ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Коробкин Евгений Александрович

Бионические нечеткие модели и алгоритмы для исследования систем многоточечных масс при формировании устойчивой сыпучей насыпи

Специальность 05.13.17
Теоретические основы информатики
Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель -
доктор технических наук,
профессор Астахова И.Ф.

ВОРОНЕЖ – 2016

Оглавление

Введение

Актуальность темы. Использование компьютерных технологий привело к пониманию важности задач, связанных с обработкой накопленной информации для извлечения знаний. В современных вычислительных алгоритмах анализ данных зачастую характеризуется наличием слабо структурированной, неполной, неточной и нечеткой информацией, возникающей вследствие неопределенности, присущей моделям сложных процессов, а также необходимостью решения задач в тех областях, где существенная роль принадлежит суждениям и знаниям экспертов. Как только в работах Заде для формализации анализа неопределенности стала использоваться теория нечетких множеств, интерес к анализу приближенной и иногда нечеткой информации существенно вырос.

Большинство научных проблем в различной степени связано с задачей оптимизации. При этом пространство параметров в задачах оптимизации не всегда является известным и предсказуемым. Современные исследования направлены на поиск эффективных модификаций существующих методов, которые позволяют справиться с данной проблемой.

В новых бурно развивающихся научных направлениях нейробионического и эволюционного моделирования сочетаются методы нечеткой логики и генетических алгоритмов, что открывает другие возможности перед исследователем для решения трудно-формализуемых задач в условиях неопределенности. Появились нейро-нечеткие системы: нечетко-генетические, нейрогенетические и нейро-нечетко-генетические. Данная работа посвящена построению нечетко-генетической системы и ее практическому использованию для оценки устойчивости сыпучей среды.

Вопрос изучения сыпучих сред на сегодняшний день является актуальным, поскольку сыпучие материалы повсеместно встречаются в природе и в промышленности в отраслях, связанных с сыпучими средами (производство сахара, работа с песком и т.д.). С появлением распределенных вычислений, графических процессоров, как ускорителей, в настоящее время

интерес к сыпучим средам возрос, но все равно существует много сложностей в стыковке области механики и компьютерного моделирования в разработке моделей, описывающих их движение. В данной работе рассмотрен вопрос об использовании разработанных алгоритмов к сыпучим средам, которые позволят на несколько порядков сократить время расчета геометрии сыпучей среды и оценить ее предельное состояние.

Диссертационная работа выполнена в рамках одного из основных научных направлений Воронежского государственного университета «Математическое моделирование, программное и информационное обеспечение, методы вычислительной и прикладной математики и их применение к фундаментальным исследованиям в естественных науках».

Цель работы и основные задачи. Целью диссертационной работы является разработка бионических нечетких моделей и алгоритмов для исследования системы многоточечных масс при формировании устойчивой сыпучей насыпи. Для достижения этой цели необходимо решить следующие задачи:

1. Разработать алгоритм для исследования системы многоточечных масс в трехмерном пространстве при формирования сыпучей насыпи.
2. Создать модель и алгоритм прогнозирования поведения насыпи на основе нечеткой логики.
3. Разработать генетический алгоритм для настройки коэффициентов прогнозной модели.
4. Создать специальное программное обеспечение, реализующее предложенные алгоритмы.

Методы исследования. При решении поставленных задач использовались методы нечеткой логики и генетических алгоритмов, методы работы с графическим процессором, методы обеспечения обработки информации на базе специализированной вычислительной технологии CUDA, методы математического моделирования и метод дискретных элементов, методы объектно-ориентированного программирования.

Научная новизна. Научная новизна настоящей работы заключается в следующем:

1. В разработке алгоритма, описывающего поведение системы многоточечных масс в трехмерном пространстве при формировании геометрии сыпучей насыпи с использованием графического процессора, который позволяет повысить производительность данного алгоритма на несколько порядков в сравнении с вычислениями на центральном процессоре.

2. В создании модели и алгоритма прогнозирования устойчивости насыпи с помощью нечеткой логики и ситуационной сети.

3. В разработке бионической модели для настройки коэффициентов модели прогнозирования, использующей модифицированный генетический алгоритм, позволяющий ускорить поиск оптимальных параметров для нечеткой модели.

4. В создании программной системы моделирования для проведения экспериментальных исследований поведения многоточечных масс и получения геометрии сыпучей среды для исследования на устойчивость.

Личный вклад автора. Основные результаты исследований по теме диссертации были получены лично автором и опубликованы в соавторстве с научным руководителем. Научным руководителем определены основные направления исследования.

Теоретическая и практическая ценность. Работа имеет теоретический и практический характер. В работе создается алгоритм для предсказания поведения многоточечных масс (порядка нескольких десятков тысяч) для формирования сыпучей насыпи, разрабатывается модель и алгоритм прогнозирования, отличающихся использованием нечеткой логики и генетического алгоритма для настройки ее коэффициентов.

Практическая ценность работы состоит в возможности использования разработанных алгоритмов в программном обеспечении для принятия решений о поведении насыпи. Результаты работы используются и

тестируются на предприятии ООО «ОГНЕБОРЕЦ+СВ» в г. Воронеже. По результатам работы получены свидетельства о государственной регистрации программы для ЭВМ № 2015660382 «Разработка нечеткой модели прогнозирования устойчивости грунтового массива» от 30 сентября 2015 г. и № 2015661068 «Технология CUDA для метода дискретных элементов в параллельной среде» от 15 октября 2015 г.

Апробация работы. Результаты, представленные в диссертации, докладывались и обсуждались на VII Международной научно-практической конференции «Современные информационные технологии и IT-образование» (Москва, 2012), на Международной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2012, 2015), на VII Международной научно-практической конференции «Теоретические и прикладные аспекты современной науки» (Белгород, 2015), на XIII Международной научно-практической конференции «Научное обозрение физико-математических и технических наук в XXI веке» (Москва, 2015),на Международной конференции «Воронежская весенняя математическая школа «Понтрягинские чтения – XXVI. Современные методы теории краевых задач» (Воронеж, 2015), на научных сессиях Воронежского государственного университета.

Публикации. Результаты диссертации опубликованы в 11 работах. 1 работа опубликована в журнале, индексируемом в Agris, 2 – в журналах из перечня ВАК. Получены 2 свидетельства о государственной регистрации программы для ЭВМ. Из совместных работ в диссертацию вошли только результаты, принадлежащие лично диссертанту.

Структура и объём диссертации. Диссертация состоит из введения, 4 глав, разбитых на параграфы, заключения, списка используемой литературы из 90 наименований и приложения. Общий объем диссертации – 116 страниц. Работа содержит 29 рисунков, 1 диаграмму и 23 таблицы.

Область исследования. Диссертационная работа соответствует следующим пунктам паспорта специальности 05.13.17 – Теоретические основы информатики:

П.4. Исследование и разработка средств представления знаний. Принципы создания языков представления знаний, в том числе для плохо структурированных предметных областей и слабоструктурированных задач; разработка интегрированных средств представления знаний, средствпредставления знаний, отражающих динамику процессов, концептуальных и семиотических моделей предметных областей.

П.13. Применение бионических принципов, методов и моделей в информационных технологиях.

П. 14. Разработка теоретических основ создания программных систем для новых информационных технологий.

На защиту выносятся:

1. Алгоритм поведения системы многоточечных масс в трехмерном пространстве при формирования сыпучей насыпи с использованием графического процессора.

2. Модель и алгоритм прогнозирования устойчивости насыпи с помощью нечеткой логики и ситуационной сети.

3. Модифицированный генетический алгоритм для настройки коэффициентов модели прогнозирования, позволяющий сделать поиск оптимальных параметров для нечеткой модели более эффективным.

4. Программный комплекс для проведения вычислительного эксперимента по предложенным алгоритмам.

Глава 1. Обзор алгоритмов прогнозирования и бионических моделей

1.1 Обзор и сравнение алгоритмов анализа данных на основе графического процессора

Графический процессор, который первоначально использовался только для ускорения трехмерной графики, стал мощным программируемым устройством, решающим целый класс вычислительных задач. Одной из первых типовых задач рендеринга, с которыми справлялись графические процессоры, является задача растеризации (перевод треугольников в массивы). Все вершины и фрагменты можно обрабатывать независимо друг от друга. На языке программирования это означает, что для каждого элемента исходных данных достаточно одного алгоритма. Это свойство определило принципы архитектуры графических процессоров.

Сравнение архитектуры графического процессора с архитектурой центрального процессора приведено на рис. 1.1.

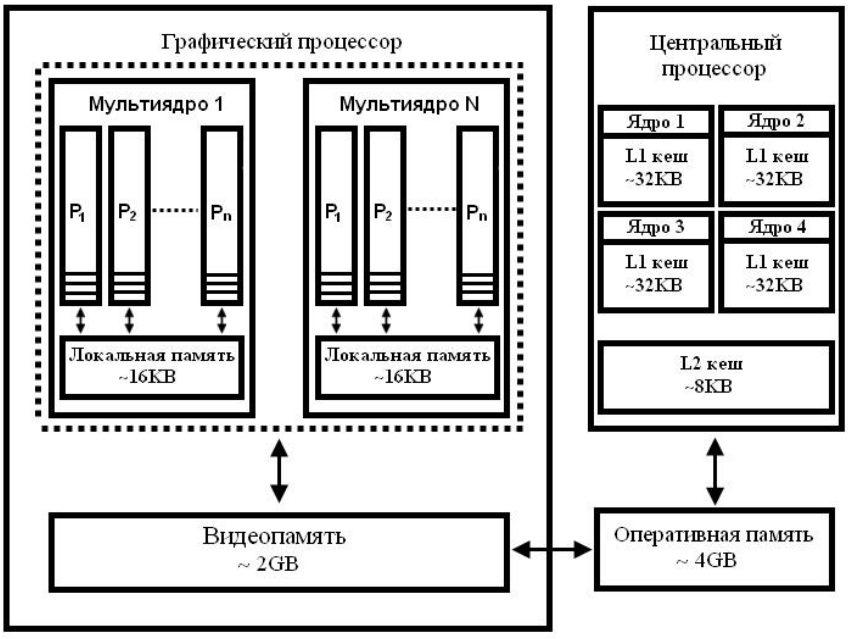

Рис. 1.1. Архитектура графического процессора

Для увеличения производительности графического процессора используется технология вычислений, основанная на том, что одна микропрограмма оперирует многими данными (single-instruction, multiple data (SIMD). Графический процессор состоит из большого числа SIMD ядер, (threads), каждое из которых выполняет одну такую микропрограмму, называемую ядерной функцией (kernel function). На рис. 1.2 SIMD ядра обозначены как P_i.

Согласно иерархии графического процессора данные распределяются между рабочими группами (workgroups), которые являются объединением нескольких SIMD ядер. За счет локальной памяти внутри группы может происходить обмен данными между всеми рабочими группами. В графических процессорах отсутствуют средства для того, чтобы синхронизировать работу отдельных рабочих групп между собой. Но благодаря синхронизирующим примитивам рабочие группы обеспечивают синхронизацию между отдельными SIMD ядрами. На рис. 1.1 Рабочие группы имеют обозначение «Мультиядро i».

Память графического процессора также выстроена иерархически. На верхнем уровне иерархии рабочие группы оперируют глобальной видеопамятью, которая является общей для всего графического процессора. Глобальная память характеризуется как высокой пропускной способностью, так и высокой задержкой при обращении к ней. Все SIMD ядра, которые входят в мультиядро, оперируют локальной памятью. Локальная память, доступная внутри всей рабочей группы, имеет объем, как правило, не превышающий 16 Кбайт. Чтобы хранить промежуточные результаты при выполнении kernel функции, для каждого SIMD ядра предусмотрел небольшой объем собственной памяти. На рис. 1.1 эта память обозначена заштрихованной областью внутри P_i.

Приложение, которое использует графический процессор, предварительно копирует исходные данные в видеопамять до того, как будут запущены kernel функции. Это связано с тем, что последние используют

память, которая физически расположена на видеокарте. Но чтобы результаты, полученные на выходе работы приложения, стали доступны пользователю, приложение копирует их из видеопамяти в оперативную память (на рис. 1.1 двусторонняя стрелка между видеопамятью и оперативной памятью).

В качестве примера, видеокарта NVIDIA GTX 760 имеет 2GB видеопамяти, пропускную способность 192 GB/s и задержку на доступ в 980 тактов. При последовательном обращении SIMD ядер к видеопамяти в рамках рабочей группы происходит группировка всех обращений в одно фактическое обращение. Эта особенность снижает количество фактических обращений к видеопамяти и, как следствие, делает алгоритм более производительным, увеличивая для него общую пропускную способность данных [1].

Схема доступа к видеопамяти может быть различной. В самом простом случае последовательный доступ к памяти осуществляется таким образом, что SIMD ядро P_i обращается к ячейке памяти с номером i. Тогда все n обращений группируются в одно фактическое обращение, которое осуществляется блоком стандартного размера (128 Кбайт в современных видеокартах) (рис. 1.2а). Кроме обращений внутри блока никаких других обращений к видеопамяти не осуществляется. Когда доступ к памяти осуществляется с некоторым смещением, то число обращений к памяти возрастает на незначительную величину, поскольку крайние SIMD ядра рабочей группы обращаются к памяти соседних блоков (рис. 1.2б). Но если доступ к памяти не последовательный, то количество фактических обращений многократно возрастает.

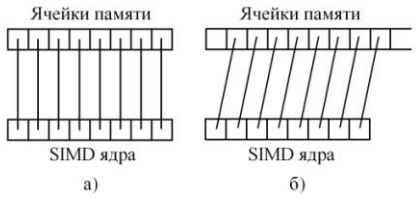

Рис. 1.2. Вариации доступа SIMD ядер к видеопамяти

Для каждой рабочей группы предусмотрен объем локальной памяти, равный 16Kb, которая хранит промежуточные данные в микропрограмме. Кроме того, локальная память характеризуется низкой задержкой на доступ. В результате, количество обращений к глобальной видеопамяти существенно сокращается.

Нужно учесть, что доступ к глобальной памяти можно оптимизировать как за счет ресурсов центрального процессора, выполняющего основную программу, так и за счет модификации микропрограмм, которые выполняются на графическом процессоре.

На сегодняшний день технологии программирования графического процессора можно разделить на интерфейсы программирования трёхмерной графики, архитектурно-зависимые интерфейсы и архитектурно-независимые интерфейсы. Интерфейсы программирования трехмерной графики в настоящее время теряют актуальность. Архитектурно-зависимые интерфейсы, напротив, применяются все чаще. К ним относят технологии программирования, как NVIDIA CUDA и AMD Brook+. OpenCL относят к последней группе – архитектурно-независимым интерфейсам. Дополняют эту группу различные системы метапрограммирования (RapidMind).

Описанные выше технологии программирования графического процессора относят к низкоуровневым средствам программирования, содержащим промежуточный ассемблер, который необходим разработчикам для создания компиляторов и языком программирования. Кроме того,

низкоуровневые средства программирования содержат С-подобный язык, применяемый непосредственно программистами графического процессора.

При таком подходе программисту предоставляется полный контроль над архитектурой графического процессора. К преимуществам низкоуровневых средств программирования относят и возможность работы с несколькими графическими процессорами, асинхронное выполнение задач на них, а также возможность загрузки скомпилированного шейдера.

Но чтобы пользоваться низкоуровневыми средствами программирования, необходимо учитывать специфику архитектуры конкретного графического процессора, средств разработки и отсутствие переносимости между графическими процессорами различных производителей. Программы, написанные на низкоуровневых средствах программирования, имеют большой объем и трудны для чтения.

В связи с этим, в последнее время наряду с низкоуровневыми средствами программирования разрабатывают и высокоуровневые. Как правило, при этом в программы, написанные на языке С или Fortran, добавляют специальные директивы размещения данных и компиляции блоков кода для использования на графическом процессоре. Однако ни одна из подобных директив не является стандартной и понимается только компиляторами соответствующих производителей (PGI, CAPS HMPP). Кроме того, новый поход не исключает обращения к низкоуровневым средствам.

С целью создания программ на основе графического процессора, в науке рассматривается парадигма расширяемых языков. Расширяемый язык – это язык программирования, в котором синтаксис и семантика когут быть как расширены, так и изменены с целью облегчения написания программы (NUDA).

1.2. Обзор и сравнительный анализ моделей и методов прогнозирования

Говоря о проблеме прогнозирования, имеется ввиду кратко- или среднесрочный прогноз, поскольку долгосрочный прогноз требует использования статистического анализа метода экспертных оценок [63].

Перед приложениями, реализующими какую-либо прогностическую модель, стоит ряд требований:

- сравнительная простота;
- экономичность вычислений;
- наличие возможности автоматического построения прогноза.

В литературе встречается множество разнообразных классификаций методов прогнозирования. Одна из классификаций – это по степени формализации, по такой классификации все методы прогнозирования разделяются на интуитивные и формализованные.

Для того чтобы определить, каким методам прогнозирования надо пользоваться в различных ситуациях при решении задачи, вводят понятие глубины прогноза:

$$\tau = \frac{\Delta t}{t}, \tag{1.1}$$

где Δt – абсолютное время упреждения, t – время продолжительности цикла прогнозирования. Когда значение глубины прогнозирования меньше единицы, можно говорить о том, что данное значение укладывается в рамки эволюционного цикла и наиболее верным будет использование формализованных методов. Если полученное значение становится близким к единице, то для того, чтобы определить силу «скачка» прогнозирования и время его существования, наиболее приемлемыми оказываются интуитивные методы и теория катастроф [34]. Если значение глубины прогноза больше единицы, это означает, что в прогнозном периоде помещаются сразу несколько эволюционных циклов прогнозирования. В этом случае все большее значение имеют интуитивные методы.

Интуитивные методы прогнозирования применяются для тех случаев, когда исследуемые процессы невозможно формализовать. Использование данных методов прогнозирования предоставляет возможность получить прогнозную оценку состояния развития объекта в будущем, не учитывая или слабо учитывая информационную обеспеченность. Интуитивные методы основаны на построении рациональной процедуры интуитивно-логического мышления человека. В тесной взаимосвязи с данными методами идут количественные методы оценки и обработки полученных результатов, базирующиеся на обобщенном мнении экспертов [35]. В сложных системах прогнозирования экспертные оценки выполняют роль исходных данных.

Для метода экспертных оценок характерна научно обоснованная организация экспертизы на всех ее этапах, которая включает в себя:

- создание репрезентативной экспертной группы;
- подготовка и проведение экспертизы;
- статистическая обработка результатов, полученных при опросе.

На оценки экспертов влияют как внешние факторы, так и внутренние. Внешние факторы, независимые от личности эксперта, определяются степенью его доступа к данным, точностью поставленных перед ним вопросов, погрешностью опросной модели.

Внутренние факторы зависят от персональных качеств эксперта. При выборе экспертов используют следующие критерии:

- уровень подготовленности эксперта;
- трудовой стаж эксперта в сфере, связанной с объектом, относительно которого осуществляется прогноз;
- отстаивание своих взглядов, несмотря на стереотипы, которые сложились ранее;
- нелинейное мышление, которое позволяет анализировать задачу с разных точек зрения.

Эксперт имеет в своем распоряжении следующие методы:

- документальный метод;
- экспериментальный метод;
- метод самооценки;
- прием исключения;
- попарное сравнение кандидатов;
- метод «приятелей».

В зависимости от того, как организована экспертная оценка и форма опроса экспертов, интуитивные методы прогнозирования делятся на два вида:

- метод индивидуальных экспертных оценок;
- метод коллективных экспертных оценок [36];

Методы коллективных экспертных оценок включают в себя: метод коллективной генерации идей, метод Дельфи, метод экспертных комиссий и другие.

Метод коллективной генерации идей направлен на получение большого их количества. При проведении опроса среди экспертов с помощью данного метода проблема формулируется в базовых терминах с обязательным выведением центрального вопроса.

Как показывает практика, эффективность группового мышления на 70% выше результатов индивидуальных экспертных оценок. Наиболее существенным недостатком метода коллективной генерации идей принято считать высокий уровень информационного шума – данных, которые не представляют ценности для проведения опроса.

Метод Дельфи предполагает, что определенное количество независимых экспертов способно более точно оценить и предсказать результат, чем организованный коллектив. Благодаря независимости мнений отсутствуют столкновения мнений и не допускается влияния большинства на меньшинство.

Слабой стороной метода Дельфи оказываются большие затраты времени: каждый тур продолжается не менее суток.

Метод экспертных комиссий представляет собой проводимую в формате дискуссий работу, результатом которой оказывается специальный документ о перспективах развития объекта прогнозирования – конкретизируются направления его развития, определяются цели и подцели, выбираются наилучшие средства для их достижения. По результатам опроса составляется таблица, в которую по каждому из направлений исследования записываются оценки в заданном диапазоне.

Среди индивидуальных методов экспертных оценок наибольшую популярность имеют метод «интервью», метод анкетного опроса, аналитический метод. Преимущества данных методов заключается во внимании к индивидуальным способностям эксперта. Однако, как правило, эксперт оказывается не в состоянии оценить масштабные стратегии ввиду одностороннего развития.

Формализованные методы прогнозирования отличаются большим разнообразием (рис. 1.3).

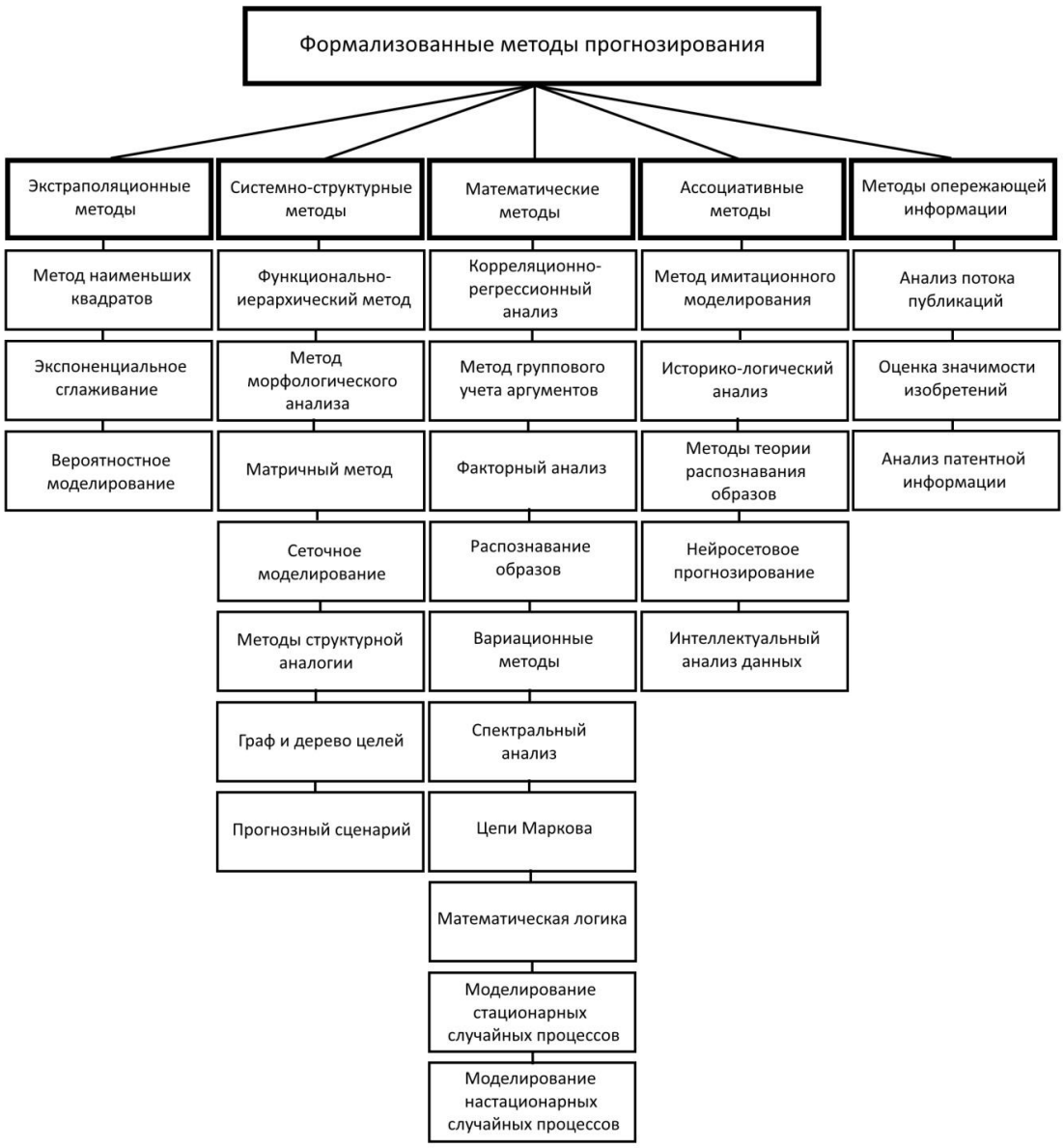

Рис. 1.3. Классификация формализованных методов прогнозирования

Несмотря на все многообразие формализованных методов прогнозирования, перечень, представленный на рисунке 1.3, не является исчерпывающим.

Среди этих методов одним из наиболее распространенных является экстраполяция. В наиболее общем смысле экстраполяция изучает тенденции прошлого и настоящего, чтобы перенести их в будущее. В экстраполяционном прогнозировании исследуются временные ряды и

находятся значения функции за пределами области определения [37, 38]. При этом используется информация о том, как ведет себя функция в точках, лежащих внутри области определения.

Экстраполяция бывает формальной и прогнозной. Формальная экстраполяция базируется только на предположении о сохранении прошлых и настоящих тенденций в будущем. Прогнозная экстраполяция опирается на гипотезы о дальнейшем развитии прогнозируемого объекта.

Выбор пределов экстраполяции в значительной мере определяет, насколько реальным окажется прогноз.

Ход экстраполяции определяется следующими шагами:

- конкретная постановка задачи, рассмотрение гипотез о развитии объекта прогнозирования, выявлении способствующих или препятствующих развитию объекта факторов, установление дальности экстраполяции;
- определение системы параметров и стандартизация связанных с ними единиц измерения;
- сведение собранных данных в таблицу, проверка данных;
- обнаружение тенденций развития прогнозируемого объекта.

Метод экстраполяции находит наиболее успешное применение в тех случаях, когда требуется определить сдвиг и некоторые новые закономерности в развитии объекта. Однако, чтобы получить конкретное поведение объекта или значение параметра, необходимо прибегать к специальным приемам, позволяющим повысить точность прогноза.

Суть метода наименьших квадратов заключается в минимизации суммы квадратичных отклонений наблюдаемых и расчетных величин.

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \to \min, \qquad (1.2)$$

где S – сумма квадратичных отклонений, y_i – расчетные значения исходного ряда, $f_i(x)$ – фактические значения исходного ряда, x – искомый набор независимых параметров, при котором левые и правые части систем будут максимально близки.

Чтобы избежать значительных ошибок при процедуре, основанной на данном методе, нужно учитывать ряд необходимых условий [62].

При экстраполировании можно использовать различные функции, однако наиболее часто применяют линейную. Экстраполирование методом наименьших квадратов позволяет получить достаточно точный прогноз только на короткое время. Недостатком данного метода является то, что требуется большое количество опытных данных о поведении системы в прошлом.

В некоторых задачах прогнозирования используются цепи Маркова [39]. Простые марковские цепи представлены в литературе [40]. Цепи Маркова относятся к моделям случайных процессов. Для анализа подобных процессов строятся множества состояний и вероятностей. На рис. 1.4 изображен пример цепи Маркова с тремя состояниями:

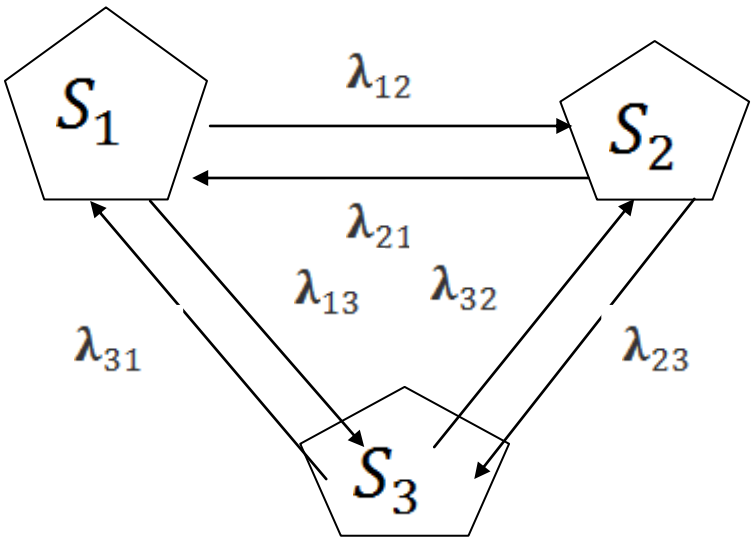

Рис. 1.4. Цепь марковского процесса, имеющего три состояния

S_1, S_2, S_3 – состояния процесса, λ_{12} – параметр, пропорциональный вероятности, с которой процесс может перейти из состояния S_1 в состояние S_2, λ_{13} – параметр, пропорциональный вероятности, с которой процесс может перейти из состояния S_1 в состояние S_3 и так далее. Состояние S_1 считается начальным и задается либо заранее, либо выбирается случайным образом.

При текущем состоянии процесса в качестве следующего выбирают такое, вероятность перехода в которое будет максимальной.

В работе [41] предлагается метод прогнозирования временных рядов с использованием сложных цепей. Преимущество такого алгоритма заключается в использовании иерархии временных приращений. Особое значение имеет процедура объединения, в результате которой несколько прогнозов с различной дискретизацией соединяются в один прогнозный ряд. К недостаткам предложенного подхода можно отнести игнорирование наименее вероятных прогнозов, при которых характер кривой существенно отличается.

В последнее время при прогнозировании применяется метод, использующий теорию распознавания образов [42, 43]. На первом этапе прогнозирования с использованием теории распознавания образов необходимо выбрать такие классы состояний прогнозируемого объекта, которые можно представить в виде диапазонов изменения определенных параметров или некоторых качественных характеристик. По набору признаков, которые определяют состояние объекта, устанавливается принадлежность нового состояния объекта к тому или иному классу. На основании этого указывается диапазон изменения параметров, который и определяет состояние исследуемой системы в будущем.

В случае, когда необходимо дать конкретный прогноз, существенным недостатком становится сложность в оценке исходных данных. Поскольку развитие сложной системы может описываться большим количеством показателей, необходимо рассматривать все возможные взаимосвязи между ними. Однако в рамках теории распознавания образов удается существенно минимизировать описываемую проблему.

Большинство реально протекающих процессов являются нелинейными. Поскольку природа таких процессов смешана и хаотична, не представляется возможным формализовать их, используя традиционные статистические

модели. В связи с этим, одним из наиболее гибких методов при решении задач прогнозирования оказываются искусственные нейронные сети [44, 45].

Искусственные нейронные сети используют принципы организации и функционирования биологических нейронных сетей – клеток, образующих нервную систему живого организма [46]. От традиционных алгоритмов искусственные нейронные сети отличает способность к обучению, самоорганизации и адаптации.

Нейронная сеть представляет собой объединение нейронных элементов с учетом существующих между ними связей [47]. Базовым элементов нейронной сети является формальный нейрон. Он отвечает за нелинейное преобразование суммы произведений входных сигналов на весовые коэффициенты [48].

Нейронные сети предполагают процесс обучения, которое происходит путем предоставления сети обучающих последовательностей. Существует множество способов обучения нейронной сети, от выбора которых значительно зависит эффективность модели прогнозирования.

Одним из классических методов обучения нейронной сети является метод обратного распространения ошибки, который наиболее подходит для многослойных нейронных сетей прямого распространения. Процесс обучения в таком алгоритме сводится к минимизации среднеквадратичной ошибки нейронной сети. Весовые коэффициенты сети модифицируется посредством реализации стохастического градиентного спуска, при котором после каждого обучающего примера веса пересчитываются. Далее осуществляется поиск ошибки в стороне, противоположной градиенту. Замечено, что если от постоянного шага обучения перейти к адаптивному, то скорость алгоритма обучения увеличивается в четыре раза. Адаптивный шаг подбирается на каждом этапе обучения таким образом, чтобы свести к минимуму среднеквадратичную ошибку.

В диссертации [49] автор описывает использование гетерогенных нейронных сетей. Гетерогенной называют нейронную сеть, у которой

имеются различные функции активации. Гетерогенная нейронная сеть показывает наилучшие результаты при решении задач прогнозирования.

Анализ существующих видов нейронных сетей показывает, что их применение позволяет решать задачи широкого спектра в различных отраслях науки.

Использование аппарата нейронных сетей при прогнозировании имеет ряд проблем:

- количество слоев в нейронной сети и количество нейронов в нем является неопределенным;
- поиск минимума среднеквадратичной ошибки существенно зависит от случайного выбора весовых коэффициентов нейронной сети;
- градиентный метод не может установить, какой из локальных минимумов будет глобальным;
- медленная сходимость градиентного метода при выборе постоянного шага обучения.

В связи с развитием нейронных сетей, как математической модели прогнозирования, особое значение приобретают генетические алгоритмы. Генетические алгоритмы необходимы в тех случаях, когда при решении задач прогнозирования применяется интуиция и опыт. Генетические алгоритмы тесно связаны с нейронными сетями. Аппарат нейронных сетей часто используется вместе с генетическими алгоритмами [50]. В частности, генетические алгоритмы помогают при подборе весов нейронной сети. Нейронные сети, в свою очередь, используются при выборе соответствующих параметров самих генетических алгоритмов. В данном случае начальная популяция состоит из нейронных сетей, генетический алгоритм и его основные операции: репродукция, скрещивания, мутации и отбора особей тоже рассматриваются на них.

Методика, основанная на генетических алгоритмах, содержит много вычислений. Однако стремительное развитие параллельных вычислений решает эту проблему.

Главным преимуществом использования генетических алгоритмов в задачах прогнозирования является отсутствие необходимости дополнительной информации, что значительно ускоряет поиск.

Еще одним методом прогнозирования, который применяется в условиях неопределенности, является аппарат нечеткой логики множеств. Нечеткая логика позволяет определить промежуточные значения для общепринятых оценок да/нет, черное/белое и др. Выражения «слегка тепло» или «довольно холодно» возможно формулировать математически и обрабатывать на компьютере. Вводится понятие лингвистической переменной, ее значениями выступают нечеткие множества. Заде создал аппарат для описания процессов интеллектуальной деятельности, включая нечеткость и неопределенность выражений. Нечеткая логика обеспечивает эффективные средства отображения неопределенностей и неточностей реального мира. Наличие математических средств отражения нечеткости исходной информации позволяет построить модель, адекватную реальности [64].

1.3. Обзор и сравнительный анализ бионических алгоритмов

Использование биологических систем является перспективным и активно применяемым в современной научной практике. Методология, предлагаемая искусственным интеллектом, имеет в качестве прообраза природные аналоги. В частности, искусственные нейронные сети обобщают наиболее существенные принципы биологических нейронных сетей, а аппарат нечеткой логики используется при решении задач в условиях наличия недостаточно структурированной и неполной информации, которую обрабатывает человеческий мозг.

Научный интерес представляют не только решения, которые «подсказывает» природа, но и предшествующие им процессы. Это находит практическое выражение в многообразии эволюционно-генетических и бионических методов моделирования. Первые работы, описывающие элементы генетических алгоритмов принадлежат Н.А. Баричелли [66, 67].

Значительный вклад в развитие эволюционного программирования внесли Л. Фогель, А. Оуэнс и М. Уолш [68]. Решающая роль в формировании современной теории генетических алгоритмов отводится Д. Холланду. В своей книге, которая переводится как «Адаптация в естественных и искусственных системах», он не только впервые вводит понятие генетического алгоритма, но и предлагает канонический генетический алгоритм [69]. Впоследствии ученики Холланда развили созданную им теорию. Отечественный вклад в развитие бионических идей принадлежит Л.А. Растригину, который разрабатывая теорию случайного поиска предложил нескольких алгоритмов, опирающихся на бионическую модель.

Развитие вычислительной техники подтолкнула исследования генетических алгоритмов к новому витку. Благодаря этому в настоящее время генетические алгоритмы находят самое широкое применение. С помощью генетических алгоритмов решаются всевозможные задачи на графах, настраивается и обучается искусственная нейронная сеть, составляются расписания, разрабатываются игровые стратегии и многое другое. Наиболее популярным применением генетических алгоритмов является оптимизация многопараметрических функций [70,71,72]. Кроме генетических алгоритмов имеются и другие оптимизационные алгоритмы. Генетические алгоритмы имеют следующие отличия:

- алгоритм оптимизационной задачи работает с закодированными параметрами;
- при решении задачи оптимизации используется непосредственно целевая функция с целью определить «вектор движения поиска» и качественно сравнить альтернативные решения;
- поиск улучшения решений осуществляется в соответствии с несколькими альтернативами;
- выбор генетических операторов опирается на использование понятие вероятности.

В зависимости от различных факторов эффективность применения генетических алгоритмов существенно варьируется.

К этим факторам относятся:

- комбинация и различные модификации используемых генетических операторов;
- выбор соответствующих значений параметров и особенности их настройки;
- способ представление задачи на хромосоме и вариации представления начальной популяции;
- особенности методологии кодирования решения в хромосомах, учитывающие специфику исходной задачи [73];

Особенности генетического алгоритма, используемого в главе 3 настоящей работы, определяются на этапе представления генов в хромосоме. От выбранного способа представления зависит тип кодирование генов [74]. С момента зарождения теории генетических алгоритмов осуществляется непрерывный поиск эффективных способов кодирования информации в хромосоме. Выделяют следующие способы кодирования:

- двоичное кодирование;
- вещественное кодирование;
- целочисленное кодирование.

Наиболее используемым в задачах оптимизации является двоичное кодирование информации в генах, когда хромосома представляет собой битовую строку. Этот способ отвечает канонической реализации генетического алгоритма. Принято считать, что алгоритмы двоичного кодирования наилучшим образом справляются с эффективной обработкой максимального количества информации.

Проблема двоичного кодирования заключается в том, что расстояние Хэмминга для двух закодированных значений признака, может быть ненулевым. Более подробно об этом говорится в работе [75]. Эту проблема нивелируют, прибегая к использованию кода Грея, особенностью которого

является отличие соседних чисел не более, чем на один бит. Данная особенность кода является наиболее оправданной в использовании оператора мутации. Такой подход используется в работе [76].

Использование двоичного кодирования генов в хромосомах на практике создало предпосылки к выходу за рамки канонического генетического алгоритма в сторону поиска более эффективных решений. В настоящей работе предлагается отказ от двоичного кодирования ввиду следующих проблем.

Чем больше размерность непрерывного пространство, внутри которого осуществляется поиск, тем сложнее получить точное решение. Чтобы показать это, необходимо разбить исходный интервал со множеством допустимых значений целевого признака на отрезки с точностью, необходимой для получения решения.

$$\rho = \frac{b_i - a_i}{2^{N-1}}, \qquad (1.3)$$

, где ρ – точность, a_i b_i – границы отрезков разбиения, N – количество разрядов битовой строки.

Из формулы (1.3) видно, что точность найденного решения при выбранном подходе двоичного кодирования сильно зависит от разрядности хромосомы.

Несмотря на простоту реализации двоичного кодирования на ЭВМ, появилась необходимость выйти за рамки элементов классического генетического алгоритма при решении задач оптимизации на непрерывных пространствах, в случаях когда решение необходимо представить набором вещественных чисел. Такой алгоритм называется непрерывным генетическим алгоритмом. Начиная с 1991 года, в научной печати стали появляться первые обоснованные теоретические работы по непрерывным генетическим алгоритмам. Сегодня алгоритмы вещественного кодирования являются реализуемой на практике альтернативой генетическим алгоритмам двоичного кодирования [77].

Успех алгоритмов вещественного кодирования объясняется естественностью представления решения в виде набора вещественных чисел в тех случаях, когда пространство поиска является непрерывным. Это базовое преимущество, из которого вытекают все остальные. В случае вещественного кодирования, диапазон чисел, представляемых в современных ЭВМ, определяет и гарантирует точность получаемых результатов. Выбранный подход обусловливает взаимооднозначное соответствие генов и переменных. Совокупные преимущества генетических алгоритмов вещественного кодирования можно охарактеризовать следующим образом:

- возможность поиска решений в больших пространствах без ущерба точности решения;
- повышенная скорость работы алгоритма в связи с отсутствием операции двоично-десятичного преобразования;
- возможность гибридизация генетического алгоритма с алгоритмами локального поиска;
- простота и естественность представления решений, соответствующие оптимизационным задачам прикладного характера.

В отличие от упомянутых работ, в данной работе в полной мере учитывается факт соответствия непрерывным генетическим алгоритмам вещественного кодирования собственному уникальному набору генетических операторов, выходящих за рамки элементарного генетического алгоритма. Например, для оператора мутации появляется возможность установить долю мутируемых генов от общего количества генов особи, участвующей в отборе.

Наибольший интерес с точки зрения эффективности решения задачи оптимизации генетическими алгоритмами вещественного кодирования представляют вариации кроссовера [78,79].

В данной работе выбор остановлен на $BLX-\alpha$ кроссовере. Удается существенно снизить разрущающую способность данного оператора,

поскольку критерии, минимизирующие это значение, адекватно вписываются в заданные постановкой задачи условия.

1.4.Вопросы моделирования динамики сыпучих материалов

1.4.1. Обзор и сравнительный анализ современных методов моделирования динамики сыпучих материалов

К сыпучим материалам относятся грунт, снег, руда, зерно, порошок, песок, удобрения, гранулированные комбикорма, лекарства и т.д. Сыпучие материалы участвуют в производственных процессах, поэтому теория их изучения является одним из наиболее интересных и развивающихся разделов механики.

Только в XX веке ученые поняли, что сыпучие материалы могут вести себя как твердое тело, жидкость или газ, это отличает их от других веществ. К концу XX века сыпучие среды начали изучаться в рамках специальной науки – механики грунтов. При моделировании динамики грунтов использовалась модель сплошной среды.

Большой вклад в этом направлении внесли советские ученые (Н.М. Герсеванов [1], Д.Е. Польшин [1], В.А. Флорин [2, 3], В.В.Соколовский [4, 5], Н.А. Цитович [6, 7], Н.Н. Маслов [8], В.Г. Березанцев [9, 10] и другие).

Однако концепция сплошной среды имеет ряд ограничений, а именно, размеры участвующих в эксперименте частиц сыпучей среды должны быть намного меньше, чем минимальные характерные размеры рассматриваемого пространства:

$$\sqrt[3]{\Delta V} \leq h, b, \tag{1.3}$$

где ΔV – элементарный бесконечно малый объём грунта, h – высота откоса, b – ширина подошвы фундамента и т.д., необходимо исключить всякое влияние отдельно взятой частицы сыпучей среды. Поэтому требуется выполнение дополнительного условия:

$$\sqrt[3]{\Delta V} \geq d_{max}, \tag{1.4}$$

где d_{max} – диаметр наибольшей по величине частицы сыпучего материала.

Современные методы компьютерного моделирования динамики сыпучих материалов опираются преимущественно на концепцию дискретного представления вещества. Семейство таких методов объединяется под общим названием – метод дискретных элементов [11,12].

Метод дискретных элементов был впервые применен Cundall в 1971 году при решении задач механики горных пород [13]. Позднее Williams, Hocking и Mustoe уточнили теоретические основы метода [14]. В частности, в 1985 году они продемонстрировали то, что метод дискретных элементов можно рассматривать как обобщение метода конечных элементов. В книге [15] проиллюстрировано применение метода дискретных элементов к задачам геомеханики. Многие из теоретических аспектов метода были вынесены в положения международных конференций. Текущие направления в области метода дискретных элементов обозначены в статьях [16, 17, 18, 19]. Сочетание метода дискретных элементов с методом конечных элементов рассматривается в книгах [20].

В течение последних десятилетий появились несколько разновидностей метода дискретных элементов: метод отдельных элементов [13], обобщенный метод дискретных элементов [14], дискретный деформационный анализ [19] и метод конечных дискретных элементов [20]. В России метод дискретных элементов получил развитие только в последние годы. Впервые для реальных расчетов его использовали Ю.М. Левкин и И.М. Иофис. Получили известность ряд работ, в которых метод дискретных элементов был применен по отношению к сыпучим средам (Г.Н. Хан, А.А. Барях, Е.Н. Дьяченко, С.О. Дорофеенко, С.В. Клишин, В.А. Андронова, И.Г. Дик)

Метод дискретных элементов позволяет производить расчет движения большого количества частиц – молекул, песчинок, гравия, гальки и других гранулированных сред. Впервые данный метод был применен в механике

горных пород. Моделирование процессов методом дискретных элементов начинается с указания следующих данных – начальных положений частиц и скоростей. Задаются физические законы взаимодействия частиц и рассчитываются силы, действующие на каждую частицу. Располагая результирующей силой для каждой частицы, переходят к решению задачи Коши на определенном временном шаге. В зависимости от сил, действующих на частицу, ее скорость и координаты центра пересчитываются. Каждая частица переходит в новое положение с другими значениями сил взаимодействия с остальными частицами. Процесс продолжается в течение времени, необходимому пользователю.

Метод дискретных элементов имеет ограничение по требованиям к вычислительным ресурсам ЭВМ. Однако благодаря появлению распределенных алгоритмов удается существенно ускорить вычислительный эксперимент, увеличить исходное количество частиц в нем.

В настоящее время метод дискретных элементов реализуют следующие комплексы программ: YADE, LIGGGHTS, Chute Maven (Hustrulid Technologies Inc.), PFC2D, PFC3D, EDEM (DEM Solutions Ltd.), GROMOS 96, ELFEN, MIMES, PASSAGE®/DEM. Эти программные продукты позволяют сделать расчеты движения частиц, однако, любая из них позволяет получить усредненную картину процесса [21].

В работе [22] для моделирования сыпучих сред методом дискретных элементов автором был выбран пакет YADE1, предоставляющий каркас, основанный на подходе Лагранжа. Данное программное обеспечение имеет открытый исходный код, пользоваться которым можно только, обладая лицензией GPL. Пакет YADE содержит центральное ядро, ряд служебных библиотек, графическую оболочку и плагины. Все это в совокупности позволяет реализовать главный цикл метода. Пакет YADE предлагает базовые средства ввода/вывода, пользовательский интерфейс и визуализацию

результатов моделирования. При разработке программных модулей используется язык программирования C++ [23].

Комплекс программ, разработанный автором при помощи пакета приложения YADE, производит распараллеливание на уровне потоков. Это становится возможным благодаря открытому стандарту для распараллеливания программ Open MP [24, 25]. Данный стандарт описывает совокупности директив компилятора, библиотечных процедур и специальных переменных, необходимых при программировании многопоточных приложений, реализуемых многопроцессорными системами с общей памятью. На сегодняшний день разработчики центральных процессоров подходят к «кремниевому тупику», когда дальнейший рост тактовой частоты становится невозможным в силу целого ряда серьезных технологических причин. В связи с этим, производители современных вычислительных систем стараются увеличить число процессоров и ядер. На сегодняшний день в свободном доступе находятся процессоры, состоящие из 4 ядер. Последние разработки увеличили их число до 8. В предложенной автором работе центральный процессор позволяет рассчитать только небольшое количество систем за приемлемое время (около 16 тысяч). Это существенно ограничивает пользователя.

Центральный процессор предусматривает параллелизм на уровне потоков, каждое ядро процессора способно исполнить только 1-2 такта.

В рассматриваемой работе авторский подход ограничен и тем, что для реальной визуализации динамики частиц необходимо использовать достаточно мелкие сеточные и временные шаги. Это приводит к потребности применения большого числа сеточных узлов (порядка более 100 на каждое пространственное измерение). В этом случае может потребоваться около терабайта оперативной памяти, тогда как современные ОЗУ имеют в своем распоряжении только 1-8 Гб памяти. Проблема эффективной реализации

метода дискретных элементов на многопроцессорных вычислительных системах остается крайне актуальной.

Среди открытого программного обеспечения, реализующего метод дискретных элементов, широко применяется пакет программирования LAMMPS (Large-skale Atomic/Molecular Massively Parallel Simulator) [26, 27]. Следует отметить, что данный пакет используется для широкого спектра задач молекулярной динамики, при моделировании движения сыпучих сред применяют модифицированную версию пакета – LIGGGHTS (LAMMPS Improved For General Granular And Granular Heat Transfer Simulations) [28]. Такое применение наглядно проиллюстрировано в статье [29], где автор ставит перед собой задачу повысить эффективность производства угольной и графитовой электродной продукции, а также уменьшить при этом энергозатраты.

Серия авторских расчетов и экспериментов подтверждает адекватность и реалистичность модели, однако, в работе не указано реальное время, за которое можно произвести расчет движения всех частиц системы. На сегодняшний день мощность центральных процессоров современных компьютеров, является недостаточной для моделирования движения сотен тысяч гранулированных частиц.

При расчетах оперативная память работает медленнее центрального процессора, поэтому требуются оптимизированный подход к ее использованию, для центральных процессоров эта проблема решается путем использования кэшей – промежуточных буферов, в которые помещаются наиболее используемые участки памяти. Однако соответствующие аппаратные части занимают большую часть центрального процессора и потребляют много энергии.

1.4.2. Методы расчета устойчивости грунтового массива

Грунт – горные породы, почвы, техногенные образования, представляющие собой многокомпонентную и многообразную

геологическую систему и являющиеся объектом инженерно-хозяйственной деятельности человека.

Различают:

- скальные и полускальные грунты – монолитные грунты с жёсткими структурными связями;
- дисперсные грунты – раздельнозернистые грунты без жёстких структурных связей: связные – глинистые, и несвязные – песчаные и крупнообломочные.

Грунты могут быть использованы в качестве оснований зданий и различных инженерных сооружений, материала для сооружений (дорог, насыпей, плотин), среды для размещения подземных сооружений (тоннелей, трубопроводов, хранилищ) и др.

Массив грунта при определенных условиях может потерять устойчивость и в результате этого перейти из состояния статического равновесия в состояние движения. Такое состояние грунтового массива называется оползнем.

Различают следующие виды оползней: оползни вращения; оползни скольжения; оползни разжижения (рис. 1.5).

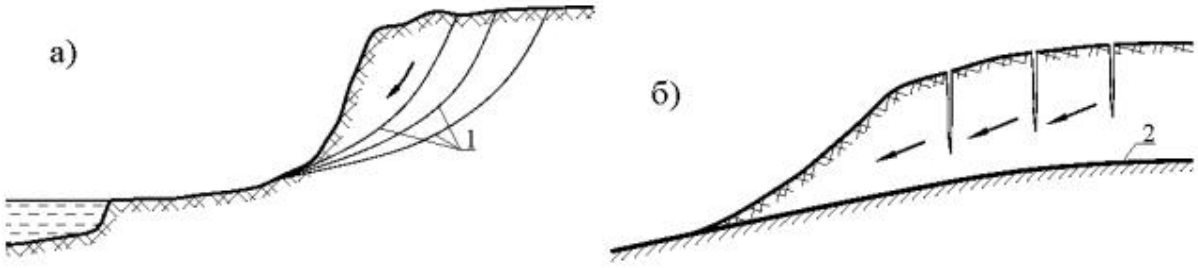

Рис. 1.5. Виды оползней: а – оползень вращения; б – оползень скольжения (пристенный оползень); 1 – поверхность скольжения в теле оползня; 2 – стационарная плоскость скольжения на границе оползня с подстилающим устойчивым массивом

Для расчета устойчивости оползней вращения используется метод круглоцилиндрических поверхностей скольжения [61]. Название метода происходит из гипотезы о круглоцилиндрической поверхности скольжения, по которой происходит поступательно-вращательное движение верхней части грунтового массива, когда он теряет устойчивое положение. В настоящей работе выдвигается гипотеза о несжимаемом составе грунта.

Чугаев Р.Р. в своей работе описывает поведение грунта при оползне следующим образом:

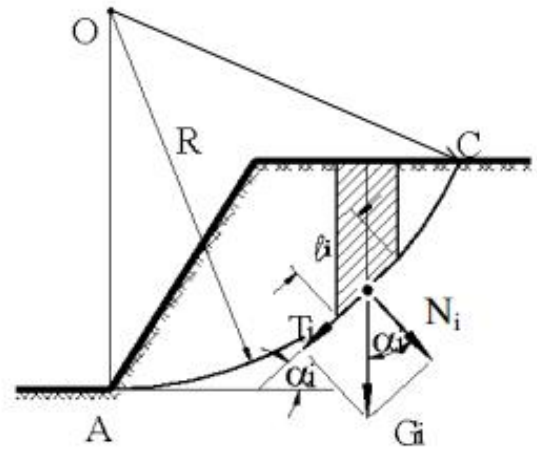

Рис. 1.6. Расчетная схема к определению устойчивости откоса методом круглоцилиндрических поверхностей скольжения

На рис. 1.6, иллюстрирующем вертикальный разрез откоса, бесконечно протяженного перпендикулярно плоскости чертежа, выбирается центр О круглоцилиндрической поверхности скольжения. Экспериментально установлено, что плоскость скольжения проходит через основание откоса в точке А. Окружность радиусом ОА с центром в точке О проходит через основание откоса А и выделяет верхнюю часть откоса. После разделения выделенной части откоса перпендикулярными чертежу вертикальными плоскостями на элементарные объемы рассматривается условие его равновесия. Для этого проводится центральная вертикальная ось площади произвольно взятого i–го элементарного объема и касательная к поверхности скольжения в точке ее пересечения с центральной осью. Вводится угол наклона касательной к горизонтальной оси α_i. Вес элементарного объема грунта обозначается G_i и прикладывается в точке пересечения центральной оси с поверхностью скольжения. Сила G_i раскладывается на нормальную и касательную к поверхности составляющие N_i и T_i:

$$N_i = G_i \cos \alpha_i; \; T_i = G_i \sin \alpha_i; \qquad (1.5)$$

Касательная составляющая силы тяжести является сдвигающей силой $T_{\text{i,сд}} = T_i$. Сила трения и сила сцепления по поверхности скольжения образуют удерживающие силы:

$$T_{\text{уд,i}} = \text{tg}\,\varphi_\text{i}\,\text{G}_\text{i}\cos\alpha_i + l_i c_i, \qquad (1.6)$$

где l_i – длина дуги поверхности скольжения в пределах i-го объема грунта;

c_i и φ_i – сцепление и угол внутреннего трения грунта в пределах дуги l_i.

Когда сумма моментов, сдвигающих и удерживающих сил относительно центра О круглоцилиндрической поверхности скольжения, равна нулю, говорят о равновесии по пересекающей откос поверхности скольжения АС:

$$R\sum_{i=1}^{n} G_i \sin\alpha_i - R\sum_{i=1}^{n}(tg\,\varphi_i\,G_i\cos\alpha_i + c_i l_i) = 0. \qquad (1.7)$$

Как правило, при анализе устойчивости грунтового массива вместо уравнения (1.7) применяют выражения для коэффициента устойчивости, определяемое как отношение момента удерживающих сил к моменту сдвигающих сил:

$$\eta = \frac{M_{уд}}{M_{сд}} = \frac{\sum_{i=1}^{n}(tg\varphi_i G_i\cos\alpha_i + c_i l_i)}{\sum_{i=1}^{n} G_i \sin\alpha_i} > 1. \qquad (1.8)$$

В формулах (1.7) и (1.8) угол α отсчитывается от горизонтали и считается положительным при повороте ее на острый угол до совмещения с касательной против хода часовой стрелки. При отрицательном угле α касательная составляющая силы тяжести и соответствующий ей момент являются удерживающими, что автоматически учитывается формулами (1.7) и (1.8). Предел суммирования по i,n определяет количество элементарных объемов грунта, на которые разбивается верхняя часть откоса, отделенная от остального массива поверхностью скольжения. От величины n зависит точность расчетов по формулам (1.7) и (1.8). Если значение коэффициента больше единицы, говорят об устойчивости. Данные формулы позволяют

получить не конечные, а промежуточные результаты в текущий момент времени.

1.5.Выводы

1. Анализ средств современных программных вычислительных систем, предназначенных для обработки информации, показывает ряд нерешенных проблем. Существующие алгоритмы обработки информационного потока сталкиваются с большим объемом вычислений и нерациональным использованием вычислительных ресурсов. В данной работе для ускорения и оптимизации вычислений предлагается использовать графический процессор.

2. К недостаткам многих моделей и методов прогнозирования можно отнести сложность оценки исходной информации, требования к объему исходной информации, отсутствие приспособленности к изменяющейся среде. В связи с этим, существует потребность в прогнозировании систем с неполной информацией и высокой сложностью объекта. В настоящей работе для этого применяются нечеткие методы.

3. Для решения задач оптимизации многопараметрических функций наиболее часто используются генетические алгоритмы. Использование канонической схемы генетического алгоритма показывает несостоятельность при решении многих современных прикладных задач. В данной работе настройка параметров оптимизации осуществляется с применением модифицированного генетического алгоритма неклассической вариации.

4. В настоящее время для расчета и моделирования движения многоточечных масс при формировании устойчивой сыпучей насыпи используется большое количество разнообразных комплексов приложений для получения усредненной картины процессов.

Реализация алгоритма моделирования динамики частиц настоящей работы опирается на распределенную вычислительную систему.

Глава 2. Алгоритм поведения системы многоточечных масс для формирования сыпучей насыпи

В настоящей работе ставится задача рассчитывать системы с большим количеством частиц за короткое время с помощью графического ускорителя [22, 29]. Необходимо разработать модель поведения частиц сыпучей среды. В рамках данной задачи изучается производительность вычислительного эксперимента относительно методов с классической реализацией, при которых основные вычисления выполняются на центральном процессоре.

Работа С.О. Дорофеенко [22] базируется на пакете приложений YADE1. Данный пакет отличается удобным интерфейсом, наглядно демонстрирующим результаты компьютерного моделирования. Преимуществом данной работы является обращение алгоритма к распараллеливанию на уровне потоков, что становится возможным за счет специальной директивы центрального процессора OpenMP. Однако в настоящее время выбранный автором подход перестает оправдывать себя за счет ограничения тактовой частоты многопроцессорных систем.

Другим пакетом, реализующим метод дискретных элементов, является LAMMPS. Анализ научной литературы, посвященный использованию данного программного обеспечения, не подтверждает данных о возможности эксперимента с моделированием движения сотен тысяч частиц.

С опорой на недостатки представленных методов в данной работе предложен алгоритм эффективного использования графических ускорителей.

2.1. Модель взаимодействия частиц

Для решения системы, содержащих большое количество частиц, применяется метод дискретных элементов. На частицы действуют силы, а их взаимодействие описывается определенными законами [11, 12].

Ряд научных работ, в которых рассматривается метод дискретных элементов, иллюстрирует применение двумерной модели [52, 53]. Это

существенно упрощает эксперимент, но ограничивает детали результата. В настоящей главе рассматривается трехмерное пространство R^3для метода дискретных элементов.

В данной работе частицы рассматриваются в сферической форме. Для каждой i-ой частицы указывается координата центра $\boldsymbol{x_i}$, масса m_i и радиус r_i. Масса находится по формуле [66]:

$$m_i = pV_i, \tag{2.1}$$

где ρ - плотность материала, V_i – объем частицы, который рассчитывается по формуле шара, ограниченного сферой:

$$V = \frac{4}{3}\pi R^3, \tag{2.2}$$

Движение частиц задается в дифференциальной форме в соответствии с законом Ньютона:

$$m_i \frac{d^2\vec{x}_i}{dt^2} = \vec{F}_i, \tag{2.3}$$

где $\vec{F}_i$ – результат сложения всех сил, которые воздействуют на частицу:

$$\vec{F}_i = \sum_{i \neq j} \vec{F}_{ij}^{\,вз} + \vec{F}_{гр}, \tag{2.4}$$

где $\vec{F}_{ij}^{\,вз}$ – сила, с которой i-я и j-я частицы взаимодействуют между собой. $\vec{F}_{гр}$ – сила гравитации.

В моделировании динамики сыпучих материалов за основу берется взаимодействие двух частиц. Чтобы детально описать данное взаимодействие, необходимо задать координаты центров частиц x_1 и x_2, массы m_1 и m_2, а также радиусы r_1 и r_2. Зная координаты центров и радиусы, становится возможным рассчитать степень проникновения ξ двух заданных частиц:

$$\xi = r_1 + r_2 - \|x_2 - x_1\|. \tag{2.5}$$

Отрицательное значение искомой величины указывает на то, что силами взаимодействия между частицами можно пренебречь. Когда показатель $\xi = 0$, следует учитывать силу трения. При положительном значении данного показателя, учитывают силу отталкивания. Обозначив силу отталкивания как $\vec{F}_o$ и силу трения как $\vec{F}_{mp}$, можно записать результирующую силу, с которой частицы взаимодействуют друг с другом:

$$\vec{F}_{вз} = \vec{F}_o + \vec{F}_{mp}. \quad (2.6)$$

Необходимо заметить, что сила взаимодействия одинакова по модулю для каждой и частиц, но противоположна по направлению.

Направление от центра одной частицы к центру другой задается нормализованным вектором N:

$$\vec{N} = \frac{\vec{x}_2 - \vec{x}_1}{\|x_2 - x_1\|}. \quad (2.7)$$

Сила отталкивания, действующая вдоль вектора N записывается в виде:

$$\vec{F}_o = -f_n \vec{N}, \quad (2.8)$$

где f_n – модуль силы отталкивания.

После задания скорости частиц $\vec{v}_1$ и $\vec{v}_2$, вычисляется относительную скорость $\vec{V}$, скорость вдоль нормального вектора V_n, а также тангенциальную скорость $\vec{V}_t$:

$$\vec{V} = \vec{v}_1 - \vec{v}_2, \quad (2.9)$$

$$V_n = (\vec{V} \cdot \vec{N}), \quad (2.10)$$

$$\vec{V}_t = \vec{V} - \vec{V}_n. \quad (2.11)$$

При известной скорости вдоль нормального вектора V_n, вычисляется значение модуля силы отталкивания [54]:

$$f_n = k_n \xi + k_d V_n, \quad (2.12)$$

где k_n – вычисляемый экспериментально коэффициент упругости, k_d – коэффициент демпфирования, выражаемый посредством коэффициента восстановления после удара [53, 55]:

$$k_d = -2\ln(\epsilon)\sqrt{\frac{m_{ij}k_n}{\pi^2+\ln^2(\epsilon)}}, \tag{2.13}$$

m_{ij} обозначает приведенную массу двух частиц. Она считается по формуле:

$$m_{ij} = \frac{m_i m_j}{m_i+m_j}. \tag{2.14}$$

Сила трения считается направленной в сторону, противоположную тангенциальной скорости:

$$\vec{F}_t = -k_T\vec{V}_t, \tag{2.15}$$

где k_T является коэффициентом трения. Данная формула не учитывает силу отталкивания. В работе [54] предлагается подход с учетом силы отталкивания:

$$\vec{F}_T = -\min(\mu f_n, k_T\|V_t\|)\frac{\vec{V}_t}{\|V_t\|}, \tag{2.16}$$

где μ – коэффициент трения.

Моделирование динамики частиц сыпучей среды должно учитывать и статическое трение. Для того, чтобы это стало возможным, используют следующие способы:

- переход от сфер к полигонам [56];
- применение виртуальных пружин, действующих при контакте между частицами [54];
- объединение нескольких сфер в одну с использованием пружин [57].

Говоря о рассматриваемых при взаимодействии физических параметрах системы k_n, ϵ и μ, необходимо сделать замечание о том, что условия рассматриваемой задачи нуждаются в двух наборах этих параметров.

Данное требование находит свое объяснение в том, что частицы взаимодействуют не только между собой, но также и с границами системы.

2.2. Разработка алгоритма, моделирующего динамику частиц

Для решения практической задачи необходимо перейти от формулы (2.5) к другой ее записи, которая позволит использовать промежуточные значения скорости v. Система уравнений запишется следующим образом:

$$\begin{cases} m_i \dfrac{d\vec{v}_i}{dt} = \vec{F}_i, \\ \dfrac{d\vec{x}_i}{dt} = \vec{v}_i. \end{cases} \tag{2.17}$$

На основе метода Эйлера для численного дифференцирования, записывается следующая система [38]:

$$\begin{cases} \vec{v}_{i+1} = \vec{v}_i + \vec{F}_i \dfrac{1}{m_i} \Delta t, \\ \vec{x}_{i+1} = \vec{x}_i + \vec{v}_{i+1} \Delta t, \end{cases} \tag{2.18}$$

где:

i – номер шага, Δt – шаг по времени. Порядок шага должен составлять одну миллионную. В противном случае дискретная система может оказаться вырожденной.

Реализовать алгоритм описанной модели предлагается посредством программно-аппаратной архитектуры распределенных вычислений [66, 70, 71]. При этом предпочтение отдается технологиям, ориентированным на выполнение большого объема вычислений, нежели на контроль выполнения и кэширование данных. Распределенная система, используемая в настоящей работе, не отвечает за основную логику приложения.

Масштабируемость технологии обусловлена тремя наиболее значимыми абстракциями: иерархия групп потоков, разделение потоков и барьерная синхронизация. Программист выделяет в задаче несколько подзадач, каждой из которых в соответствие ставится определенный блок потоков, отвечающий за ее выполнение. Эти потоки работают вместе

благодаря разделению потоков и синхронизации. О масштабируемости говорят в том случае, когда блок потоков последовательно или с использованием распределения может быть выполнен на любом из доступных ядер обрабатывающего устройства, призванного ускорить вычисления.

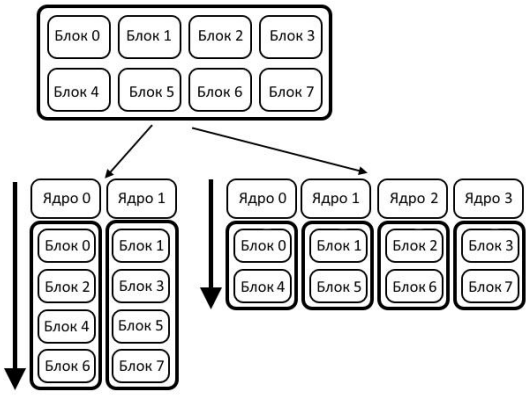

Рис. 2.1. Автоматическое распределение блоков с использованием доступных ядер

На вычислительных системы производства NVIDIA алгоритмы реализуются посредством языка программирования C с использованием ядер – специальных функций, выполняемых многократно на каждом из потоков. Ядра помечаются приставкой __global__. Количество потоков задается специальным символом <<<…>>>. Чтобы получить доступ к какому-либо из потоков, необходимо воспользоваться переменной, которая встроена в спецификацию языка – threadIdx. Данная переменная представляет собой вектор из трех компонентов. В связи с этим индекс может быть одномерным, двумерным, трехмерным. От этого зависит размерность сетки. Данная особенность позволяет наиболее естественным образом использовать массивы, матрицы или трехмерные сетки в качестве объектов.

Поскольку все потоки внутри блока должны относиться к одному ядру распределенной вычислительной системы, количество потоков на блок в

современных видеокартах, как правило, не превышает 1024. Чтобы подсчитать суммарное количество потоков, необходимо умножить количество потоков в одном блоке умножить на количество блоков, с использованием которых выполняется ядро. Оба множители в качестве переменных по очереди задаются в синтаксисе <<<…>>>. Кроме того, переменные могут быть типа int или dim3. Рис. 2.2 демонстрирует, как двумерные блоки задают двумерную сетку потоков, образуя иерархию потоков.

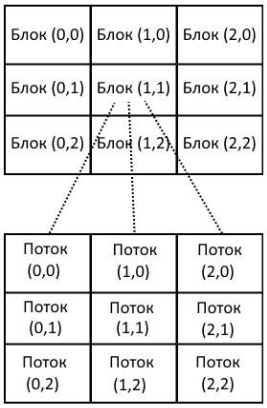

Рис. 2.2. Сетка блоков потоков

Для доступа к блокам, также как и для потоков предусмотрена специальная переменная blockIdx. Размерность блока задается переменной blockDim. Реализация каждого блока может быть выполнена на любом из ядер распределенной вычислительной системы, что в свою очередь, обеспечивает масштабируемость программного кода.

Барьерная синхронизация и разделение потоков могут быть использованы только для потоков внутри блока. Для синхронизации используется специальная функция языка – __syncthreads(). Эта функция

выделяет особую точку синхронизации. При этом отдельно взятый поток не выполняется, пока остальные потоки не дойдут до заданной точки.

2.2.1. Организация основного цикла решения системы с использованием распределенных вычислений

В настоящей работе предлагается несколько способов повышения производительности метода дискретных элементов. Система, физические параметры которой определены в п. 2.2 настоящей работы, находится в некотором начальном состоянии. Распределенная вычислительная система позволяет осуществлять цикл системы, при котором происходит ее переход к новому состоянию. Учитывая предложенную в соответствии с аппаратом распределенной вычислительной системы концепцию функций-ядер, целесообразно разделить цикл системы на отдельные части таким образом, чтобы обеспечить их раздельное выполнение.

К наиболее общим частям цикла относятся интегрирование системы и просчет сил взаимодействия.

При переходе системы к новому состоянию скорости частиц меняются в зависимости от воздействующих на них сил. При этом положение частиц также пересчитывается. В новом состоянии высчитываются силы, воздействующие на частицы, а также определяются пары частиц, которые взаимодействую между собой в соответствии с формулами, описанными в п. 2.1. Вновь полученные силы применяются для интегрирования системы на новом шаге. На рис. 2.3 графически изображен цикл системы [66,70,71].

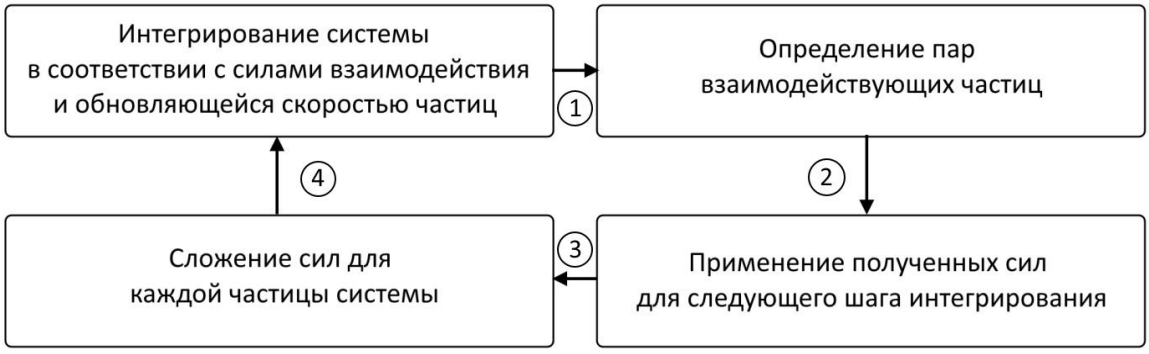

Рис. 2.3. Основной цикл работы системы

Код приложения, реализующего данный алгоритм, содержит два ядра, выполняемые в соответствии с принципами распределенной вычислительной системы. В одном ядре позиция частицы обновляется старым значением скорости, после чего обновляется скорость частицы. Другое ядро отвечает за просчет сил взаимодействия текущей частицы со всеми, с которыми она сталкивается. Ядро выполняет сложение вновь полученных сил. Данные ядра впоследствии вызываются из главной программы.

При поиске пар взаимодействующих частиц наиболее примитивным методом будет перебор всех пар частиц. При этом каждый раз проверяется степень проникновения между частицами ξ, и в случае ее положительно значения, рассчитываются силы взаимодействия. Порядок сложности такого алгоритма составляет $O(n^2)$, что не приемлемо для системы, количество частиц которое составляет несколько сотен тысяч.

Очевидно, что большая часть частиц находится сравнительно далеко от отдельно взятой частицы. Поэтому целесообразно производить расчет сил взаимодействия только с теми частицами, которые находятся рядом. Чтобы исключить частицы, находящиеся далеко от заданной, пространство представляется в виде сетки с ячейками одинакового размера. Каждая такая ячейка содержит объекты. Такое представление пространства позволяет легко находить положение заданных частиц и соседних с ними.

2.2.2. Модель оптимизации управления информационным потоком

Сетка имеет h ячеек по каждому из трех измерений. Таким образом, задать ячейку можно, указав ее координаты (i, j, k). Ячейка представляется в форме куба, сторона которого равна l. Число l выбирается так, чтобы быть равным диаметру самой большой частицы системы. Если (x, y, z) – координаты произвольной частицы, то координаты ячейки, к которой она относится, записываются как $(\left|\frac{x}{l}\right|, \left|\frac{y}{l}\right|, \left|\frac{z}{l}\right|)$.

Чтобы отыскать пары взаимодействия одной из частиц заданной ячейки с другими, необходимо рассматривать только 26 соседних ячеек.

Каждую из ячеек удобно рассматривать в качестве линейного списка объектов, находящихся в ней. Элементы списка содержат два поля: данные, а также указатель на следующий элемент. Расположение этих элементов в сетке входящих информационных потоков способствует нерациональному использованию ресурсов распределенной вычислительной системы. Поэтому при реализации существует потребность в оптимизации управления информационным потоком.

Разделение потоков предполагает возможность быстрого доступа для всех потоков в пределах блока. Использование разделенного доступа к потокам способствует минимизации использования ресурсов распределенной вычислительной системы и ускорению работы алгоритма (в пределах 15%).

В случае линейного доступа к данным запросы могут группироваться в одну транзакцию. Однако при этом ограничиваются размером сегмента, который в свою очередь, определяется запрашиваемыми данными.

Если потоки оперируют расположенными друг за другом данными, то данные выбираются посредством единой транзакции. Случайное расположение данных в сетке информационного потока увеличивает количество подобных транзакций, что замедляет работу алгоритма.

Чтобы учесть описанные выше тонкости работы с потоками, в настоящей работе предлагается для каждой ячейки задать ее порядковый номер в пространственной сетке. Если ячейка задается координатами (i, j, k), то номер ячейки можно определить по формуле:

$$n = kh^2 + jh + i, \tag{2.19}$$

На смену списку приходит массив из ключей и значений. В качестве значения выступает номер частицы, тогда в качестве ключа – номер ячейки, содержащей данную частицу. Благодаря сортировке такого массива по ключу частицы в каждой из ячеек обретают естественный порядок. Пример такой сортировки приводится в таблице 2.1:

Таблица 2.1. Сортировка массива частиц по номеру ячейки

Исходный массив		
№	Ячейка	Частица
0	2	0
1	0	1
2	1	2
3	0	3
4	1	4
…		
N	6	N
Отсортированный массив		
№	Ячейка	Частица
0	0	1
1	0	3
2	1	2
3	1	4
4	2	0
…		
N	6	N

В результате сортировки для каждой ячейки получаются два числа – индексы отсортированного массива, указывающие на начало и конец ключей с номером ячейки. Результат с полученными индексами удобно переписать в виде таблицы 2.2:

Таблица 2.2 Определение начала и конца ключей с номером ячейки

Ячейка	Индекс начала ключа	Индекс конца ключа
0	0	1
1	2	3
2	4	4
3	-	-
4	-	-
5	-	-
6	N	N

Из табл. 2.2. видно, что ячейки без частиц помечаются символом «-».

Подход, реализованный в данной работе, предоставляет ряд преимуществ:

- оптимизация управления информационным потоком благодаря линейному расположению данных;
- работа со списком существенно упрощена, поскольку сам список является постоянным;
- возможность применить к предложенной сортировке мощности распределенной вычислительной системы.

2.2.3. Алгоритм решения системы

С учетом предложенной оптимизации управления информационным потоком, алгоритм основного цикла приложения можно переписать в следующем виде:

1. Выполнение шага системы при текущих значениях сил;

2. Сортировка массива, при которой в качестве ключа выступает номер ячейки;
3. Определение начала и конца ячейки в списке;
4. Расчет сил взаимодействия для каждой частицы системы;
5. Переход к шагу 1.

Алгоритм реализации модели примет вид:

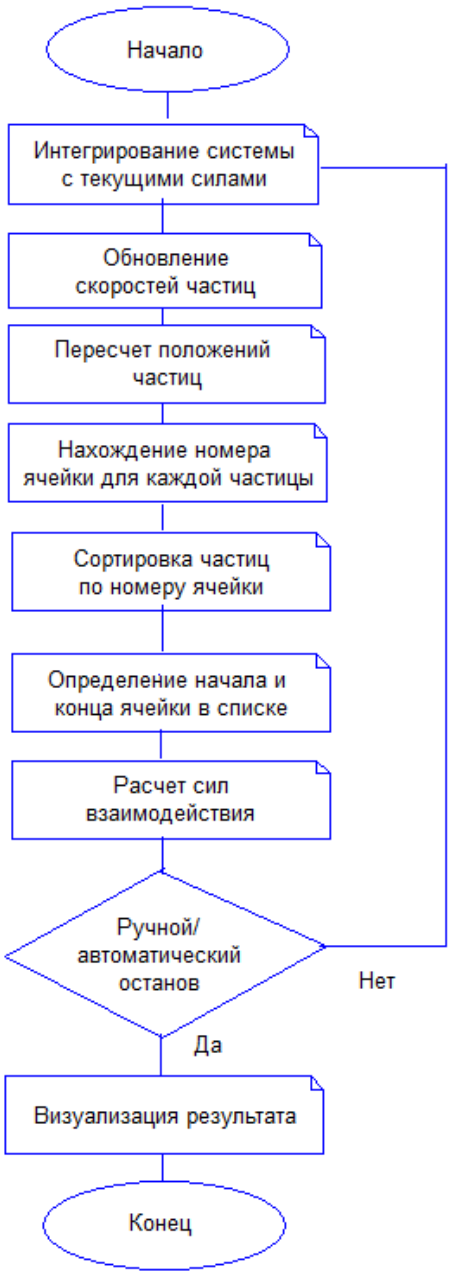

Рис. 2.4. Блок-схема алгоритма

Результат работы алгоритма описывается в п. 2.3 настоящей работы.

2.3. Вычислительный эксперимент

Технология CUDA впервые стала использоваться на чипе NVIDIA восьмого поколения G80 и впоследствии поддерживалась во всех новых сериях. Перечень устройств с заявленной полной поддержкой технологии CUDA приведён на официальном сайте NVIDIA [51]. Наиболее поздние модели поддерживают технологию CUDA версии 5.0. К популярным семействам графических ускорителей относятся GeForce, Quadro и NVidia Tesla. В настоящей работе используется GeForce GTX 690. Эта модель относится к ряду производительных и недорогих.

Чтобы работать с программной реализацией, необходимо установить CUDA Toolkit – специальная среда разработки для графических процессоров с поддержкой CUDA, основанная на языке программирования C. CUDA Toolkit содержит в себе собственный компилятор nvcc, вспомогательные библиотеки, а также большое количество справочной информации [69].

Перед началом работы следует установить CUDA Developper SDK. Он включает в себя исходный код, специальные сервисные программы, а также множество примеров для различных приложений.

CUDA Toolkit, интегрируясь в Microsoft Visual Studio версии 13, позволяет выводит на экран трехмерную графику. Это становится осуществимым благодаря технологии DirectX 11– специального набора API. Для разработки программного обеспечения на основе DirectX используется специальный набор утилит – Direct SDK.

Для того чтобы созданное приложение функционировало на компьютере, кроме видеокарты с поддержкой CUDA и DirectX необходима операционная система не ниже версии Windows XP.

Что касается пользовательского интерфейса приложения, то он состоит из окна консоли и главного графического окна. Графика выводится на экран в следующем виде:

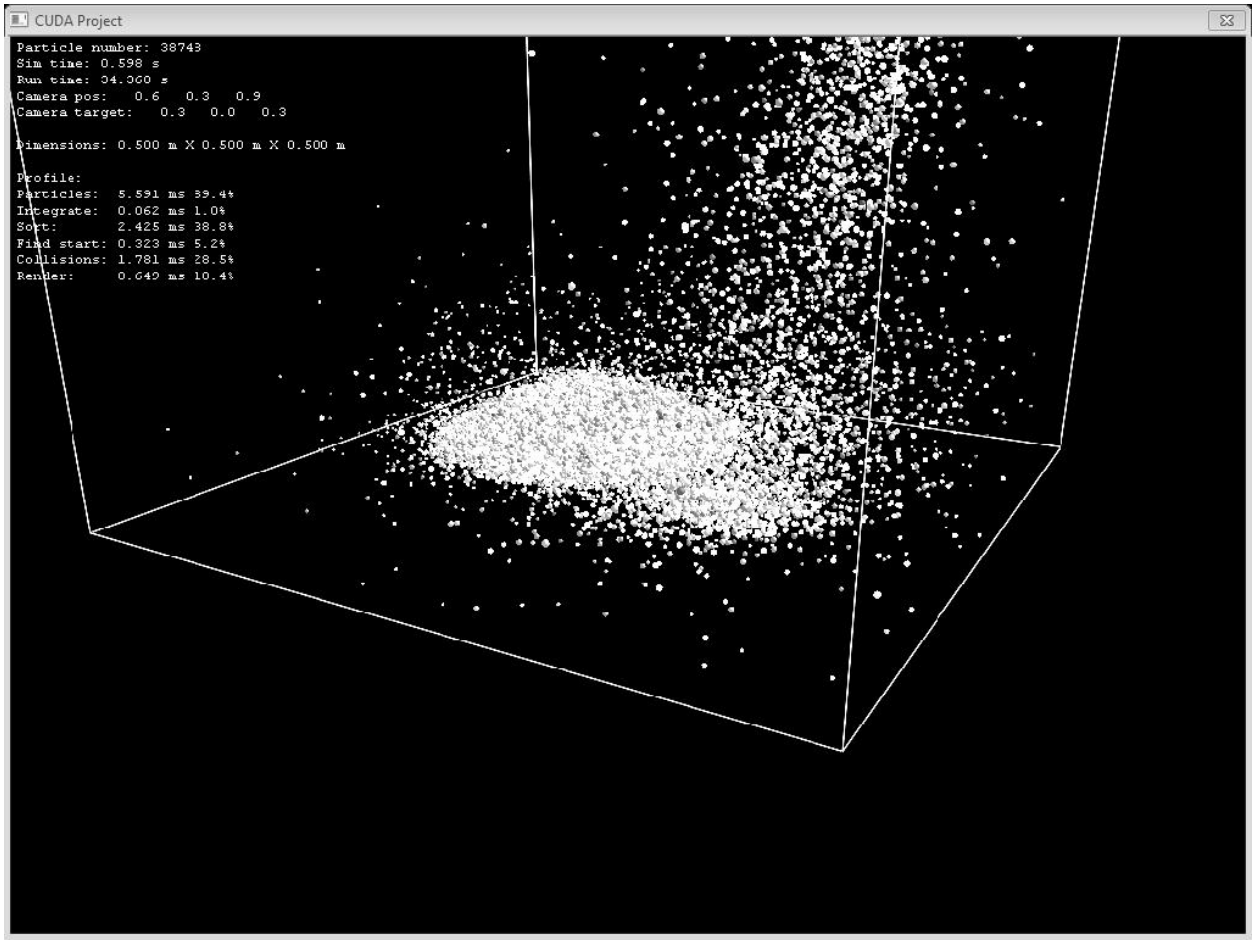

Рис. 2.5. Графическое окно приложения

На рис. 2.5 видно ограниченное пространство с частицами внутри него, которое реализуется посредством виртуальной камеры. Для того чтобы вращать камеру, предусматривается специальная точка осмотра, именуемая целью. Посредством цели можно задавать удаленность от камеры.

На рис. 2.6 изображено окно консоли, с помощью которого на экран выводятся детали визуального моделирования динамики частиц сыпучего материала:

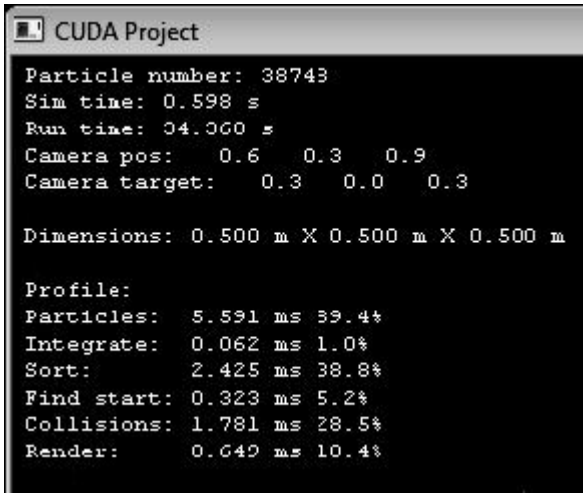

Рис. 2.6. Информационное окно консоли

В окне отображаются:

- общее количество частиц настоящего эксперимента;
- время, затраченное на визуальное моделирование;
- время, прошедшее с момента запуска приложения;
- положение камеры;
- положение цели;
- пространство эксперимента.

В разделе профиль отслеживается отображаемое в миллисекундах время, затрачиваемое на ту или иную часть цикла. Кроме того, дополнительно указывается значение времени в процентах от времени всего эксперимента. В данном разделе обозначены шесть частей цикла:

1. Обработка всех частиц;
2. Интегрирование системы;
3. Сортировка;
4. Определение начала и конца ячеек в списке;
5. Проверка на столкновения;
6. Визуализация.

2.4. Выводы

1. Разработан алгоритм поведения многоточечных масс сыпучей среды с помощью графического процессора в трехмерном пространстве для получения ее геометрии.
2. Для ускорения процесса вычисления предложены способы минимизации использования ресурсов распределенной вычислительной системы, использующие особенности расположения данных, что позволило организовать многопоточные вычисления.
3. Создано программное обеспечение, моделирующее динамику частиц многоточечных масс, позволяющее проводить вычислительный эксперимент.

Глава 3. Модели и методы прогнозирования

3.1. Метод прогнозирования, основанный на нечеткой логике

В рассматриваемой работе ставится задача прогнозирования изменения коэффициента устойчивости грунтового массива [67]. Для решения этой задачи используются методы нечеткой логики. Дана следующая постановка задачи. Известны значения коэффициента устойчивости за фиксированный период времени. Задача состоит в определении дальнейшего изменения коэффициента устойчивости с учетом исходных данных.

В соответствии с постановкой задачи предлагается следующая методика прогнозирования.

1. Структурирование исходной информации.

2. Для словесного (качественного) описания в понятиях человека значений коэффициента устойчивости – лингвистической переменной – определение соответствующих лингвистических значений этой переменной.

3. Описание экспертно-лингвистических закономерностей, наблюдаемых на графике изменения коэффициента устойчивости.

4. Выявление функциональных связей между значениями коэффициента устойчивости.

5. Построение сети зависимостей для прогнозирования.

5. Формализация экспертно-лингвистических закономерностей при помощи функций принадлежности.

6. Дефаззификация полученного результата для преобразования функций принадлежности к четким значениям.

В таблице 3.1 представлено изменение значений коэффициента устойчивости за последние 20 единиц времени.

Таблица 3.1.Данные об изменении коэффициента устойчивости

Время	Коэффициент устойчивости	Тенденция (изменение)
t_1	1,0104	
t_2	1,0136	+0,0032
t_3	1,0062	-0,0074
t_4	1,008	+0,0018
t_5	1,0077	-0,0003
t_6	1,0029	-0,0048
t_7	0,9898	-0,0131
t_8	0,9893	-0,0005
t_9	0,9964	+0,0071
t_{10}	0,9962	-0,0002
t_{11}	1,0029	+0,0067
t_{12}	1,0014	-0,0015
t_{13}	1,007	+0,0056
t_{14}	1,004	-0,003
t_{15}	1,0031	-0,0009
t_{16}	1,0029	-0,0002
t_{17}	0,9983	-0,0046
t_{18}	0,9915	-0,0068
t_{19}	0,9958	+0,0043
t_{20}	1,0059	+0,0101

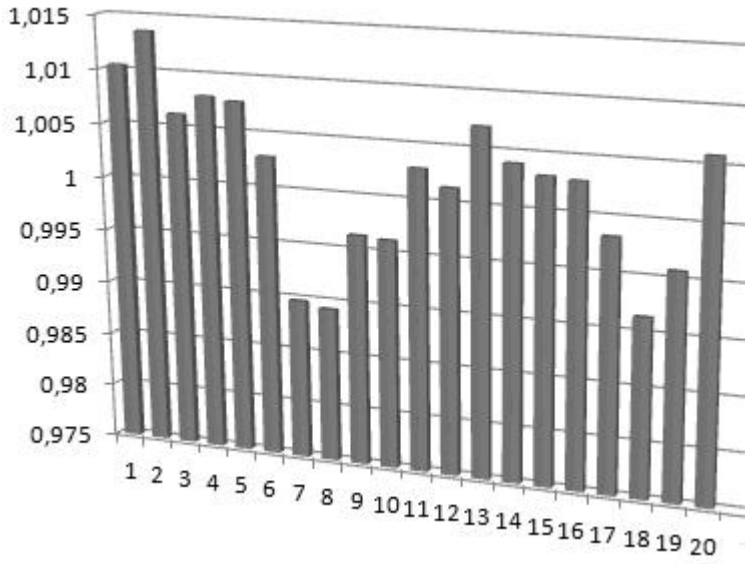

Рис. 3.1. Диаграмма, отражающая динамику изменения коэффициента устойчивости с течением времени

При разработке модели прогнозирования изменения коэффициента устойчивости, основанной на методах нечеткой логики, определяется терм-множество лингвистической переменной «значение коэффициента устойчивости». При построении графика на множестве из 20 значений коэффициента устойчивости было выделено 5 термов (рисунок 3.4). Термы определены на естественном языке – «ВЫСОКОЕ», «ВЫШЕ СРЕДНЕГО», «СРЕДНЕЕ», «НИЖЕ СРЕДНЕГО», «НИЗКОЕ». Если увеличить количество термов, то прогноз будет точнее, но при большом количестве экспериментальных данных это отрицательно скажется на длительности их обработки.

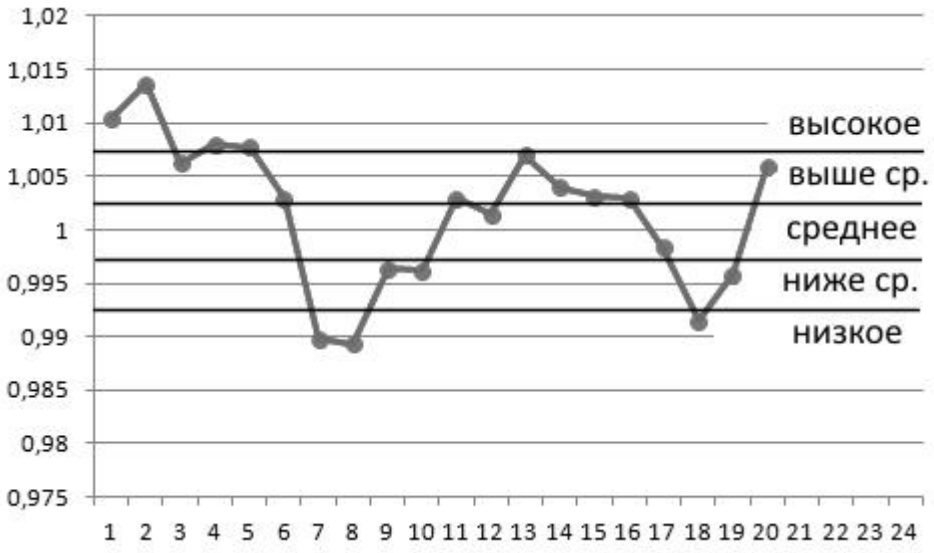

Рис. 3.2. Определение множества термов для выходной лингвистической переменной «значение коэффициента устойчивости»

На рис. 3.2 можно заметить некоторые экспертно – лингвистические закономерности, которые участвуют в построении модели прогнозирования. Следуя по точкам графика, видно, что удержание коэффициента устойчивости на одном уровне составляет 3 единицы времени, его уменьшение – 3 единицы времени, а увеличение более продолжительно – 5 единиц времени. В совокупности это 11 точек.

Описанная циклическая конструкция в виде точечного шаблона выглядит следующим образом.

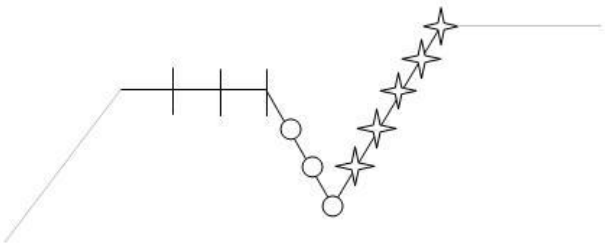

Рис. 3.3. Точечный шаблон циклической конструкции

Существующие согласно такому точечному шаблону циклические конструкции обозначим следующим образом:

$$\cdots x_{10}^{i-1}, x_{11}^{i-1}\}\{x_1^i, x_2^i, x_3^i, x_4^i, x_5^i, x_6^i, x_7^i, x_8^i, x_9^i, x_{10}^i, x_{11}^i\}\{x_1^{i+1}, x_2^{i+1}\cdots$$

где i – номер одиннадцатилетнего цикла, $x_1^i ... x_{11}^i$ – значение коэффициента устойчивости в соответствующий момент цикла. График, изображенный на рис. 3.3, подтверждает наличие цикличности.

В соответствии с закономерностью, наблюдаемой на рис. 3.2, и точечным шаблоном циклической конструкции (рис. 3.3) составляется база знаний, представленная в виде одиннадцати экспертных высказываний на естественном языке. Высказывания представлены правилами «ЕСЛИ – ТО» и связывают значение коэффициента устойчивости в i-м и $(i+1)$-м циклах (рис. 3.4) [67].

Правило F_1:	Правило F_2:
ЕСЛИ X_{10}^{i-1} = ВЫСОКОЕ	ЕСЛИ X_{10}^{i-1} = ВЫСОКОЕ
И X_{11}^{i-1} = ВЫСОКОЕ,	И X_{11}^{i-1} = ВЫСОКОЕ,
ТО X_1^i = ВЫШЕ СРЕДНЕГО;	ТО X_2^i = ВЫСОКОЕ;
ЕСЛИ X_{10}^{i-1} = СРЕДНЕЕ	ЕСЛИ X_{10}^{i-1} = СРЕДНЕЕ
И X_{11}^{i-1} = ВЫШЕ СРЕДНЕГО,	И X_{11}^{i-1} = ВЫШЕ СРЕДНЕГО,
ТО X_1^i = ВЫШЕ СРЕДНЕГО;	ТО X_2^i = ВЫШЕ СРЕДНЕГО;
Правило F_3:	Правило F_4:
ЕСЛИ X_2^i = ВЫСОКОЕ,	ЕСЛИ X_2^i = ВЫСОКОЕ
ТО X_3^i = ВЫСОКОЕ;	И X_3^i = ВЫСОКОЕ,
ЕСЛИ X_2^i = ВЫШЕ СРЕДНЕГО,	ТО X_4^i = ВЫШЕ СРЕДНЕГО;
ТО X_3^i = ВЫШЕ СРЕДНЕГО;	ЕСЛИ X_2^i = ВЫШЕ СРЕДНЕГО

Правило F_5:	И X_3^i = ВЫШЕ СРЕДНЕГО,
ЕСЛИ X_3^i = ВЫСОКОЕ	ТО X_4^i = СРЕДНЕЕ;
И X_4^i = ВЫШЕ СРЕДНЕГО,	Правило F_6:
ТО X_5^i = НИЗКОЕ;	ЕСЛИ X_5^i = НИЗКОЕ
ЕСЛИ X_3^i = ВЫШЕ СРЕДНЕГО	ТО X_6^i = НИЗКОЕ;
И X_4^i = СРЕДНЕЕ,	ЕСЛИ X_5^i = НИЗКОЕ
ТО X_5^i = НИЗКОЕ;	ТО X_6^i = НИЖЕ СРЕДНЕГО;
Правило F_7:	Правило F_8:
ЕСЛИ X_5^i = НИЗКОЕ	ЕСЛИ X_5^i = НИЗКОЕ
И X_6^i = НИЗКОЕ,	И X_6^i = НИЗКОЕ,
ТО X_7^i = НИЖЕ СРЕДНЕГО;	ТО X_8^i = НИЖЕ СРЕДНЕГО;
ЕСЛИ X_5^i = НИЗКОЕ	Правило F_9:
И X_6^i = НИЖЕ СРЕДНЕГО,	ЕСЛИ X_7^i = НИЖЕ СРЕДНЕГО
ТО X_7^i = ВЫШЕ СРЕДНЕГО;	И X_8^i = НИЖЕ СРЕДНЕГО,
Правило F_{11}:	ТО X_9^i = ВЫШЕ СРЕДНЕГО;
ЕСЛИ X_9^i = ВЫШЕ СРЕДНЕГО	Правило F_{10}:
И X_{10}^i = СРЕДНЕЕ,	ЕСЛИ X_9^i = ВЫШЕ СРЕДНЕГО
ТО X_{11}^i = ВЫШЕ СРЕДНЕГО;	ТО X_{10}^i = СРЕДНЕЕ;

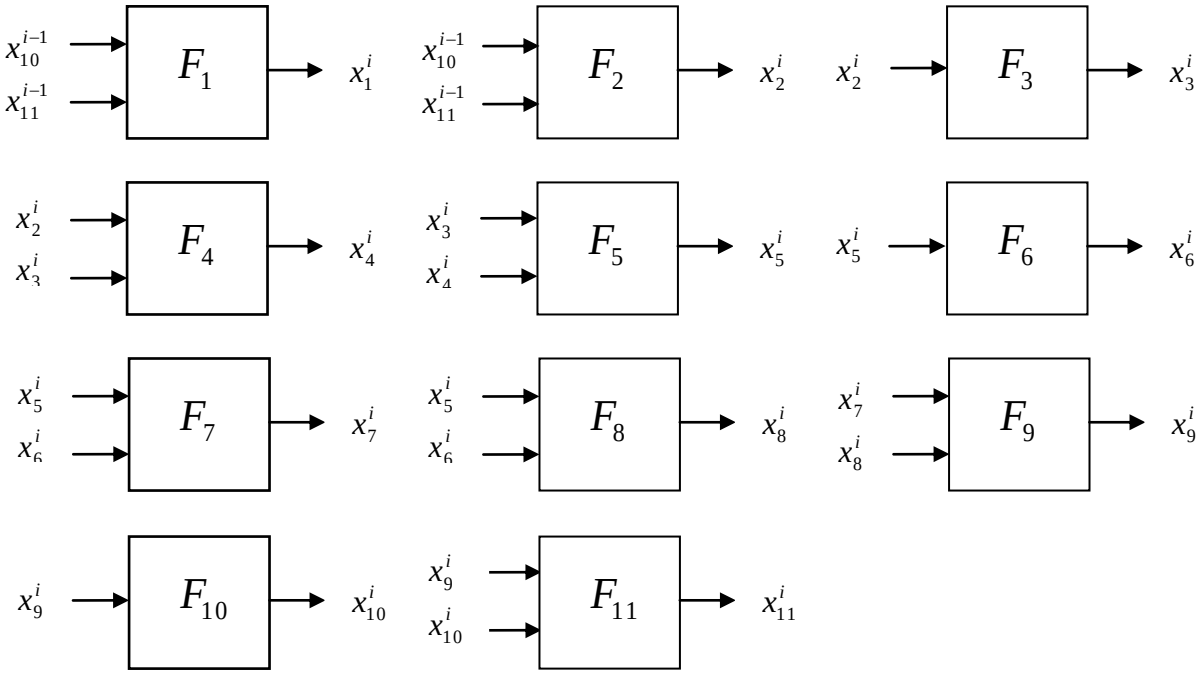

Рис. 3.4. Функциональные связи между значениями коэффициента устойчивости

Рис. 3.5 изображает сеть зависимостей, составленную по правилам $F_1 - F_{11}$. Из нее видно, что по двум последним значениям $(i-1)$-го цикла можно спрогнозировать все значения i-го цикла.

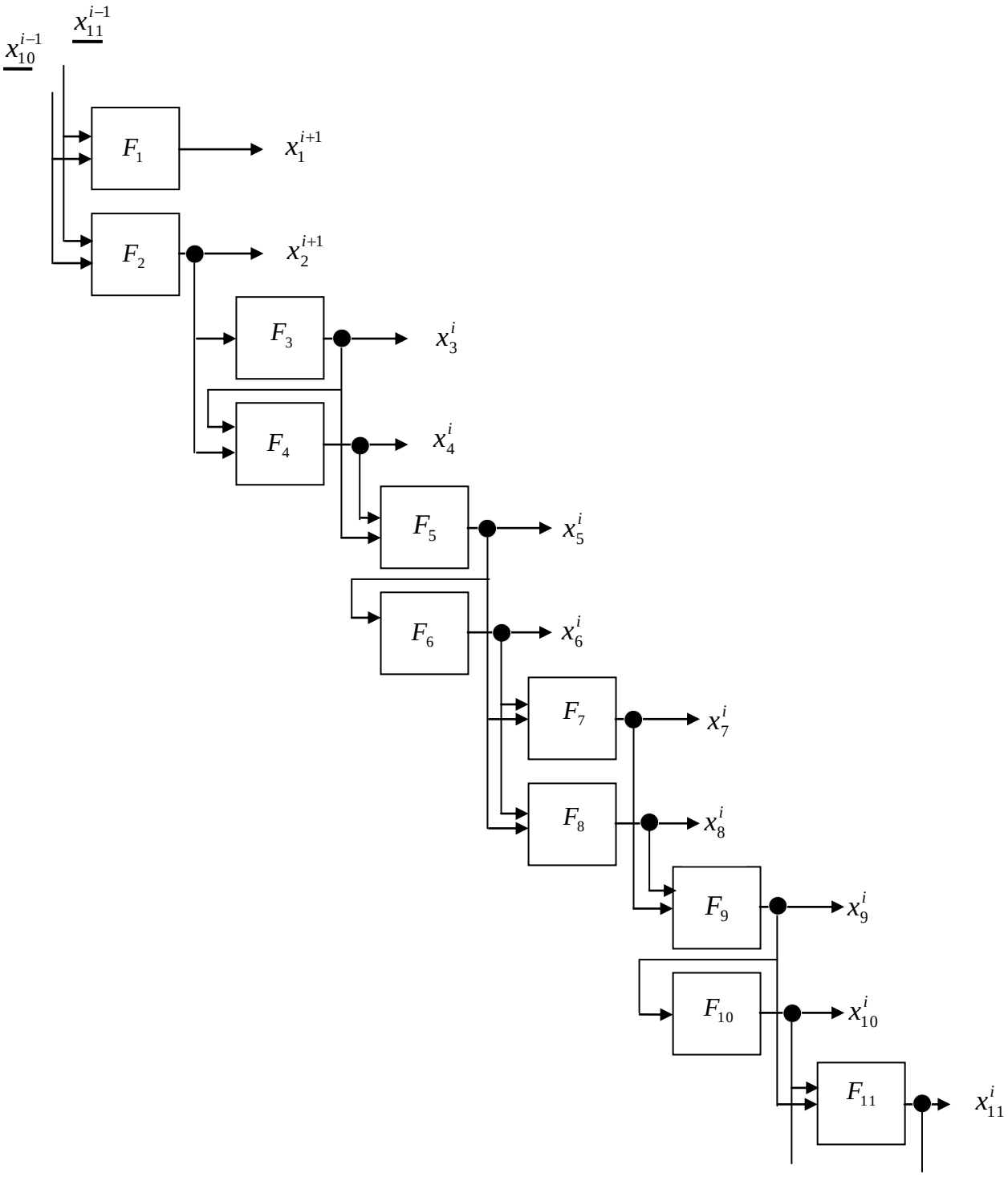

Рис. 3.5. Сеть зависимостей для прогнозирования

Выписанные правила позволяют прогнозировать изменение значения коэффициента. Ошибка прогнозирования с каждой итерацией будет расти, но это можно исправить, обладая большими объемами экспериментальных данных, накопленных с течением процесса.

Для правил $F_1 - F_{11}$ применяется аппарат теории нечетких множеств. Для того чтобы формализовать оценки «ВЫСОКОЕ» (В), «ВЫШЕ СРЕДНЕГО» (ВС), «СРЕДНЕЕ» (С), «НИЖЕ СРЕДНЕГО» (НС) и «НИЗКОЕ» (Н), в решаемой задаче используется следующая аналитическая модель функции принадлежности переменной x («значение коэффициента устойчивости») произвольному нечеткому терму T:

$$\mu^T(x) = \frac{1}{1 + (\frac{x-b}{c})^2}, \tag{3.1}$$

где b и c – параметры настройки. Параметр b – координата максимума функции ($\mu^T(b) = 1$). Параметр c – коэффициент концентрации – растяжения функции (рис. 3.6). Число b представляет собой наиболее возможное значение переменной x для нечеткого терма T.

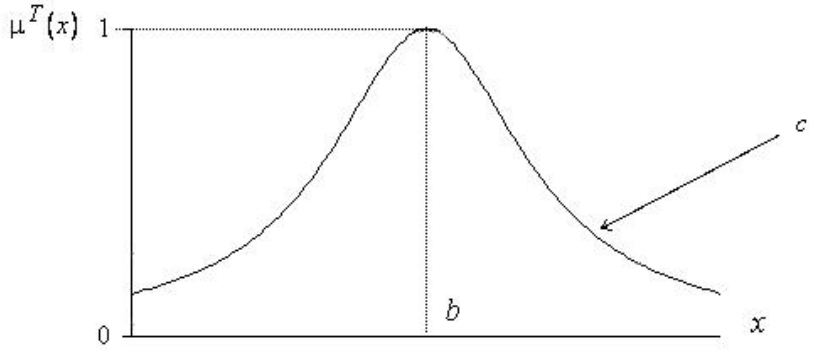

Рис. 3.6. Модель функции принадлежности

Параметры b и c для разных лингвистических оценок, используемых в правилах $F_1 - F_{11}$, выбираются экспертом и представлены в следующей таблице.

Таблица 3.2. Параметры функции принадлежности до настройки

Лингвистические оценки переменных	параметр	параметр
	b	c
ВЫСОКОЕ	1,015	0,018
ВЫШЕ СРЕДНЕГО	1,005	0,015
СРЕДНЕЕ	1	0,013
НИЖЕ СРЕДНЕГО	0,995	0,015
НИЗКОЕ	0,985	0,021

Графики полученных при этом функций принадлежности, выглядят следующим образом (рис. 3.7).

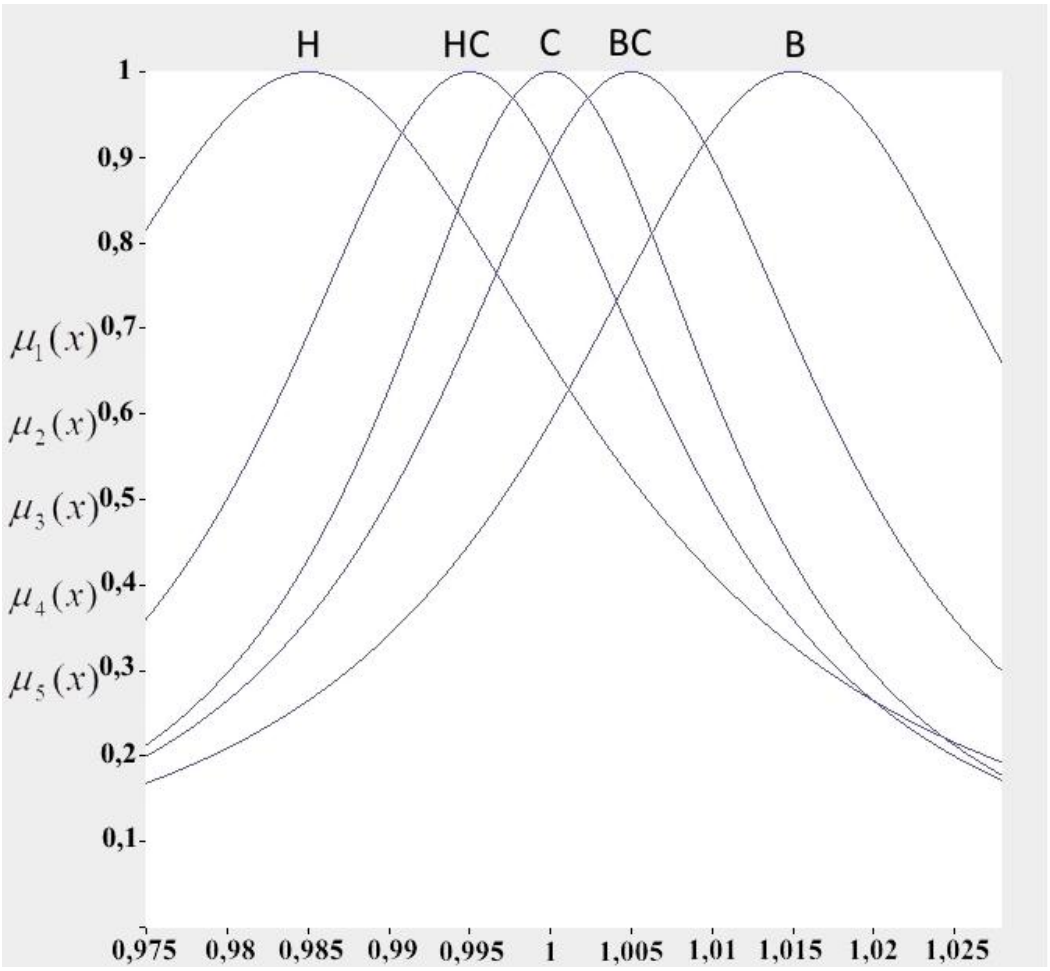

Рис. 3.7. Функции принадлежности лингвистических оценок до настройки

Чтобы преобразовать полученные функции принадлежности к четким значениям, применяется формула дефаззификации, которая в данной задаче имеет вид:

$$x_i = \frac{\sum_{j=1}^{K_A} k_j \mu^A(x_i)}{\sum_{j=1}^{K_A} \mu^A(x_i)}, \qquad (3.2)$$

где x_i – четкое значение коэффициента устойчивости, μ^A – функция принадлежности четкого значения к рассматриваемому терму, K_A – количество составленных функций принадлежности для правила F_i.

Коэффициенты k_j подбираются следующим образом. Диапазон возможных значений коэффициента устойчивости разбивается на пять частей в соответствии с введенными в настоящей задаче термами Н, НС, С, ВС, В. Тогда коэффициенты k_j примут значения коэффициентов устойчивости на границах термов (рис. 3.8). Множество значений коэффициента устойчивости, заключенных в отрезке $[k_1, k_6]$, будет базовым множеством.

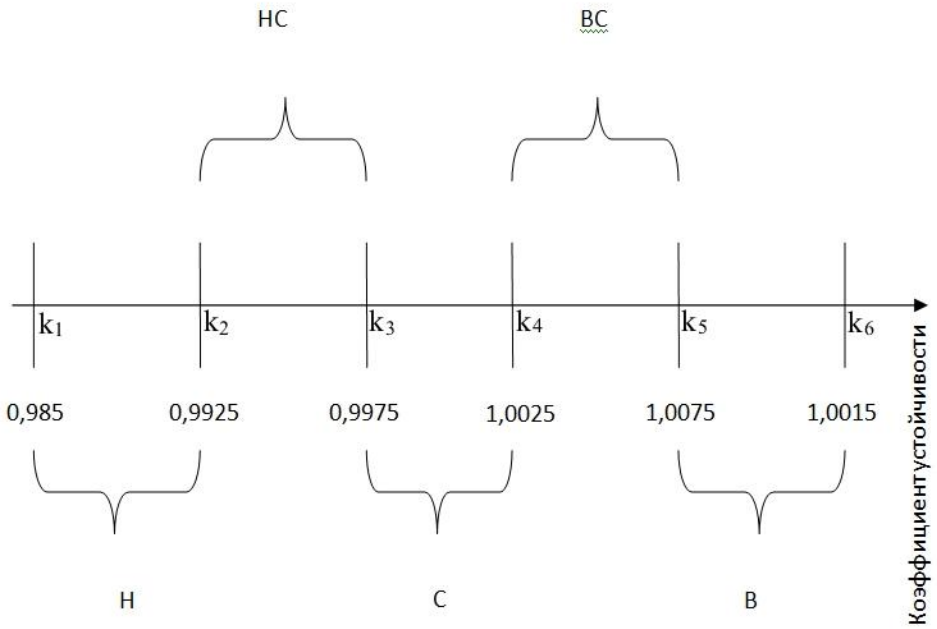

Рис. 3.8. Диапазон возможных значений коэффициента устойчивости

С помощью нечетко-логических операций «И» (min), «ИЛИ» (max) и операции дефаззификации, можно в явном виде записать модель прогнозирования записывается в явном виде следующим образом:

$$F_1 : \begin{cases} \mu^{BC}(x_1^i) = \max\begin{pmatrix} \min(\mu^B(x_{10}^{i-1}), \mu^B(x_{11}^{i-1})) \\ \min(\mu^C(x_{10}^{i-1}), \mu^{BC}(x_{11}^{i-1})) \end{pmatrix}, \\ x_1^i = \dfrac{k_4 \cdot \mu^{BC}(x_1^i)}{\mu^{BC}(x_1^i)}. \end{cases}$$

$$F_2:\begin{cases}\mu^B(x_2^i)=\min(\mu^B(x_{10}^{i-1}),\mu^B(x_{11}^{i-1})),\\ \mu^{BC}(x_2^i)=\min(\mu^C(x_{10}^{i-1}),\mu^{BC}(x_{11}^{i-1})),\\ x_2^i=\dfrac{k_5\cdot\mu^B(x_2^i)+k_4\cdot\mu^{BC}(x_2^i)}{\mu^B(x_2^i)+\mu^{BC}(x_2^i)}.\end{cases}$$

$$F_3:\begin{cases}\mu^B(x_3^i)=\mu^B(x_2^i),\\ \mu^{BC}(x_3^i)=\mu^{BC}(x_2^i),\\ x_3^i=\dfrac{k_5\cdot\mu^B(x_3^i)+k_4\cdot\mu^{BC}(x_3^i)}{\mu^B(x_3^i)+\mu^{BC}(x_3^i)}.\end{cases}$$

$$F_4:\begin{cases}\mu^{BC}(x_4^i)=\min(\mu^B(x_2^i),\mu^B(x_3^i)),\\ \mu^C(x_4^i)=\min(\mu^{BC}(x_2^i),\mu^{BC}(x_3^i)),\\ x_4^i=\dfrac{k_4\cdot\mu^{BC}(x_4^i)+k_3\cdot\mu^C(x_4^i)}{\mu^{BC}(x_4^i)+\mu^C(x_4^i)}.\end{cases}$$

(3.3)

$$F_5:\begin{cases}\mu^H(x_5^i)=\max\begin{pmatrix}\min(\mu^B(x_3^i),\mu^{BC}(x_4^i))\\ \min(\mu^{BC}(x_3^i),\mu^C(x_4^i))\end{pmatrix},\\ x_5^i=\dfrac{k_1\cdot\mu^H(x_5^i)}{\mu^H(x_5^i)}.\end{cases}$$

$$F_6:\begin{cases}\mu^H(x_6^i)=\mu^H(x_5^i),\\ \mu^{HC}(x_6^i)=\mu^H(x_5^i),\\ x_6^i=\dfrac{k_1\cdot\mu^H(x_6^i)+k_2\cdot\mu^{HC}(x_6^i)}{\mu^H(x_6^i)+\mu^{HC}(x_6^i)}.\end{cases}$$

$$F_7:\begin{cases}\mu^{HC}(x_7^i)=\min(\mu^H(x_5^i),\mu^H(x_6^i)),\\ \mu^{BC}(x_7^i)=\min(\mu^H(x_5^i),\mu^{HC}(x_6^i)),\\ x_7^i=\dfrac{k_2\cdot\mu^{HC}(x_7^i)+k_4\cdot\mu^{BC}(x_7^i)}{\mu^{HC}(x_7^i)+\mu^{BC}(x_7^i)}.\end{cases}$$

$$F_8:\begin{cases}\mu^{HC}(x_8^i)=\min(\mu^H(x_5^i),\mu^H(x_6^i)),\\ x_8^i=\dfrac{k_2\cdot\mu^{HC}(x_8^i)}{\mu^{HC}(x_8^i)}.\end{cases}$$

$$F_9 : \begin{cases} \mu^{BC}(x_9^i) = \min(\mu^{HC}(x_7^i), \mu^{HC}(x_8^i)), \\ x_9^i = \dfrac{k_4 \cdot \mu^{BC}(x_9^i)}{\mu^{BC}(x_9^i)}. \end{cases}$$

$$F_{10} : \begin{cases} \mu^{C}(x_{10}^i) = \mu^{BC}(x_9^i), \\ x_{10}^i = \dfrac{k_3 \cdot \mu^{C}(x_{10}^i)}{\mu^{C}(x_{10}^i)}. \end{cases}$$

$$F_{11} : \begin{cases} \mu^{BC}(x_{11}^i) = \min(\mu^{BC}(x_9^i), \mu^{C}(x_{10}^i)), \\ x_{11}^i = \dfrac{k_4 \cdot \mu^{BC}(x_{11}^i)}{\mu^{BC}(x_{11}^i)}. \end{cases}$$

Результаты, полученные при помощи выбранной модели, представлены на рис. 3.9.

Рис. 3.9. Результаты прогнозирования

На данном рисунке график с закрашенными точками отражает фактическое значение коэффициента устойчивости, а график с проколотыми точками – значение коэффициента устойчивости, полученное в результате прогнозирования. На графике заметно отклонение результатов

прогнозирования от исходных данных. Приведенная погрешность вычисляется по формуле:

$$E_i = \frac{\left| N_{фактическое} - N_{прогнозируемое} \right|}{K_{\max} - K_{\min}} \cdot 100\%, \qquad (3.4)$$

где E_i - приведенная погрешность прогнозирования коэффициента устойчивости в i – й момент времени. В числителе стоит соответствующая абсолютная погрешность, а в знаменателе – доверительный интервал, который зависит от типа шкалы и равен ширине диапазона измерений прогнозируемой величины. Средняя приведенная погрешность берется как среднее арифметическое приведенных погрешностей за i единиц времени, и вычисляется по формуле:

$$E_{ср} = \frac{\sum_{i=P_1}^{Pn} E_i}{P}, \qquad (3.5)$$

где P - количество прогнозируемых значений, P_1 - момент начала прогноза, а P_n - последний прогнозируемый момент.

Приведенная погрешность прогнозирования при выбранных параметрах составляет 10,5%.

При помощи полученной модели можно грубо прогнозировать изменение значения коэффициента устойчивости грунтового массива. Для повышения точности прогноза необходимо осуществить настройку модели.

3.2. Настройка модели прогнозирования с помощью модифицированного генетического алгоритма

Настройка модели прогнозирования сводится к подбору таких параметров b и c функций принадлежности лингвистических оценок (рис. 3.9), которые сведут к минимуму расхождения между исходными (экспериментальными) значениями коэффициентов устойчивости и значениями, полученными теоретически в результате прогноза. Для этого выбран метод наименьших квадратов [72,73,74]. В соответствии с этим

методом задача настройки прогнозной модели означает найти такие параметры b и c, что

$$\sum_{i=1}^{N1}(x_1^i - \hat{x}_1^i)^2 + \sum_{i=1}^{N1}(x_1^i - \hat{x}_1^i)^2 + \cdots + \sum_{i=1}^{N1}(x_{11}^i - \hat{x}_{11}^i)^2 = \min_{b,c}, \qquad (3.6)$$

где $x_1^i, x_2^i, \cdots x_{11}^i$ – значения коэффициентов устойчивости, полученные в результате прогнозирования и зависящие от экспертных параметров b и c.

$\hat{x}_1^i, \hat{x}_2^i, \cdots \hat{x}_{11}^i$ – исходные (экспериментальные) значения коэффициентов устойчивости,

$N1$ – количество циклов, которые используются для настройки модели.

На основе анализа предметной области для реализации настройки прогноза выбрана эволюционная модель, суть которой состоит в следующем. Популяция представляется в виде двумерного массива $M \times N2$, где M – количество особей, $N2$ – количество генов. В данной задаче рассматривается популяция из 20 особей (хромосом), каждая из которых состоит из 10 генов и характеризуют отклонения параметров b и c от изначально принятых ($M = 20, N2 = 10$). В классических реализациях настройки [65] формирование начальной популяции осуществляется случайным образом. В отличие от этого, в данной работе используется модифицированное формирование начальной популяции по образу мутаций, когда в популяцию помещаются особи с отклонениями параметров b и c, значения которых не больше заданного. Это позволяет осуществлять наиболее глубокий поиск оптимальных параметров в пределах экспериментально полученных значений.

Генетический алгоритм, адекватный логике решаемой задачи исследования насыпи на устойчивость, состоит из следующих шагов:

1. формирование начальной популяции, особи которой хранят в генах отклонения параметров прогнозной модели b и c от заданных экспертами значений;

2. селекция, в результате которой из популяции исключаются особи с набором генов-параметров, отвечающих наибольшему расхождению между исходными значениями коэффициентов устойчивости насыпи и значениями, полученными теоретически в результате прогноза.
3. скрещивание особей, представляющих наборы генов-параметров, значения функции пригодности которых являются наибольшими;
4. операция мутации над потомством, чьи гены участвуют в настраиваемой модели прогнозирования;
5. отбор особей в новую популяцию;
6. проверка критерия остановки алгоритма;
7. выбор наилучшей особи.

Общая схема генетического алгоритма выглядит следующим образом (рис. 3.10):

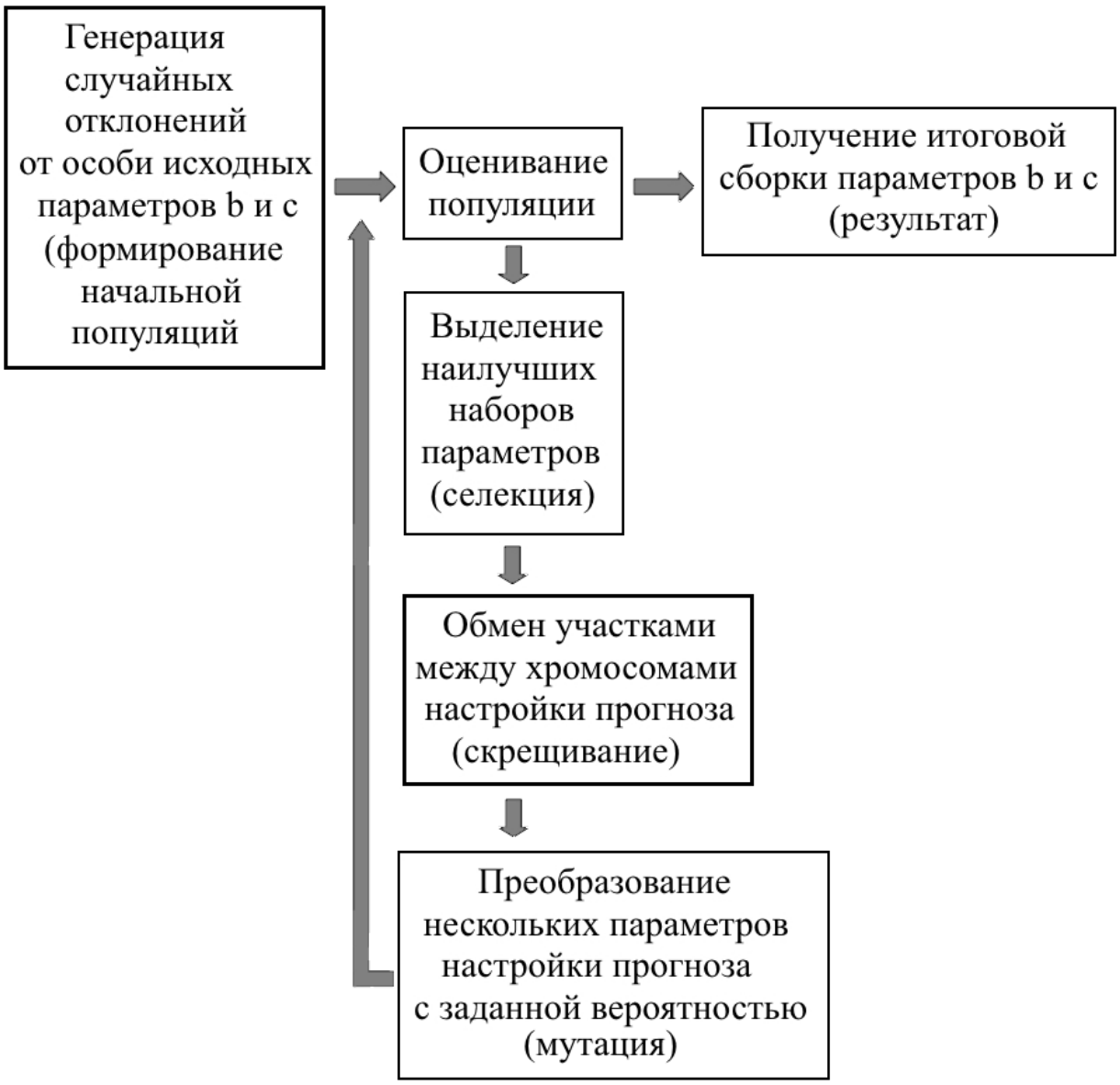

Рис. 3.10. Общая схема генетического алгоритма

Описанный алгоритм реализуется с использованием языка PHP.

В генетических алгоритмах, использующих двоичное кодирование, с увеличением пространства поиска снижается точность решения. В данной задаче каждый из генов остается вещественным значением. В связи с отсутствием операции кодирования/декодирования в двоичную систему, существенно повышается скорость работы алгоритма.

На первом этапе разработанного генетического алгоритма в качестве исходной особи выбирается особь, генами которой являются выбранные экспертами параметры b и c. Набор данных параметров, отвечающих различным лингвистическим оценкам переменной значения коэффициента устойчивости насыпи, образует хромосому. Гены этой хромосомы характеризуют случайные отклонения параметров от исходных.

```
$const_popul=20;
$popul=$const_popul;
For ($j=1; $j<=$popul; $j++)
{
For ($i=1; $i<=20; $i++)
{
 $param[$j][$i]=rand(0.04,-0.04);
}
}
```

Результатом работы следующего цикла является собственно формирование начальной популяции:

```
for ($k=1; $k<=$popul; $k++)
{
$b['n']=0.985+$param[$k][1]+0.1;
$b['ns']=0.995+$param[$k][2]+0.1;
$b['s']=1+$param[$k][3]+0.1;
$b['vs']=1.005+$param[$k][4]+0.1;
$b['v']=1.015+$param[$k][5]+0.1;
$s['n']=0.021+$param[$k][6]+0.1;
$s['ns']=0.015+$param[$k][7]+0.1;
$s['s']=0.013+$param[$k][8]+0.1;
```

$s['vs']=0.015+$param[$k][9]+0.1;

$s['v']=0.018+$param[$k][10]+0.1;

Популяция, особи которой содержат в качестве генов параметры b и c, представляется следующим образом:

$$\{\Delta b_H^1, \Delta b_{HC}^1, \Delta b_C^1, \Delta b_{BC}^1, \Delta b_B^1, \Delta c_H^1, \Delta c_{HC}^1, \Delta c_C^1, \Delta c_{BC}^1, \Delta c_B^1\},$$
$$\{\Delta b_H^2, \Delta b_{HC}^2, \Delta b_C^2, \Delta b_{BC}^2, \Delta b_B^2, \Delta c_H^2, \Delta c_{HC}^2, \Delta c_C^2, \Delta c_{BC}^2, \Delta c_B^2\},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\{\Delta b_H^M, \Delta b_{HC}^M, \Delta b_C^M, \Delta b_{BC}^M, \Delta b_B^M, \Delta c_H^M, \Delta c_{HC}^M, \Delta c_C^M, \Delta c_{BC}^M, \Delta c_B^M\}.$$

Схематически популяция примет вид, изображенный на рис. 3.11.

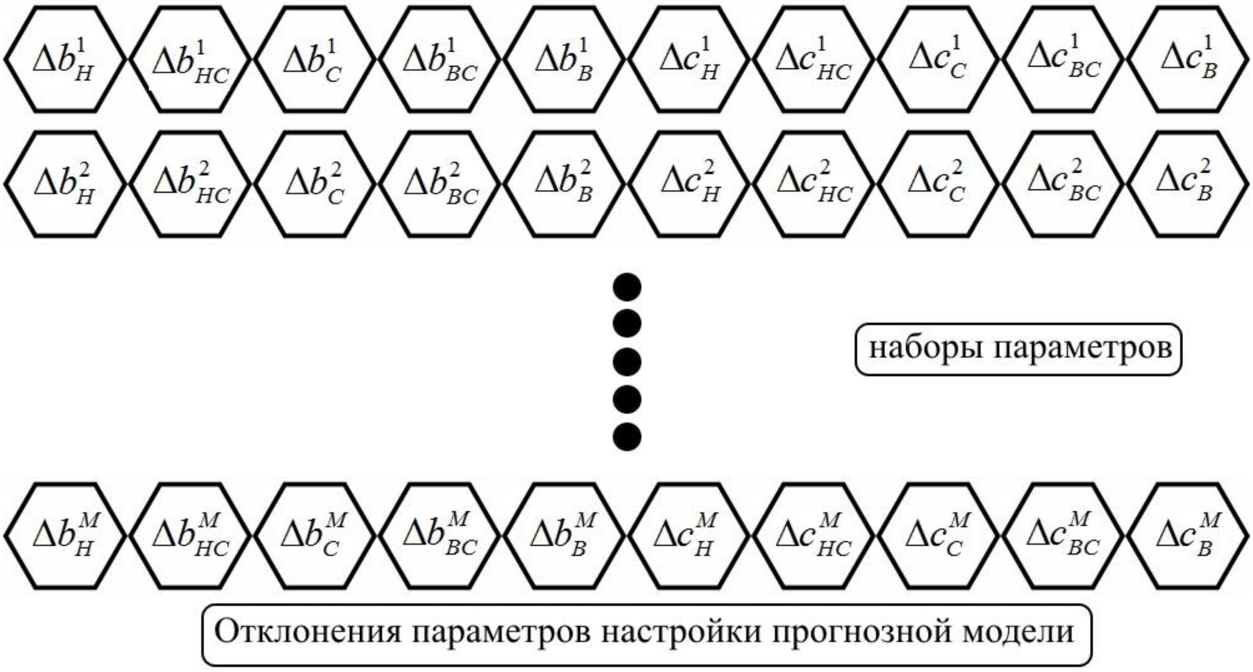

Рис. 3.11. Формирование начальной популяции

Вслед за формированием начальной популяции необходимо оценить особи, отделив наиболее приспособленные хромосомы от наименее приспособленных. Для этого используется целевая функция (3.6).

Второй этап генетического алгоритма предусматривает выбор особей, которые будут использоваться в продолжение популяции. В данном алгоритме применятся метод, схожий с селекцией на основе заданной шкалы. Популяция сортируется от «лучшей» к «худшей» особи. Определение качества особи происходит с помощью функции пригодности, которая строится на основе целевой функции. Целевая функция представлена формулой (3.12) в соответствии с математической постановкой задачи оптимизации. В разработанном алгоритме наибольшая эффективность достигается с использованием выбора хромосом на основе значений функции пригодности. Так, из оставшихся особей в популяции, особь с наибольшим значением целевой функции обменивается генетическим кодом (участками хромосом) с особью, значение целевой функции которой наименьшее. И так далее, до тех пор, пока два направления не сойдутся на среднем значении функции пригодности. Это позволяет избежать ранней сходимости алгоритма в итоге попадания в локальный оптимум.

На следующем этапе генетического алгоритма пары родительских хромосом обмениваются участками, формируя потомство. Данный оператор называется оператором скрещивания или оператором кроссовера.

Пример работы кроссовера изображен на рис. 3.12.

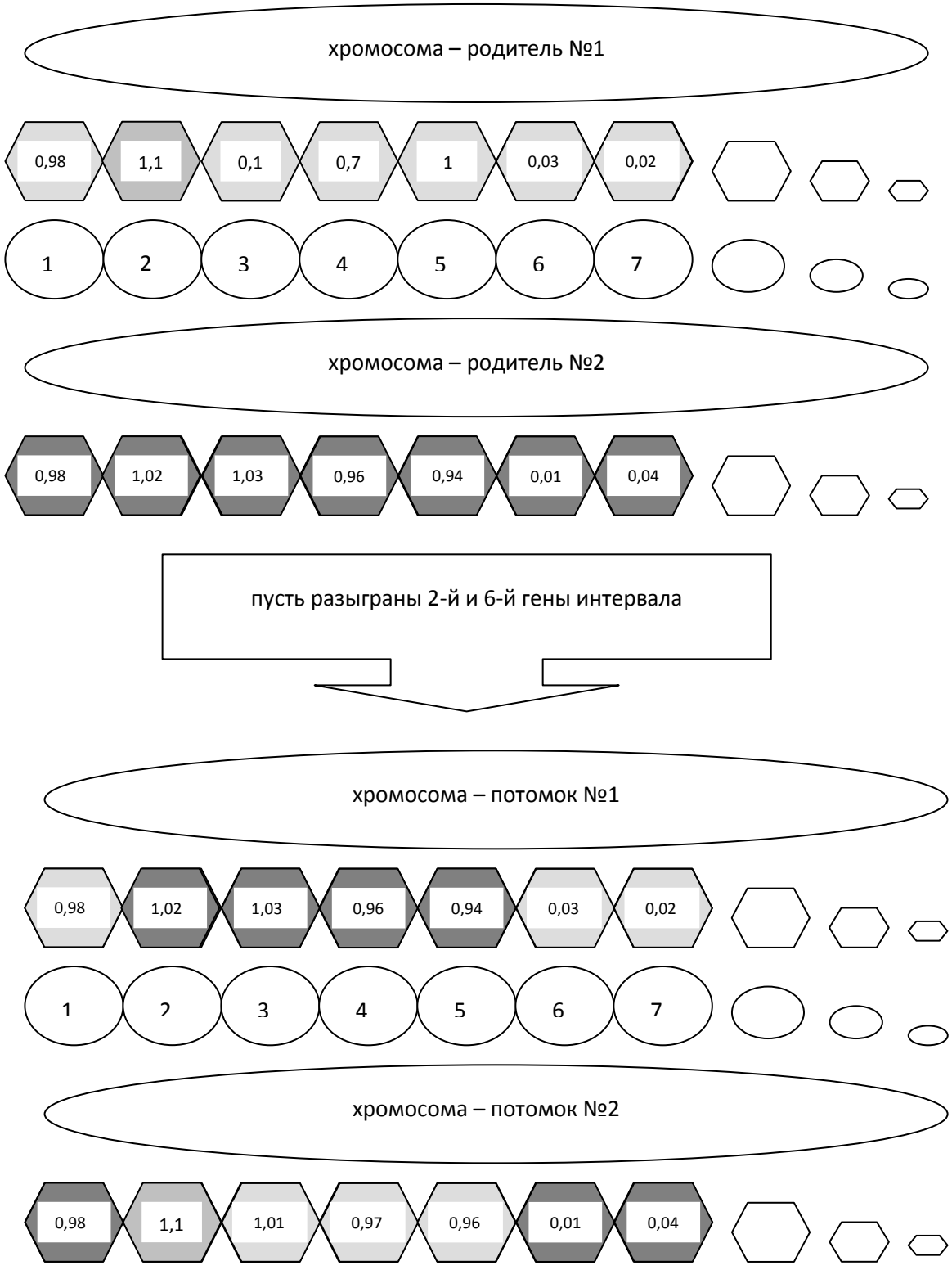

Рис. 3.12. Оператор кроссинговера для набора параметров модели прогнозирования

Выбранные для этого случайным образом пары родителей в соответствии с вероятностью скрещивания $0{,}5 \le p_c \le 1$ дают пары потомков до тех пор, пока количество потомков не достигнет начального размера популяции. В данной работе, наряду с классической схемой реализации скрещивания предлагается $BLX-\alpha$ оператор кроссовера, преимущества которого раскрываются при вещественном способе кодирования. При этом вводится обозначение: $g_k^{(1)}, g_k^{(2)}$ – гены с номером k в родительских хромосомах, $h_k^{(1)}, h_k^{(2)}$ – гены с номером k в хромосомах потомков. Тогда значения генов потомков выбираются случайно в соответствии с равномерным распределением из отрезка $[c_{\min} - \alpha\Delta_k, c_{\max} + \alpha\Delta_k]$,

где α – константа, $c_{\min} = \min\{g_k^{(1)}, g_k^{(2)}\}$, $c_{\max} = \max\{g_k^{(1)}, g_k^{(2)}\}$, $\Delta_k = c_{\max} - c_{\min}$. Обозначенный отрезок выглядит следующим образом:

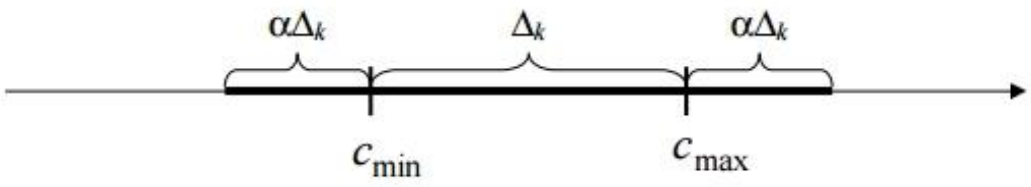

Рис. 3.13. Интервал для $BLX-\alpha$ кроссовера

Любой оператор кроссовера имеет способность к разрушению хромосом-родителей. Разрушающая способность $BLX-\alpha$ кроссовера определяется заданным значением α, а также разностью значений соответствующих генов хромосом-родителей. Описанная особенность подходит для целей данной работы, поскольку модифицированное формирование начальной популяции предусматривает незначительные отклонения в родительских хромосомах. Выбранный подход минимизирует разрушающую способность кроссовера на 12%.

В результате работы оператора кроссинговера размер популяции увеличивается на 50%.

Следующим основным оператором является оператор мутации. Это языковая конструкция, позволяющая на основе преобразования родительской хромосомы создавать хромосому потомка. Оператор мутации для вещественного кодирования заключается в изменении всех или нескольких генов при заданной вероятности P_M. Внесение случайных изменений в хромосомы помогает алгоритму поиска оптимальных параметров прогнозирования выбрать из локальных экстремумов и предотвратить преждевременную сходимость. При этом характерны следующие особенности:

- количественное изменение гена задается случайным образом из заданного отрезка $[-\xi, \xi]$;
- распределение может быть как равномерным, так и нормальным.

В зависимости от того, сколько генов мутирует, различают: одноточечный, двухточечный и многоточечный оператор мутации. Степень вероятности мутации P_M задается, как правило, величиной обратной количеству генов в хромосоме ($P_M = N2^{-1}$). Вследствие этой процедуры количество потомков удваивается и в целом размер самой популяции становится в два раза больше. Благодаря этому пространство поиска решений вырастает уже на первом шаге, что позволяет, в свою очередь, избежать попадания в локальный оптимум.

В данной работе применяется многоточечный оператор мутации, при котором множество точек мутации также задается случайным образом (рис. 3.14).

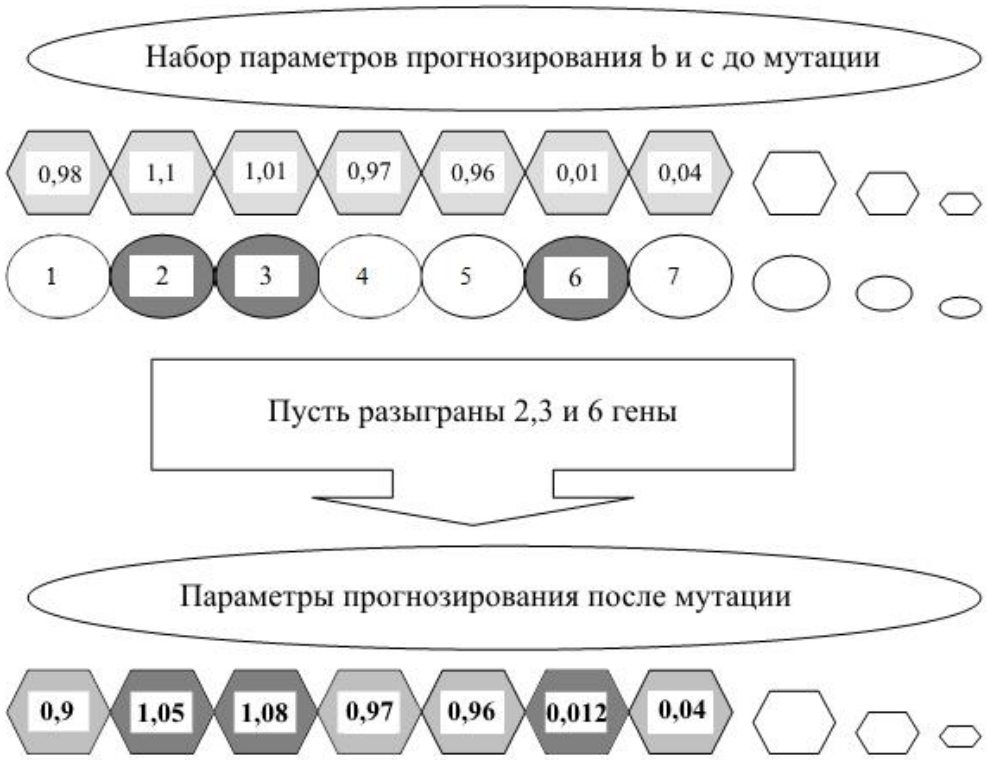

Рис. 3.14. Оператор мутации над параметрами прогнозной модели

Далее цикл работы генетического алгоритма замыкается и происходит повторное обращение этапу отбора, на котором количество особей сводится к изначальному. Для этого снова используется целевая функция (формула 3.6). Отбираются особи с наибольшим значением целевой функции.

Описанные этапы работы алгоритма объединяются в одну итерацию. Итерационный процесс генетического алгоритма продолжается до тех пор, пока не пройдет заданное число поколений или не выполнится какой-либо иной критерий останова. Описываемый генетический алгоритм использует следующий критерий останова. Алгоритм завершает свою работу, когда разница между значениями целевой функции «худшей» особи и «лучшей» практически не существует.

Целевая функция предложенного алгоритма характеризует отклонения прогнозируемых значений коэффициента устойчивости от исходных (экспериментальных) значений.

Для настройки разработанной прогнозной модели можно достаточно одного цикла изменения значений коэффициента устойчивости. Варианты сборок экспертных параметров b и c, полученные в результате запусков генетического алгоритма, представлены в Приложении 6. Модель прогнозирования с использованием настроенных функций принадлежности, дает результаты наиболее близкие к исходным. Приведенная погрешность прогнозирования составляет 2,09%.

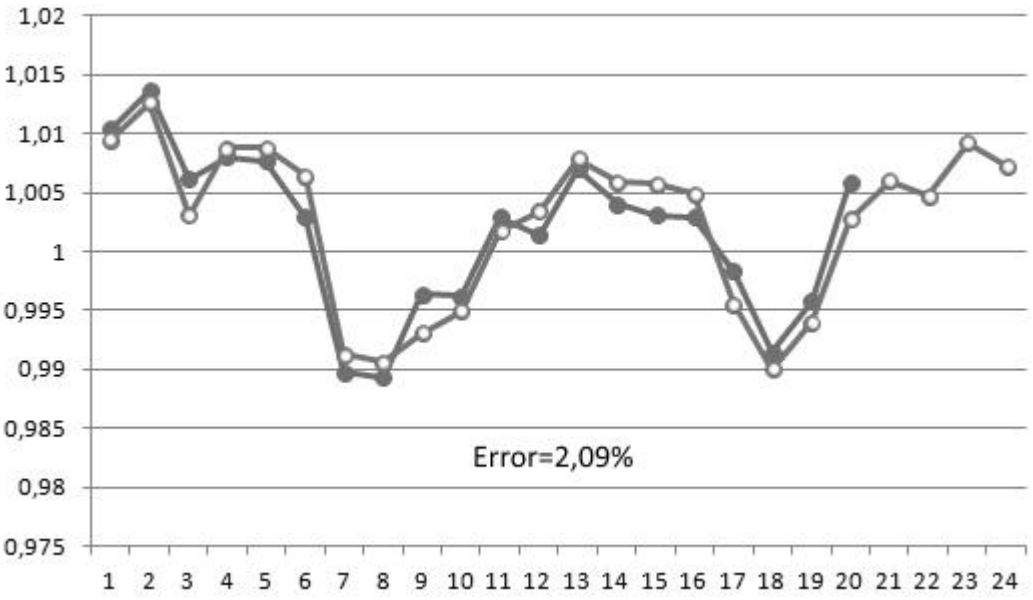

Рис. 3.15. Результаты прогнозирования после настройки модели

На рис. 3.15 видно, что прогнозируемые данные мало отличаются от экспериментальных цифр. Это позволяет с определенной степенью уверенности полагать, что в будущем значение коэффициента устойчивости грунтового массива будет мало отличаться от прогнозируемого.

3.3. Методы анализа качества прогнозов

Важным критерием оценки программного обеспечения, реализующего прогнозирование, является верификация прогнозов. Целью верификации является объяснить точность и обоснованность прогнозов. На практике это означает объединение определенных критериев и методов, позволяющих дать оценку прогнозу.

Для того чтобы оценить точность прогнозирования, наиболее часто прибегают к ретроспективной оценке прогноза. Ретроспективное прогнозирование означает прогноз тех значений, которые уже имеются.

Необходимо разделить всю имеющуюся информацию на две части. Первой части соответствуют наиболее ранние исходные данные. Соответственно, вторая часть захватывает наиболее поздние исходные данные. Данные, относящиеся к первой части информации, составляют, собственно, ретроспекцию. Данные ретроспекции позволяют оценить параметры выбранной модели прогнозирования, тогда как оставшиеся во второй части данные окажутся в роли фактических данных показателя, в отношении которого осуществляется прогноз. Ошибка прогнозирования, полученная в результате ретроспективной оценки, укажет на точность выбранного метода прогнозирования, что может доказать или опровергнуть его обоснованность.

В проведении анализа качества прогнозов все применяемые показатели, как правило, разделяют на три типа: абсолютные, сравнительные и качественные.

Говоря об абсолютных показателях, имеют в виду такие показатели, используя которые можно количественно определить ошибку прогноза с использованием единиц измерения объекта, в отношении которого осуществляется прогноз. В ряде случаев значение данной величины указывается в процентах. В число абсолютных показателей входят

среднеквадратичная ошибка, абсолютная ошибка, средняя абсолютная ошибка, относительная ошибка, средняя относительная ошибка.

Абсолютная ошибка определяется следующим образом:

$$\Delta_{пр} = \left| y_i - y^* \right|, \tag{3.7}$$

где y_i – фактическое значение, а y^* – прогноз.

Среднее абсолютное значение ошибки определяется ниже:

$$\overline{\Delta_{пр}} = \frac{\sum_{i=1}^{n} (y_i - y_i^*)}{n}. \tag{3.8}$$

Среднеквадратической ошибке соответствует следующая формула:

$$\sigma_i = \sqrt{\frac{\sum_{i=1}^{n} (y_i - y_i^*)^2}{n}}. \tag{3.9}$$

Во многих статистических распределениях среднее абсолютное значение ошибки соотносится со среднеквадратической ошибкой следующим образом:

$$\sigma_i = 1{,}25\overline{\Delta_{пр}}. \tag{3.10}$$

Слабой стороной перечисленных выше показателей является то, что они в значительной мере определяются масштабом измерения уровней рассматриваемых явлений.

Что касается относительной ошибки, то она выражается в процентах следующим образом:

$$\varepsilon_{пр} = \frac{y_i - y_i^*}{y_i} 100\%, \tag{3.11}$$

Среднюю относительную ошибку иногда называют ошибкой аппроксимации. Для ее подсчета применяют следующую формулу:

$$\overline{\varepsilon_{пр}} = \frac{\sum_{i=1}^{n} \frac{\left| y_i - y_i^* \right|}{y_i} 100\%}{n}, \tag{3.12}$$

Показатель относительной ошибки применяется, когда требуется сравнить точность прогнозов. Характерные показатели относительной ошибки и их интерпретация приведены в таблице 1. Кроме того, на основании работы [56] в последний столбец таблицы внесены значения точности некоторых известных методов, в том числе, выбранного в третьей главе диссертации метода, основанного на нечетких системах.

Таблица 3.3.Данные прогнозирования и интерпретация

$\varepsilon_{пр}$	Интерпретация	Методы прогнозирования
<10	Высокая точность	Нечеткие методы
10 – 20	Умеренная точность	Метод наименьших квадратов, экспоненциальное сглаживание
20 – 50	Удовлетворительная точность	Распознавание образов, цепи Маркова
>50	Неудовлетворительная точность	–

Из таблицы видно, что наилучший результат по сравнению с другими показывают нечеткие методы прогноза. Однако программная реализация описанного в настоящей работе метода предполагает использования генетического алгоритма, что позволяет снизить ошибку до 2%, что является преимуществом выбранного в третьей главе диссертации метода. Существует потребность в прогнозировании таких условий.

3.4. Выводы

1. Разработана модель прогнозирования коэффициента устойчивости сыпучей среды с помощью формализации понятия «предельное состояние» на основе аппарата нечеткой логики.

2. Построена формализованная модель управления коэффициентом устойчивости на базе ситуационной сети.
3. Разработан модифицированный генетический алгоритм, адаптируемый к решаемой задачи исследования насыпи на устойчивость за счет представления параметров настройки прогнозной модели в качестве генов.

Глава IV. Моделирование прогнозирования поведения насыпи, проведение вычислительного эксперимента

Данная глава описывает составные компоненты программного продукта и их функциональную взаимосвязь. Рассматриваемые в настоящей диссертации задачи решаются с использованием CUDA, PHP, MySQL, HTML. Программно-аппаратная архитектура параллельных вычислений CUDA за счет использования особенностей графического процессора позволяет не только выполнить множество вычислений за короткий срок, но и визуально отследить ход моделируемого процесса [59].

PHP представляет собой скриптовый язык общего назначения [58]. Ключевой особенностью это языка является то, что он работает практически на всех хостинг-провайдерах. Выполнение алгоритмов сразу на сервере обусловливает доступ к исходным данным на любом из ресурсов. Кроме того, PHP взаимодействует с базами данных, что предоставляет еще больше возможностей разработчику.

4.1. Описание файлов и классов программного комплекса моделирования динамики сыпучих сред

Поскольку в соответствии с принципами разделенного программирования основная логика приложения выполняется на центральном процессоре, а вычисления – на графическом, в программе можно выделить две части (таблица 4.1):

- низкоуровневая компонента, отвечающая за параллельные вычисления над данными (таблица 4.2);
- высокоуровневая компонента, определяющая начальные параметры системы и обуславливающая шаг системы.

Низкоуровневая компонента вызывается высокоуровневой компонентой. Взаимодействие данных компонент представлено на рис. 4.1 [69].

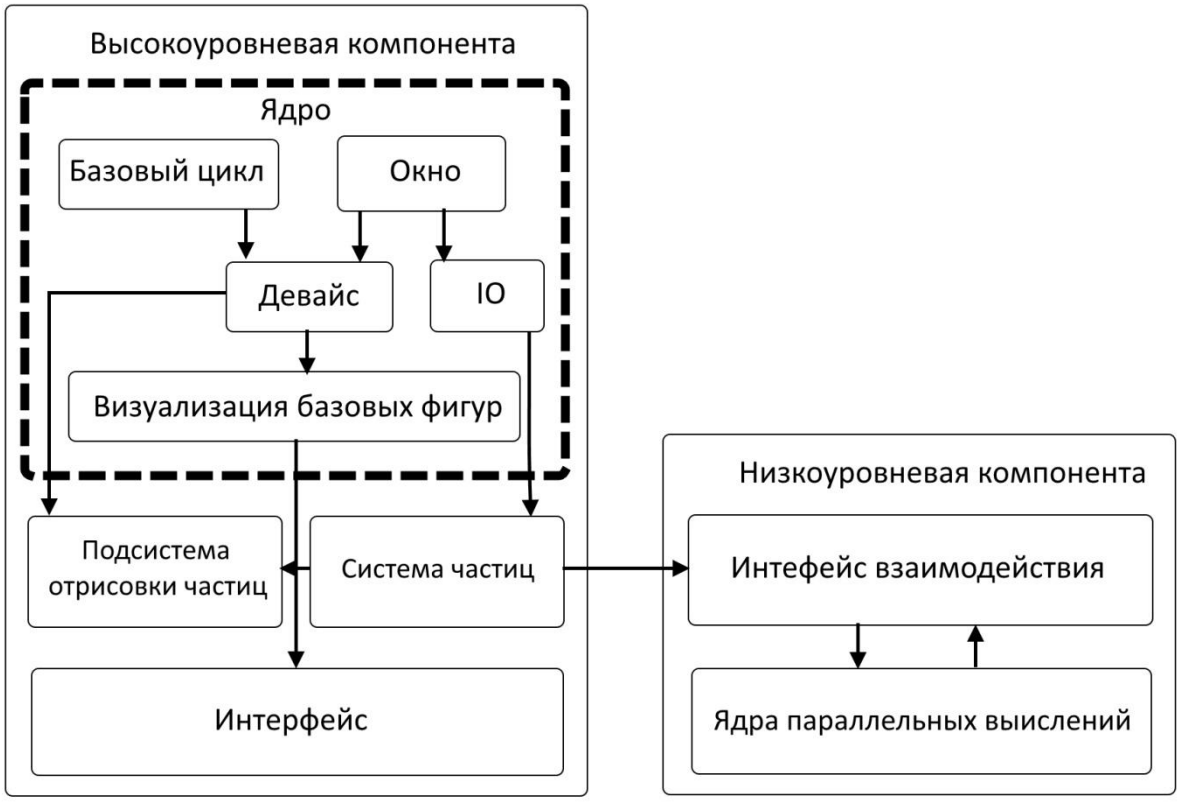

Рис. 4.1. Схема программного комплекс моделирования динамики сыпучих сред

Таблица 4.1. Данные программ, компоненты проекта и файлы

Файл	**Назначение**
Низкоуровневая компонента	
particles.cuh	Содержит код интерфейсной части компоненты. Более подробно см. **Ошибка! Источник ссылки не найден.**.
particles.cu	Реализация интерфейсной части
particles_kernel.cuh	Содержит все необходимые ядра для вычислений, а также дополнительные функции, необходимые для них
Высокоуровневая компонента - ядро	
Window.h, Window.cpp	Класс окна
Device.h, Device.cpp	Класс устройства
Loop.h, Loop.cpp	Класс цикла
ShapeRenderer.h, ShapeRenderer.cpp	Класс отрисовки фигур
Input.h, input_d.h, input.cpp	Классы для работы с устройствами ввода/вывода
ObserveSubject.h	Класс, реализующий подписку объектов
Vertex.h	Содержит доступные типы графических вершин
Log.h, log.cpp	Класс log для записи событий в консоль и в файл
Camera.h, Camera.cpp	Класс камеры для представления трехмерного мира
Stopwatch.h, Stopwatch.cpp	Класс таймера
Macro.h, macro.cpp	Несколько макросов, упрощающих программирование
Основное приложение	
Main.cpp	Содержит точку входа в приложение, а также класс Controller
ParticleSystem.cpp, ParticleSystem.h	Класс системы частиц
ParticleSystemRenderer.h, ParticleSystemRenderer.cpp	Класс, который выполняет функцию вывода частиц на экран
BasicRenderer.h, BasicRenderer.cpp	Класс базового отрисовщика частиц, выводящий их в виде простых шаров
Profile.h, profile.cpp	Несколько классов и структур для профилирования приложения

Таблица 4. 2. Используемые данные

Массив	**Назначение**	**Тип данных**	**Кол-во элементов**
pPos	Содержит позицию частиц и радиус	float4 (16 байт)	Количество частиц
pVel	Содержит скорость частиц и обратную массу		
pHash	Содержит номера ячеек, в которых находятся частицы	uint (4 байта)	
pIndex	Необходим для индексации частиц		
pCellStart	Содержит начало ячеек в глобальном списке	uint (4 байта)	Количество ячеек
pCellEnd	Содержит конец ячейки в глобальном списке		

Что касается данных о частицах, то целесообразно размещать их в двух массивах, длина которых равна общему количеству частиц системы. Элементы массивов являются 128-битными. Один из массивов хранит данные о положении и значения радиусов частиц, другой – скорость и обратную массу. Номера элементов каждого из массивов совпадают с номером частицы в системе. Размер элементов в 128 бит делает доступ к данным наиболее доступным, а линейность массива обусловливает оптимизацию данных, описанную в пункте 1.3.

Алгоритм сортировки, предложенный в п. 2.2, нуждается в дополнительных массивах, которые хранят индексы частиц и номера ячеек. Элементы этих массивов имеют тип unsigned integer, что означает целый тип без знака. Кроме того, существуют еще два массива такого же типа, которые содержат индексы начала и конца ячеек.

Низкоуровневая компонента приложения, работающая на девайсе, имеет специальный интерфейс с функциями, необходимыми для выполнения шага системы. Вся необходимая для этого память также расположена на девайсе, однако распределяется она высокоуровневой компоненты, реализованной на хосте.

Компонента высокого уровня описывается языком C++ и выполняет осуществляет следующие задачи:

- связь с операционной системой;
- визуализация результатов; связь с пользователем;
- контроль над шагом системы;

Графика в приложении реализуется исключительно на ядре высокоуровневой компоненты. К компонентам ядра относят:

- Window – класс окна, работающий с сообщениями операционной системы;
- Device – класс устройства, инициализирующий DirectX 11;
- Loop – класс цикла приложения, осуществляющий контроль над графической составляющей приложения;
- ShapeRenderer – класс рисования фигур при помощи линий;
- Input – семейство классов для работы с устройствами ввода и вывода;
- Camera – класс, отвечающий за демонстрацию виртуального мира.

Благодаря классу Window сообщения операционной системы становятся доступными для других объектов, называемых приемниками сообщений и наследуемых от интерфейса IWindowMessageReciever.

Класс Loop обрабатывает сообщения типа Update и Render и передает их объектам – участникам цикла. Частота вызова таких сообщений является различной. Объекты наследуют интерфейс ILoopParticipant.

Более детально с компонентами ядра можно ознакомиться в Приложении 1.

Чтобы объединить все компоненты приложения, используется контроллер. Контроллер также может принимать сообщения и непосредственно участвовать в цикле. В его задачи входят инициализация графики, запрос устройств ввода-вывода, задание начального состояния системы частиц. Для контроллера существует одноименный класс – Controller.

Главную роль в реализуемом приложении занимает система частиц. В связи с этим, чтобы задавать начальное положение системы, отображать результат и осуществлять контроль над системой, в дополнение к низкоуровневой компоненте используется специальный класс ParticleSystem.

Система частиц (таблица 4.3, 4.4, 4.5) раскладывается на логическую и графическую составляющие. Задание начального положения системы и контроль над возлагаются на логическую часть, тогда как графической часть отводится роль визуализации результата. Подобное разделение системы значительно упрощает разработку приложения. Объект данного класса задается следующим набором параметров:

Таблица 4.3. Настройка системы частиц

Параметр	Тип	Назначение
spawnParticles	Bool	Включение и выключение автоматического появления частиц
spawnInterval_s	Float	Интервал появления частиц
spawnAmount	Uint	Количество частиц, которые появляются одновременно
dt	Float	Шаг времени
maxParticles	Uint	Максимальное количество частиц
initialParticleCount	Uint	Начальное количество частиц
minRadius	Float	Минимальный радиус частицы
maxRadius	Float	Максимальный радиус частицы

simParams	SimulationParams	Настройки визуального моделиования
physProps	PhysProperties	Настройка физических свойств

Таблица 4.4. Настройка визуализального моделирования

Параметр	Тип	Назначение
cellSize	Float	Размер ячейки
gridSize	Uint3	Вектор размера сетки
boundaryLen	Float3	Вектор размера ограничивающего пространства
gravity	Float3	Вектор гравитации

Таблица 4.5. Настройки физических параметров частиц

Параметр	Тип	Назначение
Density	float	Плотность
Stiffness		Упругость
stiffnessStatic		Упругость статических объектов
Restitution		Коэффициент восстановления после отталкивания
coloumbFriction		Сила трения
tangentialDamping		Коэффициент тангенциального демпфирования

Метод Update отвечает за совершение шага системы, тогда как вызов метода Render, соответствующего классу ParticleSystemRenderer, становится необходимым при выводе частиц системы в графическое окно.

Существует специальный класс BasicRenderer, который осуществляет метод Render. Создать объект этого класса можно, указав параметрами объекты классов ParticleSystem и Camera:

m_pParticleSystemRenderer = new BasicRenderer (m_pParticleSystem, m_pCamera);

Таким образом, чтобы обновить систему частиц, необходимо записывают следующее:

```
//обновление системы
m_pParticleSystem->Update ( ) ;
//вывод частицы на экран
    m_pParticleSystemRenderer->Render ( ) ;
```

Управление программой с помощью клавиатуры задается в соответствии следующим образом (таблица4.6):

Таблица 4.6. Управление с клавиатуры

Клавиша	Действие
W	Движение цели камеры вперед
A	Движение цели камеры влево
S	Движение цели камеры назад
D	Движение цели камеры вправо
⇦	Вращение камеры вокруг цели по часовой стрелке
⇨	Вращение камеры вокруг цели против часовой стрелке
=	Приближение камеры к цели
-	Удаление камеры от цели
M	Включение и отключение ручного режима визуального моделирования
Space	Следующий шаг
E	Включение и выключение использования вспомогательных частиц

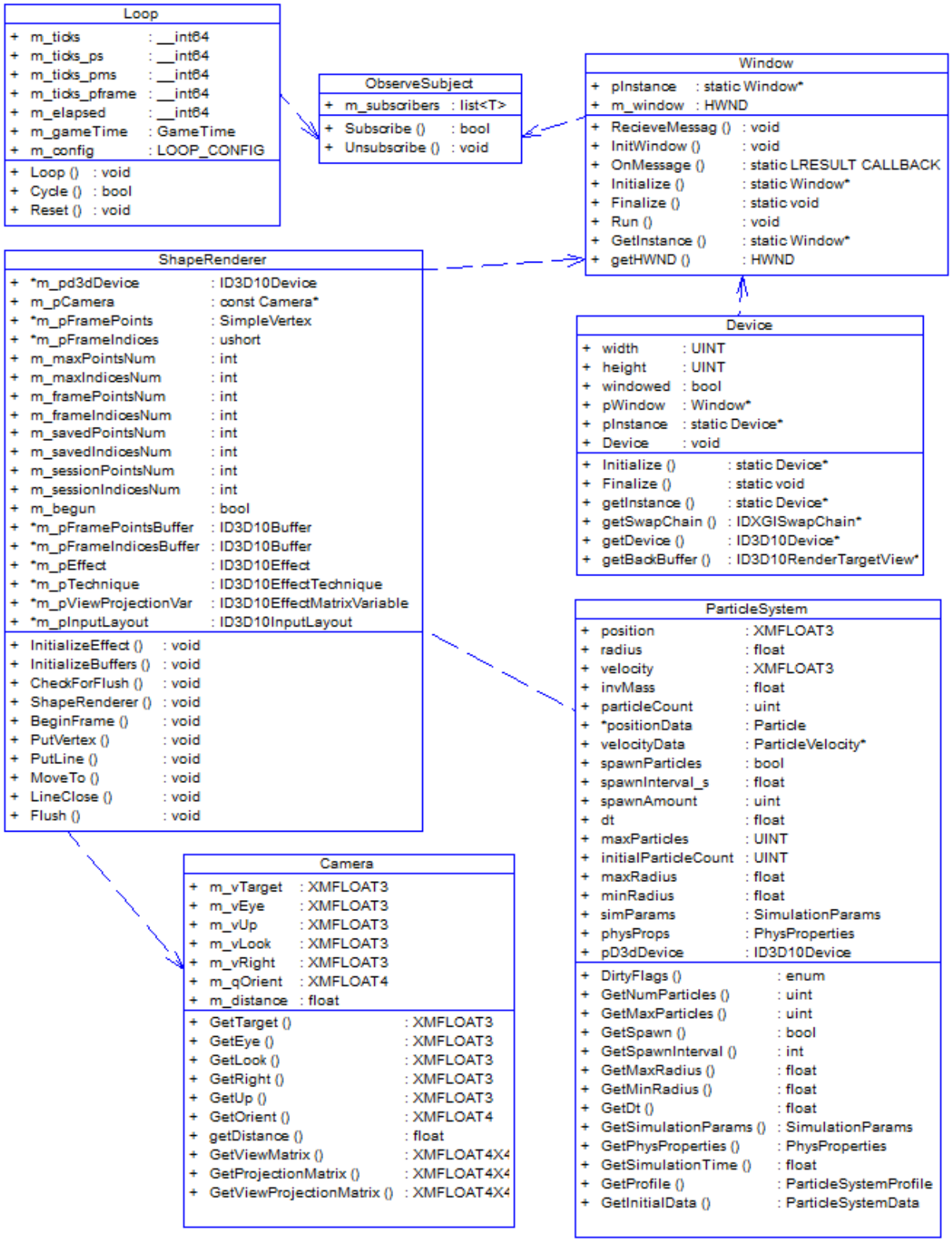

Рис. 4.2. Диаграммы классов для графического процессора

4.2. Описание программного модуля прогнозирования коэффициента устойчивости грунтового массива

Модуль прогнозирования коэффициента устойчивости грунтового массива описывается файлом PROGNOZ.PHP. Реализация настоящего модуля осуществляется в несколько шагов. Структура модуля представлена на рис. 4.2. Вначале ключевую роль играет база данных с необходимым для пользователя количеством первоначальных сведений. Вслед за этим собранные сведения необходимо представить в той форме, которая соответствует создаваемой в настоящей работе прогнозной модели.

Рис. 4.3. Блок-схема прогнозного модуля

На следующем шаге синтезируется модель прогнозирования, которая опирается на оценки экспертов. В соответствии с выбранной в третьей главе методологией осуществляется построение первичного прогноза. Здесь также участвуют и правила, также описанные в третьей главе. Далее первичный прогноз исследуется на погрешность. По завершении первичного прогноза эксперты предлагают к выбору параметры, при помощи которых становится возможной настройка прогнозной модели. С целью наиболее точно настроить эти параметры, в этой работе предлагается использовать генетический алгоритм. В терминах генетических алгоритмов для начала такой настройки необходимо хранить исходную популяцию. Для этого в программе используется специальный массив $PARAM[$k][$j]. Сам генетический алгоритм также содержит несколько шагов. В начале работы генетического алгоритма происходит обращение к той особи из популяции, которая строилась в соответствии с экспертными оценками. Следующий шаг генетического алгоритма обращается к оператору скрещивания. При выполнении этого оператора случайным образом определяются точки внутри хромосом, в которых они делятся на две части и обмениваются ими между собой. Оператор мутации играет второстепенную роль по сравнению с оператором скрещивания. Данный оператор затрагивает только одну хромосому и означает случайную перестановку генов в хромосоме. После того, как полученные особи существенно расширяют популяцию, становится необходимым оценить их пригодность. Для этого вызывается функция пригодности. Чтобы оценить работу функции пригодности, нужно обращать внимание на фактическую погрешность при прогнозе. Данная погрешность опирается на параметры, которые определяются каждой конкретной особью. В настоящей задаче насчитывается двадцать параметров. В двадцать первую ячейку массива записывается соответствующее значение функции пригодности. Чтобы исключить из популяции наименее пригодные особи, используется оператор селекции. Иными словами, данный оператор способствует воспроизводству хромосом с наибольшим значением функции

пригодности. Далее массивы попарно сливаются в соответствии по правилам точечного кроссинговера. При работе с оператором мутации необходимо проявлять наибольшую осторожность, что обусловлено спецификой этого оператора. Поскольку каждый из генов хромосом представлен определенными параметрами, предусмотренными рассматриваемой в данной задаче модели прогнозирования, обмен соседними генами в соответствии с классическим представлением об операторе мутации может исключить нужное решение из пространства поиска. В связи с этим при реализации данной задачи используется другой подход, суть которого заключается в том, чтобы сдвинуть значение отдельно взятого гена в ту или иную сторону. При этом величина сдвига задается случайно. Такая модификация оператора обеспечивает наиболее глубокий поиск лучшего решения, а также исключает существенные его отклонения, которых было бы не избежать при классическом операторе мутации. Вслед за использованием операторов скрещивания, мутации и селекции снова выполняется отсеивание наихудших по функции пригодности особей, с помощью чего удается вернуть размер популяции к ее изначальному размеру. Текущая последовательность действий по отбору наилучшей совокупности параметров представляет собой цикл. Этот цикл работает до тех пор, пока погрешность не сравняется с некоторой малой установленной заранее константой. Нельзя пренебречь и тем фактом, что цикл также может остановиться и в том случае, когда количество итераций превысит установленный в коде программы лимит. При остановке цикла работа модуля прогнозирования подходит к своему концу, когда модуль отбирает наилучшее из предоставленных решений. Далее вступает в работу модуль, воспроизводящий итог прогноза. С помощью данного модуля результаты становятся доступными пользователю.

В таблице 4.7 представлена выборка из прогноза, ставящая в соответствие определенное значение коэффициента устойчивости насыпи ее геометрии

Таблица 4.7. Результат работы прогнозной модели

Временной шаг симуляции	Значение коэффициента устойчивости	Геометрия насыпи
t=21	μ=1,002	
t=22	μ=1,0035	
t=23	μ=1,007	
t=24	μ=1,0072	

Симуляция, в которой участвовало 102555 частиц, производилась в течение 90 секунд.

4.3. Внутренние спецификации программного комплекса

В интерфейсе высокого уровня инициализируется система и осуществляется контроль над ее шагом (таблица 4.8).

Таблица 4.8. Элементы ядра высокоуровневой компоненты

Название функции	**Назначение**	**Используемые данные**
SetSimulationParameters	Установка параметров симуляции	SimulationParams
SetPhysProperties	Установка физических параметров системы	PhysProperties
IntegrateParticles	Интегрирование частиц	pPos, pVel, pHash, pIndex
SortParticles	Сортировка частиц	pHash, pIndex
FindCellStart	Поиск начала и конца ячеек	pCellStart, pCellEnd, pHash, pIndex, pPos, pVel, pPosSorted, pVelSorted
ProcessCollisions	Обработка столкновений	pCellStart, pCellEnd, pHash, pIndex, pPos, pVel, pPosSorted, pVelSorted

Класс ObserveSubject содержит набор функций для оформления подписки на события (таблица 4.9).

Таблица 4.9. Работа с событиями

Функция	**Параметры**	**Значение**
Subscribe	subscriber – объект, который желает подписаться на события	Оформление подписки
Unsubscribe	subscriber – объект, который хочет отменить подписку	Отмена подписки

Работа с инициализацией, идентификацией и финализацией окон реализуется посредством класса Window (таблица 4.10).

Таблица 4.10. Работа с окнами

Функция	**Параметры**	**Назначение**
static Initialize	1. className – имя класса окна для ОС 2. title – заголовок 3. width – ширина окна 4. height – высота окна	Инициализация окна. Возвращает ссылку на объект класса Window.
static GetInstance	нет	Статический метод. Возвращает созданный объект класса или NULL, если инициализации не было
getHWND		Возвращает идентификатор окна, созданного в ОС
Finalize		Финализация окна
Run		Запуск цикла приложения. Окно начинает принимать и рассылать сообщения подписчикам.

Параметры графического окна описываются следующим образом (таблица 4.11).

Таблица 4.11. Работа с графическим окном

Параметр	**Тип**	**Значение**
pWindow	Window*	Указатель на экземпляр класса Window
width	uint	Ширина графического окна
height	uint	Высота графического окна
windowed	bool	Вкл./выкл. оконный режим

Особенностью класса Device как элемента высокоуровневой компоненты является создание лишь одного экземпляра (таблица 4.12).

Таблица 4.12. Функции класса Device

Функция	**Параметры**	**Значение**
static Initialize	config – конфигурация устройства	Инициализация устройства
static getInstance	нет	Возвращает инициализированное устройство или NULL, если инициализации не было
Finalize		Финализация устройства
getSwapChain		Возвращает объект класса IDXGISwapChain
getDevice		Возвращает объект класса ID3D10Device
getBackBuffer		Возвращает объект класса ID3D10RenderTargetView

Класс Loop является классом цикла. Для этого класса представляется возможным оформить подписку посредством объектов, наследованных от интерфейса с названием ILoopParticipant. В таком случае возможны события Update и Render. События Render повторяются ровно такое количество раз, сколько запускается Cycle (таблица 4.13). В отличие от этого, обращение к событию Update происходит установленное количество раз в единицу рабочего времени цикла.

Таблица 4.13. Конфигурация цикла

Параметр	**Тип**	**Значение**
fixedStep	bool	Вкл./выкл. фиксированную частоту шага
targetFPS	uint	Указывает частоту фиксированного шага в кадрах в секунду

Существует ряд параметров для описания работы со временем (таблица 4.14).

Таблица 4.14. Время моделирования

Параметр	**Тип**	**Значение**
total_s	double	Время, прошедшее с начала цикла
elapsed_s	double	Время, прошедшее с предыдущего шага

Для управления временными параметрами используются следующие функции (таблица 4.15).

Таблица 4.15. Функции управления временными параметрами

Функция	**Параметры**	**Значение**
Конструктор	config – конфигурация цикла	Конструктор
Cycle	нет	Произвести один шаг цикла
Reset		Сбросить временные параметры цикла

Вывод линий организуется посредством класса ShapeRenderer. Это происходит с применением точек. Буфер пополняется значениями координат и определенным цветом для каждой точки, что становится возможным благодаря функции PutVertex (таблица 4.16). Функция PutLine, в свою очередь, соединяет созданные точки линиями. В работе использовался и другой метод прорисовки линий, когда перемещение точки задается при помощи функции MoveTo, а прорисовка линий – при помощи функций LineTo. Класс описанных функций представлен более подробно в следующей таблице.

Таблица 4.16. Работа с вершинами и линиями

Функция	**Параметры**	**Значение**
Конструктор	1. pDevice – ссылка на Device 2. pCamera – ссылка на Camera 3. cacheSize – размер кэша точек	Конструктор
BeginFrame	нет	Начать вывод линий
EndFrame		Закончить вывод линий
PutVertex	v – структура SimpleVertex	Поместить вершину
PutVertex	pos – позиция точки color – цвет точки	Поместить вершину
PutLine	Pt1, pt2 – номера точек для соединения	Соединить две точки линией
MoveTo	V - вершина	Установить текущую точку
LineTo	V - вершина	Вывод линии из текущей точки в новую точку, и переместить текущую точку в новую
LineClose	Нет	Соединить текущую точку с начальной
Flush		Вывести содержимое буфера на экран

Как и в большинстве графических приложений, в разработке настоящей работы значительную роль играет класс Camera. Этот класс исследует и изображает виртуальный мир. Камера подразумевает видовую и проекционную матрицы. Заданные матрицы способны преобразовывать вершины с целью формирования конечного изображения на экране. Необходимыми атрибутами камеры являются целевая точка и позиция. Посредством позиции задается удаленность от целевой точки и направление.

Наиболее полно функции работы с камерой представлены в таблице 4.17.

Таблица 4.17. Работа с вершинами и линиями

Функция	**Параметры**	**Значение**
Update	нет	Пересчет всех матриц
Yaw	Theta - угол	Поворот камеры вокруг своей оси
Pitch		Наклон камеры
getTarget	нет	Получить координаты цели
setTarget	vTarget – вектор	Установить координаты цели
getEye	Нет	Получить координаты камеры
GetLook, GetRight, GetUp		Получить базисные векторы системы координат камеры
getOrient		Получить кватернион ориентации камеры
setOrient	qOrient – кватернион	Установить кватернион ориентации камеры
getDistance	Нет	Возвращает расстояние от цели до камеры
setDistance	Distance - расстояние	Установить расстояние от цели до камеры
getViewMatrix	нет	Получить видовую матриц
getProjectionMatrix		Получить проекционную матрицу
getViewProjectionMatrix		Получить матрицу вид * проекция

Все возможные параметры симуляции описываются в интерфейсной части класса ParticleSystem (таблица 4.18).

Таблица 4.18. Методы симуляции

Метод	**Значение**
Update	Произвести один шаг системы частиц
GetPositionData	Возвращает ID3D10Buffer с текущими данными о положении частиц
GetNumParticles	Получить количество частиц
GetMaxParticles	Получить максимальное количество частиц
GetSpawn	Получить включено ли автоматическое появление частиц
SetSpawn	Вкл./выкл. автоматическое появление частиц
GetSpawnInterval	Получить интервал появления частиц
SetSpawnInterval	Установить интервал появления частиц
GetSpawnAmount	Получить количество частиц для автоматического появления
SetSpawnAmount	Установить количество частиц для автоматического появления
GetMaxRadius	Получить максимальный радиус частиц
GetMinRadius	Получить минимальный радиус частиц
GetDt	Получить временной шаг симуляции
SetDt	Установить временной шаг симуляции
GetSimulationParams	Получить параметры симуляции
SetSimulationParams	Установить параметры симуляции
GetPhysProperties	Получить физические параметры
SetPhysProperties	Установить физические параметры
GetSimulationTime	Получить текущее время симуляции
GetInitialData	Получить исходные данные частиц

Полная спецификация, описывающая атрибуты и методы основных классов, представлена в виде следующих диаграмм классов формата UML.

4.4. Выводы

1. Разработано программное обеспечение для алгоритмов и моделей, предложенных в предыдущих главах.
2. Описаны файлы и классы программного комплекса, участвующие в алгоритме реализации моделирования динамики сыпучей среды.
3. Приведено описание программного модуля прогнозирования коэффициента устойчивости грунтового массива.
4. Описаны внутренние спецификации программного комплекса.

Заключение

Основным результатом настоящей диссертационной работы является разработка бионических нечетких моделей и алгоритмов для исследования систем многоточечных масс при формировании устойчивой сыпучей насыпи. Следует отметить ключевые результаты:

1. Алгоритм поведения системы многоточечных масс в трехмерном пространстве при формирования сыпучей насыпи с использованием графического процессора.

2. Модель и алгоритм прогнозирования устойчивости насыпи с помощью нечеткой логики и ситуационной сети.

3. Модифицированный генетический алгоритм для настройки коэффициентов модели прогнозирования, позволяющий сделать поиск оптимальных параметров для нечеткой модели более эффективным.

4. Программный комплекс для проведения вычислительного эксперимента по предложенным алгоритмам.

Список использованных источников

1. Герсеванов Н.М. Теоретические основы механики грунтов / Н.М. Герсеванов, Д.Е. Польшин. – М.: Стройиздат, 1948. – 248 с.

2. Флорин В.А. Расчеты оснований гидротехнических сооружений / В.А. Флорин. – М.: Стройиздат, 1948. – 188 с.

3. Флорин В.А. Основы механики грунтов. Т. I. Общие зависимости и напряженное состояние оснований сооружений / В.А. Флорин. – Л.: Госстройиздат, 1959. – 356 с.

4. Соколовский В.В. Статика сыпучей среды / В.В. Соколовский. – Издание 3-е. – М.: Физматгиз, 1960. – 243 с.

5. Соколовский В.В. Теория пластичности / В.В. Соколовский. – М.: Высш. школа, 1969. – 608 с.

6. Цытович Н.А. О проектировании фундаментов по предельным состояниям грунтовых оснований / Н.А. Цытович // Сб. научн. докл. Чешской высшей школы. – Прага, 1958. – С. 53-65.

7. Цытович Н.А. Механика грунтов (краткий курс): учеб. для строитю вузов / Н.А. Цытович. – М.: Высш. шк., 1983. – 288 с.

8. Маслов М.М. Прикладная механика грунтов / М.М. Маслов. – М.: Машестройиздат, 1949. – 378 с.

9. Березанцев В.Г. Осесимметричная задача теории предельного равновесия сыпучей среды / В.Г. Березанцев. – М.: Гостехиздат, 1954. – 120 с.

10. Березанцев В.Г. Расчет оснований сооружений / В.Г. Березанцев. – М.: Стройиздат, 1970. – 207 с.

11. Официальный сайт пакета EDEM. [Электронный ресурс]. URL: http://www.dem-solutions.com (дата обращения: 14.10.2014).

12. Кривцов А.М. Деформация и разрушение твердых тел с микроструктурой / А.М. Кривцов. – М.: Физматлит, 2007. – 304 с.

13. Cundall P.A. A distinct element model for granular assemblies / P.A. Cundall, O.D.L. Strack. – Geotechnique, 1979.

14. Williams J.R. The theoretical basis of the discrete element method / J.R. Williams, G. Hocking, G.G.W. Mustoe. – NUMETA, 1985.

15. Pande G. Numerical modeling in rock mechanics / G. Pande, G. Beer, J.R. Williams. – John Wiley and Sons, 1990.

16. Williams J.R. Superquadric and modal dynamics for discrete elements in concurrent design / J.R. Williams, A.P. Pentland // National science foundation sponsored 1st U.S.: Conference of discrete element methods. – Golden, CO, 1989.

17. Williams J.R. 2nd international conference on discrete element methods / J.R. Williams, G.G.W. Mustoe // IESL Press. – 1992.

18. Williams J.R. Discrete element simulation and the contact problem / J.R. Williams, R. O'Connor //Archives of computational methods in engineering. – 1999. – p. 279-304.

19. Bicanic N. Discrete element methods in Stein, de borst / N. Bicanic // Hughes encyclopedia of computational mechanics. – 2004. – vol. 1.

20. Munjiza A. The combined finite-discrete element method / A. Munjiza. – Wiley, 2004.

21. Вознесенский Е.А. Динамическая неустойчивость грунтов / Е.А. Вознесенский. – 2-е изд. – М.: Эдиториал УРСС, 1999. – 261 с.

22. Дорофеенко С.О. Моделирование сыпучих сред методом дискретных элементов: автореф. дис. … канд. физ.-мат. наук: 01.04.17 / Дорофеенко Сергей Олегович. – Черноголовка, 2008. – 16 с.

23. Медведев В.И. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java / В.И. Медведев. – 2-е изд., испр. и доп. – Казань: РИЦ «Школа», 2010. – 444 с.

24. Официальный сайт Open MP. [Электронный ресурс]. URL: http://www.openmp.org/wp (дата обращения: 21.11.2014).

25. Антонов А.С. Параллельное программирование с использованием технологии MPI: учеб. пособие / А.С. Антонов. – М.: Изд-во МГУ, 2004. – 71 с.

26. S. Plimpton. Fast parallel algorithms for short-range molecular dynamics / S. Plimpton // J. Comp. Phys. – 1995. – vol. 117. – p. 1-19.

27. LAMMPS Molecular Dynamics Simulator. [Электронный ресурс]. URL: http://www.lammps.sandia.gov (дата обращения: 14.02.2014).

28. LIGGGHTS Open Source Discrete Element Method Particle Simulation Code. [Электронный ресурс]. URL: http://www.cfdem.com (дата обращения: 17.02.2014).

29. Карвацький А.Я. Оцінка можливості застосування моделі дискретного елемента для моделювання динаміки сипучого матеріалу в електрокальцинаторі / А.Я. Карвацький [и др.] // Збірник доповідей науково-практичної конференції студентів, аспірантів та науковців «Ресурсоенергоефективні процеси, технології та обладнання хімічних виробництв і підприємств будівельних матеріалів». – К.: Січкар, 2012.

30. Pöschel T. Computational granular dynamics models and algorithms / T. Pöschel, T. Schwager. – Springer, Berlin Heidelberg New York, 2005. – 322 p.

31. Бурдо А.И. К вопросу систематизации методов и алгоритмов прогнозирования/ А.И. Бурдо, Э.И. Тихонов // Материалы межрегиональной конференции «Студенческая наука – экономика научно-технического прогресса». – Ставрополь: СевКав ГТУ, 2001. – с. 33-34.

32. Владимирова Л.П. Прогнозирование и планирование в условиях рынка: учеб. пособие / Л.П. Владимирова. – М.: Издательско-торговая корпорация «Дашков и Ко», 2004. – 279 с.

33. Комарова Л.Г. Нейрокомпьютеры: учеб. пособие для ВУЗов / Л.Г. Комарова, А.В. Максимов. – М.: МГТУ им. Н.Э. Баумана, 2000. – 320 с.

34. Постон Т. Теория катастроф и ее приложения / Т. Постон, И. Стюарт. – М.: Мир, 1980. – 608 с.

35. Попов Э.В. Экспертные системы / Э.В. Попов. – М.: Наука, 1987. – 284 с.

36. Китаев Н.Н. Групповые экспертные оценки / Н.Н. Китаев. – М.: Знание, 1975. – 64 с.

37. Пустыльник Е.И. Использование линейной модели для экстраполяции экспертных оценок / Е.И. Пустыльник, В.В. Сысоев, М.С. Чирко // Автоматизация проектирования. – М.: МДНТП, 1981. – с. 46-50.

38. Пустыльник Е. И. Об одном методе экстраполяции экспертных оценок / Е.И. Пустыльник, В.В. Сысоев, М.С. Чирко // Экономика и математические методы. – 1983. – вып. 4. – с. 716-717.

39. Rabiner L.R. A tutorial on hidden markov models and selected applications in speech recognition / L.R. Rabiner // In Proceedings of IEEE. – 1989. – vol. 77. – p. 257-286.

40. Романовский В.И. Дискретные цепи Маркова / В.И. Романовский. – Л.-М.: ОГИЗ, 1949. – 436 с.

41. Чабаненко Д.М. Алгоритм прогнозування фінансових часових рядів на основі складних ланцюгів / Д.М. Чабаненко // Вісник Черкаського університету. – 2010. – Вип. 173. – с. 90-102.

42. Горелик А.Л. Методы распознавания / А.Л. Горелик, В.А.Скрипкин. – М.: Высшая школа, 2004. – 262 с.

43. Фомин Я.А. Распознавание образов: теория и применения / Я.А. Фомин. – Изд. 2 – М.: ФАЗИС, 2012. – 429 с.

44. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – 2-е изд. – М.: Вильямс, 2006. – 1104 с.

45. Еремин Д. М. Искусственные нейронные сети в интеллектуальных системах управления / Д.М. Еремин, И.Б. Гарцеев. – М.: МИР, 2004. – 75 с.

46. Куффлер С. От нейрона к мозгу / С. Куффлер, Дж. Николс. – М.: Мир, 1979. – 440 с.

47. Алексеев В.И. Использование нейронных сетей с двухмерными слоями для распознавания образов / В.И. Алексеев, А.В.Максимов // Труды VIII Всероссийской конференции «Нейрокомпьютеры и их применение»: Сб. докл. – М., 2002. – с. 69-72.

48. Баусов Л.И. Нелинейное программирование / Л.И. Баусов. – М.: Финансовая академия при Правительстве РФ, 1996.

49. Тищенко А.К. Интеллектуальный анализ многомерных нелинейных временных рядов на основе гетерогенных нейронных сетей: А.К. Тищенко // автореф. дис. … канд. тех. наук: 05.13.23 / Тищенко Алексей Константинович. – Харьков, 2012. – 16 с.

50. Бутенко А.А. Обучение нейронной сети при помощи алгоритма фильтра Калмана / А.А. Бутенко // Труды VIII Всероссийской конференции «Нейрокомпьютеры и их применение»: Сб. докл., 2002. – с. 1120-1125.

51. Официальный сайт NVIDIA. [Электронный ресурс]. URL: http://www.nvidia.ru/page/home.html (дата обращения: 24.11.2014).

52. Клишин С.В. Применение МДЭ при анализе гравитационного движения гранулированного материала в сходящемся канале / С.В. Клишин // Горный информационно-аналитический бюллетень (научно-технический журнал). – 2009. – № 12. – С. 273-277.

53. Kalala J.T. Discrete element method modelling of liner wear in dry ball milling./ J.T. Kalala, M.H. Moys // The journal of the South African Institute of Mining and Metallurgy, November 2004. – p. 597-602.

54. Benn N. ACM SIGGRAPH Composium on Computer Animation / N. Benn, Y. Yu, P.J. Mucha // Particle-based simulation of granular materials, 2005.

55. Коэффициент вязкости. [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Coefficient_of_restitution (дата обращения: 29.11.2014).

56. Pöschel, T. Molecular dynamics of arbitrarily shaped granular particles / T. Pöschel, V. Buchholtz // Journal of Physics I France 5, 1995. – p. 1431-1455.

57. Pöschel, T. Static friction phenomena in granular materials: Coulomb law versus particle geometry / T. Pöschel, V. Buchholtz // Physical Review Letters, 1993. – 71. – 24. – p. 3963-3966.

58. Метод Эйлера. [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/Метод_Эйлера (дата обращения: 16.12.2014).

59. CUDA. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/CUDA (дата обращения: 18.12.2014).

60. PHP. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/CUDA (дата обращения: 25.12.2014).

61. Чугаев Р.Р. Метод круглоцилиндрических поверхностей при расчете устойчивости откосов / Р.Р Чугуев. – М.: Госэнергоиздат, 1963. – 144 с.

62. Линник Ю.В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений / Ю.В. Линник. – 2-е изд. – М.: Физматгиз, 1962. – 349 с.

63. Айвазян С.А. Прикладная статистика и эконометрика: Учебник для вузов / С.А. Айвазян, В.С. Мхитарян. – М.: ЮНИТИ, 1998. – 1022 с.

64. Астахова И. Ф. Системы искусственного интеллекта. Практический курс: учеб. пособие / И.Ф. Астахова – М.: БИНОМ. Лаборатория знаний, 2008. - 292 с.

65. Ротштейн А. П. Интеллектуальные технологии идентификации: нечёткая логика, генетические алгоритмы, нейронные сети / А. П. Ротштейн. – Винница: УНИВЕРСУМ-Винница, 1999. – 320 с.

66. Barricelli N.A. Numerical testing of evolution theories: Part I Theoretical introduction and basic tests / N.A. Barricelli / Acta Biotheoretica 16 (1-2). – 1962. – p. 69-98.

67. Barricelli N.A. Symbiogenetic evolution processes realized by artificial methods / N.A. Barricelli // Methodos. – 1962. – p. 143-182.

68. Фогель Л. Искусственный интеллект и эволюционное моделирование / Л. Фогель, А. Оуэнс, М. Уолш. – Пер. с англ. под ред. А.Г. Ивахненко. – М.: Мир, 1969. – 231 с.

69. Holland J.H. Adaptation in Natural and Artificial Systems / J.H. Holland. – Ann Arbor: The University of Michigan Press. – 1975.

70. Батищев Д.И. Генетические алгоритмы решения экстремальных задач: учеб. пособие / Д.И. Батищев. – Воронеж, 1995.

71. Батищев Д.И. Глобальная оптимизация с помощью эволюционно-генетических алгоритмов / Д.И. Батищев, Л.Н. Скидкина, Н.В. Трапезникова // Межвуз. сб. ВГТУ. – Воронеж, 1994.

72. Батищев Д.И. Генетический алгоритм для решения задач невыпуклой оптимизации / Д.И. Батищев, П.А. Гуляева, С.А. Исаев // Новые информационные технологии в науке, образовании и бизнесе: тез. докл. междунар. конф. – Гурзуф, 1997.

73. Whitley D. A genetic algorithm tutorial / D. Whitley // Statistics and Computing. – 1994. – vol. 4 – p. 65-85.

74. Курейчик, В.В. Концептуальная модель представления решений в генетических алгоритмах / В.В. Курейчик, П.В. Сороколетов // Известия ЮФУ. – 2008. – № 9. – с. 7-12.

75. Курейчик В.В. О правилах представления решений в эволюционных алгоритмах / В.В. Курейчик, С.И. Родзин // Известия ЮФУ. – 2010. – № 7. – с. 13-22.

76. Chakraborty U.K. An analysis of Gray versus binary encoding in genetic search / U.K. Chakraborty, C.Z. Janikow // Information Sciences. – 2003. – Vol. 156 – p. 253-269.

77. Паклин Н.Б. Адаптивные модели нечеткого вывода для идентификации нелинейных зависимостей в сложных системах: дис. ... канд. техн. наук: 05.13.18 / Н.Б. Паклин. – Ижевск, 2004. – 167 с.

78. Herrera F. Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis / F. Herrera, M. Lozano, J.L. Verdegay // Artificial Intelligence Review. – 1998. – vol. 12 – p. 265-319.

79. Eshelman L.J. Real-Coded Genetic Algorithms and Interval-Schemata / L.J. Eshelman, J.D. Schaffer // Foundations of Genetic Algorithms 2. – San Mateo: Morgan Kaufman Publishers, 1993. – p. 187-202.

Публикации автора в журнале, индексируемом в Agris

80. Коробкин Е.А. Приложение теории искусственного интеллекта в задачах моделирования устойчивости грунтового массива / И.Ф. Астахова, Е.А.Коробкин // Лесотехнический журнал. – 2015. – № 4. – С. 7-14.

Публикации автора в изданиях, рекомендованных ВАК

81. Коробкин Е.А. Применение технологии CUDA для симуляции частиц при параллельном программировании / Е.А. Коробкин, И.Ф. Астахова // Программные продукты и системы, 2013. – № 1 (101). – с. 146-150.

82. Коробкин Е.А. Разработка нечеткой модели прогнозирования устойчивости грунтового массива / Е.А.Коробкин, И.Ф.Астахова, А.И.Шашкин // Вестник Воронежского государственного университета. Серия Системный анализ и информационные технологии. − 2015. – № 1. − с. 98-106.

Свидетельства о регистрации программ

83. Коробкин Е.А. Разработка нечеткой модели прогнозирования устойчивости грунтового массива / Е.А. Коробкин, И.Ф. Астахова // Свидетельство о государственной регистрации программы для ЭВМ № 2015660382 от 30 сентября 2015 г.

84. Коробкин Е.А. Технология CUDA для метода дискретных элементов в параллельной среде/ Е.А. Коробкин // Свидетельство о государственной регистрации программы для ЭВМ № 2015661068 от 15 октября 2015 г.

Статьи и материалы конференций

85. Коробкин Е.А. Технология CUDA для метода дискретных элементов в параллельной среде / Е.А. Коробкин, И.Ф. Астахова // Совеменные информационные технологии и ИТ-образование: Сб. труд. YII межд. научно-практ. конф. – М.: ИН-ТУИТ.РУ, 2012. – с. 899-903.

86. Коробкин Е.А. Метод дискретных элементов в параллельной среде с использованием технологии CUDA / Е. А. Коробкин, И.Ф. Астахова, А.Н. Кальной // Актуальные проблемы прикладной математики, информатики и механики: Сб. труд. межд. конф. – Воронеж: Издательско-полиграфический центр ВГУ, 2012. – с. 25-26.

87. Коробкин Е.А. Принципы функционирования модуля прогнозирования коэффициента устойчивости грунтового массива, основанного на нечеткой логике/ Е.А. Коробкин // Теоретические и прикладные аспекты современной

науки: Сб. научн. тр. по мат. VII межд. научно-практ. конф. – Белгород: ИП Петрова М.Г., 2015. – Часть I. – с. 12-15.

88. Коробкин Е.А. Использование генетического алгоритма для настройки модели прогнозирования устойчивости грунтового массива / Е.А. Коробкин // Научное обозрение физико-математических и технических наук в XXI веке: XIII научно-практ. конф. – М: Международное научное объединение “Prospero”, 2015. – с. 23-27.

89. Коробкин Е.А. Использование генетического алгоритма для настройки модели прогнозирования устойчивости горной породы / Е.А. Коробкин, И.Ф. Астахова // Современные методы теории краевых задач: мат. межд. конф.: Воронежская весенняя математическая школа "Понтрягинские чтения – XXVI" – Воронеж: Изд. дом ВГУ, 2015. – с. 19-22.

90. Коробкин Е.А. Разработка модели прогнозирования на основе нечеткой логики с использованием генетического алгоритма для настройки / Е.А. Коробкин, И.Ф. Астахова // Актуальные проблемы прикладной математики, информатики и механики: Сб. труд. межд. Научно-техн. конф. – Воронеж: Научно-исследовательские публикации, 2015. – с. 214-218.

Приложение. Акт об использовании результатов диссертационной работы, свидетельство о государственной регистрации программы для ЭВМ

УТВЕРЖДАЮ
Директор
ООО «ОГНЕБОРЕЦ+СВ»
_____ Р.В. Хлопов
«19» 10 2015 г.

АКТ
об использовании результатов диссертационной работы Коробкина Е.А.
в ООО «ОГНЕБОРЕЦ+СВ»

Настоящим актом подтверждается, что в ООО «ОГНЕБОРЕЦ+СВ» используются следующие результаты диссертационной работы Коробкина Е.А. по разработке моделей и алгоритмов поведения сыпучих сред на основе графического процессора и нечеткой логики, представленной на соискание ученой степени кандидата наук.

1. Алгоритм, моделирующий динамику частиц.

Предложенный автором алгоритм позволяет моделировать равномерное распределение огнетушащего порошка гранулированного состава для огнетушителей и систем пожаротушения, размещенного на складах и промышленных объектах.

2. Алгоритм прогнозирования устойчивости грунтового массива.

Предложенный автором подход позволяет решать задачу распределения и отгрузки огнетушащего порошка в различные виды транспорта, рассчитывать и оценивать устойчивость порошковых насыпей при транспортировке, определения величины их давления на конструкции хранилищ.

Опыт работы специалистов компании ООО «ОГНЕБОРЕЦ+СВ» позволяет сделать вывод о том, что применение результатов диссертационного исследования Коробкина Е.А. в деятельности предприятия дает возможность адекватного описания поведения огнетушащего порошка гранулированного состава с использованием доступной на предприятии техники, что позволяет уменьшить стоимость и сроки принятия обоснованных решений.

Директор ООО «ОГНЕБОРЕЦ+СВ» Р.В. Хлопов

Подписи указанных сотрудников ООО «ОГНЕБОРЕЦ+СВ» заверяю:

Начальник отдела кадров

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015660382

Разработка нечеткой модели прогнозирования устойчивости грунтового массива

Правообладатель: ***федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Воронежский государственный университет» (RU)***

Авторы: ***Астахова Ирина Федоровна (RU),***
Коробкин Евгений Александрович (RU)

Заявка № **2015614122**
Дата поступления **19 мая 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 сентября 2015 г.***

Заместитель руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015661068

Технология CUDA для метода дискретных элементов в параллельной среде

Правообладатель: ***федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Воронежский государственный университет» (RU)***

Автор: ***Коробкин Евгений Александрович (RU)***

Заявка № **2015614121**

Дата поступления **19 мая 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***15 октября 2015 г.***

Заместитель руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий