Д11-2001-175

Н. С. Амелин, М. Э. Комогоров

КОМПОНЕНТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К РАЗРАБОТКЕ И ИСПОЛЬЗОВАНИЮ ЧИСЛЕННЫХ МОДЕЛЕЙ В ФИЗИКЕ ВЫСОКИХ ЭНЕРГИЙ

СОДЕРЖАНИЕ

Введение

Полное описание релятивистских адронных и ядерных взаимодействий из первых принципов квантовой хромодинамики очень ограниченно. Как правило, мы можем получить предсказания КХД для взаимодействий с большими передачами импульса, идущих на малых расстояниях. Однако известно, что доминирующими процессами в адронных и ядерных взаимодействиях являются процессы с относительно малыми передачами импульсов. Это так называемые мягкие процессы взаимодействия. Таким образом, очень важной является задача разработки и применения КХД обоснованных моделей для предсказания, описания и анализа мягких процессов. Помимо этого, подготовка современных экспериментов в области физики высоких энергии, создание детекторов для этих экспериментов, которые дороги и сложны, а также проведение экспериментов требуют тщательного численного моделирования. Здесь так называемые адронные модели Монте-Карло или генераторы Монте-Карло (МК-генераторы) событий взаимодействия частиц и ядер, являются очень популярными. Это феноменологические модели с большим числом параметров, значения которых определяются путем сравнения результатов моделирования с экспериментальными данными.

Таким образом, феноменологические МК-адронные модели могут применяться как генераторы событий взаимодействий с целью изучения явлений при адронных и ядерных столкновениях или как источники информации об этих столкновениях с целью использования полученной информации [1].

МК-адронные модели - это сложные физические модели. Типичный МК-генератор событий использует сложные численные алгоритмы для описания многочисленных физических процессов. Значительные усилия требуются, чтобы сформулировать, программировать и тестировать такие модели.

Применение МК-генераторов для обработки полученных экспериментальных данных или при проектировании нового эксперимента также требует значительных усилий от пользователя. Большинство существующих модельных кодов в силу сложности их применения используются только их авторами. Даже для теоретиков, которые хорошо понимают физику данной модели, существуют значительные трудности, препятствующие использованию чужого кода, который бы помог в решении исследуемых проблем.

Объектно-ориентированный подход, основанный на использовании C++, может быть применен для разработки МК-кодов. Такой подход имеет много преимуществ по сравнению с традиционным процедурным программированием (см., например, [2]). Наиболее важным из них является наличие различных средства языка C++, дающих возможность определения новых типов данных и операций для этих типов. Такие типы данных (например, тип частица, тип 4-вектор), которые являются более «естественными», могут быть определены для конкретной предметной области, такой, как разработка МК-генераторов, и разработчик МК-генератора может также легко обращаться с новыми типами, как и с встроенными в C++ типами данных.

Необходимость в объектно-ориентированном фреймворке [3], в рамках которого можно создавать и использовать численные модели, обосновывается тем, что адронные МК-модели, а также их применения и разработки имеют много общего. Создание фреймворка ещё более оправданно в том случае, когда число разрабатываемых моделей велико и может возрасти в будущем.

Типичный фреймворк [3] является хорошо документированной коллекцией программного обеспечения, созданного с целью разработки связанных приложений. Он определяет основную архитектуру разрабатываемого приложения. Для того

чтобы быть полезным, фреймворк должен наилучшим образом обеспечивать необходимые свойства и функциональность разрабатываемых на его основе приложений и включать уже разработанные приложения. Он должен способствовать созданию легко модифицируемых и расширяемых приложений. Основная цель создания фреймворка состоит в его многократном использовании. Разработчик приложений должен иметь возможность использовать уже написанный код, например, библиотеки классов, входящих в фреймворк, а также саму архитектуру фреймворка. Архитектура любого фреймворка тесно связана с так называемыми «архитектурными образцами», которые широко применяются для конструирования и документирования определенных частей фреймворка.

«Архитектурные образцы» представляют собой найденные при разработке программного обеспечения успешные решения типичных проблем программирования, которые представляют общий интерес для большого числа программистов. В частности, так называемые «фабрики» и «заместители» (см. книгу [4]) были использованы при разработке предлагаемой системы.

Разработанный объектно-ориентированный фреймворк является центральной частью более широкой системы программ, в дальнейшем называемой NiMax. Основная идея этой системы состоит в том, что она должна поддерживать компонентный подход при разработке и использовании сложных численных моделей. Отдельный компонент может представлять собой программную реализацию какой-либо модели, например упругого рассеяния адронов, или модели сложного физического явления, каким является столкновение ультрарелятивистских тяжелых ионов. Компоненты, имеющие отношение к определенным областям их применения, группируются в прикладные модули. Предполагается, что данная система может быть той основой, с помощью которой можно разрабатывать библиотеку прикладных модулей из компонентов с различными численными алгоритмами, что может позволить в дальнейшем использовать эту библиотеку как хранилище «строительных блоков» для сборки сложных и реалистических физических моделей.

Данная система полезна для двух профессиональных категорий людей: это пользователи численных моделей и разработчики (продвинутые пользователи) численных моделей. Пользователем моделей считается тот, кому при работе с данным программным обеспечением или системой не требуется создавать или модифицировать код модели. Он работает с моделями посредством пользовательского интерфейса, то есть он не должен знать какой-либо язык программирования. Разработчик моделей работает с системой на уровне системных интерфейсов. Он должен знать основные концепции, которые лежат в основе системы, её структуру, её библиотеки и, конечно же, язык программирования C++.

Таким образом, в данном описании рассматриваются основные концепции компонентного подхода к разработке и применению численных моделей в физике высоких энергий, который реализован в виде системы NiMax. Эти концепции иллюстрируются многочисленными примерами работы пользователя данной системы. В приложении описаны также физика и численные алгоритмы созданных компонентов, входящих в модули адронных моделей, которые предназначены для моделирования событий взаимодействия частиц и ядер при высоких энергиях. Это описание можно рассматривать как первый вариант руководства для пользователей системы NiMax, которое мы предполагаем расширять и совершенствовать по мере развития самой системы.

Следует подчеркнуть, что созданная система может быть применена для решения очень широкого круга задач, где требуется разработка и использование

сложных численных моделей, и такая потребность не ограничивается разработкой и использованием компонентов МК-генераторов событий взаимодействий частиц и ядер для физики высоких энергий. Выбор разработанных и описанных здесь компонентов, с одной стороны, отражает профессиональные пристрастия авторов данной системы, а с другой, является, на наш взгляд, достаточно полным для демонстрации возможностей системы.

Данное руководство состоит из введения, трех глав основного текста, заключения и приложения.

Во введении раскрывается тема данной работы, подчеркивается актуальность проблемы разработки компонентно-ориентированного программного обеспечения и формулируется основная цель работы, а также кратко описывается структура руководства.

В первой главе описываются общие черты численных моделей, используемых в физике высоких энергий, формулируются требования к разрабатываемому программному обеспечению со стороны пользователей и разработчиков численных моделей. В данной главе также коротко представляются и обсуждаются основные идеи, лежащие в основе предлагаемого подхода к разработке и использованию численных моделей.

Во второй главе дано детальное описание основных концепций компонентного подхода и его функциональности. Здесь в начале приведено краткое описание архитектуры системы, а затем дано более детальное описание её составляющих. Обсуждается концепция компонента, описаны компонентные интерфейсы и их визуальные представления. Рассматривается вопрос упаковки компонентов и сопутствующего им программного обеспечения в модули, связанные с конкретными областями применения компонентов. Далее рассматриваются механизмы и средства разработки прикладных модулей и компонентов, в частности, классы прикладных типов данных. Здесь же обсуждается документирование компонентов. Затем объясняется модель данных, разработанная для системы NiMax, и, прежде всего, сформулировано понятие события данных. Описывается файл данных и его визуальные представления. Рассматривается библиотека классов для транспортировки данных, и вводится понятие предопределенных событий данных и предопределенных каналов данных. Далее дается детальное описание возможной совместной работы компонентов и объясняется механизм отбора данных на основе их конфигураций. Здесь формулируется концепция событийно-ориентированных компонентных сетей и описывается их визуальное представление. В конце второй главы приводится краткое описание методов контроля и навигации, которые имеются во фреймворке.

В третьей главе дано детальное описание возможностей пользователя при работе с моделями через графический пользовательский интерфейс. В частности, объясняется, как выбрать нужный компонент из предлагаемых списков компонентов. Здесь описывается редактирование компонентов, т.е. редактирование входных данных, параметров компонентов, замена подкомпонентов. Описывается возможность переконфигурации вывода данных. В этой главе рассказывается, как создать моделирующую компонентную сеть из отдельных компонентов и как редактировать уже созданную сеть. Здесь также обсуждаются возможности контролирования выполнения компонентной сети. Рассматриваются вопросы работы с файлом данных, например, как производить отбор данных. Затем объясняются возможности построения одномерных и двумерных гистограмм, а также двумерных плотов. В конце этой главы рассказывается, каким образом

пользователь системы может получить необходимую помощь при работе с системой.

В заключении приводится краткое описание основных идей, лежащих в основе компонентного подхода к разработке и применению численных моделей, а также краткое описание основных свойств созданного программного обеспечения и созданных моделирующих компонентов, и подчеркивается их новизна и научная значимость.

В приложении объясняется физика и описываются численные алгоритмы разработанных моделирующих компонентов (см. также [1]), входящих в прикладные модули адронных моделей. Целью данного приложения является не подробное описание физики и алгоритмов всех имеющихся разработанных, разрабатываемых или предполагаемых к разработке компонентов. Данные модули, с описанным набором компонентов, здесь рассматриваются как "учебные", чтобы дать возможность читателю руководства и потенциальному пользователю системы лучше представить её назначение. Здесь же представлены несколько примеров, демонстрирующие применение компонентов для моделирования, анализа и визуализации адронных процессов. Нужно отметить, что некоторые компоненты включают другие компоненты как их подкомпоненты. В этом случае повторное описание физики и алгоритмов их подкомпонентов опущено.

Необходимо сделать замечание по поводу используемой терминологии. К сожалению, для ряда терминов (например, framework, pattern, view, component, document и т.д.), широко применяющихся в англоязычной литературе при описании программного обеспечения, не существует устоявшихся русских аналогов, правильно отражающих суть описания, что в первую очередь связано с очень быстрым развитием программного обеспечения. В данном случае, при описании программного обеспечения мы использовали либо английские термины (например, фреймворк), либо подбирали русские термины, которые на наш взгляд максимально отражают суть описания.

1 Требования к программному обеспечению

1.1 Общие свойства численных моделей

Даже беглый взгляд на МК-модели генераторов адронных процессов или адронные модели показывает, что они имеют много общего [1]. Прежде всего, большинство из них представляют собой феноменологические модели с огромным числом параметров. Эти параметры могут быть разделены на две группы. К первой группе принадлежат физические параметры или настроечные константы, позволяющие изменять физические результаты, полученные при применении модели. Эта группа параметров выполняет очень важную функцию накопления физической информации об адронных процессах. Вторая группа параметров также позволяет менять результаты работы модели путем изменения логики численного алгоритма модели. Эта группа параметров рассматривается как конфигураторы модели.

Кроме большого количества параметров, МК-генераторы могут требовать дополнительной информации, необходимой для их применения. Это, например, информация о физических свойствах частиц: их кварковая структура, электрические заряды, массы, таблицы распадов и т.д. Модели, работающие с ядрами, требуют таблиц об основных и возбужденных состояниях ядер, информацию об их энергиях связи, спинах, параметрах ядерной плотности и т.д.

Важной особенностью таких данных является то, что они используются моделями, как правило, только в режиме чтения.

Общим для МК-генераторов является и характер их работы. Они преобразуют входные данные в выходные, давая при этом ответы на запросы пользователей, введенные посредством входных данных. В качестве входных данных могут использоваться характеристики частиц или ядер, определяющие начальные состояния системы. В качестве выходных данных могут выступать также характеристики частиц, ядер, т.е. новое состояние системы. При такой работе МК-адронные генераторы всегда имеют дело с 4-векторами (например, энергии-импульса) и их преобразованиями. Как правило, в любой адронной модели используется n-частичная кинематика.

Большинство адронных моделей представляют собой многокомпонентные модели. Многокомпонентная модель включает в себя другие модели как дополнительные или альтернативные моделирующие компоненты и имеет сложную логику выполнения подкомпонентов. Такая ситуация является типичной, когда адронные модели применяются для выполнения прикладных задач, чтобы получить наиболее реалистическое описание конечных состояний для адронных реакций.

Практически все модели адронных процессов представляют собой сложные численные модели. В этом случае непростой задачей является разделить «физику» и «алгоритм», т.е. отделить влияние входных данных и физических параметров на результаты моделирования от влияния разработанного численного алгоритма на эти результаты. С другой стороны, также частой является ситуация, когда один и тот же численный алгоритм может быть использован в разных моделях, описывающих различные физические явления. Например, распад резонансов и распад возбужденных ядер при учете только их релятивистских фазовых объемов, упругие столкновения партонов, адронов и ядер, алгоритмы поиска участников столкновений и распадов в адронной и партонной транспортных моделях и т.д.

В процессе инициализации и во время выполнения моделей могут возникать различные ошибки. Источником ошибок могут быть как некорректно заданные пользователем данные, так и ошибки, связанные со сложностью используемых численных алгоритмов. Следует заметить, что во втором случае очень часто причины возникновения ошибок непредсказуемы.

Результатом работы любого МК-генератора является большой объем новых данных, которые должны анализироваться. Эти выходные данные могут содержать либо специфическую информацию только о конечных состояниях моделируемой реакции, или же полную информацию, содержащую историю генерации данного физического события.

Приведенный выше список общности МК-генераторов адронных процессов может быть расширен. Например, помимо кинематики реакций, МК-адронные модели имеют дело с выбором случайных переменных, описываемых различными распределениями, следовательно, необходима библиотека функций для генерации таких распределений. Реализация численных моделей требует большого количества специальных математических алгоритмов: для решения уравнений, интегрирования, интерполяции и аппроксимации, сортировки и поиска и т.д. Однако уже ясно, что разработчики адронных моделей должны учитывать эту общность либо в единообразном способе программирования модельных кодов, либо в их единой структуре, либо в использовании общих методов и т.д.

1.2 Типовые требования со стороны пользователей моделей

Различные стратегии использования предъявляют различные требования к моделям. Пользователи моделей, проводящие теоретические или экспериментальные исследования адронных процессов, нуждаются в возможности «играть» с выбранной моделью, например, они должны иметь возможность изменять параметры модели, конфигурировать модель или отдельные её составляющие, иметь право выбора альтернативных подкомпонентов, а также настраивать конфигурацию её выходных данных. Помимо задачи предоставления этих возможностей, возникает необходимость создания механизма, контролирующего нежелательные изменения, вносимые пользователем. При выполнении моделей также должна выдаваться информация для пользователей о статусе выполняемых моделей, на основе которой они могли бы принимать решения, контролирующие процесс выполнения моделей. Кроме этого, требуется гибкая система обслуживания исключительных ситуаций и обработки возникающих при этом ошибок.

Другая группа пользователей, или, иначе, прикладные пользователи, интересуются только физическими событиями, полученными в результате работы моделей. Для таких пользователей должны быть предложены уже настроенные модели, например, модели с выбранными значениями параметров. Выходная информация, являющаяся результатом работы моделей, должна быть сведена к необходимому минимуму и представлена в требуемой форме.

Разумеется, обеим группам пользователей необходимо много больше, например, простой и понятный механизм ввода входных данных, предохраняющий от ошибок, связанных с неправильным вводом, средства для анализа и визуализации сгенерированных выходных данных. Таким образом, для любых пользователей их работа с моделями должна быть удобной и продуктивной.

1.3 Типовые требования со стороны разработчиков моделей

Разработчики моделей должны иметь возможность модифицировать существующие адронные модели с целью, например, расширения их области применения или улучшения результатов расчетов. Они могут объединить уже существующие модели для их совместной работы с другими моделями. Также они могут реализовывать новые модели, которые могут работать самостоятельно или вместе с другими существующими моделями.

Перечисленные ситуации определяют первичные задачи разработчиков адронных моделей, напрямую связанные с улучшением или расширением численных алгоритмов моделей. В действительности разработчикам необходимо выполнять значительно больше работы. Например, чтобы удовлетворить требованиям пользователей, разработчикам необходимо создать пользовательский интерфейс. Интерфейсы между кодами адронных моделей, а также различные адаптеры нужны, если модели будут использовать внешние пакеты или работать в различных операционных системах. Разработчикам моделей необходимо заботиться о возможности использования их разработок в составе других приложений и учитывать возможности последующего расширения моделей другими разработчиками.

При традиционном подходе (см., например [2]), создавая адронные модели, разработчик или несколько разработчиков работают независимо каждый над какой-то моделью. Такой подход даже при использовании объектно-ориентированных языков имеет несколько недостатков. Самым существенным из них является то, что

слабо учитывается отмеченная общность моделей, а также то, что опыт проектирования отдельных моделей и опыт создания их программного кода плохо используется при создании новых моделей другими разработчиками. Например, каждый новый разработчик обычно начинает создавать конкретную модель практически с нуля, в результате чего он вынужден проходить все основные стадии разработки, включая анализ и проектирование создаваемого кода. Дублирование проектирования кода ведет к значительным и неоправданным затратам времени разработчиков. К тому же различное проектирование имеет свои особенности, которые в различной степени удовлетворяют требованиям пользователей и разработчиков моделей. Дублирование в проектах кодов приводит к дублированию в создании кодов. При этом качество создаваемых кодов в сильной степени зависит от квалификации и опыта разработчика или разработчиков. Вследствие такого процесса «независимой» разработки модельных кодов более трудным являются их изучение, их поддержка и их расширение, а также их объединение для совместной работы. Как правило, разработчиками адронных моделей являются сами физики, а не специалисты по созданию программного обеспечения, и для них решение сугубо специфических программистских проблем является непростой задачей.

Таким образом, в результате анализа МК-адронных моделей, а также различных требований, предъявляемых к ним пользователями и разработчиками, и учитывая современные технологии программирования было решено разработать объектно-ориентированный фреймворк, предназначенный для создания и использования численных моделей в физике высоких энергий. Он позволяет учесть общность численных моделей, облегчает работу пользователей моделей и увеличивает производительность разработчиков моделей. В этом случае разработчики и пользователи могут уделить больше внимания решению проблем, непосредственно связанных с областями их работы. При этом разработчикам моделей требуется написание меньшего объема кода, так как значительная часть его уже создана и является частью фреймворка. К тому же к ним предъявляются значительно более низкие требования из области знания современных программных технологий при написании модельных кодов. Созданные новые модельные коды, как наследники фреймворка, значительно проще отлаживать и тестировать.

1.4 Основные идеи компонентного подхода

Как уже обсуждалось выше, адронные модели, а также их использование и разработка имеют много общего. При проектировании предлагаемой системы в первую очередь была сделана попытка выделить это общее, так как с точки зрения написания кода оно является стабильным, другими словами, код требуется создать только один раз.

При проектировании и разработке системы было уделено большое внимание концепции компонента. Компонент рассматривается как блок для конструирования составных численных моделей [6], [7], который включен в расширяемую библиотеку модельных компонентов.

Любой компонент может быть разделен на интерфейсную и алгоритмическую части. Интерфейсная часть обеспечивает взаимодействие компонента с внешним миром. Посредством этой части пользователь компонента может работать с её параметрами, с её входными и выходными данными, контролировать процесс её выполнения и т.д. Можно формализовать понятие компонента как набора определенных стандартных интерфейсов. Введение стандартных интерфейсов требует от разработчика моделей следовать определенным правилам при создании компонентов. Эти правила можно учесть,

создавая программные средства для генерации компонентных кодов, тем самым облегчая разработку компонента. Аналогичные средства можно разработать для облегчения написания компонентной документации.

В рамках данного подхода считается, что разработчик компонента должен сконцентрировать свою работу на программировании её численного алгоритма. Для обеспечения эффективности работы разработчика МК-генераторов созданы специальные библиотеки классов - классы прикладных типов данных и классы транспортировки данных. Классы прикладных типов данных представляют реальные физические объекты из данной предметной области, а также многие операции над этими объектами. Классы транспортировки данных упрощают разработку ввода и вывода данных для компонентов.

В объектно-ориентированном программировании очень часто используется механизм наследования для модификации существующего кода с минимальными усилиями. Любой компонент может также быть расширен путем наследования для добавления новых интерфейсов или изменения его алгоритма.

Кроме механизма наследования необходим другой механизм, позволяющий разработчикам компонентов использовать код (как интерфейса, так и алгоритма) уже готовых компонентов. Такой механизм не должен ограничивать возможности разработчика при использовании языка программирования C++ в ходе работы над любой сложной численной моделью или несколькими численными моделями. Этот механизм не должен ограничивать доступ пользователей компонента к любым его подкомпонентам и должен давать возможность замены подкомпонентов. Для решения указанных проблем предлагается механизм, который называется агрегацией компонентов.

Разработчику компонентов может понадобиться возможность использования одних и тех же данных, функций и классов (как правило, принадлежащих к одной области приложения) в разных компонентах. Для решения этой проблемы в предлагаемой системе вводится структура, называемая прикладным модулем, которая объединяет в себе несколько компонентов и связанные с ними общие данные, функции и классы. После компиляции на Windows платформах такие модули представляет собой динамически связываемые библиотеки. Пользователи имеют доступ только к компонентам модулей, то есть важной функцией модулей является также «скрытие» программного обеспечения используемого компонентами от пользователей.

Один или несколько компонентов могут загружаться в память компьютера, и каждый компонент может выполняться как независимый процесс. Пользователь имеет возможность управлять этими процессами, в частности, путем получения информации о выполняемых процессах.

Таким образом, в рамках данного подхода определяется модель компонента и формулируется понятие стандартного компонента [3], [4], [5], предлагается несколько механизмов его разработки и различные способы его использования. Также разработан набор классов для поддержки процедуры создания и использования компонентов.

Любой компонент имеет дело с данными. Он может создавать большой объем данных и записывать их на диск. Он может считывать данные с диска и их обрабатывать. Совместная работа нескольких компонентов требует обмена данными. Чтобы удовлетворить этим нуждам, вводится понятие события данных [6], [7], [8]. Это понятие обозначает порцию данных простых типов, имеющих древовидную структуру. Любое событие данных имеет свое собственное определение, которое описывает конфигурацию данного события и может быть

помещено в память компьютера или записано на диск в соответствии с этой конфигурацией. Для поддержки потоков данных разработаны файл данных и механизм управления виртуальными потоками данных. Компоненты имеют возможность записи своих данных в файл и считывания данных из этого файла. На базе виртуальных потоков реализуется концепция компонентной сети. Под такой сетью понимается коллекция компонентов, связанных в определенную структуру, работающих совместно посредством стандартных интерфейсов и обменивающихся событиями данных для обеспечения необходимой обработки данных.

Таким образом, в рамках данного подхода определяется модель данных [7], [8] и разработан набор классов для поддержки управления структурами и потоками данных. Пользователи системы получают возможность гибкого конфигурирования входных и выходных данных, производимых или принимаемых компонентами, возможность навигации по данным, возможность их отбора и визуализации. Они также могут собирать несколько компонентов в сеть для совместной работы.

2 Основные концепции системы

2.1 Архитектура системы

В этой главе обсуждаются основные составляющие системы NiMax, показанные на рисунке 2.1, и их взаимодействие. Данный пакет разработан с использованием языка программирования C++. Основная часть пакета не зависит от платформы, на которой он выполняется. Только графический интерфейс пользователя, разработанный с применением библиотеки MFC (Microsoft Foundation Classes) [10], является системно зависимой частью. Текущая версия пакета NiMax с графическим интерфейсом работает на различных Windows платформах (Windows 95, 98, Windows NT и Windows 2000). Однако система NiMax является переносимой и на другие платформы на уровне исходных текстов. В этом случае система используется в режиме командной строки.

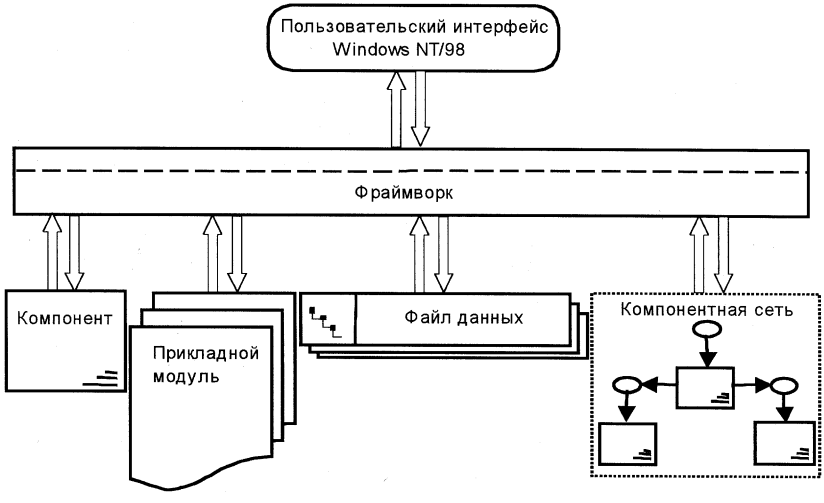

Рис. 2.1. Архитектура системы NiMax

В начале приводится концепция компонента с точки зрения проектирования программного обеспечения. Здесь объясняются различные механизмы, облегчающие процессы создания компонентов и управления компонентами. В частности, обсуждаются вопросы упаковки компонентов и различного программного обеспечения, используемого компонентами, в прикладные модули. С точки зрения разработчиков компонентов данные модули являются составной частью так называемых компонентных проектов, которые также включают в себя различные средства разработки компонентов и прикладных модулей.

Затем обсуждается вопрос управления данными. В частности, дано объяснение файла данных системы. Несмотря на то, что файл данных играет очень важную роль в системе, он составляет только часть более широкой концепции, которая определяется как модель данных. Ключевым элементом модели данных является понятие события данных.

Здесь представлена идея построения компонентных сетей. Обсуждаются детали сборки сетей пользователями, а также отличия сборки компонентных сетей от разработки сложных компонентов методом агрегации.

В заключение данной главы приводится краткое описание графического интерфейса пользователя. В следующей главе, где обсуждается взаимодействие пользователя с системой, будут более полно раскрыты детали этого интерфейса.

В рамках данного подхода разработан большой набор классов для поддержки управления и создания компонентов, управления данными, а также их визуализации. Именно этот набор и рассматривается как фреймворк. Различные методы этого фреймворка образуют программный интерфейс, который связывает отдельные части: компоненты, прикладные модули, файлы данных, компонентные сети и графический интерфейс пользователя в систему NiMax, как это показано на рисунке 2.1.

Следует обратить внимание на то, что система NiMax создана с использованием документ-отображение-технологии (см. [10]), основной идеей которой является разделение объектов хранения (носителей) данных и объектов отображения данных. Все операции по редактированию данных осуществляются посредством объектов отображения. С другой стороны, к каждому документу - носителю данных (компонент, сеть и т.д.) привязан файл с соответствующим расширением. Данный подход позволяет разрабатывать различные системы визуализации и редактирования, не изменяя при этом систему управления данными.

Текущая версия системы NiMax включает четыре типа документов. Первый тип - это документ «рабочая область». Основная информация, хранимая в данном документе, это список компонентов, зарегистрированных на данный момент в системе. Второй тип документов – это файл компонентной сети. Файл сети содержит данные о настройке компонентов: текущие значения параметров, конфигурацию выходных данных, описание связей между компонентами, входящими в сеть и т.д. Третий тип документов – это файл данных пакета NiMax. Четвертым типом документа, входящим в систему, является файл хранения графической информации.

Технология документ-отображение дает возможность независимого расширения количества типов документов и их отображений. Например, справочная система пакета разработана с применением HTML-документов, которые могут просматриваться и редактироваться стандартными программами просмотра операционной системы. Созданная справочная система может быть использована отдельно от пакета NiMax для рекламы или изучения входящих в пакет компонентов.

2.2 *Модель компонента*

В этой части представлены основные интерфейсы компонента и их отображения, описаны свойства компонентов, рассказано об упаковке компонентов модули и обсуждается документация компонентов.

2.2.1 Интерфейсы компонента

Компонент можно рассматривать как набор стандартных интерфейсов. Посредством интерфейсов клиент (пользователь, фреймворк или другой компонент) может обращаться к компоненту, запрашивая определенный сервис, а компонент его предоставляет. Только посредством интерфейсов компонент общается с внешним миром. Интерфейс представляет собой набор методов и связанных с ними данных. Ниже раскрывается функциональность стандартных интерфейсов компонента, схематично представленных на рисунках 2.2 - 2.3 .

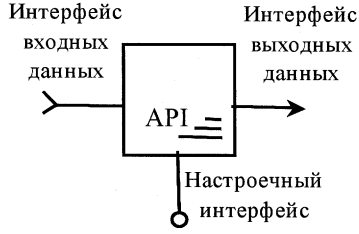

Рис. 2.2. Интерфейсы главного компонента

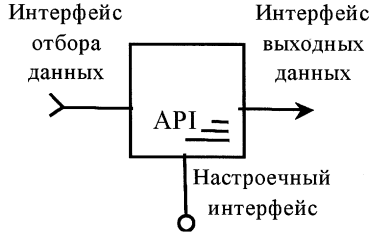

Рис. 2.3. Интерфейсы компонента

При помощи *интерфейса входных данных* пользователь посылает запрос и вводит необходимые для выполнения этого запроса данные. Запрос и входные данные представляются в виде *карты входных данных* [5],[6]. Данные в такой карте имеют линейную структуру, или, другими словами, список данных простых типов. Линейная структура позволяет организовать простой и понятный для пользователя способ задания начальных данных для выбранной модели. Разработчик компонента может предоставить его пользователю выбор из нескольких карт входных данных. Если компонент запускается посредством карты входных данных, такой компонент система рассматривает как главный.

Настроечный интерфейс дает возможность настраивать компонент для получения соответствующего результата его выполнения. Посредством этого интерфейса пользователь может редактировать параметры и конфигураторы компонента. Фреймворк имеет в своём составе набор классов для поддержки параметров и карт входных данных. Следует обратить внимание, что интерфейс входных данных и настроечный интерфейс - это схожие интерфейсы с точки зрения разработчика компонентов. Они используют одни и те же классы и одни и те же структуры данных. Списки входных данных могут рассматриваться как списки параметров обязательных для задания пользователями, которые необходимы для работы компонент.

Результаты работы компонента (выходные данные) обслуживаются посредством *интерфейса выходных данных*. Этот интерфейс управляет записью данных, производимых компонентом, в файл данных или осуществляет их передачу на вход других компонентов. Интерфейс поддерживает событийную последовательность данных. Каждое событие данных имеет древовидную конфигурацию [8], [9] (см. более детальное описание в следующей части).

Компонент может считывать входные данные из файла данных или принимать их от других компонентов. В этих случаях он запускается посредством *интерфейса отбора данных*. Интерфейс отбора данных позволяет компоненту выбирать необходимые данные из входного потока в соответствии с заданной конфигурацией, которая организована в виде *карт отбора данных* [8], [9]. Карты отбора данных отличаются от карт входных данных. Они не включают в себя значения данных, а только задают их конфигурацию (см. более детальное описание в части "Модель данных"). Пользователь компонента не может изменить эту конфигурацию, однако он может выбрать одну из предложенных для данного компонента карт. В этом случае карту отбора данных можно понимать как определенную разработчиком компонента конфигурацию входных данных.

Разработан также *интерфейс для получения информации о процессе выполнения компонента* (не показан на рис. 2.2 и рис. 2.3), который служит для управления сообщениями об ошибках, предупреждающими сообщениями или просто текстовой информацией о работе компонента.

Каждый компонент помимо стандартных интерфейсов имеет свои программные интерфейсы, т.е. набор открытых и защищенных методов, которые могут быть вызваны непосредственно или при помощи стандартных интерфейсов. Следует подчеркнуть, что функциональность компонента определяют именно программные интерфейсы.

Компоненты могут быть классифицированы по типам, которые определяются наличием или отсутствием определенных интерфейсов. Например, исполняемый компонент – это компонент, который может быть использован как стартовый в компонентной сети, обязан иметь либо интерфейс входных данных, либо интерфейс отбора данных. Компоненты, не имеющие данных интерфейсов, определяются как компоненты общего использования. Такие компоненты не могут быть запущены на выполнение, но они могут быть использованы как подкомпоненты.

Разработанная система поддерживает технологию расширения как уже представленных интерфейсов, так и добавления новых. Например, в настоящее время ведется разработка интерфейсов для компонентов, работающих с данными, приходящими с внешней аппаратуры. Также ведется разработка интерфейсов для поддержки внешних баз данных. Таким образом, добавляя новые интерфейсы,

можно расширять область применения системы NiMax и облегчить работу разработчика компонентов в этой области.

2.2.2 Отображения интерфейсов

Каждый интерфейс компонента связан, по крайней мере, с одним объектом отображения, как показано на рисунке 2.4. Объекты отображения выполняют визуализацию карт входных данных, карт отбора данных, параметров компонента, конфигурацию его выходных данных и структуры сложного компонента, а также информации, получаемой при выполнении компонента. Эти отображения позволяют пользователю производить различные действия, меняющие состояние выбранного компонента (см. следующую главу).

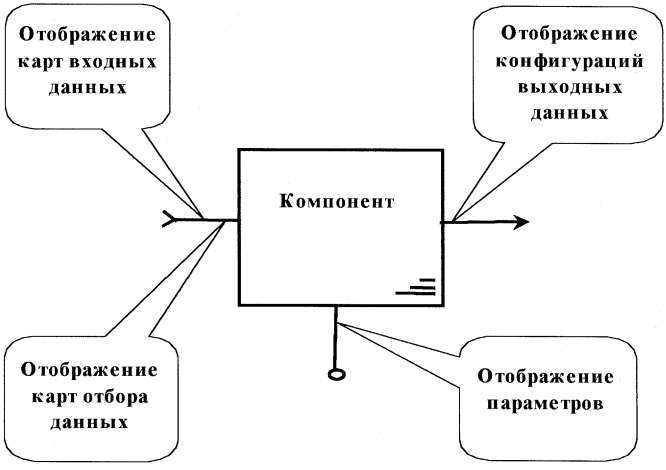

Рис. 2.4. Отображения основных интерфейсов

Один интерфейс может быть связан с несколькими отображениями для реализации различных способов представления или редактирования, но каждый объект отображения связан только с одним интерфейсом. Изменения отображений не требуют изменения связанного с ним интерфейса, что позволяет независимое развитие графического пользовательского интерфейса.

Кроме отображений, связанных с интерфейсами, существуют и другие отображения, имеющие отношение к компонентам, например, объект отображения документации компонента, который помогает пользователю работать с данным компонентом.

2.2.3 Свойства компонентов

Кроме стандартных интерфейсов, компоненты имеют и другие общие элементы (см. рис. 2.5). Все компоненты имеют собственные фабрики классов [4]. Фабрика классов это глобальная функция, зарегистрированная в системе специальным образом. Это название не случайно, так как, используя эту функцию, можно создавать любое количество объектов заданного типа. Через фабрику классов можно получить так называемую статическую информацию о компоненте. Примером статической информации является документация компонента. Она

называется статической, потому что для её получения не требуется создавать сам объект. Такая информация используется либо для создания компонента, либо для организации системы навигации по множеству компонентов [4]. В отличие от статической информации такая информация, как: конфигурация компонента, список его параметров, конфигурации его входных и выходных данных и т.д., называется динамической, так как для её получения необходимо создать реальный объект.

Основным элементом статической информации является числовой глобально-уникальный идентификатор [11]. Знание идентификатора компонента помогает получить полную информацию о компоненте. Связывание компонента с уникальным числовым идентификатором дает возможность организовать простую и эффективную систему управления компонентами. В частности, описываемая далее возможность построения агрегирующих (составных) компонентов, основана на использовании идентификаторов компонентов как значений поля данных специальных объектов, называемых «заместителями» [4].

Рис. 2.5 Шаблон компонента

Система NiMax поддерживает *наследование компонентов*. Таким образом, в системе существуют базовые и производные компоненты. При наследовании компонентов стандартные и программные интерфейсы базового и производного компонентов просто объединяются (см. рис. 2.6).

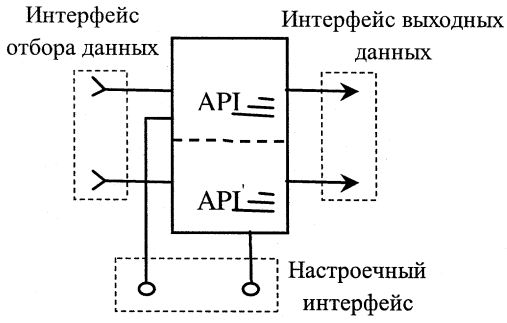

Рис. 2.6. Наследование компонентов

Система NiMax поддерживает также *агрегирование компонентов* [8], [9]. В этом случае составной компонент может включать в себя несколько других подкомпонентов, как это показано на рис. 2.7. Следует заметить, что подкомпоненты могут принадлежать различным динамически связываемым библиотекам. Любой подкомпонент может также агрегировать другие компоненты. Таким способом образуется дерево агрегированных компонентов. В случае создания новых компонентов, используя механизм агрегации, настроечные интерфейсы и интерфейсы выходных данных подкомпонентов объединяются с соответствующими интерфейсами агрегирующих компонентов (см. рис. 2.7).

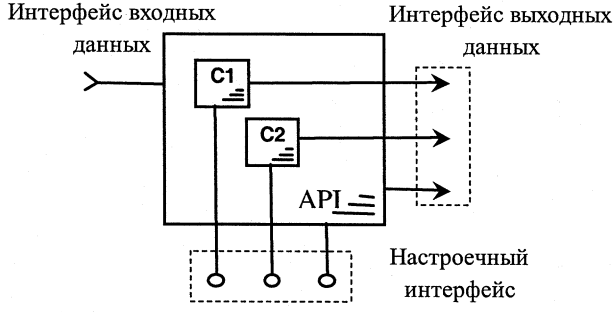

Рис 2.7. Агрегация компонентов

Следует обратить внимание на важные свойства механизма агрегирования компонентов. Пользователь агрегирующего компонента может видеть его структуру и имеет доступ к любому его подкомпоненту посредством графического интерфейса.

Совместное использование при разработке механизмов наследования и агрегации дает уникальную возможность выполнять изменение алгоритма составных компонентов при помощи только пользовательского интерфейса, без вмешательства в исходные тексты и последующей перекомпиляции. Такая возможность называется *замещением компонентов в процессе выполнения*. На рис.

2.8 показан механизм замещения одного подкомпонента на альтернативный подкомпонент.

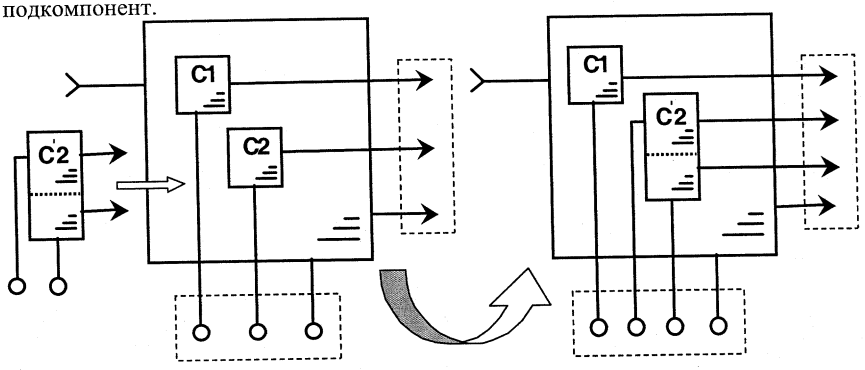

Рис. 2.8. Замещение компонентов

2.2.4 Модули компонентов

Разработанные компоненты могут обращаться к общим данным, использовать одни и те же функции и классы, которые, как правило, относятся к одной прикладной области. Правильное построение такого, связанного с компонентами математического обеспечения существенно увеличивает продуктивность работы создателя компонентов. Это также позволяет более эффективно использовать ресурсы компьютера и, что более важно, дает возможность строить масштабируемую систему из независимых блоков.

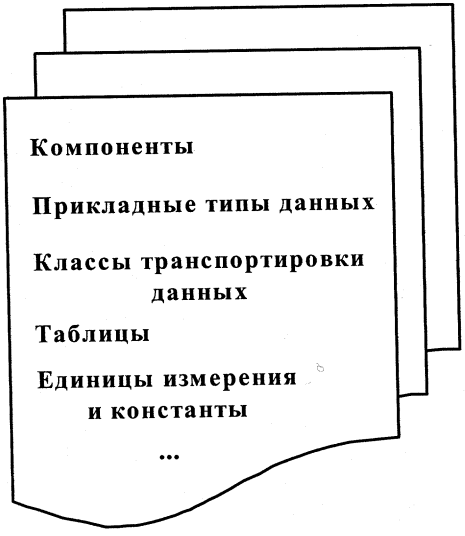

Рис. 2.9. Пример содержимого прикладного модуля

Для решения этих задач в системе NiMax предлагается объединять компоненты и связанное с ними программное обеспечение в модули, принадлежащие конкретной области приложения. Например, модули адронных моделей (см. ниже), модули для описания электромагнитных процессов или модули для описания транспорта низкоэнергетических нейтронов в делящихся средах и т.д. Пример категорий объектов, входящих в модуль, показан на рис. 2.9. Как правило, помимо компонентов в модули входят классы прикладных типов данных (см. описание компонентных проектов) и классы транспортировки данных [5], [6] (см. описание модели данных). Однако они, как и другое программное обеспечение, используемое компонентами модуля, недоступны и невидимы для пользователей компонент.

Модули могут быть самодостаточными на уровне исходных текстов. Это позволяет использовать модуль как единицу распространения или обмена между пользователями. Однако независимость модулей не запрещает использовать компоненты из других модулей. Более того, механизмы наследования и агрегации прозрачны на уровне модулей, что, в свою очередь, не препятствует замещению подкомпонентов в составном компоненте посредством пользовательского интерфейса. В случае предполагаемого совместного применения модулей каждый отдельный модуль содержит список модулей, на которые он ссылается. Для пользователей системы модули приготовлены как динамически связываемые библиотеки, объединенные со своими файлами документации.

Большое количество моделирующих компонентов для моделирования адронных процессов уже разработаны и включены в модули адронных моделей [12] (см. приложение "Модули адронных моделей").

В эти модули включены также используемые физические константы и единицы измерения, такие, как МэВ, ГэВ, Фм, постоянная Планка и т.д. Для работы с размерными физическими величинами использован подход, заимствованный у разработчиков GEANT4 [2]. Таблицы, содержащие свойства частиц и ядер, также являются частью адронных модулей. Такие таблицы, как правило, используются компонентами только в режиме чтения, что открывает возможность управлять ими с помощью внешних средств, например, систем управления базами данных. Для создания перечисленных выше таблиц была использована система управления базами данных Microsoft Access. Более того, для единообразного взаимодействия с базами данных создается специальный стандартный интерфейс компонентов, который позволит работать с любой другой базой данных, которая поддерживает язык управления SQL (Structured Query Language). В разработанные модули кроме таблиц данных включено также большое количество специальных и вспомогательных функций и классов, таких как 3- и 4- векторы, генераторы случайных чисел, методы сортировки и т.д. классов. Большое количество таких классов взято из библиотеки CLHEP [13].

2.2.5 Документация компонентов

Каждый разработанный компонент должен сопровождаться его документацией. Она включает имена разработчиков, лицензионное соглашение, описание области применимости компонента, описание его входных карт, параметров, карт отбора данных, конфигурации выходных данных, описание его программного интерфейса, используемых подкомпонентов и т.д. Документация компонента реализована в виде HTML-файлов.

2.3 Компонентные проекты

В этой части рассматриваются различные механизмы и средства разработки компонентов, а также классы прикладных типов данных. Они являются составными частями концепции создания компонентов в рамках системы NiMax, которая определяется нами как *компонентный проект*. В рамках компонентного проекта разработчик компонентов может использовать также классы транспортировки данных (см. часть "Модель данных").

2.3.1 Компонентные шаблоны и генераторы кодов компонентов

Наличие большого количества общих элементов в компонентах позволяет разработать шаблоны компонентов [5], [6] (см. рис. 2.5) с целью быстрого создания так называемых скелетонов компонента посредством простого редактирования компонентных шаблонов. Такой шаблон можно рассматривать как образец стиля программирования компонентов. Таким образом, разработчику компонента нужно проектировать только её численный алгоритм. Наличие шаблонов компонентов, в свою очередь, делает возможным создание мастеров компонентов или генераторов компонентного кода аналогичных генератору классов в системе Microsoft Visual C++ 6.0 [10]. Такие генераторы создают скелетон компонента посредством работы с диалоговыми панелями. Результатом его работы будет набор файлов, корректно связанных с необходимыми классами и интерфейсами.

2.3.2 Библиотеки классов прикладных типов данных

Библиотеки *классов прикладных типов данных* призваны повысить производительность разработчиков алгоритмов компонентов и улучшить качество их работы. Для разработчиков прикладных алгоритмов они обеспечивают возможность работать в терминах «языка» данной прикладной области, так как большинство объектов классов прикладных типов данных являются двойниками реальных объектов из областей приложения. Эти классы включают также основные операции над объектами из областей приложения. Например, библиотека классов частиц, с использованием которой написаны алгоритмы компонентов МК-генераторов событий взаимодействия частиц и ядер, содержит классы создания для объектов типа адрон, кварк, струна и атомное ядро. Эти объекты представляют физические объекты из области физики высоких энергий.

Мы работаем над существенным расширением набора классов прикладных типов данных с целью оказания помощи разработчикам в написании алгоритмов компонентов для моделирования не только адронных, но и электромагнитных процессов взаимодействия частиц и ядер в широкой области энергий. Нашей целью является получение такого набора классов прикладных типов данных, который бы поддерживал разработку алгоритмов компонентов для моделирования прохождения частиц и ядер через сложные среды. Эти библиотеки будут включать уже упомянутые и новые классы для описания частиц и ядер, классы для описания транспорта частиц, так называемые трассеры, и классы для описания материалов и геометрии сред.

2.3.3 Наследование и агрегация компонентов

Разработчик компонентов может расширять функциональность уже существующих компонентов, используя механизм наследования (см. рис. 2.6), который поддерживается рассматриваемой системой. В частности, наследование

компонентов - это удобный способ добавления новых карт входных данных или карт отбора данных.

Другим способом создания новых компонентов является механизм агрегации [8], [9] (см. рис. 2.7). Механизм агрегации не накладывает никаких ограничений при программировании численного алгоритма компонента на языке C++. Более того, он упрощает программирование, например, разработчик агрегирующих компонентов не должен заботиться о создании и уничтожении объектов подкомпонентов. Управление подкомпонентами выполняется фреймворком (см. раздел "Методы фреймворка"). Таким образом, компоненты могут использоваться C++ программистами как обычные классы за исключением операций создания и уничтожения объектов этих классов.

2.3.4 Генераторы модулей компонентов

Модуль в системе NiMax является единицей разработки. Для разработчиков компонентов такой подход более естествен и эффективен, когда коды компонентов объединяются с дополнительным программным обеспечением, описывающим конкретную область приложения. Создание полностью изолированных компонентов усложняет разработку и приводит к дублированию кода и данных.

Для поддержки модульной структуры в систему включены шаблоны модулей. В настоящее время разрабатываются генераторы модулей с использованием графического интерфейса для разработчиков, а для пользователей предлагается система навигации по модулям.

2.3.5 Генераторы документации компонентов

Создание документации компонента является очень важной частью процесса разработки компонента. Система NiMax включает документационные шаблоны и генераторы текста документов, чтобы помочь разработчикам компонентов. Эти средства обеспечивают создание и появление документов в стандартной форме. Документационные шаблоны похожи на шаблоны компонентов, которые обсуждались выше, и разработаны в дополнение к ним. Генераторы документов сканируют коды модулей и компонентов и генерируют для разработчика начальные варианты документов, которые затем можно редактировать посредством предоставляемых разработчику диалоговых панелей, которые появляются по мере их заполнения необходимой информацией. После того как процесс редактирования закончен, разработчик компилирует созданные файлы с помощью Windows HTML Help компилятора. Полученные файлы интегрируются в общую справочную систему, в которой нужная информация может быть найдена по содержанию или ключевым словам.

2.4 Модель данных

В этой части рассматриваются основные механизмы обмена, хранения и обработки данных, поддерживаемые системой NiMax. Здесь также вводятся основные понятия, используемые для описания модели данных, такие, как событие данных, файл данных и его отображения, предопределенные события и предопределенные каналы данных, а также описываются классы транспортировки данных.

2.4.1 Событие данных

Элементарной единицей обмена данными является *событие данных* [8], [9]. Событие данных - это порция данных, которая состоит только из данных простых типов (int, float, double, т.д.). Любое событие данных имеет свое определение. Это определение включает уникальный идентификатор события и описывает конфигурацию этого события. Конфигурация события данных состоит из набора определений каналов данных. Элементарной единицей обработки данных системы является канал, и большинство операций, выполняемых с данными, ведутся с каждым каналом в отдельности. Определение канала данных состоит из идентификатора канала, типа канала, имени канала, а также другой информации, которая может быть добавлена. Пример конфигурации события данных показан на рис. 2.10 в сравнении со структурами языка С.

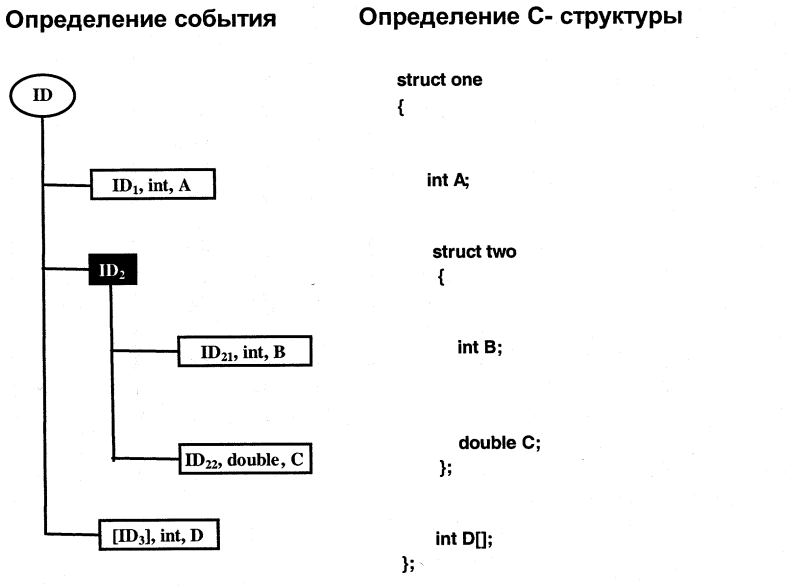

Рис. 2.10. Пример конфигурации события данных

Конфигурация события данных может быть определена с помощью пары скобок. Внутри внешней пары скобок могут быть отдельные каналы, массивы каналов или группы каналов, заключенные во внутренние скобки. Во внутренних скобках опять могут быть заключены отдельные каналы, массивы каналов и группы каналов, заключенные в скобки, и т.д. Массив каналов отличается от канала тем, что в определение канала вводится флаг повторения, а в области данных записывается количество элементов массива. Таким образом, конфигурация события данных имеет древовидную структуру, состоящую из определений каналов данных. Внутри события данных простые типы данных могут быть сгруппированы, чтобы представлять достаточно сложные структуры данных, например, они могут представлять массивы, структуры, массивы массивов и массивы структур языка С

(см. рис. 2.10). Следует подчеркнуть, что событие данных не является объектом, какого либо класса языка C++.

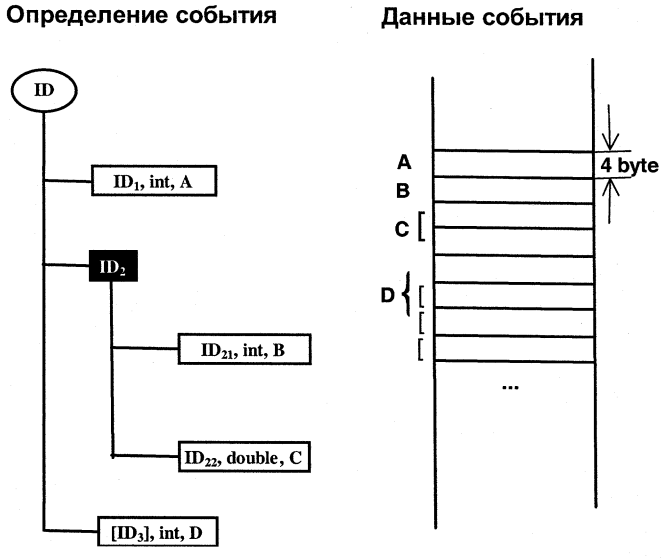

Рис. 2.11. Пример размещения события данных в памяти компьютера

Событие данных может быть помещено в память компьютера или записано на диск компьютера в соответствии с его конфигурацией (см. рис. 2.11).

2.4.2 Файл данных и его отображения

Компонент может записывать данные в файл данных и читать данные из этого файла. Упрощенная структура файла данных представлена на рис. 2.12. Конечно, файл данных имеет заголовок (не показанный на рис. 2.12), который необходим для идентификации файла и для облегчения навигации по этому файлу. Файл данных состоит из двух частей. Первая часть содержит список конфигурации событий, присутствующих в файле. Вторая часть содержит сами данные. Важно заметить, что именно наличие конфигурационной части в файле позволяет пользователю системы производить структурный отбор данных, записанных в файл. Кроме списка конфигураций событий первая часть хранит некоторую статистическую информацию о событиях данных. Часть файла, содержащая данные, состоит из последовательности записей. Каждая запись представляет одно событие. Данные в записи организованы в соответствии с конфигурацией этого события. Для облегчения процесса навигации по файлу данных каждая запись имеет свой заголовок, отражающий размер и местоположение этой записи в файле. Преимущество такой организации состоит в том, что он может быть легко прочитан вне предлагаемой системы.

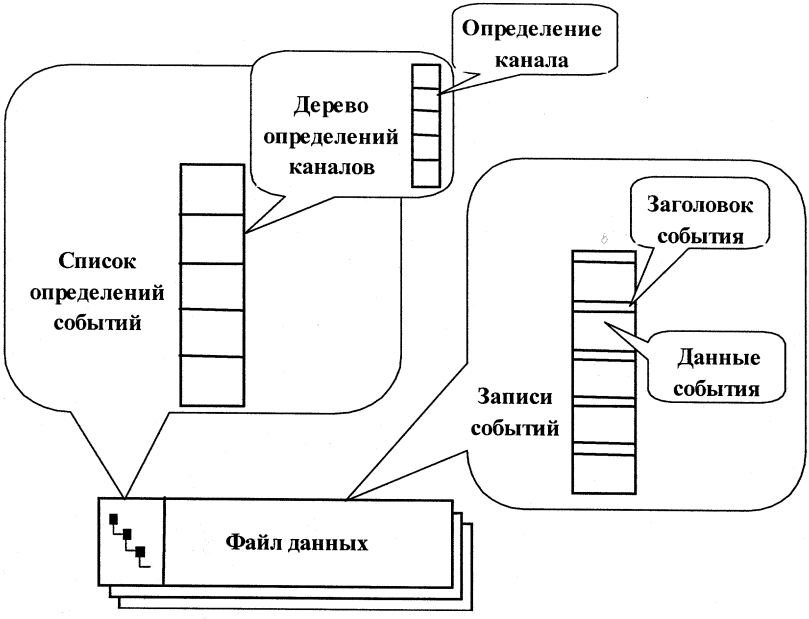

Рис. 2.12. Структура файла данных

Для пользователей системы разработано несколько отображений файла данных. Среди них наиболее важными являются отображение конфигураций данных и отображение самих данных записанных в файл (см. рис. 2.13). Именно посредством отображения конфигурации данных пользователь проводит структурный отбор данных (см. примеры работы пользователя с файлом данных в следующей главе).

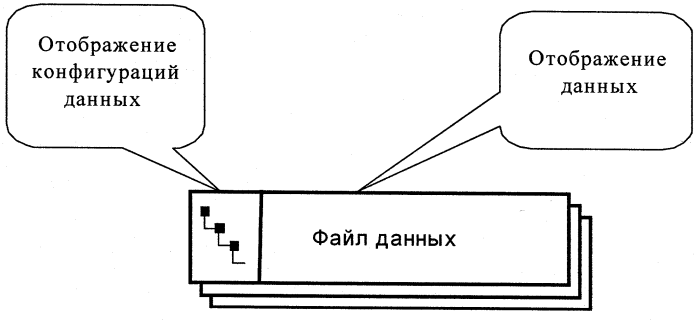

Рис. 2.13. Отображения файла данных

2.4.3 Классы транспортировки данных

Многие из разработанных компонентов могут рассматриваться как генераторы или преобразователи объектов *классов транспортировки данных* [5], [6]. Классы транспортировки данных помогают разработчику компонентов

проектировать и реализовывать вывод данных, сгенерированных компонентами, и дают возможность получать полную историю обработки или генерации данных составными компонентами. Они помогают разработчику компонентов проектировать и реализовывать ввод данных для компонентов, считываемых с файла данных или получаемых от других компонентов (см. описание компонентных сетей). Таким образом, данная библиотека классов транспортировки данных позволяет разработчику компонента уделять больше внимания непосредственно разработке численного алгоритма компонента, не углубляясь в детали операций с вводом и выводом данных. Эти классы помогают разрабатывать универсальные численные алгоритмы. Уровень универсальности алгоритма определяется количеством необходимой входной и выходной информации. Библиотека классов транспортировки данных имеет иерархическую структуру, потому что очень удобны базовые классы, которые либо определяют общие свойства объектов данной иерархии, либо включают общие операции над этими объектами. В силу этого наиболее универсальными являются алгоритмы, которые в качестве входных и выходных данных используют объекты узловых (базовых) классов транспортировки данных. По-существу классы транспортировки данных относятся к классам прикладных типов данных, которые рассматривались в предыдущей части. Использованием термина классы транспортировки данных, хочется подчеркнуть их значимость в обмене данными. Аналогичный набор классов необходимо создавать и в других областях приложения системы, а не только при разработке МК-генераторов. Важным и общим свойством объектов данных классов является их умение, т.е. наличие соответствующих методов в классах записать данные в файл данных и считывать данные из этого файла. С учётом этого факта в рамках системы создан шаблон для разработки классов транспортировки данных с целью облегчения работы разработчиков компонентов.

2.4.4 Предопределенные события данных и каналы

Для поддержки и независимого расширения сервиса предоставляемого системой, вводится понятие *предопределенных событий* данных и *предопределенных каналов* данных. Каждое такое событие или каналы связываются с определенным сервисом, который регистрируется в системе и предоставляются пользователям.

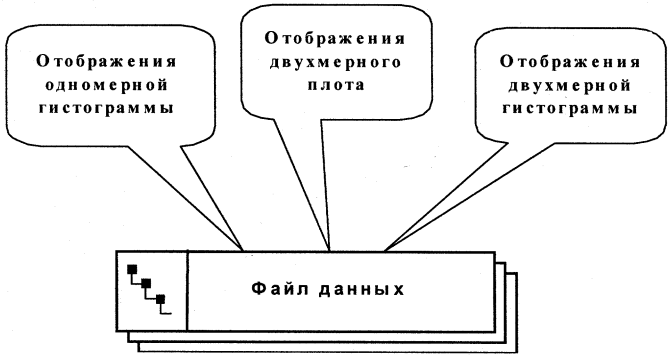

Рис. 2.14. Отображения предопределенных групповых каналов

Как правило, предопределенные каналы или предопределенные группы каналов имеют фиксированную конфигурацию (в этом смысле они аналогичны объектам C++). Благодаря уникальным идентификаторам таких событий и каналов, фреймворк знает, с каким сервисом они связаны. Например, фреймворк знает, как выдать пользователю табличное или графическое отображение для предопределенных каналов гистограммы и плота. Для предопределенных событий и каналов их идентификаторы могут рассматриваться в качестве их типов.

Основной смысл введения таких событий или каналов состоит в необходимости обеспечить продуктивную работу пользователей системы, работающих в какой-то конкретной предметной области. Например, в физике высоких энергий, где широко используются гистограммы. Так, содержания одномерной и двумерной гистограммы, а также двумерного плота могут быть записаны в файл данных, прочитаны из файла и графически изображены как предопределенные группы каналов, что демонстрируется на рис. 2.14 (см. также примеры отображений гистограмм и плота в следующей главе). Параметры компонента, карты входных данных являются другим примером предопределенных событий, которые представляют состояние компонента. Фреймворк также знает, как отобразить и выполнить такие события.

2.5 Компонентные сети

В этой части обсуждается совместная работа компонентов, когда компоненты обмениваются данными посредством файла данных или путем взаимодействия через виртуальные потоки данных, посылая друг другу события данных напрямую. Во втором случае компоненты объединены в сеть. Прежде чем перейти к описанию совместной работы компонентов, необходимо объяснить, каким образом компонент отбирает для себя входные данные из данных, сгенерированных другим компонентом.

2.5.1 Отбор данных по заданной конфигурации

Идея отбора входных данных компонента из данных, сгенерированных другими компонентами [8], [9], объясняется на рисунках 2.15 – 2.16. Сгенерированные события данных, как это объяснялось в предыдущей части, могут иметь линейную или, в общем случае, древовидную структуру. Эта структура задается конфигурацией события данных. Фреймворк анализирует конфигурации сгенерированных событий данных и ищет в этих конфигурациях только требуемые конфигурации, которые могут быть частью анализируемых конфигураций. Требуемые конфигурации определяются разработчиком компонента посредством создания карт отбора данных. Обнаружение фреймворком требуемой конфигурации определяется, как обнаружение точки входа в данные.

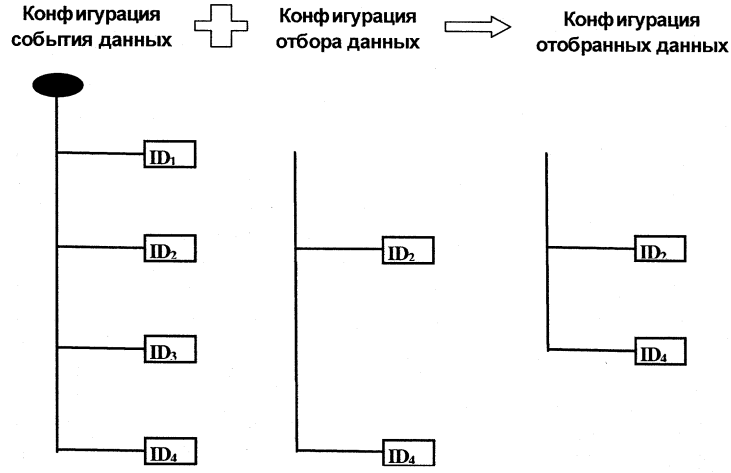

Рис. 2.15. Отбор данных, имеющих линейную структуру

Следует подчеркнуть некоторые особенности ситуаций, когда компонент получает события данных от других компонентов. Он может взять либо часть данных из события, соответствующую требуемой конфигурации, либо взять все данные из события, которое включает требуемую конфигурацию. Компоненты, которые берут данные события целиком, определяются как *фильтры событий данных*. Сами данные в отфильтрованных событиях данных также могут модифицироваться компонентом.

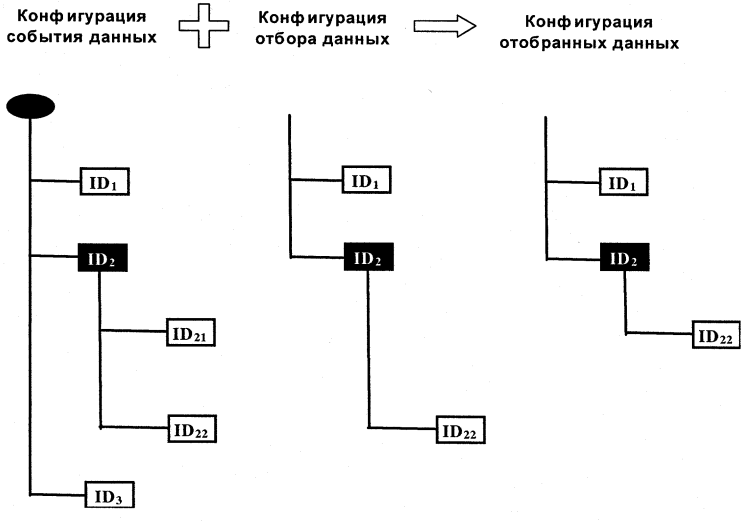

Рис. 2.16. Отбор данных, имеющих древовидную структуру

Разработчик компонента может предложить несколько карт отбора данных для компонента. Таким образом, у пользователя компонента есть возможность настраивать компонент с целью распознавания компонентом того или иного события или каналов события из посылаемых или читаемых им событий. Такая настройка осуществляется путем регистрации одной из карт отбора данных. Возможна ситуация, когда фреймворк обнаружит несколько подходящих (в соответствии с картой отбора данных) конфигураций в анализируемом событии данных. Это означает, что найдено несколько точек входа в данные анализируемого события. Тогда пользователь имеет право выбора среди найденных точек входа. Посредством интерфейса пользователя он может запретить компоненту отбирать данные с определенных точек входа.

Важно подчеркнуть, что, программируя численный алгоритм определенного компонента, разработчик этого компонента не должен изучать и использовать системные или какие-то компонентные классы с целью обеспечения возможности совместной работы компонентов [8]. Разработчик компонента не должен изучать даже конфигурацию событий данных, которые компонент может получить или может читать. От разработчика компонента не требуется знание источника событий данных (файл данных или другие компоненты), которые он может получить. Разработчик компонента должен знать, как правильно программировать конфигурацию требуемых входных данных.

2.5.2 Коллаборация компонентов

Под *коллаборацией компонентов* понимается их взаимодействие посредством стандартных интерфейсов выходных данных и интерфейсов отбора данных [8], [9]. В случае коллаборации компонентов мы имеем две различные ситуации, которые показаны на рисунках 2.17 – 2.18, где черные стрелки указывают направление потоков данных. Первая ситуация возникает, когда один компонент записывает свои выходные данные в файл данных, а другой компонент считывает эти данные из файла для последующей обработки. Первая ситуация иллюстрируется рисунком 2.17.

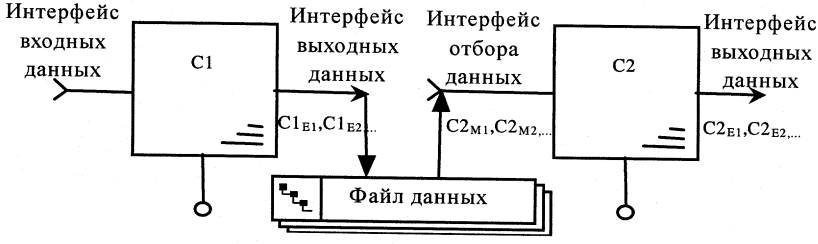

Рис. 2.17. Коллаборация компонентов через файл данных

На этом рисунке показано, что компонент C1 генерирует события данных, имеющих различные конфигурации: $C1_{E1}$, $C1_{E2}$, и т.д. и записывает эти события в файл данных. Компонент C2 читает файл данных и отбирает данные согласно заданным конфигурациям отбора данных: $C2_{M1}$, $C2_{M2}$ и т.д. В данной ситуации

коллаборация компонентов идет через файл данных. В этом случае компоненты совершенно независимы друг от друга.

Необходимо подчеркнуть, что для читающего компонента нет необходимости ждать, когда первый компонент закончит писать на файл все события. Он может начать чтение сразу же после окончания записи первого события. Эта возможность особенно важна в случае необходимости проведения мониторинга данных.

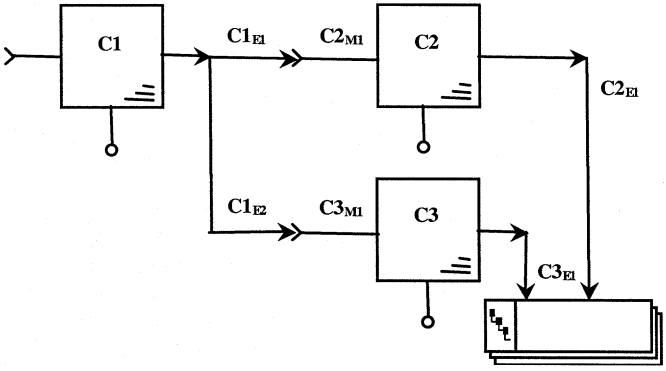

Рис. 2.18. Событийно-ориентированная компонентная сеть

Вторая ситуация (см. рис. 2.18) состоит в том, что один из компонентов генерирует данные, которые посылаются на вход других компонентов. В этой ситуации коллаборация компонентов осуществляется посредством обмена событиями данных. В этой ситуации компоненты выполняются как один процесс.

2.5.3 Событийно-ориентированные компонентные сети

Набор компонентов, которые коллаборируют посредством стандартных интерфейсов выходных данных и интерфейсов отбора данных, посылая и получая события данных, определяется нами как событийно-ориентированная компонентная сеть [8], [9]. Пример такой сети показан на рис. 2.18. На этом рисунке показано, как компонент C1 генерирует события данных, имеющих различные конфигурации $C1_{E1}$ и $C1_{E2}$, затем компоненты C2 и C3 получают данные от этих событий и отбирают среди них данные, соответствующие конфигурациям $C2_{M1}$, $C3_{M1}$. Эти компоненты генерируют новые события данных, имеющие конфигурации $C2_{E1}$ и $C3_{E1}$. Новые события данных записываются на файл данных.

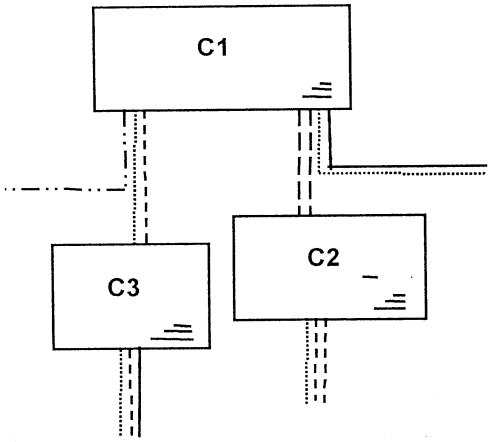

Рис. 2.19. Аналогия компонентной сети с электронным устройством

Для данной сети и способа отбора данных компонентами имеется прямая аналогия с электронным прибором, как это показано на рис. 2.19. На этом рисунке различные линии обозначают различный тип проволочек, выходящих из и входящих в электронные компоненты. Несколько проволочек могут быть скручены в кабель. Кабель может включать другие кабели («древовидная структура» кабеля). Отдельная проволочка имеет своим аналогом определение канала данных (простой тип данных), и кабель имеет своим аналогом конфигурацию события данных. Таким образом, аналогом описанного выше механизма отбора данных в соответствии с их конфигурацией является то, что только отдельные проволочки или отдельные кабели из выходящего кабеля выбираются для соединения компонентов.

Проводя и далее аналогию сети с электронным прибором, можно больше детализировать соединения компонентов в сети. Внутри интерфейса выходных данных и интерфейса отбора данных можно выделить группу методов, называемых соединителями или разъемами. Компонент выдает данные посредством выходного разъема и получает данные посредством разъема отбора данных (входного разъема). Компоненты связываются друг с другом через эти разъемы. Любой разъём имеет дело с потоками данных (файл данных или память компьютера) и конфигурациями событий данных (конфигурацией выходных данных или конфигурацией отбора входных данных). Разъемы дают возможность пользователю компонента перенаправить или переконфигурировать входные и выходные данные компонентов. Например, пользователь может заставить компонент записать выходные данные в файл данных, считывать входные данные из файла данных или от другого компонента (см. примеры в следующей главе). В настоящее время компоненты могут иметь по одному входному разъему и одному выходному разъему, то есть они могут быть объединены только в "линейную" сеть. Так как компоненты расширяемы посредством механизма наследования, то разработчик компонентов может добавить новые разъемы или соединения для наследуемых компонентов.

Следует отметить, что подкомпоненты в составном компоненте не могут получать данные от компонентов, которые не входят в составной компонент или от

файла данных посредством разъемов отбора данных. Важно заметить, что все компоненты в сети имеют дело только с одним файлом данных.

В настоящее время пользователь системы имеет возможность создавать только такие сети, для которых порядок выполнения компонентов определяется направлением потока данных (см. рис. 2.18). Таким образом, в каждой сети должен быть компонент, который выполняется первым. Это так называемый стартовый (он получает стартовое событие) или главный компонент. Он считывает входные данные посредством входного интерфейса или посредством интерфейса отбора входных данных.

Следует также отметить, что любой компонент в сети имеет возможность записывать свои данные в файл данных. Это замечание касается и подкомпонентов в составных компонентах сети.

В общем случае любая событийно-ориентированная сеть требует написания процедуры управления событиями данных. Следовательно, для каждой отдельной сети она должна быть написана и скомпилирована. Преимущество выбранного в рамках данной системы типа сетей (pipeline nets) состоит в том, что такая процедура может быть написана один раз, скомпилирована и скрыта от пользователей.

2.5.4 Отображения компонентной сети

Компонентная сеть в системе NiMax является специальным файлом, состоящим из двух частей. Как и любой файл системы, он может быть сохранен для дальнейшего использования, переименован, уничтожен и т.д. Первая часть этого файла может рассматриваться как часть, содержащая определение сети. Вторая часть файла содержит процедуру управления событиями данных. Первая часть файла создается фреймворком на основе информации, полученной от пользователя посредством отображения сети. Фреймворк генерирует вторую часть файла самостоятельно. Часть, включающая определение сети, содержит информацию о парах взаимодействующих компонентов. Она имеет информацию о том, какой компонент стартовый. Именно предоставление этой информации и является процессом сборки сети посредством отображения сети. Используя отображение сети, пользователи могут выделить любой компонент в сети, увидеть его структуру, параметры, конфигурацию выходных данных, карту отбора входных данных, а также карты входных данных для стартового компонента сети (см. примеры редактирования сетей в следующей главе). Очень частой является ситуация, когда сеть содержит только один компонент.

Таким образом, когда мы рассматриваем ситуацию с выполнением компонента, всегда имеется в виду выполнение файла сети. Компоненты, которые могут быть включены в сеть, определяются нами как исполняемые компоненты.

2.5.5 Агрегация компонентов и их коллаборация

В конце этой главы хотелось бы подчеркнуть основные различия между агрегацией компонентов, когда составной компонент включает в себя подкомпоненты, и коллаборацией компонентов, когда несколько компонентов работают совместно в рамках сети (см. рис. 2.20).

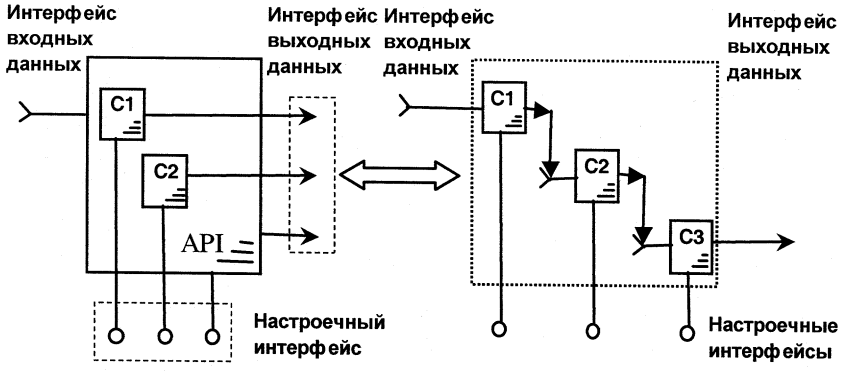

Рис. 2.20. Сравнение агрегации и коллаборации компонентов

В случае агрегации подкомпоненты взаимодействуют между собой посредством их программных интерфейсов, т.е. посредством вызова их методов. В этом случае разработчик должен написать код агрегирующего компонента и, следовательно, необходима компиляция этого кода перед выполнением компонента. В случае коллаборации компонентов их взаимодействие осуществляется посредством стандартных интерфейсов. Пользователь системы собирает сеть посредством интерфейса пользователя. Любая сеть является системным файлом, готовым к выполнению, т. е. компиляции кода не требуется. Такой файл может быть загружен в память компьютера и выполнен как отдельный процесс.

2.6 Методы фреймворка

Детальное описание классов фреймворка, классов прикладных типов данных и других классов, принадлежащих системе, требует написания отдельного руководства. Планируется написать такое руководство с тем, чтобы помочь разработчикам компонентов, а также программистам, которые захотят интегрировать предлагаемую систему с другими системами, например, такими, как GEANT4 [2] или ROOT [14]. Необходимость написания такого руководства требуется для дальнейшего развития системы.

В этой части даётся краткое объяснение функциональных возможностей программного интерфейса фреймворка, основанных на применении его управляющих и навигационных методов. Здесь важно отметить, что имеет смысл выделить в отдельную категорию управляющие и навигационные методы, связанные с файлом данных. Отображения файла данных и эти методы образуют отдельную независимую подсистему, которая имеет собственные применения.

Прежде чем начать описание управляющих и навигационных методов, необходимо сказать несколько слов о глобально-уникальных идентификаторах объектов в рассматриваемой системе. Любой объект системы и любое событие данных имеют уникальные идентификаторы. К настоящему моменту в качестве таких идентификаторов используются целые числа (типа unsigned) и имеется возможность использования 4 миллиардов различных значений. Целые числа

удобны для организации навигации и поиска. Знание уникального идентификатора объекта дает возможность получить полную информацию о данном объекте.

2.6.1 Методы управления

Фреймворк имеет много методов, позволяющих управлять жизненным циклом компонентов, который включает различные фазы: фаза создания объектов компонента, фаза редактирования состояния объектов, фаза выполнения объектов компонента и фаза уничтожения объектов компонента.

Жизненный цикл компонента определяется внутренними процессами фреймворка, которые скрыты от пользователя. Тем не менее, ниже приводится его краткое описание, потому что существует много возможностей для пользователя влиять на это цикл.

Перед тем как начать процедуру создания объекта, фреймворк создает некоторую среду, в которой и будет создаваться данный объект. Эта среда определяет режим создания объекта, различные переменные, которым присваиваются их значения по умолчанию, а также файлы входных и выходных данных, если они будут использоваться. Пользователь имеет возможность модифицировать значения по умолчанию для переменных посредством интерфейса пользователя, например, он может выбрать свои значения по умолчанию для параметров и входных карт. Пользователь также имеет возможность запретить выдачу информации во время выполнения компонентов или запретить выдачу выходных данных компонента и т.д.

Существуют два специальных режима создания любого компонента, исключая виртуальный. В первом режиме объекты создаются только с целью информирования. Это означает, что пользователь системы не имеет возможности изменить состояние объектов, однако пользователь может просматривать структуру компонента посредством интерфейса пользователя. Второй специальный режим - отладочный. Он предназначен для создания объектов компонентов общего типа без права их выполнения. Этот режим дает возможность отлаживать код интерфейсов компонентов.

В случае создания объектов составного или агрегирующего компонента первыми создаются объекты агрегирующего компонента. Затем фреймворк создает объекты подкомпонентов. Порядок их создания следует порядку их определения в составном компоненте. Для того чтобы создать любой объект компонента, фреймворк должен знать только идентификатор компонента. Он использует идентификаторы агрегируемых компонентов для поиска их «фабрик». Если какая либо «фабрика» не найдена, то фреймворк пытается найти альтернативный ей компонент в соответствии с определениями «заместителей» компонентов и иерархией наследования компонентов. Пользователь имеет возможность контролировать этот процесс, разрешая или запрещая замещение подкомпонентов. Такой процесс создания объектов подкомпонентов и объектов их подкомпонентов повторяется до тех пор, пока все объекты подкомпонентов не будут созданы.

Во время фазы уничтожения объектов компонента процесс их уничтожения проводится фреймворком в порядке, обратном процессу их создания.

В течение фазы редактирования компонента пользователь имеет возможность редактировать параметры компонента, его входные карты, выбирать карты отбора данных и конфигурировать вывод выходных данных. В этой фазе фреймворк может вызывать методы проверки правильности редактирования. В случаях обнаружения некорректного редактирования фреймворк информирует

пользователя об этом путем генерации сообщений и значениям неправильно отредактированных переменных присваивает их значения по умолчанию.

Фреймворк помогает пользователю системы создавать или собирать, редактировать и выполнять файлы компонентных сетей. Он также помогает пользователю управлять процессом выполнения сетей. Для фазы выполнения компонентов фреймворк обеспечивает пользователю получение информации о ходе выполнения. При этом фреймворк генерирует сообщения трех типов: информационные сообщения, предупреждающие сообщения и сообщения об ошибках. (см. примеры в следующей главе). В случае получения предупреждающих сообщений или сообщений об ошибках для интересующихся пользователей фреймворк предлагает дополнительную справочную информацию (см. ниже).

Фреймворк дает возможность выполнять одновременно несколько сетей как отдельных процессов и управлять их выполнением. В случае получения ошибки при выполнении фреймворк сам определяет место, где она произошла, и разработчик компонентов не должен прилагать специальных усилий для решения этой задачи.

2.6.2 Методы навигации

Фреймворк выполняет функции библиотекаря компонентов. Он дает возможность пользователям системы увидеть полный список компонентов, включенных в систему, а также возможность регистрации требуемого компонента. Фреймворк дает возможность пользователю просматривать файлы компонентных сетей, открывать их для дальнейшего редактирования и сохранять отредактированные файлы сетей.

Древовидные структуры очень часто используются в системе, например, древовидная структура составных компонентов и древовидная структура событий данных. Таким образом, мы можем использовать одинаковые методы для навигации по структурам составных компонентов и навигации по файлу данных.
Здесь хотелось бы отметить, что, используя методы навигации по файлу данных, разработчик моделей имеет возможность создать адаптеры или драйверы для преобразования формата данных, записанных в нашей системе, в формат данных, приемлемых для внешних пакетов.

Помимо информации, выдаваемой фреймворком о процессах выполнения компонентных сетей, и информационных сообщений, выдаваемых им в течение жизненного цикла компонентов, он предлагает пользователям системы более детальную справочную информацию. Пользователь системы может осуществлять навигацию по справочной информации и поиск нужной информации либо по содержанию, либо по ключевым словам.

Как уже говорилось выше, любой объект в системе, будь то компонент, параметр, сообщение об ошибке и т.д. имеет уникальный идентификатор, что дает возможность связывать эти идентификаторы с HTML - файлами, содержащими документацию об этих объектах. Таким образом, пользователь может получить помощь как бы изнутри кода посредством уникальных идентификаторов, например, зная идентификатор определенного параметра компонента, фреймворк откроет нужный файл, содержащий информацию о данном параметре.

2.7 Интерфейс пользователя

В системе NiMax графический интерфейс пользователя реализован в соответствии с технологией документ-отображение [10], кратко описанной в начале

данной главы, которая позволяет максимальным образом удовлетворить требования пользователей системы.

После начала работы системы NiMax появляется основное «рабочее» окно. Это окно может включать разные графические отображения, которые могут рассматриваться как различные редакторы и навигаторы. Прежде всего пользователю предоставляется возможность видеть все включенные в систему компоненты. Пользователь системы может выбрать требуемый компонент или открыть файл компонентной сети из списка файлов для дальнейшего редактирования. С помощью редактора компонентов пользователь имеет возможность редактировать карты входных данных, параметры и осуществлять замещение подкомпонентов. Следует подчеркнуть, что пользователь видит всю структуру составного компонента и может редактировать параметры или осуществлять замещение подкомпонентов для любого подкомпонента этого компонента. Пользователь также имеет возможность изменения конфигурации выходных данных компонентов и их подкомпонентов. С помощью редактора сетей пользователь имеет возможность собрать несколько компонентов в сеть или модифицировать уже существующую сеть. Пользователь нашей системы может запустить на выполнение одну или несколько сетей и управлять их выполнением. При этом несколько сетей выполняются как независимые процессы. Пользователь имеет возможность осуществлять навигацию, как по конфигурации данных, так и по самим данным, записанным на файле данных, выполняя при этом структурный отбор данных. Пользователь может создавать одномерные и двумерные гистограммы и двумерные плоты и осуществлять их визуализацию. Пользователь может искать справочную информацию как о компонентах, так и о самой системе NiMax.

3 Работа пользователя

3.1 Навигация по спискам компонентов и сетей

В системе NiMax реализовано три вида просмотра имеющихся в системе компонентов (см. рис. 3.1). Первый отображает список компонентов согласно их принадлежности к различным категориям или к различным областям применения компонентов. Второй отображает список компонентов согласно их принадлежности различным модулям, т.е. он показывает содержание динамически связываемых библиотек. И, наконец, третий вид просмотра дает возможность пользователю видеть иерархии компонентов, связанные с наследованием компонентов. Все три способа просмотра компонентов дают возможность видеть тип компонентов, который определяется присутствием или отсутствием того или иного стандартного интерфейса [1]. Тип компонента маркируется иконкой.

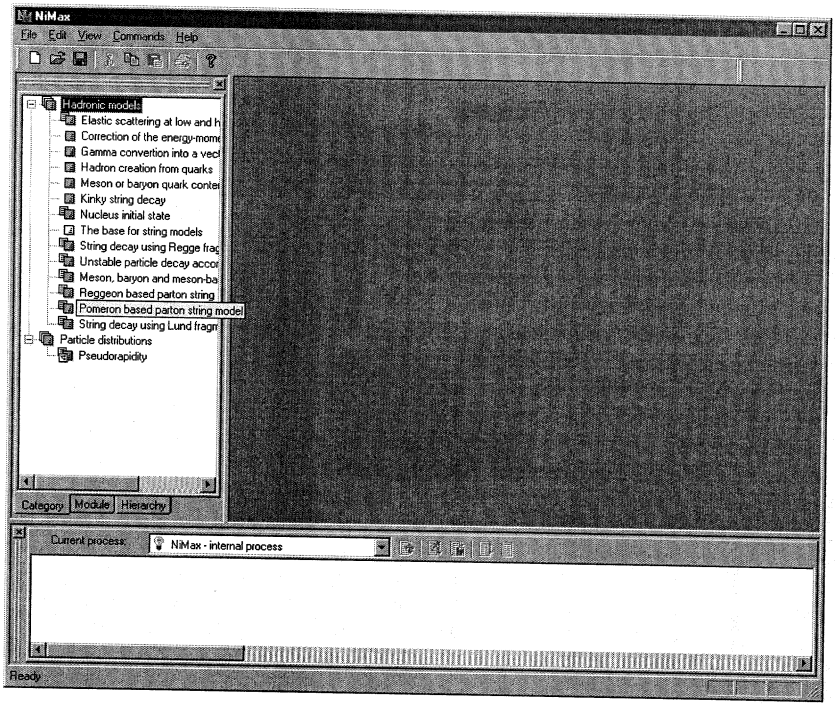

Рис. 3.1. Пример списка компонентов для категории адронные модели

Навигация по имеющимся в системе компонентным сетям осуществляется просмотрщиком файлов, который дает доступ к файлу сети через команду «открыть» из списка команд для работы с файлами. Пользователь может применять также и другие команды из этого списка для того, чтобы «переименовать», «копировать» и «уничтожить» файлы сетей.

3.2 Настройка компонентов

Посредством объектов отображения стандартных интерфейсов система предоставляет возможность пользователю изменять состояние компонента. Пользователь компонентов может просматривать карты входных данных, регистрировать карты входных данных вместо тех, которые регистрируются по умолчанию.

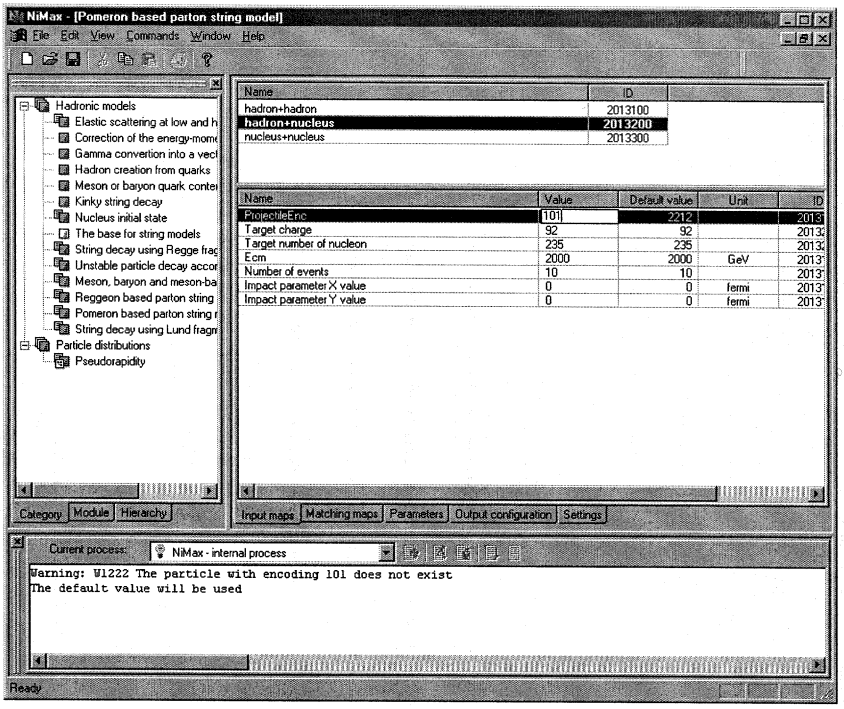

Рис. 3.2. Пример редактирования карты входных данных

Пользователь может редактировать зарегистрированные карты входных данных (см. рис. 3.2), а также редактировать параметры компонентов (см. рис. 3.3). Процесс редактирования входных карт и параметров компонентов контролируется фреймворком (см. предупреждающее сообщение на рис. 3.2). При редактировании входных карт и параметров пользователь всегда имеет возможность «вернуться» к их значениям (значениям по умолчанию), которые они имели до начала редактирования.

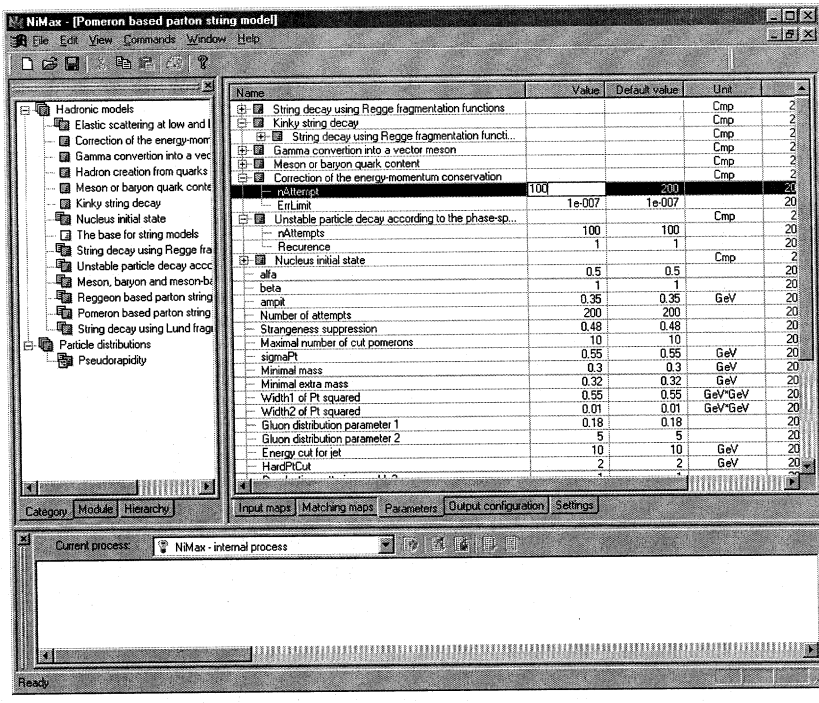

Рис. 3.3. Пример редактирования параметров

Следует отметить, что при редактировании параметров составного компонента, пользователь имеет доступ к параметрам любого его подкомпонента (см. рис. 3.3).

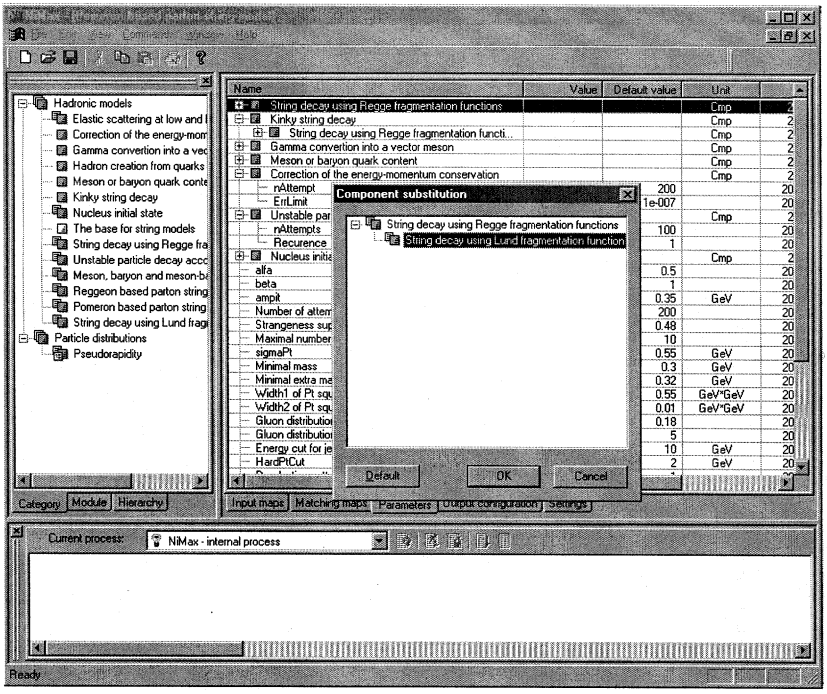

Рис. 3.4. Пример замещения подкомпонента

Подкомпонент в составном компоненте может быть замещен пользователем на другой подкомпонент, если они имеют общий базовый компонент (см. рис. 3.4). Убедиться в том, что компоненты имеют общий базовый компонент, можно посредством отображения иерархии наследования компонентов. Следует напомнить, что пользователь компонента имеет доступ к любому его подкомпоненту, и, следовательно, пользователь компонента может проделать такое замещение внутри любого подкомпонента, если он является составным. Таким образом, в составном компоненте его подкомпоненты могут замещаться на другие подкомпоненты без перепрограммирования, то есть посредством графического интерфейса пользователя, как это показано на рис. 3.4. Такое замещение является эффективным средством модификации и обновления численного алгоритма составных компонентов.

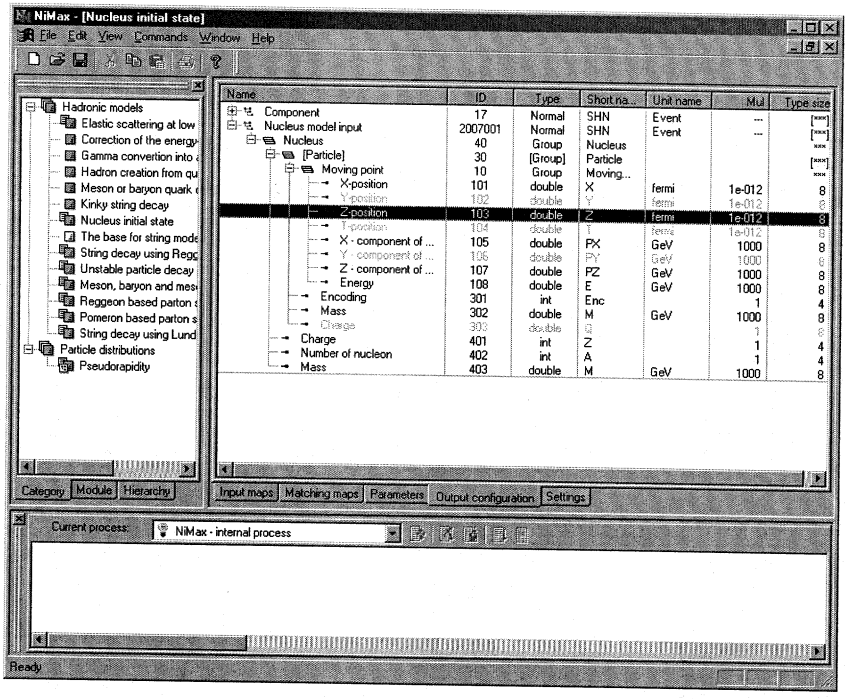

Рис. 3.5. Пример реконфигурации выходных данных

Если ожидается, что компонент будет генерировать выходные данные в процессе своего выполнения, то пользователь имеет возможность управлять процессом выдачи выходных данных. Вывод отдельных событий данных или вывод отдельных каналов событий данных может быть запрещен пользователем (см. рис. 3.5). В случае составного компонента реконфигурация выдачи выходных данных может быть проделана пользователем для каждого отдельного подкомпонента.

Если ожидается, что компонент будет получать входные данные от других компонентов (в составе компонентной сети) или считывать эти данные из файла данных, то пользователь компонента имеет возможность выбрать и зарегистрировать необходимую карту отбора входных данных, если разработчик компонента предоставил несколько таких карт (см. рис. 3.6).

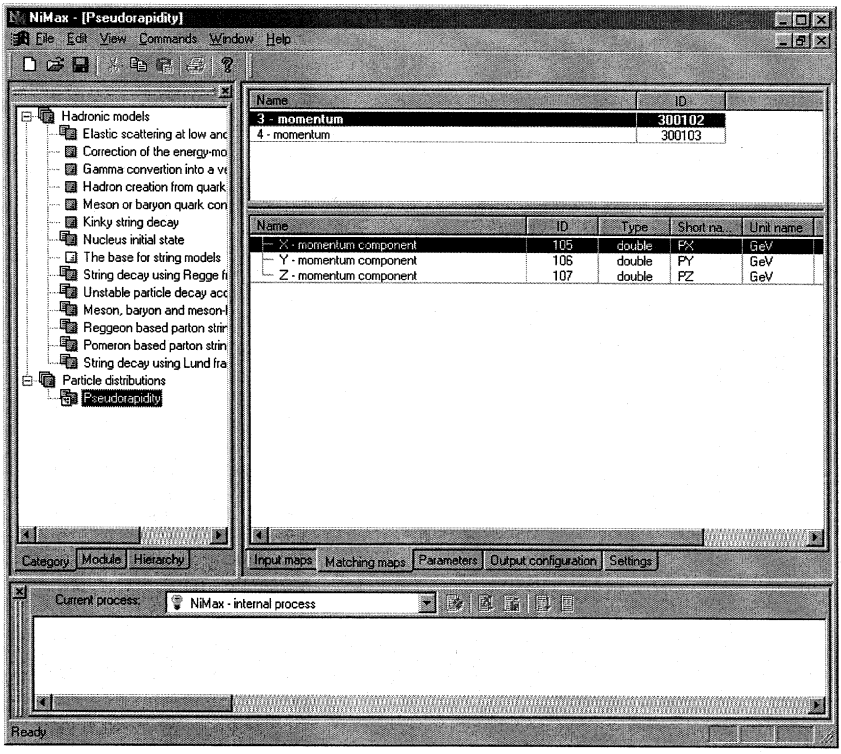

Рис. 3.6. Пример регистрации карты отбора данных

Для пользователя существует возможность проверить, что требуемая входная конфигурация данных действительно существует на присоединенном к ней для чтения файле данных (см. рис. 3.7) или будет сгенерирована другими компонентами, которые являются элементами сети (см. следующий раздел).

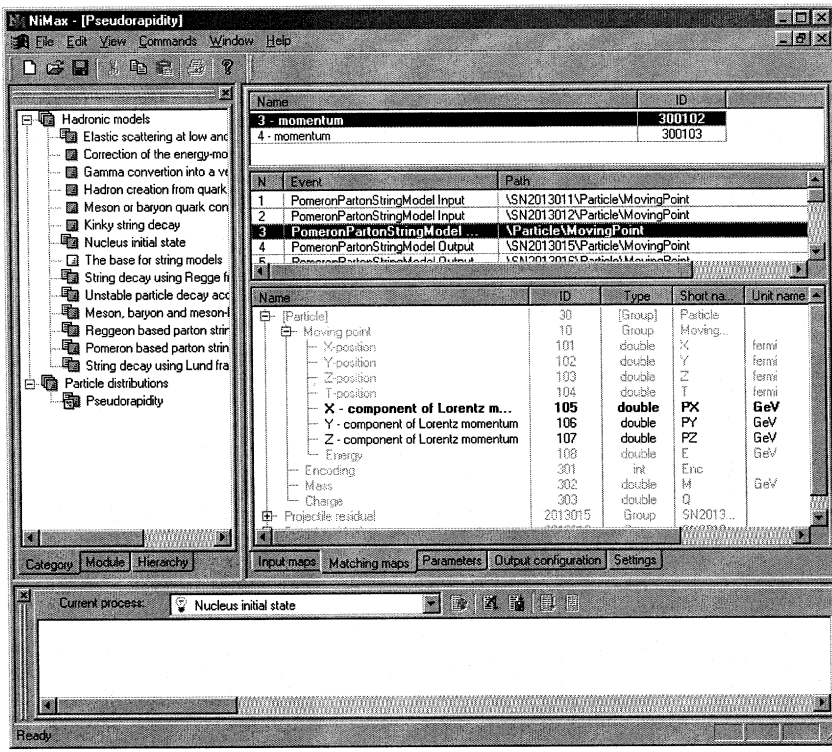

Рис. 3.7. Пример найденной конфигурации отбора данных

3.3 Сборка и редактирование компонентных сетей

Пользователь может открыть файл сети для редактирования. Он может также создать новый файл сети в качестве отправной точки для сборки сети. Фреймворк автоматически обеспечивает необходимую среду, например, именованный файл выходных данных для сборки и выполнения сети. Так как пользователь видит все доступные компоненты в системе, то он может выбрать одну или несколько компонентов из этого списка для присоединения к сети (см. рис. 3.8).

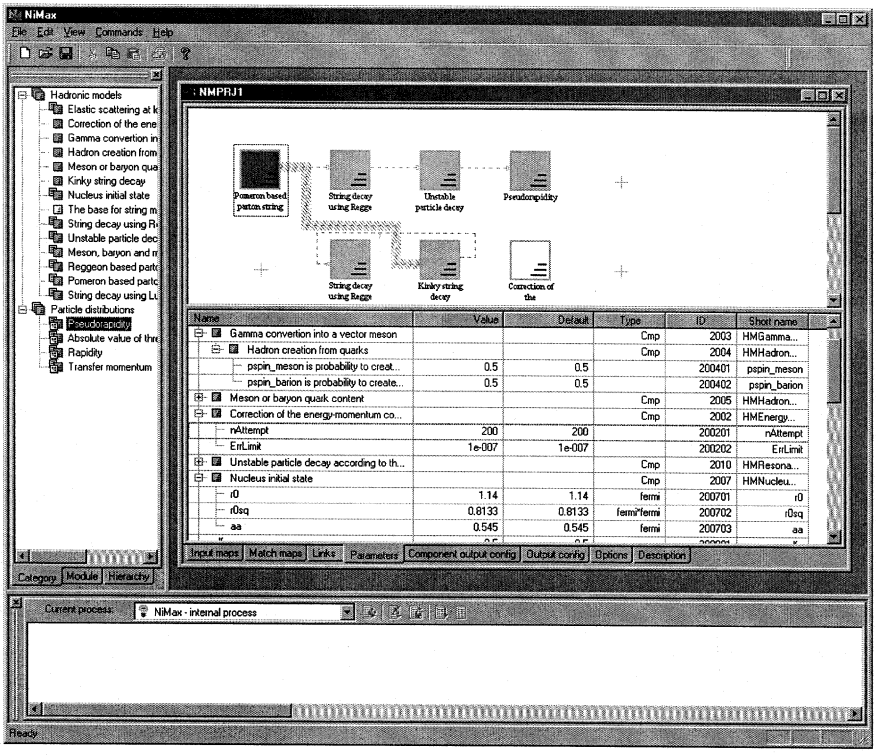

Рис. 3.8. Пример создания компонентной сети

Затем пользователь должен связать пары компонентов посредством их соединителей или разъемов. Соединители отбора данных обозначены с левой стороны символа компонентов, а соединители выходных данных обозначены с правой стороны символа компонентов (см. рис. 3.8). Таким образом, поток данных для пары соединенных компонентов направлен с правой стороны компонента, посылающей данные, на левую сторону компонента, принимающую данные (см. рис. 3.8). Связывая пары компонентов для совместной работы в сети, пользователь имеет возможность конфигурирования выходных данных и выбора (регистрации) карт отбора входных данных (см. рис. 3.9).

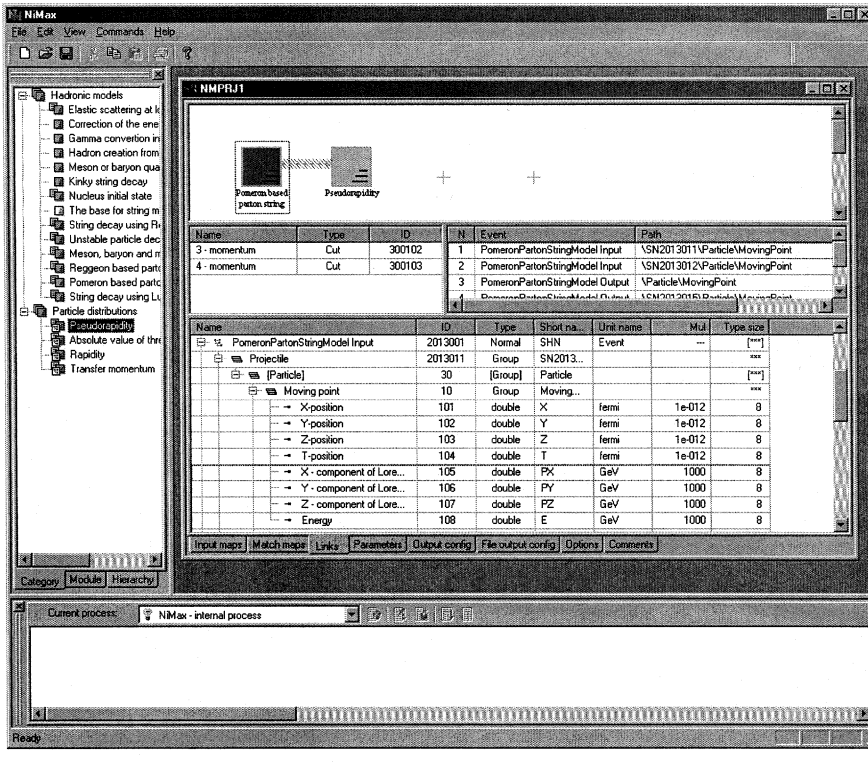

Рис. 3.9. Пример настройки компонентной сети

При настройке связи компонентов пользователь может выбрать конкретные точки входа для отбираемых данных в случае нахождения нескольких таких точек. Связь между компонентами обозначается пунктирной линией (см. рис. 3.8). Для пары компонентов, для которой пользователь ведет настройку выхода и входа данных, такая связь обозначается пунктирной линией с дополнительной штриховкой. Если связь полностью настроена, то она обозначается сплошной линией.

Стартовый компонент окрашивается в красный цвет. Остальные связанные компоненты окрашиваются в синий цвет. Несвязанные компоненты остаются неокрашенными в окне редактора сетей.

При редактировании сети пользователь может не только добавлять компоненты и их соединения, но и также удалять компоненты и их соединения. Пользователь может выделить группу связанных компонентов, не содержащую стартовый компонент. Такая группа связанных компонентов не является полноценной компонентной сетью и не может быть использована как рабочая, но она может быть сохранена и использована при создании или редактировании другой сети.

3.4 Управление выполнением компонентной сети

Пользователь имеет возможность изменить среду, в которой будет работать сеть компонентов (см. рис. 3.10), например, выбрать файл, куда будут записываться выходные данные.

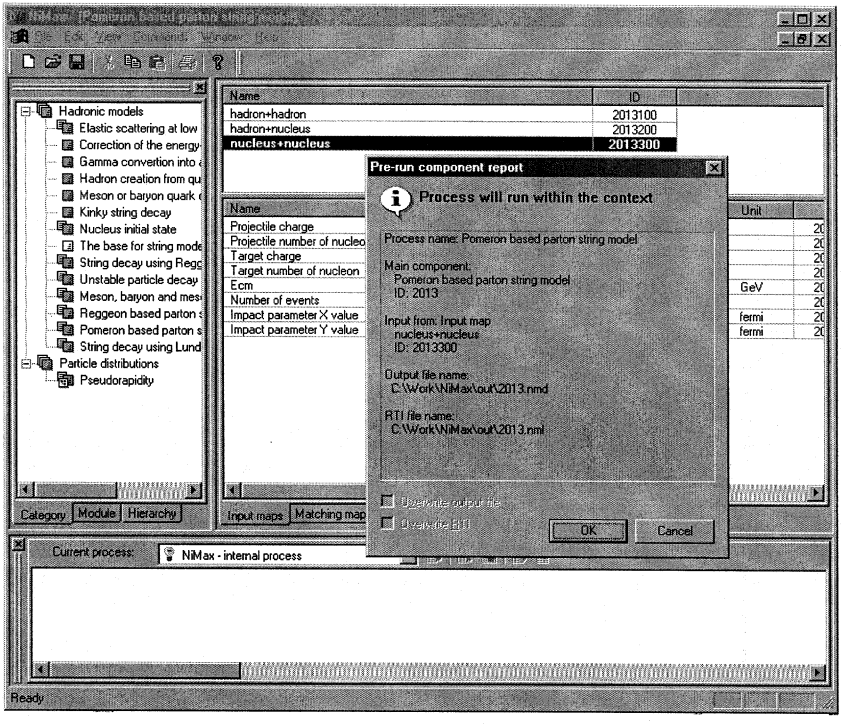

Рис. 3.10. Пример настройки среды для работы сети компонентов

Затем пользователь может запустить в работу сеть, как это показано на рис. 3.11. Работа компонентной сети может сопровождаться появлением различных сообщений. Эти сообщения являются либо просто информационными сообщениями, выдаваемыми в течение нормального процесса работы, либо предупреждающими сообщениями, которые предупреждают о потенциальных ошибках или других ситуациях, которые могут оказать влияние на нормальное выполнение процесса, либо сообщениями об ошибках. Процесс будет продолжаться в случае появления предупреждающих сообщений. Сообщения об ошибках появляются, когда процесс работы сети прерывается. Сообщения об ошибках информируют пользователя о месте и типе ошибки. Каждое предупреждающее сообщение или сообщение об ошибке имеет свой уникальный идентификатор. Наличие таких идентификаторов позволяет системе оказать помощь пользователю в получении более полной информации о причине предупреждения или ошибки.

Пользователь имеет возможность управлять выводом информации о процессе работы компонентной сети. В частности, он может полностью запретить

такой вывод или же запретить такой вывод только от отдельных компонентов, входящих в сеть.

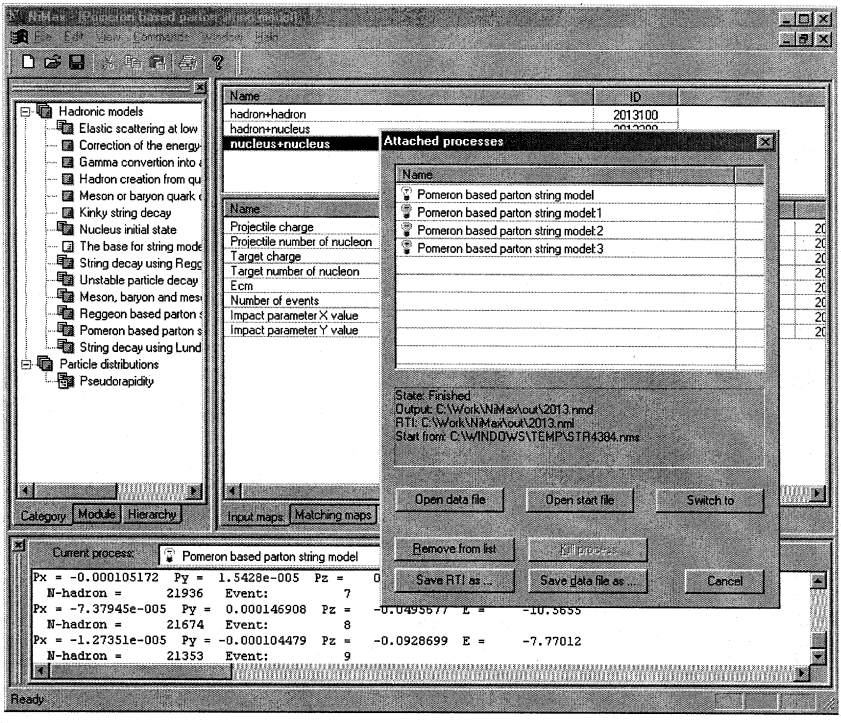

Рис. 3.11. Пример нескольких процессов работы сетей

Пользователь может запустить выполнение нескольких процессов разных сетей или одной сети, например, меняя значения параметров компонентов перед каждым запуском сети. Такая ситуация показана на рис. 3.11. Пользователь может получить информацию о состоянии работы сети, выбрать отдельный процесс и произвести над ним определенные действия, например, убить выбранный процесс или спасти информацию, поступающую от выполняемого процесса в файл.

3.5 Работа с файлами данных

Возможности просмотра файлов данных иллюстрируются рисунками 3.12 – 3.13.

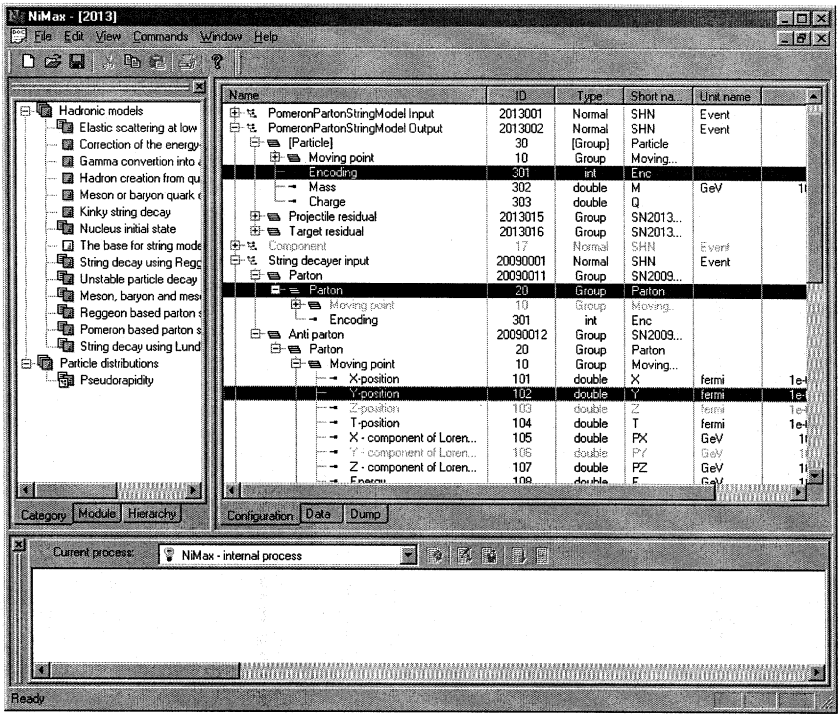

Рис. 3.12. Выделение конфигураций событий данных

Пользователь системы может просматривать конфигурации записанных в файл событий данных (см. рис. 3.12), а также сами данные (см. рис. 3.13). Пользователь может также выделять как целые конфигурации событий данных, так и отдельные их части, например, отдельные каналы этих событий. Такие действия пользователя определяются нами как структурный отбор данных. После выделения конфигураций пользователь может оперировать с самими данными, соответствующими этим конфигурациям. Например, копировать данные, уничтожать данные, защищать их от компонентного доступа и т.д.

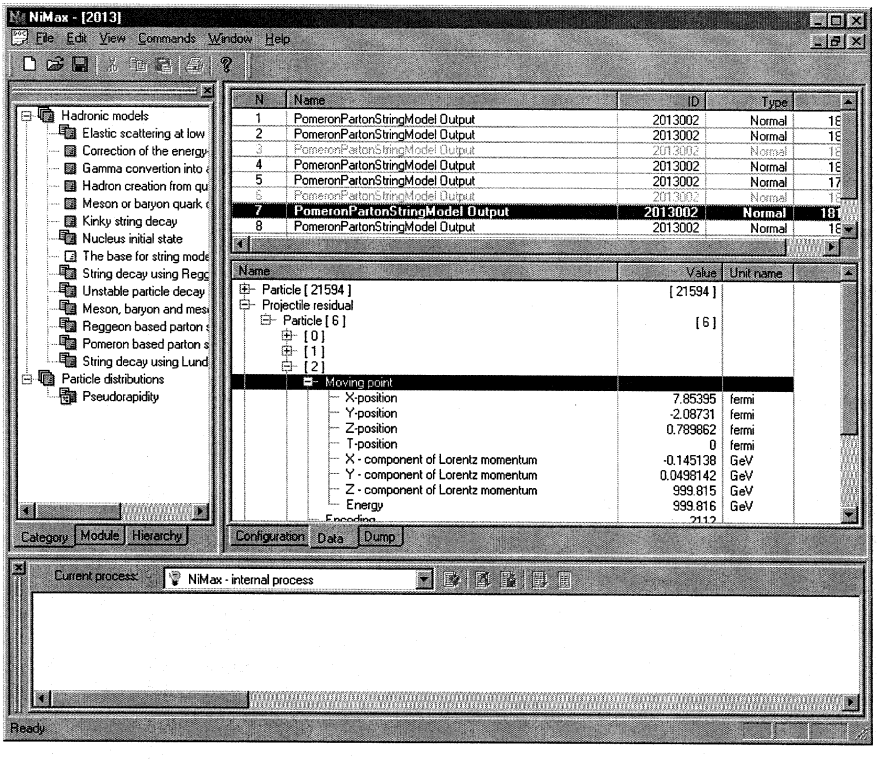

Рис. 3.13. Пример представления данных

3.6 Работа с гистограммами и плотами

Одномерные и двумерные гистограммы с фиксированной шириной каналов, а также двумерные плоты могут быть созданы и визуализированы в системе NiMax. Статистические характеристики, т.е. среднее значение и дисперсия, гистограммы вычисляются через положения и значения каналов. Статистические ошибки значений каналов всегда учитывают статистические веса.

Объекты гистограмм могут хранить значения каналов и их ошибок в виде различных типов данных, и пользователь имеет возможность выбора среди этих типов.

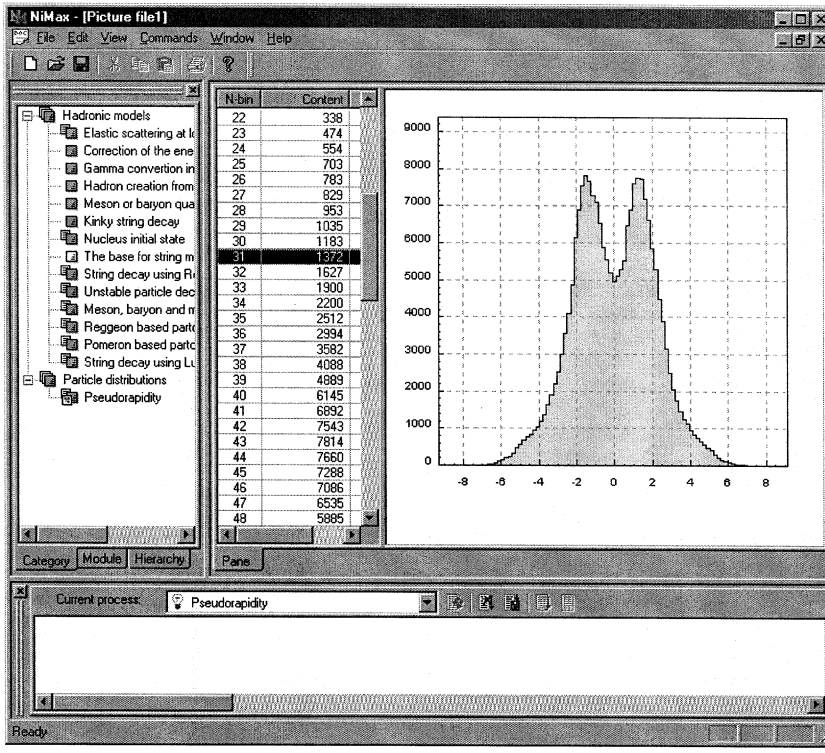

Рис. 3.14. Пример отображений одномерной гистограммы

В случае любых одномерных гистограмм пользователь имеет возможность работать с их табличными отображениями, а также с их графическими отображениями (см. рис. 3.14). В случае двумерных гистограмм пользователь имеет возможность работать с табличными отоброжениями и с графическим ячеечным отображением (см. рис. 3.15).

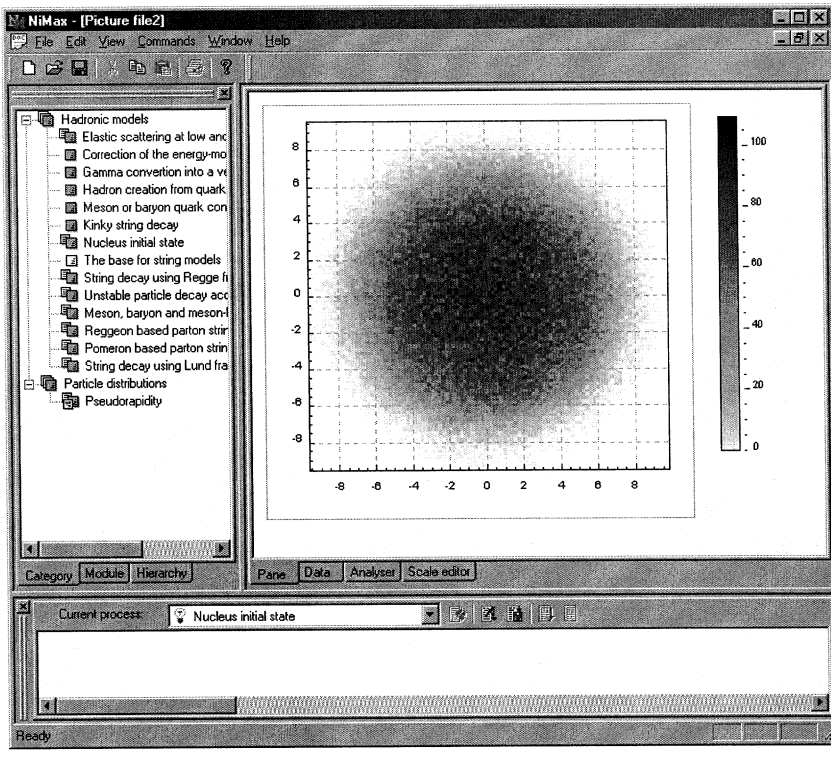

Рис. 3.15. Пример отображения двумерной гистограммы

Плоты представляются в виде табличных и графических отображений (см. рис. 3.16). Различные отображения позволяют проводить различные операции над выбранными данными. Для каждого отображения существуют свои методы выделения данных. Например, пользователь может выделить содержание одного или нескольких каналов в табличном отображении одномерной гистограммы, используя мышь. Он может воспользоваться кистью, для того чтобы выделить содержание определенной области графического отображения плота.

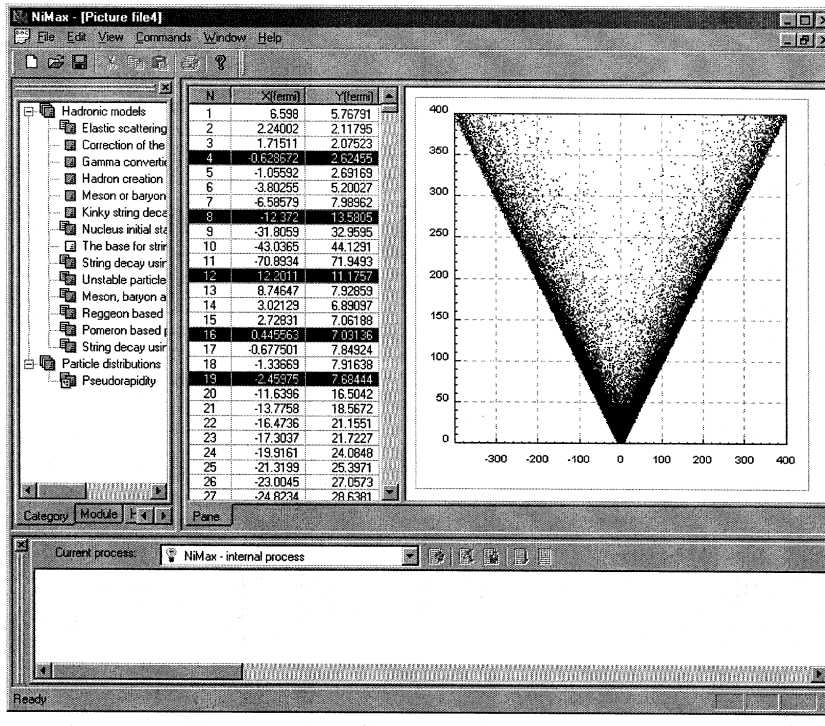

Рис. 3.16. Пример отображений двумерного плота

Необходимо сделать замечание по поводу возможностей работы с гистограммами в системе NiMax. Многие возможности из пакета построения гистограмм HBOOK [15], который может рассматриваться как некий стандарт требуемой функциональности для работы с гистограммами в области физики высоких энергий, предоставляются пользователю предлагаемой системы. Более того, в системе предусмотрено и некоторое расширение этих возможностей. Однако важно понимать, что в нашем случае эти возможности работы с гистограммами предлагаются пользователю посредством отображений объектов гистограмм. Таким образом, такие функциональные возможности пакета HBOOK, как, например, изменения ширины каналов гистограмм и фитирование гистограмм здесь не обсуждаются, так как они связаны с созданием новых объектов. Это является привилегией компонентов.

3.7 Получение справочной информации

С использованием HTMLHelp Workshop [10] разработана справочная система для пользователей NiMax и разработчиков компонентов. Она оперирует с html-файлами документации. Справочная информация предоставляется как по самой системе NiMax (см. рис. 3.17), так и по отдельным компонентам (см. рис. 3.18).

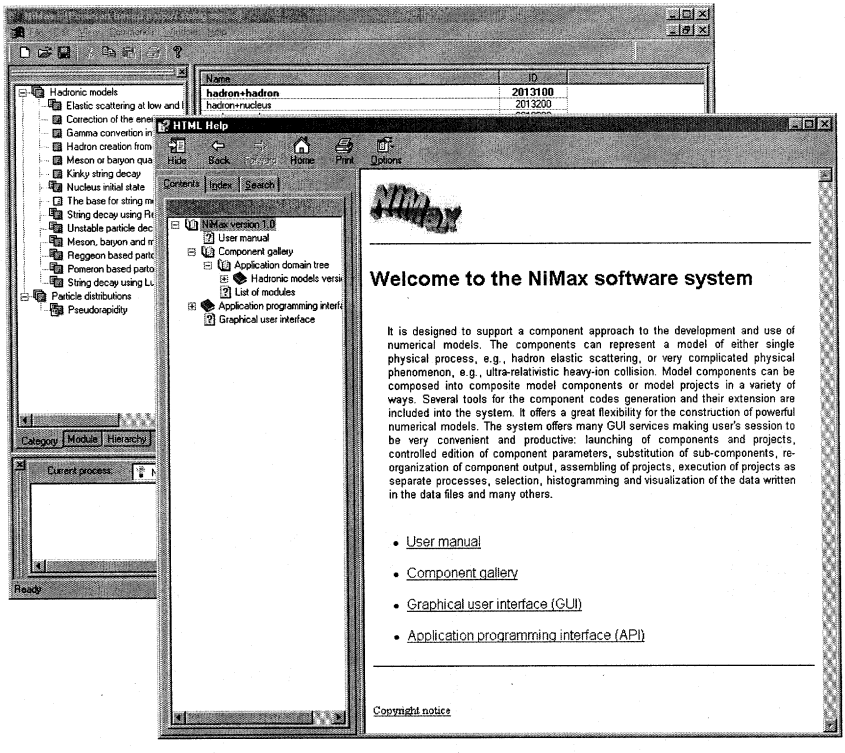

Рис. 3.17. Главное окно для получения справочной информации

В документации системы NiMax описана область применения системы, описаны основные её концепции, объясняется работа с компонентами и данными посредством графического интерфейса пользователя и т.д.

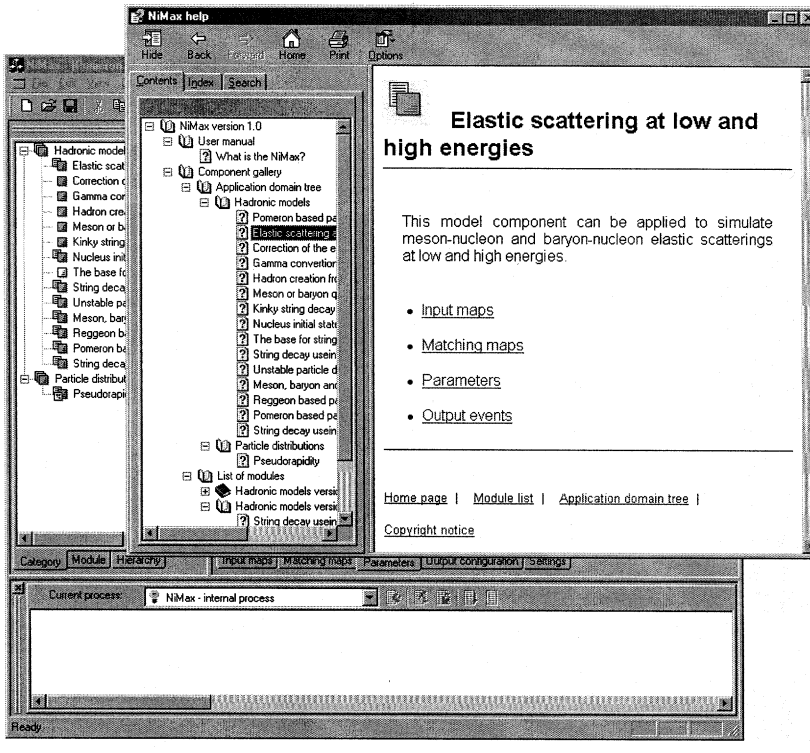

Рис. 3.18. Пример документации для отдельного компонента

В документации для отдельных компонентов содержится информация о параметрах компонентов, их картах входных данных, описаны конфигурации событий данных, которые могут генерироваться компонентами и т.д. Навигации по документации и поиск необходимой информации могут быть осуществлены по содержанию и по ключевым словам.

Заключение

Здесь был описан новый подход к разработке и использованию численных феноменологических моделей в физике высоких энергий. Это компонентный подход, при котором сложная численная модель собирается из более простых моделирующих компонентов, входящих в состав прикладных модулей.

Здесь сформулировано понятие стандартного компонента путем выделения в компоненте интерфейсной и алгоритмической частей. Любой компонент рассматривается как набор стандартных интерфейсов между его численным алгоритмом и внешним миром. Одним из преимуществ разделения «интерфейса» и «алгоритма» является возможность построения универсальной, независимой от конкретного алгоритма системы поддержки интерфейсной части: навигации по спискам компонентов, визуализации, редактирования, настройки и выполнения компонентов и т.д.

Здесь описана единица конфигурируемых численных данных: событие данных. Компонент может генерировать такие события и записывать их в файл данных для дальнейшего анализа и визуализации. Он может зачитывать такие события из файла данных или получать их от других компонентов.

В данном подходе предлагаются возможности объединения нескольких компонентов в сложный компонент или в компонентную сеть, а также различные средства разработки новых специфических компонентов. Это обеспечивает большую гибкость в создании реалистических численных моделей.

Каждый отдельный компонент может быть настроен для получения необходимых результатов его работы, а также различные версии численного алгоритма компонента могут быть взаимозаменяемы во время работы компонента. Это позволяет получить нужные свойства моделей без нового проектирования и перепрограммирования модели.

Для реализации компонентного подхода разработано программное обеспечение, объединенное в систему NiMax. Данная система представляет собой объектно-ориентированное программное обеспечение, написанное на языке программирования C++. На сегодняшний день не существует аналогичных систем для разработки и использования численных моделей в физике высоких энергий. Центральной частью системы является фреймворк, который контролирует внутренние системные процессы и обеспечивает взаимодействие между графическим пользовательским интерфейсом и остальными частями системы.

Фреймворк обеспечивает различные возможности для удобной и плодотворной работы пользователей системы через графический пользовательский интерфейс. Пользователь имеет возможность выбора требуемых компонентов и компонентных сетей. Он может проводить контролируемое редактирование параметров компонентов и подкомпонентов, замену подкомпонентов в сложных компонентах, реорганизацию вывода данных в компонентах и подкомпонентах. Пользователь может собирать компонентные сети и проводить их редактирование, а также их выполнение как независимых процессов. Он может осуществлять структурную, графическую и аналитическую селекцию сгенерированных и записанных в файл данных, а также построение гистограмм для этих данных, их визуализацию и многое другое.

Разработанная система включает различные средства, увеличивающие продуктивность работы разработчиков моделей. В систему включены различные средства автоматической генерации компонентных кодов и компонентной документации, а также библиотеки классов прикладных типов данных и классы транспортировки данных, которые облегчают разработку алгоритмических и интерфейсных частей компонентов. При создании этих средств предполагалось, что разработчик модели должен сконцентрироваться в работе над написанием численного алгоритма для данного компонента, и каждый алгоритм можно разрабатывать независимо от разработки алгоритмов для других компонентов. Функциональность компонентов может быть расширена с помощью механизма наследования, и компонентный код может быть использован при разработке других компонентов путем агрегирования.

Некоторые концепции, реализованные в системе, не имеют аналогов среди концепций реализованных в существующем программном обеспечении. Это механизм агрегирования компонентов другими компонентами. Данный механизм позволяет C++–программисту создавать сложные компоненты, используя коды готовых компонентов, без каких-либо ограничений по сравнению со стандартным C++–программированием. Для внешнего мира составной компонент выглядит так

же как, простой компонент. Вместе с тем пользователь этого компонента имеет доступ к любому его подкомпоненту и может изменить алгоритм компонента путем подмены подкомпонентов. Другой такой концепцией является механизм отбора данных в соответствии с заданной конфигурацией данных. Этот механизм позволяет организовать совместную работу нескольких компонентов путем передачи данных, имеющих сложную структуру. При этом от разработчика компонентов не требуется изучение каких-либо системных или компонентных классов, чтобы обеспечит совместную работу компонентов. Данный механизм также дает возможность структурного отбора данных, записанных в файл данных.

Идея о предопределенных событиях данных и предопределенных каналах данных, с одной стороны, облегчает визуализацию данных и их анализ, с другой стороны, дает возможность независимого наращивания пользовательского интерфейса и создания специализированных (для различных областей приложения) пользовательских интерфейсов.

Разработанный формат файла данных не только позволяет хранить огромный объем разнообразных структур данных, но и осуществлять быструю навигацию по этим данным (поиск и селекцию нужных данных). Необходимо отметить, что файл данных, отображения файла данных, а также связанные с файлом данных методы контроля и поиска могут рассматриваться как независимая система обработки большого объема данных, имеющая собственные применения.

В пакет включена справочная система, чтобы помочь пользователям. Она включает различные документы, содержащие описание как самой системы NiMax, так и описания разработанных компонентов. Поиск необходимой информации в этой системе может быть осуществлен по содержанию и по ключевым словам. Следует сказать, что справочная система может быть использована отдельно от системы NiMax, например, для обмена информацией.

Здесь показано применение созданной системы NiMax к определенному классу задач, а именно для разработки и использования МК-генераторов событий взаимодействия частиц и ядер в области физики высоких энергий. С помощью данной системы разработаны модули компонентов, которые дают возможность проводить численное моделирование взаимодействий частиц и ядер для большого набора снарядов и мишеней.

Созданная система может быть, прежде всего, использована для решения задач, связанных с разработкой и применением сложных многокомпонентных численных моделей генерирующих данные. Так мы начали разработку прикладных модулей, которые включают компоненты для проведения компьютерного моделирования транспортировки излучений в различных средах. Помимо этого система может быть использована для структурного, основанного на использовании конфигурации событий-данных, анализа данных, полученных из различных источников. А также она может использоваться для аналитического, основанного на использовании теоретических моделей, анализа этих данных. И, наконец, эти данные можно анализировать графически, т.е. с помощью визуализации.

Мы очень благодарны коллегам из Лаборатории высоких энергий Объединенного института ядерных исследований за сотрудничество. Мы хотим поблагодарить коллег с физического факультета университета города Ювяскула (Финляндия) за их интерес к разработке МК-генераторов событий взаимодействия частиц и ядер с использованием компонентного подхода, а также за многочисленные обсуждения.

Приложение: модули адронных моделей

1.1 Компоненты структуры частиц

1.1.1 Свойства частиц

Для описания свойств частиц были созданы таблицы, которые содержат характеристики частиц. В таблицы занесены характеристики лёгких кварков и антикварков (u, d, и s), а также характеристики дикварков и антидикварков, состоящих из лёгких кварков и антикварков. Созданы также таблицы для описания свойств глюона, фотона и лептонов ($e^{\pm}$, $\mu^{\pm}$ $\nu_{e,\mu}$), таблицы, содержащие характеристики $J^P = \frac{1^+}{2}, \frac{3^+}{2}$ барионов и барионных резонансов, а также характеристики J^{PC} - состояний мезонов: (0^{-+}) или (π, K, η и η') и (1^{--}) или (ρ, K^*, ω и ϕ). Индексы J, P и C обозначают спин, чётность и зарядовое сопряжение, соответственно. При составлении таблиц, предполагалось, что имеет место полная симметрия между барионнами и антибарионами.

В случае рождения резонанса, его масса m разыгрывалась в соответствии с распределением Брейта – Вигнера:

$$\mathrm{F(m)} = 2\pi \frac{\Gamma(m_R)}{(m_R - m)^2 + \Gamma^2(m_R)/4}, \qquad (1.1)$$

Характеристики частиц, значения полюсных масс m_R и полные ширины распадов $\Gamma(m_R)$ резонансов брались из [16], [17], [18], [19].

1.1.2 Кварковый состав частиц

В некоторых случаях, например, чтобы смоделировать возбуждение струны (см. описание компонента неупругого столкновения адронов), нам необходимо расщепить барионы (антибарионы) и выбрать определенные валентные кварки (антикварки) или дикварки (антидикварки), которые входили в их состав. Вероятности барионного (антибарионного) состояния, т.е. вероятности найти кварк (антикварк) и дикварк (антидикварк) с данными спином и изоспином определяются из $SU(6)$ - симметричных барионных (антибарионных) волновых функций. Эти вероятности приведены в таблице 1.1. Возникает аналогичная необходимость в расщеплении мезонов. В случае мезона он расщепляется на валентный кварк и антикварк. Для нейтральных смешанных мезонов розыгрыш кварк-антикварковых пар осуществляется в соответствии с вероятностями смешивания кварков

Таблица 1.1. Кварковый состав барионов. Индексы дикварков показывают их спин-изоспинное состояние

Тип бариона	Состав кварка
P	$1/3uu_{11}d+1/6ud_{11}u+1/2ud_{00}u$
N	$1/6ud_{11}d+1/2ud_{00}d+1/3dd_{11}u$
Σ^+	$1/3uu_{11}s+1/6us_{11}u+1/2us_{00}u$
Σ^0	$1/3ud_{11}s+1/12us_{11}d+1/4us_{00}d+1/12ds_{11}u+1/4ds_{00}u$
Σ^-	$1/3dd_{11}s+1/6ds_{11}d+1/2ds_{00}d$
Ξ^-	$1/6ds_{11}s+1/2ds_{11}s+1/3ss_{11}d$
Ξ^0	$1/6us_{11}s+1/2us_{00}s+1/3ss_{11}u$
Λ^0	$1/3ud_{00}s+1/4us_{11}u+1/12us_{00}d+1/4ds_{11}u+1/12ds_{00}u$

Δ^{++}	$uu_{11}u$
Δ^{+}	$1/3uu_{11}d+2/3ud_{11}u$
Δ^{0}	$2/3ud_{11}d+1/3dd_{11}u$
Δ^{-}	$dd_{11}d$
Σ^{*+}	$1/3uu_{11}s+2/3us_{11}u$
Σ^{*0}	$1/3ud_{11}s+1/3us_{11}d+1/3ds_{11}u$
Σ^{*-}	$1/3dd_{11}s+2/3ds_{11}d$
Ξ^{*0}	$1/3us_{11}s+2/3ss_{11}u$
Ξ^{*-}	$2/3ds_{11}s+1/3ss_{11}d$
Ω^{-}	$ss_{11}s$

Эти вероятности можно также использовать, чтобы определять рождение барионов (антибарионов) или барионных (антибарионных) резонансов при разрыве струн (см. описание компонентов для распада струн), предполагая, что дикварки (антидикварки) сохраняют свои спины и изоспины в течение реакции.

1.1.3 Конверсия фотона

Для расчета взаимодействия реального или виртуального фотона с нуклоном или ядром при высоких энергиях этот фотон конвертируется в векторный мезон или в систему мезонного типа [20] .

В случае фотон-нуклонного рассеяния используются следующие кинематические переменные: переменная Бьеркена x, которая определяется как $x = Q^2/2m\nu$, где Q^2, ν и m - это виртуальность фотона, энергия фотона и масса нуклона соответственно. Квадрат полной энергии γ–нуклонной системы определяется выражением $s = Q^2(1-x)/x+m^2$. Мы ограничим наше рассмотрение диапазоном малых величин x и значениями Q^2, которые много меньше s.

Обобщенная модель векторной доминантности (ОМВД) [20] предполагает, что виртуальный фотон сначала флуктуирует в промежуточное кварк-антикварковое состояние V массы M, которое затем взаимодействует с нуклоном N. Таким образом, полное фотон-нуклонное сечение может быть выражено формулой [5]:

$$\sigma_{\gamma N}(s,Q^2)=4\pi\alpha_{em}\int_{M_0^2}^{M_1^2}dM^2D(M^2)\left(\frac{M^2}{M^2+Q^2}\right)^2\left(1+\varepsilon\frac{Q^2}{M^2}\right)\sigma_{VN}(s,Q^2), \qquad (1.2)$$

где интегрирование по M^2 выполняется в интервале между $M^2_0 = 4m^2_\pi$ и $M^2 = s$. Здесь $\alpha_{em} = e^2/4\pi = 1/137$ и распределение кварк-антикварковой системы по квадратичной массе определяется равенством

$$D(M^2)=\frac{R_{e^+e^-}(M^2)}{12\pi^2M^2}, \qquad (1.3)$$

и

$$R_{e^+e^-}(M^2)=\frac{\sigma_{e^+e^-\to hadrons}(M^2)}{\sigma_{e^+e^-\to\mu^+\mu^-}(M^2)}\approx 3\sum\nolimits_f e_f^{\,2}, \qquad (1.4)$$

где e^2_f – это квадрат заряда кварка с ароматом f, а e - отношение между потоками продольно и поперечно поляризованных фотонов.
Таким образом, численный алгоритм конверсии фотона можно описать следующим образом:

1. Разыграть массу M^2 кварк-антикварковой флуктуации в соответствии с уравнением (1.3) при заданных s и Q^2;

2. Разыграть аромат пары в соответствии со статистическими весами: $\omega_{u\bar{u}} = 1/2, \varpi_{d\bar{d}} = 1/4, \omega_{s\bar{s}} = 1/4$, которые получены из формулы (1.4).

1.2 Компоненты сечений взаимодействия адронов

Сечения взаимодействия адронов зависят от типов сталкивающихся частиц и полной энергии в системе их центра масс $\sqrt{s_{i,j}} = \sqrt{(p_i + p_j)^2}$, где $p_i = (E_i, \mathbf{p}_i)$, $p_j = (E_j, \mathbf{p}_j)$ обозначают 4- импульсы частицы-снаряда i с массой $m_i = \sqrt{E_i^2 - p_i^2}$ и частицы-мишени j с массой $m_j = \sqrt{E_j^2 - p_j^2}$ соответственно.

При вычислении сечений взаимодействия адронов используются таблицы экспериментально известных значений сечений, а также параметризации экспериментально известных значений сечений алгебраическими функциями. Сечения взаимодействия адронов вычисляются также в рамках модели составных кварков и в рамках померонной эйкональной модели. Заметим, что для некоторых пар сталкивающихся адронов их сечения взаимодействия могут быть получены путем пересчета экспериментально известных сечений с использованием общих принципов, таких, как изотопическая инвариантность и принцип детального баланса [1].

1.2.1 Модель составных кварков

В модели составных кварков (МСК) сечение зависит только от кваркового состава сталкивающихся адронов [22], [23], [24]:

$$\sigma_{tot}^{AQM} = 40\left(\frac{2}{3}\right)^{n_M}(1-0{,}4x_1^s)(1-0{,}4x_2^s) \qquad (1.5)$$

и

$$\sigma_{el}^{AQM} = 0{,}039\sigma_{tot}^{\frac{2}{3}}, \qquad (1.6)$$

где n_M - число соударяющихся мезонов и x_i^s - отношение числа странных кварков к числу нестранных кварков в i–м адроне. Таким образом, МСК можно использовать, чтобы получить неизвестные сечения путем пересчета известных сечений.

1.2.2 Померонная эйкональная модель

Сечение адрон-адронного взаимодействия как функция квадрата полной энергии сталкивающихся частиц в их системе центра масс s может быть рассчитано путем интегрирования вероятности адрон-нуклонного взаимодействия $p_{ij}(b_{ij}^2, s)$ по прицельному параметру столкновения b. В частности, полное и неупругое адрон-адронные сечения можно рассчитать, используя выражения:

$$\sigma_{tot} = 2\pi \int_0^\infty b db p_{ij}^{tot}(b_{ij}^2, s) = \sigma_P f(\frac{z}{2}) \qquad (1.7)$$

и $$\sigma_{in} = 2\pi \int_0^\infty b db p_{ij}^{in}(b_{ij}^2, s) = \sigma_P f(z), \qquad (1.8)$$

где $$\sigma_P = 4\pi z(s)\lambda(s), \qquad (1.9)$$

и $$f(z) = \sum_{\nu=1}^{\infty} \frac{(-z)^{\nu-1}}{\nu\nu!}. \qquad (1.10)$$

В теории Редже–Грибова [25], [26] вероятность для неупругих столкновений адронов i и j как функция квадрата разности прицельного параметра $b_{ij}^2 = (\vec{b}_i - \vec{b}_j)^2$ и квадрата энергии s записывается как:

$$p_{ij}(\vec{b}_i - \vec{b}_j, s) = c^{-1}[1 - \exp\{-2u(b_{ij}^2, s)\}] = \sum_{n=1}^{\infty} p_{ij}^{(n)}(\vec{b}_i - \vec{b}_j, s), \qquad (1.11)$$

где

$$p_{ij}^{(n)}(\vec{b}_i - \vec{b}_j, s) = c^{-1} \exp\{-2u(b_{ij}^2, s)\} \frac{[2u(b_{ij}^2, s)]^n}{n!} \qquad (1.12)$$

есть вероятность иметь n померонных разрезов или вероятность рождения $2n$ струн в неупругом адрон-адронном столкновении, если предположить, что каждый померонный разрез может быть замещён двумя струнами.

Эти вероятности записаны в терминах (эйкональной) амплитуды упругого рассеяния адронов посредством обмена помероном:

$$u(b_{ij}^2, s) = \frac{z(s)}{2} \exp[-b_{ij}^2 / 4\lambda(s)]. \qquad (1.13)$$

Величины $z(s)$ и $\lambda(s)$ определяются параметрами траектории Померанчука: $\alpha_P' = 0{,}25\,\text{ГэВ}^{-2}$ и $\alpha_P(0) = 1{,}0808$ и параметрами вершины взаимодействия померона и адрона: R_P^2 и γ_P:

$$Z(s) = \frac{2c\gamma_P}{\lambda(s)} (s/s_0)^{\alpha_P(0)-1}, \qquad (1.14)$$

$$\lambda(s) = R_P^2 + \alpha_P' \ln(s/s_0), \qquad (1.15)$$

соответственно, где s_0 - это параметр размерности.

В выражения (1.14, 1.15) введён так называемый коэффициент ливневого усиления c, определяющий вклад одновершинной дифракционной диссоциации [26]. Таким образом, вероятность одновершинной дифракционной диссоциации в адронном взаимодействии может быть рассчитана следующим образом:

$$p_{ij}^d(,s) = \frac{c-1}{c}[p_{ij}^{tot}(\vec{b}_i - \vec{b}_j, s) - p_{ij}(\vec{b}_i - \vec{b}_j, s)], \qquad (1.16)$$

где

$$p_{ij}^{tot}(\vec{b}_i - \vec{b}_j, s) = (2/c)[1 - \exp\{-u(b_{ij}^2, s)\}]. \qquad (1.17)$$

Параметры вершин взаимодействия померона с адронами и величины коэффициента ливневого усиления определялись путем фитирования экспериментальных значений полного, упругого и дифракционного сечений адрон-нуклонных взаимодействий при разных энергиях.

Для нуклон-нуклонного, пион-нуклонного и каон-нуклонного столкновений были определены следующие значения параметров померонной вершины и

коэффициента ливневого усиления: $R_P^{2N} = 3.56$ ГэВ$^{-2}$ и $\gamma_P^N = 3.96$ ГэВ$^{-2}$, $s_0^N = 3{,}0$ ГэВ$^{-2}$, $c^N = 1{,}4$; $R_P^{2\pi} = 2{,}56$ ГэВ$^{-2}$ и $\gamma_P^\pi = 2{,}17$ ГэВ$^{-2}$, $s_0^\pi = 1{,}5$ ГэВ$^{-2}$, $c^\pi = 1{,}6$ ГэВ$^{-2}$; $R_P^{2K} = 1{,}96$ ГэВ$^{-2}$, $\gamma_P^K = 1{,}92$ ГэВ$^{-2}$, $s_0^K = 2{,}3$ ГэВ$^{-2}$, $c^\pi = 1{,}8$. Эти значения можно использовать для описания полного, упругого и дифракционного сечений.

1.2.3 Сечение одновершинной дифракционной диссоциации

Зависимость сечения взаимодействия протонов от квадрата их энергии s в случае одновершинной дифракционной диссоциации может быть параметризована выражением

$$\sigma_{SD}(s) = (0{,}68 \pm 0{,}05)\left(1 + \frac{36 \pm 8}{s}\right)\ln(0{,}6 + 0{,}1s). \qquad (1.18)$$

1.2.4 Сечение аннигиляции барионов

Чтобы рассчитать сечение барионно-антибарионной аннигиляции, выбран метод, описанный в работе [24]. При этом подходе используется одна и та же энергетическая зависимость для всех барион-антибарионных сечений и принимается в расчёт различный кварковый состав барионов с поправочным коэффициентом, полученным из модели составных кварков [22], [23]:

$$\sigma_{\bar{B}B}(\sqrt{s}) = \frac{\sigma_{B\bar{B}}^{AQM}}{\sigma_{N\bar{N}}^{AQM}}\sigma_{\bar{N}N}(\sqrt{s}), \qquad (1.19)$$

где сечение антипротон-протонной аннигиляции вычисляется с использованием выражения [7]:

$$\sigma_{\bar{p}p}^{ann}(s) = \sigma_0\frac{s_0}{s}\left[\frac{A^2 s_0}{(s - s_0)^2 + A^2 s_0} + B\right] \qquad (1.20)$$

со следующими значениями параметров: σ_0 = 120 мб, $s_0 = 4m_N^2$, A = 50 МэВ и $B = 0{,}6$.

1.3 Компоненты распада частиц

1.3.1 Моделирование распада частиц

Каналы распада частиц разыгрываются с использованием их распадных ширин $\Gamma(m)$, зависящих от масс m частиц. Предполагается, что все частицы являются неполяризованными. Если среди продуктов распада есть резонанс, то его масса определяется в соответствии с распределением Брейта – Вигнера, в котором значения полюсной массы и ширины распада резонанса берутся из таблиц [17]. Если выходной канал содержит две, три или четыре частицы, то розыгрыш 4-импульсов частиц продуктов распада производится в соответствии с разрешенным N-частичным (N = 2,3,4) фазовым объемом. Для определения 4-импульсов частиц продуктов распада используется алгоритм Копылова [29].

Следует подчеркнуть, что алгоритм Копылова позволяет учесть не только разрешенные фазовые объемы для продуктов распада, но и матричные элементы распадов.

1.3.2 Численный алгоритм N–частичного распада

Вероятность n–частичного состояния можно записать в виде

$$dW = \prod_{i=1}^{n} \frac{d^3 p_i}{2\omega_i} \delta(\sum_{i=1}^{n} p_i - P_n) \mid M \mid^2, \qquad (1.21)$$

где P_n = (P_n,En) – это 4-импульс распадающейся частицы и M(p_1, p_2, …, p_n) – это амплитуда реакции, которая зависит от 4-импульсов p_i продуктов распада. Мы знаем сумму масс рожденных частиц:

$$\mu_n = \sum_{i=1}^{n} m_i \qquad (1.22)$$

и их кинетическую энергию в системы центра масс

$$T_n = M_n - \mu_n, \qquad (1.23)$$

где $M_n = \sqrt{E_n^2 - P_n^2}$ - это инвариантная масса системы.

Выполняя интегрирование вероятности, описанной выражением (1.21), можно получить различные распределения частиц. Интегрирование по всему набору переменных S кратко записывается следующим образом:

$$W = \int_D \Phi(S) dS, \qquad (1.24)$$

где $\Phi(S)$ можно рассматривать как вес события с рождением n-частиц.

Алгоритм Копылова основывается на выборе нового набора переменных S_1: T_k, η_k, и φ_k вместо переменных 4-импульсов частиц для того, чтобы выполнить интегрирование для dW. Область интегрирования D_1 в новых переменных является довольно простой. Так, значения угловых переменных лежат в пределах $1 \geq \eta_k \geq -1$ и $2\pi\phi_k \geq 0$, где $k = n,n - 1,\ldots,2$. Значения кинетических энергий систем с определенным числом частиц располагаются в определенном порядке: $T_n \geq T_{n-1} \geq \ldots \geq T_2 \geq T_1 = 0$. Последующая замена переменных T_k, η_k, и φ_k набором величин S_2: ξ_k, γ_k, β_k, значения которых ограничены интервалом между 0 и 1, превращает область интегрирования D_1 в единичный гиперкуб D_2. Таким образом, мы имеем:

$$W \sim \int_0^1 \ldots \int_0^1 \prod_2^n d\beta_k d\gamma_k \prod_2^{n-1} d\xi_k \xi^{\frac{3k-5}{2}}{}_k \sqrt{1-\xi_k} |M|^2 . \qquad (1.25)$$

Два набора значений S_1 и S_2 связаны друг с другом соотношениями:

$$T_k = T_{k+1}\xi_k, k = n-1,\ldots,2$$
$$\eta_k = 2\beta_k - 1, k = n,\ldots,2 \qquad (1.26)$$

Вес события $\Phi(S_1)$ [2] может быть записан как

$$\Phi(S_1) = \frac{\pi^{\frac{3}{2}(n-1)} T_n^{\frac{3n-5}{2}}}{2M_n \Gamma(\frac{3}{2}(n-1))} \prod_{k=2}^{n} \frac{\hat{p}_k}{\sqrt{T_k - T_{k-1}}} |M|^2 , \qquad (1.27)$$

где $\Gamma(x)$ - это гамма-функция и $\hat{p}_k$ - 4-импульс k-й частицы в системе покоя 0_k, где $T_k = 0$.

Как можно видеть из уравнения (1.25), переменные ξ_k, γ_k, β_k могут рассматриваться как случайные переменные, распределённые в между 0 и 1. Таким образом, чтобы сгенерировать импульсы частиц, мы можем применить следующую рекурсивную процедуру [29].

1. Разыграть ξ_{k-1} в соответствии с распределением

$$F(\xi_k) = \xi_k^{\frac{3k-5}{2}}\sqrt{1-\xi_k}\,, \qquad (1.28)$$

где функции $F(\xi_k)$ имеют максимальные значения $F_{\max} = F((3k-5)/(3k-4))$, и вычислить $T_{k-1} = T_k\xi_{k-1}$. Если $k=2$, то T_{k-1} должно быть установлено в 0 по определению системы покоя $0_{k\text{-}1}$.

Функции $F(x)$ - это быстроменяющиеся функции при больших значениях k. Для больших k можно применить другую методику разыгрывания с табулированием [29]. В этом случае для случайного числа α_{k-1}, выбранного в соответствии с равномерным распределением между 0 и 1, необходимо решить уравнение

$$C_{k-1}(\xi_{k-1}) = \alpha_{k-1}, \qquad (1.29)$$

где

$$C_k(\xi_k) = \frac{\int_0^{\xi_k} d\xi^{\frac{3k-5}{2}}{}_k\sqrt{1-\xi_k}}{\int_0^1 d\xi^{\frac{3k-5}{2}}{}_k\sqrt{1-\xi_k}} \qquad (1.30)$$

и найти значение ξ_{k-1}. $C_k(\xi_k)$ – табулированные гладкие функции. Чтобы найти значения функций в промежуточных точках, используется линейная интерполяция.

2. Рассчитать полную энергию

$$\hat{\omega}_k = (M^2{}_k + m^2{}_k - M^2{}_{k-1})/2M_k \qquad (1.31)$$

и абсолютное значение 3-импульса

$$\hat{p}_k = \sqrt{\hat{\omega}_k - m^2{}_k} \qquad (1.32)$$

для k-й частицы в системе покоя 0_k, где

$$\mu_{k-1} = \mu_k - m_k \qquad (1.33)$$

и

$$M_{k-1} = \mu_{k-1} - T_{k-1}\,. \qquad (1.34)$$

3. Разыграть направление k-го импульса $\hat{p}_k$, т.е разыграть азимутальные и полярные углы согласно выражениям $\varphi_k = 2\pi\gamma_k$ и $\cos\theta_k = 2\beta_k - 1$, соответственно. Величины γ_k и β_k - это равномерно распределённые между 0 и 1 случайные числа.

4. Рассчитать энергию k-й частицы:

$$\omega_k = \frac{E_k\hat{\omega} + P_k\hat{p}_k}{M_k} \qquad (1.35)$$

и её 3-импульс

$$p_k = \hat{p}_k + \hat{P}_k\frac{\omega_k + \hat{\omega}_k}{E_k + M_k} \qquad (1.36)$$

в системе наблюдателя.

5. Рассчитать полную энергию E_{k-1} и 3-импульс P_{k-1} системы с $k-1$ числом частиц:

$$E_{k-1} = E_k - \omega_k\,,$$

$$P_{k-1} = P_k - p_k\,.$$

Для $k=2$ можно проверить, что $m_1 = \sqrt{E^2{}_1 - P^2{}_1}$, где $p_1 = P_1$ и $\omega_1 = E_1$.

В результате описанной рекурсивной процедуры становятся известны 4-импульсы всех продуктов распада, и можно рассчитать вес события по формуле (1.27).

Наконец, если $\Phi(S_1) < \xi\Phi_{max}$, где ξ–случайное число, равномерно распределенное между 0 и 1, и Φ_{max} - это максимально возможное значение из совокупности величин $\Phi(S_1)$, то данное событие распада принимается.

Следует заметить, что в нерелятивистском случае и для $|M|^2 = 1$ нет необходимости браковать события распада, т.е. проверять, что $\Phi(S_1) < \xi\Phi_{max}$ [2].

Следует также заметить, что описанный выше алгоритм может быть использован для вычисления объёма релятивистского фазового пространства W, т.е. для выполнения интегрирования по весу события (1.24) при $|M|^2 = 1$. Для того чтобы выполнить это интегрирование, необходимо провести усреднение весов для N событий, где $N \to \infty$. Этот метод интегрирования эффективен также при $|M|^2 \neq 1$, если $|M|^2$ не имеет полюсного поведения. Таким образом, алгоритм Копылова позволяет проводить расчеты ширин распада и получать различные распределения частиц.

Этот алгоритм позволяет также моделировать рождение ядерных фрагментов в результате распада возбуждённого атомного ядра.

1.4 Компоненты распада струн

Цветовая струна натягивается между разлетающимися и несущими цветовые заряды партонами: кварком и антикварком или кварком и дикварком или дикварком и антидикварком. Зная продольные $p_{3i} = p_{zi}$ и поперечные $p_{1i} = p_{xi}$, $p_{2i} = p_{yi}$ импульсы партонов, а также значения их энергий $p_{0i} = E_i$, где i =1,2, можно вычислить квадрат массы системы, которая в дальнейшем называется просто струной, состоящей из собственно цветовой струны и партонов на её концах:

$$M_S^2 = p^\mu p_\mu = p_0^2 - p_1^2 - p_2^2 - p_3^2, \qquad (1.39)$$

где $p_\mu = p_{\mu 1} + p_{\mu 2}$ μ - это 4-импульс струны и $\mu = 0,1,2,3$.

Компоненты распада струн можно использовать для моделирования распадов продольных $q-\overline{q}$, $qq-\overline{qq}$, $q-qq$ и $\overline{q}-\overline{qq}$ струн или кинковых $q-g-\overline{q}$, $qq-g-\overline{qq}$, $q-g-qq$ и $\overline{q}-g-\overline{qq}$ струн. Продуктами распада струн являются адроны или адронные резонансы, принадлежащие к скалярным и векторным мезонным нонетам, и к барионным (антибарионным) октетам и декуплетам.

Удобно проводить моделирование распада струны в системе её покоя, когда партоны на концах струны движутся вдоль оси z в противоположных направлениях. Поэтому, когда моделирование распада струны закончено, следует выполнить обратное преобразование Лоренца с использованием скорости $-\beta_\mu$ для 4-импульсов и 4-координат, рожденных в результате распада адронов, а также обратное вращение 3-импульсов и 3-координат адронов, используя матрицы R_{kl}^{-1} (см. ниже).

1.4.1 Моделирование распада кластеров

Моделирование распада кластеров (струн с малой массой) основано на результатах работ [30], [31]. При моделировании этого процесса предполагается,

что каждый кластер с массой M_c распадается изотропно в системе его покоя на адронную пару с массами m_1 и m_2 и спинами J_1 и J_2 путем рождения из вакуума кварк - антикварковой или дикварк - антидикварковой пары с массой кварка (дикварка) m_q. Для адрона, имеющего в своём составе $u-$, $d-$ и $s-$ кварки, разрешены $J^P = 0^+, 1^\pm$ мезонное и $J^P = \frac{1}{2}^+, \frac{3}{2}^+$ барионное состояния. Канал распада выбирается в соответствии с вероятностью:

$$P_{decay} = P_{flavor}(M_c, m_q) P_{spin}(J_1, J_2) P_{ps}(M_c, m_1, m_2), \quad (1.40)$$

где первый множитель, связанный с ароматом вакуумной пары, и определяется выражением:

$$P_{flavor}(M_c, m_q) = 1 + \frac{2m^2_q}{M^2_c}\sqrt{1 - \frac{4m^2_q}{M^2_c}}, \quad (1.41)$$

второй – это спиновый множитель, который имеет вид

$$P_{spin}(J_1, J_2) = (2J_1 + 1)(2J_2 + 1), \quad (1.42)$$

и последний множитель связан с фазовым объемом:

$$P_{ps} = \frac{\sqrt{M^4_c + m^4_1 + m^4_2 - 2(M^2_c m^2_1 + M^2_c m^2_2 + m^2_1 m^2_2)}}{M^2_c}. \quad (1.43)$$

1.4.2 Моделирование распада продольных струн

Распад струны моделируется по итерационной схеме вида

струна → адрон + новая струна, (1.44)

т.е. распад струны на новую струну и адрон происходит путем рождения вакуумной кварк - антикварковой (дикварк - антидикварковой) пары. При этом значения вероятностей рождения кварковой вакуумной пары с разными ароматами и значение вероятности рождения дикварковой вакуумной пары находятся в отношении

$$u : d : s : qq = 1 : 1 : 0.35 : 0.1. \quad (1.45)$$

При моделировании распада струны имеется возможность учесть расщепление дикварков.

Адрон формируется на одном из концов струны, который выбирается случайным образом. Кварковый состав адронов определяет их типы и заряды. При выбранной схеме распада мы учитываем, что адроны могут рождаться не только в основных состояниях, но и низколежащих возбужденных состояниях. Если для барионов их кварковый состав не позволяет однозначно определить, является ли это состояние октетным или декуплетным, то выбор состояния производится с использованием вероятностей, которые определяются спином или изоспином дикварка (антидикварка) (см. описание компонентов структуры частиц).

В случае мезонов их принадлежность к мультиплету должна быть также определена прежде, чем определен тип мезона. Вероятность выбрать определённый мультиплет зависит от спина мультиплета.

В случае рождения резонансов их массы m определяются из распределений Брейта–Вигнера, в которых полюсные массы и полные ширины распадов взяты из таблиц [17].

Вакуумная пара имеет нулевой поперечный импульс, который определяется суммой равных и противоположно направленных поперечных импульсов кварка и антикварка (дикварка и антидикварка). Поперечный импульс отдельного кварка (дикварка) из этой пары разыгрывается случайным образом в соответствии с вероятностью (1.85) с параметром a = 0,55 ГэВ$^{-2}$. Поперечный импульс рожденного адрона $\mathbf{p_t}$ определяется суммой поперечных импульсов его составляющих.

Функция фрагментации $f^h(z,p_t)$ определяет вероятность того, что адрон, имеющий поперечный импульс $\mathbf{p_t}$, может унести определенное значение переменной светового конуса $z = z^{\pm} = (E^h \pm p_z^h)/(E^q \pm p_z^q)$, где E^h и E^q - это значения энергий адрона и фрагментированного кварка соответственно. p_z^h и p_z^q - это значения продольных импульсов адрона и фрагментирующего кварка, соответственно. Значения z ограничены пределами: $z_{\min}^{\pm}$ и $z_{\max}^{\pm}$, которые зависят от массы рожденного адрона, поперечных масс его составляющих и массы струны.

Одна из наиболее популярных функций фрагментации, которая была предложена в Лунд-модели [4], и имеет вид:

$$\mathrm{f}^h(z,p_t) \sim \frac{1}{z}(1-z)^a \exp\left[-\frac{b(m^2{}_h + p^2{}_t)}{z}\right]. \qquad (1.46)$$

Эту функцию фрагментации можно использовать при моделировании распада струны. Можно также использовать функции фрагментации, полученные в работе [5]:

$$\mathrm{f}^h{}_q(z,p_t) = \left[1+\alpha^h{}_q(\langle p_t\rangle)\right](1-z)^{\alpha^h{}_q(\langle p_t\rangle)}. \qquad (1.47)$$

Преимущество последних функций по сравнению с Лунд-функцией фрагментации заключается в том, что они имеют правильное трехреджеонное ассимптотическое поведение при z→1 [33].

Для того чтобы рассчитать значение времени формирования адрона и значения его продольной координаты, предполагалось, что мы имеем дело с распадом (1+1)-струны с массой M_S и коэффициентом натяжения k. В системе её покоя струна распадается на адроны, каждый из которых имеет энергию E_i и значение продольного импульса p_{zi}. Вводя переменные светового конуса для адронов: $p_i^{\pm} = E_i \pm p_{iz}$ и нумеруя точки разрывов струны последовательно справа налево, мы получаем $p_0^+ = M_S, p_i^+ = k(z_{i+1}^+ - z_i^{+})$ и $p_i^- = kx_i^-$.

Если пространственно-временная точка формирования адрона определяется как точка пересечения траекторий кварка и антикварка, которые образуют адрон (так называемая «уо-уо» точка формирования адрона [4]), то мы получим формулы расчета времени формирования адрона:

$$t_i = \frac{1}{2k}[M_S - 2\sum_{j=1}^{i-1} p_{zj} + E_i - p_{zi}] \qquad (1.48)$$

и его продольной координаты:

$$z_i = \frac{1}{2k}[M_S - 2\sum_{j=1}^{i-1} E_j + p_{zi} - E_i]. \qquad (1.49)$$

1.4.3 Моделирование распада кинковых струн

Численный алгоритм моделирования процесса кинковой струны, например, типа $q-g-\overline{q}$, состоит в выполнении двух шагов [34]:

- расщепить кинк-глюон $g \to q_1\overline{q}_1$ и создать две продольные струны;

- смоделировать независимый распад эти продольных струн: $q\overline{q}_1 \to h$ и $\overline{q}q_1 \to h$ в адроны h;

Моделирование распадов других типов кинковых струн производится аналогичным образом.

Выбор $q_1\overline{q}_1$–пары производится аналогично (используются те же вероятности для определения ароматов кварков и такое же распределение кварков по поперечному импульсу p_t), как и выбор вакуумных $q\overline{q}$ –пар в процессе распада продольной струны. Функция расщепления $g \to q_1\overline{q}_1$:

$$f_g^q(z) = z^2 + (1-z)^2, \tag{1.50}$$

где $z = \dfrac{E_q + p_q^z}{E_g + p_g^z}$ была получена в работе [35].

1.4.4 Преобразования струн

Как уже говорилось, распад струны моделируется в системе покоя струны. В этой системе партоны, находящиеся на концах струны, движутся в противоположных направлениях вдоль оси z. Мы можем преобразовать импульсы партонов в систему покоя струны, используя преобразования Лоренца:

$$p_{\mu 1,\mu 2} \to L_\mu p_{\mu 1,\mu 2}, \tag{1.51}$$

где

$$L_0 = \beta^\nu p_\nu \tag{1.52}$$

и

$$L_k = \beta_k p_0 + \sum_{l=1}^{3}(\delta_{lk} + \frac{\beta_k \beta_l}{1+\beta_0})p_i . \tag{1.53}$$

β_μ определяется выражением

$$\beta_\mu = \frac{p_\mu}{M_S}. \tag{1.54}$$

Ориентация струны по отношению к оси z определяется двумя углами Эйлера, которые вычисляются с помощью выражений:

$$\cos\alpha = \frac{p_{32}}{\sqrt{p_{22}^2 + p_{32}^2}} \tag{1.55}$$

и

$$\cos\beta = \frac{\sqrt{p_{22}^2 + p_{32}^2}}{\sqrt{p_{12}^2 + p_{22}^2 + p_{32}^2}}. \tag{1.56}$$

Затем, вращением струны:

$$p_{k1,k2} \to R_{kl} p_{l1,l2}, \tag{1.57}$$

можно получить движение партонов вдоль оси z. Матрица R_{kl} определяется следующим образом:

$$R_{kl} = \begin{vmatrix} \cos\beta & -\sin\alpha\cos\beta & -\cos\alpha\sin\beta \\ 0 & \cos\alpha & -\sin\alpha \\ \sin\beta & -\sin\alpha\cos\beta & -\cos\alpha\cos\beta \end{vmatrix} \quad (1.58)$$

и $k,l = 1,2,3$.

1.5 Компоненты упругого рассеяния частиц

1.5.1 Упругое рассеяние адронов

С помощью модели упругого рассеяния адронов можно предсказывать конечные состояния их упругих столкновений.

При высоких энергиях угловое распределение адронов вычисляется в рамках померонной эйкональной модели [36]. Это распределение даёт возможность разыгрывать углы рассеяния при начальных энергиях $\sqrt{s} > 2$ ГэВ. Для более низких начальных энергий розыгрыш углов рассеяния проводится на основе параметризации экспериментальных данных для нуклон-нуклонных взаимодействий.

При высоких энергиях угловые распределения упругого рассеяния адронов хорошо описываются выражением

$$\frac{dW(s)}{dt} = B_{el}(s)\exp[B_{el}(s)t], \quad (1.59)$$

где $t = -2p_{cm}^2(1-\cos\theta)$ - это квадрат переданного 4-импульса, который выражается через абсолютное значение импульса частицы в системе их центра масс p_{cm} и через угол рассеяния частицы в этой системе θ. Значения квадрата переданного 4-импульса лежат в интервале $t_{\min} \le t \le t_{\max}$, где $t_{\max} = 0$ и $t_{\min} = -4p_{cm}^2$. s - это квадрат полной энергии частиц в системе центра масс сталкивающихся адронов. Зависимость параметра наклона $B_{el}(s)$ от квадрата полной энергии сталкивающихся частиц определена в теории Редже–Грибова следующим образом:

$$B_{el}(s) = 2\lambda(s), \quad (1.60)$$

где

$$\lambda(s) = R_P^2 + \alpha'_P \ln(s/s_0). \quad (1.61)$$

Как уже объяснялось, при описании компонентов для вычисления сечений взаимодействия адронов померонные параметры R_P^2, α'_P и s_0 определяются путем фитирования полного, упругого, дифференциального упругого и дифракционного сечений для нуклон-нуклонного, пион-нуклонного и каон-нуклонного взаимодействий при разных значениях s.

Для моделирования упругого рассеяния используется распределение (1.59), чтобы разыграть t и рассчитать угол рассеяния θ. Выражение для наклона (1.60) применяется в том случае, когда начальная энергия $\sqrt{s} > \sqrt{s_{th}}$, где пороговая энергия $\sqrt{s_{th}}$ = 1,6 ГэВ для пион-нуклонных столкновений, $\sqrt{s_{th}}$ = 2,0 ГэВ для каон-нуклонных столкновений и $\sqrt{s_{th}}$ = 2,5 ГэВ для нуклон-нуклонных столкновений.

Для того чтобы разыграть $cos\ \theta$ при более низких энергиях, применяется параметризация:

$$B_{el}(s) = \frac{2.15R_P^2[3.65(\sqrt{s} - m_1 - m_2)]^6}{1 + [3.65(\sqrt{s} - m_1 - m_2)]^6}, \qquad (1.62)$$

где m_1 and и m_2 – это массы соударяющихся частиц. Эта форма углового распределения была получена с помощью модифицирования параметризации

1.5.2 Упругое рассеяние глюонов

[37] для того, чтобы обеспечить плавное изменение формы угловых распределений при переходе из области высоких энергий, где эта форма определяется наклонами (1.60), полученными в теории Редже –Грибова.

Необходимость в моделировании упругого столкновения глюонов возникает при моделировании рождения адронных струй или кинковых струн в адронных и ядерных неупругих столкновениях.

Розыгрыш углов вылетающих глюонов θ в их системе центра масс производится с помощью дифференциального сечения глюон-глюонных столкновений, полученного в рамках квантовой хромодинамики (КХД) в Борновском приближении [49]:

$$\frac{d\sigma_{gg}(\hat{s})}{d\cos\theta} = \frac{9\pi\alpha_s^2(Q^2)}{32s} \frac{(3+\cos^2\theta)^3}{(1-\cos^2\theta)^2}, \qquad (1.63)$$

где $\alpha_s(Q^2)$–КХД бегущая константа связи. Значение $\hat{s} = Q^2$ должно быть достаточно большим, чтобы рассеянные глюоны имели квадраты поперечных импульсов выше выбранного порога $Q_0{}^2 = 2$ ГэВ^2. Это условие приводит к ограничениям на угол рассеяния: $-z_0 < \cos\theta < z_0$, где

$$z_0 = \sqrt{1 - \frac{4Q_0^2}{\hat{s}}}. \qquad (1.64)$$

При упругом рассеянии партонов угол рассеяния можно выразить через квадрат переданного 4-импульса $\hat{t}$:

$$\cos\theta = 1 + \frac{2\hat{s}\hat{t}}{[\hat{s} - (m_i + m_j)^2(\hat{s} - (m_i - m_j)^2)]}, \qquad (1.65)$$

где m_i и m_j обозначают партонные массы. Как правило, розыгрыш $\hat{t}$ намного проще, чем генерация $cos\ \theta$.

1.5.3 Численный алгоритм упругого рассеяния

Для розыгрыша $cos\ \theta$ при упругом рассеянии используется следующая численная (Монте-Карло) процедура:

1. Разыграть квадрат переданного импульса в соответствии с выражением

$$t = \frac{1}{B_{el}(s)} \ln\xi, \qquad (1.66)$$

где ξ - это случайные числа, которые равномерно распределены в интервале между 0 и 1, $t_{\min} \le t \le t_{\max}$.

2. Вычислить

$$\cos\theta = 1 - \frac{t}{2p_{cm}^2}. \qquad (1.67)$$

1.6 Компоненты аннигиляции частиц

Эти компоненты позволяют их пользователям моделировать специальный тип неупругих взаимодействий адронов, когда антикварк (антидикварк) из одного сталкивающегося адрона аннигилирует с подходящим по аромату кварком (дикварком) из другого сталкивающегося адрона. Результатом аннигиляции партонов является возбуждение барионной (с кварком и дикварком на концах), антибарионной (с антикварком и антидикварком на концах) и мезонной (с кварком и антикварком на концах) струн.

В частности, для компонента, моделирующего аннигиляцию барионов при различных начальных энергиях, предполагается, что процесс аннингиляции барионов идет через процесс аннигиляции дикварков. Этот компонент дает возможность рассчитывать конечные состояния данной реакции, т.е. рожденные адроны с нулевым суммарным барионным зарядом. Рожденные адроны принадлежат либо к скалярному и векторному нонетам мезонов, либо к барионному и антибарионному октетам и декуплетам.

1.6.1 Возбуждение струн через аннигиляцию партонов

В теории Редже процессы с аннигиляцией партонов соответствуют разрезанным диаграммам обмена реджионами. Поведения сечений этих процессов с изменением начальной энергии столкновения $\sqrt{s}$ определяются интерсептами реджионных траекторий. Например, $\sigma_{\pi^+ p \to S(s)} \sim s^{\alpha_\rho(0)-1}$, где S обозначает струну и $\alpha_\rho(0)$ - это интерсепт ρ–траектории. Таким образом, сечение $\sigma_{\pi^+ p \to S(s)}$ уменьшается с ростом энергии столкновения. Анализируя аналогичные диаграммы, можно показать, что процессы аннигиляции партонов важны только при относительно низких энергиях столкновения.

Моделировать процессы аннигиляции партонов с возбуждением струн довольно просто. Для каждого столкновения адронов случайным образом, в соответствии с вероятностями, которые рассчитываются с использованием адронных волновых функций (см. описание компонентов структуры частиц), выбираем кварк (антикварк) из налетающего адрона и находим для него подходящего по аромату партнера по аннигиляции из адрона-мишени. 4-импульс, возбужденной в результате аннигиляции струны, аппроксимируется полным 4-импульсом сталкивающейся системы, т.е.

$$P = P_1 + P_2. \qquad (1.68)$$

1.6.2 Статистический вес кварковой аннигиляции

Чтобы определить статистический вес процесса, идущего через аннигиляцию кварков с последующим возбуждением одной струны, и отделить его от процессов, в которых могут возбуждаться две или больше струн, можно использовать выражение, которое следует из теории Редже для полного сечения взаимодействия адронов:

$$\sigma_{tot}(s) = \sum_i \sigma_i(s), \qquad (1.69)$$

где $\sigma_i(s)$ - это сечение i-го подпроцесса. В этой формуле индекс $\sigma_i(s)$ помечает сечение упругого рассеяния $\sigma_{el}(s)$, сечение аннигиляции с рождением одной струны $\sigma_S(s)$, сечение одновершинной дифракционной диссоциации $\sigma_D(s)$ и сечение подпроцесса с возбуждением нескольких струн $\sigma_{MS}(s)$.

Зная из эксперимента полное и упругое сечения, мы можем рассчитать неупругое сечение:

$$\sigma_{in}(s) = \sigma_{tot}(s) - \sigma_{el}(s) = \sigma_S(s) + \sigma_D(s) + \sigma_{MS}(s). \qquad (1.70)$$

Сечения дифракционной диссоциации и сечение возбуждения нескольких струн рассчитываются с использованием померонной эйкнальной модели (см. описания компонентов для расчета сечений взаимодействия частиц). Таким образом, статистический вес для процесса кварковой аннигиляции может быть рассчитан следующим образом:

$$W_S(s) = \frac{\sigma_{in}(s) - \sigma_D(s) - \sigma_{MS}(s)}{\sigma^{in}(s)}. \qquad (1.71)$$

1.6.3 Статистический вес барионной аннигиляции

Сечение барионной аннигиляции $\sigma_{ann}(s)$ используется, чтобы определить статистические веса для процесса аннигиляции дикварка:

$$W_{ann}(s) = \frac{\sigma_{ann}(s)}{\sigma_{tot}(s)}, \qquad (1.72)$$

где $\sigma_{tot}(s)$ - это полное сечение барион-антибарионного взаимодействия как функция квадрата полной энергии сталкивающихся частиц в их системе центра масс.

Данную модель аннигиляции барионов можно улучшить путем добавления другого конкурирующего подпроцесса, связанного с аннигиляцией «струнных узлов» и с последующим возбуждением трех мезонных струн (см. работы [38], [39]).

1.7 Компоненты неупругих столкновений частиц

Здесь описывается составной моделирующий компонент, который мы называем реджеон-партонной струнной моделью [40], [41], [42] . Эта модель может быть использована для расчетов неупругих столкновений различных адронов. Рожденные в результате таких столкновений адроны принадлежат мезонному скалярному и векторному нонетам и барионным (антибарионным) октету и декуплету.

Эту модель можно использовать для описания неупругих столкновений фотонов с нуклонами. В случае адрон-нуклонных и фотон-нуклонных столкновений начальная энергия должна превышать порог образования двух пионов.

Следует отметить, что хотя данный компонент предназначен в основном для описания неупругих столкновений, он может быть также использования для моделирования упругих столкновений частиц и аннигиляции мезонов и антибарионов (см. описание соответствующих компонентов).

1.7.1 Численный алгоритм столкновения частиц

Численный алгоритм моделирования неупругих адрон-нуклонных столкновений включает несколько шагов, которые следует выполнить:

1. Задать типы налетающей частицы и частицы мишени, их поперечные координаты и 4-импульсы.

2. Разыграть тип взаимодействия: упругое, аннигиляционное или неупругое.

3. В случае упругого или аннигиляционного взаимодействий они моделируются, как это было описано в соответствующих разделах.

4. В случае неупругого взаимодействия разыграть прицельный параметр столкновения и определить тип неупругого взаимодействия (дифракционное или недифракционное). Сохранить суммарный 4-импульс участников взаимодействия.

5. В случае недифракционного неупругого взаимодействия определить число продольных струн, а также число кинковых струн, которые будут возбуждаться.

6. Выполнить возбуждение продольных и кинковых струн.

7. Выполнить моделирование распада струн (см. описание компонентов распада струн).

8. Откорректировать значения энергий и импульсов рожденных частиц, если это требуется (см. описание сервисных компонентов).

1.7.2 Определение числа продольных струн

Для каждой пары адронов i и j, которые имеют прицельные параметры $\vec{b}_i$ и $\vec{b}_j$, а также квадрат их полной энергии в их системе центра масс s, следует проверить, взаимодействуют ли они неупруго, используя вероятность:

$$p_{ij}^{in}(\vec{b}_i - \vec{b}_j, s) = p_{ij}(\vec{b}_i - \vec{b}_j, s) + p_{ij}^{d}(\vec{b}_i - \vec{b}_j, s). \qquad (1.73)$$

Если они взаимодействуют неупруго, то следует определить тип этого взаимодействия, т.е. это дифракционное или недифракционное взаимодействие. Вероятность дифракционного взаимодействия задается выражением

$$\frac{p_{ij}^{d}(\vec{b}_i - \vec{b}_j, s)}{p_{ij}^{in}(\vec{b}_i - \vec{b}_j, s)}, \qquad (1.74)$$

и выражение

$$\frac{p_{ij}(\vec{b}_i - \vec{b}_j, s)}{p_{ij}^{in}(\vec{b}_i - \vec{b}_j, s)} \qquad (1.75)$$

определяет вероятность недифракционного взаимодействия.

В подходе Редже–Грибова [4] вероятность найти n разрезанных померонов [5] или вероятность возбуждения $2n$ струн в неупругом взаимодействии адронов вычисляется в соответствии с выражением

$$p_{ij}^{(n)}(\vec{b}_i - \vec{b}_j, s) = c^{-1} \exp\{-2u(b_{ij}^2, s)\} \frac{[2u(b_{ij}^2, s)]^n}{n!}, \qquad (1.76)$$

если предположить, что каждый разрезанный померон представляет собой две струны. Суммируя значения $p_{ij}^{(n)}(\vec{b}_i - \vec{b}_j, s)$, получаем вероятность недифракционного взаимодействия адронов:

$$p_{ij}(\vec{b}_i - \vec{b}_j, s) = \sum_{n=1}^{\infty} p_{ij}^{(n)}(\vec{b}_i - \vec{b}_j, s). \qquad (1.77)$$

Эта вероятность и другие вероятности адронных взаимодействий выражаются в терминах эйкональной амплитуды упругого рассеяния адронов с померонным обменом. Так называемый коэффициент ливневого усиления c вводится для того, чтобы определить вклад дифракционной диссоциации [36] (см. описание компонентов адронных сечений).

1.7.3 Определение числа кинковых струн

Чтобы определить число струн, возбужденных в результате жёстких адронных столкновений, мы предполагаем [44], [45], что каждый разрезанный померон может быть замещён либо двумя продольными струнами, как результат мягкого адронного взаимодействия, либо двумя кинковыми струнами, как результат жёсткого адронного взаимодействия.

В данный момент не существует однозначного, теоретически строго обоснованного выбора, между продольными и кинковыми струнами при замещении разрезанного померона.

Один из способов сделать такой выбор, основан на использовании эйкональной модели [45], [46], в которой эйконал состоит из суммы мягкого и жесткого эйконалов:

$$u(b_{ij}^2,s)=u_{soft}(b_{ij}^2,s)+u_{hard}(b_{ij}^2,s)\,. \qquad (1.78)$$

Мягкая эйкональная часть определена выражением:

$$u_{soft}(b_{ij}^2,s)=\frac{\gamma_{soft}}{\lambda_{soft}(s)}(s/s_0)^{\Delta_{soft}}\exp[-b_{ij}^2/4\lambda_{soft}(s)], \qquad (1.79)$$

а жесткая – выражением

$$u_{hard}(b_{ij}^2,s)=\frac{\sigma_{jet}}{8\pi\lambda_{hard}(s)}(s/s_0)^{\Delta_{hard}}\exp[-b_{ij}^2/4\lambda_{hard}(s)]\,. \qquad (1.80)$$

Значения $\sigma_{jet}=0.027$ *мб* и Δ_{hard}=0.47 были найдены [46] путем фитирования экспериментального сечения рождения двух жестких струй [47] в нуклонных взаимодействиях. Затем, из фитирования экспериментального полного и упругого сечений для *pp*–взаимодействий, были найдены значения $\gamma_{soft}=35{,}5$ *мб*, $\Delta_{soft}=0{,}07$ и $R_{hard}^2=R_{soft}^2=3{,}56$ ГэВ$^{-2}$.

Таким образом, замещение разрезанного померона двумя кинковыми струнами происходит с вероятностью

$$P_{hard}(b_{ij}^2,s)=\frac{u_{hard}(b_{ij}^2,s)}{u_{soft}(b_{ij}^2,s)+u_{hard}(b_{ij}^2,s)}. \qquad (1.81)$$

1.7.4 Возбуждение продольных струн

Рассмотрим неупругое взаимодействие двух адронов, которые имеют следующие значения 4-импульсов в системе их центра масс: $P_1=\{E_1^+,m_1^2/E_1^+,0\}$ и $P_2=\{E_2^-,m_2^2/E_2^-,0\}$. Переменные светового фронта: $E^{\pm}_{1,2}=E_{1,2}\pm P_{z1,2}$ определяются через энергии адронов $E_{1,2}=\sqrt{m_{1,2}^2+P_{z1,2}^2}$, их продольные импульсы– $P_{z1,2}$ и адронные массы– $m_{1,2}$, соответственно. При этом считается, что взаимодействие

адронов происходит посредством столкновения двух партонов с импульсами $p_1 = \{x^+ E_1^+, 0, 0\}$ и $p_2 = \{0, x^- E_2^-, 0\}$, соответственно.

1.7.4.1 Дифракционное возбуждение струн

При дифракционном возбуждении струны (такая процедура используется в модели FRITIOF [48]) может быть передан только 4-импульс:

$$P_1' = P_1 + q, \qquad P_2' = P_2 - q, \tag{1.82}$$

где

$$q = \{-q_t^2/(x^- E_2^-), q_t^2/(x^+ E_1^+), q_t\} \tag{1.83}$$

переданный партонный 4-импульс и $\mathbf{q_t}$ - его поперечная составляющая.

1.7.4.2 Возбуждение струн путем обмена партонами

В этом случае (такая процедура используется в модели QGSM [40]), разрешен как партонный обмен, т.е. замещение партонов, так и передача 4-импульса [40], [41], [42]:

$$P_1' = P_1 - p_1 + p_2 + q \qquad P_2' = P_2 + p_1 - p_2 - q, \tag{1.84}$$

где q={0,0,$\mathbf{q_t}$} - переданный партонный 4-импульс.

1.7.4.3 Генерация поперечных и продольных импульсов партонов

Поперечная составляющая переданного партонного импульса разыгрывается в соответствии с вероятностью

$$P(\mathbf{q_t}) d\mathbf{q_t} = \sqrt{\frac{a}{\pi}} \exp(-a q_t^2) d\,\mathbf{q_t}, \tag{1.85}$$

где параметр $a = 0{,}6$ ГэВ$^{-2}$.

Значения долей переменных светового фронта для партонов x^+ *и* x^- генерируются независимо и в соответствии с распределением

$$u(x) \sim x^{\alpha} (1-x)^{\beta}, \tag{1.86}$$

где $x = x^+ > x_{min}$ или $x = x^- > x_{min}$.

Значения параметров α = -1 и β = 0 выбраны в случае дифракционного возбуждения струн [48]. Выбор значений параметров α и β для случая использования возбуждения струн путем обмена партонами [40] объясняется ниже. Массы возбуждённых струн должны удовлетворять кинематическим условиям:

$$P_1'^{+} P_1'^{-} \geq m_{h1}^2 + q_t^2 \tag{1.87}$$

и

$$P_2'^{+} P_2'^{-} \geq m_{h2}^2 + q_t^2, \tag{1.88}$$

где адронные массы m_{h1} и m_{h2} определяются типами кварков на концах струн. Таким образом, случайный отбор значений x^+ *и* x^- ограничен описанными выше условиями.

1.7.4.4 Обмен партонами и структурные функции адронов

При описании адронных и ядерных взаимодействий в рамках модели QGSM [42] струны (как результат партонных обменов) должны натягиваться не только между валентными кварками сталкивающихся адронов, но также и между валентными и морскими кварками и между морскими кварками. Таким образом, при моделировании возбуждения струн каждый адрон, который участвует в столкновениях, должен быть расщеплён на несколько партонов: на валентный кварк и антикварк в случае мезона или валентный кварк (антикварк) и дикварк (антидикварк) в случае бариона (антибариона), а также на (n-1) морских кварк-антикварковых пар.

Выбор валентных кварков (антикварков) и дикварков (антидикварков) при расщеплении частиц описан в разделе "Компоненты структуры частиц". Ароматы морских кварк-антикварковых пар выбираются в соответствии с отношениями их вероятностей *u:d:s* = (1:1:0.35). Как допустимый вариант имеется возможность учета расщепления адронов, участвующих во взаимодействиях на морские дикварк-антидикварк пары. Этот учет приводит к усилению рождения барионов и антибарионов в центральной области быстрот.

Таким образом, для каждого адрона–участника взаимодействий следует сгенерировать набор значений переменных светового фронта x_{2n}, где $x_{2n} = x_{2n}^{+}$ или $x_{2n} = x_{2n}^{-}$ в соответствии с распределением

$$f^h(x_1, x_2, ..., x_{2n}) = f_0 \prod_{i=1}^{2n} u_{q_i}^h(x_i)\delta(1 - \sum_{i=1}^{2n} x_i), \qquad (1.89)$$

где f_0 – это нормировочная постоянная. В этом выражении $u_{q_i}^h(x_i)$ обозначают адронные структурные функции для валентного кварка (антикварка) q_v, для морского кварка и антикварка q_s и валентного дикваврка (антидикварка) qq:

$$u_{q_v}^h(x_v) = x_v^{\alpha_v}, u_{q_s}^h(x_s) = x_s^{\alpha_s}, u_{qq}^h(x_{vqq}) = x_{qq}^{\beta qq}, \qquad (1.90)$$

где $\alpha_v = -0{,}5$ и $\alpha_s = -0{,}5$ для нестранных кварков (антикварков) и $\alpha_v = 0$ и $\alpha_s = -0$ для странных кварков (антикварков), $\beta_{uu} = 1.5$ и $\beta_{ud} = 2.5$ в случае протона (антипротона) и $\beta_{dd} - 1{,}5$ и $\beta_{ud} = 2{,}5$ в случае нейтрона (антинейтрона).

Значение x_i выбирается в интервале $x_i^{\min} \le x_i \le 1$, где параметр модели $x^{\min} = 0{,}3/\sqrt{s}$ является функцией начальной энергии $\sqrt{s}$. Введенные ограничения на величины x препятствуют рождению струн с малыми массами (меньшими адронных масс), когда процедура возбуждения всех струн должна быть повторена.

Поперечные импульсы партонов $\mathbf{q_{it}}$ генерируются в соответствии с выражением (1.85) с параметром $a = 1/4\lambda(s)$ при условии $\sum_{i=1}^{2n} \mathbf{q_{it}} = 0$, где $\lambda(s)$ вычисляется по формуле (1.61). Таким образом, взаимодействующие адроны расщепляются на виртуальные партоны.

1.7.5 Возбуждение кинковых струн

Возбуждение кинковых струн выполняется в два шага. Сначала для каждой кинковой струны определяется её полный 4-импульс, а затем генерируется импульс глюона-кинка. Первый шаг выполняется таким же образом, как и возбуждение продольных струн. Предполагается, что кинки-глюоны образуются в результате жестких $gg \to gg$ взаимодействий. Генерация импульсов вылетающих глюонов

основана на использовании инклюзивного сечения рождения двух жестких адронных струй:

$$\frac{d\sigma_{gg}}{dx_g^+ dx_g^- d\cos\theta} \cong f(x_g^+, Q^2) f(x_g^-, Q^2) \frac{d\sigma_{gg}(\hat{s})}{d\cos\theta}, \qquad (1.91)$$

где мы принимаем, что $\hat{s} = Q^2 = x_g^+ x_g^- s$ и s – это квадрат полной энергии сталкивающихся адронов, который рассчитывается с использованием значений x_i и q_{it} и m_i^2 для партонов, находящихся на концах струны. Значение s должно быть достаточно большим, чтобы образовать глюоны с поперечным импульсом выше выбранного порога $Q_0^2 = 2$ ГэВ^2. $f(x,Q^2)$ – это функция распределения глюонов в адроне [50]. Сечение глюон-глюонного взаимодействия $\frac{d\sigma_{gg}(\hat{s})}{d\cos\theta}$, где θ - это угол рассеяния глюонов в их системе центра масс, рассчитывалось в Борновском приближении [49].

Таким образом, численная (Монте-Карло) процедура возбуждения кинковых струн может быть описана следующим образом:

- Разыграть x_i , q_{it} и m_i^2, где i = 1,2,…,2n, для партонов, которые появятся на концах 2n струн, Такой розыгрыш выполняется одинаковым образом, как в случаях возбуждения продольных струн, так и в случаях возбуждения кинковых струн.
- Для каждой пары кинковых струн рассчитать квадрат их энергии s и разыграть $x^+ > x_{\min}^+$ и $x^- > x_{\min}^-$, где $x_{\min} = 2Q/\sqrt{s}$, используя глюонные функции распределения.
- Разыграть угол рассеяния вылетающих глюонов θ и определить 4-импульсы рассеянных глюонов.
- Для каждой кинковой струны пересчитать значения энергий и импульсов партонов на концах струны.

Эту процедуру можно усовершенствовать, приняв во внимание излучение глюонов.

1.8Компоненты моделирования ядер

1.8.1 Свойства ядер

Для компонентов моделирования ядер в модуль адронных моделей включена таблица экспериментальных значений [51] масс ядер и их энергий связи. Существует также возможность рассчитать массы ядер и их энергии связи в рамках модели жидкой капли.

1.8.1.1 Формула Бете – Вайцзэккера

В модели жидкой капли [52] значение энергии связи ядра с числом нуклонов A и числом протонов Z может быть вычислено, используя выражение:

$$B(A,Z) = -0{,}01587A + 0{,}01834A^{2/3} + 0{,}09286\left(Z - \frac{A}{2}\right)^2 + 0{,}00071\frac{Z^2}{A^{1/3}}, \quad (1.92)$$

где соответственно суммируются объёмная, поверхностная и кулоновская энергии.

1.8.1.2 Нуклонный потенциал

Для i-го нуклона с радиусом r_i этот потенциал состоит из суммы Фермиевского потенциала

$$T^F(r_i) = \frac{[p^F(r_i)]^2}{2m_i}, \qquad (1.93)$$

энергии связи $B(A,Z)$ и кулоновского потенциала $V_C(Z,r_i)$ (только для протонов):

$$V(A,Z,r_i) = T^F(r_i) + B(A,Z) + V_C(Z,r_i). \qquad (1.94)$$

Фермиевский импульс в локальном приближении Томаса – Ферми [53] следующим образом зависит от плотности протонов или нейтронов:

$$p^F(r) = \hbar(3\pi^2\rho_{Z,N})^{1/3}, \qquad (1.95)$$

где $\hbar = 197{,}327$ МэВ Фм.

Кулоновский барьер (в МэВ) определяется как

$$V_C(A,Z) = C\frac{Z}{r_0(1+A^{1/3})}, \qquad (1.96)$$

где $C = 1{,}44$ МэВ Фм и $r_0 = 1{,}3$ Фм.

1.8.2 Моделирование начальных состояний ядер

В процессе моделирования ядерного столкновения необходимо выполнить инициализацию сталкивающегося ядра или ядер, т.е. для ядра, состоящего из A нуклонов и Z протонов ($N = A - Z$ – число нейтронов), необходимо определить начальные значения нуклонных координат r_i и нуклонных импульсов p_i , где i = 1,2,...,A (см., например, описание компонентов столкновения ядер).

Численный (Монте-Карло) алгоритм может быть сформулирован следующим образом:

1. В системе покоя ядра радиусы нуклонов r_i выбираются случайным образом в соответствии с протонной или нейтронной плотностью $\rho(r_i)$. Для тяжёлых ядер с A>16 [54] нормированная на единицу плотность нуклонов выражается как

$$\rho(r_i) = \frac{\rho_0}{1+\exp[(r_i - R)/a]}, \qquad (1.97)$$

где

$$\rho_0 \approx \frac{3}{4\pi R^3}\left(1+\frac{\alpha^2\pi^2}{R^2}\right)^{-1}. \qquad (1.98)$$

Радиус ядра $R = r_0 A^{1/3}$ Фм, где $r_0 = 1{,}16(1-1{,}16A^{-2/3})$ Фм, и его параметр диффузности $a \approx 0{,}545$ Фм. Для лёгких ядер с A < 17 нормированная на единицу плотность нуклонов берется из оболочечной модели гармонического осциллятора [55] в виде

$$\rho(r_i) = (\pi R^2)^{-3/2}\exp(-r_i^2/R^2), \qquad (1.99)$$

где квадрат радиуса ядра $R^2 = 2/3 < r^2 >= 0{,}8133A^{2/3}$ Фм^2 или в виде:

$$\rho(r_i) = \frac{4}{\pi^{3/2}R^3}\left[1+\frac{A-4}{6}\left(\frac{r_i}{R}\right)^2\right]\exp(-r_i^2/R^2), \qquad (1.100)$$

где

$$R^2 = \left(\frac{5}{2} - \frac{4}{A}\right)^{-1} (< r_{ch}^2 > A - < r_{ch}^2 >_p). \qquad (1.101)$$

Среднеквадратичные значения зарядовых радиусов ядра $< r_{ch}^2 >_A$ и протона $< r_{ch}^2 >_p$ берутся из результатов экспериментов по лептон-ядерному рассеянию [56]. Чтобы учесть отталкивание нуклонов, сделано предположение, что расстояние между двумя любыми нуклонами d > 0,8 Фм.

2. Радиус ядра с учетом диффузности ядра определяется из условия

$$\frac{\rho(R)}{\rho(0)} = 0{,}01. \qquad (1.102)$$

3. Значения импульсов нуклонов выбираются случайно с равными вероятностями в интервале между 0 и $p_{\max}^F(r)$, где $p_{\max}^F(r)$ рассчитывается из уравнения (1.95). При определении координат и фермиевских импульсов нуклонов сделано предположение об изотропии их распределений в конфигурационном и импульсном пространствах.

4. В системе покоя ядра оно должно быть отцентровано, т.е. $\sum_i r_i = 0$ и должно находится в покое, т.е. $\sum_i p_i = 0$ и $\sum_i r_i$ x $p_i = 0$. Чтобы обеспечить первые два условия, осуществляется «сдвиг» координат нуклона $r_i' = r_i - 1/A \sum_i r_i$ и их импульсов $p_i' = p_i - 1/A \sum_i p_i$.

5. Имеется возможность «распаковать» ядро в отдельные «независимые» нуклоны путем замены табличных нуклонных масс на их эффективные массы, учитывающие энергии связи нуклонов в ядре. Это делается с помощью предположения, что все нуклоны, принадлежащие ядру, имеют одинаковые энергии $e = E/A = m_N + B(A,Z)/A$, где m_N - это табличная масса нуклона, и $B(A,Z)$ - энергия связи нуклона, которая определяется формулой Бете – Вайцзэккера (1.92). Таким образом, эффективная масса для каждого нуклона определяется как $m_i^{eff} = \sqrt{(E/A)^2 - p_i^{2'}}$.

1.8.3 Примеры моделирования начального состояния ядра

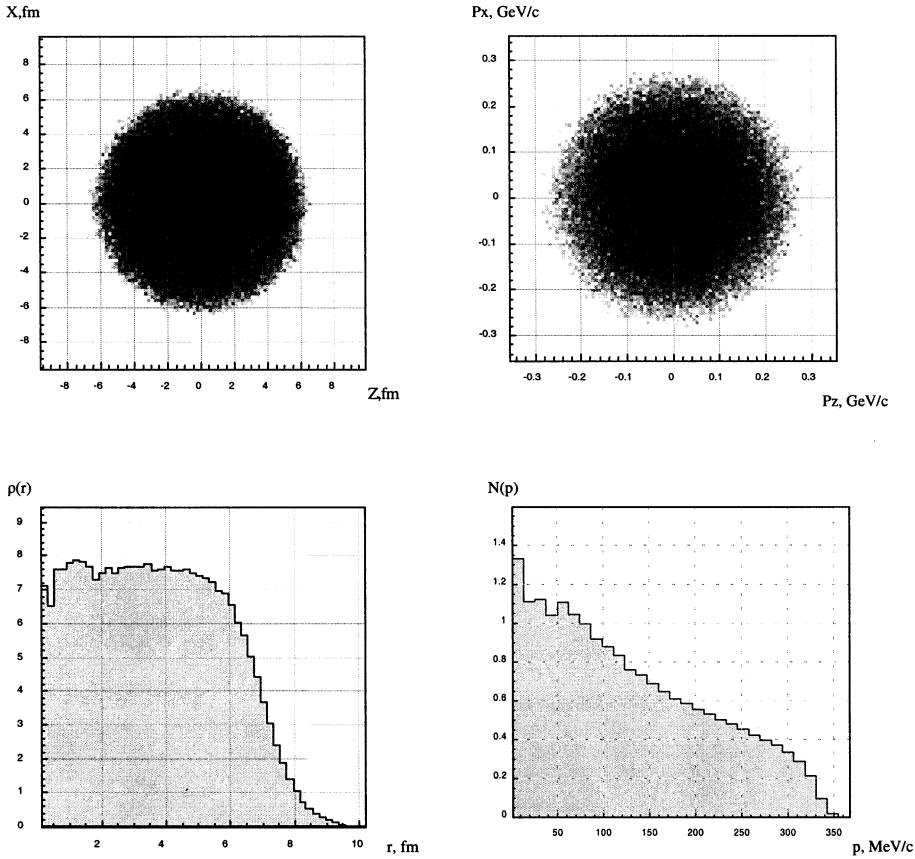

Рис 1.1. Результат работы компонента, моделирующего начальное состояние ядра. Слева показаны координаты нуклонов, справа – импульсы нуклонов. Распределения получены для ^{238}U

1.9 *Компоненты неупругих столкновений ядер*

Здесь описывается составной моделирующий компонент, который называется померон-партонной струнной моделью. Эта модель может быть использована для расчетов неупругих столкновений различных адронов и различных ядер с ядерными мишенями. Рожденные в результате столкновения адроны принадлежат мезонному скалярному и векторному нонетам и барионным (антибарионным) октету и

декуплету. В случае адрон-нуклонных столкновений начальная энергия должна превышать порог образования двух пионов. В случае адрон-ядерных и ядро-ядерных столкновений рекомендуемая начальная энергия $\sqrt{s} > 5$ А ГэВ. Эту модель можно использовать для описаний неупругих столкновений фотонов с нуклонами и атомными ядрами при значениях энергии $\sqrt{s} > 5$ А ГэВ .

1.9.1 Численный алгоритм столкновения ядер

Численный алгоритм моделирования неупругих ядерных столкновений включает несколько шагов, которые следует выполнить:

1. Создать два вектора частиц (вектор налетающих частиц и вектор частиц мишени). Задать значения этих векторов, т.е. типы частиц, значения их координат и импульсов. При этом в случае адрон-ядерных или ядро-ядерных столкновений следует выполнить моделирование начального состояния ядра-мишени или начальных состояний ядра-мишени и налетающего ядра (как это было описано в разделе "Компоненты моделирования ядер") и разыграть прицельный параметр столкновения.

2. Определить адроны и нуклоны, которые участвуют в столкновении, и разделить этих участников на две группы: участников дифракционных и участников недифракционных столкновений. Сохранить суммарный 4-импульс частиц – участников столкновения.

3. Для каждого недифракционного неупругого столкновения определить количество продольных и кинковых струн, которые могут рождаться в данном столкновении.

4. Возбуждать и перевозбуждать продольные струны для соударяющейся частицы в случае дифракционных столкновений.

5. Выполнить возбуждение продольных и кинковых струн для частиц, участвующих в недифракционных столкновениях.

6. Выполнить моделирование распада струн (см. описание компонентов распада струн).

7. Откорректировать значения энергий и импульсов рожденных частиц, если это требуется (см. описание сервисных компонентов).

1.9.2 Моделирование начальной стадии реакции

Прицельный параметр $0 \le b \le R_p + R_t$ выбирается случайным образом с помощью распределения

$$P(b) = b. \qquad (1.103)$$

Затем следует модифицировать поперечные компоненты координат налетающей частицы:

$$r_{xi} \to r_{xi} + b_x \qquad (1.104)$$

и

$$r_{yi} \to r_{yi} + b_y . \qquad (1.105)$$

Если центр ядра-снаряда находится в начале координат, выполнить сдвиг продольных координат нуклонов мишени:

$$r_{zi} \to r_{zi} + \Delta r_z / \gamma_i , \qquad (1.106)$$

где можно принять для $\Delta_r = R_p + 1{,}5$ *Фм* $+ R_t + 1{,}5$ *Фм*.

В качестве «рабочей» системы отсчета, в которой разыгрывается процесс взаимодействия, необходимо выбрать либо систему центра масс сталкивающихся частиц в случае адрон-нуклонных столкновений, либо систему равных скоростей в случае ядерных столкновений.

В случае, когда ядро быстро движется с начальным импульсом (на один нуклон), равным $P_0 = \{0,0,P_{z0}\}$, следует выполнить преобразование Лоренца для продольных импульсов нуклонов ядра:

$$p_{zi} \to \gamma_i (p_{zi} - \beta_i e_i) \tag{1.107}$$

и значений энергий этих нуклонов

$$e_i \to \gamma_i (e_i - \beta_i p_{zi}), \tag{1.108}$$

где скорость β_i определена выражением

$$\beta_i = \frac{P_{z0}}{\sqrt{P_{z0}^2 + m_i^{eff\,2}}} \tag{1.109}$$

и γ_i фактор вычисляется как

$$\gamma_i = \frac{1}{\sqrt{1-\beta_i^2}}. \tag{1.110}$$

Здесь m_i^{eff} обозначает эффективную массу i-го нуклона.

Для быстро движущихся ядер необходимо также выполнить сжатие Лоренца координат нуклонов, т.е.,

$$r_i' = r_i - \frac{\gamma}{1+\gamma} v(v r_i), \tag{1.111}$$

где **v** обозначает скорость ядра и $\gamma = 1/\sqrt{1-v^2}$.

При моделировании большого числа событий ядерных столкновений, когда участвует то же самое ядро или те же самые два ядра, инициализацию ядра или ядер можно выполнить только один раз перед моделированием первого столкновения. При моделировании повторных столкновений можно просто вращать ядро или ядра случайным образом в координатном и импульсном пространствах. Это существенным образом сокращает время работы процессора, необходимое для выполнения моделирования. Для того чтобы выполнить это вращение, случайным образом (с равными вероятностями), выбираются три угла Эйлера: $0 < \theta_1 < 2\pi$, $0 < \theta_2 < 2\pi$, $0 < \theta_3 < 2\pi$. Затем координаты нуклона $\mathbf{r_i} = \{r_{xi}, r_{yi}, r_{zi}\}$, их импульсы $\mathbf{p_i} = \{p_{xi}, p_{yi}, p_{zi}\}$ последовательно преобразовываются (вращаются) в соответствии с

$$\mathbf{r_i} \to \mathbf{r_i}R_1 \to \mathbf{r_i}R_2 \to \mathbf{r_i}R_3 \tag{1.112}$$

и

$$\mathbf{p_i} \to \mathbf{p_i}R_1 \to \mathbf{p_i}R_2 \to \mathbf{p_i}R_3, \tag{1.113}$$

где матрицы вращения определены как:

$$R_1 = \begin{vmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{vmatrix}, \tag{1.114}$$

и

$$R_2 = \begin{vmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{vmatrix} \qquad (1.115)$$

и

$$R_3 = \begin{vmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{vmatrix}. \qquad (1.116)$$

1.9.3 Определение участников ядерных столкновений

Неупругие адрон-ядерные или ядерно-ядерные взаимодействия при ультрарелятивистских энергиях рассматриваются как независимые адрон-нуклонные или нуклон-нуклонные неупругие столкновения. Такое описание в случае адрон-ядерного неупругого столкновения было получено много лет назад [57] в рамках теории Редже–Грибова [58], когда амплитуда адрон-ядерного упругого рассеяния является результатом реджионных обменов между начальным адроном и нуклонами из ядра-мишени. Этот результат может быть обобщен на случай ядро-ядерного взаимодействия, что дает возможность сформулировать простой и эффективный численный (Монте-Карло) алгоритм [59] для определения сечений взаимодействия и числа нуклонов, участвующих в ядро-ядерном столкновении:

1. Мы должны случайным образом распределить A нуклонов из налетающего ядра и B нуклонов из ядра-мишени в плоскости прицельного параметра, используя весовые функции $T([\vec{b}_i^{\,A}])$ и $T([\vec{b}_j^{\,B}])$, соответственно. Эти функции представляют собой распределения вероятностей для нахождения наборов прицельных параметров нуклонов $[\vec{b}_i^{\,A}]$ и $[\vec{b}_j^{\,B}]$, где $i = 1,2,\ldots,A$ и $j = 1,2,\ldots,B$ соответственно.

2. Для каждой пары нуклонов i и j с выбранными прицельными параметрами $\vec{b}_i^{\,A}$ и $\vec{b}_i^{\,B}$ необходимо проверить, взаимодействуют ли они неупруго или нет, с помощью вероятностей $p_{ij}(\vec{b}_i^{\,A} - \vec{b}_j^{\,B}, s)$, где $s_{ij} = (p_i + p_j)^2$ - это квадрат полной энергии в системе центра масс пары нуклонов с 4-импульсами p_i и p_j.

Описываемая Монте-Карло-процедура основывается на вероятности $P(\vec{B}, s)$ найти конфигурацию, когда несколько пар нуклонов из ядра-снаряда и ядра-мишени взаимодействуют неупруго, а остальные нуклоны не участвуют в столкновении:

$$P(\vec{B}, s) = \left\langle \prod_{i,j=1} p_{ij}(\vec{b}_i^{\,A} - \vec{b}_j^{\,B}, s) \prod_{k,l=1} [1 - p_{kl}(\vec{b}_k^{\,A} - \vec{b}_k^{\,B}, s)] \right\rangle, \qquad (1.117)$$

где $\vec{B}$ – это прицельный параметр столкновения ядер и s – это квадрат полной энергии пары нуклонов в их системе центра масс. Чтобы упростить систему обозначений, предполагается, что все взаимодействующие нуклонные пары имеют одно и то же значение s.

Последнее уравнение может быть выписано более детально:

$$P(\vec{B}, s) = \int \prod_{i,j=1} p_{ij}(\vec{b}_i^{\,A} - \vec{b}_j^{\,B}, s) \prod_{k,l=1} [1 - p_{kl}(\vec{b}_k^{\,A} - \vec{b}_l^{\,B}, s)] \times$$
$$\times T_A(\vec{b}_1^{\,A}) T_A(\vec{b}_2^{\,A}) \ldots T_A(\vec{B} - \vec{b}_B^{\,B}) d\vec{b}_1^{\,A} d\vec{b}_2^{\,A} \ldots d\vec{b}_B^{\,B}. \qquad (1.118)$$

Вероятности для неупругого столкновения нуклонов i и j как функции разности квадратов прицельных параметров $b_{ij}^2 = (\vec{b}_i^{\,A} - \vec{b}_j^{\,B})^2$ и s могут быть рассчитаны в подходе Редже–Грибова [60], [58] (см. описание компонентов сечений взаимодействий адронов).

Следует подчеркнуть, что при использовании померонной эйкональной модели [58] в амплитуде адронного упругого рассеяния принимаются в рассмотрение только померонные обмены. При использовании такой амплитуды невозможно получить правильное описание экспериментальных сечений адронных взаимодействий, а также соответствующих вероятностей в полном диапазоне изменения начальных энергий столкновения адронов и нуклонов. При этом в значительной степени недооцениваются значения экспериментальных сечений и соответствующих вероятностей взаимодействия при относительно низких энергиях.

В этом диапазоне энергий, пренебрегая реальной частью амплитуды рассеяния адронов на нуклонах, мы можем "откорректировать" эйкональные переменные (см. описание компонентов сечений взаимодействий адронов), выражая их через экспериментальные значения полных $\sigma_{tot}(s)$, упругих $\sigma_{el}(s)$ и одновершинных дифракционных $\sigma_d(s)$ сечений:

$$\frac{z}{2} = \frac{\sigma_{tot}(s)}{4\pi B_{el}(s)}, \tag{1.119}$$

$$\lambda(s) = 2B_{el}(s), \tag{1.120}$$

где

$$B_{el}(s) = \frac{\sigma_{tot}^2(s)}{8\pi[\sigma_{el}(s) + \sigma_d(s)]}. \tag{1.121}$$

1.9.4 Пример моделирования столкновений ядер

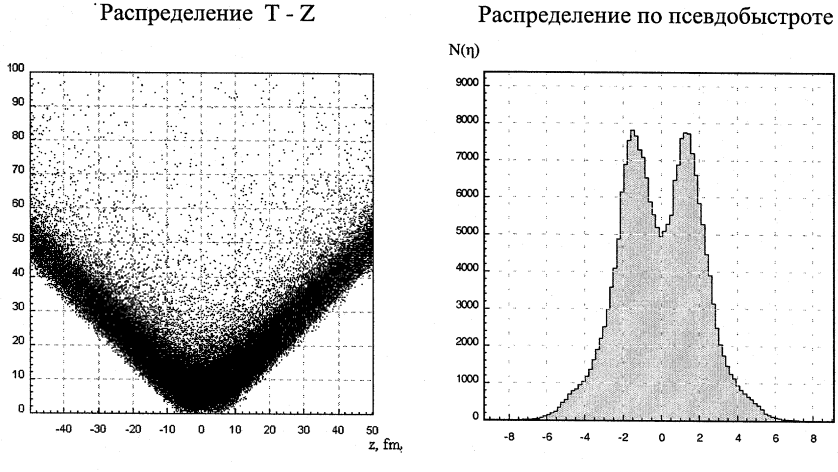

Рис 1.2. Центральное столкновение $^{238}U + ^{238}U$ при $E_{cm} = 2000$ АГэВ

1.10 Сервисные компоненты

Разработан набор компонентов, которые выполняют сервисные функции при моделировании столкновений частиц и ядер. Как правило, эти компоненты имеют дело с преобразованиями 4-импульса частиц [1].

1.10.1 Коррекция энергии и импульса системы частиц

Иногда при моделировании столкновений частиц и ядер требуется коррекция значений энергии и импульса для каждой рожденной в данном столкновении частицы с тем, чтобы обеспечить сохранение полной энергии и полного импульса в реакции. Такая процедура коррекции выполняется за три шага:

- Преобразование 4-импульсов рожденных частиц в их систему центра масс, используя скорость системы центра масс рождённых частиц.
- Масштабирование 3-импульса каждой рожденной частицы до тех пор, пока суммарная масса этих частиц не станет близка к эффективной массе сталкивающейся системы. Масштабирование контролируется заданным значением отклонения коэффициента масштабирования от единицы.
- Выполнение обратного преобразования 4-импульсов рожденных частиц в начальную систему столкновения, используя скорость системы столкновения.

1.10.2 Распределения частиц

В результате столкновения частиц или ядер мы получаем значения 3-импульсов частиц (p_x, p_y, p_z), их энергии (E), их квантовые числа, т.е. типы частиц. Эта информация даёт нам возможность строить различные распределения частиц. Распределение по быстроте $y = -0.5 \ln \frac{E + p_z}{E - p_z}$ для частиц определенного типа строится в соответствии с выражением

$$\frac{d\bar{n}}{dy} = \frac{1}{N_{evt}} \sum_{k=1}^{N_{evt}} \sum_{j=1}^{n^k} \delta(y_j^k - y), \qquad (1.122)$$

где $\bar{n}$ - средняя множественность частиц требуемого типа, N_{evt} – число сгенерированных событий столкновений, n^k – число частиц данного типа в k–м событии. Подобным образом можно построить другие распределения частиц, например, распределение по поперечному импульсу $\bar{p}_t$:

$$\frac{d\bar{n}}{d^2 p_t} = \frac{1}{N_{evt}} \sum_{k=1}^{N_{evt}} \sum_{j=1}^{n^k} \delta^2(p_{tj}^k - p_t). \qquad (1.123)$$

Распределение $P(n)$ по множественности частиц n строится с помощью выражения

$$P(n) = \frac{1}{N_{evt}} \sum_{i=1}^{Nevt} \delta(n^i - n), \qquad (1.124)$$

где n^i – множественность частиц требуемого типа в i – м событии.

Литература

[1] Amelin N., Physics and Algorithms of the Hadronic Monte-Carlo Event Generators. Notes for a developer. CERN/IT/99/6.

[2] Wenaus T. *et al.*, GEANT4: An Object-Oriented Toolkit for Simulation in HEP,CERN/LHCC/97-40.

[3] Taligent's Guide: Building Object-Oriented Frameworks, http://www.ibm.com/java/education/oobuilding/index.html

[4] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns. Elements of Reusable Object-Oriented Software, Eddison-Wesley, 1995.

[5] Amelin N. and Komogorov M., An Object-Oriented Framework for the Hadronic Monte-Carlo Event Generators. JINR Rapid Communications, 1999, No. 5-6 [97]-99, p. 52-84.

[6] Amelin N. and Komogorov M., An Object-Oriented Framework for the Hadronic Monte-Carlo Event Generators. In Proc. of Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2000), 7 - 11 February 2000, Padova, Italy, p. 119-123.

[7] Komogorov M. and Amelin N., Component-oriented framework for hadron Monte-Carlo event generators. The abstract published in Proc. of XXXIV annual conference of the Finish Physical Society, March 9-11, 2000, Espoo, Finland, p. 91.

[8] Amelin N. and Komogorov M., NIMAX: A New Approach to Develop Hadronic Event Generators in HEP. PHYS. P&N LETTERS, 2000, No. 3 [100]-2000, p. 35-47.

[9] Amelin N. and Komogorov M., NIMAX System: A New Approach to Develop, Assemble and Use Numerical Models in HEP. The talk presented at the XV Int. Seminar on High Energy Physics Problems, Dubna, Russia, 25 - 29 September, 2000 (will be published in the Proceeding); JINR Report E1-2001-31.

[10] Kruglinski D. J., Shepherd G., Wingo S. - Programming Microsoft Visual C++, Fifth Edition, Microsoft Press, 1998.

[11] Orfali R., Harkey D., Edwards J., The essential distributed objects. Survival guide., John Wiley and Sons, Inc., 1996.

[12] Amelin N. and Komogorov M., Hadronic Model Components, in preparation.

[13] Class Library for High Energy Physics, http://wwwc1.cern.ch/asd/lhc++/clhep/index.html

[14] Brun R. *et al.,* ROOT System, CERN/HP, 1997,
http://root.cern.ch/.

[15] HBOOK Statistical Analysis and Histograming Reference Manual, CERN Program Library Long Writeups Y250, CERN Geneva, Switzerland, 1998.

[16] Particle-Data-Group, Phys. Rev. **D45** 1342 (1992).

[17] Particle-Data-Group, Phys. Rev. **D54** 1 (1996).

[18] Particle-Data-Group 98, Eur. Phys. J. **C3** 1 (1998).

[19] Particle-Data-Group 2000, Eur. Phys. J. **C15** 1 (2000).

[20] Piller G., Ratzka W., Weise W., Z. Phys. **A352** 427 (1995).

[21] Bauer T. H., Spital R. D., Yennie D. R., Rev. Mod. Phys. **50** 261 (1978).

[22] Levin E. M., Frankfurt L. L., Pisma ZhETP **3** 105 (1965).

[23] Lipkin H. J., Sheck F., Phys. Rev. Lett. **16** 71 (1966).

[24] Bass S. A. *et al.*, Prog. Part. Nucl. Phys. **V. 41** 225 (1998).

[25] Abramovskii V. A., Gribov V. N., Kancheli O. V., Sov. J. Nucl. Phys. **18** (1974) 308.

[26] Baker M. and Ter-Martirosyan K. A., Phys. Rep. **28C** (1976) 1.

[27] Goulianos K., Phys. Rep. **101** 169 (1983).

[28] Koch P., Dover C. B., Phys. Rev. **C40** 145 (1989).

[29] Копылов Г. И., Основы кинематики резонансов, Наука, Москва, 1970.

[30] Gottschalk T. D., Nucl. Phys. **B214** 201 (1983); **B227** 413 (1983).

[31] Weber B. R., Nucl. Phys. **B238** 492 (1984).

[32] Andersson B., Gustafson G., Ingelman G., Sjőstrand T., Phys. Rep. **97** 31 (1983).

[33] Kaidalov A. B., Sov. J. Nucl. Phys. **45** 1452 (1987).

[34] Ali A, Pietarinen E., Kramer G., Willrodt J, Phys. Lett. **B93** 155 (1980).

[35] Altarelli G. and Parisi G., Nucl. Phys. **B126** 298 (1977).

[36] Baker M. and Ter-Martirosyan K. A., Phys. Rep. **28C** 1 (1976).

[37] Cugnon J., Mizutani T., Vandermeulen J., Nucl. Phys. **A352** 505 (1981).

[38] Volkovitskii P. E., Sov. J. Nucl. Phys. **44** 729 (1986).

[39] Amelin N. S., Bravina L. V., Sarycheva L. I., Sov. J. Nucl. Phys. **50** 1705 (1989).

[40] Kaidalov A. B., Ter-Martirosyan K. A., Phys. Lett. **B117** 247 (1982).

[41] Capella A., Sukhatme U., Tan C. I., Tran Thanh Van. J., Phys. Rep. **236** 225 (1994).

[42] Amelin N. S., Bravina L. B., Sov. J. Nucl. Phys. **51** (1990) 211;

[43] Abramovskii V. A., Gribov V. N., Kancheli O. V., Sov. J. Nucl. Phys. **18** 308 (1974).

[44] Amelin N. S., Stöcker H, Greiner W., Armesto N., Braun M. A., Pajares C., Phys. Rev. **C52** 362 (1995).

[45] Werner K., Drescher H. J., Furler E., Hladik M., Ostapchenko S., in Proc.: 3rd Inter. Conf. on Physics and Astrophysiks of Quark-Gluon Plasma, Jaipur, India, March 17-21, 1997.

[46] Ranft J, Capella A., Tran Thanh Van, Phys. Lett. **B320** 346 (1994).

[47] UA1 Collaboration, Albajar C. *et al.*, Preprint CERN-EP/88-29.

[48] Andersson B., Gustafson G., Nielsson-Almquist, Nucl. Phys. **281** 289 (1987).

[49] Combrige B. L. , Kripfganz J. , Ranft J. , Phys. Lett. **70B** 234 (1977); Cutler R., Sivers D., Phys. Rev. **D17** 196 (1978).

[50] Capella A., Kaidalov A., Merino C., Tran Than Van J., Phys. Lett. **B343** 403 (1995).

[51] Audi G., Wapstra A. H., Nucl. Phys. **A595** V. 4 409 (1995).

[52] Bohr A., Mottelson B. R., Nuclear Structure, W. A. Benjamin, New York, Vol. 1, 1969.

[53] DeShalit A., Feshbach H., Theoretical Nuclear Physics, Vol. 1: Nuclear Structure, Wyley, 1974.

[54] Grypeos M. E., Lalazissis G. A., Massen S. E., Panos C. P., J. Phys. **G17** 1093 (1991).

[55] Elton L. R. B., Nuclear Sizes, Oxford University Press, Oxford, 1961.

[56] Barrett R. C. and Jackson D. F., Nuclear Sizes and Structure. Oxford University, New York, 1977.

[57] Capella A. and Krzywicki A., Phys. Rev. D18 (1978) 4120.

[58] Baker M. and Ter-Martirosyan K. A., Phys. Rep. **28C** (1976) 1.

[59] Amelin N. S., Gudima K. K., Toneev V. D., Sov. J. Nucl. Phys. **51** (1990) 327; Amelin N. S., JINR Report **P2-86-56** (1986).

[60] Abramovskii V. A., Gribov V. N., Kancheli O. V., Sov. J. Nucl. Phys. **18** (1974) 308.

Рукопись поступила в издательский отдел
21 августа 2001 года.

Амелин Н. С., Комогоров М. Э.
Д11-2001-175

Компонентно-ориентированный подход
к разработке и использованию численных моделей
в физике высоких энергий

Обсуждаются основные концепции компонентного подхода к разработке и применению численных моделей в физике высоких энергий. Данный подход реализован в виде системы программ NiMax. Обсуждаемые концепции иллюстрируются многочисленными примерами работы пользователя системы. В приложении описаны физика и численные алгоритмы компонентов для моделирования событий взаимодействия частиц и ядер при высоких энергиях. Эти компоненты входят в состав прикладных модулей адронных моделей, созданных в рамках данной системы. Данное описание может рассматриваться как предварительная версия руководства для пользователей моделирующих компонентов, работающих с системой NiMax.

Работа выполнена в Лаборатории высоких энергий ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2001

Перевод авторов

Amelin N. S., Komogorov M. E.
D11-2001-175

Component-Oriented Approach to the Development
and Use of Numerical Models in High Energy Physics

We discuss the main concepts of a component approach to the development and use of numerical models in high energy physics. This approach is realized as the NiMax software system. The discussed concepts are illustrated by numerous examples of the system user session. In appendix chapter we describe physics and numerical algorithms of the model components to perform simulation of hadronic and nuclear collisions at high energies. These components are members of hadronic application modules that have been developed with the help of the NiMax system. Given report is served as an early release of the NiMax manual in the main for model component users.

The investigation has been performed at the Laboratory of High Energies, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2001

Редактор М. И. Зарубина. Макет Н. А. Киселевой

Подписано в печать 20.12.2001
Формат 60 × 90/16. Офсетная печать. Уч.-изд. л. 8,8
Тираж 245. Заказ 53024. Цена 8 р. 80 к.

Издательский отдел Объединенного института ядерных исследований
Дубна Московской области