y, z)) == (x, y, z, t
): io_dec.append((x, y, z, t))
```

Сложность данного метода составляет $O(n^2)$, однако из-за ограничения количества ФЭУ мы можем точно рассчитать максимальное время работы подпрограммы. Время работы также сокращается за счет отбрасывания ФЭУ по условию слишком маленького количества зарегистрированных частиц и отсутствия перевода между координатными пространствами.

Результат выполнения данного алгоритма представлен на Рисунок 7. Сравнительные характеристики представлены в таблице 3.

Таблица 3 – сравнительные характеристики для поиска экстремума по определению

Характеристика	**Результат**	**Единица измерения**
Время работы	7,46	с/40 изображений
Количество верно найденных максимумов	56	шт. из 60
Количество ненайденных максимумов	4	шт. из 60
Количество неверно найденных максимумов	6	шт.

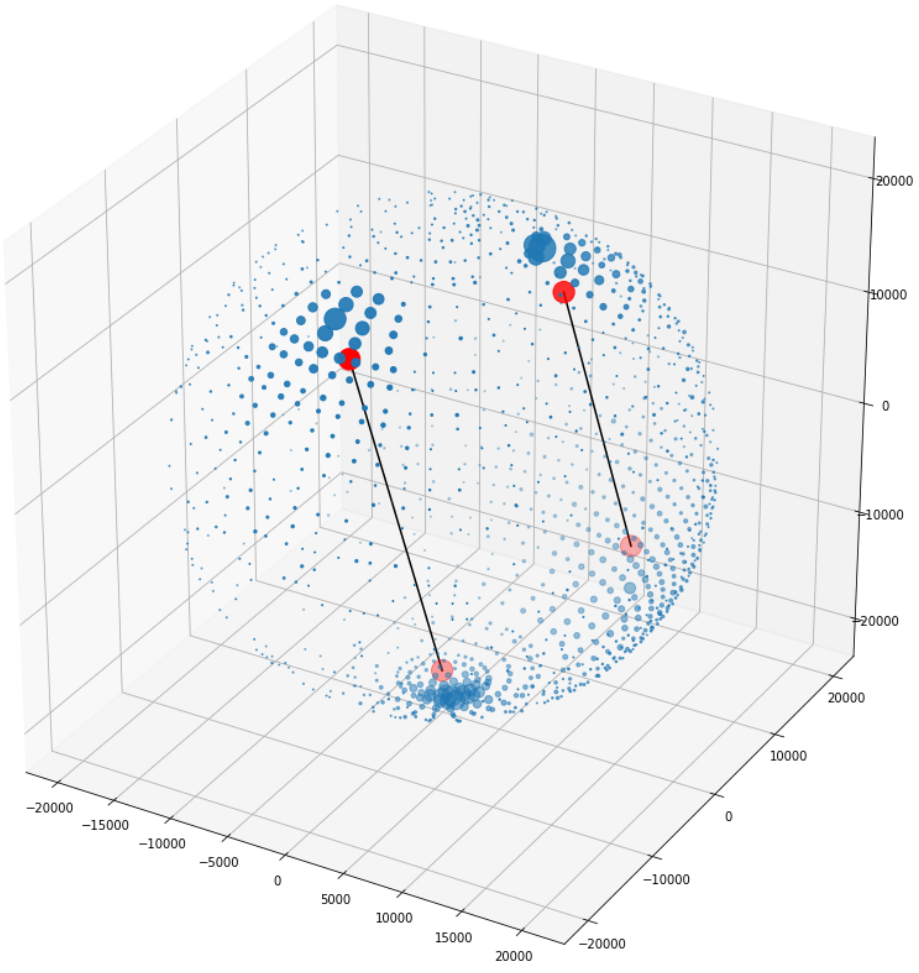

Рисунок 7 – результат поиска максимумов по определению

Плюсами данного метода являются высокая скорость работы, определение точки входа мюона с точностью до ближайшего детектора, независимость от относительной разницы в количестве зарегистрированных фотоэлектронов. Этот метод также показал самое большое количество определенных точек входа/выхода мюона из центральной сферы. Для дальнейшей разработки было принято решение выбрать именно этот метод.

3. Программный продукт

Для дальнейшей разработки выбран язык C++ 11 как основной язык коллаборации JUNO. Этот язык используется в связке с набором библиотек и утилит CERN ROOT, упрощающим обработку научных данных. Также C++ доказал свою устойчивость в работе высоконагруженных систем как в других экспериментах в физике высоких энергий, так и в индустрии разработки.

Задачей было интегрировать в уже существующий код, производящий поиск мюонов в центральном детекторе, свой метод для поиска мюонов в водном бассейне и сопоставление результатов этих операций.

Процедура регистрации мюона в исходном коде была следующая:

- симуляция калибровочных мюонов проходящих сквозь детектор на разном расстоянии от центра и построение отклика детектора на прохождение мюонного трека в зависимости от этого расстояния;
- получение отклика детектора: суммарного количества зарегистрированных фотоэлектронов на протяжении 300 нс с первой зарегистрированной частицы;
- расчет функции отклика детектора в виде сферических функций;
- для реконструкции мюонного трека производится вращение полученной функцию отклика и подбор расстояния до тех пор, пока она не будет максимально совпадать с откликом детектора на искомый мюонный трек.

Для того, чтобы поворот опорных мюонов производить не случайным образом, а заранее задать начальные условия поиска, находятся максимумы энерговыделения, и опорные треки поворачиваются на уже известный угол. Эта операция особенно важна для реконструкции нескольких мюонных треков в одном событии, так как дополнительно дает информацию о количестве мюонов.

На полученных точках строятся все возможные комбинации треков, из них отбираются наиболее перспективные и далее уже происходит сравнение смоделированных событий с найденными. Данная процедура дает многократный

выигрыш в скорости, однако при недостаточной точности алгоритма поиска максимумов могут быть потеряны некоторые мюонные треки.

Код должен вносить изменение в алгоритм отбора перспективных мюонных треков. Для сокращения количества всевозможных комбинаций необходимо дополнительное подтверждение мюонных треков основываясь на информации о найденных точках входа и выхода в водном бассейне.

Код поиска максимума, переписанный на язык C++, представлен в листинге в приложении 1.

После исполнения функции `GetIOpointsWP(...)` мы получаем вектор найденных в водном бассейне максимумов. Далее эти максимумы используются в функции `GetConfirmedMuonTracks(...)` (приложение 2) Эта функция проверяет условие попадания на трек, составленный по максимумам центрального детектора, максимума, найденного в водном бассейне. По результатам исполнения этой функции получаем 2 новые категории объектов: подтвержденные мюонные треки (`muonTrack`) и неиспользованные максимумы в водном бассейне (`unusedWPioPoints`).

Неиспользованные максимумы в водном бассейне, наряду с неиспользованными максимумами в центральном детекторе далее используются как опорные точки для поиска дополнительных треков.

ЗАКЛЮЧЕНИЕ

Разработанный алгоритм позволил добиться значительных результатов в оптимизации алгоритма поиска мюонных треков. Скорость реконструкции для однотрековых событий выросла на 36%, для двухтрековых – на 52%, для трёхтрековых – на 87%.

Выигрыш во времени для однотрековых событий появился из-за отказа от использования подпрограммы поиска дополнительного максимума, если находилось нечетное количество опорных точек для построения треков. Такое часто происходило при прохождении мюонного трека близко к краю детектора. В текущей версии за опорную точку принимается неиспользованный максимум из водного бассейна.

Кроме того, максимумы, неиспользованные в подтверждении треков, полученных в центральном детекторе, были использованы как опорные точки для дальнейшего поиска треков, что позволило увеличить эффективность поиска. Эффективность реконструкции для двухтрековых событий выросла на 7,4%, для трёхтрековых – на 34,7%.

Далее (Рисунок 8) представлены сведения о качестве реконструкции. Слева на рисунках показано разрешение для расстояния – расстояние между двумя точками на искомом и найденном мюонных треках, к которым проведены перпендикуляры из центра детектора. Справа на рисунках показано разрешение для угла – угол между искомым и найденным мюонными треками.

Плюсом данного метода является возможность легко распараллелить его с помощью технологии OpenMP, что еще сильнее сократит время работы приложения при работе на многопроцессорных системах.

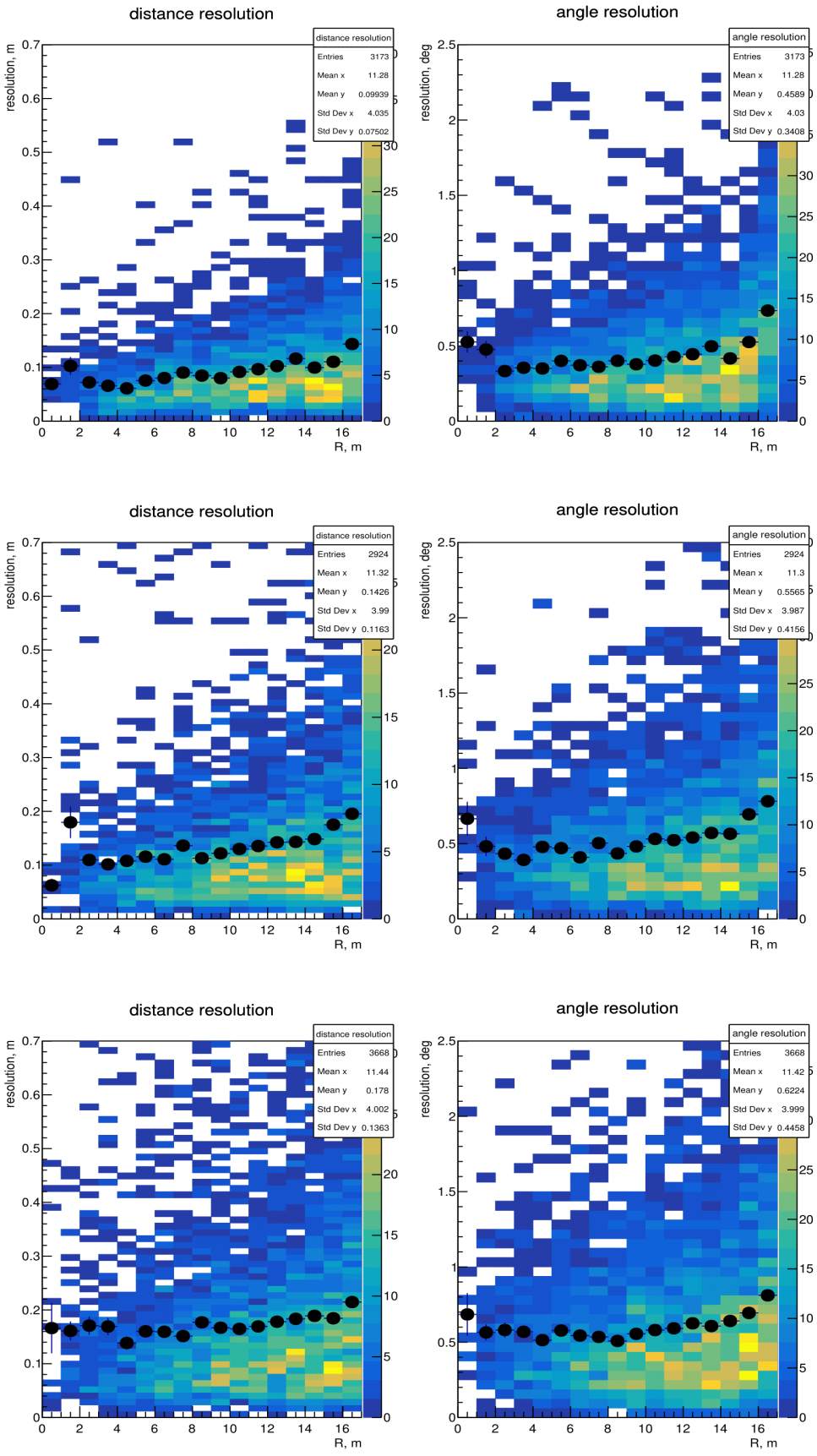

Рисунок 8 – разрешения реконструкции для расстояния (слева) и угла (справа) в случаях одного (сверху), двух (по центру) и трех (снизу) мюонных треков в зависимости от расстояния до центра детектора

СПИСОК ЛИТЕРАТУРЫ

Muon reconstruction with a geometrical model in JUNO [Journal] / auth. C. Genster M. Schever, L. Ludhova, M. Soiron, A. Stahlb and C. Wiebuschb // Journal of Instrumentation. - 2018. - Vol. 13.

Muon track reconstruction with help of the spherical function [Report] / auth. Chuckanov A.. - [б.м.] : JINR, 2019.

Neutrino Physics with JUNO [Journal] / auth. JUNO collaboration // Journal of Physics G: Nuclear and Particle Physics. - 2016. - Vol. 43.

[В Интернете] // European Organization for Nuclear Research. - http://cern.ch/.

[В Интернете] // The Jiangmen Underground Neutrino Observatory (JUNO). - http://juno.ihep.cas.cn/.

ПРИЛОЖЕНИЯ

Приложение 1. Листинг программы поиска максимумов

```
#define WPCHECK_SEARCHRADIUS 3000.0
#define WPCHECK_BACKGROUNDDIVIDER 6.0
#define WPCHECK_INNER_RADIUS 17.70
#define WPCHECK_OUTER_RADIUS 20.70
#define WPCHECK_CONDITION 3.00
#define sqr(x) (x)*(x)

std::pair<int, WPsignal_t> MuonCandidates::GetPMTmax(cons
t std::map<int, WPsignal_t> &fWPsignal)
{
    pair<int, WPsignal_t> maxval;

    for (auto &iPmt : fWPsignal)
    {
        if (iPmt.second.nPE > maxval.second.nPE)
            maxval = iPmt;
    }
    return maxval;
}

std::map<int, WPsignal_t> MuonCandidates::GetPMTcondition
(const std::map<int, WPsignal_t> &fWPsignal, int c_max)
{
    map<int, WPsignal_t> result;

    for (auto &iPmt : fWPsignal)
    {
        if (iPmt.second.nPE > c_max)
        {
            result.insert(iPmt);
        }
    }

    return result;
}
```

```
void MuonCandidates::GetIOpointsWP
    (std::vector<TVector3> &WPioPoints)
{
    WPioPoints.clear();

    map<int, WPsignal_t> maxCandidates = GetPMTcondition(
fWPsignal, GetPMTmax(fWPsignal).second.nPE / WPCHECK_BACK
GROUNDDIVIDER);
    map<int, bool> is_maximums;

    for (auto &MCandidate : maxCandidates)
        is_maximums.insert(pair<int, bool>(MCandidate.fir
st, true));

    for (auto &MCandidate : maxCandidates)
    {
        for (auto &iPmt : fWPsignal)
        {
            if (sqr(MCandidate.second.pmtCenter.X() - iPm
t.second.pmtCenter.X()) +
                                sqr(MCandidate.second.pmtCent
er.Y() - iPmt.second.pmtCenter.Y()) +
                                sqr(MCandidate.second.pmtCent
er.Z() - iPmt.second.pmtCenter.Z()) <=
                        sqr(WPCHECK_SEARCHRADIUS))
            {
                if (MCandidate.second.nPE < iPmt.second.n
PE)
                    is_maximums[MCandidate.first] = false
;
            }
        }
    }

    TVector3 v1;

    for (auto &MCandidate : maxCandidates)
    {
        if (is_maximums[MCandidate.first])
        {

            v1.SetMagThetaPhi(WPCHECK_OUTER_RADIUS, MCand
idate.second.pmtCenter.Theta(),
                                                MCandidat
e.second.pmtCenter.Phi());
```

```
                WPioPoints.push_back(v1);
            }
        }

        // return maxCandidatesResult;
}

#undef sqr
```

Приложение 2. Листинг функции проверки мюонов

```
Void MuonCandidates::GetConfirmedMuonTracks
(std::vector<track_t> &muonTrack, std::vector<TVector3> &
CDioPoints, std::vector<TVector3> &WPioPoints, std::vecto
r<TVector3> &unusedWPioPoints)
{
    // find muon tracks in CD confirmed by WP

    muonTrack.clear();

    if (WPioPoints.empty())
        return;

    vector<bool> wpUsage(WPioPoints.size());

    for (auto iPoint1 = std::begin(CDioPoints); iPoint1 !
= std::end(CDioPoints);
            ++iPoint1)
        for (auto iPoint2 = std::next(iPoint1); iPoint2 !
= std::end(CDioPoints);
                ++iPoint2)
        {

            // skip tracks with high angle

            double theta = ((*iPoint1) - (*iPoint2)).Thet
a();
            if (theta > fMaxSlope && pi - theta > fMaxSlo
pe)
                continue;

            int n_wp = 0;
            bool confirmedTrack = false;

            for (auto &wpPoint : WPioPoints)
            {

                // Distance between point WP point and CD
track

                double dist = ((wpPoint - (*iPoint1)).Cro
ss((*iPoint2) - (*iPoint1))).Mag() / ((*iPoint2) - (*iPoi
nt1)).Mag();
```

```
                if (dist < WPCHECK_CONDITION)
                {
                    wpUsage[n_wp] = true;
                    confirmedTrack = true;
                }

                ++n_wp;
            }

            if (confirmedTrack)
            {

                if (iPoint1->z() > iPoint2->z())
                    muonTrack.push_back(std::make_pair(*i
Point1, *iPoint2));
                else
                    muonTrack.push_back(std::make_pair(*i
Point2, *iPoint1));
            }
        }

    // fill unused WP points

    for (size_t i = 0; i < wpUsage.size(); ++i)
        if (!wpUsage[i])
            unusedWPioPoints.push_back(WPioPoints[i]);
}
```