ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Южно-Уральский государственный университет»
(национальный исследовательский университет)

На правах рукописи

Голодов Валентин Александрович

Интервальный подход к регуляризации неточно заданных систем линейных уравнений

05.13.17 – Теоретические основы информатики

Диссертация на соискание ученой степени
кандидата физико - математических наук

Научный руководитель
доктор физико - математических наук,
профессор А. В. Панюков

Челябинск
2014

Содержание

Введение

Система линейных алгебраических уравнений используется при математическом моделировании объектов различной природы. В частности, физических, химических и социально-экономических процессов, процессов управления. В ходе качественного анализа и/или использования математических моделей необходимо решать различные системы линейных алгебраических уравнений.

В данной работе предлагается считать моделью **неточно заданной системы** линейных алгебраических уравнений интервальную систему линейных алгебраических уравнений, т.е. СЛАУ с интервальными коэффициентами. Таким образом, интервальная неопределённость $a_{ij} \in [a_{ij} - \delta_{ij}, a_{ij} + \delta_{ij}]$ является следствием природы самой математической модели. В том случае, когда система линейных алгебраических уравнений возникает как результат промежуточных вычислений, например, в ходе качественного и/или количественного исследования математических моделей, интервальная неопределённость является следствием аналитических оценок приближённых величин, выступающих в роли коэффициентов.

На практике необходимо решать плохо обусловленные и даже вырожденные системы линейных алгебраических уравнений, неустойчивых к возмущениям и погрешностям входных данных. Для решения сильно вырожденных или несовместных СЛАУ применяют различные методы решения некорректных задач, например, псевдорешение по М.М. Лаврентьеву, регуляризация А.Н. Тихонова, матричной коррекции (И.И. Еремин, В.А. Горелик, В.И. Ерохин и др. [18–20]).

В данной работе предлагается подход к регуляризации исходной СЛАУ за счет её погружения в интервальную СЛАУ, данный подход назван **интервальной регуляризацией**. Классическим примером плохо обусловленной матрицы является матрица Гильберта.

При решении такого рода задач существенно проявляются недостатки многих пакетов прикладных программ. В настоящее время широко распространены заблуждения, порождающие ошибки в расчетах: (1) распространение свойства ассоциативности операций сложения и умножения в поле действительных чисел на конечное множество машинных «действительных»

чисел; (2) распространение свойства непрерывной зависимости от параметров решения системы, полученной после «эквивалентных» преобразований, на исходную систему. Это присутствует в популярных коммерческих пакетах «MatLab», «MathCad» и т.п., а также свободно распространяемом пакете «SciLab»©. Использование в параллельных вычислениях разного числа процессоров во многих случаях дает существенно различающиеся результаты, демонстрируя необходимость применения надежных, а в некоторых случаях доказательных вычислений (см. работы К.И. Бабенко [1–3], П.С. Панков [30], А.Н. Малышев [28] С.К. Годунов [9, 11, 12], Э.А. Бибердорф и Н.И. Попова [4, 5]).

Несмотря на богатую и длительную историю, результаты упомянутых исследований сравнительно недавно вышли за рамки учебников, монографий и решения конкретных прикладных задач, результатом стал «GALA–2.0 — пакет для решения задач линейной алгебры с гарантированной оценкой точности» [6]. Все алгоритмы приведённые в работах выше, а также многие упомянутые в работе У. Кулиш, Д. Рац, Р. Хаммер, М. Хокс [21] направлены на то, чтобы по известной погрешности в исходных данных в рамках предполагаемой точности проводимых вычислений дать оценки на погрешность результата и, соответственно, гарантированные оценки точности предлагаемого приближённого решения.

Общей особенностью является то, что сами авторы, зачастую, приводят простые примеры, на которых приводимые методы и алгоритмы оказываются бессильны в рамках двойной точности, обеспечиваемой стандартом IEEE-754-1985/754-2008. Существенным шагом вперед относительно бездоказательных вычислений является стопроцентное диагностирование данной ситуации.

Иной подход, весьма доступно изложенный, например, в работе М.М. Максимова применительно к разрешимости интегральных уравнений [27], диссертационной работе В.А. Шишкина, а также в работах А.В. Панюкова и М.И. Германено [31], состоит в применении рациональных вычислений и вычислений с произвольной точностью. Как подчеркивается, например, в работах В.П. Максимова [27] многие выкладки при данном подходе существенно зависят от точности, предоставляемой рациональным типом данных, и не могут быть выполнены стандартными вычислениями с двойной точностью. Получение гарантированного результата здесь требует вычислительных

и программных средств, точно оперирующих с рациональными числами. «... По этой причине реализация описанной схемы исследований требует значительных усилий и соответствующей квалификации. Как замечено в (Х.Д. Икрамов Численные методы линейной алгебры.), существует «контраст между простотой описания метода и сложностями грамотной его программной реализации»...» [27].

В данной работе предлагается новый теоретический подход к решению систем неточно заданных систем, как систем с интервальной неопределённостью исходных данных, позволяющий решать плохо обусловленные системы, неустойчивые к погрешностям во входных данных. Подход основан на погружении СЛАУ в интервальную СЛАУ, учитывающую существующие интервальные неопределенности в данных еще на этапе постановки задачи, и поиска решения из допускового множества решений интервальной СЛАУ. Вырожденные и не только системы линейных уравнений могут давать вырожденные интервальные системы, подход предлагает получение решения, за счет коррекции правой части системы.

Поиск решения СЛАУ с интервальной неопределённостью как точечного решения интервальной СЛАУ, принадлежащего допусковому множеству решений, обусловлен его структурой, обеспечивающей максимальную устойчивость среди всех возможных множеств решений интервальной СЛАУ. В рамках предлагаемого подхода требуется точное решение соответствующей задачи линейного программирования большой размерности.

В работе демонстрируется применение данного подхода к модели межотраслевого баланса (модели Леонтьева), к анализу бинарной смеси на основе спектроскопических данных методом Фирордта, а также указываются другие возможные отрасли применения подхода.

Реализованный программный комплекс использует безошибочные рациональные вычисления, проблема высокой вычислительной сложности использования точных дробно-рациональных вычислений за счет применения параллельных вычислительных технологий начиная с самого низкого уровня арифметических операций и заканчивая крупнозернистым параллелизмом на уровне решения возникающей задачи линейного программирования.

Ключом к разработке эффективных арифметических алгоритмов является реализация операции сравнения и проблема ускорения переносов/заемов

при реализации операций сложения/вычитания. Известные схемы ускоренного переноса хороши для использования на аппаратном уровне (hardware), что ограничивает разрядность операндов технологическими ограничениями. Распространение данных схем на программное обеспечение (software) оказывается нерациональным.

Требуется разработать новые масштабируемые алгоритмы выполнения операций сравнения, сложения/вычитания, выполняемые за константное время и не зависящие от разрядности операндов, а также эффективного алгоритма для операции умножения больших чисел.

Задача решения интервальной системы в данной работе сводится к решению соответствующей задачи линейного программирования. Эффективное решение задачи линейного программирования за счет параллельных вычислений является хорошо изученным вопросом (см. работы V.Y. Pan и J.H. Reif [58], Ju. Hall [53], Ю.Г. Евтушенко, А.И. Голиков, В.А. Гаранжа [38], А.В. Панюков и В.В. Горбик [35, 61, 62]).

Степень разработанности темы исследования. На сегодняшний день разработаны различные теоретические и практические подходы для гарантированного решения систем линейных алгебраических уравнений с условиях интервальной неопределенности исходных данных: П.С. Панков [30], А.Н. Малышев [28], С.К. Годунов [9, 11, 12], Э.А. Бибердорф и Н.И. Попова [4, 5], У. Кулиш, Д. Рац, Р. Хаммер, М. Хокс [21].

Для решения несовместных и плохо обусловленных систем были предложены различные подходы, например, псевдорешение СЛАУ М.М. Лаврентьев, регуляризация А.Н. Тихонова.

Интервальные системы линейных алгебраических уравнений рассматривались в работах J. Rohn, С.П. Шарого, И.А. Шарой [48, 64–66] и др., подробный обзор современного состояния исследований в данной области можно найти в [49]. Предложены методики для грубого исследования разрешимости системы [46, 47], а так же полного исследования разрешимости и коррекции исходных данных системы в случае ее несовместности путем оптимизации негладких выпуклых функционалов специального вида [48].

Целями диссертационной работы являются:

1) предложение нового подхода к регуляризации СЛАУ за счет погружения в ИСЛАУ и перехода к поиску точки из допускового множества решений ИСЛАУ,
2) развитие концепции псевдорешения интервальной системы линейных алгебраических уравнений и ее применение в математическом моделировании,
3) создание масштабируемого алгоритма для вычисления псевдорешения интервальной системы линейных алгебраических уравнений,
4) применение технологии GPGPU для развития программного обеспечения безошибочных дробно-рациональных вычислений для распределённых параллельных вычислительных систем с графическими процессорами (гетерогенных вычислительных систем),
5) применение разработанного параллельного программного обеспечения для вычисления псевдорешения интервальной системы.

В соответствии с поставленной целью в диссертационной работе **решаются следующие задачи**:

1) Разработка концепции псевдорешения для интервальных систем линейных алгебраических уравнений и ее применение в математическом моделировании.
2) Разработка и тестирование алгоритма численного нахождения псевдорешения интервальных систем линейных алгебраических уравнений.
3) Применение методов программирования в многопроцессорных и распределённых средах для реализации алгоритма нахождения псевдорешения.
4) Применение методов программирования графических процессоров для реализации программного комплекса поддержки рациональных вычислений.
5) Реализация библиотеки безошибочных вычислений для параллельных алгоритмов в гетерогенных вычислительных системах.
6) Реализация программного комплекса для эффективного численного нахождения псевдорешения интервальных систем линейных алгебраических уравнений в гетерогенной вычислительной системе за счет масштабируемого параллельного алгоритма.

Основными **объектами исследования** являются: система линейных алгебраических уравнений с интервальной неопределённостью, интервальная система линейных алгебраических уравнений, псевдорешение интервальной системы линейных алгебраических уравнений, методика вычисления псев-

дорешения интервальной системы, методы реализации дробно-рациональных вычислений без округления в распределённых вычислительных системах с графическими ускорителями, параллельный метод вычисления псевдорешения системы линейных алгебраических уравнений.

Методами исследования являются: математический анализ, алгебраические методы, анализ алгоритмов, численные методы, программирование, вычислительный эксперимент.

Научная новизна заключается: (1) в создании подхода к регуляризации неточно заданных систем за счет погружения в интервальную СЛАУ, в создании нового подхода к коррекции правой части системы в задаче о допусках для интервальной системы линейных алгебраических уравнений, учет интервальной неопределенности в исходных данных повышает адекватность математического моделирования, (2) в разработке и реализации параллельного программного обеспечения для эффективного решения поставленной задачи в распределённых вычислительных системах с графическими ускорителями, (3) в разработке и реализация программного комплекса для безошибочных дробно-рациональных вычислений в гетерогенных вычислительных системах, реализации программного комплекса для численного решения задач математического моделирования.

Теоретическая значимость. Предлагаемый подход сводит задачу решения неточно заданной системы линейных уравнений к задаче поиска точки из допускового множества решений интервальной системы, подобная техника применяется впервые, метод решения нахождения точки из допускового множества ИСЛАУ продолжает ряд исследований М.М. Лаврентьева, Н.А. Хлебалина, J. Rohn, O. Beaumont, B. Philippe позволяет, вместе с определением совместности задачи о допусках для интервальной системы, решать задачу коррекции правой части системы. Предлагаемый метод решения продолжает работы по применению техники линейного программирования в задачах интервального анализа. Коррекция задачи о допусках осуществляется за полиномиальное время. Расширение допустимых интервалов правой части системы позволяет предложить решение в том случае, когда допусковое множество интервальной системы пусто.

Комплекс программ для безошибочных дробно-рациональных вычислений продолжает и существенно развивает работы А.Н. Румянцева [27], библиотеки GMP [52] на техническую базу современных кластерных решений с графическими ускорителями. Разработаны параллельные масштабируемые алгоритмы для эффективного использования GPGPU ускорителей.

Практическая значимость. Предложенный метод решения неточно заданных систем линейных алгебраических уравнений может быть использован в исследовании с помощью средств вычислительной техники, информационных процессов, задачах обработки зашумленных данных, относящихся к области исследования специальности 05.13.17 – теоретические основы информатики. Программное обеспечение позволяет получить решения интервальных систем в рамках предлагаемого подхода. Реализованные алгоритмы адаптированы для использования в гетерогенных вычислительных кластерах с графическими процессорами и эффективно масштабируются. Программный комплекс может быть полезен для исследователей, которым необходимо использовать методы безошибочных вычислений. Реализованная функциональность позволяет решать некорректные задачи, неустойчивые к погрешностям входных данных, а также решать плохо обусловленные СЛАУ, востребованные в различных математических моделях, с абсолютной точностью.

Работа выполнена в рамках соглашения № 14.B37.21.0395 с Министерством образования и науки Российской Федерации от 06 августа 2012 года. Правообладателем разработанного в рамках данной модели программных комплексов [13–15]является ЮУрГУ .

Положения, выносимые на защиту.

√ Предложен новый подход к регуляризации СЛАУ за счет погружения в ИСЛАУ и перехода к поиску точки из допускового множества решений ИСЛАУ.

√ Предложен новый подход к поиску точки допускового из множества интервальной системы линейных алгебраических уравнений, позволяющий наряду с исследованием совместности задачи о допусках для ИСЛАУ проводить коррекцию правой части системы. Учет интервальной неопределённости в исходных данных повышает качество математического моделирования. Метод обладает рядом преимуществ для решения практических задач,

поскольку применим для решения сильновырожденных или несовместных ИСЛАУ и СЛАУ.

- √ Разработан и протестирован масштабируемый параллельный алгоритм для нахождения псевдорешения интервальной системы линейных алгебраических уравнений в распределённых гетерогенных вычислительных системах с применением точных дробно-рациональных типов данных. Эффективность реализованного численного метода достигнута за счет применения технологии крупно-зернистого параллелизма и библиотеки MPI.
- √ Разработан и протестирован комплекс программ для безошибочных дробно-рациональных вычислений. Выполнена реализация последовательных алгоритмов для базовых арифметических операций и их параллельных версий для распределённых вычислительных систем с GPU. Применение технологии GPGPU позволяет на порядок увеличить эффективность данного ПО по сравнению с последовательной версией. Разработан и реализован программный комплекс для численного решения задач математического моделирования.

Апробация результатов. Все результаты диссертационной работы, разработанные методы, алгоритмы и результаты вычислительных экспериментов докладывались и получили одобрение на следующих международных, всероссийских и внутривузовских конференциях:

1. Третья научная конференция аспирантов и докторантов ЮУрГУ. (г. Челябинск, 2011).
2. Алгоритмический анализ неустойчивых задач. Международная конференция, посвященная памяти В. К. Иванова (г. Екатеринбург, 2011).
3. Четвертая научная конференция аспирантов и докторантов ЮУрГУ. (г. Челябинск, 2012).
4. 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numeric (г. Новосибирск, 2012).
5. Параллельные вычислительные технологии (ПаВТ'2012) (г. Новосибирск, 2012).
6. Параллельные вычисления и задачи управления (PACO'2012). Шестая международная конференция (г. Москва, 2012).
7. Информационные технологии и системы.(оз. Банное, респ. Башкортостан, 2013).

8. Пятая научная конференция аспирантов и докторантов ЮУргУ. (г. Челябинск,2013).
9. GPU Technology Conference-2013 (San Jose, California, 2013).

Публикации. По теме работы опубликовано 15 работ (6 статей, 2 постера, тезисов конференци – 4, 3 свидетельства о государственной регистрации программы для ЭВМ). В их числе 2 статьи из перечня ведущих российских рецензируемых изданий [33, 34], а также 1 статья в рецензируемом международном научном журнале [60].

Структура работы. Во введении обосновывается актуальность темы диссертации, выделяются и формулируются цели и задачи исследования, описывается структурно-логическая схема диссертационной работы.

В первой главе детально рассматривается интервальная система линейных алгебраических уравнений, источники интервальной неопределенности дается обзор известных подходов к решению различных постановок интервальных задач, приводятся данные о свойствах решений, а также производится сопоставление подходов и сравнение сложности численного решения задач в рамках приводимых методов.

Во второй главе описывается предлагаемый численный метод решения ИСЛАУ, вводятся понятие и дается определение псевдорешения ИСЛАУ, доказывается его существование для любой ИСЛАУ. Приводится конструктивный способ поиска псевдорешения, описываются свойства, которыми оно обладает. Показывается необходимость применения точных дробно-рациональных вычислений для реализации предлагаемого численного метода нахождения псевдорешения, а также параллельных технологий `MPI` и `CUDA` для его эффективного вычисления.

В третьей главе рассматриваются проблемы использования аппаратных приближенных типов данных, которые реализованы в вычислительных процессорах и используются в программных кодах без учета погрешностей. Приводится краткий обзор библиотек позволяющих производить точные дробно-рациональные вычисления. Описывается разработанный программный комплекс для решения интервальный СЛАУ и методы, позволяющие осуществлять безошибочные дробно-рациональные вычисления на современных процессорах и графических ускорителях. Предлагаются методы адаптации разработанных компонентов библиотеки для эффективного проведения параллель-

ных расчетов методами MPI. Рассматриваемые во второй главе методы с применением безошибочной дробно-рациональной арифметики требуют больших объемов вычислений. Разработка параллельных алгоритмов для проведения базовых арифметических операций в параллельном режиме с использованием массового параллелизма графических ускорителей позволит сократить время решение задач за счет эффективного использования ресурса параллельности современных вычислительных систем.

В четвертой главе диссертации приводятся результаты численных экспериментов тестирования программного обеспечения для дробно-рациональных вычислений в гетерогенных вычислительных системах с графическими ускорителями. Строятся таблицы для выбора оптимальных параметров для параллельных алгоритмов базовых арифметических операций. Важными показателем для оценки эффективности реализации параллельных алгоритмов являются показатель ускорения, т.е. сокращение времени вычислений при введении дополнительных вычислителей, а также масштабируемость, то есть увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров [10].

На основе разработанных алгоритмов, а также методов для вычислений с применением безошибочных дробно-рациональных вычислений на графических ускорителях, адаптированных для распределенной гетерогенной среды, созданы программные комплексы «ISLinPSolutor 1.0» (для решения задачи линейного программирования) и «ISLAYSolutor 1.0» (для вычисления псевдорешения интервальной системы) с применением дробно-рационального типа данных. В данной главе приводятся вычислительные эксперименты с программным комплексом и описана схема работы с ним.

В заключении сформулированы основные результаты диссертации.

Благодарности. Автор выражает искреннюю и глубокую благодарность своему научному руководителю профессору Панюкову Анатолию Васильевичу за постоянное внимание и помощь на протяжении всей работы.

ГЛАВА 1

Интервальная неопределённость в математическом моделировании

1.1 Источники интервальной неопределенности

Работать с приближёнными числами, границами ошибок и интервалами значений приходится во многих случаях, ниже приводятся некоторые их них.

Представление чисел. При вычислениях, как правило, мы можем эффективно оперировать лишь объектами, которые имеют конечное описание (с конечной конструктивной сложностью). Например,

$$\frac{2}{3} \approx 0.66667, \sqrt{2} \approx 1.4142, \pi \approx 3.1415926. \tag{1.1}$$

Если для дробно-рациональных чисел возможны точные машинные представления [13, 16, 32, 52], то, например, вещественные числа точных аналогов в современных ЭВМ не имеют.

Тем самым уже в постановке вычислительной задачи допускается некоторая неизбежная ошибка представления из-за приближенного характера данных. Кроме того, теряется информация о характере ошибки приближения, какое именно приближение выбрано, с недостатком или с избытком.

Более правильно указывать границы этой ошибки, к примеру, путём уточнения, что все выписанные знаки верны и, таким образом, ошибка не превосходит половины единицы последнего разряда. Предпочтительный способ представления ошибки, состоит в том, чтобы дать наиболее узкие границы – нижнюю и верхнюю – для интересующей нас величины:

$$\frac{2}{3} \in [0.66666, 0.66667], \sqrt{2} \in [1.4142, 1.4143], \pi \in [3.1415926, 3.1415927]. \tag{1.2}$$

Частным случаем ошибок представления являются ошибки машинного представления чисел в формате с плавающей точкой IEEE-754-1985/754-2008 [55]. Многие десятичные числа (к примеру, 0.1) не имеют точного конечного представления в формате с плавающей точкой одинарной и двойной точности [55], с которыми оперируют современные цифровые ЭВМ. При

вводе подобных чисел в ЭВМ они неизбежно заменяются их некоторыми приближением.

Ошибки округления. При выполнении арифметических операций с десятичными числами результат часто не представим тем же числом десятичных знаков и, в некоторых ситуациях, должен быть округлён. Например, если мы ограничиваемся десятичной арифметикой с пятью значащими цифрами, то

$$12345/6789 \approx 1.8184, \tag{1.3}$$

однако включение

$$12345/6789 \in [1.8183, 1.8184] \tag{1.4}$$

является более корректным и даёт больше информации [49].

Замечание. Ошибки округления можно полностью исключить за счет точных вычислений.

Физические константы и результаты измерений. Чаще всего они известны неточно. К примеру, современные значения атомных весов некоторых химических элементов рекомендуется принимать интервальнозначными, например, для кислорода установлен интервал $[15.99903; 159977]$ [68]. Для некоторых физических констант серьезные источники указывают стандартное (среднеквадратичное) отклонение, например, для гравитационной постоянной G.

Замечание. Здесь интервальная неопределённость носит существенный характер, поскольку заложена в саму математическую модель реального процесса, её учет позволяет создать более тонкий инструмент исследования.

Допуски. Допуски (техн.) – допускаемые отклонения (т.е. интервал, прим. автора) числовой характеристики какого-либо параметра, например, размеров деталей машин и механизмов, физико-химические свойства материалов) от его номинального расчетного значения в соответствии с заданным классом точности [42].

Наиболее широко понятие допуска распространено в машиностроении, где допуски устанавливают для обеспечения необходимого качества и взаимозаменяемости изделий.

Замечание. Интервалы в такой математической модели являются существенной ее частью, переход к точечной постановке приводит к искажению модели.

Неопределённость в физических законах и явлениях. Ограниченные по величине неопределённости и неоднозначности естественно возникают при математическом моделировании многих механических, физических, химических и пр. явлений. Интересным примером является сила сухого трения, играющая важнейшую роль в технике. Хорошо известно, что её максимальная величина T_{max} пропорциональна силе, с которой прижаты соприкасающиеся поверхности. Но, вообще говоря, если движение одной поверхности по другой не наступило, то абсолютная величина силы трения покоя может принимать любое значение из интервала $[0, T_{max}]$. Аналогично коэффициент силы трения покоя может задаваться некоторым интервалом [43].

Замечание. Существенность, как и влияние интервального характера величин, должны задаваться экспертом, в целом, учет интервальной неопределённости позволяет построить более общую и точную модель.

Неопределённости в технических, экономических и пр. системах. Здесь имеются в виду величины, точные значения которых неизвестны или же могут изменяться. На протяжении нескольких столетий вплоть до второй половины XX века для их описания преимущественно использовалась теория вероятностей. Основоположником исследования ограниченных неопределённостей, для которых известны лишь границы изменения, является Л.В.Канторович, впервые сформулировавший идею их использования в работе [25] и наметивший основы математического аппарата для их обработки.

Показательный пример даёт понятие управления. Управление по смыслу подразумевает неоднозначность параметров объекта, которые можно выбирать из некоторого множества значений для достижения тех или иных желаемых целей управления. С другой стороны, даже цели управления и параметры функционирования объекта могут быть определены неточно. Эти соображения приводят к задаче так называемого робастного управления [49].

Замечание. Здесь интервальных характер входных данных и результата существенен, поэтому исследование имеет смысл проводить интервальными методами.

Внутриматематические потребности. Помимо прикладной (практической) значимости интервальных методов решения задач существует также ВНУТРИМАТЕМАТИЧЕСКИЕ ПОТРЕБНОСТИ, которые ощущаются в математике и

естественно приводят к необходимости допущения в различных ситуациях интервальных величин и оперирования ими как с частным случаем числовых множеств. Таковыми являются, например, РАСШИРЕНИЕ ПОНЯТИЯ ПРЕДЕЛА, ИНТЕРВАЛЬНОЕ ИНТЕГРИРОВАНИЕ, ТЕОРЕМЫ БРАУЭРА И МИРАНДЫ, ЗАДАЧИ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ, ПРЕДСТАВЛЕНИЕ ОСТАТОЧНЫХ ЧЛЕНОВ ПРИБЛИЖЁННЫХ ФОРМУЛ.

Замечание. Учет интервальной неопределённости позволяет провести численное исследование математической модели в виде доказательного эксперимента и повысить её качество.

1.2 Некорректные задачи

Рассмотрим операторное уравнение $Az = u$, A – линейный оператор, действующий из гильбертова пространства Z в гильбертово пространство U.

Французским математиком Ж. Адамаром были сформулированы следующие условия корректности постановки математических задач, примере записанного операторного уравнения. Задача решения операторного уравнения называется корректно поставленной (по Адамару), если выполнены следующие три условия:

- решение существует $\forall u \in U$;
- решение единственно;
- если $u_n \to u$, $Az_n = u_n$, $Az = u$, то $z_n \to z$.

Многочисленные обратные (в том числе и некорректные) задачи можно найти в различных областях физики. Так, астрофизик не может активно воздействовать на процессы, происходящие на далеких звездах и галактиках, ему приходится делать заключения о физических характеристиках весьма удаленных объектов по их косвенным проявлениям, доступным измерениям на Земле или вблизи Земли (на космических станциях). Прекрасные примеры некорректных задач можно найти в медицине, прежде всего, нужно отметить вычислительную (или компьютерную) томографию. Хорошо известны приложения некорректных задач в геофизике (на самом деле, легче и дешевле судить о том, что делается под поверхностью Земли, решая обратные задачи, чем заниматься бурением глубоких скважин), радиоастрономии, спектроскопии, ядерной физике и т.д., и т.п. [Спец. курс для аспирантов МГУ им. М.В.Ломоносова, Москва, Лектор: профессор Ягола А.Г., 21 с.].

Хорошо известным примером некорректно поставленной задачи является интегральное уравнение Фредгольма 1-го рода, когда оператор A имеет вид:

$$Az \equiv \int_a^b K(x,s)z(s)ds = u(x), x \in [c,d], \tag{1.5}$$

Ядро интегрального оператора $K(x,s)$ - функция, непрерывная по совокупности аргументов $x \in [c,d], s \in [a,b]$, а решение $z(s)$ - непрерывная на отрезке $[a,b]$ функция.

Приведем пример простой системы линейных алгебраических уравнений, для которой традиционные методы решения некорректных задач, такие как, например, нормальное псевдорешение по М.М. Лаврентьеву имеет недостатки.

$$\begin{cases} x+y=1 \\ x+y=1 \end{cases} \tag{1.6}$$

Нормальное псевдорешение СЛАУ есть $(x,y)^T = (1/2, 1/2)$, однако в случае минимальных колебаний значений коэффициентов, например,

$$\begin{cases} (1+\varepsilon)x+y=1 \\ x+y=1, \end{cases} \tag{1.7}$$

$\varepsilon > 0$, система имеет единственное точное решение $(x,y)^T = (0,1)$, и сходимости при $\varepsilon \to 0$ к нормальному псевдорешению нет.

Если же принять изначально интервальную постановку, то есть допустить колебание элемента a_{11} в закрытом интервале $[1, 1+\varepsilon]$, то, вычислив `псевдорешение интервальной СЛАУ` (см. ниже определение [1.2.1]), мы получим точку $(x,y)^T = (0,1)$, которая является точным решением для любой системы из семейства СЛАУ описываемых соотношениями (1.7), то есть является устойчивым к колебаниям коэффициентов в рамках заданных интервалов, тем самым можно устранить влияние возможных колебаний коэффициентов на результат.

В данной работе предлагается способ поиска решения подобных некорректных задач исходя из следующего определения

Определение 1.2.1. *Псевдорешениями интервальной системы линейных уравнений* $\boldsymbol{A}x = \boldsymbol{b}$ *назовем точки допускового множества системы* $\boldsymbol{A}x =$

$\boldsymbol{b}(z)$ *с расширенной правой частью* $\boldsymbol{b}(z) = \left[\underline{b} - zp, \overline{b} + zq\right]$, *где* p, q *– константные положительные векторы, определяющие характер расширения исходя из содержательного смысла задачи,* $z \geq 0$ *– параметр, отвечающий за величину расширения.*

Различные возможные интервальные постановки для системы (1.6) и их **псевдорешения интервальных систем** приведены ниже.

$$\begin{cases} [1.000; 1.100] * x_1 + [1.000; 1.000] * x_2 = [1.000; 1.000] \\ [1.000; 1.000] * x_1 + [1.000; 1.000] * x_2 = [1.000; 1.000] \end{cases}$$
$$x = (0.000, 1.000)^T \quad z^* = 0.000000000$$
$$b = ([1.000, 1.000], [1.000, 1.000])^T$$

$$\begin{cases} [0.990, 1.010] * x_1 + [1.000, 1.000] * x_2 = [1.000, 1.000] \\ [1.000, 1.000] * x_1 + [0.990, 1.010] * x_2 = [1.000, 1.000] \end{cases}$$
$$x = (0.500, 0.500)^T \quad z^* = 0.005000000$$
$$b = ([0.995; 1.005], [0.995; 1.005])^T$$

$$\begin{cases} [0.990, 1.010] * x_1 + [1.000, 1.000] * x_2 = [1.000, 1.000] \\ [0.990, 1.010] * x_1 + [0.990, 1.010] * x_2 = [1.000, 1.000] \end{cases}$$
$$x = (1.000, 0.000)^T \quad z^* = 0.010000000$$
$$b = ([0.990, 1.010], [0.990; 1.010])^T$$

$$\begin{cases} [0.990, 1.010] * x_1 + [0.990, 1.010] * x_2 = [1.000, 1.000] \\ [0.990, 1.010] * x_1 + [0.990, 1.010] * x_2 = [1.000, 1.000] \end{cases}$$
$$x = (1.000, 0.000)^T \quad z^* = 0.010000000$$
$$b = ([0.990, 1.010], [0.990; 1.010])^T$$

Для решения некорректных задач, в частности, вырожденных и плохо обусловленных СЛАУ, разработан очень эффективный прием, называемый регуляризацией (по А.Н Тихонову). В его основе лежит учет дополнительной априорной информации о структуре решения (векторе априорной оценки x_0), которая очень часто присутствует в практических случаях.

Задачу решения СЛАУ $Ax = b$ можно заменить задачей отыскания минимума функционала Тихонова:

$$\Omega(x, \lambda) = \|Ax - b\|^2 + \lambda\|x - x_0\|^2.$$

Здесь λ – малый положительный параметр регуляризации, который необходимо подобрать некоторым оптимальным способом. Можно показать, что задачу минимизации функционала Тихонова можно, в свою очередь, свести к решению другой СЛАУ:

$$(A^T A + \lambda I)x = A^T b + \lambda x_0. \tag{1.8}$$

Которая при $\lambda \to 0$ переходит в исходную плохо обусловленную систему, а при больших λ , будучи хорошо обусловленной, имеет решение x_0. Очевидно, оптимальным будет некоторое промежуточное значение λ, устанавливающее определенный компромисс между приемлемой обусловленностью и близостью к исходной задаче. Отметим, что регуляризационный подход сводит некорректную задачу к условно-корректной (по Тихонову) задаче отыскания решения системы (1.8), которое, в силу линейности задачи, является единственным и устойчивым. Задача выбора оптимального значения параметра может решаться исходя из различных соображений.

Определение [1.2.1] также содержит параметр параметр z, отвечающий за расширение интервалов в правой части системы. Значение параметра z имеет смысл выбирать минимизируя расширения вектора правой части системы, то есть самого значения z из определения [1.2.1].

1.3 Интервальные системы линейных алгебраических уравнений

Всюду далее в данной работе рассматривается интервальная система линейных алгебраических уравнений

$$\begin{cases} \boldsymbol{a}_{11}x_1 + \boldsymbol{a}_{12}x_2 + \ldots + \boldsymbol{a}_{1n}x_n = \boldsymbol{b}_1 \\ \boldsymbol{a}_{21}x_1 + \boldsymbol{a}_{22}x_2 + \ldots + \boldsymbol{a}_{2n}x_n = \boldsymbol{b}_2 \\ \vdots \qquad\quad \vdots \qquad\quad \ddots \qquad\quad \vdots \qquad\quad \vdots \\ \boldsymbol{a}_{m1}x_1 + \boldsymbol{a}_{m2}x_2 + \ldots + \boldsymbol{a}_{mn}x_n = \boldsymbol{b}_m \end{cases} \tag{1.9}$$

с интервалами $\boldsymbol{a}_{ij} = [\underline{a}_{ij}, \overline{a}_{ij}]$ и $\boldsymbol{b}_i = [\underline{b}_i, \overline{b}_i]$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, кратко,

$$\boldsymbol{A}x = \boldsymbol{b}. \tag{1.10}$$

Как и в случае систем общего вида, следующие хорошо изученные множества решений интервальных линейных систем —

- объединённое множество решений

$$\Xi_{uni}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\exists A \in \boldsymbol{A})(\exists b \in bsb)(Ax = b)\}, \tag{1.11}$$

- допусковое множество решений

$$\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(Ax = b)\}, \tag{1.12}$$

- управляемое множество решений

$$\Xi_{ctl}(\boldsymbol{A},\boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall b \in \boldsymbol{b})(\exists A \in \boldsymbol{A})(Ax = b)\}, \tag{1.13}$$

это крайние точки обширного семейства $2^{m(n+1)}$ всевозможных множеств AE-решений для интервальных линейных систем вида (1.9).

Четвертой крайней точкой этого семейства является множество

$$\Xi(\boldsymbol{A},\boldsymbol{b}) = \{x \in \mathbb{R}^n | (\forall A \in \boldsymbol{A})(\forall b \in \boldsymbol{b})(Ax = b)\}. \tag{1.14}$$

Его рассмотрение хотя и не бессмысленно, но по большей части бессодержательно, так как для уравнений с интервальными параметрами ненулевой ширины это множество решений по большей части пусто. Значительный интерес представляет изучение этого множества решений для интервальных неравенств.

«Универсальное» решение ИСЛАУ

Введено и исследовано Л.Т. Ащепковым и Д.В. Давыдовым, определяется как точка, обеспечивающая минимальное в некоторой норме расхождение правой части и образа по всем возможным точечным системам в рамках заданных интервалов.

Однако подход не учитывает положение интервалов в правой части системы относительно начала координат из-за чего «полоса разброса» $[-\varepsilon, +\varepsilon]$ такая что $|Ax - b| < \varepsilon$ для любых $A \in \boldsymbol{A}, b \in \boldsymbol{b}$ может содержать большой «зазор».

Нахождение «универсального» решения сводится его авторами к решению соответствующей задачи линейного программирования. Минимальная невязка ε, соответствующая «универсальному» решению дает информацию о минимальном симметричном относительно начала координат брусе, в котором могут лежать значения $|Ax - b|$, для $A \in \boldsymbol{A}, b \in \boldsymbol{b}$.

1.4 Линейная задача о допусках для ИСЛАУ

В этом пункте все внимание будет уделено *линейной задаче о допусках* и допусковому множеству решений интервальной линейной системы уравнений.

Задачей о допусках для ИСЛАУ (1.9) называется поиск решений принадлежащих допусковому множеству решений

$$\Xi_{tol}(\boldsymbol{A},\boldsymbol{b}) = \{x \in \mathbb{R}^n \mid (\forall A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(Ax = b)\}. \tag{1.15}$$

или в эквивалентной форме

$$\Xi_{tol}(\boldsymbol{A},\boldsymbol{b}) = \bigcap_{A\in\boldsymbol{A}} \{x \in \mathbb{R}^n \mid (\forall A \in \boldsymbol{A})(\exists b \in \boldsymbol{b})(Ax = b)\}. \tag{1.16}$$

В представлении (1.15) квантор всеобщности $\forall A \in \boldsymbol{A}$ эквивалентен теоретико-множественной операции пересечения $\bigcap$. Поэтому из (1.16) следует, что допусковое множество наименеее чувствительно среди всех других множеств решений ИСЛАУ к изменению интервальной матрицы системы $\boldsymbol{A}x = \boldsymbol{b}$, так как (1.16) не шире чем множество решений $\{x \in \mathbb{R}^n : (\exists b \in \boldsymbol{b})(Ax = b)\}$ самой «лучшей» и самой робастной (устойчивой) матрицы $A \in \boldsymbol{A}$. Хотя некоторые матрицы $A \in \boldsymbol{A}$ могут быть плохо обусловленными или вырожденными влияние «хороших» матриц обеспечивает ограниченность и оценки для Ξ_{tol}. Следовательно мы можем использовать допусковое множество как инструмент регуляризации для неустойчивых и плохо обусловленных систем, когда необходимо сгладить эффект изменения решения задачи, вызванный неточностями или возмущениями в данных. Подобный подход назван **интервальной регуляризацией**.

В качестве областей приложений, в которых напрямую возникает допусковое множество решений ИСЛАУ можно привести задачи из математической экономики, технологического проектирования, автоматического управления, исследования асимптотической устойчивости динамических систем, естественно-научных исследований.

Допусковое множество решений системы линейных алгебраических уравнений обладает следующими свойствами.

Предложение 1.4.1. *Допусковое множество решений интервальной линейной системы уравнений есть выпуклое многогранное множество в* $\mathbb{R}^n$.

Прямое описание допускового множества решений интервальной линейной системы уравнений, при котором выписываются все ограничивающие его гиперплоскости, является трудоёмким и практически бесполезным: его сложность растёт пропорционально $m \cdot 2^n$. Следовательно на практике прибегают

к оцениванию множества вместо его полного описания, в частности нахождению (по-возможности, большего) бруса, который содержится в допусковом множестве решений данной интервальной линейной системы уравнений.

Специфической особенностью задачи о допусках является возможность для допускового множества решений оказаться пустым даже для простых интервальных данных, например, $\Xi_{tol} = \varnothing$ для одномерного уравнения $[1,2]x = [2,3]$ или двумерной системы

$$\begin{pmatrix} [1,2] & [-1,1] \\ [-1,1] & [1,2] \end{pmatrix} x = \begin{pmatrix} [1,3] \\ [1,3] \end{pmatrix} \tag{1.17}$$

В подобных случаях будем линейная задача о допусках (ЛЗД) считается неразрешимой или несовместной, исследование разрешимости ЛЗД является отдельной подзадачей, которая автоматически решается предлагаемым в данной диссертационной работе подходом.

Один из подходов к полному описанию допускового множества решений дает следующая теорема.

Теорема 1.4.1 (И.А. Шарой о допусковом множестве решений)**.** *Точка $x \in \mathbb{R}^n$ принадлежит допусковому множеству решений интервальной линейной системы $\boldsymbol{A}x = \boldsymbol{b}$ тогда и только тогда, когда она является решением системы двусторонних линейных неравенств*

$$\begin{cases} \underline{b}_i \le ax \le \overline{b}_i \\ a \in \operatorname{vert} \boldsymbol{A}_i : \\ i = 1, \ldots, m, \end{cases} \tag{1.18}$$

где вектор-строки a пробегают всевозможные вершины интервальных строк матрицы $\boldsymbol{A}$. Количество неравенств в этой системе не превосходит суммы числа вершин во всех интервальных векторах $\operatorname{vert} A_i, i = 1, \ldots, m$, и, тем более, не превосходит $m \cdot 2^n$.

Решение систем линейных неравенств может быть выполнено различными способами, и один из наиболее популярных состоит в применении развитых методов линейного программирования. В стандартной форме задачи линейного программирования система линейных неравенств имеет специальный вид, требующий неотрицательности переменных. Очень удобен для пред-

ставления допускового множества решений ИСЛАУ в таком виде результат полученный И.Роном в [64].

Теорема 1.4.2 (теорема Рона о допусковом множестве решений)**.** *Точка $x \in \mathbb{R}^n$ принадлежит допусковому множеству решений интервальной линейной системы $\boldsymbol{A}x = \boldsymbol{b}$ тогда и только тогда, когда $x = x' - x''$, где n-векторы x' и x'' удовлетворяют системе линейных неравенств*

$$\begin{cases} \overline{\boldsymbol{A}}x' - \underline{\boldsymbol{A}}x'' \leq \overline{\boldsymbol{b}}, \\ -\overline{\boldsymbol{A}}x' + \underline{\boldsymbol{A}}x'' \leq -\underline{\boldsymbol{b}}, \\ x', x'' \geq 0. \end{cases} \tag{1.19}$$

Ограниченность допускового множества определяется по следующему критерию:

Теорема 1.4.3 (критерий неограниченности допускового множества)**.** *Непустое допусковое множество решений $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ интервальной линейной системы уравнений $\boldsymbol{A}x = \boldsymbol{b}$ неограничено тогда и только тогда, когда в матрице A существуют линейно зависимые точечные столбцы.*

Ниже будет рассмотрен вопрос о методах исследования разрешимости и коррекции задачи в случае пустого Ξ_{tol}.

1.4.1 Исследование пустоты допускового множества

Для исследования пустоты допускового множества существует ряд методов основанных как на эмпирических предпосылках, так и позволяющих исследовать Ξ_{tol} на пустоту в самом общем случае, см., например, [23].

Кратко упомянем эмпирические методики и чуть подробнее остановимся на методах работающих для всех ИСЛАУ:

Разрешимость некой «средней» системы $(\operatorname{mid} \boldsymbol{A})x = \operatorname{mid} \boldsymbol{b}$. Контрпримером может служить простейшая ИСЛАУ из одного уравнения с $\boldsymbol{A} = [-1, 2]$, $\boldsymbol{b} = [-2, 6]$. Здесь $\Xi_{tol} = [-1, 2]$, но решение «средней системы» $x = 4$.

Несколько более точный критерий дает следующая теорема.

Теорема 1.4.4. *Пусть в системе уравнений $\boldsymbol{A}x = \boldsymbol{x}$ интервальная $m \times n$-матрица $\boldsymbol{A}$ и интервальный m-вектор $\boldsymbol{b}$ таковы, что для некоторого $k \in \{1, 2, \ldots, m\}$ выполнены следующие условия:*

∘ $0 \notin \boldsymbol{b}_k$,

- $\max\{\chi(\boldsymbol{a}_{kj})|1 \le j \le n, a_{kj} \ne 0\} < \chi(\boldsymbol{b}_k)$,

Тогда допусковое множество решений $\Xi_{tol}(A, b)$ *пусто,*

доказательство см., например, в [49].

В то же время, невыполнение условий Теоремы [1.4.4] не обязательно влечёт разрешимость линейной задачи о допусках. Например, второе условие из формулировки теоремы неверно для системы

$$\begin{pmatrix} [1,2] & [-1,1] \\ [-1,1] & [1,2] \end{pmatrix} x = \begin{pmatrix} [1,3] \\ [1,3] \end{pmatrix} \tag{1.20}$$

Однако, её допусковое множество решений всё-таки пусто. Информацию о разрешимости задачи (1.9) для всех ИСЛАУ дают методы приведенные ниже.

Метод «распознающего» функционала

Метод «распознающего» функционала, предложенный С.П. Шарым [48], основан на максимизации функционала задаваемого следующим образом:

Предложение 1.4.2. *Пусть даны интервальная* $m \times n$*-матрица и интервальный* m*-вектор правой части* $\boldsymbol{b}$*, а выражением*

$$Tol(x) = Tol(x; \boldsymbol{A}, \boldsymbol{b}) = \min_{1 \le i \le m} \left\{ \operatorname{rad} \boldsymbol{b}_i - \left| \operatorname{mid} \boldsymbol{b}_i - \sum_{j=1}^{n} \boldsymbol{a}_{ij} x_j \right| \right\} \tag{1.21}$$

определяется функционал $Tol : \mathbb{R}^n \to \mathbb{R}$*. Тогда принадлежность* $x \in \Xi_{tol}(A, b)$ *равносильна* $Tol(x; A, b) \ge 0$*, т. е. допусковое множество решений интервальной линейной системы* $\boldsymbol{A}x = \boldsymbol{b}$ *есть множество уровня* $\{x \in \mathbb{R}^n | Tol(x; A, b) \ge 0\}$ *функционала Tol .*

Функционал обладает следующими свойствами:

- Функционал $Tol(x)$ вогнутый.
- Функционал $Tol(x)$ достигает конечного максимума на всём пространстве $\mathbb{R}^n$.
- Если $Tol(y; A, b) > 0$, то $y \in \operatorname{int} tol(A, b) \ne \varnothing$.

В силу указанных свойств функционала, алгоритм выяснения разрешимости задачи о допусках состоит в следующем: пусть $\tau = \operatorname{Arg\,max}_{x \in \mathbb{R}^n} Tol(x; \boldsymbol{A}, \boldsymbol{b})$, $T = Tol(\tau; \boldsymbol{A}, \boldsymbol{b})$, тогда

- если $T \geq 0$, то $\tau \in \Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) \neq \varnothing$, т. е. линейная задача о допусках для интервальной линейной системы $\boldsymbol{A}x = \boldsymbol{b}$ совместна и точка τ лежит в допусковом множестве решений;
- если $T > 0$, то $\tau \in \operatorname{int} \Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) \neq \varnothing$, и принадлежность точки τ допусковому множеству решений устойчива к малым возмущениям данных матрицы и правой части системы;
- если $T < 0$, то $\tau \in \operatorname{int} \Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) = \varnothing$, т. е. линейная задача о допусках для интервальной линейной системы $\boldsymbol{A}x = \boldsymbol{b}$ несовместна.

Теорема Рона

Одним из первых и важных результатов сводящих задачу выяснения непустоты допускового множества к задаче линейного программирования является теорема [1.4.2] (Рона) о допусковом множестве. Следует подчеркнуть, что этот результат показал возможность решения постановок интервальных задач неинтервальными методами, причем, результат применим для любых ИСЛАУ.

Для исследования непустоты $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ достаточно решить задачу совместности системы неравенств размерности $2n$, например, развитыми методами линейного программирования [22, 35].

На практике прибегают к оцениванию допускового множества, как правило максимальным в некотором смысле вписанным(внутреннее оценивание) или описанным(внешнее оценивание) брусом . Исследование методов построения оценок допускового множества $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ не входит в круг рассматриваемых в работе вопросов. Некоторые алгоритмы изложены в [49], там же приведены ссылки на работы по этой теме.

Отметим, что задача внутреннего оценивания может быть решена методами линейного программирования. Техника сведения задачи оценивания $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ максимальным брусом к задаче линейного программирования с соответствующими результатами экспериментов приводится в [51].

Формально-алгебраический подход

Предложенный и активно развиваемый Новосибирской научной школой, в частности, С.П. Шарым и его коллегами подход, требует реализации интервальной арифметики, например, можно воспользоваться `JInterval`.

Для ИСЛАУ с точечной абсолютно неособенной матрицей системы формальной решение единственно для любой интервальной правой части. Что касается интервальных систем уравнений с существенно интервальными матрицами, то для них единственность формальных решений является в настоящее время сравнительно малоисследованной. В рамках данной работы формальные решения не рассматривались.

1.4.2 Коррекция правой части системы

Помимо информации о разрешимости или неразрешимости ИСЛАУ важной является возможность провести коррекцию правой части системы, т.е. скорректировать вектор «ожиданий».

«Распознающий» функционал

Применение метода «распознающего» функционала описанного выше позволяет помимо выяснения разрешимости системы провести коррекцию системы. Легко заметить, что матрица $\boldsymbol{A}$ и середина правой части $\operatorname{mid}\boldsymbol{b}$ зафиксированы, то увеличение радиусов всех компонент вектора $\boldsymbol{b}$ на одну и ту же величину K приводит добавлению константы K к функционалу $Tol(x;\boldsymbol{A},\boldsymbol{b})$. Следовательно,

$$\max_{\mathbb{R}^n} Tol(x;\boldsymbol{A},\boldsymbol{b}+K\boldsymbol{e}) = K + \max_{\mathbb{R}^n} Tol(x;\boldsymbol{A},\boldsymbol{b}), \tag{1.22}$$

где $e = ([-1,1],\ldots,[-1,1])^T$.

Таким образом задачу можно сделать разрешимой за счет увеличения интервалов правой части на одну и ту же величину K.

Наиболее важный частный случай расширения интервалов правой части системы это обеспечение одинаковых относительных (пропорциональных абсолютным значениям) увеличение радиусов компонент правой части, для этого используется несколько модифицированный функционал.

$$Tol_0(x) = Tol(x;\boldsymbol{A},\boldsymbol{b}) = \min_{1\le i\le m}\left\{|\boldsymbol{b}|^{-1}\left(\operatorname{rad}\boldsymbol{b}_i - \left|\operatorname{mid}\boldsymbol{b}_i - \sum_{j=1}^{n}\boldsymbol{a}_{ij}x_j\right|\right)\right\} \tag{1.23}$$

для $|\boldsymbol{b_i}|\neq 0,\ i=1,\ldots,m$ для $|\boldsymbol{b_i}|\neq 0,\ i=1,\ldots,m$

Максимизация данного функционала дает информацию о необходимом расширении правой части, а также об имеющемся запасе устойчивости.

В [49] приводятся также соображения по коррекции матрицы системы $\boldsymbol{A}$, для обеспечения ее совместности при фиксированной правой части $\boldsymbol{b}$.

1.5 Примеры математических моделей приводящих к решению интервальных систем линейных алгебраических уравнений

Системы с интервальной неопределённостью возникают во многих математических и прикладных областях, при решении экономических задач, при вычислении значений собственных функций, в распознавании образов для анализа зашумленных изображений, в физических задачах, как линеаризованные постановки нелинейных задач, в химии при анализе состава смесей хемометрическими методами и многих других. Подробно будут рассмотрены пример из области экономики и пример анализа химического состава одним хемометрическим алгоритмом (методом Фирордта).

1.5.1 Линейное уравнение межотраслевого баланса

В качестве примера возникновения множеств AE-решений интервальных линейных систем на практике приведем задачу математического моделирования в экономике. Рассмотрим линейное уравнение межотраслевого экономического баланса

$$x = Ax + y, \tag{1.24}$$

т. е. классическое уравнение Леонтьева [29], в котором

- $x \in \mathbb{R}^n$ — вектор объёмов продукции по n отраслям,
- $y \in \mathbb{R}^n$ — вектор потребления в непроизводственной сфере по этим отраслям
- $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ — матрица коэффициентов прямых производственных затрат.

Разумно считать, что коэффициенты прямых производственных затрат заданы интервалами, т.е. $a_{ij} \in \boldsymbol{a}_{ij}$ и $\boldsymbol{A} = (\boldsymbol{a}_{ij}), i, j = 1, \ldots, n$, аналогично, требование на вектор конечного потребления y также естественно сформулировать в интервальной форме: как правило, любая ситуация, когда реальное потребление будет выдерживаться в пределах некоторого интервала $\boldsymbol{y} \in \mathbb{IR}^n$ является приемлемой.

В вещественном случае решение системы (1.24) относительно x позволяет спрогнозировать объёмы производства по отраслям, необходимые для получения запланированного конечного потребления y.

Поскольку в подавляющем большинстве практических случаев данные матрицы A представляют собой статистики, а значит имеют доверительный интервал, то естественно будет принять интервальный характер данных. Это позволит уменьшить затраты на предобработку данных, а также повысить достоверность результатов.

В интервальном случае вместо (1.24) получится уравнение

$$x = \boldsymbol{A}x + \boldsymbol{y} \tag{1.25}$$

Если же интервалы некоторых коэффициентов прямых производственных затрат представляют пределы их возможного управления, то задача определения объёмов производства x, которые обеспечивают конечное потребление $y \in \boldsymbol{y}$, приводит уже к рассмотрению множеств AE-решений интервальной линейной системы

$$(I - \boldsymbol{A})x = \boldsymbol{y}, \text{где } I \text{ – единичная матрица соответствующего размера.} \tag{1.26}$$

1.5.2 Определение компонентов бинарных смесей методом Фирордта

В спектрофотометрическом анализе смесей при наложении полос поглощения их компонентов используют метод Фирордта (МФ). Предварительно определяют молярные (или удельные) коэффициенты поглощения каждого компонента на аналитических длинах волн (АДВ). К сожалению, неизбежные погрешности при определении коэффициентов и при измерениях оптической плотности смеси ведут к тому, что точность метода невысока. Погрешности определения компонентов реальных смесей обычно не меньше 5-10% отн. и быстро растут при увеличении числа компонентов в смеси или соотношения их концентраций. Одна из причин неточности МФ - неаддитивность светопоглощения некоторых смесей. Это означает, что оптическая плотность смеси не равна сумме оптических плотностей компонентов, измеренных порознь в тех же условиях. Для смеси веществ X и Y абсолютное отклонение от аддитивности (ΔA) можно найти как разность $A_{x+y} - (A_x + A_y)$. Величина

ΔA зависит от условий проведения анализа, прежде всего от длины волны, на которой измеряют оптическую плотность. Очевидно, при разработке методик спектрофотометрического анализа смесей методом Фирордта выбирать АДВ необходимо с учетом возможных отклонений от аддитивности. Обычно такие отклонения рассматривают как следствие случайных погрешностей фотометрических измерений. Однако значения ΔA могут быть и хорошо воспроизводимыми, статистически значимыми. Причинами появления воспроизводимых отклонений должны быть постоянно действующие факторы, например химическое или сольватационное взаимодействие компонентов, влияние примесей, некорректная градуировка измерительного прибора (нелинейная функция преобразования) и др. Какова бы ни была причина возникновения подобных отклонений, применение МФ к анализу соответствующих смесей должно вести к систематическим погрешностям.

Введем следующие допущения (ограничения модели):

1. Компоненты смеси (X и Y) растворимы, их растворы стабильны во времени, поглощают излучение в используемой области спектра и хорошо подчиняются закону Бугера-Ламберта-Бера; ионная сила, pH и температура растворов постоянны.
2. Для анализа смеси X и Y используется набор, включающий лишь две АДВ, причем вклады каждого компонента в поглощение смеси при обеих АДВ статистически достоверны.
3. Случайные погрешности фотометрических измерений и коэффициентов поглощения на обеих АДВ пренебрежимо малы.

Анализ смеси методом Фирордта ведут, решая систему линейных уравнений:

$$\begin{cases} a_1 \cdot l \cdot c_x + a_2 \cdot l \cdot c_y = A_1, \\ b_1 \cdot l \cdot c_x + b_2 \cdot l \cdot c_y = A_2. \end{cases} \tag{1.27}$$

Здесь и далее a_1, a_2, b_1, b_2 – молярные коэффициенты поглощения веществ X и Y, соответственно при c_x и c_y –молярные концентрации X и Y в смеси, l – толщина поглощающего слоя, A_1 и A_2 – поглощение смеси на сответствующих длинах волн. Выбор АДВ должен обеспечивать хорошую обусловленность матрицы коэффициентов a_1, a_2, b_1, b_2, определитель матрицы должен быть как можно большим. В противном случае, при линейной или близкой к ней корреляции коэффициентов, определитель будет близок к нулю, и ре-

шение системы может привести к заведомо неверным или даже абсурдным результатам [7].

Для более точного определения величин a_1, a_2, b_1, b_2, которое проводят решая одномерные уравнения $\alpha \cdot l \cdot c = A$, соответствующие измерения проводят многократно. В том случае, когда проведение большого числа измерений затруднено или невозможно имеет смысл решать интервальную постановку задачи, когда величины a_1, a_2, b_1, b_2 представляют собой интервалы $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ которые включают известную погрешность измерительного прибора. Также можно рассматривать систему с числом уравнений превосходящим число неизвестных, за счет рассмотрения большего числа (АДВ).

1.5.3 Доказательные вычисления и интервалы

Доказательные вычисления — целенаправленные вычисления на ЭВМ, комбинируемые с аналитическими исследованиями, которые приводят к строгому установлению новых фактов и доказательству теорем [1].

Одним из часто применяемых методов доказательных вычислений являются численные методы с автоматической верификацией точности получаемых результатов. Часто доказательные вычисления строятся на основе интервального анализа, где вместо вещественных чисел рассматриваются интервалы, определяющие точность величин. Интервальный анализ широко применяется для вычислений с гарантируемой точностью.

Ведущим среди специализирующихся на интервальной тематике изданий, отражающих современное состояние исследований, является международный журнал «Reliable Computing».

Примеры доказательных вычислений в различных разделах математики:

- В теории чисел – опровержение гипотезы Л. Эйлера предполагавшего, что уравнение: $x_1^5 + x_2^5 + x_3^5 + x_4^5 = x_5^5$ не имеет решений в целых положительных числах. Решение найдено перебором на компьютере.
- В теории графов был достигнут один из наиболее известных успехов – решение проблемы четырёх красок. Это была первая крупная математическая теорема, доказанная с помощью компьютера [40].
- В математических задачах гидродинамики – доказана теорема о неустойчивости течения Пуазёйля при некоторых параметрах для спектральной задачи Орра-Зоммерфельда [3].

- Задача исследования разрешимости линейного интегрального уравнения Фредгольма и вариационных задач для квадратичных функционалов (под руководством В.П. Максимова).

Тщательный обзор применения и методики доказательных вычислений выполнен в [21]. Рассмотрены численные методы для решения задач: вычисления полиномов, дифференцирования функций, решения линейных и нелинейных систем уравнений, глобальная оптимизация, нахождение нулей комплексных полиномов, линейное программирование.

Изложенные численные методы приведены с *автоматической* верификацией точности полученных результатов и необходимо содержат математическое обоснование, описание алгоритма, полный текст соответствующей программы.

Нередко схема исследования доказательных вычислений требует решения СЛАУ с матрицей рациональных коэффициентов, а также исследования ИСЛАУ методами интервального анализа и совокупности этих методов.

Даже простая, непосредственная замена операций машинной арифметики на их интервальные варианты всегда обеспечивает надежные двусторонние оценки, однако, получаемые интервалы могут быть слишком широкими. Это явление было замечено еще в конце 60-х годов прошлого века и заставило многих специалистов по численному анализу *без достаточных оснований* отвергнуть эту полезную и необходимую технологию вычислений. С другой стороны, применяемая изолированно, высокая точность вычислений бывает неоправданно расточительна. Однако ее объединение с интервальными методами позволяет получать конечные результаты с высокой и гарантированной точностью [21].

Заметим в заключение, что проверка правильности алгоритма в общепринятом смысле вовсе не гарантирует корректность вычисленных с его помощью результатов. Чтобы убедиться в этом, достаточно вспомнить общеизвестный пример решения СЛАУ $Hx = b$ с матрицей Гильберта $H = \left(\frac{1}{i+j-1}\right)_{i,j=1,\ldots,n}$ и единичной правой частью $b = (1,\ldots,n)^T$. Аналитическое решение такой системы дает вектор x состоящий из целых чисел, тогда как численное решение в арифметике двойной точности IEEE-754-1985/754-2008, например, методом SVD-разложения системы размерности $n = 7$, уже не яв-

ляется целочисленным

$$x = (7.000000, -336.000000, 3780.000004, \\ -16800.000013, 34650.000023, -33264.000019, \\ 12012.000006)^T$$

а с ростом размерности системы n порядок ошибки растет экспоненциально. Этот простой пример демонстрирует разницу между математическим результатом и его численным аналогом, полученным на ЭВМ.

Естественно, сами по себе интервальные методы тоже не обеспечивают абсолютной гарантии, поскольку этап верификации может завершиться с неверным результатом из-за программной ошибки. Интервальная арифметика никогда не может избавить от необходимости проверять корректность алгоритма, компилятора, операционной системы и тестировать аппаратную часть. Интервальный метод может помочь выявить программную ошибку, если на этапе верификации будет достоверно установлено, что вычисленный результат явно ошибочен. Возникновение такой ситуации свидетельствует о необходимости дальнейшей отладки. Если верификация завершается успешно, то с высокой вероятностью можно утверждать, что комплекс из самой программы и среды ее выполнения обеспечил вычисление корректного результата *независимо* от того, проводилось ли доказательство правильности алгоритма.

Допустим, наконец, что с помощью верификации достоверность полученного решения не может быть ни подтверждена, ни опровергнута. Даже такой отрицательный результат приносит определенную пользу. Во-первых, он устанавливается автоматически, без необходимости для пользователя проводить какой-либо дополнительный анализ. Во-вторых, при написании программы на такой случай можно заранее предусмотреть использование альтернативного алгоритма или повтор вычислений с повышенной разрядностью.

Использование рациональной арифметики на этапе вычислений позволяет полностью исключить влияние ошибок округления и представления данных на результат. Использование интервального представления входных данных позволяет учесть как неточности, возникающие по ходу вычислений/измерений, так и существенно интервальный характер исходных величин.

1.6 Основные выводы

1) Погружение неточно заданной СЛАУ в интервальную СЛАУ и переход к поиску точки допускового множества ИСЛАУ позволяет получить устойчивое решение исходной СЛАУ, это обусловлено свойствами допускового множества.

2) Учет интервальной неопределённости повышает адекватность математических моделей.

3) Подход позволяет единообразно решать как задачи с изначально интервальным характером данных, так и те практические задачи, в которых правая часть b и элементы матрицы A, т. е. коэффициенты системы $Ax = b$, известны приближенно. В этих случаях вместо системы мы имеем дело с некоторой другой системой $\tilde{A}x = \tilde{b}$ такой, что $\|\tilde{A} - A\| \leq h$ $\|\tilde{b} - b\| \leq \delta$, где смысл норм обычно определяется характером задачи [45].

В следующей главе описывается подход к регуляризации **неточно заданной системы** линейных уравнений за счет погружения в интервальную СЛАУ и переходу к поиску точки из допускового множества решений ИСЛАУ. Подход позволяет как находить некоторую точку из допускового множества для совместной интервальной системы, так и корректировать правую часть системы в случае пустоты Ξ_{tol} и опять же предоставить точку из допускового множества скорректированной системы. Коррекция правой части системы позволяет решать некорректные задачи.

ГЛАВА 2

Псевдорешение интервальной системы. Существование. Алгоритм поиска

2.1 Новизна предлагаемого подхода

В данной главе описывается подход к решению неточно заданных систем линейных уравнений за счет погружения в интервальную систему и переходу к поиску точки из допускового множества интервальной системы. Предлагается метод поиска точки из допускового множества интервальной СЛАУ как решения задачи линейного программирования, тем самым вся технологическая цепочка получения результата не требует специальных пакетов интервальной арифметики и может быть реализован стандартными средствами языка C++. Подобный подход, использующий свойство устойчивости допускового множества к изменению матрицы интервальной системы, **назван интервальной регуляризацией и применяется впервые**. Метод поиска точки из допускового множества отличается от имеющихся наработок по применению техники линейного программирования к интервальным системам линейных уравнений тем, позволяет корректировать правую часть интервальной СЛАУ системы в случае её несовместности.

Возникающая задача линейного программирования требует применения аппарата безошибочных рациональных вычислений. Таким образом ликвидируется проблема накопления погрешностей в ходе численного решения задачи и гарантируется разрешимость поставленной задачи линейного программирования для любого набора входных данных. Разрешимость задачи о допусках для интервальной системы линейных уравнений с рациональными коэффициентами может служить целям доказательных вычислений.

Применение проверенного временем и детально исследованного симплекс-метода, достаточно простого в реализации и доказавшего свою эффективность [53, 61], упрощает кодирование и оптимизацию программного обеспечения воплощающего данную разработку.

2.2 Предпосылки рассматриваемого подхода

В соответствии с теоремой Рона [1.4.2] любая точка допускового множества может быть представлена в виде $x = x' - x''$, где x', x'' являются решением системы неравенств

$$\begin{cases} \overline{\boldsymbol{A}}x' - \underline{\boldsymbol{A}}x'' \leq \overline{\boldsymbol{b}}, \\ -\overline{\boldsymbol{A}}x' + \underline{\boldsymbol{A}}x'' \leq -\underline{\boldsymbol{b}}, \\ x', x'' \geq 0. \end{cases} \tag{2.1}$$

Во многих практических задачах система неравенств (2.1) оказывается плохо обусловленной или вообще несовместной. В этом случае по аналогии с псевдорешением СЛАУ М.М. Лаврентьева, работами [24,45] разумным представляется введение понятия *псевдорешения интервальных систем линейных алгебраических уравнений*.

Задачами данной диссертационной работы являются введение и исследование понятия псевдорешения интервальных систем линейных алгебраических уравнений и способы построения инструментальных программных средств поиска псевдорешения ИСЛАУ.

2.3 Понятие псевдорешения ИСЛАУ

Определение 2.3.1 ($\equiv$ 1.2.1)**.** *Псевдорешениями интервальной системы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ назовем точки допускового множества системы $\boldsymbol{A}x = \boldsymbol{b}(z)$ с расширенной правой частью $\boldsymbol{b}(z) = \left[\underline{b} - zp, \overline{b} + zq\right]$, где p, q – константные положительные векторы, определяющие характер расширения исходя из содержательного смысла задачи, $z \geq 0$ – параметр, отвечающий за величину расширения.*

Существование объекта из определения [2.3.1] подтверждает следующая теорема

Теорема 2.3.1. *Для любой системы интервальных уравнений $\boldsymbol{A}x = \boldsymbol{b}$ при всех $z \geq \max\{0, \max\limits_{i=1,\ldots,m} \underline{b}_i/p_i, -\min\limits_{i=1,\ldots m} \overline{b}_i/q_i\}$ множество $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}(z)) \neq \varnothing$.*

Доказательство. В соответствии с теоремой Рона условие $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}(z)) \neq \varnothing$ эквивалентно совместности системы линейных неравенств

$$\sum_{j=1}^{n} \left[\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-\right] \geq \underline{b}_i - z p_i, \quad i = 1, 2, \ldots, m, \tag{2.2}$$

$$\sum_{j=1}^{n} \left[\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-\right] \leq \overline{b}_i + z q_i, \quad i = 1, 2, \ldots, m, \tag{2.3}$$

$$x^+, x^- \geq 0. \tag{2.4}$$

Полагая в (2.2) – (2.4) $x^+ = x^- = 0$, получим

$$0 \geq \underline{b}_i - z p_i, \quad 0 \leq \overline{b}_i + z q_i, \quad i = 1, 2, \ldots, m,$$

что эквивалентно

$$z \geq \underline{b}_i / p_i \quad z \geq -\overline{b}_i / q_i \quad i = 1, 2, \ldots, m,$$

Таким образом, для всех $z \geq \max\{0, \max\limits_{i=1,\ldots,m} \underline{b}_i/p_i, -\min\limits_{i=1,\ldots m} \overline{b}_i/q_i\}$ имеет место включение $0 \in \Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}(z))$. Теорема доказана. □

2.4 Вопрос единственности псевдорешения

Рассмотрим вопрос единственности объекта из определения [2.3.1] объекта. Очевидно, что для случая совместной интервальной системы линейных уравнений, то есть, когда $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) \neq \varnothing$, псевдорешение может быть не единственным. Приведем соответствующий пример.

$$\begin{cases} [0,1]x_1 + [1,2]x_2 = [1,3] \\ [0,1]x_1 + [1,2]x_2 = [1,3] \end{cases} \tag{2.5}$$

Решением данной системы является множество точек Ξ_{tol} изображенное на Рисунок [2.1].

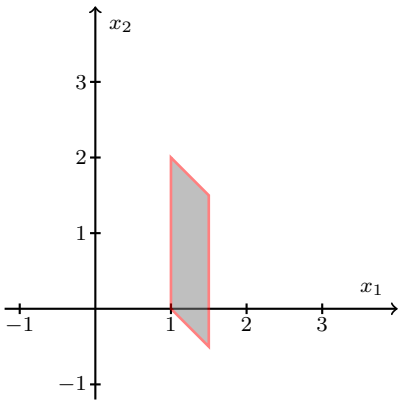

Рисунок 2.1. Допусковое множество системы (2.5)

Поскольку это множество непусто, то любая его точка является псевдорешением, соответствующим минимально возможному $z^* = 0$.

В данном случае имеет смысл ставить дополнительную задачу оптимизации для выбора того или иного решения среди возможных, либо ставить задачу описания и/или оценки множества решений.

Покажем влияние выбора параметров p, q, отвечающих за вид расширения интервалов правой части системы, на получаемое точечное решение. В случае на Рисунок [2.2] $p_i = q_i = 1,\ i = 1, \ldots, m$, то есть расширение всех интервалов одинаковое и равно $[\underline{b}_i - z^* \cdot 1, \overline{b}_i - z^* \cdot 1],\ i = 1, \ldots, m$.

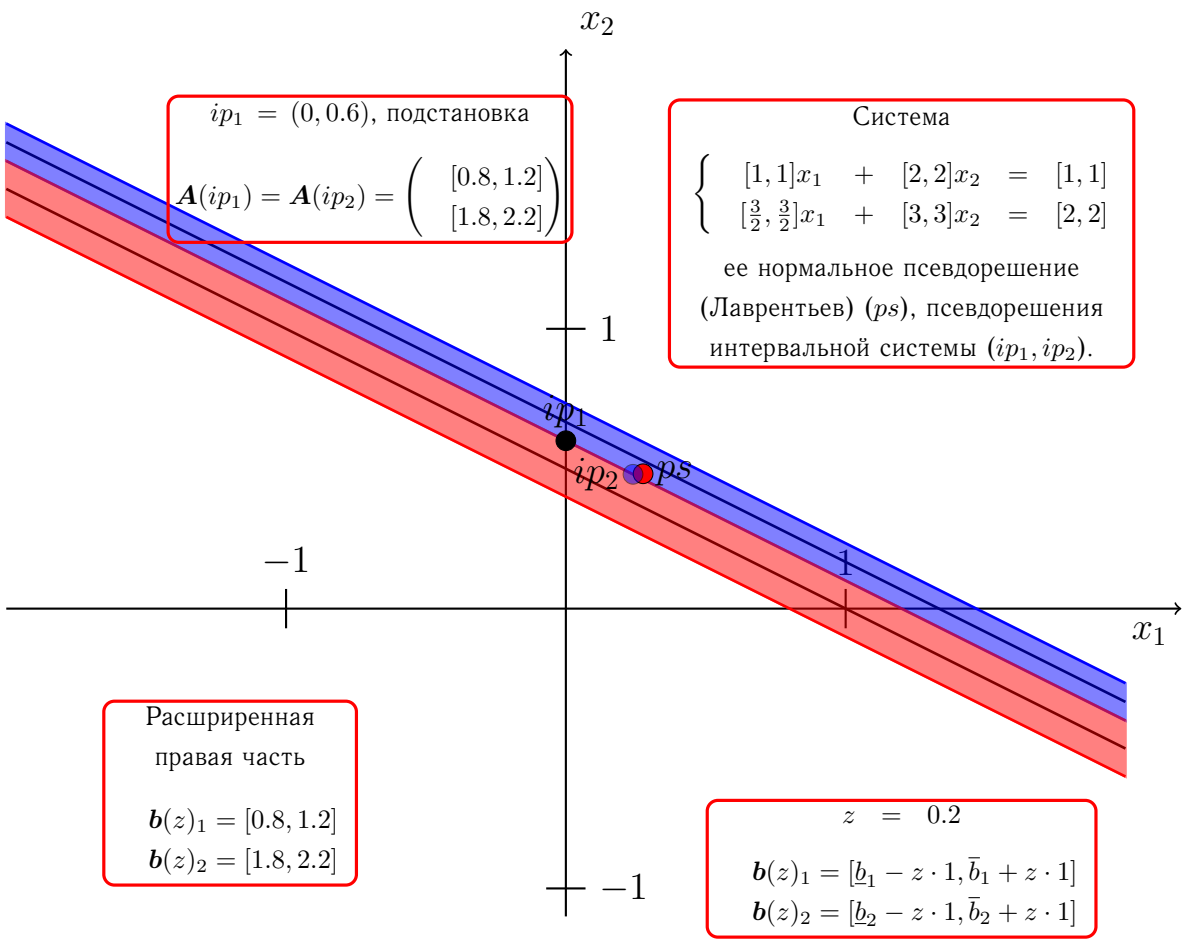

Рисунок 2.2. Пример несовместной системы. Нормальное псевдорешение (М.М. Лаврентьев) ps, псевдорешения интервальной системы ip_1, ip_2 (равномерное расширение интервалов)

Рассмотрим другой способ выбора векторов параметров p, q, отвечающих за вид расширения интервалов в правой части, – пропорциональное расширение Рисунок [2.3], в этом случае $p_i = |\underline{b}_i|, q_i = |\overline{b}_i|,\ i = 1, \ldots, m$.

2.5 Наилучшее возможное псевдорешение

В случае несовместности системы (1.10), то есть, когда $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) = \varnothing$, наибольший интерес представляет не всякое расширение правой части $\left[\underline{b} - zp, \overline{b} + zq\right]$, а наилучшее, то есть соответствующее $z^* = \inf\limits_{\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}(z)) \neq \varnothing} z$, расширение $\left[\underline{b} - z^*p, \overline{b} + z^*q\right]$.

Далее, если не оговорено иное, под псевдорешением ИСЛАУ будет подразумеваться *наилучшее возможное псевдорешение ИСЛАУ*.

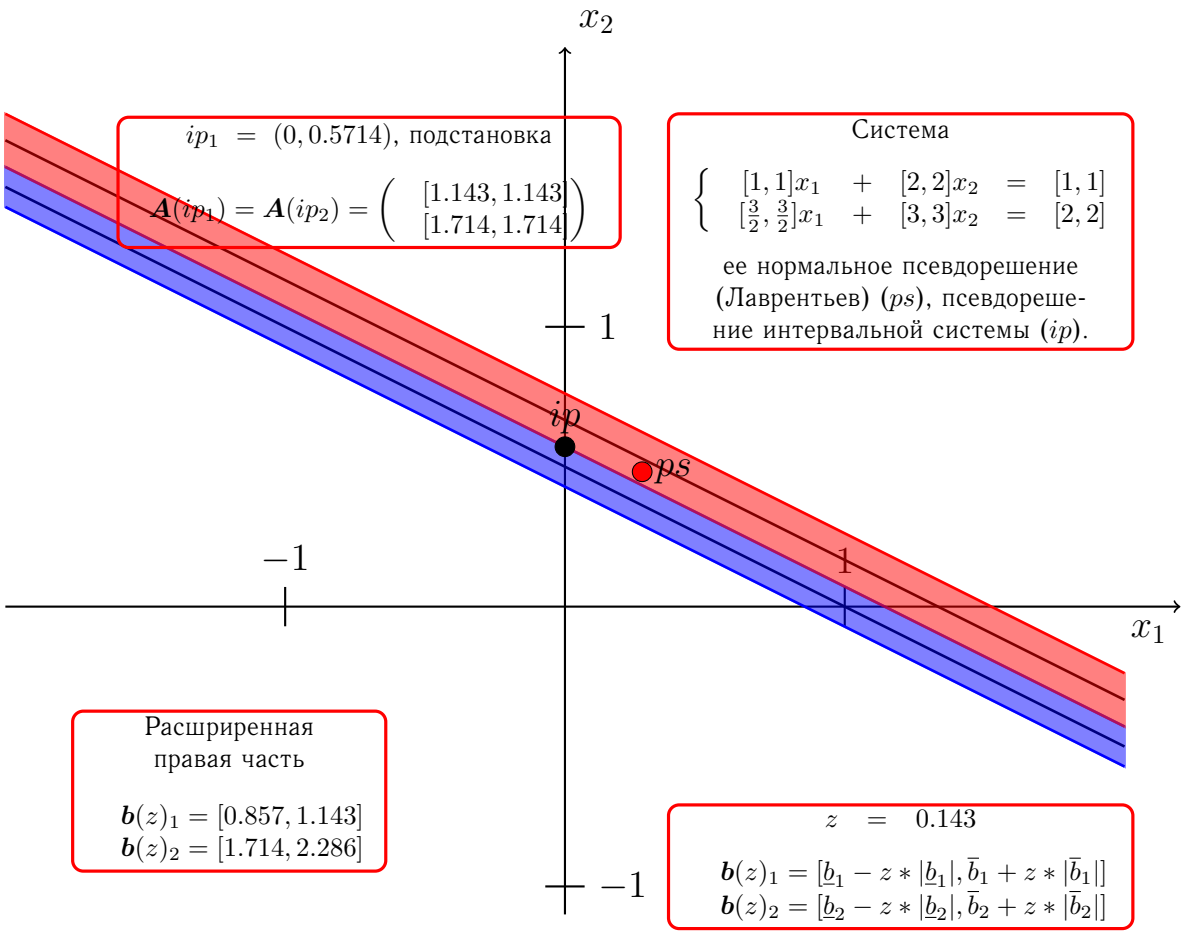

Рисунок 2.3. Пример несовместной системы. Нормальное псевдорешение (М.М. Лаврентьев) ps, псевдорешение интервальной системы ip (пропорциональное расширение интервалов)

2.5.1 Поиск наилучшего псевдорешения

Способ нахождения псевдорешения x^* интервальной системы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$, а также значения z^* параметра z дает теорема ниже.

Теорема 2.5.1. *Существует решение x^{+^*}, $x^{-^*} \in \mathbb{R}^n$, $z^* \in \mathbb{R}$ задачи линейного программирования*

$$z \to \min_{x^+,\ x^-,\ z}, \tag{2.6}$$

$$\sum_{j=1}^{n} (\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-) \geq \underline{b}_i - z p_i, \quad i = 1, 2, \ldots, m, \tag{2.7}$$

$$\sum_{j=1}^{n} (\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-) \leq \overline{b}_i + z q_i, \ i = 1, 2, \ldots, m, \tag{2.8}$$

$$x_j^+, x_j^-, z \geq 0, \quad j = 1, 2, \ldots, n, \tag{2.9}$$

при этом $x^* = x^{+^*} - x^{-^*}$ *является псевдорешением системы* $\boldsymbol{A}x = \boldsymbol{b}$.

Доказательство. Сначала докажем существование оптимального решения $x^{+^*},\ x^{-^*} \in \mathbb{R}^n,\ z^* \in \mathbb{R}$ задачи линейного программирования (2.6) – (2.9). Из теоремы 2.3.1 и теоремы Рона 1.4.2 следует, что множество решений рассматриваемой задачи не пусто. Задача, двойственная рассматриваемой, имеет вид

$$\sum_{i=1}^{m} \underline{b}_i y_{1i} - \sum_{i=1}^{m} \overline{b}_i y_{2i} \quad \to \max_{y_{1i}, y_{2i}}, \tag{2.10}$$

$$\sum_{i=1}^{m} \overline{a}_{ji} y_{1i} - \sum_{i=1}^{m} \underline{a}_{ji} y_{2i} \quad \leq \quad 0, \quad j = 1, 2, \ldots, n, \tag{2.11}$$

$$-\sum_{i=1}^{m} \overline{a}_{ji} y_{1i} + \sum_{i=1}^{m} \underline{a}_{ji} y_{2i} \quad \leq \quad 0, \ j = 1, 2, \ldots, n, \tag{2.12}$$

$$\sum_{i=1}^{m} p_i y_{1i} + \sum_{i=1}^{m} q_i y_{2i} \quad \leq \quad 1, \tag{2.13}$$

$$y_{1i}, y_{2i} \quad \geq \quad 0, \quad i = 1, 2, \ldots, m. \tag{2.14}$$

Легко заметить, что решение $y_{1i} = y_{2i} = 0, \quad i = 1, 2, \ldots, m$ является допустимым решением задачи (2.10) – (2.14). Таким образом, показано существование решений прямой (2.6) – (2.9), и двойственной (2.10) – (2.14)) задач линейного программирования. Из теоремы двойственности в линейном программировании следует существование у этих задач оптимальных решений [44].

Пусть $x^{+^*},\ x^{-^*},\ z^*$ оптимальное решение задачи (2.6) – (2.9). Из теоремы Рона следует, что $x^* = x^{+^*} - x^{-^*}$ является допустимым решением системы $\boldsymbol{A}x = \boldsymbol{b}(z^*)$. Из оптимальности z^* следует, что x^* является *наилучшим возможным псевдорешением* интервальной системы линейных уравнений $\boldsymbol{A}x = \boldsymbol{b}$. Теорема доказана.□

Таким образом, введенное понятие – *псевдорешение интервальной системы линейных уравнений* является вполне конструктивным и позволяет получать результаты решении любых интервальных систем в том числе несовместных, когда допусковое множество решений $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$ пусто.

Покажем, что для точечных невырожденных систем наилучшее возможное псевдорешение интервальной системы совпадает с обычным решением системы.

Утверждение 2.5.1. *Если точечная система* $Ax = b$ *невырождена, то ее решение и псевдорешение интервальной системы* $\boldsymbol{A}x = \boldsymbol{b}$*, где* $\boldsymbol{A} = [A, A]$*,* $\boldsymbol{b} = [b, b]$ *совпадают.*

Доказательство. В данном случае система неравенств из теоремы Рона о точках допускового множества

$$\begin{aligned}
\sum_{j=1}^{n} \left[\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-\right] &\geq \underline{b}_i - z p_i, \quad i = 1, 2, \ldots, m, \\
\sum_{j=1}^{n} \left[\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-\right] &\leq \overline{b}_i + z q_i, \quad i = 1, 2, \ldots, m, \\
x^+, x^- &\geq 0.
\end{aligned}$$

Примет вид

$$\sum_{j=1}^{n} \left[a_{ij} x_j^+ - a_{ij} x_j^-\right] \geq b_i - z p_i, \quad i = 1, 2, \ldots, m, \tag{2.15}$$

$$\sum_{j=1}^{n} \left[a_{ij} x_j^+ - a_{ij} x_j^-\right] \leq b_i + z q_i, \quad i = 1, 2, \ldots, m, \tag{2.16}$$

$$x^+, x^- \geq 0. \tag{2.17}$$

Допусковое множество системы $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}) \neq \varnothing$, следовательно, $z^* = 0$. Подставим $z^* = 0$ в (2.15) – (2.16), получим.

$$\begin{aligned}
\sum_{j=1}^{n} \left[a_{ij} x_j^+ - a_{ij} x_j^-\right] &\geq b_i, \quad i = 1, 2, \ldots, m, \\
\sum_{j=1}^{n} \left[a_{ij} x_j^+ - a_{ij} x_j^-\right] &\leq b_i, \quad i = 1, 2, \ldots, m, \\
x^+, x^- &\geq 0.
\end{aligned}$$

Затем, очевидно.

$$\sum_{j=1}^{n}\left[a_{ij}(x_j^+ - x_j^-)\right] = b_i, \quad i = 1, 2, \ldots, m, \tag{2.18}$$

$$x^+, x^- \geq 0. \tag{2.19}$$

Сделав замену $t_j = x_j^+ - x_j^- \quad t = 1, \ldots, n$ получим обычную точечную систему $At = b$, решение которой t^* существует и единственно. Возвращаясь обратно к переменным x_j^+, x_j^- получим единственное псевдорешение интервальной системы $\boldsymbol{A}x = \boldsymbol{b}$, с $\boldsymbol{A} = [A, A]$, $\boldsymbol{b} = [b, b]$. Утверждение доказано.□

В том случае когда $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b}(z^*))$ неограниченно, см. критерий [1.4.3] является конечным пересечением гиперполос отличным от точки (ответом на задачу линейного программирования является грань или ребро симплекса), имеет смысл ставить дополнительную оптимизационную задачу, которая может обеспечить единственность выбора оптимального псевдорешения интервальной системы. Для решения совместной задачи о допусках имеет смысл применять методы интервального анализа, дающие оценку $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$.

Полученная задача линейного программирования (2.6)-(2.9) и двойственная (2.10) – (2.14)) потенциально имеют высокую степень вырожденности, что будет приводить, при использовании приближенных вычислений, к зацикливанию симплекс-метода и погрешностям при применении других методов решения задачи линейного программирования.

Выходом является использование рациональных вычислений без округления [52, 59], подобные разработки были предложены в совместной работе А.В. Панюкова, М.И. Германенко, В.В. Горбика библиотеке классов – «Exact Computation» 2009 года. В работах А.В. Панюкова и В.В. Горбиком доказывается что количество требуемых на каждой итерации бит памяти полиномиально зависит от числа бит, достаточных для представления элемента матрицы исходных данных. Симплекс допускает эффективное распараллеливание, при этом эффективность (т.е. отношение ускорения к числу процессоров) в асимптотике близка к 100 % [36].

Вычислительные процедуры основанные на рациональной арифметике позволяют говорить о доказательном установлении разрешимости задачи о допусках для интервальной системы с рациональными коэффициентами. Вычислительная сложность применения безошибочных дробно-рациональных

вычислений является приемлимой для ресурса параллелизма современных гетерогенных вычислительных систем, например, систем кластерного типа с графическими ускорителями. На сегодняшний день ресурс параллелизма имеют не только многоядерные/многопроцессорные системы кластерного типа, но также бурно развивающийся сегмент графических ускорителей (GPU) и специализированных сопроцессоров с массовым параллелизмом. Таким образом численные методы должны использовать иерахрическую структуру параллелизма современных систем.

2.6 Выводы

В данной главе введено и описано понятие *псевдорешения интервальной системы линейных уравнений*, доказано существование псевдорешения интервальной системы для любой интервальной СЛАУ. Описаны свойства псевдорешения интервальной системы, приведен пример, когда псевдорешение интервальной системы является не единственным, вследствие неограниченности допускового множества решений ИСЛАУ. Выписана задача линейного программирования для нахождения псевдорешения, предложен подход к коррекции правой части системы для несовместной ИСЛАУ.

Предложен стратегический путь для эффективной реализации подхода, а именно использование ресурса параллелизма современных гетерогенных вычислительных систем.

В следующей главе будут рассмотрены разработка, реализация и применение соответствующего аппарата безошибочных дробно-рациональных вычислений.

ГЛАВА 3

Программный комплекс вычисления псевдорешения интервальной системы

Основным инструментом программного комплекса является модуль решения задачи линейного программирования с помощью безошибочных вычислений, не подверженных погрешностям округления и представления. Таким фундаментальным инструментом является разработанная и реализованная автором диссертационной работы библиотека поддержки точных дробно-рациональных вычислений в гетерогенных вычислительных средах с графическими ускорителями. Существующие программные продукты, использующие представление действительных чисел в формате с плавающей точкой IEEE-754-1985/754-2008, следует весьма осторожно применять для решения сложных практических задач, чувствительных к накоплению погрешностей округления.

Напомним, что требуемое количество памяти для хранения исходных данных задачи и промежуточных вычислений можно оценить заранее. Для широко известного симплекс-метода решения задачи линейного программирования на каждой итерации количество требуемых бит памяти не превосходит величины $4lm^4 + O(lm^3)$, где m – минимальная из размерностей задачи, l – число бит, достаточных для представления одного элемента матрицы исходных данных. Даже современные условно настольные ПК позволяют проводить решение задач размерностей порядка нескольких тысяч переменных и уравнений. Ключевым фактором является ресурс параллелизма системы, для производительных суперкомпьютеров объем оперативной памяти уже давно не является сдерживающим фактором, на одно «лезвие» (blade) может устанавливаться до терабайта оперативной памяти.

3.1 Параллельный алгоритм нахождения псевдорешения интервальной системы

Для малых и средних размерностей задачи параллельное программное обеспечение для дробно-рациональных вычислений позволяет достаточно быстро получать ответ для задач любой степени вырожденности с помощью

последовательного алгоритма симплекс-метода. Для больших размерностей задачи с количеством уравнений m и количеством переменных n много больше 100 рекомендуется использовать дополнительно параллельный алгоритм для реализации симплекс-метода. Таким образом реализация программного комплекса использует крупнозернистый и мелкозернистый параллелизм в рамках одного приложения.

Параллельный алгоритм симплекс-метода решения задачи линейного программирования описан в [35] и других работах. Для реализации был выбран параллельный алгоритм решения задачи линейного программирования с помощью симплекс-таблиц, код программы в целом следует схеме предложенной в [35].

Поскольку метод детально описан в работе [35] и не является разработкой автора диссертационного исследования, то рассмотрение его реализации выходит за рамки данной работы, отметим лишь, что в рамках распределённой вычислительной среды требуется обеспечение обмена данных симплекс-таблицы между процессами через коммуникационную сеть. Для этого необходимо обеспечить работу с рациональным типом данных методами пересылки сообщений библиотеки `MPI` в гетерогенной вычислительной системе. Новинкой с такой возможностью является разработанная автором диссертации библиотека поддержки рациональных вычислений в гетерогенных вычислительных системах.

3.2 Точные вычисления в существующих программных пакетах

Непонимание или пренебрежение ошибками представления и округления вводит в опасное заблуждение относительно процесса расчетов на ЭВМ: (1) распространение свойства ассоциативности операций сложения и умножения в поле действительных чисел на конечное множество машинных «действительных» чисел; (2) распространение свойства непрерывной зависимости от параметров решения системы, полученной после «эквивалентных» преобразований, на исходную систему.

Использование в параллельных вычислениях разного числа процессоров во многих случаях даёт существенно различающиеся результаты, демонстрируя необходимость применения надежных, а в некоторых случаях доказательных вычислений (см. работы К.И. Бабенко, Э.А. Бибердорф, В.А. Гаранжа,

С.К. Годунов, А.И. Голиков, Ю.Г. Евтушенко, А.Н. Малышев, П.С. Панков, Н.И. Попова, Ju. Hall, J.H. Reif [1, 4, 11, 12, 22, 38, 53, 58, 69]).

Одним из возможных выходов может быть применение символьных вычислений. Однако, потенциал имеющихся пакетов, поддерживающих символьные вычисления, не позволяет решать реальные проблемы математического и имитационного моделирования.

Другой подход – вычисления с произвольной точностью. Точные вычисления могут использовать многие востребованные программные пакеты, среди которых известные инженерные пакеты (MatLab©, SciLab©и др.)

Возможность использования точных вычислений представляет достаточно известная библиотека GMP(The GNU Multiple Precision Arithmetic Library) [52]. Библиотека распространяется под лицензией GNU LGPL, актуальная версия библиотеки GMP 6.0.1 доступна для загрузки с официального сайта проекта. Программный код оптимизирован под большинство существующих архитектур центральных процессоров, однако библиотека *не предоставляет* возможности использовать ее объекты в распределённых вычислениях. Надстройки именно над этим программным продуктом используют специализированные части пакетов MatLab©, SciLab©и пр. Объекты библиотеки изначально не предназначены для использования в распределённых параллельных вычислениях. Написание эффективной реализации оболочки для пересылок объектов в рамках коммуникационной среды является нетривиальной задачей для разработчика алгоритмов и представляет непреодолимое препятствие для большинства пользователей библиотеки.

Гораздо менее известный проект (gpuprec©) посвящен поддержке вычислений высокой точности на графических процессорах (Supporting High Precision on Graphics Processors). Данный проект не получил пока широкой огласки и является концептуальной попыткой портировать часть функциональности библиотеки GMP на новое оборудование.

3.3 Описание разработанного программного комплекса поддержки дробно-рациональных вычислений

Применение объектно-ориентированного подхода при создании разработанного программного комплекса позволяет использовать ряд преимуществ ООП. А именно: использовать понятия близкие к предметной области, инкап-

сулировать свойства и поведение объектов, создавать библиотеки и программы из имеющихся модулей, исключать избыточный код. Гибкая структура классов разработанного комплекса поддержки рациональных вычислений в полной мере использует описанные преимущества.

Одним из преимуществ, вытекающим из парадигмы ООП, является использование различных представлений чисел, а также переключение между представлениями, например, в зависимости от различных конфигураций оборудования. Например, уже известные алгоритмы Шёнхаге-Штрассена, Фюрера быстрого умножения длинных чисел оперируют комплексными чисел для вычисления произведения операндов, а алгоритм Тоома-Кука оперирует последовательностью бит.

Программа пользователя может быть запущена на кластере, где на одном из узлов используется представление чисел в десятичной системе счисления и последовательные алгоритмы арифметических операций над числами, а на другом узле моделью числа является последовательность бит и базовые арифметические операции реализованы параллельно. Одно представление может быть заменено на другое без необходимости проводить перекомпиляцию клиентских программ.

В рамках исследований был создан комплекс поддержки дробно-рациональных вычислений. Интерфейсные классы `overlong` и `rational` реализованы в объектно-ориентированной парадигме на языке C++, программный комплекс реализован в виде библиотеки классов `Exact Computation 2.0` [13] с возможностью динамической загрузки.

Классы библиотеки `Exact Computation 2.0` [13] позволяют производить безошибочные дробно-рациональные вычисления в любых современных операционных системах и имеют следующие характеристики.

- Класс `overlong` расширяет возможности целочисленных вычислений в гетерогенной вычислительной среде.
- Объектами класса `rational` являются обыкновенные дроби p/q, где p, q – объекты класса `overlong`.
- Объём памяти, занимаемый такими объектами, определяется значениями представляемых чисел, их диапазон ограничен только объёмом адресуемой памяти.

- Объекты класса `overlong` по умолчанию используют позиционную систему счисления по основанию 2^{32}.
- Для объектов классов `overlong` и `rational` определены все операторы, операции и бинарные отношения, используемые для стандартных числовых типов данных.
- Для объектов классов `overlong` и `rational` описаны все необходимые операции для пересылки в распределенной вычислительной среде с помощью функций библиотеки `MPI`.

Для облегчения сопровождения и модификации классов операции с памятью инкапсулированы в отдельный класс `MemHandle`, а выполнение базовых арифметических операций над числами полностью производится в рамках реализации класса `ArifRealization` (см. фрагмент листинга [3.1]).

Современная версия комплекса – библиотека классов «Exact Computation 2.0», зарегистрирована в Федеральной службе по интеллектуальной собственности. Основные особенности комплекса «Exact Computation 2.0» *с технологической точки зрения*:

- Для облегчения сопровождения и модификации классов операции с памятью инкапсулированы в отдельный класс `MemHandle`,
- Выполнение базовых арифметических операций над числами производится полностью в рамках реализации класса `ArifRealization`.
- Архитектура позволяет использовать несколько типов вычислителей в рамках одного приложения, исполняемого на узлах различной конфигурации. Для этого достаточно иметь соответствующую реализацию классов `MemHandle` и `ArifRealization` для каждого типа вычислителя.
- Для пользователя библиотеки все нюансы, связанные с различными вычислительными элементами, скрыты интерфейсом библиотеки, интерфейс независим от конкретной реализации.
- Для объектов классов `overlong` и `rational` определены все операторы, операции и бинарные отношения, используемые для стандартных числовых типов данных языка C++.

Поскольку увеличение показателей производительности микропроцессоров начиная с 2004 года происходит не за счет роста тактовых частот процессоров, а за счет увеличения количества параллельно работающих блоков(ядер, мультипроцессоров и т.п.) [57], то на первый план выходит раз-

```
class overlong {
 private: static ArifRealization realization;
 private: MemHandle mhandle;
 ...
 public: inline int32 size() const {return leng;}//length
 public: inline int32 sign() const {return sgn;} //sign
 ...
 //addition
 template<typename Type>friend const overlong operator+
  (const overlong &num,Type v)
                  {overlong rez(num); return (rez+=v);}
 friend const overlong operator+
                   (const overlong&,const overlong&);
 ...
}
```

Листинг 3.1. Фрагмент класса `overlong`

работка параллельного программного обеспечения на всех уровнях вычислительной системы, от микропроцессора, до узла кластера. Одной из наиболее перспективных и динамично развивающихся архитектур является архитектура графических ускорителей.

3.3.1 Особенности гетерогенных систем с GPU

Самые мощные гетерогенные системы для научных вычислений строятся на базе двух специализированных ускорителей: семействе продукции Intel®Xeon Phi™ [41] и графических ускорителях nVidia® [54]. Xeon Phi является высокоинтегрированным кластером, состоящим из процессоров архитектуры x86. CUDA устройства от Nvidia®отличаются массовым параллелизмом и большим количеством легковесных ядер, объединенных в блоки. Структура GPU обладает рядом преимуществ для решения вычислительных задач, однако требует дополнительных усилий при программировании.

Программная модель CUDA (Compute Unified Device Architecture) [26, 39, 63] включает описание вычислительного параллелизма и иерархической структуры памяти непосредственно в язык программирования. С точки зрения программного обеспечения, реализация CUDA представляет собой кроссплатформенную систему компиляции и исполнения программ, части которых одновременно работают на CPU и GPU. Платформа CUDA предназначена для разработки GPGPU-приложений без привязки к графическим API и поддерживается всеми GPU nVidia®, начиная с серии GeForce®8.

GPGPU – General-purpose graphics processing units, техника использования графического процессора видеокарты в приложениях для общих вычислений.

Концепция CUDA отводит GPU роль массивно-параллельного сопроцессора. В литературе о CUDA основная система к которой подключен GPU называется термином *хост* (host), а GPU термином *устройство* (device). CUDA-программа задействует как CPU, так и GPU, на CPU выполняется последовательная часть кода и подготовительные стадии для GPU-вычислений. Параллельные участки кода выполняются на GPU одновременно большим множеством *нитей* (threads). Важно отметить ряд принципиальных отличий нитей на GPU от обычных потоков на CPU:

- Нить на GPU чрезвычайно легковесна, ее контекст минимален, регистры распределены заранее;
- Для эффективного использования ресурсов GPU программе необходимо задействовать тысячи отдельных нитей, в то время как на многоядерном CPU максимальная эффективность обычно достигается при числе потоков, равном или в несколько раз большем количества ядер.

В целом работа нитей на GPU соответствует принципу SIMD, однако есть существенное различие. Только нити в пределах одной группы (для GPU архитектуры Fermi/Kepler - 32 нити), называемой *варпом* (warp) выполняются *физически одновременно*. Нити различных варпов могут находиться на разных стадиях выполнения программы. Такой метод обработки данных получил название SIMT (Single Instruction - Multiple Threads). Управление работой варпов производится на аппаратном уровне.

Новые возможности современных версий платформы CUDA демонстрируют тенденцию к постепенному превращению GPU в самодостаточное устройство, полностью замещающее обычный CPU за счет реализации некоторых системных вызовов (в терминологии GPU системными вызовами являются, например, `malloc` и `free`, реализованные в CUDA 3.2) и добавления облегченного энергоэффективного CPU-ядра в сам GPU (архитектура Maxwell®).

Важным преимуществом платформы CUDA является использование для программирования GPU языков высокого уровня. В настоящее время существуют компиляторы языков `C/C++` и `Fortran`. Эти языки расширяются

небольшим множеством новых конструкций: атрибуты функций и переменных, встроенные переменные и типы данных, оператор запуска ядра.

Ключевым приемом программирования для CUDA устройств является группировка множества нитей в блоки. На это есть две причины. Во-первых, очень мало параллельных алгоритмов имеют эффективную реализацию с помощью набора полностью независимых нитей: результат одной нити может зависеть от результата некоторых других, исходные данные нити могут частично совпадать с данными соседних. Во-вторых, размер одной выборки данных из глобальной памяти намного больше размера вещественного или целочисленного типа, т.е. одна выборка может покрыть запросы группы из нескольких нитей, работающих с подряд идущими в памяти элементами. В результате группировки нитей исходная задача распадается на независимые друг от друга подзадачи (блоки нитей) Рисунок [3.1]. Нити в рамках одного блока могут взаимодействовать, а запросы в память объединяются в рамках варпов. Разбиение нитей на варпы происходит независимо для каждого блока. Объединение в блоки является удачным компромиссом между необходимостью обеспечить взаимодействие нитей между собой и возможностью сделать соответствующую аппаратную логику эффективной и дешевой.

На время выполнения ядра каждый блок получает в распоряжение часть быстрой разделяемой памяти, которую могут совместно использовать все нити блока. Поскольку нити блока выполняются физически неодновременно, то необходим механизм синхронизации. Для этой цели в CUDA предусмотрен вызов __syncthreads(), который блокирует дальнейшее исполнение ядра до тех пор, пока все нити блока не войдут в эту функцию.

3.3.2 Дробно-рациональные вычисления в гетерогенных системах

Одной из главных задач при реализации библиотеки поддержки точных вычислений в гетерогенной среде являет обеспечение ее функциональности для любой конфигурации, от минимальной гомогенной, когда на вычислительном узле присутствует лишь один центральный процессор, до полного набора из нескольких центральных процессоров и 4–6 специализированных графических ускорителей.

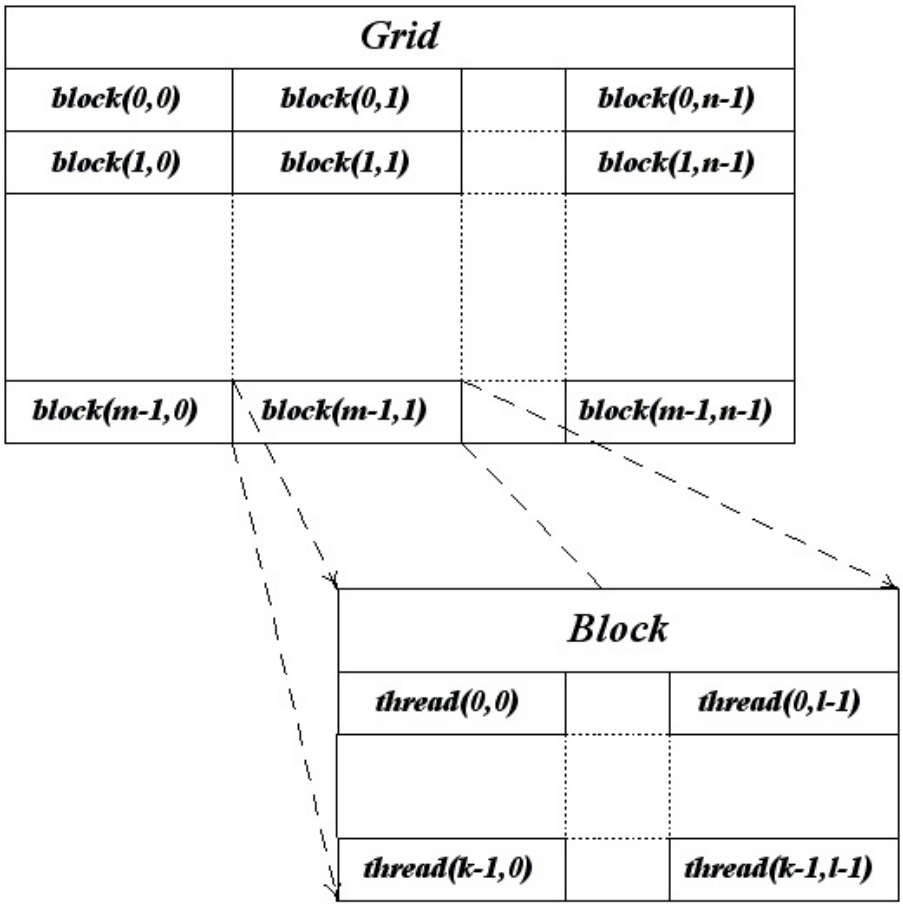

Рисунок 3.1. Разбиение задачи на набор независимых подзадач

Для этого необходима гибкая масштабируемая архитектура библиотеки, которая обеспечит возможность проверки конфигурации системы в момент запуска приложения. Такая структура библиотеки классов была разработана автором диссертации.

В рамках предшествующих данной работе исследований были созданы первые упрощенные версии классов `overlong` и `rational`, реализованные в объектно-ориентированной парадигме на языке C++ как библиотека классов `Exact Computation` [59].

Современная версия программного комплекса – библиотека классов«Exact Computation 2.0», зарегистрированная в Федеральной службе по интеллектуальной собственности, выполнена с учетом возможности использования рациональных чисел в гетерогенных распределённых компьютерных системах и имеет гибкую масштабируемую на различные архитектуры процессоров и сопроцессоров архитектуру [13].

Для более полноценного использования ресурсов современных процессорных архитектур классы `overlong` и `rational` хранят и оперируют чис-

лами по основанию 2^{32}, это позволяет эффективно использовать современные существующие 32-х и 64-х битные процессоры. Программный код операций оптимизирован для системы счисления по основанию степень числа два как того требует двоичный характер современных ЭВМ.

Работа с памятью также оптимизирована, поскольку в C++ нет автоматического сборщика мусора, то лишние операции перевыделения памяти приводят к фрагментации памяти и снижению быстродействия приложения в целом, краткое описание возможностей современных реализаций классов для процессоров архитектуры x86 и x64 дано в [17].

Ключевым фактором достижения эффективности вычислений является минимизация количества операций пересылки данных операндов от места хранения к арифметико-логическому устройству вычислителя. Хранение данных операндов организуется с учетом устройства, на котором производятся вычисления. При наличии в системе GPU nVidia®, все данные (разряды) чисел хранятся непосредственно на GPU, там же выполняются арифметические операции над числами, это снижает до минимума количество пересылок данных по PCI шине, копирование в основную оперативную память системы происходит лишь для вывода во внешний источник (поток вывода, файл). Для систем без GPU вычисления проводятся на процессоре, данные хранятся непосредственно в оперативной памяти системы. Описанную методику обеспечивает реализация специальных классов `MemHandle` и `ArifRealization`.

Техника разбиения класса `overlong`, на три части «интерфейс-память-арифметика», позволяет гибко использовать возможности вычислительной системы. Учтена возможность использовать современные гетерогенные вычислительные среды, например, системы с GPU ускорителями nVidia®. Язык OpenCL®дает возможность задействовать GPU от компании AMD®и встроенные графические ускорителей, например, Intel HD Graphics®, однако эксперименты показали большую трудоемкость процесса кодирования при малом ускорении и, следовательно, малом приросте эффективности приложений в целом.

Традиционные последовательные алгоритмы не позволяют в полной мере использовать массовый параллелизм и легковесные нити графического процессора. Параллельная архитектура и низкая тактовая частота ядер GPU

требуют разработки параллельных алгоритмов базовых арифметических операций.

Алгоритмы параллельного выполнения базовых арифметических операций, а также некоторые аспекты их реализации в гетерогенной среде описаны в работах автора диссертации и других работах [13, 17, 37]. Некоторые выдающиеся авторские разработки изложены в следующем параграфе.

3.3.3 Параллельные алгоритмы для базовых арифметических операций. Реализация на GPU nVidia®

Сравнение чисел

Алгоритм сравнения чисел разной длины или разного знака тривиален, поэтому остановимся подробнее на алгоритме сравнения двух чисел одинаковой длины в len разрядов, напомним, что разряд числа в современной версии библиотеки [13] составляет 2^{32}.

Традиционный последовательный алгоритм сравнения чисел не может быть распараллелен в силу характера информационных зависимостей [8]. Основной идеей, заложенной в параллельный алгоритм, является то, что сравнение чисел требует лишь узнать номер старшего разряда, в котором числа различны. В такой постановке задача уже может быть эффективно решена параллельными методами, можно сконструировать параллельный алгоритм [3.1], граф которого не будет иметь последовательных зависимостей длины более $\log_2 len$. Предлагаемый ниже алгоритм [3.1–3.2] для сравнения двух чисел длины len может эффективно выполняться вычислителем с массовым параллелизмом.

Алгоритм 3.1. Алгоритм операции параллельного сравнения. Часть 1.

Предусловия: Ядро запущено блоком в $BS = 1024$ нитей.
Постусловия: $outptr[bi]$ содержит результат редукции блока bi

Процедура KERNEL_CMP(Число $a[]$, Число $b[]$, Длина len, Указатель $*outptr$)
Выделить в shared памяти массив sh из $BS/32$ ячеек типа ***int*32**.
$ti \leftarrow$ относительный индекс нити в блоке
$gi \leftarrow$ глобальный индекс нити в сетке блоков
$bi \leftarrow$ индекс блока в сетке блоков
$v \leftarrow 0$
Если $gi < len$ **тогда** $v \leftarrow a[gi] \oplus b[gi]$
Конец условия
$b\_v \leftarrow \_\_ballot(v)$ ▷ каждый бит 32хбитного числа b_v равен значению выражения $v = 0$
Если нить имеет индекс 0 в $warp$ **тогда**
$wi \leftarrow \max_i\{(2^i \ \& \ v) \neq 0\}$ ▷ относительный индекс нити в $warp$
$ind \leftarrow$ абсолютный индекс нити wi в сетке блоков, или 0 при $v = 0$
$sh[ti/32] \leftarrow ind$
Конец условия
Синхронизация всех нитей в блоке
Если $ti < 16$ **тогда** ▷ редукция массива $data$
Если $data[ti] < data[ti + 16]$ **тогда** $data[ti] \leftarrow data[ti + 16]$
Конец условия
Если $data[ti] < data[ti + 8]$ **тогда** $data[ti] \leftarrow data[ti + 8]$
Конец условия
Если $data[ti] < data[ti + 4]$ **тогда** $data[ti] \leftarrow data[ti + 4]$
Конец условия
Если $data[ti] < data[ti + 2]$ **тогда** $data[ti] \leftarrow data[ti + 2]$
Конец условия
Если $data[ti] < data[ti + 1]$ **тогда** $data[ti] \leftarrow data[ti + 1]$
Конец условия
Конец условия
Если $ti = 0$ **тогда** ▷ результат редукции записать во внешний массив
$outptr[bi] \leftarrow data[0]$
Конец условия
Конец процедуры

Алгоритм 3.2. Алгоритм операции параллельного сравнения. Часть 2.

Предусловия: A, B имеют одинаковую длину L
Постусловия: Результат сравнения двух чисел

1: **Функция** CMP_GPU(Число $A[]$, Число $B[]$, Длина L)
2: $threads \leftarrow BS$
3: $blocks \leftarrow (L + threads - 1)/threads$
4: Выделить на GPU $blocks$ ячеек типа ***int32*** (d_ptr_outp) KERNEL_CMP$\lll blocks, threads \ggg$($A$, B, L, d_ptr_outp)
5: Дождаться завершения $Kernel\_cmp$
6: $ind \leftarrow$ REDUCTIONMAX($blocks, d\_ptr\_outp$)
7: Освободить память, связанную с указателем d_ptr_outp
8: $a \leftarrow A[ind]$, $b \leftarrow B[ind]$
9: **Возвратить** Результат сравнения a и b
10: **Конец функции**

Оценку количества шагов данного алгоритма дает утверждение [3.3.1].

Утверждение 3.3.1. *Алгоритм [3.1–3.2] требует менее* $\log_2 len - 3$ *параллельных шага, где* len *длина сравниваемых чисел. При этом быстрее выполнить операцию сравнения теоретически невозможно.*

Доказательство.

- Действия над разрядами исходных данных осуществляются полностью параллельно за 1 шаг.
- Редукция данных одного warp (32 нити) осуществляется полностью параллельно за 1 шаг.
- Редукция данных массива data (32 целых) осуществляется за 5 параллельных шагов, т.е. за $\log_2 32$ шагов.
- Редукция данных массива outptr($len/2^x$ целых) осуществляется за $\log_2 (len/2^x)$ параллельных шагов.

Итого сравнение требует $\log_2 len/2^x + ((\log_2 2^{x-5}) + 2) = \log_2 len - x + x - 3 = \log_2 len - 3$ параллельных шага, x выбирается исходя из ограничений на оборудование и соотношения $x >= 10$.□

Таким образом для алгоритма сравнения достигнута теоретическая нижняя оценка для алгоритмов с len входными данными [8] с применением метода сдваивания [50].

```
{
  int threadsPerBlock =  S_F_BLOCK_SIZE;
  int blocksPerGrid = (LA + threadsPerBlock - 1) / threadsPerBlock;
  uint32 d_ptr_outp=NULL;
  checkCudaErrors( cudaMalloc((void**)&d_ptr_outp, sizeof(uint32)*(blocksPerGrid))
      );
  KernelS_F_part_1_of_2 <<< blocksPerGrid, threadsPerBlock >>> (d_buffA, d_buffB,
     LA, d_ptr_outp);
  getLastCudaError("KERNEL KernelS_F_part_1_of_2 failed");
  cudaDeviceSynchronize();

  int ind=reductionMax(blocksPerGrid,d_ptr_outp);
  checkCudaErrors(cudaFree(d_ptr_outp));
  d_t alpha=0,beta=0;
  checkCudaErrors(cudaMemcpy(&alpha,&d_buffA[ind], sizeof(d_t),
     cudaMemcpyDeviceToHost) );
  checkCudaErrors(cudaMemcpy(&beta,&d_buffB[ind], sizeof(d_t),
     cudaMemcpyDeviceToHost) );
  return (alpha^beta)? ((alpha<beta) ? -1: 1) : 0;
}
```

Листинг 3.2. Вызов device кода операции сравнения

В листинге [3.2] представлен вызов GPU функций для осуществления сравнения на графическом ускорителе.

Ядро `KernelS_F_part_1_of_2` существенно использует особенности архитектуры CUDA, в частности, функцию __ballot [39], которая возвращает наибольший номер нити, для которой некоторый предикат дает значение истина.

Сложение (вычитание) чисел

Операция сложения (вычитания), строго говоря, не является параллельной, однако операции поразрядного сложения и распространения переносов из разных разрядов могут быть выполнено полностью параллельно. Операция поразрядного сложения аналогична операции сложения векторов, для операции распространения переносов верны следующие соображения. Если есть перенос из некоторого разряда j, то значение в данном разряде после переноса меньше либо равно $b-1+b-1-b=b-2$, где b – основание системы счисления и, следовательно, еще одного переноса из данного разряда произойти не может, даже в случае переноса из разряда $j-1$.

Решающим фактором, влияющим на эффективность параллельного сложения, является длина переноса, в случае переноса из самого младшего разряда вплоть до самого старшего, операция является чисто последовательной, но в случае сложения двух n-разрядных чисел из равномерного распределе-

```
__global__ void KernelS_F_part_1_of_2(const d_t* d_buffA, const d_t* d_buffB,int32
    LA,uint32 *d_ptr_outp)
{
  __shared__ uint32 data[(S_F_BLOCK_SIZE>>5)];
  int tid=threadIdx.x, g_i=blockDim.x*blockIdx.x + threadIdx.x;
  int v=0;
  if(g_i<LA)  v=d_buffA[g_i]^d_buffB[g_i];
  uint32 bit_values = __ballot(v);
  if(!(tid&31))//первая нить в варпе
  {
    v=32-__clz(bit_values);//ind_of_first_one(bit_values);
    //либо номер максимальной позиции в варпе где есть отличие, либо 0, если
        отличий нет
    data[tid>>5]=v+(g_i>>5)*(!!v);
  }
  __syncthreads();
  if (tid < 16){
  if(data[tid]<data[tid+16]) data[tid]=data[tid+16];
  if(data[tid]<data[tid+8]) data[tid]=data[tid+8];
  if(data[tid]<data[tid+4]) data[tid]=data[tid+4];
  if(data[tid]<data[tid+2]) data[tid]=data[tid+2];
  if(data[tid]<data[tid+1]) data[tid]=data[tid+1];
}
if(tid==0)  d_ptr_outp[blockIdx.x]=data[0];
}
```

Листинг 3.3. Вызов device кода операции сравнения

ния на интервале $[0, b^n]$ длина переноса, как правило, не более 2 разрядов [56].

Для обеспечения корректности вычислений требуется следить за тем, чтобы поразрядное сложение выполнялось до распространения переносов. То есть операция сложения будет выполняться в два этапа, на первом этапе производится поразрядное сложение и сохранение переносов, на втором этапе выполняется распространение переносов одновременно из всех разрядов. Этот подход требует дополнительной памяти для сохранения переносов, но прост в реализации.

Более эффективный по памяти подход предполагает распространение переносов внутри блоков, где возможна синхронизация и запоминание лишь переносов между блоками. Таким образом, сложение выполняется уже в три этапа, (1) поразрядное сложение, (2) распространение переносов внутри блоков нитей и сохранение переноса в следующую группу, (3) распространение переносов из разрядов, которые попали на границы групп.

Ниже приводится алгоритм [3.3–3.4] использующий описанную трех-этапную схему параллельного сложения чисел.

Алгоритм 3.3. Алгоритм операции параллельного сложения. Часть 1.

Предусловия: Ядро запущено блоком в $BS \geq 1024$ нитей. $LA >= LB$. $MAX\_DIG = 2^{32} - 1$,

Постусловия: $bCarry[]$ содержит переносы между блоками, $flag$ показывает наличие переносов между блоками

Процедура KERNEL_ADD_P1(Число $a[]$, Длина LA, Число $b[]$, Длина LB, Результат $c[]$, Переносы между блоками $bCarry$, Флаг $flag$)

$ti \leftarrow$ относительный индекс нити в сетке блоков
$gi \leftarrow$ глобальный индекс нити в сетке блоков
$bi \leftarrow$ индекс блока в сетке блоков
Если $gi \geq LA$ **тогда Возвратить**
Конец условия
Если $gi \geq LB$ **тогда**
$c[gi] \leftarrow a[gi]$
иначе
$t \leftarrow a[gi] + b[gi]$, $c[gi] \leftarrow t \mod MAX\_DIG$
Конец условия
Синхронизация
$t \leftarrow t > 0 \operatorname{div}(MAX\_DIGIT + 1)$, $i \leftarrow gi + 1$, $li \leftarrow ti + 1$
До тех пока $t \operatorname{and} i < LA$ **до**
Если $li = BS$ **тогда** ▷ последняя нить в блоке
$bCarry[bi] \leftarrow t$ ▷ перенос в следующий блок
$flag \leftarrow 1$,**Возвратить**
Конец условия
$t \leftarrow t + c[i]$
$c[gi] \leftarrow t \mod MAX\_DIG$, $t \leftarrow t \operatorname{div}(MAX\_DIG + 1)$
$i \leftarrow i + 1$, $li \leftarrow li + 1$
Конец цикла
Если $i = LA \operatorname{and} t > 0$ **тогда** ▷ перенос за пределы числа $a[]$
$bCarry[(\max_{bi} bi) + 1] \leftarrow 1$
Конец условия
Конец процедуры

Оценку количества параллельных шагов алгоритма [3.3–3.4] дает следующее утверждение.

Утверждение 3.3.2. *Алгоритм [3.3–3.4] в лучшем случае выполняется полностью параллельно за 1 параллельный шаг, в среднем случае за 3 параллельных шага, в худшем случае является полностью последовательным и выполняется за число шагов равное длине большего из операндов.*

Доказательство.

Алгоритм 3.4. Алгоритм операции параллельного сложения. Часть 2.

Предусловия:
Постусловия: Переносы выполнены
Процедура KERNEL_ADD_P2(Результат сложения $c[]$,Переносы между блоками $bCarry[]$, Размер блока BS, Длина массива L, Длина числа LA)
$gi \leftarrow$ глобальный индекс нити в сетке блоков
$i \leftarrow$ индекс разряда из которого есть перенос
Если $gi \geq L$ **тогда Возвратить**
Конец условия
$t \leftarrow bCarry[i]$
До тех пока $t > 0 \,\text{and}\, i < LA$ **до**
$t \leftarrow t + c[i]$
$c[gi] \leftarrow t \mod MAX\_DIG$, $t \leftarrow t \,\text{div}(MAX\_DIG + 1)$
$i \leftarrow i + 1$
Конец цикла
Если $i = LA \,\text{and}\, t$ **тогда** ▷ перенос за пределы числа $a[]$
$bCarry[L] \leftarrow 1$
Конец условия
Конец процедуры

- Действия над разрядами исходных данных осуществляются полностью параллельно за 1 шаг. Если переносов из разрядов нет, то на этом алгоритм завершается.
- В случае сложения двух n-разрядных чисел из равномерного распределения на интервале $[0, b^n]$, b – основание системы счисления, средняя длина максимального переноса не превосходит 2 разрядов [56]. Все переносы могут быть выполнены независимо, поскольку, если произошел перенос из некоторого разряда $c[j]$, то $c[j] <= b - 1 + b - 1 - b = b - 2$ и перенос из данного разряда уже невозможен. Тогда на распространение переносов потребуется не более 2 параллельных шагов. Итого алгоритм будет завершен за 3 параллельных шага.
- В худшем случае имеется перенос длины равной длине наибольшего числа и алгоритм является полностью последовательным.

□

Этапы операции сложения длинных чисел на GPU: параллельное сложение разрядов, синхронизация, параллельное распространение переносов из разрядов представлены в листинге ниже. Особенность архитектуры GPU, а

```
__global__ void DNumAdd_part1
(d_t *A,int32 LA,d_t *B,int32 LB, d_t *C,d_t *bGCarry,int32 *f){
 int32 gId=blockDim.x*blockIdx.x + threadIdx.x; int64 tmp=0;
 if (gId >= LA) return; //bound check
 if(gId>=LB) C[gId]=A[gId];
 else{tmp=(int64)A[gId]+(int64)B[gId]; C[gId]=tmp&MAX_DIGIT;}
 __syncthreads();//carry propagation in the block
 int32 lId=threadIdx.x+1,i=gId+1,gS=blockDim.x;
 for(tmp>>=BIT_IN_DIGIT;tmp && i<LA;lId++,i++){
  if(lId==gS){bGCarry[blockIdx.x]=tmp; *f=1; return;}
  tmp+=(int64)C[i]; C[i]=tmp&MAX_DIGIT;  tmp>>=BIT_IN_DIGIT;
 }if(i==LA && tmp) {bGCarry[Lcarry]=1;}
}
```

Листинг 3.4. Поразрядное сложение

```
__global__ void DNumAdd_part2
(d_t *C, d_t *bGCarry, int32 gS, uint32 Lcarry,uint32 LA){
 //carry propagation between blocks
 int gId = blockDim.x*blockIdx.x + threadIdx.x,i=(gId+1)*gS;
 if (gId >= Lcarry) return;
 uint64 tmp=(uint64)bGCarry[gId];
 for(;tmp && i<LA;i++){
  tmp+=(uint64)C[i];
  C[i]=tmp&MAX_DIGIT;
  tmp>>=BIT_IN_DIGIT;
 }
 if (i==LA && tmp) {bGCarry[Lcarry]=1;}
}
```

Листинг 3.5. Параллельное распространение переносов

именно, выполнение нитей *блоками* и отсутствие синхронизации между блоками требует сохранения «пограничных» переносов во временном массиве (за это отвечает строка номер 16 в листинге [3.4]. (*DNumAdd_part2*) (листинг [3.5]).

Установка параметров и запуск счета на графическом ускорителе представлены на листинге [3.6]. Этот код выполняется на стороне CPU, или, так называемой, стороне host.

Операция вычитания реализуется аналогичным кодом, это возможно, если уменьшаемое строго больше вычитаемого. Код претерпевает незначительные изменения и основное внимание следует уделить определению количества значащих цифр разности.

Умножение чисел

Операция умножения одна из наиболее затратных по времени. Разработка эффективной реализации операции умножения позволяет также эф-

```
void ArifRealization::add(const d_t *A,int32 LA,
     const d_t *B,int32 LB,d_t *C, int32 &NL,d_t &Carry){
 int tPerBlock = 128,bPerGrid = (LA + tPerBlock - 1) / tPerBlock;
 d_t *bGCarry_d=NULL,*ansCarry_h=new d_t[1];
 int32 *cF_d=NULL,*cF_h=new int32[1];
 cudaMalloc((void**)&bGCarry_d, sizeof(d_t)*(bPerGrid+1));
 cudaMemset((void*)bGCarry_d, 0, sizeof(d_t)*(bPerGrid+1));
 cudaMalloc((void**)&cF_d, sizeof(int32));
 cudaMemset((void*)cF_d, 0, sizeof(int32));
 DNumAdd_part1 <<< bPerGrid, tPerBlock >>>
  (const_cast<d_t*>(A), LA, const_cast<d_t*>(B), LB, C, bGCarry_d,CF_d);
 cudaMemcpy(cF_h,cF_d,sizeof(int32), cudaMemcpyDeviceToHost);
 if(*cF_h){
  int gS=tPerBlock,LCarry=bPerGrid;
  bPerGrid = (bPerGrid + tPerBlock - 1) / tPerBlock;
  DNumAdd_part2 <<< bPerGrid, tPerBlock>>>
  (d_buffC, bGC_d, gS, LCarry, LA);
 }
 cudaMemcpy(ansCarry_h,&bGCarry_d[bPerGrid],
            sizeof(d_t), cudaMemcpyDeviceToHost);
 NL= (carry = *ansCarry_h)? LA+1: LA;
 delete[] ansCarry_h; cudaFree(bGCarry_d);  cudaFree(cF_d);
}
```

Листинг 3.6. Вызов функции для исполнения на GPU

фективно решить задачу реализации операции деления. Основное время работы приложения реализующего любой численный метод приходится на две операции: выделение-освобождение памяти и операцию умножения-деления чисел. Ниже приводится алгоритм [3.5–3.6], реализованный в разработанной библиотеке классов.

Для выполнения операции параллельного умножения используется быстрая разделяемая между нитями блока *__shared__* память. В реализации существенно используется важная особенность архитектуры GPU nVidia®, а именно, полностью синхронное выполнение инструкций в рамках одного *warp*. Это избавляет от необходимости выполнять дополнительную синхронизацию между нитями. Исходный код для выполнения на GPU (*kernel*) представлен на листинге [3.7].

Окончательное формирование результата происходит последовательным проходом по массиву **rez[]** и преобразованием 64-х битного числа **rez[j]** в непосредственно разряд ответа и разряд переноса. Эти действия также выполняются на GPU, но в однопоточном режиме.

Алгоритм 3.5. Алгоритм операции параллельного умножения. Часть 1.

Предусловия: Ядро запущено блоком в $BS = 32$ нитей. $LA >= LB$. $MAX\_DIG = 2^{32} - 1$,

Постусловия: $c64[]$ содержит результат без распространения переносов,

Процедура KERNEL_MULT_P1(Число $a[]$, Длина LA, Число $b[]$, Длина LB, Результат $c64[]$)

$ti \leftarrow$ относительный индекс нити в блоке

$gi \leftarrow$ глобальный индекс нити в сетке блоков

$bi \leftarrow$ индекс блока в сетке блоков

Выделить в shared памяти массив $sha[]$ размера LA

Выделить в shared памяти массив $shrez[]$ размера $LA + BS$

Если $gi \geq LB$ **тогда Возвратить**

Конец условия

$copyBlockSize \leftarrow$ размер части числа a, обрабатываемой нитью

Цикл $i \leftarrow ti * copyBlockSize$ **до** $(ti + 1) * copyBlockSize \text{ and } i < LA$

$sha[i] \leftarrow a[i]$, $shrez[i] \leftarrow 0$ ▷ скопировать число a в разделяемую память

Конец цикла

$shrez[LA + ti] \leftarrow 0$

$d \leftarrow b[gi]$, $ti \leftarrow 0$

Цикл $i \leftarrow 0$ **до** LA

$t \leftarrow sha[i] * d$

$shrez[i + ti] \leftarrow shrez[i + ti] + t \mod MAX\_DIG$

$t \leftarrow t \operatorname{div}(MAX\_DIG + 1)$

Конец цикла

$shrez[LA + ti] \leftarrow shrez[LA + ti] + t \mod MAX\_DIG$

$t \leftarrow t \operatorname{div}(MAX\_DIG + 1)$

Цикл $i \leftarrow ti * copyBlockSize$ **до** $(ti + 1) * copyBlockSize \text{ and } i < LA + BS$

$atomicAdd(c64[i + gi], shrez[i])$ ▷ атомарное сложение в общий результат

Конец цикла

Конец процедуры

Алгоритм 3.6. Алгоритм операции параллельного умножения. Часть 2.

Предусловия: Ядро запущено 1 нитью, $L = LA + LB - 1$

Постусловия: Окончательный результат умножения и длина результата

Процедура KERNEL_MULT_P2(Результат умножения массив $c64[]$,Результат массив $c[]$,Размер массива L)

$t \leftarrow 0$

Цикл $i \leftarrow 0$ **до** L

$t \leftarrow t + c64[i]$,

$c[i] \leftarrow t \mod MAX\_DIG$, $t \leftarrow t \operatorname{div}(MAX\_DIG + 1)$

Конец цикла

$c[L] \leftarrow t$

Конец процедуры

```
__global__ void DNumMult(d_t *A,int32 LA,d_t *B,int32 LB,d_t *rez)
{
 int32 lld=threadIdx.x,gld=blockDim.x*blockIdx.x + lld;
 if(gld>=LB) return;
 int32 cBS=(LA+blockDim.x-1)/blockDim.x;
 __shared__ uint64 sha[],shrez[];
 for(int i=lld*cBS;i<(lld+1)*cBS && i<LA;i++){
  sha[i]=A[i]; shrez[i]=0;
 }
 shrez[LA+lld]=0;
 uint64 digit=(uint64)B[gld], t=0UL;
 for(int i=0;i<LA;i++){
  t+=sha[i]*digit;
  shrez[i+lld]+=t&MAX_DIGIT;
  t>>=BIT_IN_DIGIT;
 }
 shrez[LA+lld]+=t&MAX_DIGIT;
 cBS=(LA+blockDim.x+blockDim.x-1)/blockDim.x;
 for(int i=lld*cBS;i<(lld+1)*cBS && i<LA+blockDim.x;i++){
  AtomicAdd(rez[i+gld],shrez[i]);
 }
}
```

Листинг 3.7. Умножение на GPU

Деление

Поскольку стандартный алгоритм деления столбиком является абсолютно последовательным, для реализации в вычислительной системе с массовым параллелизмом, какой является GPU, эффективным является использование итеративных методов деления, выражающих результат через операцию умножения [56].

3.4 Схема работы с программным комплексом для вычисления псевдорешения

Описанные в главах 2-3 диссертации алгоритмы были реализованы в виде программного комплекса, состоящего из нескольких независимых частей: (1) библиотека классов «Exact Computation 2.0» (приложение А), (2) программного комплекса для точного решения задач линейного программирования «ExactLinPSolutor 1.0» с помощью дробно-рационального типов данных (приложение Б), (3) программного комплекса для нахождения псевдорешения интервальной системы линейных уравнений «ExactISLAYSolutor 1.0» (приложение В).

3.4.1 Входные данные

Входными данными для программного комплекса вычисления псевдорешения системы линейных уравнений является файл с задачей (интервальной системой линейных алгебраических уравнений) в следующем формате:

1) Два натуральных числа m, n - количество строк и столбцов матрицы системы.

2) Затем следует m строк, в каждой по $n+1$ паре строк, i-тая строка содержит n пар чисел – интервалы, являющиеся коэффициентами матрицы системы $\underline{a}_{ij}, \overline{a}_{ij}$, и последняя $(n+1)$-я пара чисел описывает интервал в правой части системы $\underline{b}_{ij}, \overline{b}_i$

Замечание. Рациональные числа записываются либо в формате с фиксированной десятичной частью (например, 0.777) либо в формате числитель/знаменатель (например, 656756/989877), до и после знака / пробелов не ставится, для записи отрицательного числа знак минус ставится перед числителем(н-р, -12/13). Все значения разделены пробелом и/или знаком табуляции и/или переводом строки.

Система:

$$\begin{cases} [3,3.5] & x_1 & + & [1,2] & x_2 & + & [12,15] & x_3 & = & [-5/67,7] \\ [1/2,2/3] & x_1 & + & [3,3] & x_2 & + & [1,2] & x_3 & = & [7,9] \end{cases} \tag{3.1}$$

описывается следующим входным набором параметров (файл `islay1.txt`)

$$\begin{array}{cccccccc} 2 & 3 & & & & & & \\ 3 & 3.5 & 1 & 2 & 12 & 15 & -5/67 & 7 \\ 1/2 & 2/3 & 3 & 3 & 1 & 2 & 7 & 9 \end{array} \tag{3.2}$$

3.4.2 Выходные данные

Выходными данными для программного комплекса вычисления псевдорешения системы линейных уравнений является файл в имени которого зафиксировано имя входного файла, тип применяемого расширения интервалов в правой части системы (параметров p, q) и маркер ответа Ans.

Для примера входных данных из раздела выше `islay1.txt` будет сформирован выходной файл `islay1UAns.txt`, где суффикс 'U' указывает на равномерное расширение интервалов в правой части, суффикс 'Ans' указывает на файл ответа.

В выходной файл выводится исходная система, псевдорешение интервальной системы x, полученное минимальное значение z^* – коэффициент расширения интервалов правой части, интервалы правой части, полученные после коррекции системы. Данные дублируются в точном формате (для дальнейших вычислений) и приближённом формате (для удобства восприятия пользователем).

Для примера входных данных `islay1.txt` выходной файл выглядит следующим образом:

```
Precise format:

Initial ISLAY is:
[3;7/2]*x1 [1;2]*x2 [12;15]*x3 = [-5/67;7]
[1/2;2/3]*x1 [3;3]*x2 [1;2]*x3 = [7;9]

Optimal values:
X[1]=0
X[2]=7045/2881
X[3]=-484/2881
```

```
Minimum expansion coefficient is 0

Right hand part vector is:
b[1]=[-5/67;7]
b[2]=[7;9]

Fixed format:

 Initial ISLAY is:
[3.0000;3.5000]*x1 [1.0000;2.0000]*x2 [12.0000;15.0000]*x3 = [-0.0746;7.0000]
[0.5000;0.6666]*x1 [3.0000;3.0000]*x2 [1.0000;2.0000]*x3 = [7.0000;9.0000]

Optimal values:
X[1]=0.000000000000000
X[2]=2.445331482124262
X[3]=-0.167997223186393

Minimum expansion coefficient is 0.000000000000000

Right hand part vector is:
b[1]=[ -0.074626865671641;7.000000000000000 ]
b[2]=[ 7.000000000000000;9.000000000000000 ]

Substitution:
Row[1]=[ 0.429364803887539;2.370704616452620 ]
Row[2]=[ 7.167997223186393;7.000000000000000 ]
```

3.4.3 Запуск приложения и параметры

Запуск производится из командной строки с параметрами:(1) имя файла, содержащего задачу в требуемом формате, (2) способ выбора расширения интервалов в правой части в случае несовместности системы. Комплекс реализован в виде консольного приложения, пример запуска из командной строки:

```
> ./ExactISLAYSolutor.exe islay.txt U.
```

При запуске без параметров программа запросит имя файла и тип расширения интервалов в правой части с клавиатуры.

```
std::cout<<"Correct command is:"<<'\n';
std::cout<<"ExactLinPSolutor <input file> [U,P(default),A,B]"<<'\n';
std::cout<<"where:\n\
1. 'U' means Uniform right part extension\n \
2. 'P' means Proportional right part extension\n \
3. 'A' means LowerBound:UpperBound extension as 1:100\n \
4. 'B' means LowerBound:UpperBound extension as 100:1\n"<<'\n';
std::cout<<"Input <input filename>"<<'\n';
```

Программа автоматически сформирует файл ответа с именем и в формате описанном в разделе выходные данные.

3.5 Выводы

В данной главе построена полная технологическая цепочка реализации эффективного программного комплекса вычисления псевдорешения интервальной системы линейных уравнений. Описаны особенности построения эффективных приложений в гетерогенной вычислительной среде. Описана уникальная архитектура программного комплекса поддержки рациональных вычислений в гетерогенной вычислительной среде. Описаны и обоснованы разработанные параллельные алгоритмы базовых арифметических операций для устройств с массовым параллелизмом на примере GPU. Описана схема работы с программным комплексом вычисления псевдорешения интервальной системы, формат входного и выходного файлов, параметры запуска. Все описанные разработки реализованы в виде программных комплексов и зарегистрированы в Федеральной службе по интеллектуальной собственности в виде самодостаточных программных комплексов.

В следующей главе приводятся результаты тестирования программных комплексов поддержки рациональных вычислений и комплекса вычисления псевдорешения интервальной системы линейных уравнений. Приводятся результаты применения концепции псевдорешения интервальной системы в математическом моделировании.

ГЛАВА 4

Численные эксперименты

Вычислительный эксперимент проводился на компьютере с процессором Intel Core i7-950 3.06 ГГц, 6 Гб ОЗУ, GPU Nvidia 460(1гб GDDR5) GPU Nvidia 660Ti, под управлением ОС Win 7 x64, в качестве компилятора был выбран 64-разрядный Visual C++ 2011.

4.1 Производительность программного обеспечения для точных вычислений в гетерогенных средах с графическими ускорителями

4.1.1 Производительность сравнения

Операция сравнения реализована с помощью алгоритма описанного в главе 3, алгоритм обеспечивает минимальное теоретически достижимое количество шагов для параллельного алгоритма результат которого зависит от всех входных данных, отчет о тестировании представлен в Таблица 4.1.

Таблица 4.1. Время выполнения сравнения (в миллисекундах) в зависимости от длины операндов

Длина	Время вычисления в мс					Fer/CPU	Kep/CPU
	CPU(S)	Fer(S)	Kep(S)	Fer(P)	Kep(P)		
10^1	0.0001	0.130	0.120	0.146	0.148	0.0068	0.0067
10^2	0.0001	0.135	0.130	0.192	0.138	0.0052	0.0073
10^3	0.0001	0.320	0.410	0.155	0.149	0.0065	0.0067
10^4	0.0200	2.260	3.040	0.200	0.210	0.1000	0.0952
10^5	0.1900	21.62	29.91	0.230	0.220	0.8370	0.8636
10^6	1.9000	218.0	301.1	0.521	0.360	3.6538	5.2777
10^7	19.000	—	—	3.170	1.581	6.2776	12.5950
10^8	198.10	—	—	—	—	—	—

4.1.2 Производительность сложения (вычитания)

Производительность сложения случайных чисел, когда длина переноса в среднем не превосходит двух разрядов, на первый план выходит масштабируемость операции. В том случае когда длина переносов велика важным фактором является частота ядер графического процессора, которая отвечает

за скорость последовательного исполнения инструкций. Отчет о тестировании представлен в Таблица 4.2.

Таблица 4.2. Время выполнения сложения (в миллисекундах) в зависимости от длины операндов

Длина	Время вычисления в мс					Fer/CPU	Kep/CPU
	CPU(S)	Fer(S)	Kep(S)	Fer(P)	Kep(P)		
10^1	0.0001	0.300	0.180	0.240	0.240	0.0041	0.0041
10^2	0.0001	0.300	0.200	0.240	0.240	0.0041	0.0041
10^3	0.0001	0.610	0.620	0.240	0.240	0.0041	0.0041
10^4	0.0200	3.910	3.040	0.240	0.240	0.0041	0.0041
10^5	0.5000	37.60	4.010	0.540	0.700	0.9259	0.7142
10^6	6.0100	369.5	42.20	0.880	0.860	6.8181	6.9767
10^7	60.000	—	409.2	4.300	2.76	14.000	21.814
10^8	602.10	—	—	—	—	—	—

Важным фактором, влияющим на производительность и масштабируемость операций на графическом ускорителе, является размер блока [39], нити в пределах блока могут синхронизироваться друг с другом, но слишком маленькие блоки препятствуют полноценной загрузке вычислительного устройства.

Оптимальный размер блока зависит от типа задачи, а также архитектуры ускорителя. Операции сравнения, сложения-вычитания являются операциями, требующими интенсивной работы с памятью. Для ускорителей архитектуры Fermi®при решении задач таких задач лучшим выбором является размер блока равный 512 или 1024 Таблица 4.3.

Таблица 4.3. Зависимость времени (в миллисекундах) выполнения параллельного сложения на GPU Fermi от размера блока B_S

N|B_S	64	128	256	512	1024
10^1	0.24	0.24	0.24	0.2	0.24
10^2	0.24	0.24	0.24	0.2	0.24
10^3	0.24	0.24	0.24	0.2	0.24
10^4	0.24	0.24	0.24	0.2	0.24
10^5	1.02	0.64	0.6	0.54	0.54
10^6	0.96	0.9	1.2	1.1	0.88
10^7	—	—	3.59	3.6	4.3

Для ускорителей архитектуры Kepler®лучшим выбором является размер блока 1024. Это обусловлено большим количеством вычислительных ядер, примерно в 4 раз больше, чем на Fermi®Таблица 4.4.

Таблица 4.4. Зависимость времени (в миллисекундах) выполнения параллельного сложения на GPU Kepler от размера блока B_S

N|B_S	64	128	256	512	1024
10^1	0.23	0.24	0.24	0.24	0.24
10^2	0.23	0.24	0.24	0.24	0.24
10^3	0.24	0.24	0.24	0.24	0.24
10^4	0.24	0.24	0.24	0.24	0.24
10^5	1.06	0.62	0.68	0.68	0.7
10^6	0.82	1.2	1.2	1.2	0.86
10^7	3.3	3.08	3.08	3.05	2. 67

4.1.3 Производительность умножения

Операция умножения существенно зависит от объема быстрой (устанавливаемой непосредственно на чипе процессора) памяти ускорителя. Время выполнения умножения двух операндов в зависимости от их длины указаны в Таблица 4.5.

Таблица 4.5. Время выполнения умножения (в миллисекундах) в зависимости от длины операндов

Длина	Время вычисления в мс		CPU/GPU
	CPU(S)	GPU(P)	
10^2	0.022654	0.654048	0.034637
10^3	2.022228	6.511008	0.310586
10^4	208.7785	89.222176	2.339985
10^5	21069.13	4438.26976	4.74715
10^6	1972030.305	408511.9569	4.82735

4.2 Примеры решения ИСЛАУ небольшой размерности

Рассмотрим насколько хорошо предложенный метод решения интервальной системы линейных алгебраических уравнений справляется с задачами разной степени сложности, нижеследующие примеры взяты из книги [49].

$$\begin{cases} [2,4] \quad x_1 \quad + \quad [-2,1] \quad x_2 \quad = \quad [-2,2] \\ [-1,2] \quad x_1 \quad + \quad [2,4] \quad x_2 \quad = \quad [-2,2] \end{cases} \tag{4.1}$$

Псевдорешение данной задачи равно $(0,0)^T$ и отвечает $z^* = 0$. Псевдорешение принадлежит $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$.

$$\begin{cases} [1,2] & x_1 & + & [-\frac{2}{3}, -\frac{1}{2}] & x_2 & = & [-1,1] \\ [-\frac{2}{3}, -\frac{1}{2}] & x_1 & + & [1,2] & x_2 & = & [-1,1] \end{cases} \tag{4.2}$$

Псевдорешение данной задачи так же равно $(0,0)^T$ и отвечает $z^* = 0$. Псевдорешение принадлежит $\Xi_{tol}(\boldsymbol{A}, \boldsymbol{b})$, то есть допусковому множеству решений исходной задачи.

$$\begin{cases} [3,3]x_1 & + & [1,2]x_2 & = & [5,7] \\ [1,2]x_1 & + & [3,3]x_2 & = & [7,9] \end{cases} \tag{4.3}$$

Псевдорешение данной задачи равно $(1,2)^T$ и отвечает $z^* = 0$. Данный пример примечателен тем, что допусковое множество $\Xi_{tol} = \{(1,2)^T\}$ состоит из одной точки.

$$\begin{cases} [2,3] & x_1 & + & [-1,2] & x_2 & = & [0,60] \\ [1,2] & x_1 & + & [1,3] & x_2 & = & [10,72] \\ [-1,1] & x_1 & + & [0,1] & x_2 & = & [-10,36] \end{cases} \tag{4.4}$$

Псевдорешение данной задачи равно $(10,0)^T$ и отвечает $z^* = 0$.

Таким образом предложенный метод успешно справляется с нахождением точки из допускового множества Ξ_{tol}, даже в том случае, когда множество содержит единственную точку.

4.3 Линейное уравнение межотраслевого экономического баланса

Рассмотрим простой пример. Экономика представлена двумя отраслями производства: промышленностью и сельским хозяйством. За отчетный период получены следующие данные о межотраслевых поставках X_{ij} и векторе объемов конечного использования Y_0.

Таблица 4.6. Данные о межотраслевых поставках за отчетный период, вектор объемов конечного использования Y_0 и конечного продукта Y_n

Отрасли	Отрасли потребители		Y_0	Y_n
	1	2		
1	62	42	102	152
2	32	12	62	202

Валовые выпуски отраслей:

$$\begin{aligned} X_1 &= 62 + 42 + 102 = 206 \\ X_2 &= 32 + 12 + 62 = 106 \end{aligned}$$

Матрица прямых затрат имеет вид.

$$A = \begin{pmatrix} 62/206 & 42/106 \\ 32/206 & 12/106 \end{pmatrix}$$

Для вычисления вектора X – вектора валового объема выпуска для конечного продукта Y_n решалась система

$$(I - \boldsymbol{A})X = \boldsymbol{Y}_n, \tag{4.5}$$

где $\boldsymbol{A}$ – интервальная матрица удельных прямых затрат, $\boldsymbol{Y}_n$ – интервальный вектор конечного продукта или спроса.

Решения задач приведены в таблице [4.7]. Решение точечной системы единственно, следовательно, псевдорешние интервальной системы с ним совпадает, столбцы X_{ps} в таблице [4.7]. Решение системы в интервальной постановке, когда коэффициенты матрицы заданы неточно, приведено в таблице [4.7]. Использовалось равномерное расширение интервалов правой части системы $[\underline{b} - z, \overline{b} + z]$.

Таблица 4.7. (Слева направо) вектор валового объема, конечный продукт, псевдорешение для матрицы с погрешностью 1%, псевдорешение для матрицы с погрешностью 5%

X	Y_n	$X_{ps}(z^* = 2.317)$	$X_{ps}(z^* = 11.396)$
384.763	152	383.777	379.921
295.186	202	293.446	286.644

Для численного эксперимента с реальными данными были использованы данные реального предприятия, представленные в диссертации на соискание ученой степени доктора физико-математических наук А.В. Келлер. Использованы данные модели построенной для предприятия МУП «Производственное объединение водоснабжения и водоотведения» по данным за 2007 год: A– матрица удельных прямых затрат, b – вектор конечного продукта или спроса.

Решалась система

$$(I - \boldsymbol{A})x = \boldsymbol{b}, \tag{4.6}$$

где $\boldsymbol{A}$ – интервальная матрица удельных прямых затрат, $\boldsymbol{b}$ – интервальный вектор конечного продукта или спроса.

Сначала рассматривалась точечная постановка задачи, когда $\underline{A} = \overline{A} = A$ и $\underline{b} = \overline{b} = b$, затем в матрицу коэффициентов была внесена интервальная неопределенность. Матрица удельных прямых затрат имеет размер 24×24 элемента, она доступна в указанном источнике, приведем лишь известный годовой выпуск продукции предшествующего периода X_0 и вектор конечного продукта или спроса b (см. первый и второй столбцы таблицы [4.8]). Псевдорешение интервальной системы для *точечной* постановки($\underline{A} = \overline{A} = A$) системы и отклонение от выпуска прошлого года указаны в таблице (см. соответствующие столбцы таблицы [4.8]).

Таблица 4.8. (Слева направо) годовой выпуск предшествующего периода, конечный продукт, псевдорешение точечной системы, разница по сравнению с показателем прошлого года.

X_0	b	$x_{ps}(z^* = 0)$	$x_{ps} - X_0$
269 548 334.00	269 548 334.00	269 548 334.00	0.00
159 780 927.00	0.00	159 790 202.96	9 275.96
237 747 661.00	237 747 661.00	237 747 661.00	0.00
719 836 394.00	719 836 394.00	719 836 394.00	0.00
235 870 778.00	0.00	235 835 213.91	-35 564.09
447 165 745.00	447 165 745.00	447 165 745.00	0.00
43 239 207.00	0.00	41 222 205.17	-2 017 001.83
171 397 752.00	0.00	170 964 933.18	-432 818.82
41 979 676.00	0.00	41 417 900.09	-561 775.91
15 832 201.00	0.00	15 822 231.37	-9 969.63
11 680 720.00	0.00	11 669 323.33	-11 396.67
29 104 558.00	0.00	29 061 751.11	-42 806.89
67 798 960.00	0.00	66 822 545.73	-976 414.27
13 424 520.00	0.00	13 327 566.80	-96 953.20
210 435 013.00	0.00	207 539 492.35	-2 895 520.65
772 070.00	0.00	785 947.24 13	877.24
23 480 270.00	0.00	23 427 984.41	-52 285.59
6 578 256.00	0.00	6 368 956.83	-209 299.17
3 406 520.00	0.00	3 390 238.41	-16 281.59
576 366.00	0.00	560 356.63	-16 009.37
2 420 882.00	0.00	2 436 247.58	15 365.58
57 570 045.00	0.00	45 756 793.17	-11 813 251.83
637 179 092.50	0.00	627 260 027.47	-9 919 065.03
1 412 502 957.00	0.00	1 403 785 504.94	-8 717 452.06

Разумно считать, что коэффициенты прямых производственных затрат заданы интервалами, т.е. $a_{ij} \in \boldsymbol{a}_{ij}$ и $A = (\boldsymbol{a}_{ij}), i, j = 1, \ldots, n$, рассмотрим интервальную матрицу $\boldsymbol{A}$, полученную из A следующим образом $\boldsymbol{A}_{ij} = [a - a * \underline{t}, a + a * \overline{t}]_{ij}$, $\underline{t}, \underline{t}$ отвечают за интервальный характер коэффициента.

Данные экспериментов, а именно псевдорешение, соответствующее минимальному расширению правой части, а также характер расширения интервалов приведены в таблице [4.9], поскольку расширение интервалов в правой части в отрицательную часть действительной оси для данной математической модели не имеет смысла, то параметры p, q, отвечающие за вид расширения были выбраны как $p_i = 0.00000001, q_i = 1, i = 1 \ldots m$. Вектор псевдорешения x_{ps}, оптимальное значение параметра $z = z^*$, расширенные интервалы в правой части и интервал подстановки $\boldsymbol{A}x$ для определения минимально необходимого расширения по каждому компоненту приведены в таблице [4.9]. Фрагмент файла ответа, выданного программным комплексом приведен в приложении [Г].

Таблица 4.9. (Слева направо) годовой выпуск предшествующего периода, конечный продукт, псевдорешение точечной системы, разница по сравнению с показателем прошлого года.

x_{ps}	X_0	$x_{ps} - X_0$	$\boldsymbol{b}(z^*)$	$\boldsymbol{A}x_{ps}$
269548334	269548334	-0.001	[269548333.999 ; 269828726.529]	[269548333.999 ; 269548333.999]
159806182	159780927	25254.978	[-0.001 ; 280392.529]	[-0.001 ; 31958.041]
237747661	237747661	-0.001	[237747660.999 ; 238028053.529]	[237747660.999 ; 237747660.999]
719836394	719836394	-0.001	[719836393.999 ; 720116786.529]	[719836393.999 ; 719836393.999]
235858797.4	235870778	-11980.565	[-0.001 ; 280392.529]	[-0.001 ; 47167.04275]
447165745	447165745	-0.001	[447165744.999 ; 447446137.529]	[447165744.999 ; 447165744.999]
41206744.94	43239207	-2032462.056	[-0.001 ; 280392.529]	[-0.001 ; 8240.525]
170831915.9	171397752	-565836.115	[-0.001 ; 280392.529]	[-0.001 ; 34162.967]
41421291.68	41979676	-558384.317	[-0.001 ; 280392.529]	[-0.001 ; 8283.429]
15824427.88	15832201	-7773.119	[-0.001 ; 280392.52993]	[-0.001 ; 3164.569]
11670698.74	11680720	-10021.259	[-0.001 ; 280392.529]	[-0.001 ; 2333.906]
29065121.9	29104558	-39436.095	[-0.001 ; 280392.529]	[-0.001 ; 5812.443]
66765962.04	67798960	-1032997.964	[-0.001 ; 280392.529]	[-0.001 ; 13351.85]
13316319.99	13424520	-108200.007	[-0.001 ; 280392.529]	[-0.001 ; 2662.997]
205147473.3	210435013	-5287539.697	[-0.001 ; 280392.529]	[-0.001 ; 41025.392]
782975.5832	772070	10905.583	[-0.001 ; 280392.529]	[-0.001 ; 156.579]
23430327.21	23480270	-49942.788	[-0.001 ; 280392.529]	[-0.001 ; 4685.596]
6363450.854	6578256	-214805.145	[-0.001 ; 280392.529]	[-0.001 ; 1272.562]
3390577.43	3406520	-15942.569	[-0.001 ; 280392.529]	[-0.001 ; 678.047]
553953.5677	576366	-22412.432	[-0.001 ; 280392.529]	[-0.001 ; 110.779]
2436491.205	2420882	15609.204	[-0.001 ; 280392.529]	[-0.001 ; 487.249]
45714055.45	57570045	-11855989.547	[-0.001 ; 280392.529]	[-0.001 ; 9141.896]
626702810.8	637179092.5	-10476281.712	[-0.001 ; 280392.529]	[155095.516 ; 280392.529]
1402102846	1412502957	-10400110.898	[-0.001 ; 280392.529]	[-0.001 ; 280392.529]

На основе полученных данных можно сделать вывод о неустойчивости модели. Применение равномерного расширения интервалов в правой части системы приводит к его сильному расширению в отрицательную область и отрицательным значениям вектора x – объемов производства. Расширение интервалов в сторону увеличения вектора спроса или вектора потребления и введения «штрафа» (p) позволяет добиться устойчивого в рамках заданных колебаний вектора x.

4.4 Определение компонентов бинарных и тернарных смесей методом Фирордта

Для проведения эксперимента электронные спектры поглощения регистрировались на приборе фотоэлектрическом фотометре КФК-3 в диапазоне 400–850 нм, для составления модельных смесей использовались растворы со-

лей $NiSO_4, CoCl_2, CuSO_4, Cr_2(SO_4)_3$ в водных растворах. Данные соли диссоциируют и поглощающими центрами выступают аквакомплексы ионов металлов ($[Ni(H_2O)_6]^{2+}, [Co(H_2O)_4]^{2+}, [Cu(H_2O)_6]^{2+}, [Cr(H_2O)_6]^{3+}$). Для ионов Ni, Co, Cu, Cr растворы солей были взяты с концентрациями соответственно $1/10 = 0.1$ моль на литр(мольл), $1/12 = 0.08(3)$ мольл, $39/250 = 0.156$ мольл, $1/16 = 0.0625$ мольл. Концентрации ионов составили соответственно $1/10 = 0.1$ моль на литр(мольл), $1/12 = 0.08(3)$ мольл, $39/250 = 0.156$ мольл, $1/8 = 0.125$ мольл. Длина кюветы l (толщина поглощающего слоя) равна 1 см.

Для приготовления модельных смесей использовалась мерная посуда (пипетка) второго класса точности для объемов более 5 мл, и первого класса точности для объемов менее 5 мл. При смешивании модельных растворов пересчет концентраций ионов смеси проводился с учетом разбавления.

По результатам измерения составлялась система из m уравнений и n неизвестных, где m – количество различных длин волн, n – количество компонентов смеси. В соответствии с применяемой в методе Фирордта математической моделью уравнения имеют вид

$$\varepsilon_{1i} * x_1 + \varepsilon_{2i} * x_2 + \ldots + \varepsilon_{in} * x_n = A_i, i = 1 \ldots, n,$$

где $\varepsilon_{1i}, \varepsilon_{2i}, \ldots \varepsilon_{3i}$ отвечают за уровни поглощения отдельных компонентов на данной длине волны λ_i, а A_i–поглощение смеси на этой длине волны. Примеры данных для различных компонентов модельных смесей приводятся в таблице [4.10].

Таблица 4.10. Коэффициенты поглощения для компонентов модельных смесей (левая часть таблицы) и данные о поглощении для смесей (правая часть таблицы).

λ	$(Ni)\varepsilon_1$	$(Co)\varepsilon_2$	$(Cu)\varepsilon_3$	$(Cr)\varepsilon_4$	$Ni{:}Co(1{:}2)$	$Ni{:}Co(1{:}1)$	$Ni{:}Co(2{:}1)$	$Ni{:}Co{:}Cu$	$Ni{:}Co{:}Cr$
410	3.75	0.396	0.070512821	4.344	0.154	0.209	0.269	0.203	0.236
420	2.56	0.624	0.070512821	4.304	0.125	0.155	0.185	0.15	0.189
430	1.36	1.056	0.064102564	3.912	0.111	0.13	0.117	0.114	0.152
440	0.67	1.716	0.057692308	3.184	0.13	0.109	0.93	0.112	0.142
450	0.41	2.592	0.051282051	2.528	0.179	0.138	0.103	0.137	0.158
460	0.3	3.54	0.051282051	1.928	0.235	0.174	0.126	0.166	0.182
470	0.2	4.2	0.044871795	1.432	0.275	0.201	0.138	0.188	0.198
480	0.08	4.74	0.038461538	1.104	0.306	0.221	0.149	0.204	0.209
490	0.01	5.196	0.044871795	1.016	0.333	0.239	0.159	0.219	0.223
500	0	5.868	0.038461538	1.128	0.337	0.269	0.179	0.245	0.25
510	0	6.192	0.038461538	1.336	0.398	0.284	0.191	0.258	0.264
520	0.01	6.012	0.057692308	1.696	0.389	0.277	0.187	0.25	0.262
530	0.05	5.232	0.064102564	2.144	0.34	0.243	0.166	0.224	0.238
540	0.07	4.032	0.070512821	2.656	0.266	0.189	0.13	0.174	0.198
550	0.09	2.796	0.08974359	3.144	0.182	0.133	0.093	0.125	0.155
560	0.13	1.764	0.102564103	3.6	0.119	0.088	0.063	0.086	0.123
570	0.2	1.056	0.128205128	3.88	0.075	0.058	0.045	0.061	0.103
580	0.3	0.648	0.16025641	4.024	0.051	0.044	0.038	0.051	0.093
590	0.44	0.468	0.205128205	3.976	0.044	0.04	0.042	0.051	0.091
600	0.61	0.396	0.256410256	3.784	0.045	0.047	0.05	0.057	0.093
610	0.83	0.36	0.314102564	3.44	0.049	0.056	0.062	0.065	0.097
620	1.1	0.336	0.397435897	3	0.057	0.068	0.081	0.077	0.102
630	1.37	0.312	0.506410256	2.512	0.065	0.08	0.099	0.09	0.1
640	1.65	0.288	0.621794872	2.12	0.072	0.091	0.114	0.103	0.114
650	1.86	0.264	0.756410256	1.696	0.078	0.102	0.128	0.113	0.118
660	1.94	0.228	0.923076923	1.288	0.77	0.104	0.134	0.116	0.115
670	1.91	0.18	1.102564103	1.024	0.074	0.1	0.13	0.116	0.1
680	1.93	0.144	1.320512821	0.824	0.072	0.1	0.128	0.118	0.106
690	2	0.108	1.512820513	0.608	0.073	0.102	0.134	0.122	0.104
700	2.11	0.06	1.756410256	0.384	0.073	0.104	0.14	0.129	0.105
710	2.2	0.036	1.974358974	0.256	0.075	0.107	0.145	0.134	0.106
720	2.24	0	2.179487179	0.168	0.075	0.1	0.146	0.138	0.107
730	2.21	0	2.384615385	0.112	0.073	0.106	0.144	0.138	0.104
740	2.28	0	2.583333333	0.064	0.068	0.1	0.136	0.134	0.097
750	1.9	0	2.730769231	0.048	0.062	0.091	0.124	0.128	0.09
760	1.69	0	2.891025641	0.024	0.055	0.081	0.111	0.121	0.081
770	1.49	0.012	2.980769231	0.016	0.049	0.071	0.094	0.113	0.071
780	1.29	0	3.083333333	0.008	0.043	0.063	0.084	0.105	0.063
790	1.1	0.024	3.134615385	0.008	0.036	0.053	0.072	0.097	0.055
800	0.94	0.36	3.192307692	0.008	0.031	0.046	0.06	0.091	0.049
810	0.79	0.048	3.185897436	0.008	0.027	0.039	0.051	0.085	0.043
820	0.65	0.06	3.185897436	0	0.024	0.032	0.042	0.038	0.038
830	0.55	0.084	3.16025641	0	0.021	0.028	0.036	0.033	0.033
840	0.45	0.096	3.108974359	0	0.2	0.024	0.031	0.029	0.029
850	0.38	0.12	3.057692308	0	0.018	0.021	0.027	0.027	0.027

Составленные системы, с числом уравнений существенно превышающим число неизвестных, являются несовместными.

Результаты экспериментов для точечных систем, без учета погрешностей, вносимых при составлении смесей мерными и непосредственно прибором КФК-3 приведены в таблицах [4.11,4.12,4.13] (столбцы x_{ps}). Поскольку такая система может быть решена традиционными метода решения некорректных задач было найдено псевдорешение по Лаврентьеву (столбцы N_{ps}). Результаты так же приведены в таблице [4.11,4.12,4.13] (столбцы $x_{ps}(1\%)$).

Для учета указанной в описании эксперимента интервальной неопределенности, вносимой погрешностями мерной посуды, а также погрешностей измерения значений уровня поглощения прибором КФК-3 была внесена однопроцентная относительная погрешность в коэффициенты системы, т.е. величины $\varepsilon_{1i}, \varepsilon_{2i}, \quad i = 1, \ldots, m.$

Таблица 4.11. Результаты для смеси из двух элементов$Ni : Co$ в соотношении (1:2).

	Смесь $Ni : Co(1:2)$					
	x_{ps}	Δ в %	$x_{ps}(1\%)$	Δ в %	N_{ps}	Δ
	0.032393417	2.819747859	0.032182923	3.45123206	0.033	1
	0.061083844	9.950919065	0.061047921	9.886257284	0.063	13.4
z^*	0.021439996150514	—	0.021441490731914	—	—	—

Таблица 4.12. Результаты для смеси из двух элементов$Ni : Co$ в соотношении (1:1).

	Смесь $Ni : Co(1:1)$					
	x_{ps}	Δ в %	$x_{ps}(1\%)$	Δ в %	N_{ps}	Δ
	0.046757601	6.484797187	0.047194476	5.611048138	0.04811	3.78
	0.048345704	16.02968943	0.048012066	15.22895956	0.04589	10.136
z^*	0.015356598736896	—	0.016263622822178	—	—	—

Таблица 4.13. Результаты для смеси из двух элементов$Ni : Co$ в соотношении (2:1).

	Смесь $Ni : Co(1:2)$					
	x_{ps}	Δ в %	$x_{ps}(1\%)$	Δ в %	N_{ps}	Δ
	0.06511194	2.332089552	0.065078613	2.382080015	0.06489	2.665
	0.03125	12.5	0.03112595	12.05341853	0.03061	10.196
z^*	0.012455223880597	—	0.012503030760339	—	—	—

Пример выходного файла для случая анализа трехкомпонентной смеси $Ni : Co : Cu$ в соотношениях $5 : 5 : 1$ приведен в приложении [Д]. Данная модель на модельных смесях из трех компонентов также показала хорошую устойчивость. Близость векторов псевдорешения интервальной системы и псевдорешения по М.М. Лаврентьеву говорит о правомочности упо-

требления данного термина и распространение его на случай интервальной системы.

4.5 Результаты нагрузочного тестирования

В качестве модельных задач с высокой степенью вырожденности использованы системы с модифицированной матрицей Гильберта:

$$\mathbf{A} = \left[\frac{i(1-\delta)}{i+j-1}, \frac{i(1+\delta)}{i+j-1}\right]_{n\times n}; \ \mathbf{b} = [1, 1/2, \ldots, 1/(n-1), 1/n]^T$$

Зависимость минимального расширения правой части (параметр z^*), соответствующего псевдорешению, при фиксированном значении $n = 20$ приведена в таблице [4.14].

Таблица 4.14. Минимальное расширение правой части системы

δ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
z^*	0,81	0,389	0,1	0,025	0,0062	0,0017

В таблице [4.15] приведены результаты времени затраченного на решение задачи для различных размерностей модельной интервальной системы.

Таблица 4.15. Время работы

Размерность матрицы (n)	10	20	50	100
Время работы	0,46с	7,73с	7,39м	15,1ч

4.6 Выводы

Проведенные вычислительные эксперименты показывают, что комплекс поддержки дробно-рациональных вычислений позволяет получить в гетерогенных вычислительных средах существенный прирост производительности приложений для больших чисел. В случае вычислений на процессорах архитектуры x_86 программный комплекс вычисления псевдорешения интервальной системы позволяет получать прирост производительности за счет использования крупнозернистого параллелизма.

Классы `overlong` и `rational` обеспечивают необходимую точность вычислений и, тем самым, позволяют избежать зацикливания симплекс-метода в случае сильной вырожденности задачи линейного программирова-

ния, что важно для разработанного **подхода к регуляризации неточно заданных систем**.

Рассмотренные примеры задач математического моделирования показали жизнеспособность подхода как в случае неустойчивых задач, так и в случае сильно вырожденных систем.

Дальнейшая работа направлена на получение более эффективных реализаций программных комплексов за счет оптимизации параллельных версии программ, расширение списка поддерживаемых библиотекой «Exact Computation 2.0» архитектур, реализацию графического пользовательского интерфейса и разработку проблемно-ориентированных программных решений применяющих разработанных подход к регуляризации неточно заданных систем к задачам математического моделирования из различных прикладных областей.

Заключение

Учет интервальной неопределённости повышает адекватность математических моделей. Подход позволяет единообразно решать как задачи с изначально интервальным характером данных, так и те практические задачи, в которых правая часть b и элементы матрицы A, т. е. коэффициенты системы $Ax = b$, известны приближенно. В этих случаях вместо системы мы имеем дело с некоторой другой системой $\tilde{A}x = \tilde{b}$ такой, что $\|\tilde{A} - A\| \leq h$ $\|\tilde{b} - b\| \leq \delta$, где смысл норм обычно определяется характером задачи [45].

В работе введено и описано понятие *псевдорешения интервальной системы линейных уравнений*, доказано существование псевдорешения интервальной системы для любой интервальной СЛАУ, описаны свойства псевдорешения интервальной системы. Выписана задача линейного программирования для нахождения псевдорешения, предложен подход к коррекции правой части системы для несовместной ИСЛАУ.

Погружение **неточно заданной СЛАУ** в интервальную СЛАУ и переход к поиску точки допускового множества ИСЛАУ позволяет получить устойчивое решение исходной СЛАУ, это обусловлено свойствами допускового множества. Подход назван **интервальной регуляризацией**.

Предложен стратегический путь для эффективной реализации подхода, а именно использование ресурса параллелизма современных гетерогенных вычислительных систем.

В работе построена полная технологическая цепочка реализации эффективного программного комплекса вычисления псевдорешения интервальной системы линейных уравнений. Описаны особенности построения эффективных приложений в гетерогенной вычислительной среде. Описана уникальная архитектура разработанного программного комплекса поддержки рациональных вычислений в гетерогенной вычислительной среде. Описаны и обоснованы разработанные параллельные алгоритмы базовых арифметических операций для устройств с массовым параллелизмом на примере GPU. Описана схема работы с программным комплексом вычисления псевдорешения интервальной системы, формат входного и выходного файлов, параметры запуска.

Проведенные вычислительные эксперименты показывают, что комплекс поддержки дробно-рациональных вычислений позволяет получить в гетеро-

генных вычислительных средах существенный прирост производительности приложений для больших чисел. В случае вычислений на процессорах архитектуры x_86 программный комплекс вычисления псевдорешения интервальной системы позволяет получать прирост производительности за счет использования крупнозернистого параллелизма.

Классы `overlong` и `rational` обеспечивают необходимую точность вычислений и, тем самым, позволяют избежать зацикливания симплекс-метода в случае сильной вырожденности задачи линейного программирования, что важно для разработанного `подхода к регуляризации неточно заданных систем`.

Рассмотрены примеры задач математического моделирования: межотраслевая модель Леонтьева и анализ смеси веществ методом Фирордта. Предлагаемый в работе подход показал свою жизнеспособность. Метод погружения неточно заданных систем в интервальные СЛАУ позволил получить устойчивые к колебаниям входных данных решения. Решения хорошо согласуются с известными методами.

Все описанные разработки реализованы в виде программных комплексов и зарегистрированы в Федеральной службе по интеллектуальной собственности в виде самодостаточных программных комплексов.

Цели поставленные в работе были успешно достигнуты, решения всех поставленных задач позволяют говорить о высоком потенциале разработки для целей математического моделирования в различных прикладных областях.

- Предложен новый подход к регуляризации СЛАУ за счет погружения в ИСЛАУ и перехода к поиску точки из допускового множества решений ИСЛАУ.
- Предложен новый подход к поиску точки допускового из множества интервальной системы линейных алгебраических уравнений, позволяющий наряду с исследованием совместности задачи о допусках для ИСЛАУ проводить коррекцию правой части системы. Учет интервальной неопределённости в исходных данных повышает качество математического моделирования.
- Разработан и протестирован масштабируемый параллельный алгоритм для нахождения псевдорешения интервальной системы линейных алгебраических уравнений в распределённых гетерогенных вычислительных системах

с применением точных дробно-рациональных типов данных. Эффективность реализованного численного метода достигнута за счет применения технологии крупно-зернистого параллелизма и библиотеки MPI.

- Разработан и протестирован комплекс программ для безошибочных дробно-рациональных вычислений. Выполнена реализация последовательных алгоритмов для базовых арифметических операций и их параллельных версий для распределённых вычислительных систем с GPU. Применение технологии GPGPU позволяет на порядок увеличить эффективность данного ПО по сравнению с последовательной версией. Разработан и реализован программный комплекс для численного решения задач математического моделирования.

Дальнейшая работа направлена на получение более эффективных реализаций программных комплексов за счет оптимизации параллельных версии программ, расширение списка поддерживаемых библиотекой «Exact Computation 2.0» архитектур, реализацию графического пользовательского интерфейса и разработку проблемно-ориентированных программных решений применяющих разработанных подход к регуляризации неточно заданных систем к задачам математического моделирования из различных прикладных областей.

Основные обозначения

Основные обозначения и наименования соответствуют принятому международному стандарту обозначений интервального анализа (Standardized notation in interval analysis) [67].

Неинтервальные (точечные) величины никак специально не выделяются,

$\mathbb{R}$ — множество вещественных (действительных) чисел

$\mathbb{IR}$ — классическая интервальная арифметика

$\boldsymbol{A}, \boldsymbol{b}$ — интервалы и интервальные величины (векторы, матрицы и пр.),

$\overline{\boldsymbol{a}}, \underline{\boldsymbol{a}}$ — нижняя и верхняя граница интервала $\boldsymbol{a}$,

$\operatorname{mid} \boldsymbol{a}$ — середина (медиана) интервала $\boldsymbol{a}$,

$\operatorname{rad} \boldsymbol{a}$ — радиус интервала $\boldsymbol{a}$,

$\operatorname{vert} \boldsymbol{a}$ — множество крайних точек интервала $\boldsymbol{a}$,

$|\boldsymbol{a}|$ — абсолютная величина (магнитуда) интервала $\boldsymbol{a}$,

$\langle \boldsymbol{a} \rangle$ — мигнитуда интервала $\boldsymbol{a}$

$\operatorname{int} set$ — внутренне точки множества set

$\chi(\boldsymbol{a})$ — функционал Рачека от интервала $\boldsymbol{a}$

$$\chi(\boldsymbol{a}) = \begin{cases} \underline{\boldsymbol{a}}/\overline{\boldsymbol{a}}, & \text{если } |\underline{\boldsymbol{a}}| < |\overline{\boldsymbol{a}}|; \\ \overline{\boldsymbol{a}}/\underline{\boldsymbol{a}}, & \text{иначе.} \end{cases}$$

Список рисунков

Список таблиц

Список литературы

1. Бабенко, К. О доказательных вычислениях и математическом эксперименте на эвм / К.И. Бабенко // Успехи математических наук. —1985. — Т. 49, № 4. —С. 137–138. 6, 32, 48

2. Бабенко, К. Основы численного анализа / К.И. Бабенко. — Москва: Наука, 1986. 6

3. Бабенко, К. И. О доказательных вычислениях в задаче об устойчивости течения Пуазейля / К. И. Бабенко, М. М. Васильев. —ДАН СССР, 1983. —Т. 273. —С. 1289–1294. 6, 32

4. Бибердорф, Э. Гарантированная точность в прикладных задачах линейной алгебры. Учебное пособие. / Э.А. Бибердорф. —Новосибирск: Издательство НГУ, 2008. 6, 8, 48

5. Бибердорф, Э. Гарантированная точность современных алгоритмов линейной алгебры / Э.А. Бибердорф, Н.И. Попова. —Новосибирск: Изд-во СО РАН, 2006. 6, 8

6. Бибердорф, Э. А. Программа «GALA 2.0» регистрационный номер в фап: Pr10019 . — Новосибирск: Институт вычислительной математики и математической геофизики СО РАН, 2010. — URL: http://fap.sbras.ru/node/571. 6

7. Власова, И. Возможность определения компонентов бинарных смесей методом фирордта с погрешностями, не превышающими заданный предел / И.В. Власова, В.И. Вершинин // Журнал аналитической химии. —2009. —Т. 64, № 6. —С. 571–576. 32

8. Воеводин, В. В. Вычислительная математика и структура алгоритмов. / В. В. Воеводин. —Изательство московского университета, 2010. 56, 58

9. Гарантированная точность решения систем линейных уравнений в евклидовых пространствах / С.К. Годунов, Г.А. Антонов, О.П. Кирилюк, В.И. Костин. —Новосибирск: Наука, 1988. 6, 8

10. Гергель, В. П. Высокопроизводительные вычисления для многоядерных процессорных систем / В. П. Гергель. — Издательство Московского университета, 2010. 14

11. Годунов, С. Современные аспекты линейной алгебры / С.К. Годунов. — Новосибирск: Научная книга, 2002. 6, 8, 48

12. Годунов, С. К. О гарантированной точности в спектральных задачах / С. К. Годунов // Заседания Санкт-Петербургского математического общества. — 2004. 6, 8, 48

13. Голодов, В. Библиотека классов «Exact Computation 2.0» свидетельство о государственной регистрации программы для ЭВМ №2013612818 от 14 марта / В.А. Голодов, А. В. Панюков // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. — М.: ФИПС, 2013. — № 3. — С. 251. 11, 15, 49, 54, 56

14. Голодов, В. Программа «ExactISLAYSolutor 1.0» свидетельство о государственной регистрации программы для ЭВМ № 2014610333 от 09 января / В.А. Голодов, А. В. Панюков // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. — М.: ФИПС, 2014. — № 2. 11

15. Голодов, В. Программа «ExactLinPSolutor 1.0» свидетельство о государственной регистрации программы для ЭВМ № 2014610445 от 09 января / В.А. Голодов, А. В. Панюков // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. — М.: ФИПС, 2014. — № 2. 11

16. Голодов, В. А. Адаптация библиотеки «Exact Computational» для гетерогенных вычислительных сред / В. А. Голодов // Научный поиск [текст]: материалы четвертой научной конференции аспирантов и докторантов. Естественные науки. — Челябинск: Издательский центр ЮУрГУ, 2012. 15

17. Голодов, В. А. Распределенные символьные дробно-рациональные вычисления на процессорах x86 и x64 / В. А. Голодов // Параллельные вычислительные технологии (ПаВТ'2012): труды международной научной

конференции (Новосибирск, 26 - 30 марта). — Челябинск: Издательский центр ЮУрГУ, 2012. — С. 719. 55, 56

18. Горелик, В. А. Матричная коррекция задачи линейного программирования с несовместной системой ограничений / В. А. Горелик // Журнал вычислительной матетатики и математической физики. — 2001. — Т. 41, № 11. — С. 1697–1705. 5

19. Горелик, В. А. Оптимальная матричная коррекция несовместных систем линейных алгебраических уравнений уравнений с блочными матрицами / В. А. Горелик, В. И. Ерохин, Р. В. Печенкин // Дискретный анализ и исследование операций, серия 2. — 2005. — Т. 12, № 5. — С. 3–23. 5

20. Горелик, В. А. Методы коррекции несовместных линейных систем с разреженными матрицами / В. А. Горелик, И. А. Золтоева, Р. В. Печенкин // Дискретный анализ и исследование операций, серия 2. — 2007. — Т. 14, № 2. — С. 62–75. 5

21. Достоверные вычисления. Базовые численные методы / У. Кулиш, Д. Рац, Р. Хаммер, М. Хокс. — НИЦ «Регулярная и хаотическая динамика», 2005. 6, 8, 33

22. Евтушенко, Ю. Параллельные алгоритмы решения задач линейного программирования / Ю.Г. Евтушенко, А.И. Голиков // Российская конференция «Дискретная оптимизация и исследование операций»: Материалы конференции (Алтай, 27 июня – 3 июля 2010 г.). — 2010. — С. 25–29. 27, 48

23. Задачи линейной оптимизации с неточными данными / М. Фидлер, Й. Недома, Я. Равик [и др.]. — Институт компьютерных исследований, R&C Dynamics, 2008. — С. 288. 25

24. Иванов, В. К. О линейных некорректных задачах / В. К. Иванов // Доклады Академии наук СССР. — 1962. — Т. 145. 37

25. Канторович, Л. В. О некоторых новых подходах к вычислительным методам и обработке наблюдений / Л. В. Канторович // Сибирский Математический Журнал. — 1962. — Т. 3, № 5. — С. 701–709. 17

26. Линеев, А. В. Технологии параллельного программирования для процессоров новых архитектур / А. В. Линеев, Д. К. Боголепов, С. И. Бастраков. —Изательство московского университета, 2010. —С. 160. 51

27. Максимов, В. Арифметика рациональных чисел и компьютерное исследование интегральных уравнений / В.П. Максимов // Соросовский образовательный журнал. —1999. —№ 3. —С. 121–126. 6, 7, 11

28. Малышев, А. Введение в вычислительную линейную алгебру / А.Н. Малышев. —Новосибирск: Наука, 1991. 6, 8

29. Никайдо, Х. Выпуклые структуры и математическая экономика / Х. Никайдо. —Москва: Мир, 1972. 29

30. Панков, П. С. Доказательные вычисления на электронных вычислительных машинах / П. С. Панков. —Фрунзе: Илим, 1978. 6, 8

31. Панюков, А. В. Сложность нахождения гарантированной оценки решения приближенно заданной системы линейных алгебраических уравнений / А. В. Панюков, М. И. Германенко // Известия Челябинского научного центра УрО РАН. —2000. —№ 4. —С. 21–30. 6

32. Панюков, А. В. Библиотека классов «Exact Computation» номер гос. регистрации 2009612777 от 29 мая 2009 г. / А. В. Панюков, М. И. Германенко, В. В. Горбик // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. — М.: ФИПС, 2009. — № 3. — С. 251. 15

33. Панюков, А. В. Подход к решению систем линейных алгебраических уравнений с интервальной неопределенностью в исходных данных / А. В. Панюков, В. А. Голодов // Вестник Южно-Уральского Государственного Университета. Серия: “Математическое моделирование и программирование”. —2013. —Т. 6, № 2. —С. 109–119. 13

34. Панюков, А. В. Программная реализации алгоритма решения системы линейных алгебраических уравнений с интервальной неопределенностью

в исходных данных / А. В. Панюков, В. А. Голодов // Управление большими системами. М.: ИПУ РАН. — 2013. — № 43. — С. 78–94. 13

35. Панюков, А. В. Параллельные реализации симплекс-метода для безошибочного решения задач линейного программирования / А. В. Панюков, В. В. Горбик // Вестник Южно-Уральского государственного университета. Серия: "Математическое моделирование и программирование". — № 25. — 2011. — С. 107–118. 8, 27, 47

36. Панюков, А. В. Применение массивно-параллельных вычислений для решения задач линейного программирования с абсолютной точностью / А. В. Панюков, В. В. Горбик // Автоматика и телемеханика. — 2012. — № 2. — С. 73–88. 44

37. Панюков, А. В. Реализация базовых операций целочисленной арифметики в гетерогенных системах. / А. В. Панюков, С. Ю. Лесовой // Параллельные вычислительные технологии (ПаВТ'2012). Труды международной научной конференции (Новосибирск, 26 марта – 30 марта 2012 г.). — Челябинск: Издательский центр ЮУрГУ., 2012. — С. 77–84. 56

38. Параллельная реализация метода ньютона для решения больших задач линейного программирования / В.А. Гаранжа, А.И. Голиков, Ю.Г. Евтушенко, М. Х. Нгуен // Журнал вычислительной математики и математической физики. — 2009. — Т. 49, № 8. — С. 1369–1384. 8, 48

39. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / А. В. Боресков, А. А. Харламов, Н. Д. Марковский [и др.]. — Москва, Издательство московского университета, 2012. — С. 285. 51, 59, 72

40. Самохин, А. Проблема четырех раскрасок: неоконченная история доказательства / А.В. Самохин // Соросовский образовательный журнал. — 2000. — Т. 6, № 7. — С. 91–96. 32

41. Семейство продукции intel® xeon phi™ , Intel Corporation. — 2013. —August. — URL: http://www.intel.ru/content/www/ru/ru/processors/xeon/xeon-phi-detail.html. 51

42. Советский энциклопедический словарь / Под ред. А. М. Прохоров. — 4 изд. — М.: «Советская энциклопедия», 1988. — С. 408–409. 16

43. Справочник по элементарной математике механике и физике. / Под ред. Е. Волкинд. — Наука и техника. Минск, 1965. — С. 107. 17

44. Схрейвер, А. Теория линейного и целочисленного программирования / А. Схрейвер. — Москва, Мир, 1991. — Т. 1. — С. 360. 42

45. Тихонов, А. Н. Методы решения некорректных задач / А. Н. Тихонов, В. Я. Арсенин. — Москва, Наука, 1979. — С. 285. 35, 37, 84

46. Хлебалин, Н. А. Аналитический метод синтеза регуляторов в условиях неопределённости параметров объекта / Н. А. Хлебалин // Аналитические методы синтеза регуляторов / Саратовский политехнический институт. — 1981. — С. 107–123. 8

47. Решение интервальной алгебраической задачи о допусках : Препринт : 5 / ВЦ СО АН СССР. Красноярск ; исполн.: В. В. Шайдуров, С. П. Шарый : 1988. 8

48. Шарый, С. П. Решение интервальной линейной задачи о допусках / С. П. Шарый // Автоматика и телемеханика. — 2004. — № 10. — С. 147–162. 8, 26

49. Шарый, С. П. Конечномерный интервальный анализ / С. П. Шарый. — Новосибирск, «XYZ», 2013. — С. 285. 8, 16, 17, 26, 27, 29, 73

50. Якобовский, М. В. Введение в параллельные методы решения задач / М. В. Якобовский. — Изательство московского университета, 2013. — С. 328. 58

51. Beaumont, O. Linear interval tolerance problem and linear programming techniques / O. Beaumont, B. Philippe // Reliable Computing. — 2001. — Vol. 6, no. 4. — P. 365–390. 27

52. The gnu mp bignum library , Free Software Foundation. — 2013. — August. — URL: http://gmplib.org/. 11, 15, 44, 48

53. Hall, J. Towards a practical parallelization of the simplex method / Ju. Hall // Journal Computational Management Science. — 2010. — Vol. 7, no. 2. — P. 139–170. 8, 36, 48

54. High performance computing (hpc) and supercomputing — nvidia tesla , NVIDIA Corporation. — 2013. —September. — URL: http://www.nvidia.com/object/tesla-supercomputing-solutions.html. 51

55. Ieee 754-2008 standart for floating-point arithmetic // [E-ISBN: 978-0-7381-5752-8], IEEE. — 2008. —August. — URL: http://ieeexplore.ieee.org/servlet/opac?punumber=4610933. 15

56. Knuth, D. E. The Art of Computer Programming / D. E. Knuth. — 2 edition. — Addison-Wesley Longman, 1981. — Vol. 2. — P. 688. 60, 62, 67

57. The OpenCL Programming Book / Ryoji Tsuchiyama, Takashi Nakamura, Takiro Lizuka [et al.]. — Fixstars Corporation and Impress Japan Corporation, 2010. 50

58. Pan, V. Fast and efficient parallel linear programming and linear least squares computations / V.Y. Pan, J.H. Reif // Proceedings of the VLSI Algorithms and Architectures, Aegean Workshop on Computing. — 1986. — P. 283–295. 8, 48

59. Panyukov, A. V. Library of classes "Exact Computation" state. reg. 2009612777 dated may 29, 2009. / A. V. Panyukov, M. I. Germanenko, V. A. Gorbik // Programs for computers, data bases, topology of VLSI. Official bulletin of Russian Agency for patents and trademarks. — Federal Service for Intellectual Property, 2009. — No. 3. — P. 251. 44, 54

60. Panyukov, A. V. Computing best possible pseudo-solutions to interval linear systems of equations / A. V. Panyukov, V. A. Golodov // Reliable Computing. — 2013. — Vol. 19. — P. 215–228. 13

61. Panyukov, A. V. Exact and guaranteed accuracy solutions of linear programming problems by distributed computer systems with mpi / A. V. Panyukov,

V. V. Gorbik // Tambov University Reports. Series: Natural and Technical Sciences. — 2010. — Vol. 15, no. 4. — P. 1392–1404. 8, 36

62. Panyukov, A. V. Using massively parallel computations for absolutely precise solution of the linear programming problems / A. V. Panyukov, V. V. Gorbik // Automation and Remote Control. — 2012. — Vol. 73, no. 2. — P. 276–290. 8

63. Parallel programming and computing platform — cuda — nvidia , NVIDIA Corporation. — 2013. —Febbuary. — URL: htpp://www.nvidia.com/object/cuda_home.html. 51

64. Rohn, J. Inner solutions of linear interval systems / J. Rohn // Interval Mathematics and Springer Verlag. — 1985 and 1986. — P. 157–158. 8, 25

65. Rohn, J. Systems of linear interval equations / J. Rohn // Linear Algebra and its Applications. — 1989. — Vol. 126. — P. 39–78. 8

66. Shary, S. P. A new technique in systems analysis under interval uncertainty and ambiguity / S. P. Shary // Reliable Computing. — 2002. — Vol. 8, no. 5. — P. 321–418. 8

67. Standardized notation in interval analysis / R. B. Kearfott, M. T. Nakao, A. Neumaier [et al.] // Computational Technologies. — 2010. — Vol. 15, no. 1. — P. 7–13. 87

68. Weiser, M. E. Atomic weights of the elements 2009 (iupac technical report) / M. E. Weiser, T. B. Coplen // Pure and Applied Chemistry. — 2011. — Vol. 83, no. 2. — P. 359–396. 16

69. Yarmish, G. A distributed implementation of the simplex method / G.G. Yarmish // UMI Dissertations Publishing. — 2001. 48

Приложение А

(справочное)

Свидетельство о государственной регистрации программы для ЭВМ. Библиотека классов «Exact Computation 2.0»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2013612818

Библиотека классов «Exact Computation 2.0»

Правообладатель(ли): ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет) (ФГБОУ ВПО «ЮУрГУ» (НИУ)) (RU)***

Автор(ы): ***Голодов Валентин Александрович (RU), Панюков Анатолий Васильевич (RU)***

Заявка № **2012661364**

Дата поступления **19 декабря 2012 г.**

Зарегистрировано в Реестре программ для ЭВМ **14 марта 2013 г.**

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

Приложение Б

(справочное)

Свидетельство о государственной регистрации программы для ЭВМ «ExactLinPSolutor 1.0»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014610445

Программа «ExactLinPSolutor 1.0»

Правообладатель: *Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет) (ФГБОУ ВПО «ЮУрГУ» (НИУ)) (RU)*

Авторы: *Голодов Валентин Александрович (RU), Панюков Анатолий Васильевич (RU)*

Заявка № 2013660285
Дата поступления 11 ноября 2013 г.
Дата государственной регистрации в Реестре программ для ЭВМ *09 января 2014 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

Приложение В

(справочное)

Свидетельство о государственной регистрации программы для ЭВМ «ExactISLAYPSolutor 1.0»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014610333

Программа «ExactISLAYSolutor 1.0»

Правообладатель: *Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет) (ФГБОУ ВПО «ЮУрГУ» (НИУ)) (RU)*

Авторы: *Голодов Валентин Александрович (RU), Панюков Анатолий Васильевич (RU)*

Заявка № 2013660335
Дата поступления 11 ноября 2013 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *09 января 2014 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

Приложение Г

(справочное)

Фрагмент выходного файла для эксперимента с интервальной моделью Леонтьева

Optimal values:

$$X[1] = 20644197993971915143541243070174834734293864772132311653481767006212917817081431721865755252932130913695546924247170071235407853/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101770470$$

$$X[2] = 15299067204687698271623975978989043283385842300462894376530461817457948162680997148685383794201500675695065494529913126179722212142043395793787898904328338584230046289437653046181745794816268099714868538379420150067569506549452991312617972221214204339573514000078211676588379488887728000006007673841234150099956940241643774815911357743355218685535050884816088772213056625000000$$

$$X[3] = 18208644488552641671925492576366628559564784387561785724491673822447277379758180263589095595074033393829735747413697120034910984/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101770447300$$

$$X[4] = 55130910365796377416503873618935207525736946242524968241115834339852483469664900381147526447434413538694947884807390864630139033/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101770447300$$

$$X[5] = 22579974992793460834882485031834457717272282109604990860159368672874402528539768028256940123744741468033451784547126688301946294676448330/957351400007821167658837948888772800000600767384123415009956940241643774815911357743355218685535050884816087722130566250000000$$

$$X[6] = 34247580160901081187586021290188766870560065311922811131766084949710537670417985137608711704102239384798861267438103286100439036/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101770447300$$

$$X[7] = 15779733984728642940623695549307939570093206380986421590089433098913999689805723525390712039667007143937311339395499193561643220218735393567130/382940560003128467063535179555509120000240306953649366003998277609665750992636454309734208747421402035392643550888522265000000000000000$$

$$X[8] = 65418469535219583617393153648880520611789605496085413197768854353915104443388399457324043203633370883544240456241892711930462382007899905003301/382940560003128467063535179555509120000240306953649366003998277609665750992636454309734208747421402035392643550888522265000000000000000$$

$$X[9] = 31723785266187377887698084783025023692273964499880091904573198586416341793427280156882947700111346495253548066841257835353131367278955268432809/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101777044530000000000000000$$

$$X[10] = 18936922732326945990340505913840548170933678492020322752127314201576265930083417891316948658093084236739060206615622674299256095180664687149 3/119668925000977645957354743611096600000075095923015426876249461753020547185198891971791940233569181360602011096526632078125000000000000$$

$$X[11] = 14301388516516295237746107040387934403165346857123739168965736318470901747250417463391302782452909222924831152601660995005766277924058 1/122540979201001109460331257457762918400076898225167797121279448835093040317643665379114946799174848651325645936284327124800000000$$

$$X[12] = 22260428117353686327922619782785401435944704356297490289953521826992900391891257107259491328786805404007745075366348448879547135134606475804321/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101777044530000000000000000$$

$$X[13] = 5113478978224662878660180342896018353997339130878585185985832923011487475425077201020657343257721403142670682620768$$
$$2014133845712857332993007001/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494$$
$$84280407078528710177704453000000000000000000$$
$$X[14] = 5099359035080864073652701767709930104232335114352305396231796402347732047350104373427707199015079455192407933413510095$$
$$5708703115965586405373427/382940560003128467063535179555509120000240306953649366003998277609665750992636454309734208747421400$$
$$2035392643550888522265000000000000000000$$
$$X[15] = 1571185766190737985453544034871560579934587124158662029759151335850782379990325302518081583765662102710320678558854290$$
$$4227997935865273261/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070780$$
$$5287101777044530000000$$
$$X[16] = 1499165541513645424927176409531341781686920435325991509232836137962639620055270624940979991126445614273767995563945000$$
$$8771338962421450267359 51/191470280001564233531767589777754560000120153476824683001999138804832875496318227154867104373710701$$
$$0176963217754442611325000000000000000$$
$$X[17] = 1794484524663588772451049200059878779573000663353914214243303112945489995420361970826021844604441929316463314808733594$$
$$77295934775104687/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785$$
$$287101777044530000000$$
$$X[18] = 4873646867334408578085338201505400313421543920691751871512217616672926113443178813352234221541187563320331675817011067$$
$$37624812378864900855744 3/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804$$
$$0407078528710177704453000000000000000$$
$$X[19] = 2596779239872184992119581231686552999604777603642776831529045285199978835622664570455733614823522733694376659530613651$$
$$8953199468396342 7/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785$$
$$2871017770445300000000$$
$$X[20] = 4242625788657116715109352376959225959149358006887039136594982827851464306879597354086506437894681483954367943452216683$$
$$5626910862789214284804 7/765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804$$
$$0707852871017770445300000000000000000$$
$$X[21] = 5831445665126325561776643679356524335898039438314943157606636490362161668180387584526042735856181742597308364569111570$$
$$92257413676371/2393378500019552919147094872221932000001501918460308537524989235060410943703977839435838804671383762721204020$$
$$219305326415625000$$
$$X[22] = 8752882997405505986917849101764140717808338659005164713697444724687984933976892544308884700694227968407609158906315430$$
$$11118947162255104861176 3/191470280001564233531767589777754560000120153476824683001999138804832875496318227154867104373710701$$
$$0176963217754442611325000000000000000$$
$$X[23] = 1199949626592929300620953397378937707125848168526868373195172206060265569103760125158352385716467510251392367088268800$$
$$8087254774706809486696043096232219 7/191470280001564233531767589777754560000120153476824683001999138804832875496318227154867104373710701$$
$$0437371070101769632177544426113250000000000000000000000$$
$$X[24] = 2147688196272486899944914528420120865953383202118294954197595104930467581344325859431468765162620881913264468197338156$$
$$33219366551/153176224001251386825414071822203648000096122781459746401599311043866300397054581723893683498968560814157057420$$
$$355408906$$

Minimum expansion coefficient is

$$21474734487128897408340292803060956686487118250276074980632783033687835054399372030022292213122967523990080151234490000000000/$$
$$765881120006256934127070359111018240000480613907298732007996555219331501985272908619468417494842804070785287101775044 53$$

Приложение Д

(справочное)

Пример выходного файла для случая анализа смеси $Ni : Co : Cu$ в соотношениях $5 : 5 : 1$

Precise format: Initial ISLAY is:

$$[15/4; 15/4] * x1[99/250; 99/250] * x2[70512821/1000000000; 70512821/1000000000] * x3 = [203/1000; 203/1000]$$
$$[64/25; 64/25] * x1[78/125; 78/125] * x2[70512821/1000000000; 70512821/1000000000] * x3 = [3/20; 3/20]$$
$$[34/25; 34/25] * x1[132/125; 132/125] * x2[16025641/250000000; 16025641/250000000] * x3 = [57/500; 57/500]$$
$$[67/100; 67/100] * x1[429/250; 429/250] * x2[14423077/250000000; 14423077/250000000] * x3 = [14/125; 14/125]$$
$$[41/100; 41/100] * x1[324/125; 324/125] * x2[51282051/1000000000; 51282051/1000000000] * x3 = [137/1000; 137/1000]$$
$$[3/10; 3/10] * x1[177/50; 177/50] * x2[51282051/1000000000; 51282051/1000000000] * x3 = [83/500; 83/500]$$
$$[1/5; 1/5] * x1[21/5; 21/5] * x2[8974359/200000000; 8974359/200000000] * x3 = [47/250; 47/250]$$
$$[2/25; 2/25] * x1[237/50; 237/50] * x2[19230769/500000000; 19230769/500000000] * x3 = [51/250; 51/250]$$
$$[1/100; 1/100] * x1[1299/250; 1299/250] * x2[8974359/200000000; 8974359/200000000] * x3 = [219/1000; 219/1000]$$
$$[0; 0] * x1[1467/250; 1467/250] * x2[19230769/500000000; 19230769/500000000] * x3 = [49/200; 49/200]$$
$$[0; 0] * x1[774/125; 774/125] * x2[19230769/500000000; 19230769/500000000] * x3 = [129/500; 129/500]$$
$$[1/100; 1/100] * x1[1503/250; 1503/250] * x2[14423077/250000000; 14423077/250000000] * x3 = [1/4; 1/4]$$
$$[1/20; 1/20] * x1[654/125; 654/125] * x2[16025641/250000000; 16025641/250000000] * x3 = [28/125; 28/125]$$
$$[7/100; 7/100] * x1[504/125; 504/125] * x2[70512821/1000000000; 70512821/1000000000] * x3 = [87/500; 87/500]$$
$$[9/100; 9/100] * x1[699/250; 699/250] * x2[8974359/100000000; 8974359/100000000] * x3 = [1/8; 1/8]$$
$$[13/100; 13/100] * x1[441/250; 441/250] * x2[102564103/1000000000; 102564103/1000000000] * x3 = [43/500; 43/500]$$
$$[1/5; 1/5] * x1[132/125; 132/125] * x2[16025641/125000000; 16025641/125000000] * x3 = [61/1000; 61/1000]$$
$$[3/10; 3/10] * x1[81/125; 81/125] * x2[16025641/100000000; 16025641/100000000] * x3 = [51/1000; 51/1000]$$
$$[11/25; 11/25] * x1[117/250; 117/250] * x2[41025641/200000000; 41025641/200000000] * x3 = [51/1000; 51/1000]$$
$$[61/100; 61/100] * x1[99/250; 99/250] * x2[16025641/62500000; 16025641/62500000] * x3 = [57/1000; 57/1000]$$
$$[83/100; 83/100] * x1[9/25; 9/25] * x2[78525641/250000000; 78525641/250000000] * x3 = [13/200; 13/200]$$
$$[11/10; 11/10] * x1[42/125; 42/125] * x2[397435897/1000000000; 397435897/1000000000] * x3 = [77/1000; 77/1000]$$

$$[137/100; 137/100] * x1[39/125; 39/125] * x2[31650641/62500000; 31650641/62500000] * x3 = [9/100; 9/100]$$
$$[33/20; 33/20] * x1[36/125; 36/125] * x2[77724359/125000000; 77724359/125000000] * x3 = [103/1000; 103/1000]$$
$$[93/50; 93/50] * x1[33/125; 33/125] * x2[47275641/62500000; 47275641/62500000] * x3 = [113/1000; 113/1000]$$
$$[97/50; 97/50] * x1[57/250; 57/250] * x2[923076923/1000000000; 923076923/1000000000] * x3 = [29/250; 29/250]$$
$$[191/100; 191/100] * x1[9/50; 9/50] * x2[1102564103/1000000000; 1102564103/1000000000] * x3 = [29/250; 29/250]$$
$$[193/100; 193/100] * x1[18/125; 18/125] * x2[1320512821/1000000000; 1320512821/1000000000] * x3 = [59/500; 59/500]$$
$$[2; 2] * x1[27/250; 27/250] * x2[1512820513/1000000000; 1512820513/1000000000] * x3 = [61/500; 61/500]$$
$$[211/100; 211/100] * x1[3/50; 3/50] * x2[109775641/62500000; 109775641/62500000] * x3 = [129/1000; 129/1000]$$
$$[11/5; 11/5] * x1[9/250; 9/250] * x2[987179487/500000000; 987179487/500000000] * x3 = [67/500; 67/500]$$
$$[56/25; 56/25] * x1[0; 0] * x2[2179487179/1000000000; 2179487179/1000000000] * x3 = [69/500; 69/500]$$
$$[221/100; 221/100] * x1[0; 0] * x2[476923077/200000000; 476923077/200000000] * x3 = [69/500; 69/500]$$
$$[57/25; 57/25] * x1[0; 0] * x2[2583333333/1000000000; 2583333333/1000000000] * x3 = [67/500; 67/500]$$
$$[19/10; 19/10] * x1[0; 0] * x2[2730769231/1000000000; 2730769231/1000000000] * x3 = [16/125; 16/125]$$
$$[169/100; 169/100] * x1[0; 0] * x2[2891025641/1000000000; 2891025641/1000000000] * x3 = [121/1000; 121/1000]$$
$$[149/100; 149/100] * x1[3/250; 3/250] * x2[2980769231/1000000000; 2980769231/1000000000] * x3 = [113/1000; 113/1000]$$
$$[129/100; 129/100] * x1[0; 0] * x2[3083333333/1000000000; 3083333333/1000000000] * x3 = [21/200; 21/200]$$
$$[11/10; 11/10] * x1[3/125; 3/125] * x2[626923077/200000000; 626923077/200000000] * x3 = [97/1000; 97/1000]$$
$$[47/50; 47/50] * x1[9/25; 9/25] * x2[798076923/250000000; 798076923/250000000] * x3 = [91/1000; 91/1000]$$
$$[79/100; 79/100] * x1[6/125; 6/125] * x2[796474359/250000000; 796474359/250000000] * x3 = [17/200; 17/200]$$
$$[13/20; 13/20] * x1[3/50; 3/50] * x2[796474359/250000000; 796474359/250000000] * x3 = [19/500; 19/500]$$
$$[11/20; 11/20] * x1[21/250; 21/250] * x2[316025641/100000000; 316025641/100000000] * x3 = [33/1000; 33/1000]$$
$$[9/20; 9/20] * x1[12/125; 12/125] * x2[3108974359/1000000000; 3108974359/1000000000] * x3 = [29/1000; 29/1000]$$
$$[19/50; 19/50] * x1[3/25; 3/25] * x2[764423077/250000000; 764423077/250000000] * x3 = [27/1000; 27/1000]$$

Optimal values:

$$X[1] = 140168753834777/2535136538307210$$
$$X[2] = 146032384608809/3802704807460815$$
$$X[3] = 517211600000/84504551276907$$

Minimum expansion coefficient is 4220563384156117/211261378192267500
Right hand part vector is:

$$b[1] = [77330992777748371/422522756384535000; 94213246314372839/422522756384535000]$$
$$b[2] = [6867160836171002/52815344548066875; 17954885056498121/105630689096133750]$$
$$b[3] = [9931616864881189/105630689096133750; 7076090124518653/52815344548066875]$$
$$b[4] = [19440710973377843/211261378192267500; 27881837741690077/211261378192267500]$$
$$b[5] = [49444490856369061/422522756384535000; 66326744392993529/422522756384535000]$$
$$b[6] = [7712206348940072/52815344548066875; 19644976082036261/105630689096133750]$$
$$b[7] = [35496575715990173/211261378192267500; 43937702484302407/211261378192267500]$$
$$b[8] = [38876757767066453/211261378192267500; 47317884535378687/211261378192267500]$$
$$b[9] = [84091356879900931/422522756384535000; 100973610416525399/422522756384535000]$$
$$b[10] = [95076948545898841/422522756384535000; 111959202082523309/422522756384535000]$$
$$b[11] = [25142436094724449/105630689096133750; 14681499739440283/52815344548066875]$$
$$b[12] = [24297390581955379/105630689096133750; 14258976983055748/52815344548066875]$$
$$b[13] = [43101985330911803/211261378192267500; 51543112099224037/211261378192267500]$$
$$b[14] = [8134729105324607/52815344548066875; 20490021594805331/105630689096133750]$$
$$b[15] = [44374217779754641/422522756384535000; 61256471316379109/422522756384535000]$$
$$b[16] = [3486978785094722/52815344548066875; 11194520954345561/105630689096133750]$$
$$b[17] = [17332761371144401/422522756384535000; 34215014907768869/422522756384535000]$$
$$b[18] = [131075333807299051/422522756384535000; 29989787343923519/422522756384535000]$$
$$b[19] = [131075333807299051/422522756384535000; 29989787343923519/422522756384535000]$$
$$b[20] = [15642670345606261/422522756384535000; 32524923882230729/422522756384535000]$$
$$b[21] = [19022852396682541/422522756384535000; 35905105933307009/422522756384535000]$$
$$b[22] = [24093125473296961/422522756384535000; 40975379009921429/422522756384535000]$$
$$b[23] = [7396480326573979/105630689096133750; 5808521855365048/52815344548066875]$$
$$b[24] = [35078717139294871/422522756384535000; 51960970675919339/422522756384535000]$$
$$b[25] = [39303944703140221/422522756384535000; 56186198239764689/422522756384535000]$$
$$b[26] = [20285756486146913/211261378192267500; 28726883254459147/211261378192267500]$$
$$b[27] = [20285756486146913/211261378192267500; 28726883254459147/211261378192267500]$$
$$b[28] = [51770698106328625281534454806687514574703005421841/105630689096133750]$$
$$b[29] = [10776662377650259/105630689096133750; 7498612880903188/52815344548066875]$$
$$b[30] = [46064308805292781/422522756384535000; 62946562341917249/422522756384535000]$$
$$b[31] = [60221153234019325281534454806687516264794030959981/105630689096133750]$$
$$b[32] = [12466753403188399/105630689096133750; 83436583936722585281534454806687]$$
$$b[33] = [12466753403188399/105630689096133750; 83436583936722585281534454806687]$$
$$b[34] = [60221153234019325281534454806687516264794030959981/105630689096133750]$$
$$b[35] = [228208930244541232112613781922675003126201979276635721126137819226750]$$
$$b[36] = [42684126754216501/422522756384535000; 59566380290840969/422522756384535000]$$
$$b[37] = [39303944703140221/422522756384535000; 56186198239764689/422522756384535000]$$
$$b[38] = [35923762652063941/422522756384535000; 52806016188688409/422522756384535000]$$
$$b[39] = [32543580600987661/422522756384535000; 49425834137612129/422522756384535000]$$
$$b[40] = [30008444062680451/422522756384535000; 46890697599304919/422522756384535000]$$
$$b[41] = [27473307524373241/422522756384535000; 44355561060997709/422522756384535000]$$
$$b[42] = [951842246787512/52815344548066875; 61242478777311411056306890961337500]$$
$$b[43] = [5502124192377421/422522756384535000; 22384377729001889/422522756384535000]$$
$$b[44] = [3812033166839281/422522756384535000; 20694286703463749/422522756384535000]$$
$$b[45] = [2966987654070211/422522756384535000; 19849241190694679/422522756384535000]$$

Fixed format: Initial ISLAY is:

$$[3.75000; 3.75000] * x1[0.39600; 0.39600] * x2[0.07051; 0.07051] * x3 = [0.20300; 0.20300]$$
$$[2.56000; 2.56000] * x1[0.62400; 0.62400] * x2[0.07051; 0.07051] * x3 = [0.15000; 0.15000]$$
$$[1.36000; 1.36000] * x1[1.05600; 1.05600] * x2[0.06410; 0.06410] * x3 = [0.11400; 0.11400]$$
$$[0.67000; 0.67000] * x1[1.71600; 1.71600] * x2[0.05769; 0.05769] * x3 = [0.11200; 0.11200]$$
$$[0.41000; 0.41000] * x1[2.59200; 2.59200] * x2[0.05128; 0.05128] * x3 = [0.13700; 0.13700]$$
$$[0.30000; 0.30000] * x1[3.54000; 3.54000] * x2[0.05128; 0.05128] * x3 = [0.16600; 0.16600]$$
$$[0.20000; 0.20000] * x1[4.20000; 4.20000] * x2[0.04487; 0.04487] * x3 = [0.18800; 0.18800]$$
$$[0.08000; 0.08000] * x1[4.74000; 4.74000] * x2[0.03846; 0.03846] * x3 = [0.20400; 0.20400]$$
$$[0.01000; 0.01000] * x1[5.19600; 5.19600] * x2[0.04487; 0.04487] * x3 = [0.21900; 0.21900]$$
$$[0.00000; 0.00000] * x1[5.86800; 5.86800] * x2[0.03846; 0.03846] * x3 = [0.24500; 0.24500]$$
$$[0.00000; 0.00000] * x1[6.19200; 6.19200] * x2[0.03846; 0.03846] * x3 = [0.25800; 0.25800]$$
$$[0.01000; 0.01000] * x1[6.01200; 6.01200] * x2[0.05769; 0.05769] * x3 = [0.25000; 0.25000]$$
$$[0.05000; 0.05000] * x1[5.23200; 5.23200] * x2[0.06410; 0.06410] * x3 = [0.22400; 0.22400]$$
$$[0.07000; 0.07000] * x1[4.03200; 4.03200] * x2[0.07051; 0.07051] * x3 = [0.17400; 0.17400]$$
$$[0.09000; 0.09000] * x1[2.79600; 2.79600] * x2[0.08974; 0.08974] * x3 = [0.12500; 0.12500]$$
$$[0.13000; 0.13000] * x1[1.76400; 1.76400] * x2[0.10256; 0.10256] * x3 = [0.08600; 0.08600]$$
$$[0.20000; 0.20000] * x1[1.05600; 1.05600] * x2[0.12820; 0.12820] * x3 = [0.06100; 0.06100]$$
$$[0.30000; 0.30000] * x1[0.64800; 0.64800] * x2[0.16025; 0.16025] * x3 = [0.05100; 0.05100]$$
$$[0.44000; 0.44000] * x1[0.46800; 0.46800] * x2[0.20512; 0.20512] * x3 = [0.05100; 0.05100]$$
$$[0.61000; 0.61000] * x1[0.39600; 0.39600] * x2[0.25641; 0.25641] * x3 = [0.05700; 0.05700]$$
$$[0.83000; 0.83000] * x1[0.36000; 0.36000] * x2[0.31410; 0.31410] * x3 = [0.06500; 0.06500]$$
$$[1.10000; 1.10000] * x1[0.33600; 0.33600] * x2[0.39743; 0.39743] * x3 = [0.07700; 0.07700]$$
$$[1.37000; 1.37000] * x1[0.31200; 0.31200] * x2[0.50641; 0.50641] * x3 = [0.09000; 0.09000]$$
$$[1.65000; 1.65000] * x1[0.28800; 0.28800] * x2[0.62179; 0.62179] * x3 = [0.10300; 0.10300]$$
$$[1.86000; 1.86000] * x1[0.26400; 0.26400] * x2[0.75641; 0.75641] * x3 = [0.11300; 0.11300]$$
$$[1.94000; 1.94000] * x1[0.22800; 0.22800] * x2[0.92307; 0.92307] * x3 = [0.11600; 0.11600]$$
$$[1.91000; 1.91000] * x1[0.18000; 0.18000] * x2[1.10256; 1.10256] * x3 = [0.11600; 0.11600]$$
$$[1.93000; 1.93000] * x1[0.14400; 0.14400] * x2[1.32051; 1.32051] * x3 = [0.11800; 0.11800]$$
$$[2.00000; 2.00000] * x1[0.10800; 0.10800] * x2[1.51282; 1.51282] * x3 = [0.12200; 0.12200]$$
$$[2.11000; 2.11000] * x1[0.06000; 0.06000] * x2[1.75641; 1.75641] * x3 = [0.12900; 0.12900]$$
$$[2.20000; 2.20000] * x1[0.03600; 0.03600] * x2[1.97435; 1.97435] * x3 = [0.13400; 0.13400]$$
$$[2.24000; 2.24000] * x1[0.00000; 0.00000] * x2[2.17948; 2.17948] * x3 = [0.13800; 0.13800]$$
$$[2.21000; 2.21000] * x1[0.00000; 0.00000] * x2[2.38461; 2.38461] * x3 = [0.13800; 0.13800]$$
$$[2.28000; 2.28000] * x1[0.00000; 0.00000] * x2[2.58333; 2.58333] * x3 = [0.13400; 0.13400]$$
$$[1.90000; 1.90000] * x1[0.00000; 0.00000] * x2[2.73076; 2.73076] * x3 = [0.12800; 0.12800]$$
$$[1.69000; 1.69000] * x1[0.00000; 0.00000] * x2[2.89102; 2.89102] * x3 = [0.12100; 0.12100]$$
$$[1.49000; 1.49000] * x1[0.01200; 0.01200] * x2[2.98076; 2.98076] * x3 = [0.11300; 0.11300]$$
$$[1.29000; 1.29000] * x1[0.00000; 0.00000] * x2[3.08333; 3.08333] * x3 = [0.10500; 0.10500]$$
$$[1.10000; 1.10000] * x1[0.02400; 0.02400] * x2[3.13461; 3.13461] * x3 = [0.09700; 0.09700]$$
$$[0.94000; 0.94000] * x1[0.36000; 0.36000] * x2[3.19230; 3.19230] * x3 = [0.09100; 0.09100]$$
$$[0.79000; 0.79000] * x1[0.04800; 0.04800] * x2[3.18589; 3.18589] * x3 = [0.08500; 0.08500]$$
$$[0.65000; 0.65000] * x1[0.06000; 0.06000] * x2[3.18589; 3.18589] * x3 = [0.03800; 0.03800]$$
$$[0.55000; 0.55000] * x1[0.08400; 0.08400] * x2[3.16025; 3.16025] * x3 = [0.03300; 0.03300]$$
$$[0.45000; 0.45000] * x1[0.09600; 0.09600] * x2[3.10897; 3.10897] * x3 = [0.02900; 0.02900]$$
$$[0.38000; 0.38000] * x1[0.12000; 0.12000] * x2[3.05769; 3.05769] * x3 = [0.02700; 0.02700]$$

Optimal values:

$$X[1] = 0.055290416005905$$
$$X[2] = 0.038402240511095$$
$$X[3] = 0.006120517678452$$

Minimum expansion coefficient is 0.019977922232027

Right hand part vector is:

$$b[1] = [0.183022077767972; 0.222977922232027]$$
$$b[2] = [0.130022077767972; 0.169977922232027]$$
$$b[3] = [0.094022077767972; 0.133977922232027]$$
$$b[4] = [0.092022077767972; 0.131977922232027]$$
$$b[5] = [0.117022077767972; 0.156977922232027]$$
$$b[6] = [0.146022077767972; 0.185977922232027]$$
$$b[7] = [0.168022077767972; 0.207977922232027]$$
$$b[8] = [0.184022077767972; 0.223977922232027]$$
$$b[9] = [0.199022077767972; 0.238977922232027]$$
$$b[10] = [0.225022077767972; 0.264977922232027]$$
$$b[11] = [0.238022077767972; 0.277977922232027]$$
$$b[12] = [0.230022077767972; 0.269977922232027]$$
$$b[13] = [0.204022077767972; 0.243977922232027]$$
$$b[14] = [0.154022077767972; 0.193977922232027]$$
$$b[15] = [0.105022077767972; 0.144977922232027]$$
$$b[16] = [0.066022077767972; 0.105977922232027]$$
$$b[17] = [0.041022077767972; 0.080977922232027]$$
$$b[18] = [0.031022077767972; 0.070977922232027]$$
$$b[19] = [0.031022077767972; 0.070977922232027]$$
$$b[20] = [0.037022077767972; 0.076977922232027]$$
$$b[21] = [0.045022077767972; 0.084977922232027]$$
$$b[22] = [0.057022077767972; 0.096977922232027]$$
$$b[23] = [0.070022077767972; 0.109977922232027]$$
$$b[24] = [0.083022077767972; 0.122977922232027]$$
$$b[25] = [0.093022077767972; 0.132977922232027]$$
$$b[26] = [0.096022077767972; 0.135977922232027]$$
$$b[27] = [0.096022077767972; 0.135977922232027]$$
$$b[28] = [0.098022077767972; 0.137977922232027]$$
$$b[29] = [0.102022077767972; 0.141977922232027]$$
$$b[30] = [0.109022077767972; 0.148977922232027]$$
$$b[31] = [0.114022077767972; 0.153977922232027]$$
$$b[32] = [0.118022077767972; 0.157977922232027]$$
$$b[33] = [0.118022077767972; 0.157977922232027]$$
$$b[34] = [0.114022077767972; 0.153977922232027]$$
$$b[35] = [0.108022077767972; 0.147977922232027]$$
$$b[36] = [0.101022077767972; 0.140977922232027]$$
$$b[37] = [0.093022077767972; 0.132977922232027]$$
$$b[38] = [0.085022077767972; 0.124977922232027]$$
$$b[39] = [0.077022077767972; 0.116977922232027]$$
$$b[40] = [0.071022077767972; 0.110977922232027]$$
$$b[41] = [0.065022077767972; 0.104977922232027]$$
$$b[42] = [0.018022077767972; 0.057977922232027]$$
$$b[43] = [0.013022077767972; 0.052977922232027]$$
$$b[44] = [0.009022077767972; 0.048977922232027]$$
$$b[45] = [0.007022077767972; 0.046977922232027]$$

Substitution:

$$
\begin{aligned}
Row[1] &= [0.222977922232027; 0.222977922232027] \\
Row[2] &= [0.165938038021529; 0.165938038021529] \\
Row[3] &= [0.116140072623944; 0.116140072623944] \\
Row[4] &= [0.103295930232021; 0.103295930232021] \\
Row[5] &= [0.122521550666913; 0.122521550666913] \\
Row[6] &= [0.152844928910782; 0.152844928910782] \\
Row[7] &= [0.172622131962343; 0.172622131962343] \\
Row[8] &= [0.186685257826334; 0.186685257826334] \\
Row[9] &= [0.200365584470272; 0.200365584470272] \\
Row[10] &= [0.225579751842377; 0.225579751842377] \\
Row[11] &= [0.238022077767972; 0.238022077767972] \\
Row[12] &= [0.231780280903789; 0.231780280903789] \\
Row[13] &= [0.204077384030542; 0.204077384030542] \\
Row[14] &= [0.159139737828638; 0.159139737828638] \\
Row[15] &= [0.112898079138677; 0.112898079138677] \\
Row[16] &= [0.075557051747926; 0.075557051747926] \\
Row[17] &= [0.052395530933290; 0.052395530933290] \\
Row[18] &= [0.042452628843451; 0.042452628843451] \\
Row[19] &= [0.043555522406842; 0.043555522406842] \\
Row[20] &= [0.050503804510780; 0.050503804510780] \\
Row[21] &= [0.061638322164705; 0.061638322164705] \\
Row[22] &= [0.076155123851864; 0.076155123851864] \\
Row[23] &= [0.090828861891950; 0.090828861891950] \\
Row[24] &= [0.106094738183386; 0.106094738183386] \\
Row[25] &= [0.117607987609924; 0.117607987609924] \\
Row[26] &= [0.121668826513779; 0.121668826513779] \\
Row[27] &= [0.119265360947315; 0.119265360947315] \\
Row[28] &= [0.120322647590549; 0.120322647590549] \\
Row[29] &= [0.123987518681151; 0.123987518681151] \\
Row[30] &= [0.129717052225589; 0.129717052225589] \\
Row[31] &= [0.135105494875369; 0.135105494875369] \\
Row[32] &= [0.137190121662258; 0.137190121662258] \\
Row[33] &= [0.136786899993253; 0.136786899993253] \\
Row[34] &= [0.141873485827426; 0.141873485827426] \\
Row[35] &= [0.121765511765330; 0.121765511765330] \\
Row[36] &= [0.111135376594580; 0.111135376594580] \\
Row[37] &= [0.101087397508655; 0.101087397508655] \\
Row[38] &= [0.090196232820806; 0.090196232820806] \\
Row[39] &= [0.080926580257803; 0.080926580257803] \\
Row[40] &= [0.085336373293491; 0.085336373293491] \\
Row[41] &= [0.065022077767972; 0.065022077767972] \\
Row[42] &= [0.057742246413278; 0.057742246413278] \\
Row[43] &= [0.052977922232027; 0.052977922232027] \\
Row[44] &= [0.047595834817837; 0.047595834817837] \\
Row[45] &= [0.044333286769957; 0.044333286769957]
\end{aligned}
$$