МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СМОЛЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Физико-математический факультет
Кафедра прикладной математики и информатики

КУРСОВАЯ РАБОТА

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЙ В ФИЗИКЕ ВЫСОКИХ ЭНЕРГИЙ С ИСПОЛЬЗОВАНИЕМ PYTHIA И АНАЛИЗ РЕЗУЛЬТАТОВ В СРЕДЕ ROOT

Выполнил:
студент 4 курса физико-
математического факультета
направления подготовки
«Педагогическое образование
с двумя профилями
(Физика и Информатика)»
Тверской Егор Андреевич

Научный руководитель:
кандидат педагогических наук,
доцент кафедры информатики
Козлов Сергей Валерьевич

Смоленск
2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Моделирование в научных исследованиях стало использоваться уже на ранних этапах развития науки и постепенно стало стандартной процедурой в области научного познания мира. Естественным является желание «увидеть невидимое», заглянуть в протекающие физические явления, попытаться увидеть механизм, даже тогда, когда он скрыт от непосредственного восприятия. В настоящее время эту задачу успешно решают компьютерные технологии, а именно компьютерное моделирование, позволяет создать и провести «виртуальные» эксперименты, на основе компьютерной (математической) модели воспроизвести (испытать) реальность.

Методы компьютерного моделирования появились в физике в конце 50-х - начале 60-х годов. Основной из них – метод Монте - Карло. Развитие методов машинного моделирования оказало сильное влияние на физику, так как впервые появилась возможность теоретически исследовать системы с достаточно сложным взаимодействием частиц друг с другом. Сегодня эти методы успешно применяют в физике высоких энергий [11].

В настоящее время, эксперимент в физике частиц, практически всегда, представляет собой многократное измерение совокупности случайных процессов – взаимодействия частиц и прохождения их через детектор. Основным методом решения подобной задачи является вышеупомянутый метод Монте-Карло, который помогает в подсчёте результатов таких измерений. Благодаря современному оборудованию удаётся избежать главного недостатка данного метода, который заключается в потреблении большой вычислительной мощности.

В настоящее время наука шагнула так далеко, что для получения новых знаний, нужны гораздо более сложные и дорогие установки – коллайдеры, синхротроны и др. Однако такие установки являются очень дорогостоящими и не общедоступными. Чего нельзя сказать про высокопроизводительные и недорогие компьютерные системы, на которых можно рассчитать все

возможные риски и предвидеть поведение частиц в той или иной интересующей нас среде, попытаться предвосхитить результат срабатывания детектора включая ил выключая различные процессы [12].

Для выполнения подобных задач были разработаны многофункциональные пакеты программ, например, одним из таких является черновский программный комплекс ROOT и программа моделирования процессов столкновения элементарных частиц при высоких энергиях на ускорителях элементарных частиц методом Монте-Карло – Pythia.

Интерес к данной тематике и явился основным движущим фактором к данной теме курсовой работы. Такова природа человека, что жить в мире явлений, не прибегая к попыткам их объяснения неприемлемо для Homo sapiens. Несомненно, задача моделирования объектов и явлений еще долгое время будет оставаться актуальным в различных областях человеческой деятельности.

Целью курсовой работы является моделирование взаимодействия протонов с энергией 14 ТэВ в системе центра масс в рамках Квантовой хромодинамики КХД/QCD. Встречные пучки протонов используются на современных коллайдерах для изучения структуры адронов и поиска Новой физики на ускорителях. Конкретная цель работы состоит в демонстрации возможности Pythia8 – как средства моделирования и программного решения ROOT – как среда анализа данных, получаемых моделировании и эксперименте.

Объектом исследования является средство моделирования Pythia8 и среда анализа данных ROOT.

Предметом исследования является моделирование протонов с энергией 14 ТэВ в системе центра масс в рамках КХД.

Для достижения указанной цели поставлены следующие *задачи*:

- Обзор понятия математическое моделирование и приведение классификаций математических моделей.
- Математическое моделирование в физике высоких энергий.

- Обзор возможностей ROOT и Pythia8
- Демонстрация возможностей изученных сред на практическом примере.

Раскрытие темы курсовой работы проведено по следующему плану: анализ понятий «моделирование» и «модель», приведение классификаций моделей и математического моделирования, построение математической модели (имитационное моделирование), метод Монте-Карло, изучение программ ROOT и Pythia (среда ROOT, гистограммы, история создания Pythia, назначение и философия программ моделирования, структура программы Pythia) и практическая часть, в которой рассмотрено моделирование взаимодействия в физике высоких энергий.

ГЛАВА 1. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

1.1. Понятие «моделирование» и «модель»

Роль моделирования в социальном и техническом прогрессе общества непрерывно растет. В настоящее время на идее моделирования по существу базируется любой метод научных исследований – как теоретический (при котором используются различного рода знаковые, абстрактные модели), так и экспериментальный (использующий физические модели). По мере развития методологии познания и расширения сферы активной деятельности человека моделирование находит все новые применения, все более разнообразными становятся его формы и способы реализации; совершенствуются технические средства моделирования. В настоящее время с моделированием, в тех или иных его проявлениях, явно или неявно связаны и научные исследования в естественных, технических и гуманитарных областях знания, и работы инженеров по проектированию, конструированию, экспериментальным исследованиям и испытаниям новых технических систем и деятельность руководителя по принятию инженерных и управленческих решений.

Методологическая основа моделирования заключается в следующем. Все то, на что направлена человеческая деятельность, называется объектом. Выработка методологии направлена на упорядочение получения и обработки информации об объектах, которые существуют вне нашего сознания и взаимодействуют между собой и внешней средой. Объект – некоторая часть окружающего мира, рассматриваемого человеком как единое целое. Каждый объект имеет имя и обладает параметрами. Параметр – признак или величина, характеризующая какое-либо свойство объекта и принимающая различные значения.

В научных исследованиях большую роль играют гипотезы, то есть определенные предсказания, основывающиеся на небольшом количестве опытных данных, наблюдений, догадок. Быстрая и полная проверка гипотез может быть проведена в ходе специально поставленного эксперимента. При

формулировании и проверки правильности гипотез большое значение в качестве метода суждений имеет аналогия.

Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводится к удобным для исследования логическим схемам. Такие логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие природу явлений, называются моделями. Другими словами, модель – это объект заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

Моделированием называется замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели. Таким образом, моделирование может быть определено как представление объекта моделью для получения информации об этом объекте путем проведения экспериментов с его моделью.

Модель вместо исходного объекта используется в случаях, когда эксперимент опасен, дорог, происходит в неудобном масштабе пространства и времени (например, долговременен или, наоборот, слишком кратковременен), невозможен, неповторим, ненагляден и т. д.

Модель, таким образом – это естественный или искусственный объект, находящийся в соответствии и изучаемым объектом или, точнее, с какой-либо из его сторон [10].

1.2 Классификация моделей

При анализе поведения объекта-оригинала формируется мысленный его образ или идеальная модель, называемая когнитивной.

Представление когнитивной модели на естественном языке называют содержательной моделью. В зависимости от целей модели классифицируются на описательные, объяснительные и прогностические.

Описательной моделью можно назвать любое описание объекта.

Объяснительная модель должна обеспечить объяснение причин нахождения системы в текущем состоянии.

Прогностическая модель должна обеспечивать понимание поведения объекта в будущем.

Концептуальной моделью принято называть содержательную модель, при формулировке которой используются понятия и представления предметных областей знания, занимающихся изучением объекта моделирования.

Выделяют три вида концептуальных моделей: логико-семантические, структурно-функциональные и причинно-следственные.

Логико-семантическая модель – модель с описанием объекта в терминах и определениях соответствующих предметных областей.

Структурно-функциональная модель – модель рассмотрения объекта как единого целого, с последующим изучением его отдельных элементов или подсистем.

Причинно-следственная модель – модель, применяемая для объяснения и прогнозирования поведения объекта.

Формальная модель является представлением концептуальной модели с помощью одного или нескольких формальных языков.

Информационная модель – модель, содержащая автоматизированные справочники, реализованные с помощью систем управления базами данных.

Формальная классификация моделей основывается на классификации используемых математических средств.

Классификация по способу представления объекта.

Наряду с формальной классификацией, модели различаются по способу представления объекта:

- структурные;
- функциональные.

Структурные модели представляют объект как систему со своим устройством и механизмом функционирования.

Функциональные модели не используют таких представлений и отражают только внешне воспринимаемое поведение (функционирование) объекта. В их предельном выражении они называются также моделями «черного ящика». Возможны также комбинированные типы моделей, которые иногда называют моделями «серого ящика».

Модели, замещающие технологический объект, в зависимости от типа образа разделяют на три вида: абстрактные, аналоговые и физические.

Абстрактные модели основываются на возможности описания технического объекта (системы) на языке символов, принятом в той или иной области науки путем отвлечения от несуществующих признаков. Абстрактные модели могут быть математическими и нематематическими. Процесс исследования технического объекта с помощью абстрактной модели включает три этапа:

- построение описательной модели процесса, которая должна отвечать на вопросы «что происходит», «почему так происходит», «при каких условиях это возможно», «что может произойти при изменении данных параметров и внешних условий»;
- запись информативной модели с помощью определенной системы символов;
- исследование функционирования созданной абстрактной модели различными методами анализа, большинство из которых опирается на математический анализ.

Аналоговые модели основаны на подобии явлений, имеющих различную физическую природу, но описываемых одинаковыми математическими уравнениями. Подобие математического описания этих процессов позволяет экспериментально и теоретически подтверждать результаты, полученные в одной области, соответствующими результатами из другой. Примерами аналоговых моделей могут служить электрические и механические колебания.

Физические модели имеют ту же физическую природу, что и исследуемый объект, и применяются в тех случаях, когда трудно провести

испытания реальных объектов в реальных условиях. В химической технологии применяют физические модели – уменьшенные копии реальных аппаратов и технологических процессов (различают лабораторные установки и так называемые пилотные). При использовании результатов необходимо учитывать эффект масштабирования [14].

1.3 Математическое моделирование

Математической моделью называется совокупность уравнений или других математических соотношений, отражающих основные свойства изучаемого объекта или явления в рамках принятой умозрительной физической модели и особенности его взаимодействия с окружающей средой на пространственно-временных границах области его локализации. Математические модели различных процессов в континуальных системах строятся, как правило, на языке дифференциальных уравнений, позволяющих наиболее точно описать состояние процесса в любой точке пространства в произвольный момент времени. Основными свойствами математических моделей являются адекватность и простота, указывающие на степень соответствия модели изучаемому объекту и возможности ее реализации. Процесс формулировки математической модели называется постановкой задачи.

Под математическим моделированием можно понимать процесс построения и изучения математических моделей. Развернутое определение: математическое моделирование – это идеальное научное знаковое формальное моделирование, при котором описание объекта осуществляется на языке математики, а исследование модели проводится с использованием тех или иных математических методов.

Математическая модель описывает зависимость между исходными данными и искомыми величинами.

Элементами обобщенной математической модели являются:

- множество входных данных (переменные): первая переменная – совокупность варьируемых переменных; вторая переменная – независимые константы;
- математический оператор, определяющий операции над этими данными, под которым понимается полная система математических операций, описывающих численные или логические соотношения между множествами входных и выходных данных;
- множество выходных данных: представляет собой совокупность критериальных функций, включающую целевую функцию.

Математическая модель является математическим аналогом проектируемого объекта. Степень адекватности ее объекту определяется постановкой и корректностью решений задачи проектирования.

Множество варьируемых параметров образует пространство поиска, которое является метрическим с размерностью, равной числу варьируемых параметров.

Множество независимых переменных образуют метрическое пространство входных данных. В том случае, когда каждый компонент пространства задается диапазоном возможных значений, множество независимых переменных отображается некоторым ограниченным подпространством пространства.

Множество независимых переменных определяет среду функционирования объекта, т. е. внешние условия, в которых будет работать проектируемый объект.

Математическая модель никогда не бывает тождественна рассматриваемому объекту и не передает всех его свойств и особенностей. Она является лишь приближенным описанием объекта и носит всегда приближенный характер. Точность соответствия определяется степенью соответствия адекватности модели и объекта.

При построении математической модели приходится выдвигать дополнительные предположения – гипотезы. Модель поэтому еще называют

гипотетической. Основным критерием применимости модели является эксперимент. Критерий практики позволяет сравнивать гипотетические модели и выбирать из них наиболее подходящую.

Каждый объект описывается ограниченным числом моделей или их систем. Процесс моделирования значительно легче реализуется при использовании унификации математических моделей, т. е. использования наборов готовых моделей. Существует возможность переноса готовых моделей из одних процессов на другие, идентичные, аналогичные. Аналогичными называют объекты и процессы, описываемые одинаковыми по форме уравнениями, содержащими различные физические величины и параметры, связанные между собой одинаковыми операторами. Величины, которые в аналогичных уравнениях стоят на одинаковых местах, называют аналогами.

Математическая модель описывает реальный объект с каким-то приближением. Степень соответствия описания реальному процессу определяется полнотой учета возмущающих воздействий. При отсутствии или незначительности возмущений, действующих как внутри, так и вне объекта, можно однозначно определить влияние входных и управляющих параметров на выходные [14].

1.4 Построение математической модели.

Построение математических моделей является достаточно трудным процессом, включающим большие затраты материальных и временных ресурсов, а также предполагает необходимость в специалистах высокого уровня с компетенциями как в предметной области, так и в таких областях, как прикладная математика, численные методы, программирование, современные вычислительные системы.

Среди этапов процесса построения моделей можно выделить следующие:

1. Обследование объекта моделирования и формулировка технического задания на разработку модели.

Конструирование модели начинается со словесно-смыслового описания объекта или явления. Данная стадия содержит сведения общего характера о природе объекта, информацию о целях его исследования и некоторые предположения. Данный этап можно также назвать формулировкой предмодели. Цель этапа – разработка содержательной постановки задачи моделирования, т. е. создание совокупности вопросов об объекте моделирования, записанных в словесной форме.

2. Концептуальная и математическая постановка задачи.

На этом этапе происходит завершение идеализации объекта, отбрасываются несущественные факторы и эффекты. Цель концептуальной постановки задачи заключается в формулировке основных вопросов и наборе гипотез касательно свойств и поведения объекта моделирования в терминологии специальных дисциплин. В итоге предположения описываются математически для количественного анализа их выполнения. На этапе составления математического описания предварительно выделяют основные явления и элементы в объекте и затем устанавливают связи между ними. Далее для каждого выделенного элемента и явления записывают уравнение, отражающее его функционирование. Кроме того, в математическое описание включают уравнения связи между различными выделенными явлениями. В зависимости от процесса математическое описание может быть представлено в виде системы алгебраических, дифференциальных уравнений. Процесс получения совокупности математических уравнений, однозначно описывающих объект моделирования, называется математической постановкой задачи моделирования.

3. Качественный анализ и проверка корректности модели. Для контроля правильности полученной системы математических соотношений требуется проведение ряда обязательных проверок:

- контроль размерности;
- контроль порядков;
- контроль характера зависимостей;

- контроль экстремальных ситуаций;
- контроль граничных условий;
- контроль физического смысла;
- контроль математической замкнутости.

Понятие «корректность модели» очень важно, особенно в прикладной математике, поскольку невозможно применение численных методов к некорректно поставленным задачам. Установить корректность математической задачи является сложной задачей. Для обеспечения корректности математической модели должны быть выполнены все контрольные проверки.

4. Выбор и обоснование выбора методов решения задачи. Созданная модель исследуется любыми возможными методами, в том числе с взаимной проверкой. Поскольку не все модели решаются теоретически, в последнее время широко используются вычислительные методы. Данное обстоятельство важно при анализе нелинейных объектов, поскольку качественное поведение таких объектов неизвестно. В зависимости от метода решения задачи все методы подразделяются на:

Аналитические: данные методы являются подходящими для анализа результатов, однако они применимы только для относительно простых моделей. При наличии аналитического решения задачи численное решение практически не применяется;

Алгоритмические: для алгоритмических методов реализуется вычислительный эксперимент с использованием компьютера.

Этап выбора метода решения и разработки моделирующей программы подразумевает выбор наиболее эффективного (по быстроте получения решения и его наибольшей точности) метода решения из имеющихся методов, реализацию его в форме алгоритма решения.

5. Поиск решения или реализация алгоритма в виде программ для ЭВМ. Данный этап будет рассмотрен при описании вычислительного эксперимента.

6. Проверка адекватности модели. На данном этапе определяется соответствие объекту и сформулированным предположениям. При этом также

выполняется исследование модели на достижение поставленной цели любыми способами, например, сравнение с экспериментом или сопоставление с другими подходами. Модель необходимо отбросить или модифицировать в случае получения с ее помощью результата, существенно отличающегося от истинного. Этап установления степени соответствия модели объекту является заключительным. Для проверки адекватности математической модели реальному процессу нужно сравнить результаты измерений на объекте в ходе процесса с результатами предсказания модели в идентичных условиях.

7. Практическое использование модели. Независимо от области применения созданной модели необходимо провести качественный и количественный анализ результатов моделирования, который позволяет:

- выполнить модификацию рассматриваемого объекта, найти его оптимальные характеристики;
- обозначить область применения модели;
- проверить обоснованность гипотез, принятых на этапе математической постановки, оценить возможность упрощения модели с целью повышения ее эффективности при сохранении требуемой точности;
- показать, в каком направлении следует развивать модель в дальнейшем [14].

1.4.1 Имитационное моделирование

Имитационное моделирование – метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

Существует класс объектов, для которых по различным причинам не разработаны аналитические модели, либо не разработаны методы решения

полученной модели. В этом случае математическая модель заменяется имитатором или имитационной моделью.

Имитационная модель – логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

Имитационную модель можно рассматривать как множество правил, которые определяют, в какое состояние система перейдет в будущем из заданного текущего состояния.

К имитационному моделированию прибегают, когда:

- дорого или невозможно экспериментировать на реальном объекте;
- невозможно построить аналитическую модель: в системе есть время, причинные связи, последствие, нелинейности, стохастические (случайные) переменные;
- необходимо имитировать поведение системы во времени.

В имитационном моделировании различают два метода:

- метод статистического моделирования;
- метод статистических испытаний (Монте-Карло) [14].

1.5 Метод Монте-Карло

При существовании теоретического описания метода на протяжении длительного периода времени метод Монте-Карло получил широкое распространение только с появлением ЭВМ, т. е. задача генерации и использования в расчетах случайных величин достаточно трудоемкая задача.

Метод Монте-Карло – общее название группы численных методов, основанных на получении большого числа реализаций стохастического (случайного) процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи. Название метода происходит от одноименного города в княжестве Монако, где развита игорная индустрия, поскольку наиболее

простым механическим устройством для генерации случайных величин является рулетка.

Сущность метода Монте-Карло состоит в следующем: требуется найти значение a некоторой изучаемой величины. Для этого выбирают такую случайную величину *X*, математическое ожидание которой равно: *M(X) =* a. Практически же поступают так: производят *n* испытаний, в результате которых получают n возможных значений *X*; вычисляют их среднее арифметическое $\bar{x} = \frac{\sum x_i}{n}$ и принимают x в качестве оценки (приближенного значения) a^*искомого числа a:

$$a \simeq a^* = \bar{x}\,.$$

Поскольку метод Монте–Карло требует проведения большого числа испытаний, его часто называют методом статистических испытаний. Теория этого метода указывает, как наиболее целесообразно выбрать случайную величину *X*, как найти ее возможные значения. В частности, разрабатываются способы уменьшения дисперсии используемых случайных величин, в результате чего уменьшается ошибка.

Метод Монте-Карло применяется очень часто, порой некритично и неэффективным образом. Он имеет некоторые очевидные преимущества:

1. Он не требует никаких предложений о регулярности, за исключением квадратичной интегрируемости. Это может быть полезным, так как часто случайная величина – очень сложная функция, чьи свойства регулярности трудно установить.

2. Он приводит к выполнимой процедуре даже в многомерном случае, когда численное интегрирование неприменимо, например, при числе измерений, большем 10.

3. Его легко применять при малых ограничениях или без предварительного анализа задачи.

Он обладает, однако, некоторыми недостатками, а именно:

1. Границы ошибки не определены точно, но включают некую случайность. Это, однако, более психологическая, чем реальная, трудность.

2. Статическая погрешность убывает медленно.

3. Необходимость иметь случайные числа.

Применение метода Монте-Карло к моделированию физических процессов.

Суть решения физических задач методом Монте-Карло заключается в том, что физическому явлению сопоставляется имитирующий вероятностный процесс, отражающий его динамику.

Затем этот процесс реализуется с помощью набора случайных чисел. Интересующие нас значения физических величин находятся усреднением по множеству реализаций моделируемого процесса.

Основным преимуществом метода Монте-Карло по сравнению с классическими численными методами состоит в том, что с его помощью можно исследовать физические явления практически любой сложности, которые иначе решить просто невозможно. Например, решить уравнения, описывающие взаимодействие двух атомов, будет сравнительно несложно, однако решить такую же задачу для сотни атомов уже не реально. Кроме того, для метода Монте-Карло часто характерна простая структура вычислительного алгоритма. Как правило, составляется программа для осуществления одного случайного испытания (шага модели). Затем это испытание повторяется необходимое число раз, причем каждый последующий шаг не зависит от всех остальных [14].

ГЛАВА 2. ПРОГРАММЫ ROOT И PYTHIA

2.1 Среда ROOT

Многофункциональный пакет программ ROOT был создан для анализа экспериментальных данных и их визуализации в физике высоких энергий. Авторами этого программного продукта является группа физиков: Рене Бран (ЦЕРН), Фон Рademакер (ФНАЛ), Масахару Готу (Япония), Валерий Файн (ЦЕРН), Ненад Банкик, Филипп Кэнал, Сюзанна Панасек (все ФНАЛ) и др. Являясь разработчиками таких известных пакетов, как PAW, PIAF и GEANT, Р. Бран и Ф. Рademакер пришли к выводу, что дальнейшее развитие программного обеспечения для обработки данных физических экспериментов ввиду усложнения их структуры требует внедрения объектно-ориентированных технологий программирования. Поэтому для написания ROOT был использован язык программирования общего назначения C++. Разработчики новой системы поставили перед собой задачи поддержки полного цикла анализа данных, управления комплексными структурами со сложной иерархией объектов и сохранения таких ранее использовавшихся методов анализа (например, в PAW), как гистограммирование, фитирование и визуализация. Эти задачи были успешно решены. Современные эксперименты в области физики высоких энергий характеризуются большой статистикой и большой множественностью событий. На стадии анализа ROOT позволяет выделить из массива данных только интересующие исследователя события и тем самым существенно упростить их обработку. Поэтому в настоящее время ROOT является одним из самых популярных пакетов прикладных программ, использующихся в физических лабораториях всего мира [15].

ROOT – пакет объектно-ориентированных программ и библиотек, разработанных в Европейском центре ядерных исследований [5].

ROOT был разработан в контексте эксперимента NA49 в ЦЕРНе. NA49 произвел впечатляющее количество данных, около 10 терабайт на запуск. Эта скорость обеспечила идеальную среду для разработки и тестирования следующего поколения анализа данных.

ROOT был и продолжает развиваться как продукт с открытым кодом. Это означает либеральный, неформальный стиль разработки, который в значительной степени зависит от разнообразных и глубоких талантов сообщества пользователей. В результате физики разработали ROOT для себя; это сделало его конкретным, уместным, полезным, со временем усовершенствованным. Развитие ROOT – это постоянный диалог пользователей и разработчиков, границы между которыми со временем стерлись, и пользователи становятся соавторами. На основе ROOT (Церн) создаются решения для других экспериментов и установок, например, MPDROOT для коллайдера НИКА в Дубне [2, 7].

Программный пакет ROOT хорошо документирован. Информационный узел ROOT [6] своевременно обновляется. Исходный текст ROOT позволяет легко генерировать документацию, и каждый класс имеет в сети свою собственную страницу, содержащую описание класса и объяснение каждого метода [15].

Он предоставляет кроссплатформенный интерфейс к графической подсистеме и операционной системе используя механизмы абстракции данных. Частями абстрактной платформы являются:

- графический интерфейс пользователя,
- графический интерфейс разработки,
- классы-контейнеры,
- система средств самоизменения программ,
- скриптовый язык на основе C/C++,
- командный интерпретатор (CINT),
- система сериализации объектов,
- система долговременного сохранения данных (persistence).

Пакеты, включённые в ROOT, содержат:

- средства для создания гистограмм и графиков функций для визуализации и анализа вероятностных распределений и функций;

- средства «подгонки» (фитирования) теоретических кривых под экспериментальные данные и минимизации функций (для подборки наиболее простой зависимости, описывающей экспериментальные данные);
- инструменты статистического анализа;
- инструменты матричной алгебры;
- средства для четырёхвекторных вычислений (четырёхмерное пространство Минковского удобно применяется в физике высоких энергий);
- стандартные математические функции;
- инструменты многовариантного анализа данных, то есть использования нейронных сетей;
- средства обработки изображений, используемые, например, для анализа астрономических снимков;
- средства доступа к распределённым данным (в контексте таблиц баз данных);
- инструменты распределённых вычислений, параллелизации обработки данных;
- средства сериализации и долговременного сохранения объектов;
- инструменты доступа к базам данных;
- средства геометрической 3D-визуализации;
- инструменты для создания файлов в различных графических форматах, таких как PostScript, JPEG, SVG;
- двусторонние интерфейсы к языкам Python и Ruby (возможности использования средств ROOT из кода на Python или Ruby и использование модулей, написанных на Python или Ruby, из ROOT);
- интерфейсы к Монте-Карло-генераторам событий физики элементарных частиц.

Ключевой возможностью пакета ROOT является специальный контейнер данных, называемый деревом (Tree), вместе с его подмножествами ветвями (Branch) и листьями (Leaf). Дерево может быть представлено как удобное

средство чтения и записи данных в файле. Следующий элемент данных, записанный в файле, может быть получен инкрементированием индекса дерева. Такой подход позволяет избежать проблем с выделением памяти при создании объектов, и даёт возможность дереву выступать в качестве «лёгкого» контейнера при буферизации данных.

ROOT разрабатывался как высокопроизводительная вычислительная библиотека, необходимая для обработки данных Большого Адронного Коллайдера, поток которых достигает нескольких петабайт в год. С 2009 года ROOT используется в подавляющем большинстве экспериментов физики высоких энергий; абсолютное большинство современных результатов и иллюстраций в этой области науки получено именно с использованием ROOT.

Включение в пакет интерпретатора C++ CINT значительно увеличило гибкость пакета, так как позволило использовать средства ROOT в интерактивном режиме или посредством написания скриптов, что сделало его похожим на MATLAB.

Основная критика ROOT связана с утверждениями о том, что для начинающих пользователей достаточно сложно освоить этот продукт, его широкие возможности и средства. Периодически эти проблемы обсуждаются пользователями и разработчиками ROOT в специальном списке рассылки [5].

2.2 Графическое представление данных в виде гистограмм

Гистограммы – это основной графический объект, возможности работы с которым предоставляет ROOT. Гистограмма является графическим объектом, предназначенным для отображения распределений количества физических событий по интересующим параметрам этих событий. Экспериментальные данные отображаются на гистограмме следующим образом. Допустим, было набрано 1000 событий и, необходимо установить, как распределились эти события по некоему параметру или группе параметров (например, по суммарной энергии зарегистрированных в единичном событии частиц). Тогда нужно создать гистограмму, в которой по оси X будет располагаться величина

этой суммарной энергии, а то оси Y количество распределенных событий, такая гистограмма называется одномерной. При создании гистограммы необходимо определить, какова будет область определения этой энергии (например, от 0 до 5 ТэВ) и на какое количество частей эта область будет разделена. Такие части называются бинами гистограммы; если, например, определено, что бинов будет 50, это означает, что на гистограмме будет отображено 50 экспериментальных точек. Если определить, что бины должны быть равной ширины, то первый бин будет со значения 0 ТэВ и заканчиваться значением 0.1 ТэВ, второй бин будет 0.1-0.2 ТэВ и т.д. При необходимости можно создавать гистограммы и с разной шириной бинов. Итак, если из 1000 событий в диапазон первого бина попало, например, 9 штук, то первая отображаемая точка гистограммы будет иметь координаты (0.0, 9), если в диапазон второго бина попало 18 событий, то координата второй точки (0.1, 18) и т.д. Для разного типа ожидаемых распределений и разных объемов выборки существуют схемы подбора оптимального количества бинов (интервалов разбиения). Поскольку в физике высоких энергий речь идет о выборках 10^4 – 10^{10} элементов в выборке, то число бинов определяется десятками и сотнями, часто число бинов определяется количеством каналов в регистрирующей аппаратуре.

Таким образом, гистограмма предназначена для работы со статистическими данными, и важным отличием её от других графиков является возможность заполнять её постепенно, по мере того, как новые статистические данные становятся доступны. Гистограммы широко используются для представления, часто, в режиме реального состояния значений параметров установок, экранов, мониторов и т.п. В практической части работы гистограммы будут использованы для визуализации модельных данных [9].

2.3 История программы Pythia

PYTHIA – программа моделирования процессов столкновения элементарных частиц при высоких энергиях на ускорителях элементарных частиц методом Монте-Карло [4]. Родоначальникамим (создателями) Pythia

считается Ханс-Уно Бенгссон, Бо Андерсон, Гёст Густафсон университет г. Лунд, Швеция. Наиболее полное и исчерпывающее руководство пользователя было подготовлено группой шведских физиков под руководством Т. Съестранда. Программный продукт Pythia в своей идеологии базируется на проверенных временем теоретических представлениях о взаимодействии элементарных частиц и экспериментальных данных, которые задают границы свободных параметров имплементированных в коде моделей. В настоящее время самое большое сообщество пользователей можно найти среди экспериментаторов LHC (БАК), RHIC (БНЛ) и ряда других институтов и лабораторий специализирующихся на физике частиц, также программа используется для множества других феноменологических или экспериментальных исследований. Основные задачи, выполняемые программой, включают исследование экспериментальных данных и теоретической модели, разработка поисковых стратегий, интерпретация экспериментальных данных, а также изучение работы детекторов. Таким образом, он охватывает всю жизнь эксперимента, от первоначальной концепции (идеи эксперимента) до окончательного представления данных.

Разработка Jetset (основного элемента Pythia) началась в 1978 году, и многие из его компонентов были позже включены в Pythia. Таким образом, текущий генератор Pythia – это продукт более 35 лет разработки.

Первоначально код программы был написан на Fortran 77. Pythia 8.100 была первым полным выпуском полностью переписанной на C++. Мотивация перехода на C++ связана с широким кругом возможностей, которые открывает разработчику объектное программирование. Таким образом, некоторые важные функции долгое время отсутствовали, и код еще не был протестирован и настроен на тот же уровень зрелости, что и Pythia 6.4., последняя версия старого поколения на Fortran. С тех пор отсутствующие функции были добавлены, новые функции разработаны и исправлены старые ошибки. Накопление опыта экспериментальным сообществом происходит медленно, LHC-коллаборации (эксперименты на Большом Адронном коллайдере

представляют собой сообщества физиков с разных континентов объединенных установкой ATALS, ALISA, LHCb на коллайдере), в частности, уже совершили или находятся в процессе совершения полномасштабного перехода от Pythia6 к Pythia 8. В настоящее время эти продукты развиваются самостоятельно.

Разработка Pythia 8.1 после первоначального выпуска была непрерывным процессом, и обратная несовместимость появлялась лишь в нескольких случаях. Чтобы ввести дополнительный набор незначительных возможностей, в частности, удалить некоторые устаревшие функции, была выпущена Pythia 8.200. Большинство пользовательских программ должны работать без изменений или требовать минимальных корректировок.

С одной стороны, Pythia задумывалась автономной, пригодной для любого количества независимых физических исследований. С другой стороны, постоянная тенденция заключается в том, что Pythia8 связана с другими программными пакетами. Это достигается через ряд интерфейсов, с Les Houches Accord (LHA) и связанные с ним Les Houches Event Files(LHEF) являются ярким примером. Таким образом, вычисления на основе матричных элементов (ME) основанные на расчетах из различного количества источников можно комбинировать со специальностями Pythia, такими как начальное состояние излучения (ISR), излучения в конечном состоянии (FSR), многопартонные взаимодействия (MPI) и струнная фрагментация [8].

2.4 Назначение и философия программ моделирования

Назначение программ моделирования физических событий:

- Дают физикам представление о типе событий, которые они надеются увидеть, и об их скорости набора.
- Помогают в планировании новых детекторных установок, то есть, оптимизировать их характеристики для изучения интересующих сценариев физических событий в рамках существующих ограничений.
- Являются инструментом для проработки стратегии анализа данных (оптимизации отношения «сигнал/шум»).

- Используются в качестве метода оценки коррекций на геометрические и кинематические ограничения области чувствительности (acceptance) детекторов.
- Используются в качестве удобной рабочей оболочки для интерпретации наблюдаемых феноменов в терминах Стандартной Модели.

Квантовая механика вносит концепцию случайности в поведение физических процессов. Достоинством генераторов событий (event generators) является то, что эта случайность может быть смоделирована при помощи метода Монте-Карло. Сущность метода заключается в том, что, во-первых, подразумевается наличие генератора псевдослучайных чисел, т.е. функции, которая при вызове возвращает число R в пределах от 0 до 1, при этом распределение R является плоским, и с достаточной точностью значения R являются некоррелированными. Затем эти значения R используется для розыгрыша сценария конкретного цикла события (выбор конкретного значения для различных известных распределений величин, выбор времени распада и т.п.) Разыграв статистически достаточное количество событий, мы можем построить интересующие нас распределения (например, диапазон энергий для продуктов интересующего механизма реакции).

Что касается упомянутых известных распределений, то, например, дифференциальное сечение реакции рассчитывается из кинематических соотношений при введении матричного элемента (известного, либо предложенного теоретиками исходя из перспективных моделей), для учета высших порядков КХД вводится значение дополнительного параметра (K-множителя).

Следует заметить, что при генерации значительного числа событий (миллионов), становится актуальной проблема скоррелированости псевдослучайных чисел, и вместо встроенных в C++ Random генераторов приходится применять специально созданные программы [9].

2.5 Структура программы Pythia

С точки зрения описания физики событий полная процедура генерации события разделяется на 3 стадии:

1.Генерация «процесса», который определяет природу события. Зачастую это могут быть «жесткие процессы», такие как $gg{-}{>}\, h^0 \to ZZ \to m^+m^-qq_{bar}$ (а также другие процессы), которые могут быть просчитаны в рамках теории малых возмущений. Для получения исчерпывающего описания возможностей Pythia рекомендуем обратиться к ресурсу [13].

2.Генерация всех включенных процессов на партонном (кварковом) уровне, включая гамма-излучение, многократное партонные взаимодействия и структуру непровзаимодействовавшего пучка. Такие феномены приблизительно описываются теорией малых возмущений, однако непертурбативные поправки уже существенны.

3.Адронизация этой партонной (кварковой) конфигурации (фрагментация струй, распады нестабильных частиц). Только феноменологическое описание.

Этим стадиям отвечают три класса ProcessLevel, PartonLevel и HaronLevel, соответственно. Классы: Event (члены класса process и event), BeamParticles, база данных Settings.

С точки зрения технического устройства, взаимодействия пользователя и генератора проявляется в трех фазах:

1.Инициализация, когда формулируется задание.

2.Генерация индивидуального события (цикл события).

3.Вывод окончательной статистики.

Программа содержит теорию и модели для ряда аспектов физики, включая так называемые мягкие и жесткие взаимодействия, распределения партонов (кварков), партонные струи начального и конечного состояний, многократные партонные взаимодействия, фрагментацию и распады.

Встроенные C++ методы программы обеспечивают доступ к информации как об отдельной частице либо процессе на любом этапе розыгрыша события,

так и о событии в целом. Встроенные средства вывода позволяют получить статистическую информацию и гистограммы в виде ASCII кода (который можно сохранить в файл для дальнейшего использования) [9].

ГЛАВА 3. МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ В ФИЗИКЕ ВЫСОКИХ ЭНЕРГИЙ

3.1 Реализация проекта

Практическая часть данной курсовой работы состояла в освоении программного обеспечения, используемого для моделирования и анализа данных.

Существующая современная политика в области использования программных продуктов, авторских прав, привела к тому, что все программные элементы пришлось развернуть на платформе Линукс, а именно на Scientific Linux 7. Данный дистрибутив операционной системы Linux, который создан совместными усилиями Fermilab и CERN при поддержке различных лабораторий и университетов со всего мира. Его исходной целью было стремление уменьшить дублирование усилий лабораторий и иметь общую инсталляционную базу для различных экспериментов и других научно-исследовательских проектов. Конечно же, в работе не преследовалось цель проводить вычисления, требующие серьезные компьютерные мощности.

Для демонстрационных целей Scientific Linux 7 был установлен в качестве «гостевой» операционной системы с использованием VirtualBox6. VirtualBox (Oracle VM VirtualBox) – программный продукт виртуализации для операционных систем Microsoft Windows, Linux, FreeBSD, macOS, Solaris/OpenSolaris, ReactOS, DOS и других. Программа была создана компанией Innotek с использованием исходного кода Qemu. Первая публично доступная версия VirtualBox появилась 15 января 2007 года. В феврале 2008 года Innotek был приобретён компанией Sun Microsystems, модель распространения VirtualBox при этом не изменилась. В январе 2010 года Sun Microsystems была поглощена корпорацией Oracle, модель распространения осталась прежней [3].

Но вернемся к дистрибутиву Scientific Linux 7. Набор программных продуктов и имеющихся в дистрибутиве библиотек для поддержки физических

вычислений позволяет относительно просто провести настройку среды для разработки. Дополнительно пришлось установить cmake [1], а с его помощью собрать библиотеки Pythia8 и собрать ROOT версии 6.x для запуска программы (скрипта) моделирования.

Для запуска программы (скрипта) моделирования можно воспользоваться Терминалом Linux. После запуска Терминала необходимо прописать команды для запуска необходимых файлов и программ (см. рис. 1):

ls – команда показать список файлов в текущем каталоге.

source scr.sh – запуск заготовленного скрипта.

cd works – переход в каталог works (каталог где лежит программа).

root – запуск среды ROOT.

.x pythia82.C – запуск программы.

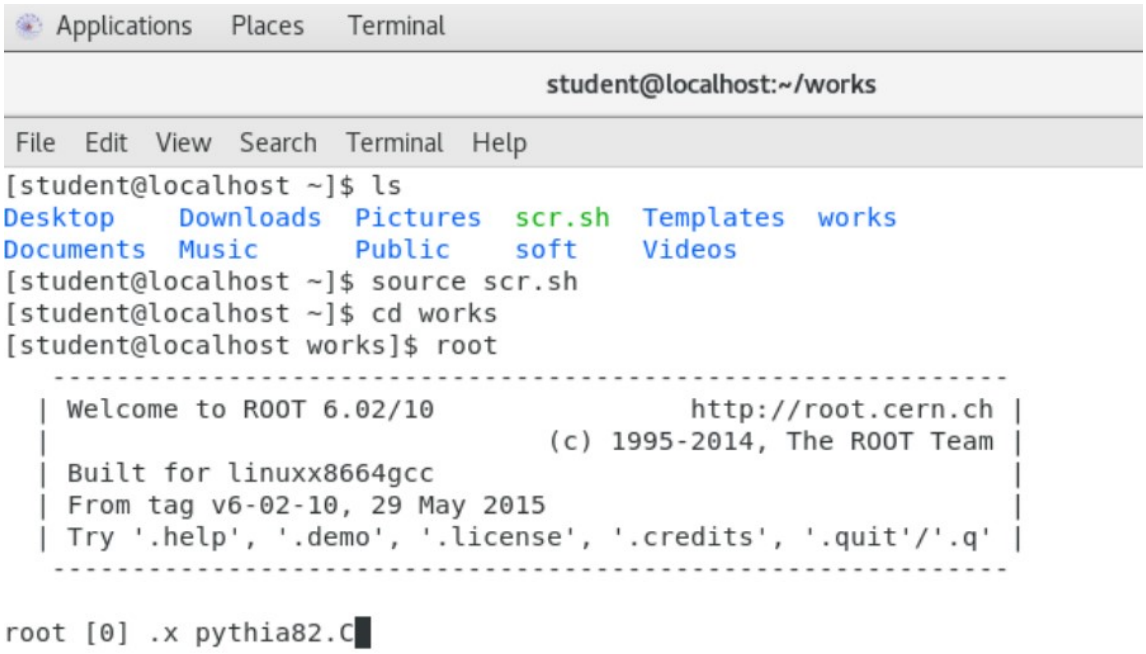

Рисунок 1.

На всех этапах работы над курсовой работой оказывалась поддержка и консультация специалистов из ОИЯИ. Помимо работы над данным проектом был получен бесценный опыт обучения взаимодействия с научным

коллективом, к сожалению, из-за реалий настоящего времени в удаленном формате.

3.2 Пример моделирования

Описываемый ниже пример призван продемонстрировать возможности Pythia8+ROOT для моделирования взаимодействия элементарных частиц, и не претендует на роль подробного исчерпывающего описания по данным программным продуктам. Моделирование в физике частиц является отдельной компетенцией, достаточно сложной, для освоения в сжатые сроки, поэтому описание физики будет далее минимальным. За основу скрипта взят набор команд в котором позволяется провести моделирование, пример из библиотеки примеров ROOT ($ROOTSYS/tutorials/pythia8).

Физическая теория, на основе которой строится моделирование – Квантовая хромодинамика (КХД/QCD). КХД/QCD – это теория, описывающая сильные взаимодействия между кварками и глюонами, фундаментальными частицами, составляющими составные адроны, такие как протон, нейтрон и пион (π+, -, 0). КХД – это тип квантовой теории поля, называемый неабелевой калибровочной теорией, с группой симметрии SU (3). Аналог электрического заряда КХД – это свойство, называемое цветом. Глюоны являются носителем силы теории, подобно фотонам для электромагнитной силы в квантовой электродинамике. Теория является важной частью Стандартной модели физики элементарных частиц. За прошедшие годы было собрано большое количество экспериментальных доказательств КХД. Подробное описание КХД выходит за рамки данной работы. Используя в перечне инструкций флаг HardQCD: all=on подключается список процессов, позволяющих провести моделирование взаимодействия двух сталкивающихся в коллайдере (речь не идет о какой-то конкретной установке) протонов с энергиями 7 ТэВ. Фактически моделируется столкновение двух встречных пучков протонов без привязки к установке в реакции p+p→X, где X – может быть, как набором промежуточных виртуальных состояний, так и конкретных, рожденных частиц. Далее, в

качестве частиц, которые будут отбираться для анализа, будут выступать π+ и π- (положительно и отрицательно заряженные пионы). С точки зрения КХД, пион (π+) представляет $u\bar{d}$ - кварковое состояние (pdg = 211), а π- $d\bar{u}$ - кварковое состояние (pdg = -211), и, в данном случае, будут являться продуктами взаимодействия сталкивающихся протонов в реакции p+p→X. Pythia8 позволяет пользователю подключать/выключать широкий спектр процессов и состояний используемых для описания сильных и электрослабых взаимодействий в физике высоких энергий и элементарных частиц, давая возможность пользователю подобрать оптимальный набор, исходя из имеющихся задач, возможностей установки, и теоретических воззрений, с целью получения искусственного набора данных. Полный перечень процессов и флагов для их подключения приведено в руководстве пользователя [13].

Схема эксперимента приведена на рисунке 2. Столкновению подвергаются два встречных пучка протонов.

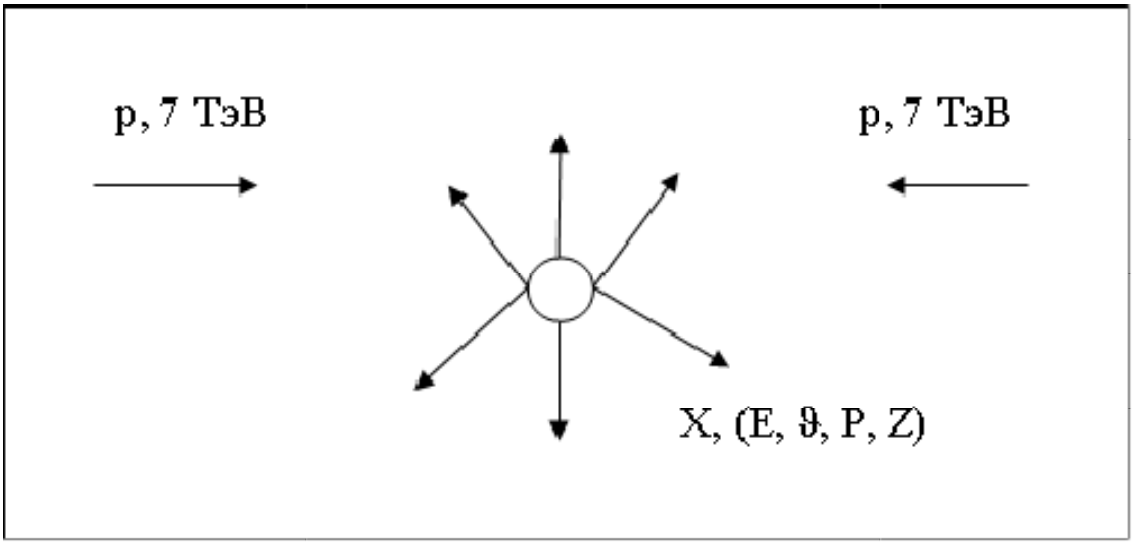

Рисунок 2 Схема эксперимента.

Подробно рассмотрим этапы запуска моделирования. В данном случае генератор Pythia8 вызывается командным интерпретатором ROOT (фактически генератор является плагином среды ROOT). Для работы в данном режиме, среда ROOT распространяемая как в виде готового к работе релиза под Linux/Windows, так и в виде исходного кода должна быть скомпилирована (настроена в случае релиза) для использования генератора Pythia8.

Сам скрипт реализован в виде функции void pythia82(Int_t nev = 1000, Int_t ndeb = 1) не возвращающей значения (void) с двумя переменными nev = 1000 (количество испытаний) и ndeb = 1 – переменной счетчика.

Первая часть кода скрипта pythia82.C содержит необходимые процедуры инициализации переменных для работы Pythia8 в качестве плагина ROOT:

```
const char *p8dataenv = gSystem->Getenv("PYTHIA8DATA");
  if (!p8dataenv) {
    const char *p8env = gSystem->Getenv("PYTHIA8");
    if (!p8env) {
      Error ("pythia82.C",
          "Environment variable PYTHIA8 must contain path to pythia directory!");
      return;
    }
    TString p8d = p8env;
    p8d += "/xmldoc";
    gSystem->Setenv ("PYTHIA8DATA", p8d);
  }
  const char* path = gSystem->ExpandPathName ("$PYTHIA8DATA");
  if (gSystem->AccessPathName(path)) {
      Error ("pythia8.C",
          "Environment variable PYTHIA8DATA must contain path to $PYTHIA8/xmldoc directory!");
    return;
  }
  gSystem->Load("libEG");
  gSystem->Load("libEGPythia8");
```

Далее следует описание гистограмм, создаются экземпляры классов для последующего вывода данных моделирования в графическом виде:

```
  TH1F* ek = new TH1F ("ek", "Total energy", 120, 0, 50.);
```

```
TH1F* pth = new TH1F ("pth", " Theta ", 36, 0., 3.1415926);
TH2F* ekth = new TH2F ("ekth", "Et, theta", 120, 0.,50., 36, 0., 3.1415926);
```

Создается экземпляр класса TClonesArray для копирования данных и последующего доступа к ним из ROOT при помощи синтаксиса C++. Также создается экземпляр класса TPythia8.

```
// Array of particles
TClonesArray* particles = new TClonesArray ("TParticle", 1000);
// Create pythia8 object
TPythia8* pythia8 = new TPythia8();
```

Инициализируется набор физический процессов посредством «установки флага» в режим "HardQCD: all = on" и инициализируется процесс моделирования взаимодействия протона с протоном с энергией столкновения в системе центра масс 14 ТэВ:

```
// Configure
pythia8->ReadString("HardQCD:all = on");
// Initialize
pythia8->Initialize (2212 /* p */, 2212 /* p */, 14000. /* TeV */);
```

Запускается цикл моделирования для числа событий nev:

```
for (Int_t iev = 0; iev < nev; iev++) {
  pythia8->GenerateEvent();
```

if (iev < ndeb) pythia8->EventListing(); - листинг первого события выводится на экран.

Результат моделирования «копируется» в экземпляр класса TCloneArrays и задается переменная-итератор:

```
pythia8->ImportParticles(particles,"All");
Int_t np = particles->GetEntriesFast();
```

Производится перебор всех частиц в сгенерированном событии:

```
// Particle loop
for (Int_t ip = 0; ip < np; ip++) {
  TParticle* part = (TParticle*) particles->At(ip);
```

```
Int_t ist = part->GetStatusCode();
// Positive codes are final particles.
if (ist <= 0) continue; //промежуточные состояния не рассматриваются
```

Осуществляется доступ лишь к финальным частицам в моделировании (ist>0) к пионам, каонам и т.п. по средствам PDG-кода, кодировка от ParticleDataGroup [https://pdg.lbl.gov] применяемая в физике частиц:

```
Int_t pdg = part->GetPdgCode();
Float_t charge = TDatabasePDG::Instance()->GetParticle(pdg)->Charge();
if (charge == 0.) continue; нейтральные частицы опускаются (часто подобные частицы можно зафиксировать только по заряженным продуктам их распада в детектируемой области
if (pdg == 211 || pdg == -211){ для пионов π+ и π- берется полная энергия и полярный угол вылета
Double_t eka = part->Energy();
Double_t theta = part->Theta();
```

Промежуточный вывод данных на экран:

```
cout<<pdg<<"\t"<<eka<<"\t"<<theta<<endl;
```

Передача данных в гистограммы (заполнение гистограмм):

```
ek->Fill(eka);
pth->Fill(theta);
ekth->Fill (eka, theta);
}
}
} завершение циклов генерации и разбора событий.
```

Вывод статистики моделирования pythia8->PrintStatistics();

Построение гистограмм посредством создания экземпляра класса TCanvas и вызова метода Draw() для соответствующих гистограмм.

```
TCanvas* c1 = new TCanvas ("c1","Pythia8 test example, Et, theta ", 800, 800);
c1->Divide(1, 2);
c1->cd(1);
ek->Draw();
ek->SetXTitle("E_{total}, GeV");
ek->SetYTitle("couts");
c1->cd(2);
pth->Draw();
pth->SetXTitle("#theta, rad.");
pth->SetYTitle("counts");
TCanvas* c2 = new TCanvas ("c2","Pythia8 test example, Et, theta", 800, 800);
ekth->Draw();
ekth->SetXTitle("E_{total}, GeV");
ekth->SetYTitle("#theta, rad");
}
```

Результатом работы функции, посредством которой реализовано моделирование и обработка данных является набор гистограмм:

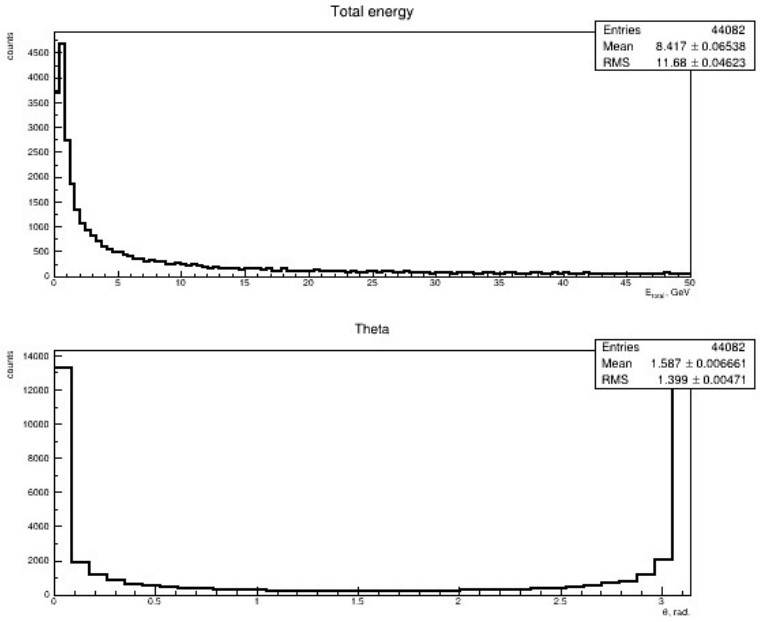

Рисунок 3. Распределение пионов, сгенерированных в событиях $p+p \rightarrow X$

X – набор сгенерированных моделированием по величине полной энергии – верхняя гистограмма, по величине угла вылета в полярной системе координат – нижняя гистограмма. Можно видеть, что подавляющая часть пионов по энергии лежит в диапазоне до 15 ГэВ, а вылетает в направлении близком к направлению движение первичных протонов вблизи углов 0 и 180 градусов. Несколько десятков пионов (π+-) рождается в одном столкновении двух протонов с энергией 14 ТэВ в системе центра масс (максимальная энергия БАК в Церне).

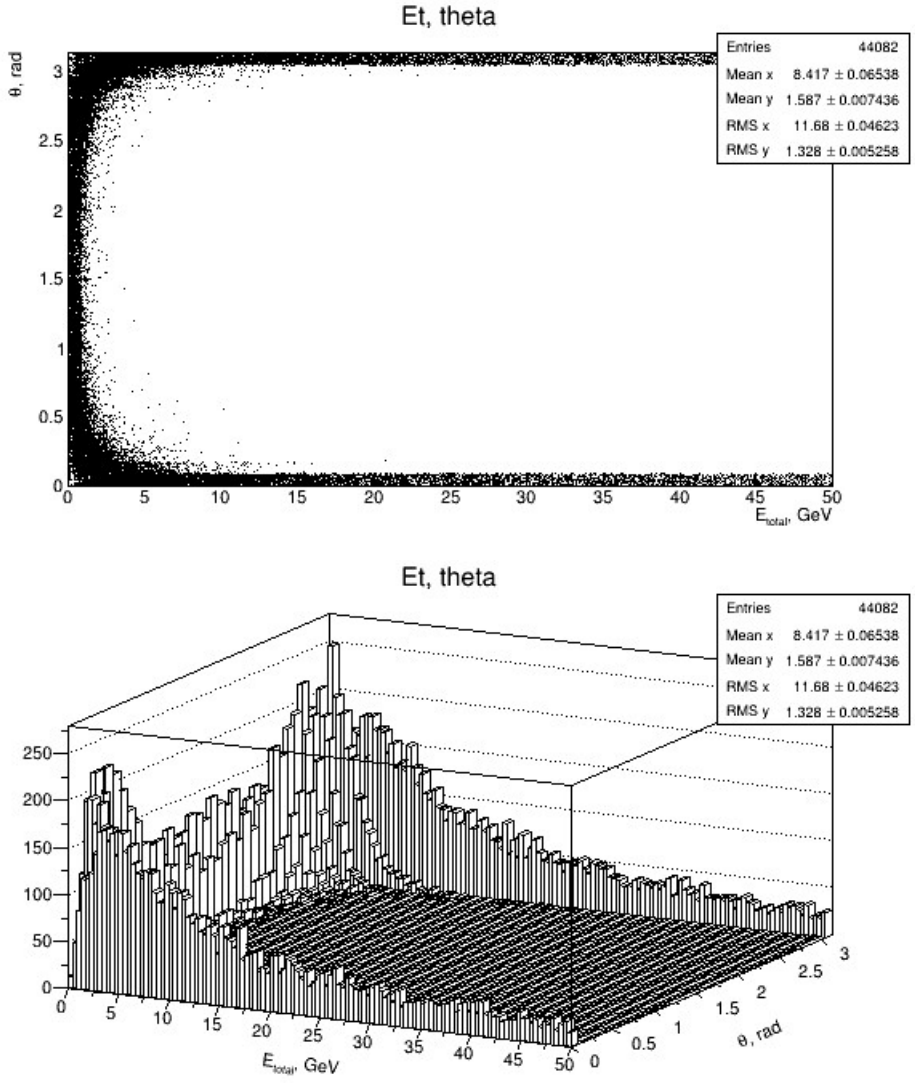

Рисунок 4. Распределение заряженных пионов по переменным энергия.

На рисунке 4 представлены 2D и 3D варианты демонстрирующие, что основную долю среди всех событий (1000 испытаний) составляют пионы, вылетающие вблизи 0 и 180 градусов при энергии до 15 ГэВ.

Данный пример призван продемонстрировать идею моделирования в физике частиц с использованием Pythia8 совместно со средой для анализа данных ROOT. Перечень переменных, связанных со свойствами рожденных частиц, доступных для анализа, конечно, не ограничивается энергией и углом вылета, а на основе первичных модельных данных могут быть оценены вторичные косвенные переменные, например, может быть проведено сравнение выхода пионов разной зарядности, соотношение пионов и каонов. Однако, подобные изыскания выходят за рамки данной курсовой работы.

Полный листинг работающего примера, pythia82.C

```
//Tverskoy Egor
// to run, do
// root > .x pythia82.C
//
// Note that before executing this script,
//     -the env variable PYTHIA8 must point to the pythia8100 (or newer) directory
//    -the env variable PYTHIA8DATA must be defined and it must point to $PYTHIA8/xmldoc
//
void pythia82(Int_t nev = 1000, Int_t ndeb = 1)
{
  const char *p8dataenv = gSystem->Getenv("PYTHIA8DATA");
  if (!p8dataenv) {
    const char *p8env = gSystem->Getenv("PYTHIA8");
    if (!p8env) {
      Error ("pythia8.C",
          "Environment variable PYTHIA8 must contain path to pythia directory!");
      return;
    }
```

```
    TString p8d = p8env;
    p8d += "/xmldoc";
    gSystem->Setenv("PYTHIA8DATA", p8d);
  }
  const char* path = gSystem->ExpandPathName("$PYTHIA8DATA");
  if (gSystem->AccessPathName(path)) {
      Error ("pythia8.C",
          "Environment variable PYTHIA8DATA must contain path to
$PYTHIA8/xmldoc directory !");
    return;
  }
// Load libraries
#ifndef G__WIN32 // Pythia8 is a static library on Windows
  gSystem->Load("$PYTHIA8/lib/libpythia8");
#endif
  gSystem->Load("libEG");
  gSystem->Load("libEGPythia8");
// Histograms
  TH1F* ek = new TH1F ("ek", "Total energy", 120, 0, 50.);
  TH1F* pth = new TH1F ("pth", " Theta ", 36, 0., 3.1415926);
  TH2F* ekth = new TH2F ("ekth", "Et, theta", 120, 0.,50., 36, 0., 3.1415926);
// Array of particles
  TClonesArray* particles = new TClonesArray ("TParticle", 1000);
// Create pythia8 object
  TPythia8* pythia8 = new TPythia8();
// Configure
  pythia8->ReadString ("HardQCD:all = on");
// Initialize
  pythia8->Initialize (2212 /* p */, 2212 /* p */, 14000. /* TeV */);
// Event loop
```

```
        for (Int_t iev = 0; iev < nev; iev++) {
          pythia8->GenerateEvent();
          if (iev < ndeb) pythia8->EventListing();
          pythia8->ImportParticles(particles,"All");
          Int_t np = particles->GetEntriesFast();
       // Particle loop
          for (Int_t ip = 0; ip < np; ip++) {
            TParticle* part = (TParticle*) particles->At(ip);
            Int_t ist = part->GetStatusCode();
            // Positive codes are final particles.
            if (ist <= 0) continue;
            Int_t pdg = part->GetPdgCode();
            Float_t charge = TDatabasePDG::Instance()->GetParticle(pdg)-
>Charge();
            if (charge == 0.) continue;
            if (pdg == 211 || pdg == -211) {
            Double_t eka = part->Energy();
            Double_t theta = part->Theta();
             cout<<pdg<<"\t"<<eka<<"\t"<<theta<<endl;
            ek->Fill(eka);
            pth->Fill(theta);
            ekth->Fill(eka, theta);
           }
          }
        }
        pythia8->PrintStatistics();
        TCanvas* c1 = new TCanvas ("c1","Pythia8 test example, Et, theta ", 800,
800);
        c1->Divide(1, 2);
        c1->cd(1);
```

```
        ek->Draw();
        ek->SetXTitle("E_{total}, GeV");
        ek->SetYTitle("couts");
        c1->cd(2);
        pth->Draw();
        pth->SetXTitle("#theta, rad.");
        pth->SetYTitle("counts");
        TCanvas* c2 = new TCanvas ("c2","Pythia8 test example, Et, theta", 800,
800);
        ekth->Draw();
        ekth->SetXTitle("E_{total}, GeV");
        ekth->SetYTitle("#theta, rad");
      }
```

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была проведена классификация моделей математического моделирования, рассмотрен метод Монте-Карло, изучены программы ROOT и Pythia. В качестве демонстрации возможностей данных сред выполнено моделирование взаимодействия протонов с энергией 14 ТэВ в системе центра масс в рамках КХД.

В результате исследования курсовой работы, поставленные задачи достигнуты, получены следующие результаты и выводы:

- Современные достижения в IT позволяют проводить подготовительные работы по подготовке к проведению эксперимента любой сложности. Для ряда отраслей науки разработаны и активно используются соответствующие программные решения, доступные для освоения даже неопытными пользователями.
- На примере, программного продукта ROOT, являющегося одним из стандартных современных решений для физики ядра и частиц, продемонстрирован уровень развития средств обработки данных.
- Дано описание возможностей средства моделирования Pythia8, применяемого как теоретиками, так и экспериментаторами в области физики частиц.
- Данная работа позволила ознакомиться с актуальными программными продуктами, применяемыми в области физики высоких энергий и физике частиц. В режиме удаленного взаимодействия с сотрудниками ОИЯИ (Объединенного института ядерных исследований, г. Дубна) реализовать практическую часть и получить представления об инструментах, средствах и методах в арсенале физика-исследователя.

СПИСОК ЛИТЕРАТУРЫ

1. CMake [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/CMake
2. MPD Эксперимент [Электронный ресурс] URL: http://mpd.jinr.ru
3. Oracle VM VirtualBox [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/VirtualBox
4. PYTHIA [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/PYTHIA
5. ROOT [Электронный ресурс] URL: https://root.cern/
6. ROOT [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ROOT
7. ROOT Data Analysis Framework. User's Guide, 2014 [Электронный ресурс] URL: http://web.ihep.su/spitsky/mipt/literature/src/mctools/ROOTUsersGuideA4.pdf
8. Stefan Ask, Jesper R. Christiansen, Richard Corkea, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestelf, Christine O. Rasmussen, Peter Z. Skands. An Introduction to PYTHIA 8.2. CERN-PH-TH-2014-190, 2014.
9. В.В. Леонтьев, И.И. Белотелов. Задачи раздела Информационные методы в физике высоких энергий. МГУ имени М.В. Ломоносова, научно-исследовательский институт ядерной физики имени Д.В. Скобельцына. Москва, 2011
10. Зайцева Н.А. Математическое моделирование. Учебное пособие. – М.: РУТ (МИИТ), 2017.
11. Компьютерное моделирование [Электронный ресурс] URL: https://clck.ru/VKKjS
12. Моделирование прохождения частиц через вещество средствами программного комплекса GEANT4 [Электронный ресурс] URL: https://clck.ru/VKKcM
13. Руководство пользователя [Электронный ресурс] URL: https://pythia.org/latest-manual/Welcome.html
14. С.В. Звонарев. Основы математического моделирования: учебное пособие / С. В. Звонарев. – Екатеринбург: Изд-во Урал. ун-та, 2019.

15. Т.М. Соловьева. Введение в объектно-ориентированный анализ на примере пакета ROOT. Учебное пособие. Учебно-методические пособия Учебно-научного центра ОИЯИ Дубна, 2003.