Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Томский государственный университет
систем управления и радиоэлектроники»

На правах рукописи

Ахунов Роман Раисович

Алгоритмы и комплекс программ для итерационного решения систем линейных алгебраических уравнений при анализе полосковых структур методом моментов

Специальность 05.13.18 – Математическое моделирование,
численные методы и комплексы программ

Диссертация на соискание учёной степени кандидата технических наук

Научный руководитель
к.т.н. Куксенко Сергей Петрович

Томск – 2018

ОГЛАВЛЕНИЕ

Введение 4
1 Обзор состояния проблемы 13
1.1 Математическое моделирование методом моментов 14
1.2 Метод моментов 16
1.3 Методы решения СЛАУ 21
1.3.1 Общие подходы 21
1.3.2 Прямые методы 24
1.3.3 Итерационные методы 26
1.4 Хранение разреженных матриц 36
1.5 Многократное решение СЛАУ 43
1.6 Основные результаты главы 45
2 Применение разреженного строчного формата хранения матрицы для ускорения решения СЛАУ 47
2.1 Аналитические оценки коэффициента сжатия данных 48
2.2 Разработка программ 51
2.3 Вычислительные эксперименты 62
2.4 Сравнение с результатами других авторов 70
2.5 Основные результаты главы 73
3 Ускорение многократного решения СЛАУ итерационными методами 74
3.1 Оценки ускорения по сравнению с прямым методом 75
3.1.1 Аналитические оценки 75
3.1.2 Вычислительные оценки 78
3.2 Выбор критерия переформирования предобусловливателя 86
3.2.1 Увеличение числа итераций выше заданного порога 86
3.2.2 Среднее арифметическое время решения 90
3.2.3 Средняя арифметическая сложность решения 97
3.3 Оптимальная очередность решения СЛАУ 105

3.4 Выбор предобусловливателя 110
3.5 Математическая модель с дополнительными параметрами..... 114
3.6 Основные результаты главы 115
4 Комплекс программ для итерационного решения СЛАУ..... 117
Заключение 125
Список литературы 127
Приложение А 147
Приложение Б..... 163
Приложение В..... 169
Приложение Г 190

ВВЕДЕНИЕ

Актуальность работы

С ростом сложности создаваемых устройств и протекающих в них процессов всё большую роль играет использование математического моделирования, поскольку натурное моделирование часто оказывается невозможным или очень затратным. Ярким примером являются широко проникающие во все сферы жизни общества современные радиоэлектронные устройства (РЭУ), для которых создание, оперативная модернизация и удешевление по запросам рынка требуют особенно тщательного математического моделирования посредством соответствующих программных продуктов. В общем случае, в основе математического моделирования РЭУ лежит численное решение уравнений Максвелла или, в частном случае, Пуассона/Лапласа. Оно часто выполняется методом моментов, сводящимся к решению системы линейных алгебраических уравнений (СЛАУ), на которое приходится значительная часть вычислительных затрат. Моделирование в диапазоне геометрических и электрофизических параметров структуры требует многократного решения СЛАУ и часто такого же увеличения вычислительных затрат, что является серьезной преградой для эффективного математического моделирования.

В развитие методов решения СЛАУ сделали большой вклад такие отечественные и зарубежные ученые, как В.В. Воеводин, С.К. Годунов, В.П. Ильин, Л.Ю. Колотилина, Г.И. Марчук, В.В. Радченко, А.А. Самарский, Е.Е. Тыртышников, M. Bebendorf, S Borm, C. Calgaro, J. Dongarra, D. Golub, W. Hackbusch, S. Karimi, Q. He, S. Rjasanow, Y. Saad, T. Topa, I. Tsukerman, H.A. van der Vorst и др. Известен ряд математических библиотек для решения СЛАУ. В развитие метода моментов большой вклад внесли R. Harrington, T. Sarkar, A. Djordjevic, и он широко используется в ряде программных продуктов. Однако остается неиспользованным ряд алгоритмических ресурсов, позволяющих уменьшить вычислительные затраты на решение СЛАУ, в

частности итерационными методами, при математическом моделировании методом моментов. Кроме того, для уменьшения вычислительных затрат также может быть использована специфика итерационного решения, а также изменений матрицы при многократном решении СЛАУ. Между тем с неуклонным ростом порядка и числа решаемых СЛАУ, из-за сложности задач моделирования, использование этих ресурсов весьма актуально.

Цель работы – разработка алгоритмов и комплекса программ для итерационного решения СЛАУ с уменьшенными вычислительными затратами при математическом моделировании методом моментов.

Для достижения этой цели надо решить следующие задачи:

1. Выполнить обзор способов уменьшения вычислительных затрат при математическом моделировании методом моментов.

2. Разработать алгоритмы и программы для однократного решения СЛАУ с уменьшенными вычислительными затратами за счет использования форматов хранения разреженных матриц.

3. Разработать алгоритмы и программы для многократного решения СЛАУ с уменьшенными вычислительными затратами за счет использования специфики итерационного решения СЛАУ.

4. Выполнить апробацию разработанных алгоритмов и комплекса программ на примере математического моделирования методом моментов полосковых структур.

Научная новизна

1. Предложена математическая модель с дополнительными параметрами (выбором очерёдности решения и матрицы предобусловливания), позволяющая уменьшить время моделирования полосковых структур методом моментов в диапазоне геометрических и электрофизических параметров структуры. (*Математическое моделирование – Область исследований 5. Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента*).

2. Разработаны алгоритмы, использующие разреженный строчный формат хранения матрицы и уменьшающие время ILU(0)-разложения. Разработаны алгоритмы многократного решения СЛАУ итерационным методом, отличающиеся адаптивным переформированием предобусловливателя на основании оценок средних арифметических значений времени и сложности решения (*Численные методы – Область исследований 3. Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий*).

3. Разработан комплекс программ, отличающийся наличием программных модулей для однократного и многократного решения СЛАУ (стабилизированным методом бисопряженных градиентов и квадратичным методом сопряженных градиентов).

(*Комплекс программ – Область исследований 4. Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента*).

Теоретическая значимость

1. Получены оценки максимального значения коэффициента сжатия для форматов хранения разреженных матриц.

2. Получены аналитические оценки максимально возможного ускорения многократного решения СЛАУ итерационным методом с предобусловливанием относительно метода исключения Гаусса.

3. Получены формулы для аналитической оценки усредненного ускорения многократного решения СЛАУ итерационным методом с предобусловливанием относительно прямого метода.

4. Сформулированы и доказаны теоремы об условиях существования минимума и убывания зависимости среднеарифметического времени решения ряда СЛАУ от их числа.

5. Получены формулы для оценки арифметической сложности LU-разложения, стабилизированного метода бисопряженных градиентов и квадратичного метода сопряженных градиентов с учетом программной реализации.

Практическая значимость

1. Показана перспективность использования итерационных методов с предобусловливанием для многократного решения СЛАУ.

2. Программно реализованы усовершенствованные алгоритмы ILU(0)-разложения и многократного решения СЛАУ итерационными методами с адаптивным переформированием предобусловливателя.

3. Использование разработанных комплексов программ позволило уменьшить вычислительные затраты на моделирование реальных полосковых структур методом моментов.

4. Оценена арифметическая сложность алгоритмов LU-разложения, стабилизированного метода бисопряженных градиентов и квадратичного метода сопряженных градиентов с учетом программной реализации.

5. Полученные оценки возможного ускорения многократного решения СЛАУ итерационными методами с предобусловливанием относительно прямого метода позволяют априорно выбрать наиболее подходящий метод.

Методология и методы исследования

В работе применены математическое моделирование, квазистатический подход, метод моментов, LU-разложение, метод Гаусса, ILU(0)-разложение, стабилизированный метод бисопряженных градиентов, квадратичный метод сопряженных градиентов, оценки вычислительной сложности.

Положения, выносимые на защиту

1. При математическом моделировании полосковых структур в диапазоне параметров методом моментов выбор очередности (прямой или обратной) при итерационном решении последовательности СЛАУ с предобусловливателем из СЛАУ, решаемой 1-й, позволяет ускорить решение 100 СЛАУ до 1,84 раза, а из 50-й – до 2,21 раза. *(Математическое*

моделирование – Область исследований 5. Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента.)

2. При решении СЛАУ итерационным методом хранение предобусловливателя в разреженном строчном формате позволяет уменьшать затраты не только оперативной памяти, но и времени, а переформирование предобусловливателя по адаптивным условиям минимизирует время решения последовательности СЛАУ. *(Численные методы – Область исследований 3. Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий.)*

3. Разработанный комплекс программ обеспечивает уменьшение вычислительных затрат при решении СЛАУ (стабилизированным методом бисопряженных градиентов и квадратичным методом сопряженных градиентов):

- при однократном решении СЛАУ, за счет ILU(0)-разложения и использования разреженного строчного формата с двумя временными векторами (для хранения статуса наличия ненулевых элементов строк и адреса столбцов этих элементов);

- при многократном решении СЛАУ, за счет адаптивного переформирования предобусловливателя на основании оценок средних арифметических значений времени и сложности, а также выбора матрицы предобусловливания и очередности решения СЛАУ. *(Комплекс программ – Область исследований 4. Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента.)*

Достоверность результатов подтверждена сравнением полученных результатов с результатами других авторов, использованием проверенных алгоритмов и численных методов, согласованностью результатов теоретических оценок и вычислительного эксперимента, а также использованием результатов на практике.

Использование результатов

1. ОКР «Разработка комплекса программных и технических средств для контроля информационных магистралей, обеспечения электромагнитной совместимости и исследования надёжности унифицированного ряда электронных модулей на основе технологии «система-на-кристалле» для систем управления и электропитания космических аппаратов связи, навигации и дистанционного зондирования Земли с длительным сроком активного существования», тема «УЭМ-ТУСУР», хоздоговор 95/10 от 24.11.2010 в рамках реализации Постановления 218 Правительства РФ, 2011–2013 гг. (акт внедрения)

2. ОКР «Разработка принципов построения и элементов системы автономной навигации с применением отечественной специализированной элементной базы на основе наногетероструктурной технологии для космических аппаратов всех типов орбит», тема «САН», хоздоговор 96/12 от 16.11.2012 в рамках реализации Постановления 218 Правительства РФ, 2013–2015 гг. (акт внедрения).

3. НИР «Выявление, исследование и реализация новых возможностей уменьшения времени многократного решения СЛАУ с частично изменяющейся матрицей в задачах вычисления емкостной матрицы произвольной системы проводников и диэлектриков», грант РФФИ 14-07-31267 мол_а, 2014–2015 гг. (акт внедрения).

4. НИР «Комплексные исследования по разработке алгоритмов, математического обеспечения и средств проектирования для создания новых элементов защиты и контроля вычислительных систем на основе модальных явлений», грант РФФИ 14-29-09254, 2014–2016 гг.

5. НИР «Комплексное обоснование возможностей создания модальной технологии помехозащиты критичной радиоэлектронной аппаратуры и совершенствования существующих и разработки новых помехозащитных устройств на её основе», грант РНФ 14-19-01232, 2014–2016 гг.

6. НИР «Разработка новых программных и аппаратных средств для моделирования и обеспечения электромагнитной совместимости радиоэлектронной аппаратуры» в рамках проектной части государственного задания в сфере научной деятельности 8.1802.2014/К, 2014–2016 гг.

7. НИР «Выявление новых подходов к совершенствованию обеспечения электромагнитной совместимости радиоэлектронной аппаратуры и моделирования систем активного зрения роботов» в рамках базовой части государственного задания в сфере научной деятельности 8.9562.2017/БЧ, 2017–2019 гг.

8. ПНИ «Теоретические и экспериментальные исследования по синтезу оптимальной сети высоковольтного электропитания для космических аппаратов» в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014–2020 годы», соглашение №14.574.21.0172 от 26.09.2017, 2017–2019 гг.

9. Учебный процесс Томского государственного университета систем управления и радиоэлектроники (акт внедрения).

10. Учебный процесс Национального исследовательского Томского государственного университета (акт внедрения).

Апробация результатов

Подготовка заявок и победа в конкурсах: темы «УЭМ-ТУСУР» и «САН» по Постановлению 218 Правительства РФ; грант РФФИ 14-07-31267 мол_а; грант РФФИ 14-29-09254; грант РНФ 14-19-01232; проектная часть госзадания 8.1802.2014/К, базовая часть госзадания 8.9562.2017/БЧ, ФЦП ИР, заявка 2017-14-576-0053-108.

Результаты докладывались и представлялись в материалах следующих конференций:

1. Всероссийская научно-техническая конференция студентов, аспирантов и молодых ученых «Научная сессия ТУСУР-2010», г. Томск, 2010, 2017 гг.

2. Международный симпозиум по электромагнитной совместимости и электромагнитной экологии, г. Санкт-Петербург, 2011 г.

3. Международная конференция по численному электромагнитному моделированию и оптимизации для ВЧ, СВЧ и терагерцовых приложений, Италия, 2014 г.

4. Международная конференция «Актуальные проблемы вычислительной и прикладной математики 2015», г. Новосибирск, 2015 г.

5. Международная конференция по прикладной физике, моделированию и компьютерам, Австрия, 2015 г.

6. Международная конференция по численному анализу и прикладной математике, Греция, 2015 г.

7. Международная конференция по моделированию и прикладной математике, Таиланд, 2015 г.

8. Международная научно-практическая конференция «Электронные средства и системы управления», г. Томск, 2015 г.

Публикации. Результаты исследований, представленных в диссертации, опубликованы в 37-и работах:

Публикация	Количество
Монография	1
Статьи в журналах из перечня ВАК	3
Статеи в журналах из перечня ВАК, индексируемых в Scopus	6
Публикации в зарубежных журналах	2
Статьи в отечественном журнале	2
Доклады в трудах зарубежных конференций	4
Доклады в трудах отечественных конференций	4
Свидетельства о регистрации программы для ЭВМ	15
ИТОГО:	37

В состав диссертации входят введение, 4 главы, заключение, список литературы из 164 наим. приложения на 53 с., в т.ч. 3 табл. Объём диссертации без приложений – 146 с., в т.ч. 64 рис. и 22 табл.

Личный вклад

Все результаты работы получены автором лично или при непосредственном его участии. Разработка исходных кодов программ, получение аналитических оценок и обработка результатов выполнены лично автором. Разработка алгоритмов, их исследование, анализ и обобщение полученных результатов выполнены совместно с С.П. Куксенко. Часть результатов получена совместно с соавторами публикаций.

Краткое содержание работы

Во Введении представлена краткая характеристика работы. В главе 1 выполнен обзор актуальных задач. В главе 2 проведен сравнительный анализ наиболее распространённых форматов хранения разреженных матриц, разработаны алгоритмы ILU(0)-разложения с применением разреженного строчного формата хранения матриц, проведен вычислительный эксперимент, подтверждающий эффективность использования разреженного формата, проведено сравнение результатов, полученных с помощью разработанных алгоритмов, показавшее согласованность с результатами других авторов. В главе 3 представлены алгоритмы многократного решения СЛАУ с изменяющейся матрицей итерационными методами с предобусловливанием и результаты вычислительных экспериментов, проведен комплексный анализ результатов вычислительного эксперимента. В главе 4 представлено описание разработанного комплекса программ. Далее приведен список литературы. В заключении подведены итоги работы. В Приложении А приведены исходные тексты программ. В Приложении Б приведены расчеты сложности алгоритмов LU-разложения, стабилизированного метода бисопряженных градиентов и квадратичного метода сопряженных градиентов. В Приложении В представлены копии свидетельств о регистрации программы для ЭВМ, а в Приложении Г – актов использования результатов работы.

1 ОБЗОР СОСТОЯНИЯ ПРОБЛЕМЫ

1.1 Математическое моделирование методом моментов

С ростом сложности создаваемых устройств и протекающих в них процессов всё большую роль играет использование математического моделирования, поскольку натурное моделирование часто оказывается невозможным или очень затратным. Ярким примером является сложность современных радиоэлектронных устройств (РЭУ), широко проникающих во все сферы жизни общества. Их создание, оперативная модернизация и удешевление по запросам рынка требуют особенно тщательного математического моделирования посредством соответствующих программных продуктов.В общем случае, в основе такого математического моделирования лежит численный анализ, требующий построения математической модели исследуемого объекта с помощью решения уравнений Максвелла. Основные численные методы, применяемые при таком моделировании: метод конечных разностей во временной области [1]; метод моментов (МоМ) [2]; метод конечных элементов [3]; метод конечного интегрирования [4]; метод матрицы линий передачи [5], а также, так называемые, «гибридные» методики [6].

Процесс построения математической модели можно разбить на следующие этапы [7]:

1. Постановка задачи – определение целей расчета, объема необходимой входной и выходной информации и допустимой погрешности.

2. Аналитическая обработка – формулировка уравнений Максвелла, начальных и граничных условий, описание параметров расчетной области, выбор метода решения, преобразование уравнений к виду, наиболее подходящему для данного численного метода.

3. Дискретизация (сегментация) модели – переход от непрерывных функций к дискретным и от функциональных уравнений к СЛАУ.

4. Вычисление элементов СЛАУ – численным интегрированием или дифференцированием в зависимости от используемого на предыдущем этапе метода.

5. Решение СЛАУ – выбор наиболее подходящего метода решения (прямой или итерационный).

6. Обработка результатов (вычисление требуемых характеристик) – вычисление поля, погонных матриц, отклика и прочих характеристик, и параметров исследуемого объекта/системы по данным из решения СЛАУ.

Важно, что перечисленные этапы не являются независимыми. Так, выбор метода дискретизации влияет на затраты на формирование СЛАУ и на размер и свойства получаемой СЛАУ, что, в свою очередь, определяет выбор метода её решения. От предыдущих этапов зависят и способы вычисления параметров и характеристик исследуемого объекта/системы. На втором этапе широко используются электродинамический и квазистатический (TEM-аппроксимация) подходы к решению уравнений Максвелла. На третьем этапе при решении задач ЭМС широко применяется метод моментов, использующий «поверхностный» подход, в соответствии с которым в качестве неизвестного выступает распределение плотности поверхностного тока на проводящих поверхностях исследуемого объекта/системы [8]. Найденный поверхностный ток рассматривается как источник, возбуждающий поле во всей расчетной области. Таким образом, при использовании МоМ неизвестная функция определена на поверхности, а не в объеме (как при использовании, например, методов конечных разностей и элементов), что уменьшает требования к вычислительным ресурсам. Получаемая при этом матрица СЛАУ является плотной и плохо обусловленной, что требует построения эффективных предобусловливателей для ускорения решения при использовании итерационных методов. Один из самых трудоёмких этапов приходится на решение СЛАУ. Поэтому актуально совершенствование методов решения СЛАУ.

В развитие методов решения СЛАУ сделали большой вклад такие отечественные и зарубежные ученые, как В.В. Воеводин, С.К. Годунов, В.П. Ильин, Л.Ю. Колотилина, Г.И. Марчук, В.В. Радченко, А.А. Самарский, Е.Е. Тыртышников, M. Bebendorf, S Borm, C. Calgaro, J. Dongarra, D. Golub,

W. Hackbusch, S. Karimi, Q. He, S. Rjasanow, Y. Saad, T. Topa, I. Tsukerman, H.A. van der Vorst и др.

1.2 Метод моментов

При проектировании требуется выполнять как одновариантный, так и многовариантный анализы, что может потребовать больших вычислительных затрат. Поэтому актуальна разработка алгоритмов, позволяющих уменьшить вычислительные затраты, а также соответствующих комплексов программ. При этом большая часть этих затрат приходится на решение СЛАУ. Примечательно, что специфика самого решения, а также изменений матрицы СЛАУ при многовариантном анализе, может быть использована для уменьшения вычислительных затрат, например, за счет итерационных методов.

При использовании метода моментов рассматривается операторное уравнение, для решения которого применяют систему N базисных функций в области определения оператора. Затем задается система N весовых, или тестовых, функций в области значений оператора и берется скалярное произведение с каждой функцией. В результате получается матрица размером $N \times N$. Следующем шагом вычисляются элементы вектора воздействий размером N, таким образом формируется СЛАУ. Далее решается СЛАУ. Из вектора решения СЛАУ вычисляются требуемые характеристики моделируемой структуры. Одной из таких задач является вычисление методом моментов ёмкостных матриц произвольных двумерных [9] и трехмерных [10] структур проводников и диэлектриков.

Методом моментов решаются уравнения Максвелла в интегральной форме в частотной области. Реализация такого подхода на практике становится нереальной из-за крайне высоких требований к ресурсам компьютера, поэтому прибегают к различным упрощениям. Уравнения Максвелла сводят к частному случаю эллиптического дифференциального уравнения с частными производными, известному как уравнение Пуассона/Лапласа (полагается, что все заряды и токи сосредоточены на поверхности проводников, как в

случае бесконечной проводимости проводников) с соответствующими граничными условиями. Рассмотрим последнее моделирование подробнее [11].

Задача вычисления ёмкостной матрицы сводится к решению СЛАУ

$$\mathbf{S}\,\boldsymbol{\sigma} = \mathbf{V}, \tag{1.1}$$

где **S** – матрица, связывающая плотность заряда элементов дискретизации на проводниках и диэлектрических границах, составляющих вектор $\boldsymbol{\sigma}$, с потенциалами этих элементов, составляющих вектор **V**. Система решается N_{cond} раз (N_{cond} – число проводников в системе, не считая опорного).

Для вычисления элементов матрицы **S** границы проводник-диэлектрик и диэлектрик-диэлектрик делятся на подынтервалы, характеризуемые величинами: x_n – X координата центра подынтервала n; y_n – Y координата центра подынтервала n; d_n – длина подынтервала n; θ_n – угол, образуемый подынтервалом n с положительным направлением оси координат X; ε_n – диэлектрическая проницаемость около n-го подынтервала проводник-диэлектрик; ε_n^+ и ε_n^- – диэлектрические проницаемости, соответственно, на положительной (к которой указывает $\underline{n}_n$) и отрицательной (от которой указывает $\underline{n}_n$) сторонах n-го подынтервала диэлектрик-диэлектрик, где $\underline{n}_n$ – единичный вектор, проведённый нормально от центра n-го подынтервала. Примеры значений этих параметров показаны в рамках на рисунке 1.1.

Центру подынтервала n соответствует вектор $\underline{r}_n$, определяемый как

$$\underline{r}_n = \underline{x}x_n + \underline{y}y_n, \tag{1.2}$$

где $\underline{x}$ и $\underline{y}$ – единичные векторы в направлениях X и Y, соответственно. Аналогично, вектор $\underline{r}'_n$ подынтервала, по которому ведётся интегрирование, и вектор $\underline{r}_n$ его образа относительно бесконечной плоскости определяются как

$$\underline{r}'_n = \underline{x}x'_n + \underline{y}y'_n; \tag{1.3}$$

$$r_n = \underline{x}x'_n - \underline{y}y'_n; \tag{1.4}$$

где

$$x'_n = x_n + t\cos(\theta_n); \quad (1.5)$$

$$y'_n = y_n + t\sin(\theta_n), \quad (1.6)$$

где t – текущее расстояние от центра (x_n, y_n) подынтервала вдоль этого подынтервала. В результате, исходное интегрирование по двум декартовым координатам для текущей точки и её образа сводится к интегрированию по одной переменной t (поскольку угол $\theta = \theta_n$ = const для выбранного подынтервала).

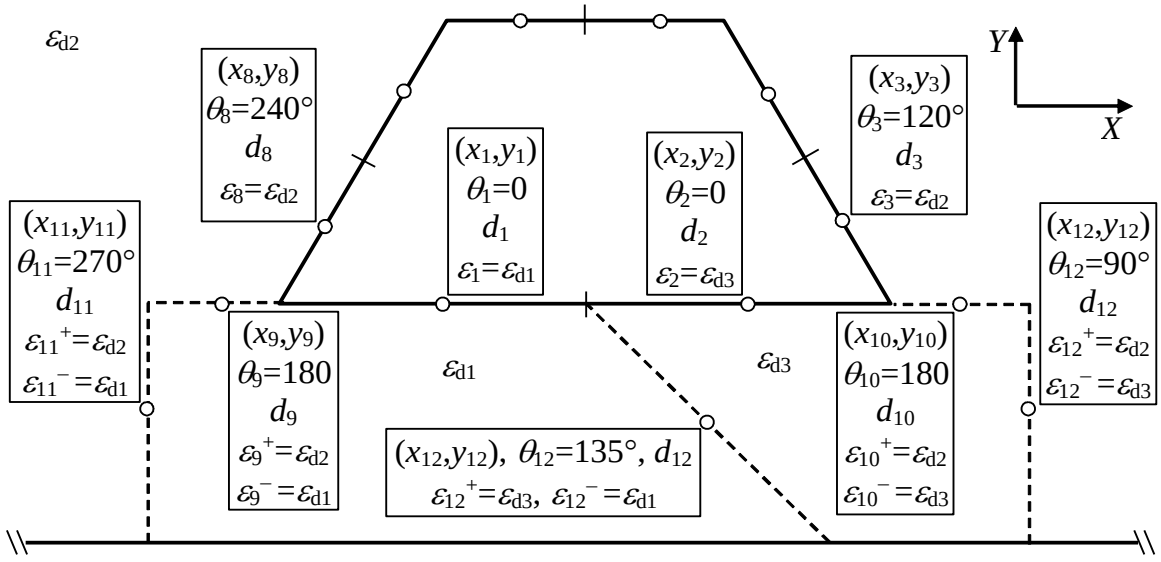


Рисунок 1.1 – Схема двумерной дискретизации

Порядок дискретизации таков. Сначала дискретизируются границы проводник-диэлектрик и полученным подынтервалам проводник-диэлектрик присваиваются номера с 1 по N_c. Если есть другие проводники, которые всегда находятся под нулевым потенциалом, то все они дискретизируются как обычные проводники. Затем, дискретизируются границы диэлектрик-диэлектрик, и полученным подынтервалам диэлектрик-диэлектрик присваиваются номера с N_c+1 по N.

Для строк матрицы **S** с номерами $m = 1 \ldots N_c$, соответствующими подынтервалам проводник-диэлектрик, элементы S_{mn} вычисляются по формуле

$$S_{mn} = \frac{1}{2\pi\varepsilon_0}\left(\text{iflg}\cdot\hat{I}_{mn} - I_{mn}\right), \begin{cases} m=1...Nc \\ n=1...N \end{cases}, \tag{1.7}$$

где iflg – коэффициент, здесь и далее равен единице, если есть бесконечная плоскость и равен нулю, если её нет;

$$I_{mn} = \int_{\ell_n} \ln\left|\underline{r}_m - \underline{r}'_n\right| d\ell' \ ; \tag{1.8}$$

$$\hat{I}_{mn} = \int_{\ell_n} \ln\left|\underline{r}_m - \underline{r}_n\right| d\ell' \ . \tag{1.9}$$

Для строк матрицы **S** с номерами $m = (N_c+1)\ldots N$, соответствующими подынтервалам диэлектрик-диэлектрик, элементы вычисляются по формулам

$$S_{mn} = \frac{1}{2\pi\varepsilon_0}\left(I_{mn} - \text{iflg}\cdot\hat{I}_{mn}\right), \begin{cases} m = (Nc+1)...N \\ n = 1...N \end{cases}, m \neq n; \tag{1.10}$$

$$S_{mm} = \frac{1}{2\pi\varepsilon_0}\left(I_{mm} - \text{iflg}\cdot\hat{I}_{mm}\right) + \frac{1}{2\varepsilon_0}\frac{\varepsilon_m^+ + \varepsilon_m^-}{\varepsilon_m^+ - \varepsilon_m^-}, m = (Nc+1)...N, \tag{1.11}$$

где

$$I_{mn} = \int_{\ell_n} \frac{\underline{r}_m - \underline{r}'_n}{\left|\underline{r}_m - \underline{r}'_n\right|^2}\underline{n}_m d\ell' \ ; \tag{1.12}$$

$$\hat{I}_{mn} = \int_{\ell_n} \frac{\underline{r}_m - \underline{r}_n}{\left|\underline{r}_m - \underline{r}_n\right|^2}\underline{n}_m d\ell' \ . \tag{1.13}$$

Следующим шагом является подстановка формул (1.2)–(1.6) в (1.7)–(1.13).

Исходя из приведенных формул, можно сделать вывод о структуре матрицы **S**. Условно матрицу **S** можно разделить на подматрицы, причем каждая подматрица будет соответствовать определённым подынтервалам. Для каждой подматрицы определены свои формулы определения элементов (согласно (1.7), (1.10) и (1.11)), как показано на рисунке 1.2.

Подробный расчет для каждой области матрицы **S** можно найти в [11]. Из приведенных данных следует, что матрица **S** является плотной. Так же можно сделать выводы о влиянии изменений параметров структуры на матрицу **S**. Так, при изменении диэлектрической проницаемости ε (согласно (1.11)) будут изменяться элементы матрицы **S** на главной диагонали нижней правой подматрицы с индексами элементов матрицы больше N_c (область 4 на

рисунке 1.2). А при изменении линейных размеров проводника могут изменяться все элементы матрицы **S**.

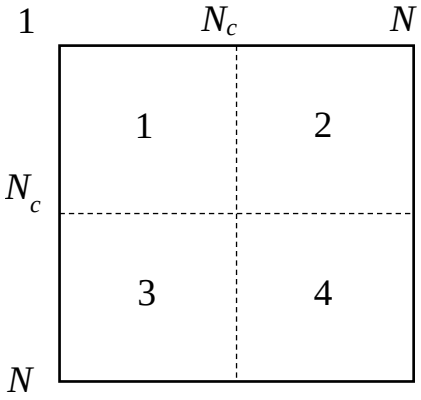


Рисунок 1.2 – Структура матрицы **S**

После вычисления вектора $\boldsymbol{\sigma}$ N_{cond} раз, можно рассчитать матрицу ёмкостей. Каждый элемент ёмкостной матрицы определяется следующим образом:

$$C_{ij} = \int_{S_i} \frac{\varepsilon(\underline{r})}{\varepsilon_0} \sigma^{(j)}(\underline{r}) da_i ,$$

где индекс i относится к проводнику, по поверхности которого производится интегрирование, а индекс j над плотностью заряда означает распределение плотности заряда, когда j-й проводник находился под потенциалом одного вольта, а все остальные – под потенциалом нуля.

На практике часто требуется, для получения точных результатов, учет частотной зависимости ε диэлектриков [12]. При этом решение СЛАУ выполняется для каждой частотной точки из диапазона, что увеличивает общее время вычисления пропорционально числу частотных точек. Однако, как сказано выше, изменяются не все, а только определенные элементы матрицы СЛАУ, поэтому можно использовать этот ресурс для ускорения решения СЛАУ. Так, описаны [13] усовершенствования алгоритма вычисления емкостных матриц одной и той же структуры при многократном изменении значения диэлектрической проницаемости и рассмотрено [14–16] использование алгоритма, использующего блочное LU-разложение, для вычисления

емкостных матриц. Эти усовершенствования апробированы в системе TALGAT [17] на практических задачах. Чем меньше изменяющихся элементов, тем быстрее повторные вычисления.

Однако чаще необходимо моделирование при изменении размеров структуры, что приводит к изменению элементов матрицы, расположенных в произвольных местах [18]. В данном случае алгоритм блочного LU-разложения неприменим. В качестве первого шага в этом направлении было исследовано уменьшение нормы невязки 10-кратного решения систем линейных уравнений, полученных при малом изменении нескольких параметров структуры [19].

1.3 Методы решения СЛАУ

1.3.1 Общие подходы

Решение СЛАУ, вида (1.14) или в матричной форме (1.15), заключается в определении вектора неизвестных **x**.

$$\left.\begin{array}{c} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2 \\ \vdots \\ a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N \end{array}\right\}, \tag{1.14}$$

$$\mathbf{Ax} = \mathbf{b}, \tag{1.15}$$

где N – порядок матрицы, $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ – вектор неизвестных, $\mathbf{b} = (b_1, b_2, \ldots, b_N)$ – вектор свободных членов, $\mathbf{A} = (a_{ij})$ ($i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, N$) – квадратная матрица коэффициентов.

Для анализа матрицы СЛАУ применяют портреты. В случае разреженной матрицы портретом называют множество пар индексов (i, j), таких, что $a_{ij} \neq 0$: $P_{\mathbf{A}} = \{(i, j) \mid a_{ij} \neq 0\}$ [20]. Для анализа плотных матриц можно использовать несколько множеств, ограниченных значениями, например: $P_{\mathbf{A}}(c_1, c_2) = \{(i, j) \mid c_1 < a_{ij} < c_2\}$. Также портреты можно представлять графически, присвоив каждому множеству свой цвет.

Методы решения СЛАУ делят на две больших группы. Это прямые (или точные) и итерационные методы [21]. К прямым методам решения СЛАУ относятся методы, которые позволяют получить решение за конечное число элементарных операций [22]. Однако они имеют ряд недостатков: накапливание погрешности решения при плохой обусловленности СЛАУ; как правило, вычислительные затраты пропорциональны N^3, что существенно ограничивает использование таких методов. Несмотря на перечисленные недостатки, прямые методы широко используются на практике [23], что связано с их сравнительной простотой и универсальностью (т.е. они подходят для решения широкого класса задач) [21]. Итерационные методы, в отличие от прямых, основаны на последовательном приближении к решению СЛАУ [24]. С точки зрения вычислительных затрат итерационные методы обычно пропорциональны $N_{it}N^2$ (N_{it} – количество итераций). Из этого следует, что при $N_{it}<N$ (а это часто имеет место) итерационные методы эффективней прямых. Еще одним преимуществом итерационных методов является меньшая погрешность при вычислении плохо обусловленных СЛАУ [21].

При решении СЛАУ также необходимо соблюдать корректность постановки задачи её решения [24]. Задача является корректной, если решение существует и единственно. Для подтверждения существования единственного решения СЛАУ необходимо и достаточно условие неравенства нулю определителя (детерминанта) матрицы **A**. Кроме того, СЛАУ должна быть устойчива к малым возмущениям входных данных [24]. Для оценки устойчивости используют число обусловленности:

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|\times\|\mathbf{A}^{-1}\|, \qquad (1.16)$$

а также «естественное» число обусловленности ($\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\|\times\|\mathbf{b}\| / \|\mathbf{x}\|$) [25]. Обычно, если СЛАУ плохо обусловлена, то $\mathrm{cond}(\mathbf{A}) >> 1$. На практике вычисление обратной матрицы ($\mathbf{A}^{-1}$) очень трудоёмкий процесс, поэтому используют приближенные оценки обусловленности. Один из методов оценки –

замена $\|\mathbf{A}^{-1}\| \approx \max_k \left\|\mathbf{w}_i\right\| / \left\|\mathbf{y}_i\right\|$, где $\mathbf{y}_i$ – случайный вектор, $i = 1, 2, \ldots, k$; $\mathbf{w}_i$ – решение системы $\mathbf{A}\mathbf{w}_i = \mathbf{y}_i$ [25, 26].

С ростом числа обусловленности и порядка матрицы растёт погрешность решения из-за представления чисел с плавающей запятой конечным числом разрядов. Одним из важных следствий этого является невозможность получения корректного решения СЛАУ прямым методом (исключения Гаусса) при росте числа обусловленности матриц больших порядков и малой разрядности представления чисел. Поэтому применяют итерационные методы решения СЛАУ, поскольку в них погрешность из-за конечного числа разрядов много меньше, чем в методе Гаусса, т.к. она не накапливается, а определяется только последней итерацией и не зависит от их числа [8]. Поэтому решение с заданной точностью при росте числа обусловленности матрицы достигается просто увеличением числа итераций. Очевидно, что снижение числа обусловленности матрицы позволяет снизить время решения с заданной точностью за счёт уменьшения числа итераций. Кроме того, можно даже уменьшить число разрядов представления чисел с плавающей запятой для снижения времени вычисления и требований к памяти компьютера [27].

Следует отметить, что в настоящее время активно развиваются программные средства, позволяющие ускорить решение СЛАУ. Ускорение происходит за счет параллельных вычислений, которые реализуются за счет многоядерных процессоров и многопроцессорных станций, кластеров, а также графических процессоров. Наиболее популярные программные средства: Open MP – открытый стандарт для распараллеливания программ [28]; MPI – программный интерфейс для передачи информации, который позволяет обмениваться сообщениями между процессами [29]; CUDA – программно-аппаратная архитектура параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы Nvidia [30]; GPGPU – техника использования графического процессора видеокарты для выполнения расчё-

тов для общих вычислений [31]; POSIX Threads – стандарт POSIX реализации потоков (нитей) выполнения [32]; Windows API – общее наименование целого набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows [33] и др.

Так же активно развиваются программные библиотеки для решения СЛАУ. Наиболее популярные программные библиотеки: LAPACK – библиотека с открытым исходным кодом, содержащая методы для решения основных задач линейной алгебры [34]; Automatically Tuned Linear Algebra Software (ATLAS) – программная библиотека для линейной алгебры [35]; Intel Math Kernel Library (Intel MKL) – библиотека оптимизированных математических подпрограмм [36]; Eigen – библиотека линейной алгебры для C++ с открытым исходным кодом; Basic Linear Algebra Subprograms (BLAS) – стандарт интерфейса программирования приложений для создания библиотек, выполняющих основные операции линейной алгебры [37], написанный на шаблонах и предназначенный для векторно-матричных вычислений и связанных с ними операций [38] и др.

1.3.2 Прямые методы

Исторически первым прямым методом является метод, основанный на исключении неизвестных (метод Гаусса). В настоящее время разработано большое число различных методов, основанных на методе Гаусса [39], далее рассмотрены наиболее распространенные из них.

1.3.2.1 Метод Гаусса

Это один из самых важных и распространенных прямых методов решения СЛАУ [40], который также называют методом последовательного исключения неизвестных. Он известен в различных вариантах, которые алгебраически тождественны, уже более 2000 лет [24, 41, 42].

Существует множество методов, которые являются модификациями метода Гаусса. Они отличаются порядком исключения неизвестных (этап

прямого хода), способами, уменьшающими погрешность вычислений. Например, такие методы как метод Гаусса-Жордана [40, 43], метод единственного деления [39, 44], с выбором главного элемента [39, 45], метод основанный на LU-разложения [24, 44, 46] и др. Из перечисленных методов широкое распространение получил метод, основанный на LU-разложении.

1.3.2.2 Метод LU-разложения

Метод LU-разложения получен из метода Гаусса путем обобщения операций на этапе прямого хода [23, 24]. Метод LU-разложения заключается в представлении матрицы **A** в виде произведения двух матриц, т.е. **A** = **LU**, где **L** – нижняя треугольная матрица, а **U** – верхняя треугольная матрица с единицами на главной диагонали. При выполнении LU-разложения метод Гаусса разбивается на два этапа вычисления: LU-разложения матрицы **A** и вектора решения [23]. Этот подход позволяет независимо от **b** разложить матрицу **A**, а затем найти решение системы. Если требуется решить несколько систем уравнений с фиксированной матрицей **A** и различными векторами $\mathbf{b}_1$, $\mathbf{b}_2$, …, $\mathbf{b}_p$, то этап LU-разложения проводится один раз. Затем p раз проводятся вычисления второго этапа. Этот подход – главное преимущество перед методом Гаусса, позволяющее снизить число арифметических операций [23, 47].

Существуют разные версии алгоритма метода LU-разложения, которые получены путем изменения порядка обработки элементов [48, 49]. Ниже приведены некоторые из них [46]:

LU-разложение (*ikj*-версия):

```
1       Для i = 2, …, N
2           Для k = 1, …, i – 1
3               a_{i,k} = a_{i,k} / a_{k,k}
4               Для j = k + 1, …, N
5                   a_{i,j} = a_{i,j} – a_{i,k}a_{k,j}
6               Увеличить j
7           Увеличить k
8       Увеличить i
```

LU-разложение (*kji*-версия):

```
1   Для k = 1, …, N–1
2       Для i = k+1, …, N
3           a_{i,k} = a_{i,k} / a_{k,k}
4       Увеличить i
5       Для j = k + 1, …, N
6           Для i = k+1, …, N
7               a_{i,j} = a_{i,j} – a_{i,k}a_{k,j}
8           Увеличить i
9       Увеличить j
10  Увеличить k
```

LU-разложение (*jki*-версия):

```
1   Для j = 1, …, N
2       Для k = 1, …, j–1
3           Для i = k+1, …, N
4               a_{i,j} = a_{i,j} – a_{i,k}a_{k,j}
5           Увеличить i
6       Увеличить k
7       Для i = j + 1, …, N
8           a_{i,k} = a_{i,k} / a_{k,k}
9       Увеличить i
10  Увеличить j
```

По арифметической сложности эти три версии LU-разложения не отличаются, но очередность важна для компьютерной реализации [49, 50].

Применение LU-разложения возможно не только для решения СЛАУ, но и для следующих важных математических операций:

- определение обратной матрицы;
- нахождение собственных значений матрицы;
- методы неявного предобусловливания итерационных методов, такие как: ILU(0), ILUT и др. [46, 49] (подробнее см. гл. 1.3.3.2).

1.3.3 Итерационные методы

Моделирование всё более сложных научно-технических задач в различных областях знаний требует решения СЛАУ с всё матрицей большего порядка [24]. Прямые методы, несмотря на их простоту и универсальность,

не позволяют эффективно решать СЛАУ с большими порядками, из-за возрастающих вычислительных затрат на реализацию. Поэтому в последнее время широко применяются итерационные методы. К настоящему времени разработано много различных итерационных методов. Ранние итерационные методы получили название классических, первые из них были опубликованы еще в XIX веке в работах Гаусса-Зейделя [51]. Они основаны на расщеплении исходной матрицы и циклическом изменении вектора решения таким образом, чтобы уменьшить норму вектора невязки (**r** = **Ax**–**b**). Такая методика получила название релаксации [20]. Эти методы редко применяются на практике из-за ряда недостатков, например медленной или неустойчивой сходимости и невозможности применения предобусловливания [46], и поэтому не рассматриваются в данной работе. Другое направление развития итерационных методов, используемое и по сей день, основано на подпространствах Крылова. Предпосылкой его стали проекционные методы, возникшие в начале 30-х годов XX века [51]. Ниже рассмотрены принципы проекционных методов решения СЛАУ.

1.3.3.1 Проекционные методы

Класс проекционных методов охватывает методы, использующие теорию линейных подпространств. Так, для решения систем вида (1.14), задают K и L – два m-мерных линейных подпространства пространства R^N. Тогда решение сводится к нахождению такого вектора $\tilde{\mathbf{x}} \in K$, чтобы выполнялось условие $\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} \perp L$, т.е. выполнялось условие: $\forall l \in L : (\mathbf{A}\tilde{\mathbf{x}}, l) = (\mathbf{b}, l)$, где l – вспомогательный вектор называемый условием Петрова-Галёркина [20]. Если задать начальное приближение $\mathbf{x}_0$, тогда можно представить $\tilde{\mathbf{x}} = \mathbf{x}_0 + \boldsymbol{\delta}$, и постановку задачи сформулировать следующим образом: необходимо найти $\boldsymbol{\delta} \in K$, чтобы выполнялось условие $\forall l \in L : (\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}, l) = 0$ [46]. Далее необходимо ввести матричные базисы в подпространствах K и L, например, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$ из базисных векторов пространства K и $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ из L. После некоторых преобразований получается

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{V}(\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{r}_0. \quad (1.17)$$

Из этого условия сразу вытекает важное требование: подпространства K и L и их базисы выбираются так, чтобы матрица $\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V}$ либо была малого порядка, либо имела простую структуру, удобную для обращения [20]. Общий вид алгоритма любого проекционного метода:

1 Выбрать пару подпространств K и L
2 Построить для K и L базисы $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$ и $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$
3 Для $i = 1, 2, \ldots$, пока не достигнута требуемая точность,
4 $\quad \mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_{i-1}$
5 $\quad \mathbf{y} = (\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{r}_i$
6 $\quad \mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{V}\mathbf{y}$
7 Увеличить i

Выбор пространств K и L будет определять вычислительную схему метода. Наиболее простой случай, когда подпространства K и L одномерны. Тогда соотношение (1.17) принимает вид:

$$x_{k+1} = x_k + y_k \boldsymbol{V}_k. \quad (1.18)$$

Примером являются методы: наискорейшего спуска [20], наискорейшего уменьшения невязки [52] и Гаусса-Зейделя [51]. Зачастую эти методы обладают медленной сходимостью [52].

Выделяются методы, для которых в качестве подпространства K выбирается подпространство Крылова. Такие методы называют методами крыловского типа или методами подпространств Крылова [46]. Такие подпространства предложил А.Н. Крылов в 1931 г. [53]. Подпространством Крылова размерности m называют линейное пространство

$$K_m(\mathbf{A}, \mathbf{v}) = span\{\mathbf{v}, \mathbf{A}\mathbf{v}, \mathbf{A}^2\mathbf{v}, \ldots, \mathbf{A}^{m-1}\mathbf{v}\}. \quad (1.19)$$

В качестве вектора $\mathbf{v}$ обычно выбирается невязка начального приближения $\mathbf{r}_0$ ($\mathbf{r}_0 = \mathbf{b}–\mathbf{A}\mathbf{x}_0$). Выбор подпространства L и способ построения базисов подпространства определяют вычислительную схему метода [52].

1.3.3.2 Предобусловливание

Однако проекционные методы не лишены недостатков. Некоторые методы медленно сходятся [52], у других может происходить стагнация и сбой,

что не дает возможности получить решение. Наличие таких недостатков объясняется тем, что скорость сходимости сильно зависит от спектральных свойств матрицы системы, от обусловленности и даже от начального приближения [52]. Для ускорения сходимости применяют предобусловливание.

Условно методы предобусловливания делят на два вида: явные и неявные [54]. Существует различные методы предобусловливания, такие как: предобусловливание Якоби и Гаусса-Зейделя [52]; неполное LU-разложение [46]; полиноминальное предобусловливание [20] и др.

1.3.3.2.1 Явное предобусловливание

Предобусловливание заключается в преобразовании системы $\mathbf{Ax} = \mathbf{b}$ к алгебраически эквивалентной системе [52]

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{1.20}$$

где $\tilde{\mathbf{A}} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}$, $\tilde{\mathbf{x}} = \mathbf{M}_2\mathbf{x}$, $\tilde{\mathbf{b}} = \mathbf{M}_1^{-1}\mathbf{b}$, матрицы $\mathbf{M}_1$, $\mathbf{M}_2$ – невырожденные.

Если $\mathbf{M}_1 = \mathbf{E}$ и $\mathbf{M}_2 \neq \mathbf{E}$, то предобусловливание называется правым, если $\mathbf{M}_1 \neq \mathbf{E}$ и $\mathbf{M}_2 = \mathbf{E}$ – то левым, а если $\mathbf{M}_1 \neq \mathbf{E}$ и $\mathbf{M}_2 \neq \mathbf{E}$, то предобусловливание называется двухсторонним. Чаще всего используют левое предобусловливание [52], поэтому далее будет рассматриваться только оно. При левом предобусловливании система (1.20) будет выглядеть следующим образом:

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}. \tag{1.21}$$

Основные требования к матрице $\mathbf{M}$: «близость» к матрице $\mathbf{A}$, легкие вычислимость и обратимость [52]. Выбор $\mathbf{M} = \mathbf{A}$ приводит систему $\mathbf{Ax} = \mathbf{b}$ к виду $\mathbf{Ex} = \mathbf{A}^{-1}\mathbf{x}$ ($\mathbf{E}$ – единичная матрица), что дает возможность сразу получить решение, однако это не практично, т.к. это требует нахождения $\mathbf{A}^{-1}$, что является решением системы $\mathbf{Ax} = \mathbf{b}$ [54].

Явное предобусловливание основано на нахождении матрицы $\mathbf{M}^{-1}$. Большое распространение получили методы минимизации функционала. Основная идея таких методов сводится к минимизации:

$$\min_{\mathbf{B}\in S} \left\|\mathbf{AB}-\mathbf{E}\right\|_F,$$

где S – множество разреженных матриц, а $\left\|\cdot\right\|_F$ – норма Фробениуса, которая находится следующим образом:

$$\left\|\mathbf{AB}-\mathbf{E}\right\|_F^2 = \sum_{j=1}^{n}\left\|\mathbf{Ab}_j-\mathbf{e}_j\right\|_2^2,$$

где $\mathbf{b}_j$ – j-й столбец матрицы **B**. Задача нахождения матрицы **B** (обратной к матрице **A**) сводится к решению N независимых линейных задач наименьших квадратов [55].

Такие алгоритмы трудоемки, но имеют высокий потенциал распараллеливания [52, 56]. Самые известные методы минимизации функционала это – SPAI [57], AINV (подробно описан в [58]) и др. [59–61].

Один из первых способов предобусловливания основан на итерационном методе Якоби [46]. В этом случае матрица предобусловливания **M** – диагональная матрица, элементы которой обратны соответствующим диагональным элементам исходной матрицы **A**. Тогда матрица **M** будет состоять только из диагональных элементов, которые равны $1/a_{ii}$, $i = 1, 2, \ldots, N$. На практике предобусловливатель Якоби применятся к матрицам, диагональные элементы которых сильно различаются. В ряде случаев это позволяет уменьшить число обусловленности матрицы, а соответственно и число итераций метода [52].

1.3.3.2.2 Неявное предобусловливание

Неявное предобусловливание не требует явного вычисления $\mathbf{M}^{-1}$, но требует решения СЛАУ в каждой итерации [54], при этом система $\mathbf{Ax} = \mathbf{b}$ преобразуется к виду

$$\mathbf{MAx} = \mathbf{Mb}, \qquad (1.22)$$

Большая часть неявных методов основана на представлении $\mathbf{M} = \mathbf{LU}$, где **L** и **U** получены из матрицы $\mathbf{A}_s$ путем LU-разложения (см. разд. 1.3.2.2). Матрицу $\mathbf{A}_s$ обычно получают из исходной **A** путем уменьшения количества ненулевых элементов (предфильтрации). Способы предфильтрации доста-

точно разнообразны, они отличаются подходами к уменьшению количества ненулевых элементов (подробно рассмотрены в разд. 1.3.3.2.3).

При LU-разложении матрицы **L** и **U** получаются менее разреженными, поэтому часто используют неполное LU-разложение (или ILU-разложение) [52]. Введем S – множество позиций, которые требуется заполнить в произведении **LU**. В общем виде ILU-разложение выглядит следующем образом:

$$a_{ij} \leftarrow \begin{cases} a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}, a_{ij} \in S \\ a_{ij}, \text{в противном случае} \end{cases}, \quad (1.23)$$

для любого k и для $i, j > k$.

Самый простой способ ILU-разложения – это ILU(0), в этом случае множество S совпадает со множеством ненулевых элементов матрицы $\mathbf{A}_s$ [52]. Более сложные способы, такие как ILU(p) [62], ILUT [46] и др., используют анализ значений элементов матрицы $\mathbf{A}_s$ для определения необходимости обнуления [54].

1.3.3.2.3 Способы предфильтрации

Исторически предобусловливание применялось к разреженным матрицам, однако Л.Ю. Колотилиной предложен способ, позволивший использовать предобусловливание и для плотных матриц, используя предфильтрацию [63]:

$$a_{ij}^s = 0, \text{ если } \left| a_{ij} / a^{\max} \right| < \varepsilon \qquad i, j = 1, \ldots, N, \quad (1.24)$$

где $a^{\max}$ – максимальный элемент матрицы, ε – порог обнуления.

В данном способе используется нормировка элементов по максимальному элементу в матрице. Таким образом предфильтрация заключается в получении матрицы $\mathbf{A}_s$ из **A** путем обнуления незначительных (малых по значению) элементов. Часто используют сравнение значений элементов матрицы с некоторым порогом (алгебраическая предфильтрация) [54].

Рассмотрим далее другие способы. В [54] приведен способ, основанный на максимальном элементе в строке:

$$a_{ij}^{s} = 0, \text{ если } |a_{ij} / a_i^{\max}| < \varepsilon, \qquad i, j = 1, \ldots, N,$$

и описаны способы, использующие нормы матриц, например, бесконечную:

$$a_{ij}^{s} = 0, \text{ если } |a_{ij}| < \varepsilon, \text{ где } \varepsilon = \|\mathbf{A}\|_{\infty} \tau, \qquad i, j = 1, \ldots, N,$$

где $\|\mathbf{A}\|_{\infty} = \max_j \sum_i a_{ij}$.

Еще один способ предфильтрации реализуется с помощью порога обнуления, получаемого из произведения евклидовой нормы матрицы **A** и задаваемого допуска обнуления τ [64]:

$$a_{ij}^{s} = 0, \text{ если } |a_{ij}| < \varepsilon, \text{ где } \varepsilon = \|\mathbf{A}\|_{F} \tau, \qquad i, j = 1, \ldots, N,$$

где $\|\mathbf{A}\|_{F} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} |a_{ij}|^2}$.

1.3.3.3 Методы подпространства Крылова

На основе подпространств Крылова разработано много итерационных методов, которые отличаются размерностью *m*, способом выбора подпространства *L* и способом построения базисов [20]. Итерационные методы подпространств Крылова получили широкое распространение в конце XX века и активно развиваются в настоящее время [52]. Такие методы получили название вариационных. В таблице 1.1 приведены наиболее популярные из них [65].

Ниже приведены алгоритмы наиболее распространенных итерационных методов.

1.3.3.3.1 Метод бисопряженных градиентов

Впервые метод BiCG (метод бисопряженных градиентов) был предложен Ланцошом [69]. В качестве подпространства *K* выбрано подпространство Крылова $K_m(\mathbf{r}_0, \mathbf{A})$, а в качестве подпространства $L - L_m(\tilde{\mathbf{r}}_0, \mathbf{A}^T)$, где $\tilde{\mathbf{r}}_0$, такой вектор, при котором $(\mathbf{r}_0, \tilde{\mathbf{r}}_0) \neq 0$. Далее строятся два базиса, ортогональных друг другу. Для построения биортогонального базиса используется процедура Ланцоша [52]. Приближенное решение уточняется по формуле

$$\mathbf{x}_m = \mathbf{x}_0 + \beta \mathbf{V}_m \mathbf{T}_m^{-1} \mathbf{e}_1, \qquad (1.25)$$

где $\beta = (\mathbf{r}_0, \tilde{\mathbf{r}}_0)$. Метод требует хранить одновременно все базисные вектора или перевычислять их после нахождения коэффициентов, что неэффективно. Поэтому на практике используют следующий подход:

Таблица 1.1 – Наиболее популярные итерационные методы подпространств Крылова

Год	**Автор(ы)**	**Метод**	**Источник**
1950	Lanczos	Lanczos	[66]
1951	Arnoldi	Arnoldi	[67]
1952	Hestenes, Stiefe	CG	[68]
1952	Lanczos	Lanczos (CG)	[69]
1975	Paige, Saunders	MINRES	[70]
1975	Paige, Saunders	SYMMLQ	[70]
1975	Fletcher	BiCG	[71]
1976	Concus, Golub	CGW	[72]
1977	Vinsome	ORTHOMIN	[73]
1977	Meijerink, van der Vorst	ICCG	[74]
1978	Widlund	CGW	[75]
1980	Jea, Young	ORTHODIR	[76]
1981	Saad	FOM	[77]
1982	Paige, Saunders	LSQR	[78]
1983	Eisenstat et al.	GCR	[79]
1986	Saad, Schultz	GMRES	[80]
1989	Sonneveld	CGS	[81]
1991	Freund and Nachtigal	QMR	[82]
1992	van der Vorst	BiCGStab	[83]
1993	Gutknecht	BiCGStab2	[84]
1993	Sleijpen, Fokkema	BiCGStab(l)	[85]
1994	Freund	TFQMR	[86]
1994	Weiss	GMERR	[87]
1994	Chan et al.	QMR-BiCGStab	[88]
1995	Kasenally, Ebrahim	GMBACK	[89]
1996	Fokkema et al	CGS2	[90]
1999	de Sturler	GCROT	[91]

Запишем LU-разложение матрицы $\mathbf{T}_m$:

$$\mathbf{T}_m = \mathbf{L}_m \mathbf{U}_m$$

и определим матрицу $\mathbf{P}_m$ следующим образом:

$$\mathbf{P}_m = \mathbf{V}_m \mathbf{U}_m^{-1}. \qquad (1.26)$$

Подставим полученное выражение в (1.25):

$$\mathbf{x}_m = \mathbf{x}_0 + \beta \mathbf{V}_m \mathbf{U}_m^{-1} \mathbf{L}_m^{-1} \mathbf{e}_1 = \mathbf{x}_0 + \beta \mathbf{P}_m \mathbf{L}_m^{-1} \mathbf{e}_1$$

и по аналогии с (1.26) запишем:

$$\tilde{\mathbf{P}}_m = \mathbf{W}_m(\mathbf{L}_m^{\mathrm{T}})^{-1}. \qquad (1.27)$$

Тогда можно записать:

$$\tilde{\mathbf{P}}_m^{\mathrm{T}}\mathbf{A}\mathbf{P}_m = \mathbf{L}_m^{-1}\mathbf{W}_m^{\mathrm{T}}\mathbf{A}\mathbf{V}_m\mathbf{U}_m^{-1} = \mathbf{L}_m^{-1}\mathbf{T}_m\mathbf{U}_m^{-1} = \mathbf{D}_m,$$

где $\mathbf{D}_m$ – диагональная матрица с элементами $d_{ii} = (v_i, w_i)$.

Если представить столбцы матрицы $\mathbf{P}_m$ векторами $\mathbf{p}_k$, а столбцы матрицы $\tilde{\mathbf{P}}_m$ – векторами $\tilde{\mathbf{p}}_k$, тогда, согласно (1.27), можно записать:

$$i \neq j \rightarrow \tilde{\mathbf{p}}_i^{\mathrm{T}}\mathbf{A}\mathbf{p}_j = (\mathbf{A}\mathbf{p}_i, \tilde{\mathbf{p}}_i) = 0. \qquad (1.28)$$

Векторы $\{\tilde{\mathbf{p}}_i\}_{i=1}^{m}$ и $\{\mathbf{p}_i\}_{i=1}^{m}$, удовлетворяющие условию (1.28), называются бисопряженными [52]. Бисопряженность эквивалентна биортогональности скалярного A-произведения, а потому для нахождения векторов $\tilde{\mathbf{p}}_i$ и $\mathbf{p}$ вместо (1.26) и (1.27) можно использовать биортогонализацию Ланцоша [54].

Используя эту информацию, можно получить алгоритм метода BiCG:

Алгоритм 1.1 – BiCG

1 Выбрать начальное приближение $\mathbf{x}_0$, вычислить $\mathbf{r}_0 = \mathbf{b}–\mathbf{A}\mathbf{x}_0$ и выбрать $\tilde{\mathbf{r}}_0$ так, что $(\mathbf{r}_0, \tilde{\mathbf{r}}_0) \neq 0$ (например, $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$)

2 $\mathbf{p}_0 = \mathbf{r}_0$, $\tilde{\mathbf{p}}_0 = \tilde{\mathbf{r}}_0$

3 Для $j = 1, 2, \ldots$ до сходимости или до N_{it}^{max}

4 Найти $\mathbf{z}_i$ из системы $\mathbf{M}\,\mathbf{z}_i = \mathbf{r}_i$

5 Найти $\tilde{\mathbf{z}}_i$ из системы $\mathbf{M}^{\mathrm{T}}\,\tilde{\mathbf{z}}_i = \tilde{\mathbf{r}}_j$

6 $\alpha_j = (\mathbf{z}_j, \tilde{\mathbf{r}}_j)/(\mathbf{p}_j, \mathbf{A}\tilde{\mathbf{p}}_j)$

7 $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$

8 $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{A}\,\mathbf{p}_j$

9 $\tilde{\mathbf{r}}_{j+1} = \tilde{\mathbf{r}}_j + \alpha_j\,\mathbf{A}\tilde{\mathbf{p}}_j$

10 $\beta_j = (\mathbf{z}_{j+1}, \tilde{\mathbf{r}}_{j+1})/(\mathbf{z}_j, \tilde{\mathbf{r}}_j)$

11 $\mathbf{p}_{j+1} = \mathbf{r}_j + \beta_j\mathbf{p}_j$

12 $\tilde{\mathbf{p}}_{j+1} = \tilde{\mathbf{r}}_j + \beta_j\tilde{\mathbf{p}}_j$

13 Увеличить j

1.3.3.3.2 Стабилизированный метод бисопряженных градиентов

Описанный выше алгоритм BiCG обладает рядом недостатков он обладает неустойчивостью; может возникнуть осциллирующее поведение нормы

невязки; итерационный процесс может полностью оборваться, без возможности его дальнейшего продления [54]. Это происходит если коэффициент $\beta_j = 0$. Поэтому разработан алгоритм BiCGStab [83]. Далее приведен алгоритм BiCGStab с неявным предобусловливанием.

Алгоритм 1.2 – BiCGStab

1 Выбрать начальное приближение $\mathbf{x}_0$, вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
2 Выбрать вектор $\tilde{\mathbf{r}}$, удовлетворяющий условию $(\mathbf{r}_0, \tilde{\mathbf{r}}) \neq 0$ (например, $\tilde{\mathbf{r}} = \mathbf{r}_0$)
3 Для $i = 1, 2, \ldots$ до сходимости или до N_{it}^{max}
4 $\rho_{i-1} = (\tilde{\mathbf{r}}, \mathbf{r}_{i-1})$
5 Если $\rho_{i-1} = 0$
6 то метод не может решить данную систему
7 Если $i = 1$
8 $\mathbf{p}_i = \mathbf{r}_{i-1}$
9 Иначе
10 $\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})\,(\alpha_{i-1} / \omega_{i-1})$
11 $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_{i-1}(\mathbf{p}_{i-1} - \omega_{i-1}\,\mathbf{v}_{i-1})$
12 Найти $\tilde{\mathbf{p}}$ из системы $\mathbf{M}\,\tilde{\mathbf{p}} = \mathbf{p}_i$
13 $\mathbf{v}_i = \mathbf{A}\,\tilde{\mathbf{p}}$
14 $\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}}, \mathbf{v}_i)$
15 $\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\,\mathbf{v}_i$
16 Если $\|\mathbf{s}\|_2 / \|\mathbf{r}_0\|_2 \leq Tol$
17 то КОНЕЦ ($\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\,\tilde{\mathbf{p}}$ – полученное решение)
18 Найти $\tilde{\mathbf{s}}$ из системы $\mathbf{M}\,\tilde{\mathbf{s}} = \mathbf{s}$
19 $\mathbf{t} = \mathbf{A}\,\tilde{\mathbf{s}}$
20 $\omega_i = (\mathbf{t}, \mathbf{s}) / (\mathbf{t}, \mathbf{t})$
21 $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\,\tilde{\mathbf{p}} + \omega_i\,\tilde{\mathbf{s}}$
22 $\mathbf{r}_i = \mathbf{s} - \omega_i\,\mathbf{t}$
23 Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \leq Tol$
24 то КОНЕЦ ($\mathbf{x}_i$ – полученное решение)
25 Увеличить i

1.3.3.3.3 Квадратичный метод сопряженных градиентов

Основная идея алгоритма CGS заключается в следующем: невязку $\mathbf{r}_j$ можно выразить как полином от начальной невязки $\mathbf{r}_j = \mathbf{p}_j(\mathbf{A})\mathbf{r}_0$. Аналогично полином сопряженного пространства: $\tilde{\mathbf{r}}_j = \mathbf{p}_j(\mathbf{A}^T)\,\tilde{\mathbf{r}}_0$. Для BiCG скалярное

произведение $(\mathbf{r}_j,\ \tilde{\mathbf{r}}_j)$ примет вид $(\mathbf{p}_j(\mathbf{A})\mathbf{r}_0,\ \mathbf{p}_j(\mathbf{A}^{\mathrm{T}})\ \tilde{\mathbf{r}}_0) \equiv (\mathbf{p}^2{}_j(\mathbf{A})\mathbf{r}_0,\ \tilde{\mathbf{r}}_0)$. Это наблюдение привело к соотношению $\tilde{\mathbf{x}}_j \in K^{2j}(\mathbf{A},\ \mathbf{r}_0)$, для которого можно записать невязку $\tilde{\mathbf{r}}_j = \mathbf{p}^2{}_j(\mathbf{A})\mathbf{r}_0$. Следовательно, невязка уменьшается в два раза быстрее [52]. Далее приведен алгоритм метода CGS.

Алгоритм 1.3 – CGS

1 Выбрать начальное приближение $\mathbf{x}_0$, вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
2 Выбрать вектор $\tilde{\mathbf{r}}$, удовлетворяющий условию $(\mathbf{r}_0,\ \tilde{\mathbf{r}}) \neq 0$ (например, $\tilde{\mathbf{r}} = \mathbf{r}_0$)
3 Для $i = 1, 2, \ldots$ до сходимости или до N_{it}^{max}
4 $\rho_{i-1} = (\tilde{\mathbf{r}},\ \mathbf{r}_{i-1})$
5 Если $\rho_{i-1} = 0$
6 то метод не может решить данную систему
7 Если $i = 1$
8 $\mathbf{u}_1 = \mathbf{r}_0$
9 $\mathbf{p}_1 = \mathbf{u}_1$
10 Иначе
11 $\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})$
12 $\mathbf{u}_i = \mathbf{r}_i + \beta_{i-1}\mathbf{q}_{i-1}$
13 $\mathbf{p}_i = \mathbf{u}_i + \beta_{i-1}(\mathbf{q}_{i-1} + \beta_{i-1}\mathbf{p}_{i-1})$
14 Найти $\tilde{\mathbf{p}}$ из системы $\mathbf{M}\tilde{\mathbf{p}} = \mathbf{p}_i$
15 $\tilde{\mathrm{v}} = \mathbf{A}\tilde{\mathbf{p}}$
16 $\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}}, \tilde{\mathrm{v}})$
17 $\mathbf{q}_i = \mathbf{u}_i - \alpha_i\,\tilde{\mathrm{v}}$
18 Найти $\tilde{\mathbf{u}}$ из системы $\mathbf{M}\tilde{\mathbf{u}} = \mathbf{u}_i + \mathbf{q}_i$
19 $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\tilde{\mathbf{u}}$
20 $\tilde{\mathbf{q}} = \mathbf{A}\tilde{\mathbf{u}}$
21 $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i\,\tilde{\mathbf{q}}$
22 Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \leq Tol$
23 то КОНЕЦ ($\mathbf{x}_i$ – полученное решение)
24 Увеличить i

1.4 Хранение разреженных матриц

Для уменьшения вычислительных затрат в итерационных методах используют предобусловливание (см. разд. 1.3.3.2). Однако это требует хранения дополнительной матрицы, что увеличивает требуемую память компьютера. Особенностью дополнительной матрицы является то, что она,

как правило, является разреженной. Хранить её можно с помощью любого из форматов хранения разреженных матриц для экономии машинной памяти. Далее выполнен обзор этих форматов.

Определение разреженной матрицы

Разреженная матрица – матрица, имеющая преимущественно нулевые элементы. Объем ненулевых элементов, в общем случае, определяется для каждой отдельной задачи [92]. Одно из определений принадлежит Альварадо (1979 г.): матрица порядка N, число ее ненулевых элементов должно выражаться как $N^{1+\gamma}$, где $\gamma < 1$. Матрица разрежена, если $\gamma \leq 0{,}2$ и для ленточных матриц, если $\gamma \leq 0{,}5$ [93].

Другое определение заключается в том, что матрицу следует считать разреженной, если удалось извлечь выгоду из наличия в ней нулей [92]. Такой выгодой может стать, например, уменьшение памяти компьютера для хранения матрицы в сжатом формате, в отличие от формата, не учитывающего наличие нулевых элементов.

Форматы хранения разреженных матриц

Существует много различных форматов хранения разреженных матриц [92, 94]. Они отличаются по степени сжатия матрицы и по затратам машинного времени, необходимого для элементарных операций с матрицей (например, поиск элемента, добавление нового элемента и др.) [94]. Как правило, форматы, обеспечивающие хорошее сжатие, требуют для операций больше машинного времени и наоборот [92, 94]. Поэтому требуется выбирать формат для каждой конкретной задачи.

Из наиболее известных форматов следует выделить следующие: из двух векторов [94], Кнута [95], Рейнболдта и Местеньи [96], Ларкума [97], Шермана [98], разреженный строчный (Compressed Storage Row, CSR), разреженный столбцовый (Compressed Storage Column, CSC) [99, 100], разреженный неравномерный диагональный (Jagged Diagonal Storage, JDS) [101], сжатый диагональный (Compressed Diagonal Storage, CDS) [102], разрежен-

ный блочный строчный (Sparse Block Compressed Row Storage, SBCRS) [103] и др. [104–107].

Существуют форматы хранения разреженных матриц, предназначенные только для хранения определенного вида матриц (симметричных, ленточных и др.). Так, формат CDS предназначен для хранения только ленточных матриц. Далее рассмотрим самые распространенные форматы хранения разреженных матриц.

1.4.1.1 Формат из двух векторов

При использовании формата из двух векторов матрица хранится в виде двух векторов (**I** и **A**) [94]. Вектор **I** содержит номер столбца, **A** – значения ненулевых элементов. Если элемент в векторе **I** равен нулю, то это означает начало строки, при этом элемент вектора **A** содержит номер следующей строки. Если нули в двух векторах, то это означает конец матрицы. На рисунке 1.3 приведен пример формата.

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Вектор	Индекс элемента													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I	0	1	4	0	2	5	0	1	3	5	0	1	5	0
A	1	a_{11}	a_{14}	2	a_{22}	a_{25}	3	a_{31}	a_{33}	a_{35}	5	a_{51}	a_{55}	0

Рисунок 1.3 – Представление матрицы с помощью формата из двух векторов

Основным преимуществом формата является то, что он обеспечивает хороший коэффициент сжатия [94]. Недостатком является сложность выполнения элементарных операций, таких как поиск и добавление элемента матрицы.

1.4.1.2 Формат Кнута

Формат предложен Кнутом в 1968 г. [95]. Матрица хранится в виде семи векторов, трех основных (**AN**, **I**, **J**) и четырех дополнительных (**NR**, **NC**, **JR**, **JC**). Первый вектор (**AN**) содержит ненулевые элементы матрицы в произвольном порядке. Второй (**I**) и третий (**J**) векторы содержат индексы

столбцов и строк элемента, записанного в первом векторе. Для ускорения поиска ненулевого элемента использованы дополнительные четыре вектора, в которых хранятся указатели на позиции ненулевого элемента в векторе **AN**. **NR** – указатель на следующий элемент в строке (**NC** – в столбце) исходной матрицы, а **JR** – указатель на первый элемент в строке (**JC** – в столбце). На рисунке 1.4 приведен пример представления матрицы с помощью формата Кнута.

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Вектор	Индекс элемента								
	1	2	3	4	5	6	7	8	9
AN	a_{11}	a_{14}	a_{22}	a_{25}	a_{31}	a_{33}	a_{35}	a_{51}	a_{55}
I	1	4	2	5	1	3	5	1	5
J	1	1	2	2	3	3	3	5	5
NR	2	0*	4	0	6	7	0	9	0
NC	5	0	0	7	8	0	9	0	0
JR	1	3	5	0**	8				
JC	1	3	6	2	4				

Рисунок 1.4 – Представление матрицы с помощью формата Кнута
(* – текущий элемент последний в строке/столбце,
** – в данной строке нет ненулевых элементов)

Недостаток формата заключается в том, что для хранения каждого элемента матрицы требуется 5 значений (значения векторов **AN**, **I**, **J**, **NR**, **NC**), вследствие чего мал коэффициент сжатия [92]. Достоинством формата является легкость добавления (или удаления) элементов матрицы, поэтому он хорошо подходит в алгоритмах, где нельзя предсказать конечное число ненулевых элементов [92] (например, LU-разложение).

1.4.1.3 Формат Рейнболдта и Местеньи

Формат Рейнболдта и Местеньи (1973 г.) является модификаций формата Кнута [96]. В отличие от формата Кнута векторы **NR** и **NC** закольцовываются, т.е. последний элемент в строке (в столбце) содержит указатель на первый элемент строки (столбца). Векторы **I**, **J** не хранятся. Формат требует значительно меньше памяти, однако при поиске элемента необходимо просматривать все элементы текущей строки (столбца), т.к. заранее неизвестны индексы её ненулевых элементов. Пример приведен на рисунке 1.5.

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Вектор	Индекс элемента								
	1	2	3	4	5	6	7	8	9
AN	a_{11}	a_{14}	a_{22}	a_{25}	a_{31}	a_{33}	a_{35}	a_{51}	a_{55}
NR	2	1	4	3	6	7	5	9	8
NC	5	2	3	7	8	9	8	1	4
JR	1	3	5	0**	8				
JC	1	3	6	2	4				

Рисунок 1.5 – Представление матрицы с помощью формата Рейнболдта и Местеньи
(* – в данной строке нет ненулевых элементов)

1.4.1.4 Формат Ларкума

Усовершенствованный вариант формата Кнута использовался Ларкумом (1971 г.) для хранения симметричных (или треугольных) матриц (рисунок 1.6) [97]. Все элементы главной диагонали (даже если элемент нулевой) и ненулевые элементы матрицы хранятся в векторе **AN**. Еще в двух векторах хранятся указатели на ненулевые элементы: **NR** – описывает строку слева направо; **NC** – описывает столбец над диагональю снизу вверх. Если добавить еще два аналогичных вектора, то возможно хранить несимметричные матрицы.

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3			a_{33}		a_{35}
4					
5					a_{55}

Вектор	Индекс элемента							
	1	2	3	4	5	6	7	8
AN	a_{11}	a_{22}	a_{33}	0*	a_{55}	a_{14}	a_{25}	a_{35}
NR	6	7	8	0**	0	0	0	0
NC	0**	0	0	6	8	0	0	7

Рисунок 1.6 – Представление матрицы с помощью формата Ларкума
(* – диагональный элемент нулевой; ** – ноль в данном векторе означает,
что текущий элемент последний в строке/столбце)

1.4.1.5 Формат Шермана

Еще один из форматов (рисунок 1.7) для треугольных матриц был предложен Шерманом (1975 г.) [98]. Формат состоит из пяти векторов. В **UD** – хранятся диагональные элементы, **UN** – недиагональные ненулевые, **IU** – указатели на первые элементы строк недиагональных элементов, **JU** – индексы столбцов, которые сжаты специальным образом (повторяющиеся

индексы столбцов, для разных строк, записываются один раз, рисунок 1.8), **U** – указатели на первые элементы столбца вектора **JU**.

	1	2	3	4	5
1	a_{11}		a_{13}	a_{14}	a_{15}
2		a_{22}			a_{25}
3			a_{33}	a_{34}	a_{35}
4					
5					a_{55}

Вектор	Индекс элемента					
	1	2	3	4	5	6
UD	a_{11}	a_{22}	a_{33}	0	a_{55}	
UN	a_{13}	a_{14}	a_{15}	a_{25}	a_{34}	a_{35}
IU	1	4	5	0*	0*	
U	1	3	1	0	0	
JU**	3	4	5			

Рисунок 1.7 – Представление матрицы с помощью формата Шермана
(* – в данной строке нет элементов; ** – правило сжатия представлено на рисунке 1.8)

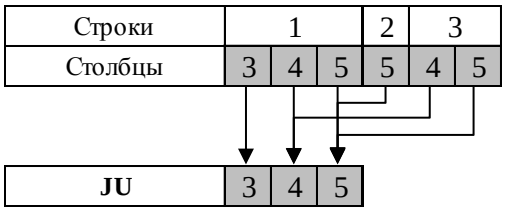


Рисунок 1.8 – Сжатие вектора JU

1.4.1.6 Формат CSR (CSC)

Форматы CSR и CSC, предложенные Чангом (1969 г.) и Густавсоном (1972 г.), широко используются для хранения разреженных матриц [99, 100]. Эти форматы предъявляют минимальные требования к памяти и эффективны для реализации нескольких важных операций над разреженными матрицами: сложения, умножения, перестановок строк и столбцов, транспонирования, решения СЛАУ с разреженными матрицами как прямыми, так и итерационными методами и т. д. [46]. В формате CSR значения ненулевых элементов матрицы и соответствующие им индексы столбцов хранятся по строкам в двух векторах (**Values** – ненулевые элементы и **Columns** – индексы столбцов). Используется также вектор указателей на ненулевые элементы (**RowIndex**), с которых начинается очередная строка (рисунок 1.9). Формат CSC отличается только порядком хранения ненулевых элементов матрицы: в CSR элементы хранятся по строкам, в CSC – по столбцам. При этом используются векторы **Values**, **Rows**, **ColumnsIndex**, – соответственно (рисунок 1.10).

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Вектор	Индекс элемента								
	1	2	3	4	5	6	7	8	9
Values	a_{11}	a_{14}	a_{22}	a_{25}	a_{31}	a_{33}	a_{35}	a_{51}	a_{55}
Columns	1	4	2	5	1	3	5	1	5
RowIndex	1	3	5	0*	8				

Рисунок 1.9 – Представление матрицы с помощью формата CSR
(* – в данной строке нет элементов)

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Вектор	Индекс элемента								
	1	2	3	4	5	6	7	8	9
Values	a_{11}	a_{31}	a_{51}	a_{22}	a_{33}	a_{14}	a_{25}	a_{35}	a_{55}
Rows	1	3	5	2	3	1	2	3	5
ColumnsIndex	1	4	5	6	7				

Рисунок 1.10 – Представление матрицы с помощью формата CSC

1.4.1.7 Формат хранения JDS

На первом шаге сжатия все ненулевые элементы матрицы сдвигаются влево, игнорируя нулевые, и хранятся в матрице **Val**. Номера столбцов этих элементов хранятся в отдельной матрице **Col**. Если в строке содержатся только нулевые элементы, то в соответствующих строках матрицы **Val** и **Col** хранятся нули (рисунок 1.11).

	1	2	3	4	5
1	a_{11}			a_{14}	
2		a_{22}			a_{25}
3	a_{31}		a_{33}		a_{35}
4					
5	a_{51}				a_{55}

Val

a_{11}	a_{14}	
a_{22}	a_{25}	
a_{31}	a_{33}	a_{35}
a_{51}	a_{55}	

Col

1	4	
2	5	
1	3	5
1	5	

Рисунок 1.11 – Первый шаг сжатия матрицы с помощью формата JDS

Однако при таком способе хранения возникает избыточность, которая устраняется дополнительным сжатием матрицы **Val** по столбцам. В результате получаются три вектора (см. рисунок 1.12): **Values** (значения ненулевых элементов), **Columns** (индекс столбца ненулевого элемента) **StartPosition** (указатель на начальный элемент в столбце матрицы **Val**).

Val

a_{11}	a_{14}	
a_{22}	a_{25}	
a_{31}	a_{33}	a_{35}
a_{51}	a_{55}	

Col

1	4	
2	5	
1	3	5
1	5	

Вектор	Индекс элемента								
	1	2	3	4	5	6	7	8	9
Values	a_{11}	a_{22}	a_{31}	a_{51}	a_{14}	a_{25}	a_{33}	a_{55}	a_{35}
Columns	1	3	5	2	3	1	2	3	5
StartPosition	1	5	9						

Рисунок 1.12 – Представление матрицы с помощью формата JDS

1.4.1.8 Сжатый диагональный формат хранения (CDS)

Сжатый диагональный формат (CDS) описан в [92, 94]. Этот формат предназначен для хранения ленточных матриц (матрицы, у которых ненулевые элементы расположены только на главной диагонали и примыкающих к ней [46]). Исходная матрица представляется матрицей **Val**, где диагонали исходной матрицы хранятся по строкам, начиная слева направо (см. рисунок 1.13). При этом нет необходимости хранить дополнительную информацию [92].

	1	2	3	4	5
1	a_{11}	a_{12}			
2	a_{21}	a_{22}			
3		a_{32}	a_{33}	a_{34}	
4			a_{43}	a_{44}	a_{45}
5					a_{55}

Val

	a_{21}	a_{32}	a_{43}	
a_{11}	a_{22}	a_{33}	a_{44}	a_{55}
	a_{12}		a_{34}	a_{45}

Рисунок 1.13 – Представление матрицы с помощью формата CDS

1.5 Многократное решение СЛАУ

Существуют затратные по времени задачи, в которых необходимо многократное решение СЛАУ. В случае, когда изменяются все составляющие СЛАУ, задача многократного решения сводится к уравнению:

$$\mathbf{A}_k\mathbf{x}_k = \mathbf{b}_k,\ k = 1, 2, \ldots m, \qquad (1.29)$$

где k – порядковый номер СЛАУ, m – количество СЛАУ.

Необходимость решения уравнения вида (1.29) возникает во многих задачах научных и инженерных вычислений [108], например, в методах, использующих рекурсивные вычисления наименьших квадратов [78, 109], задачах рассеяния волн [110], реставрации изображений [111] и др. Далее

рассмотрены несколько подходов для решения уравнения вида (1.29). Подробно подходы рассмотрены в работе [112].

Частным случаем многократного решения СЛАУ являются задачи, когда неизменной остается одна из составляющих СЛАУ, например, при неизменной матрице **A**, задача может быть записана в виде [113]:

$$\mathbf{AX} = \mathbf{B}, \tag{1.30}$$

где $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ и $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$. В этом случае задача упрощается и для её решения могут быть использованы итерационные методы. Например, описана методика использования подпространств Крылова для такой задачи. При этом подпространство будет выглядеть следующим образом: $K_m(\mathbf{A}, \mathbf{R}) = span\{\mathbf{R}, \mathbf{AR}, \mathbf{A}^2\mathbf{R}, \ldots, \mathbf{A}^{m-1}\mathbf{R}\}$, где **R** – блок векторов $N \times m$, а матричные базисы подпространств K и L представляются в виде: $\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_m]$, где **V** – блок векторов $N \times m$; $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_m]$, где **W** – блок векторов $N \times m$ [114, 108]. Таким образом, практически любой проекционный итерационный метод может быть адаптирован для решения таких задач. Например, разработаны новые итерационные методы, такие как GL-LSQR [114], MHGMRES(m) [115], MEGCR [116] и др.

Еще одним частным случаем являются задачи, в которых на каждом шаге меняется матрица **A**, а вектор **b** – неизменен. Пример такой задачи – вычисление емкостных матриц полосковых структур в диапазоне изменения их параметров. Так как **b** неизменен, то

$$\mathbf{A}_k\mathbf{x}_k = \mathbf{b}, \qquad k = 1, 2, \ldots m. \tag{1.31}$$

Примером такого случая может служить задача вычисления емкостных матриц, полученных из одной и той же структуры при изменении значения диэлектрической проницаемости диэлектриков [13, 117]. В этом случае изменяются элементы, находящихся только на главной диагонали нижней правой подматрицы (с индексами элементов матрицы СЛАУ больше N_C), соответствующих подынтервалам диэлектрик–диэлектрик (рисунок 1.2), и поэтому можно использовать алгоритмы, основанные на блочном LU–разложении [13,

14, 15, 117]. В упомянутых работах показана возможность существенного снижения временных затрат.

Однако при изменении размеров структуры, что приводит к изменению элементов матрицы, расположенных в произвольных местах [18], использование блочного LU–разложения невозможно. Такие задачи могут решаться итерационными методами с предобусловливанием, причем предобусловливатель, полученный для одной матрицы, можно использовать для других. В качестве первого шага в этом направлении было исследовано уменьшение нормы невязки 10-кратного решения СЛАУ, полученных при малом изменении нескольких параметров структуры [19]. Недостатком данного подхода является то, что переформирование предобусловливателя – это трудоёмкий процесс, и при сильных изменениях в матрице будет требоваться постоянное переформирование, что неэффективно. Однако существуют методы, использующие корректировку предобусловливателя [112]. В случае неявного предобусловливания, как сказано в той же работе, корректировка **L** и **U** матриц достаточно сложна и почти всегда неэффективна. В случае явного предобусловливания существуют методы пошагового приближения к обратной матрице, причем эту процедуру можно повторять на каждом шаге. Однако эти методы отличаются большой трудоёмкостью, особенно для плотных матриц.

1.6 Основные результаты главы

Обзор задач математического моделирования методом моментов показал актуальность уменьшения вычислительных затрат. При этом большая их часть приходится на решение СЛАУ, поэтому, уменьшая, например, время решения СЛАУ, можно сократить время моделирования в целом. Обзор методов решения СЛАУ (см. разд. 1.3) показал перспективность использования итерационных методов подпространств Крылова с применением предобусловливателя. При предобусловливании используют разреженную матрицу, полученную из исходной путем предфильтрации. Для хранения разреженной

матрицы можно использовать специальные форматы (см. разд. 1.4), тем самым, можно добиться экономии машинной памяти и, в перспективе, времени решения СЛАУ.

На практике часто требуется математическое моделирование, например вычисление ёмкостной матрицы, при изменении параметров структуры (см. разд. 1.2), очень затратное по времени, т.к. приходится многократно решать СЛАУ. Из обзора (см. разд. 1.5) следует, что для решения таких задач можно использовать итерационные методы с предобусловливанием, что в перспективе может обеспечить снижение временных затрат.

2 ПРИМЕНЕНИЕ РАЗРЕЖЕННОГО СТРОЧНОГО ФОРМАТА ХРАНЕНИЯ МАТРИЦЫ ДЛЯ УСКОРЕНИЯ РЕШЕНИЯ СЛАУ

2.1 Аналитические оценки коэффициента сжатия данных

В данном разделе представлены результаты анализа сжатия для наиболее часто используемых форматов хранения разреженных матриц. Выполнено сравнение форматов и выбран оптимальный (по критерию минимизации машинной памяти и числа операций, требуемых при использовании формата в алгоритмах). Раздел основан на работах [118–125].

Для формата хранения разреженных матриц важным показателем является коэффициент сжатия (k), который он может обеспечить [118]. Коэффициент сжатия определяется как отношение N^2 (N – порядок матрицы) к числу хранимых данных в разреженном формате. Для определения коэффициента сжатия можно использовать подсчет объема данных конкретной матрицы, хранящихся в сжатом формате, но удобнее использовать аналитические выражения, позволяющие получить простую оценку числа данных. Аналитические выражения получены исходя из структуры конкретного формата хранения. При этом тип данных не учитывается (хранятся как сами ненулевые элементы, так и указатели, для хранения которых требуется меньше машинной памяти). В результате, получена нижняя граница коэффициента сжатия, который зависит от порядка матрицы (N) и количества ненулевых элементов ($N_н$) в ней. Исключением, в рассматриваемых форматах, являются формат Шермана, который зависит от портрета ненулевых элементов (поэтому здесь приведены значения для оптимального, обеспечивающего максимальное сжатие, и неоптимального портретов матрицы), а также формат CDS, который зависит от числа диагоналей, содержащих ненулевые элементы (N_c). В таблице 2.1 представлены полученные выражения для рассматриваемых форматов хранения разреженных матриц. Видно, что некоторые форматы, при одинаковом числе ненулевых элементов $N_н$, имеют одинаковый коэффициент сжатия, например, формат двух векторов, CSR и JDS.

Таблица 2.1 – Коэффициент сжатия для разных форматов

Формат	**Коэффициент сжатия**
Два вектора	$k = N^2/(2N_н+N)$
Кнута	$k = N^2/(5N_н+2N)$
Рейнболдта и Местеньи	$k = N^2/(3N_н+2N)$
Ларкума*	$k = N^2/(3N_н)$
Шермана*	$k_{\min}^{***} = N^2/(3N_н)$ $k_{\max}^{***} = N^2/(2N_н)$
CSR	$k = N^2/(2N_н+N)$
JDS	$k = N^2/(2N_н+N)$
CDS**	$k = N^2/(N_cN)$

* – только для треугольных матриц; ** – только для ленточных матриц; *** – для оптимального ($k_{\max}$) и неоптимального ($k_{\min}$) портретов матрицы; $N_н$ – число ненулевых элементов матрицы; N_c – число диагоналей, содержащих ненулевые элементы.

Оценка коэффициента сжатия с помощью числа ненулевых элементов не всегда удобна. Это связано с тем, что число ненулевых элементов не может однозначно определить коэффициент сжатия формата, т.к. при одинаковом числе ненулевых элементов, но при разном порядке, коэффициент сжатия будет разный. Чтобы избежать неоднозначности, предложено использовать оценку числа ненулевых элементов в матрице с помощью показателя плотности матрицы (q), представляющего собой отношение числа её ненулевых элементов к N^2. Для примера, на рисунке 2.1 представлены рассчитанные по полученным формулам (см. таблицу 2.1) зависимости коэффициента сжатия форматов CSR и Кнута от плотности матрицы при $N = 1000$. Видно, что формат хранения разреженной матрицы не всегда эффективен, т.е. он может не уменьшать, а увеличивать число хранимых данных (все значения q на рисунке 2.1, для которых $k<1$). Очевидно, что для каждого формата существует граница эффективности (значение q, при котором $k = 1$), значение которой можно оценить аналитически. В таблице 2.2 приведены формулы для оценки плотности матрицы, ниже которой использование форматов неэффективно.

Оценим зависимость коэффициента сжатия от порядка матрицы. На рисунке 2.2 показаны зависимости k от N при $q = 10\%$ для разных форматов. Видно, что коэффициент сжатия с ростом N стремится к максимально возможному значению. Предел выражения для k при $N\to\infty$ даёт аналитическую оценку для определения максимального значения k (таблица 2.3).

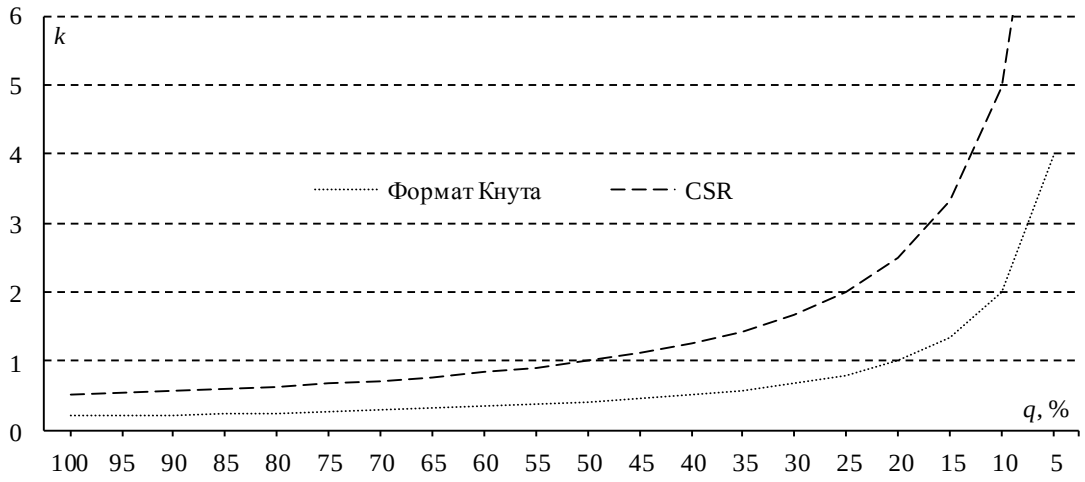


Рисунок 2.1 – Зависимости коэффициентов сжатия от плотности матрицы для форматов Кнута и CSR при $N = 1000$

Таблица 2.2 – Граница эффективности форматов хранения разреженных матриц

Формат	**Граница эффективности формата**
Два вектора	$q = 100\%(N^2–N)/(2N^2)$
Кнута	$q = 100\%(N^2–2N)/(5N^2)$
Рейнболдта и Местеньи	$q = 100\%(N^2–2N)/(3N^2)$
Ларкума*	$q = 100\%(N^2)/(3N^2)$
Шермана*	$q_{\min}^{***} = 100\%(N^2)/(3N^2)$ $q_{\max}^{***} = 100\%(N^2)/(2N^2)$
CSR	$q = 100\%(N^2–N)/(2N^2)$
JDS	$q = 100\%(N^2–N)/(2N^2)$
CDS**	$q = 100\%(N^2)/(N_cN^2)$

* – только для треугольных матриц; ** – только для ленточных матриц; *** – для оптимального ($q_{\min}$) и неоптимального ($q_{\max}$) портретов матрицы; N_c – число диагоналей, содержащих ненулевые элементы

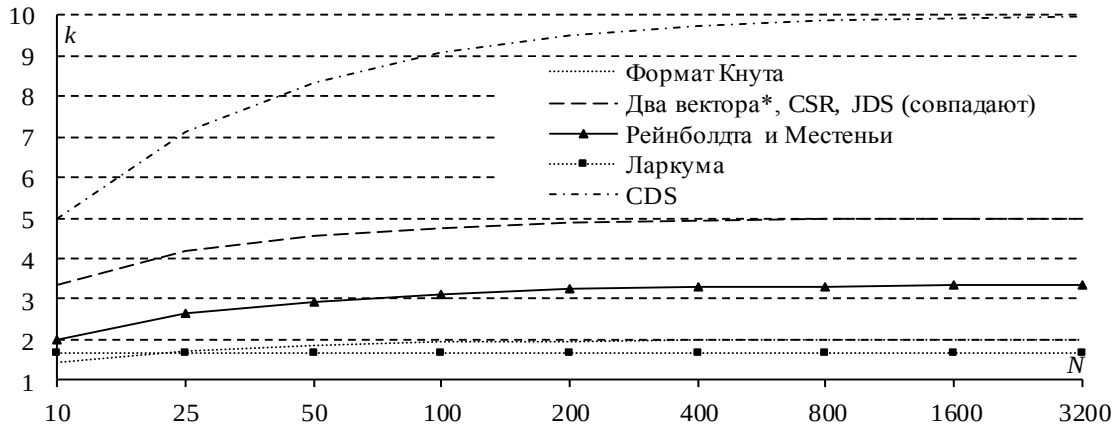


Рисунок 2.2 – Зависимости k от N, при $q = 10\%$ для разных форматов

Таблица 2.3 – Максимальный коэффициент сжатия для разных форматов

Формат	**Максимальный коэффициент сжатия**
Два вектора	$k = 1/(2q)$
Кнута	$k = 1/(5q)$
Рейнболдта и Местеньи	$k = 1/(3q)$
Ларкума*	$k = 1/(3q)$
Шермана*	$k^{***} = 1/(2q)$
CSR	$k = 1/(2q)$
JDS	$k = 1/(2q)$
CDS**	$k = 1/(N_c q)$

* – только для треугольных матриц; ** – только для ленточных матриц; *** – для оптимального портрета матрицы; q – плотность матрицы; N_c – число диагоналей, содержащих ненулевые элементы

Из рисунка 2.2 видно, что формат CDS обеспечивает максимальное сжатие, однако он подходит только к ленточным матрицам и не пригоден для хранения других видов матриц. Из таблицы 2.3 видно, что максимальный коэффициент сжатия обеспечивают форматы: двух векторов, CSR и JDS. Однако, при выборе формата хранения разреженной матрицы необходимо учитывать не только эффективность сжатия формата, но и применимость его в алгоритмах. Поэтому для дальнейшего использования выбран формат CSR, т.к. он наиболее удобен для многих важных операций над разреженными матрицами, таких как: сложение, умножение, перестановка строк и столбцов, транспонирование, решение СЛАУ с разреженными матрицами как прямыми, так и итерационными методами и т.д. [46, 20, 92].

2.2 Разработка программ

В данном разделе приведены результаты разработки программы для ILU(0)-разложения с применением формата CSR, предназначенной для ускорения решения СЛАУ с плотной матрицей итерационными методами с неявным предобусловливанием [101].

ILU(0)-разложение

Для реализации ILU(0)-разложения выбран алгоритм *ikj*-версии (см. разд. 1.3.2.2), т.к. в этом алгоритме элементы матриц **L** и **U** вычисляются

по строкам, поэтому наиболее выгодно использовать формат CSR. Приведем алгоритм ILU(0)-разложения [101].

Алгоритм 2.1 – ILU(0)-разложение (*ikj*-версия)

1 Для $i = 2, \ldots, N$
2 Для $k = 1, \ldots, i - 1$
3 Если $a_{i,k} \neq 0$
4 $a_{i,k} = a_{i,k} / a_{k,k}$
5 Для $j = k + 1, \ldots, N$
6 Если $a_{i,j} \neq 0$
7 $a_{i,j} = a_{i,j} - a_{i,k} \cdot a_{k,j}$
8 Увеличить j
9 Увеличить k
10 Увеличить i

Данный алгоритм позволяет пересчитать i-ю строку матрицы **A** в i-ю строку матриц **L** и **U**. Первые (j–1) строк матрицы **A** участвуют только в определении j-й строки матриц **L** и **U**, что позволяет при вычислениях не хранить их. В формате CSR получить прямой доступ к ненулевому элементу невозможно, т.к. не хранится прямой адрес элемента. Самый простой вариант доступа к элементу – использование функции его поиска. Но при этом алгоритм будет очень медленным. В результате тестирования оказалось, что алгоритм, использующий разреженный формат хранения работает примерно в 500 раз дольше (для $N = 1000$) обычного алгоритма без сжатия матрицы [118]. Следовательно, для разработки программы в таком виде его использовать неэффективно. Поэтому алгоритм был изменен.

Если отметить элементы матрицы, используемые в цикле по j (строки 5–8) алгоритма 2.1, то получится картина, изображенная на рисунке 2.3. Как видно, расчет производится с использованием строк i и k. Поэтому здесь возможно использование цикла по ненулевым элементам в формате CSR. Тогда поиск элементов не нужен, операция (строка 7 в алгоритме 2.1) будет выполняться только с элементами строк i и k, у которых равны индексы столбцов. В результате разработаны алгоритм и соответствующая программа для формирования ILU(0)-разложения с использованием формата CSR. Алгоритм работы программы приведен на рисунке 2.4 (далее программа 1).

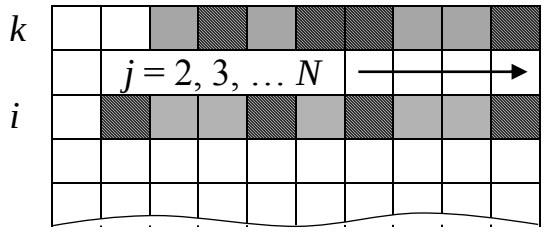


Рисунок 2.3 – Схематичное расположение элементов, используемых за один цикл алгоритма ILU(0)-разложения (*ikj*-версия). Штриховкой отмечены ненулевые элементы, серым – нулевые, белым – элементы, не используемые в цикле

При реализации происходит поиск элемента $a^s_{i,k}$ (блок 1), который используется согласно алгоритма 2.1 в строках 3, 4 и 7, поэтому позиция элемента $a^s_{i,k}$ запоминается (переменная *s*). В блоке 2 присутствует функция *Poisk*(*k, k*) (рисунок 2.5), которая предназначена для поиска элемента $a^s_{k,k}$ в матрице, который используется для выполнения операции в строке 4 алгоритма 2.1. Переменная *pr* сигнализирует, когда достигнут конец строки *i* или *k* в матрице и требуется прервать цикл. Переменные *Elem1* и *Elem2* определяют текущий элемент в первой (*i*-й) и второй (*k*-й) строках соответственно. Переход по элементам строк в матрице осуществляется в цикле по следующему принципу: сравниваются индексы элементов двух текущих строк, индекс с меньшим значением инкрементируется (блоки 4–7), если индексы элементов равны (т.е. существуют ненулевые элементы в двух строках текущего столбца), то выполняется операция и значения индексов обеих переменных инкрементируются (блок 3).

При анализе разработанной программы установлено, что поиск диагонального элемента (блок 2) существенно увеличивает время его работы [120]. Поэтому в формат хранения введен дополнительный вектор **Diag**, в котором хранятся указатели на диагональные элементы. С учетом этого разработана соответствующая программа, алгоритм работы которой приведен на рисунке 2.6.

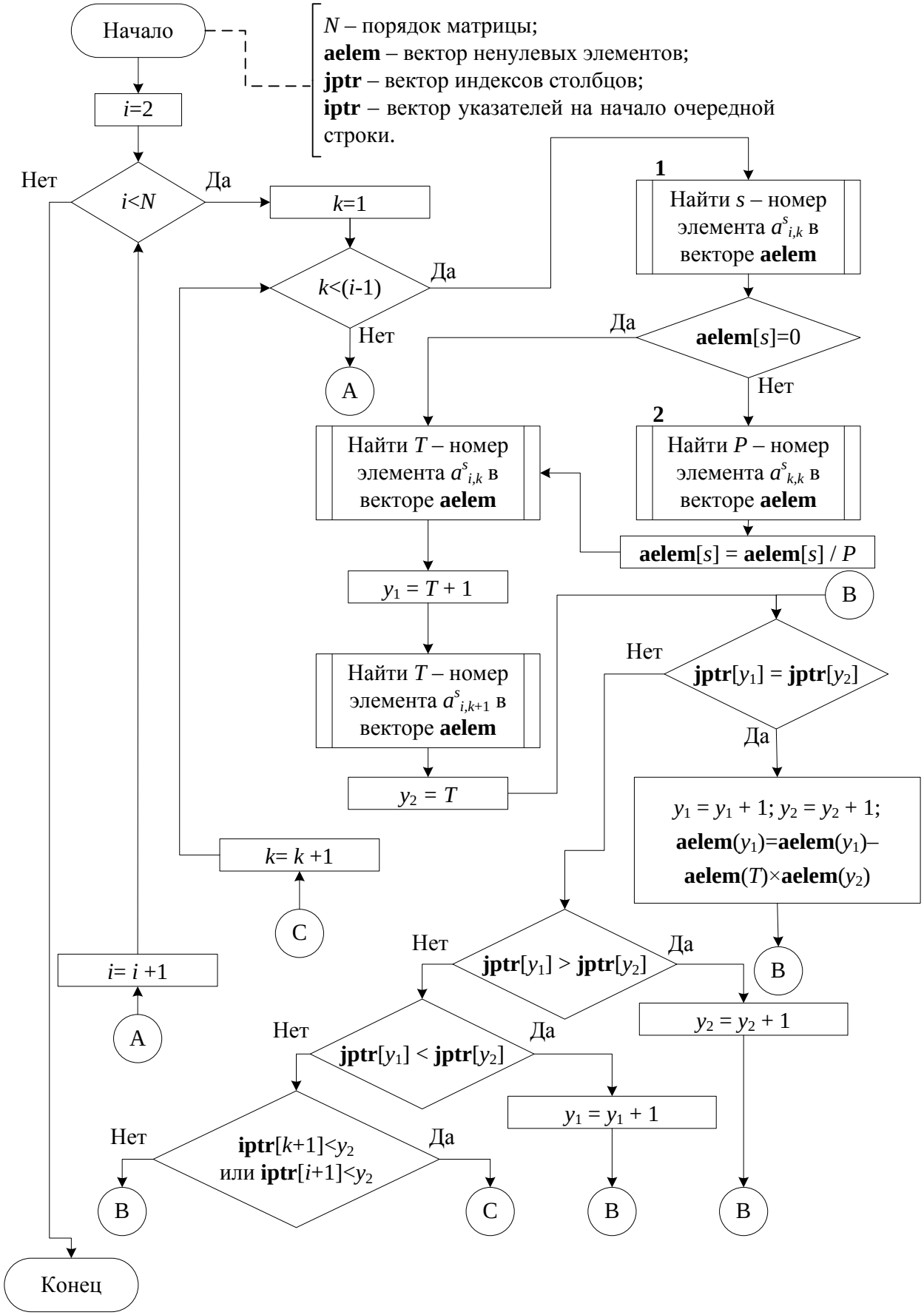


Рисунок 2.4 – Алгоритм работы программы для формирования ILU(0)-разложения с использованием формата CSR

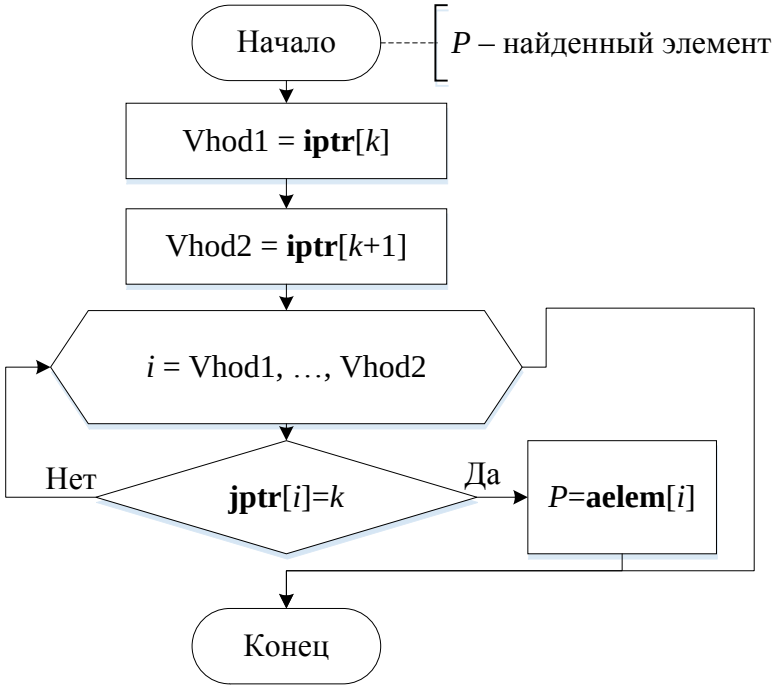


Рисунок 2.5 – Функция поиска элемента $a^s_{k,k}$ в векторе **aelem**

Анализ показал что, разработанная программа (рисунок 2.6) обладает недостатком: тратится время на поиски ненулевых элементов матрицы в блоках 1, 2 и 3. Данные поиски необходимы, так как при использовании формата CSR нет прямой ссылки на первый ненулевой элемент в цикле j (строка 5 в алгоритме 2.1). Таким образом, если уйти от упомянутого поиска, то можно сократить затрачиваемое машинное время. Для этого обратимся к исходному алгоритму 2.1 и схематично расположим элементы матрицы, используемые в нём. Так, на примере $N = 4$, получим рисунок 2.7. Изображения матрицы поделены по шагам так, что один шаг соответствует одному значению переменных i и k. Видно, что элементы a_{ik} участвуют в цикле, который всегда начинается с первого элемента строки, и перемещаются по строкам до последнего элемента в нижнетреугольной матрице. Данную цикличность можно использовать в алгоритме с форматом CSR, поскольку движение элемента происходит по строкам. Причем в этом случае нет необходимости искать элемент a_{ik} в каждом цикле (блок 1 на рисунке 2.6).

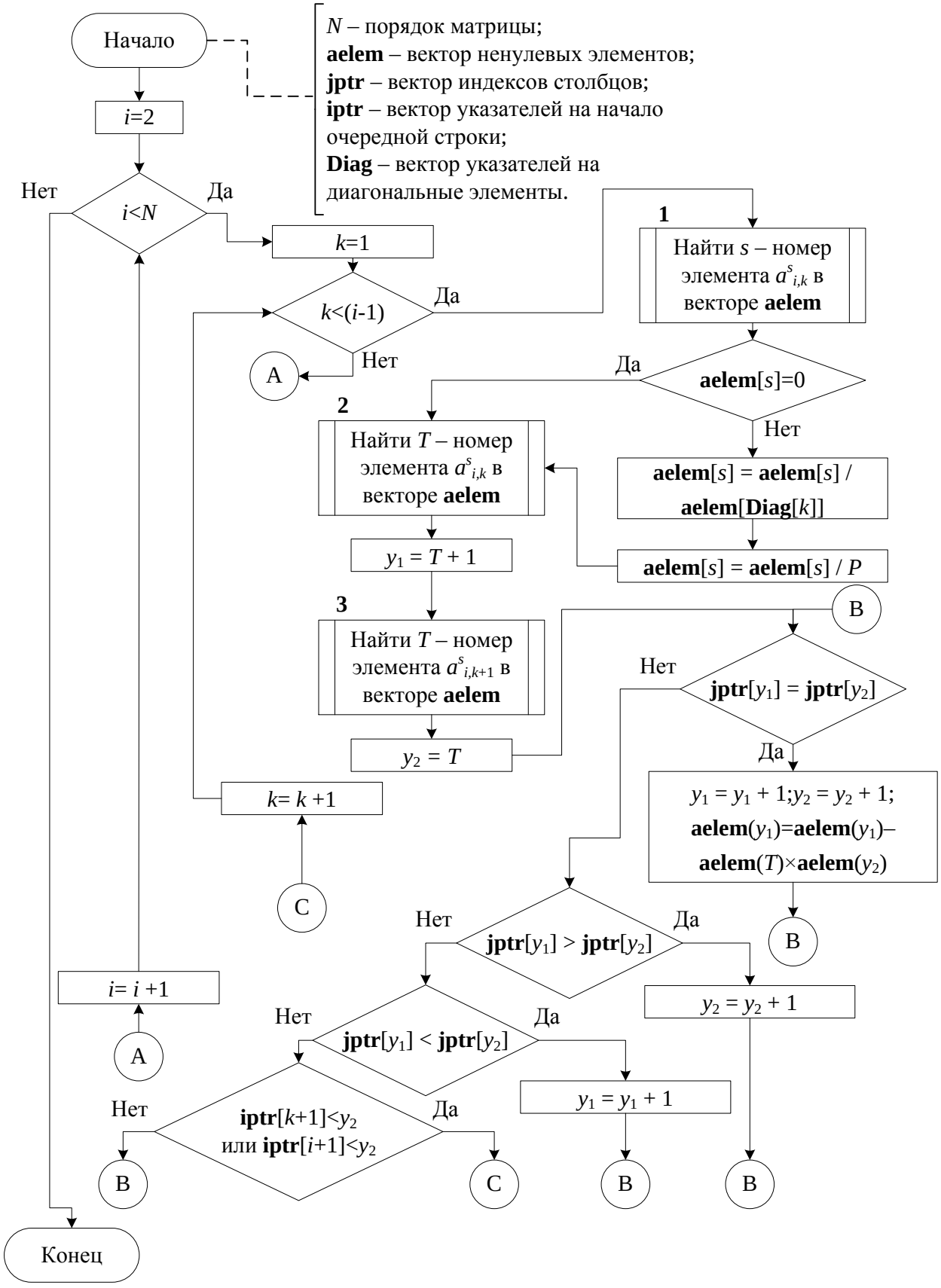


Рисунок 2.6 – Алгоритм работы программы для формирования ILU(0)-разложения с использованием формата CSR с дополнительным вектором Diag

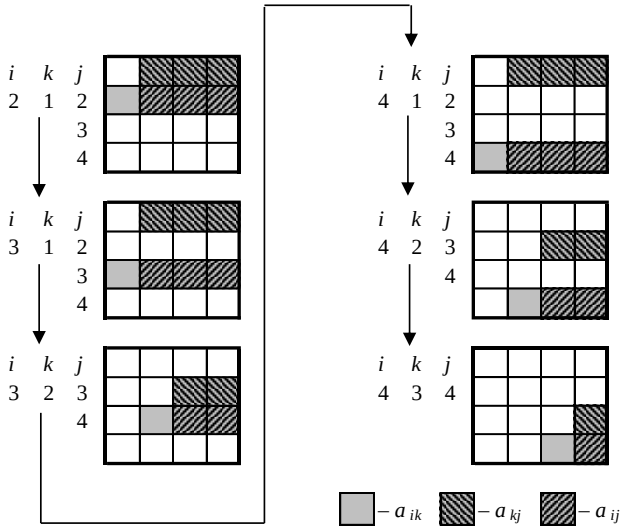


Рисунок 2.7 – Схематичное расположение элементов, используемых за все циклы ILU(0)-разложения

В алгоритме работы программы (рисунке 2.6) имеется еще одна особенность. Так, начальный элемент цикла *kj* всегда находится после соответствующего диагонального элемента. Этот факт позволяет исключить поиск элемента a_{kj} (блок 3 рисунок 2.6). С учетом этих особенностей программы (рисунке 2.6), поиски ненулевых элементов можно исключить. В результате разработана ещё одна программа, алгоритм работы которой приведен на рисунке 2.8. Из данного алгоритма видно, что в программе существует избыточность: большое количество сравнений в блоках 1–3. В данных строках происходит сравнение индексов столбцов ненулевых элементов, находящихся в одной строке, с индексами ненулевых элементов другой строки, и если совпадение обнаружено, то происходит операция над этими элементами. Схематично данная операция изображена на рисунке 2.9 (результат выполнения операций над элементами строк обозначен как Вектор–результат). Для выполнения этой операции необходимо перебирать и сравнивать между собой все индексы столбцов ненулевых элементов двух векторов (строк), что неэффективно.

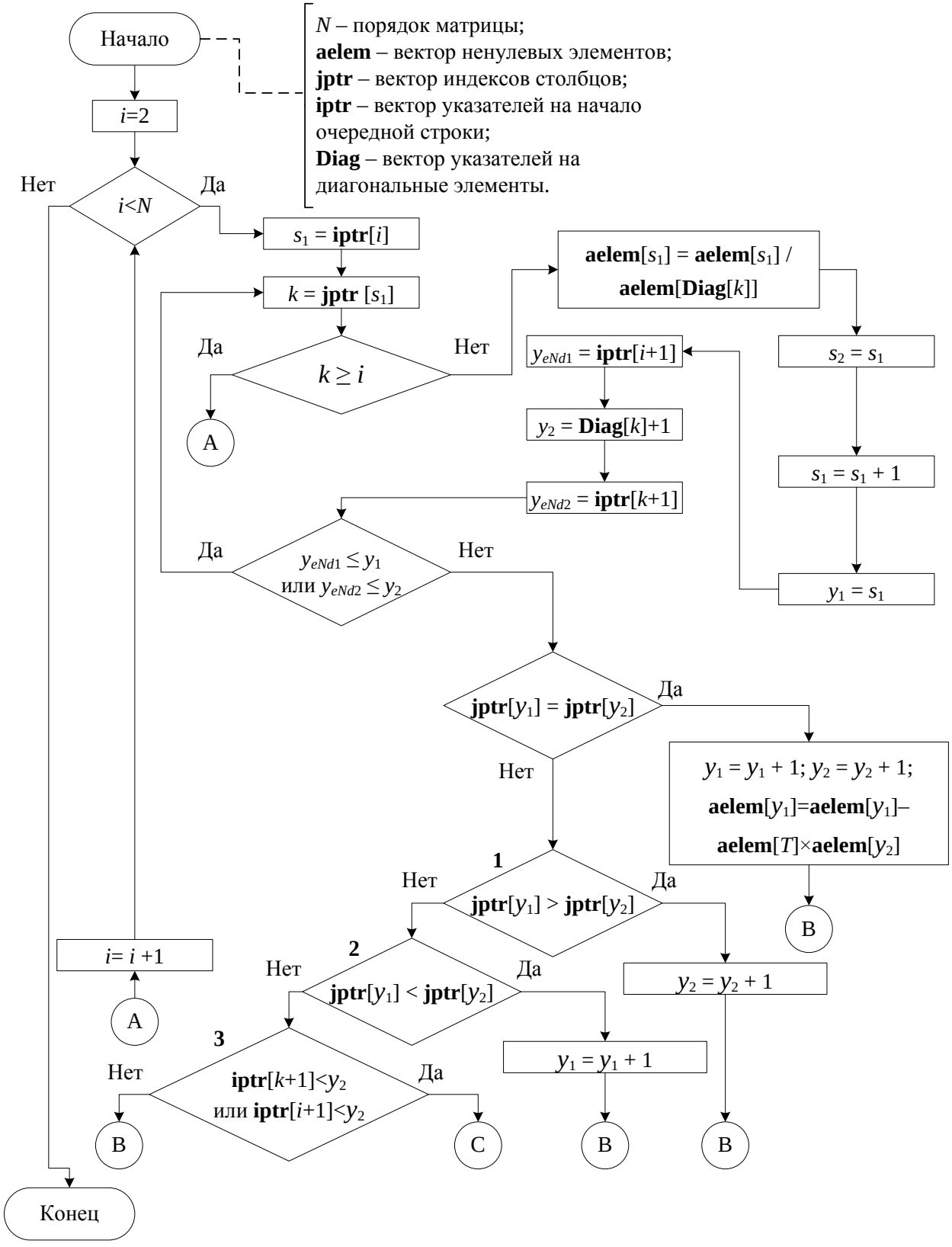


Рисунок 2.8 – Алгоритм работы программы для формирования ILU(0)-разложения с использованием формата CSR без поиска ненулевых элементов

Для этой операции можно воспользоваться временными векторами, так как в формате CSR хранятся не только сами элементы (массив **aelem**), но и адреса

столбцов, в которых находятся эти элементы (массив **jptr**), что дает возможность построить циклы без дополнительных условий. Таким образом, суть предлагаемого усовершенствования заключается в том, что операция поиска одинаковых индексов столбцов двух векторов (строк) разделена на два этапа. На первом этапе создаются векторы **tmpvec** и **tmpjptr**: в первый записываются статусы наличия ненулевых элементов в анализируемом векторе (строке), а во втором хранятся соответствующие адреса столбцов этих ненулевых элементов. На втором этапе сравниваются элементы второго вектора (строки) с элементами временного вектора (**tmpvec**), и если найдены ненулевые элементы с одинаковыми индексами столбцов, то выполняется соответствующая операция над элементами. Схематически это показано на рисунке 2.10.

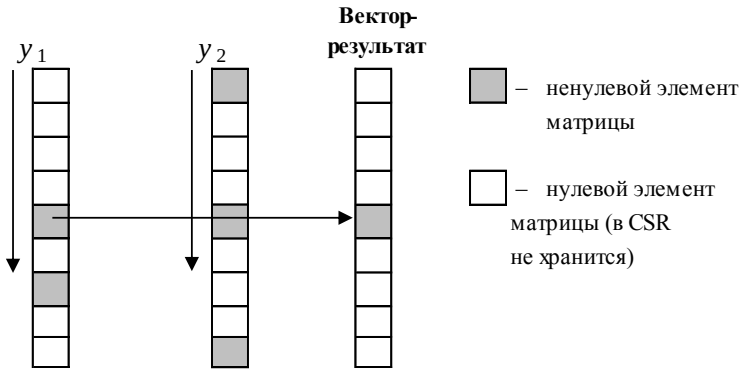


Рисунок 2.9 – Схематичное изображение операции сравнения ненулевых элементов в программе (рисунок 2.8)

Очевидно, что выгоднее записывать данные строки *i* во временные векторы, поскольку она используется на протяжении всего шага по *i* (см. алгоритм 2.1), т.е. временные векторы будут создаваться один раз и использоваться на протяжении одного шага цикла *i*. С учетом сказанного разработана программа, алгоритм работы которой приведен на рисунке 2.11. В векторе **tmpvec** хранится значение *Истина*, если элемент ненулевой, а в векторе **tmpjptr** хранится адрес этого ненулевого элемента. Из приведенного алгоритма видно, что необходима дополнительная память компьютера для хранения *N* булевых элементов вектора **tmpvec** и *N* целочисленных значений их адресов в векторе **tmpjptr**. Такое увеличение машинной памяти, по срав-

нению с памятью для хранения основной матрицы, незначительно и далее не учитывалось.

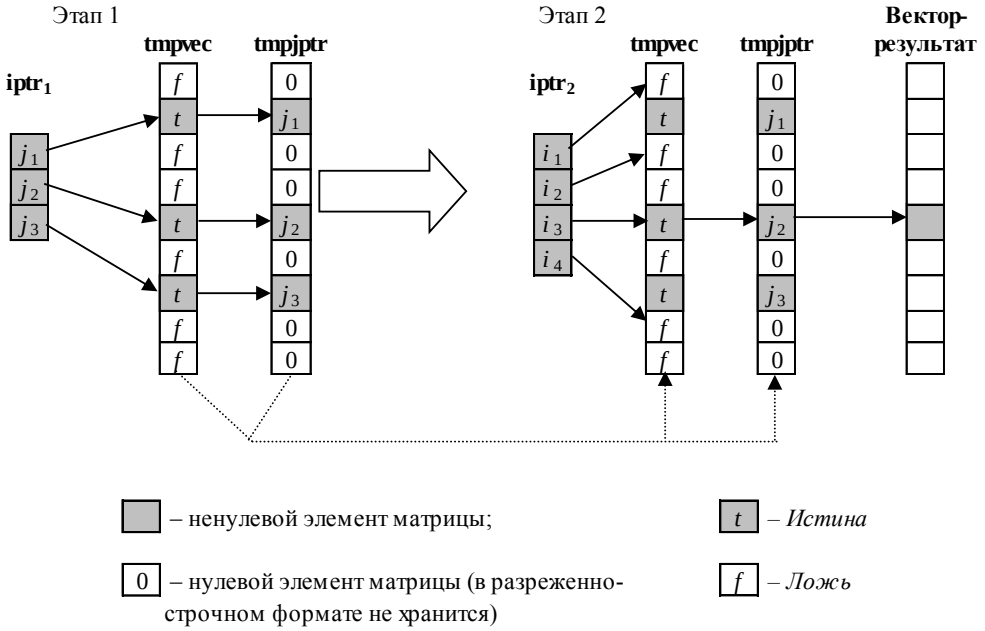


Рисунок 2.10 – Схематичное изображение этапов операции над ненулевыми элементами в алгоритме ILU(0)-разложения с использованием временных векторов

Использование разреженного строчного формата в итерационном методе

В итерационном методе, при неявном предобусловливании, на каждой итерации решается СЛАУ вида **My** = **z** (см. разд. 1.3.3.2.2). При этом матрица **M** хранится в формате CSR. Тогда за основу можно использовать алгоритм 2.2 [20].

Алгоритм 2.2 – Решение СЛАУ с матрицей в формате CSR

1 Для $i = 1, \ldots, N$
2 $y_i = z_i$
3 Для $k =$ **iptrL**$(i), \ldots,$ **iptrL**$(i+1)$
4 $y_i = y_i -$ **aelemL**$(k)\ y_{\textbf{jptrL}(k)}$
5 Увеличить k
6 $z_i = y_i /$ **aelemL**(**DiagL**(i))
7 Увеличить i
8 Для $i = N, \ldots, 1$
9 $z_i = y_i /$ **aelemU**(**DiagU**(i))
10 Для $k =$ **iptrU**$(i), \ldots,$ **iptrU**$(i+1)-1$
11 $y_i = y_i -$ **aelemU**$(k)\ y_{\textbf{jptrU}(k)}$
12 Увеличить k
13 Уменьшить i

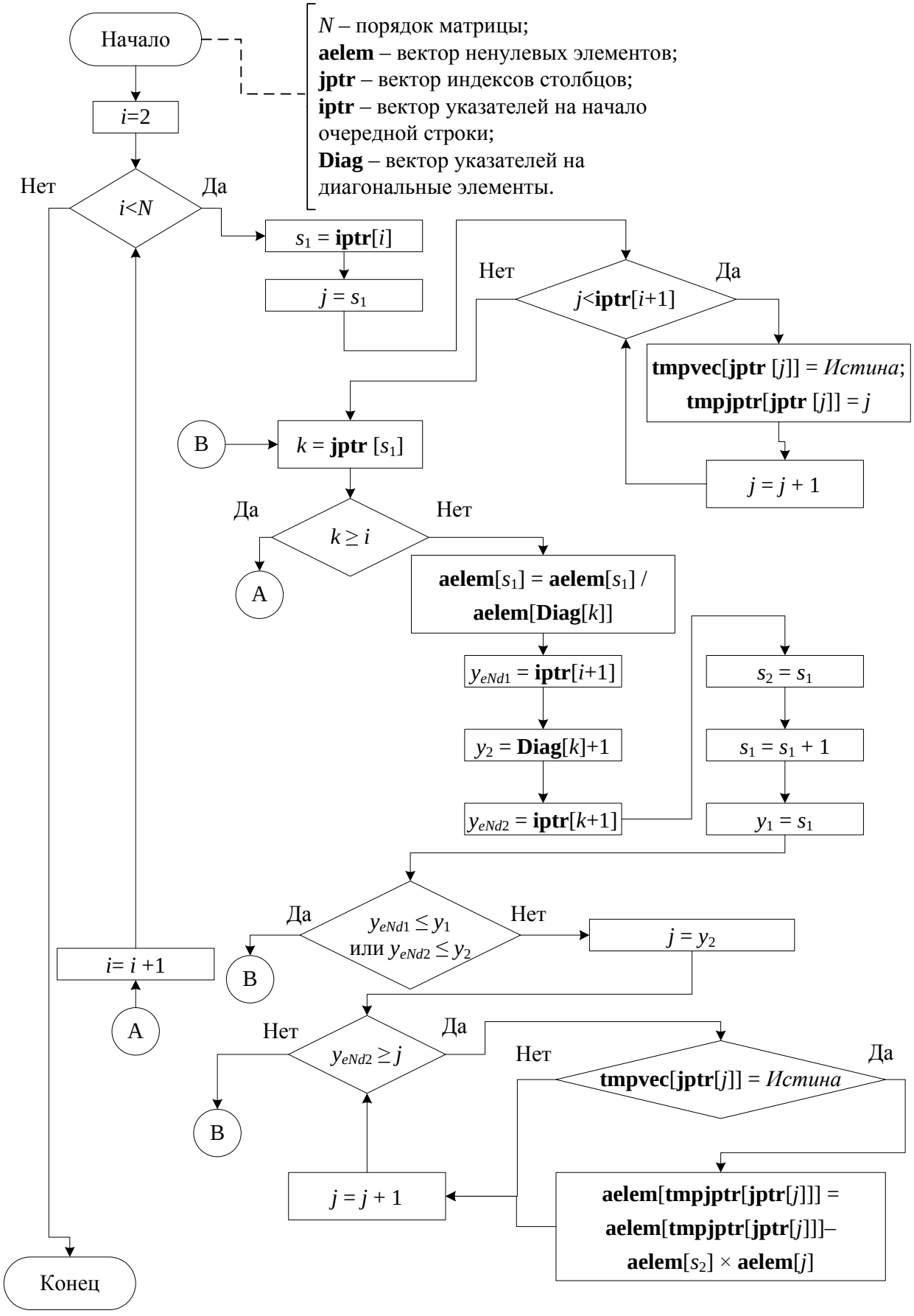


Рисунок 2.11 – Алгоритм работы программы для формирования ILU(0)-разложения с использованием формата CSR с двумя дополнительными векторами

Строки 1–7 – прямой ход, строки 8–13 – обратный ход. Циклы в строках 3 и 10 выполняют обход по элементам матрицы. Приведенный алгоритм требует разделения на подматрицы **L** (массивы **aelemL**, **iptrL** и **jptrL**) и **U** (массивы **aelemL**, **iptrL** и **jptrL**). Однако в рассматриваемом случае (см. алгоритм 2.1) такого разделения нет. С учетом этой особенности был разработан алгоритм 2.3.

Алгоритм 2.3 – Решение СЛАУ с матрицей в модифицированном формате CSR

1 Для $i = 1, \ldots, N$
2 $s_1 = \mathbf{iptr}(i)$
3 $s_2 = \mathbf{iptr}(i+1)$
4 $y_i = z_i$
5 Для $k = s_1, \ldots, s_2$
6 Если $\mathbf{jptr}(k)<i$
7 $y_i = y_i - \mathbf{aelem}(k)\, y_{\mathbf{jptr}(k)}$
8 Увеличить k
9 Увеличить i
10 Для $i = N, \ldots, 1$
11 $s_1 = \mathbf{iptr}(i)$
12 $s_2 = \mathbf{iptr}(i+1)$
13 Для $k = s_1, \ldots, s_2$
14 Если $\mathbf{jptr}(k)>i$
15 $y_i = y_i - \mathbf{aelem}(k)\, y_{\mathbf{jptr}(k)}$
16 Увеличить k
17 $y_i = y_i / \mathbf{aelem}(\mathbf{Diag}(i))$
18 Уменьшить i

Строки 1–9 – прямой ход, 10–18 – обратный ход. Циклы в строках 5 и 13 выполняют обход по элементам матрицы. Условие в строке 6 обеспечивает обход только элементов нижней треугольной подматрицы (**L**), а условие в строке 14 – верхней треугольной подматрицы (**U**). Тем самым, обеспечивается решение без разделения на подматрицы **L** и **U**, как это сделано в алгоритме 2.2.

2.3 Вычислительные эксперименты

Для подтверждения эффективности алгоритмов, предложенных в разделе 2.2, проведены вычислительные эксперименты [118, 120]. Использовался персональный компьютер (при вычислениях не использовалось

распараллеливание, т.е. работало одно ядро процессора) с параметрами: платформа AMD Athlon(tm) 64 X2 Dual; частота процессора 2100 МГц; объем ОЗУ 2 Гбайт; число ядер – 2; операционная система Windows XP.

Для подтверждения эффективности использования дополнительного вектора **Diag** проведено сравнение времени работы программы с использованием формата CSR (рисунок 2.4) (T) и программы с дополнительным вектором **Diag** (рисунок 2.6) (T_D) при различной плотности (q) матрицы **M** порядка $N = 1000$. Разная плотность матрицы **M** получена с помощью предфильтрации (1.24) с различным порогом обнуления (ε). Полученные результаты сведены в таблицу 2.4. Как видно, использование вектора **Diag** ускоряет ILU(0)-разложение в 1,14–1,23 раза.

Таблица 2.4 – Сравнение времени выполнения программ, алгоритмы которых представлены на рисунках 2.4 и 2.6

ε	q, %	T, с	T_D, с	T / T_D
0,012	59	285	258	1,14
0,014	56	261	234	1,13
0,016	53	242	213	1,16
0,018	50	224	195	1,15
0,02	47	207	178	1,16
0,022	45	192	162	1,17
0,024	42	175	146	1,17
0,026	40	163	133	1,17
0,028	38	152	123	1,18
0,03	36	142	112	1,19
0,032	35	132	103	1,19
0,034	33	124	95	1,20
0,036	31	116	88	1,20
0,038	30	109	81	1,20
0,04	29	103	75	1,21
0,042	27	97	70	1,21
0,044	26	91	65	1,23
0,046	25	86	60	1,22
0,048	24	81	56	1,23
0,05	23	73	52	1,23

Далее сравнивалась эффективность программы, основанной на алгоритме 2.1 (ILU(0)-разложение, *ikj*-версия), и программы для формирования ILU(0)-разложение с использованием формата CSR с дополнительным вектором **Diag** (рисунок 2.6). Для реализации выбран итерационный метод BiCGStab, т.к. он, используемый совместно с неявным предобусловливанием,

хорошо зарекомендовал себя при решении СЛАУ с плотной матрицей [54, 126]. Измерялось время трёх составляющих полного решения СЛАУ: предфильтрации (в программе ILU(0)-разложения, использующей разреженный формат, в ходе предфильтрации выполняется и конвертация матрицы в формат CSR [127]); ILU(0)-разложения; итерационного процесса.

Оценка временных затрат сделана для трёх СЛАУ, матрицы которых получены в системе TALGAT последовательным учащением сегментации границ проводник-диэлектрик и диэлектрик-диэлектрик из задачи вычисления электрической ёмкости двух полосок на диэлектрическом слое над идеально проводящей плоскостью (рисунок 2.12) для двух вариантов алгоритма BiCGStab: без использования формата хранения разреженных матриц (*T*) и с использованием формата CSR с дополнительным вектором **Diag** (*Tcsrd*).

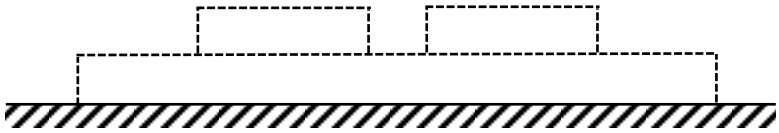


Рисунок 2.12 – Исследуемая конфигурация – два проводника с диэлектриком над идеально проводящей плоскостью

В экспериментах использовались матрицы с $N = 4800, 6000, 8000$. При помощи предфильтрации (1.24) изменялся порог обнуления (ε), меняющий плотность матрицы **M**. Так, для $N = 4800$ порог менялся в диапазоне 0, $1{,}5\times10^{-6}$, … $1{,}75\times10^{-5}$; для $N = 6000$ – 0, $1{,}0\times10^{-7}$, … $1{,}0\times10^{-5}$; для $N = 8000$ – 0, $5{,}0\times10^{-3}$, … $1{,}0\times10^{-1}$. Итерации продолжались, пока относительная норма вектора невязки не становилась меньше 10^{-6}.

Сравнение времени двух алгоритмов для различных порядков матриц приведено на рисунке 2.13. Видно, что ускорение алгоритма с использованием формата CSR получается не при всех значениях плотности, а только для значений меньших определенного порогового значения. При этом максимальное ускорение от использования формата хранения разреженной матрицы достигается приблизительно при оптимальном значении порога обнуления, соответствующем минимальному времени решения СЛАУ. В таблице 2.5 приведены данные сравнения алгоритмов при различных поряд-

ках матриц. Видно, что ускорение имеет место на различных матрицах, изменяясь в пределах от 1,5 до 1,6 раза. Дополнительно приведено время решения методом Гаусса (*Tge*) и ускорение относительно него, изменяющееся в пределах от 1,2 до 4,4 раза (использовался алгоритм решения методом Гаусса, основанный на LU-разложении и последующем решении).

Таблица 2.5 – Сравнение времени решения СЛАУ при разных *N*

N	*T*, с	*Tcsrd*, с	*T*/*Tcsrd*	*Tge*, с	*Tge*/*Tcsrd*
4800	164	101	1,6	129	1,3
6000	335	206	1,6	254	1,2
8000	216	140	1,5	616	4,4

Если проанализировать время работы алгоритма решения СЛАУ по составляющим, то выясняется, что ускорение получается именно за счет вычисления ILU(0)-разложения. На рисунке 2.13 приведено время работы алгоритма ILU(0)-разложения для $N = 4800$ (рисунок 2.13*б*), $N = 6000$ (рисунок 2.13*г*), $N = 8000$ (рисунок 2.13*е*). Вычисление ILU(0)-разложения занимает часть общего времени решения СЛАУ, следовательно, его сокращением достигается общее ускорение. Ускорение ILU(0)-разложения достигается за счет использования формата CSR, т.к. в данном формате нулевые элементы матрицы не хранятся, и, следовательно, не участвуют в алгоритме, в отличие от алгоритма, который работает с полной матрицей.

Для вычислительного эксперимента последующих алгоритмов использовался персональный компьютер (при вычислениях не использовалось распараллеливание, т.е. работало одно ядро процессора) с параметрами: платформа – Intel(R) Core(TM) i7 CPU 970; частота процессора – 3,20 ГГц; объем ОЗУ – 16 Гбайт; число ядер – 6; операционная система – Linux Kubuntu11.10 x64.

Оценка временных затрат сделана на нескольких СЛАУ (полученных последовательным учащением сегментации границ проводник-диэлектрик и диэлектрик-диэлектрик из задачи вычисления электрической ёмкости), решаемых методом BiCGStab с использованием предобусловливания и формата CSR (рисунок 2.4), с дополнительным вектором **Diag** (рисунок 2.6) и

программы без поиска ненулевых элементов (рисунок 2.8). Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-6}.

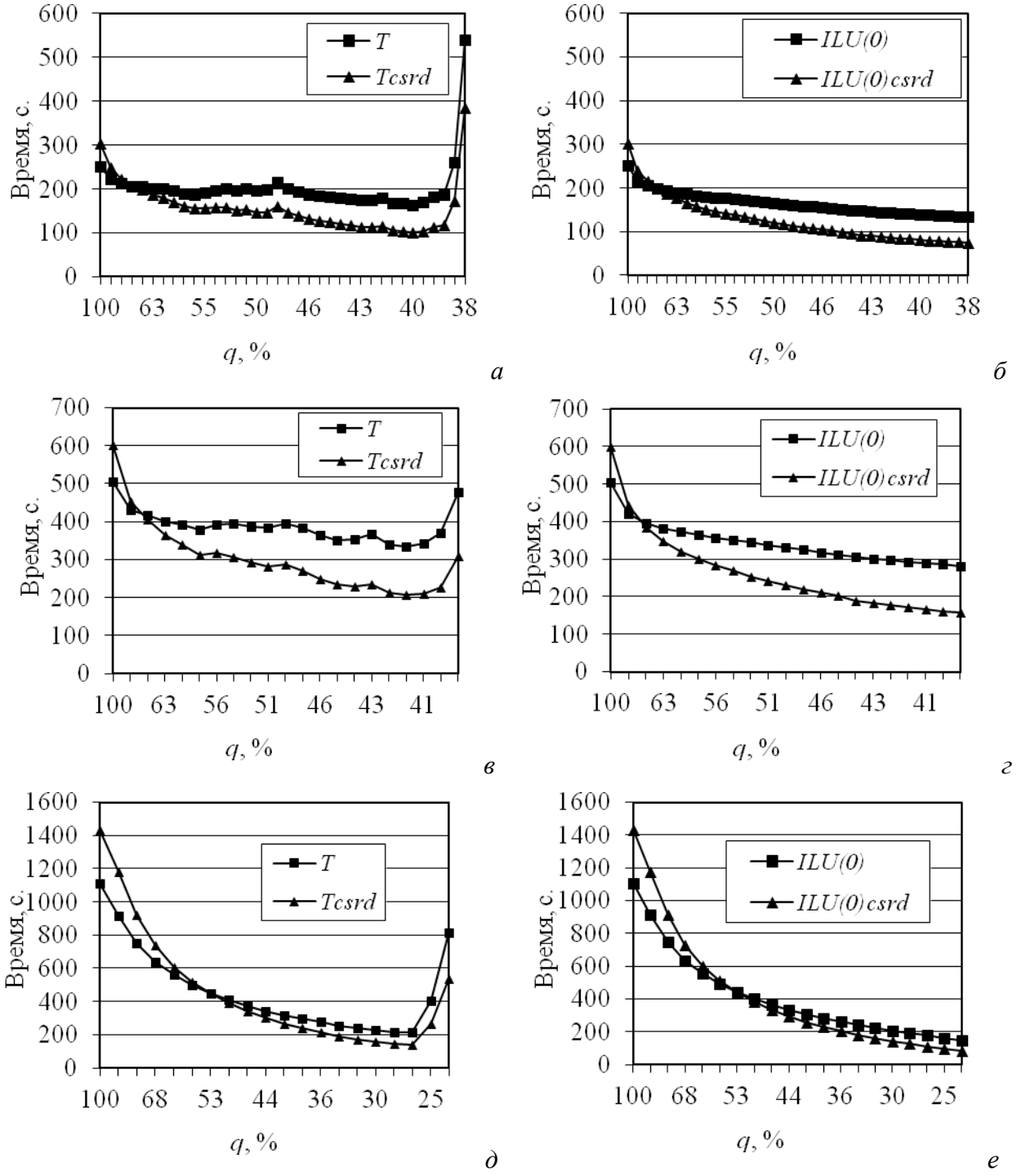


Рисунок 2.13 – Сравнение времени работы программ:
без использования формата хранения разреженных матриц
(*T* – время решения СЛАУ методом BiCGStab, *ILU(0)* – время вычисления ILU(0)-разложения); с использованием формата CSR с дополнительным вектором **Diag**
(*Tcsrd* – время решения СЛАУ методом BiCGStab,
ILUcsrd (0) – время вычисления ILU(0)-разложения)

Для оценки ускорения решения СЛАУ использовано отношение времени решения СЛАУ с помощью программы, использующей дополнительный вектор **Diag** (рисунки 2.6) (*Ts*) и программы без поиска ненулевых элементов (рисунок 2.8) (*Tr*) ко времени работы программы (рисунок 2.4) (*T*). Вид конфигурации 1, состоящей из двух проводников на слое диэлектрика над идеально проводящей плоскостью, приведен на рисунке 2.12, а конфигурации 2, состоящей из двух проводников (без диэлектрика) над идеально проводящей плоскостью – на рисунке 2.14. Полученные зависимости ускорения решения СЛАУ от плотности (*q*) матрицы **M** приведены на рисунке 2.15.

Рисунок 2.14 – Исследуемая конфигурация – два проводника
над идеально проводящей плоскостью

Из рисунка 2.15 следует, что усовершенствованные программы, алгоритмы которых приведены на рисунках 2.6 и 2.8, работают быстрее программы (рисунке 2.4). Максимальные ускорения получены при $N = 4800$: в 1,6 и 2,5 раза. При $N = 8000$ максимальные ускорения составили 1,2 и 1,7 соответственно. Также видно, что при малых плотностях матрицы **M** ускорение падает. Этот факт объясняется тем, что в данной работе оптимизировался только этап ILU(0)-разложения, т.е. одна из трех составляющих общего времени: предфильтрации (*Tp*), ILU(0)-разложения (*Tlu*) и итерационного процесса (*Ti*). На рисунке 2.16 приведено ускорение ILU(0)-разложения при использовании программы без поиска ненулевых элементов (рисунок 2.8) (*Tlus*) и программы с двумя дополнительными векторами (рисунок 2.11) (*Tlur*) относительно программы, использующей дополнительный вектор **Diag** (рисунок 2.6) (*Tlu*). Видно, что для $N = 4800$ максимальное ускорение для программы без поиска ненулевых элементов (рисунок 2.8) составляет 2 раза, а для программы с двумя дополнительными векторами (рисунок 2.11) – почти 4 раза. На всех графиках ускорение увеличивается с уменьшением плот-

ности матрицы. Нет отсутствия ускорения при минимальной плотности матрицы, как это было на рисунке 2.10.

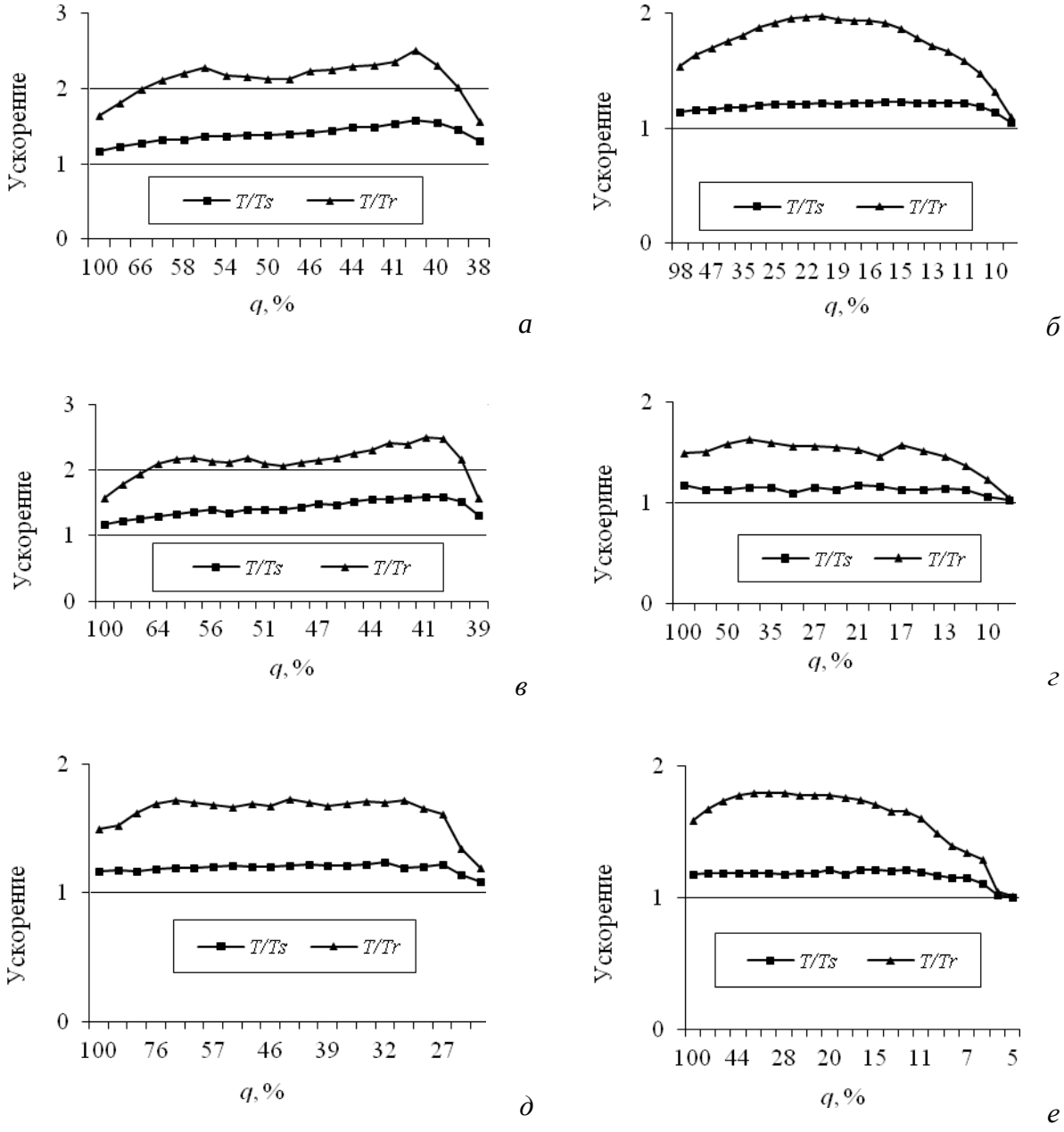


Рисунок 2.15 – Ускорение решения СЛАУ методом BiCGStab при использовании программы, использующей дополнительный вектор **Diag** (рисунок 2.6) (*T*) относительно программы без поиска ненулевых элементов (рисунок 2.8) (*Ts*) и программы с двумя дополнительными векторами (рисунок 2.11) (*Tr*)
для конфигурации 1 (рисунок 2.12) при N = 4800 (*а*), 6000 (*в*), 8000 (*д*)
и конфигурации 2 (рисунок 2.14) при N = 2240 (*б*), 6800 (*г*), 7999 (*е*)

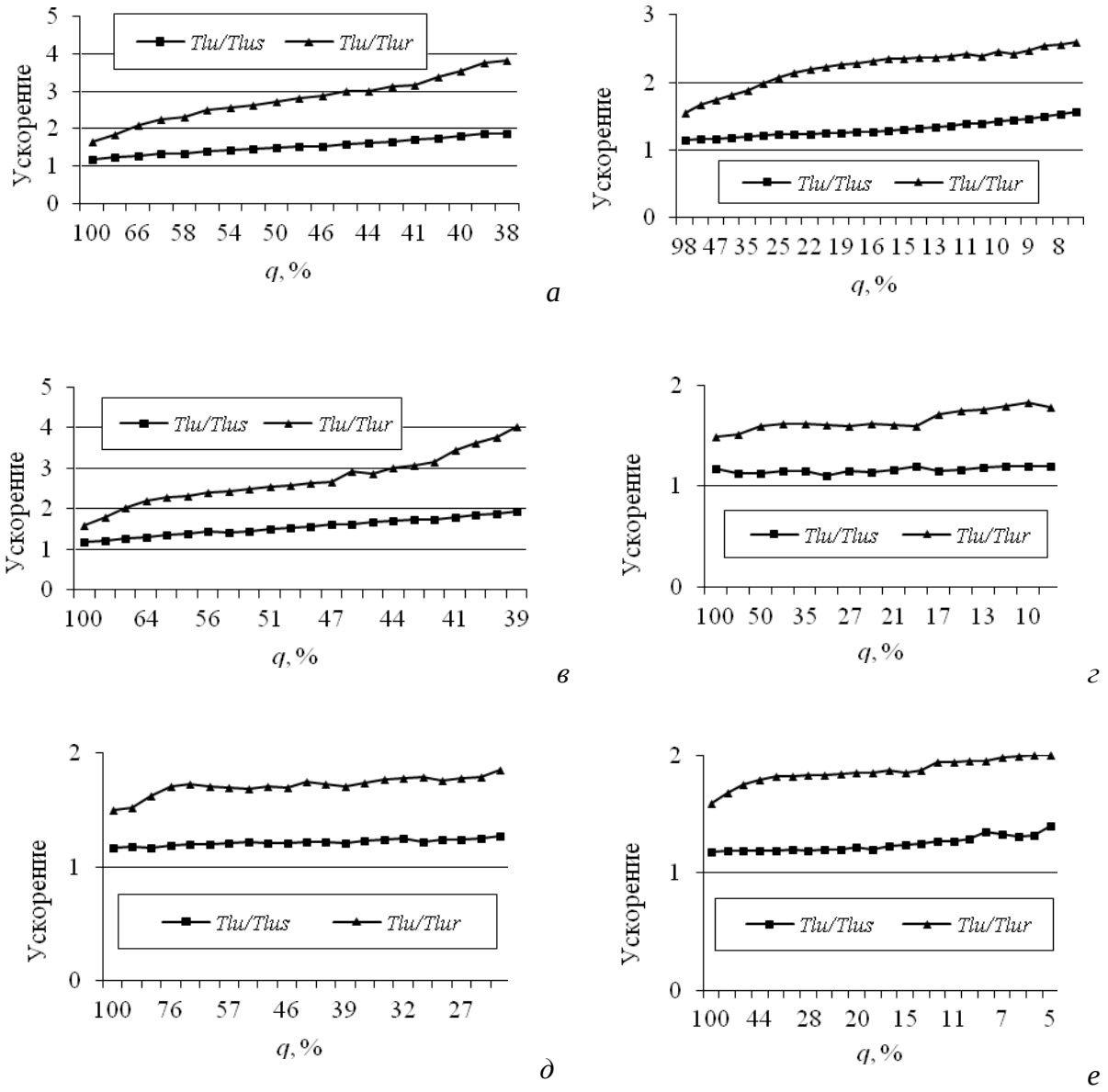


Рисунок 2.16 – Ускорение ILU(0)-разложения при использовании программы, использующей дополнительный вектор **Diag** (рисунок 2.6) (*Tlu*) относительно программы без поиска ненулевых элементов (рисунок 2.8) (*Tlus*) и программы с двумя дополнительными векторами (рисунок 2.11) (*Tlur*) для конфигурации 1 (рисунок 2.12) при N = 4800 (*а*), 6000 (*в*), 8000 (*д*) и для конфигурации 2 (рисунок 2.14) при N = 2240 (*б*), 6800 (*г*), 7999 (*е*)

Показательно сравнение ускорения на разных этапах решения при использовании программы, использующей дополнительный вектор **Diag** (рисунок 2.6) и программы с двумя дополнительными векторами (рисунок 2.11). Для наглядности зависимость соотношения времени этапов решения СЛАУ по программе с двумя дополнительными векторами (рисунок 2.11) от плотности матрицы **M** (q) для N = 4800 изображена на рисунке 2.17.

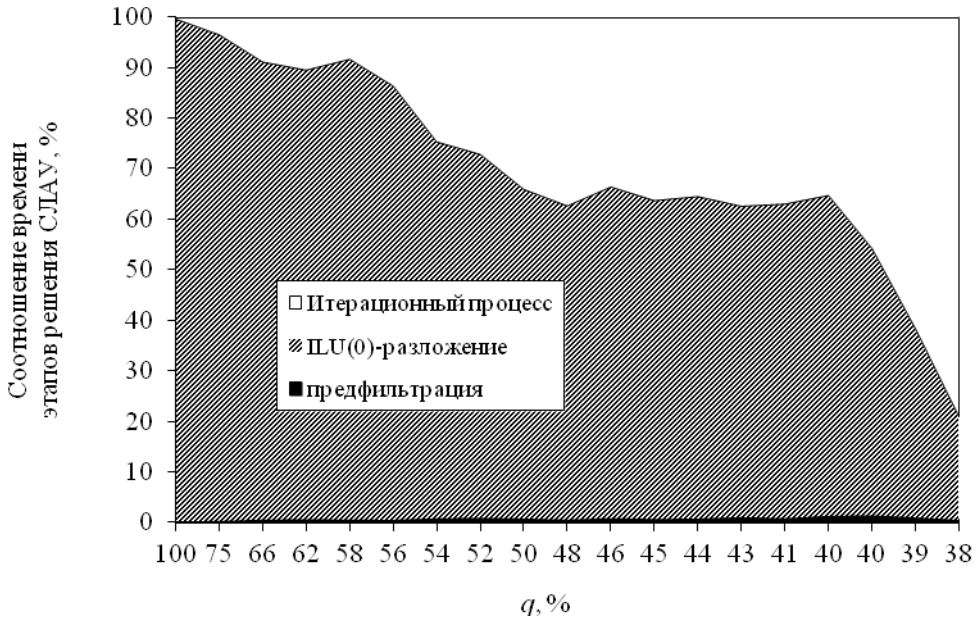


Рисунок 2.17 – Зависимость соотношения времени, затрачиваемого на этапы решения СЛАУ итерационным методом с предобусловливанием, от плотности матрицы **M** для N = 4800

Из рисунка 2.17 ясно видно, что при уменьшении плотности матрицы часть времени, затрачиваемая на итерационный процесс, увеличивается, а на ILU(0)-разложение (за счет которого получено ускорение) – уменьшается. Следовательно, ускорение общего времени решения СЛАУ также уменьшается. Между тем, уменьшение ускорения при малой плотности матрицы не уменьшает значимости усовершенствования алгоритма. Действительно, на практике требуется минимизировать время решения СЛАУ за счет выбора оптимального значения порога обнуления. При этом значение времени ILU(0)-разложения примерно равно времени итерационного процесса [128]. Поэтому уменьшение времени вычисления ILU(0)-разложения весьма важно.

2.4 Сравнение с результатами других авторов

Проведено сравнение результатов, полученных в данной работе, с результатами других авторов [129]. В качестве анализируемой структуры взята широко используемая связанная микрополосковая линия, поперечное сечение которой приведено на рисунке 2.18. Использовано 2 набора параметров структуры, которые взяты из работ [130, 131]. Для наглядности параметры

сведены в табл. 2.6. Для сравнения использовались значения погонных коэффициентов электростатической (C_{11}, C_{12}) и электромагнитной (L_{11}, L_{12}) индукций.

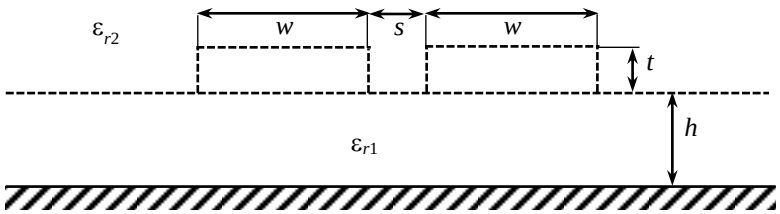


Рисунок 2.18 – Связанная микрополосковая линия

Таблица 2.6 – Значения параметров для связанной микрополосковой линии

Параметр	Набор	
	1	2
w, мм	0,9	3
t, мм	0,05	1
s, мм	0,8	2
h, мм	1	1
ε_{r1}	10	2
ε_{r2}	1	1

Учащением сегментации структуры были получены СЛАУ с матрицами разных порядков. Для реализации выбран итерационный метод BiCGStab. Итерации продолжались, пока относительная норма вектора невязки не становилась меньше 10^{-8}. В табл. 2.7 приведено сравнение значений погонных коэффициентов электростатической и электромагнитной индукций, полученных с использованием программ: по алгоритму 2.1, программы с использованием формата CSR (рисунок 2.4), использующей дополнительный вектор **Diag** (рисунок 2.6), без поиска ненулевых элементов (рисунок 2.8) и программы с двумя дополнительными векторами (рисунок 2.11), с результатами, полученными другими авторами. Из табл. 2.7 видно, что для обоих наборов параметров результаты для разных алгоритмов не отличаются, а из-за учащения сегментации значения коэффициентов матрицы электростатической индукции отличаются не существенно (C_{12} менее 4%, C_{11} менее 1%), тогда как электромагнитной – ещё меньше. Результаты алгоритмов, в целом, согласу-

ются с результатами других авторов (отклонения для C_{11} не более 9%, для C_{12} не более 24%, для L_{11} не более 7%, для L_{12} не более 20%).

Таблица 2.7 – Значения погонных коэффициентов связанной микрополосковой линии

Источник	N	C_{11}, пФ/м	$-C_{12}$, пФ/м	L_{11}, мкГн/м	$-L_{12}$, мкГн/м
Набор параметров 1					
Алгоритм 2.1	650	146,65	23,71	0,421	0,099
	975	146,80	23,98	0,421	0,099
	1300	146,74	24,29	0,421	0,099
	1950	147,03	24,33	0,421	0,099
Программа с использованием формата CSR (рисунок 2.4)	650	146,65	23,71	0,421	0,099
	975	146,80	23,98	0,421	0,099
	1300	146,74	24,29	0,421	0,099
	1950	147,03	24,33	0,421	0,099
Программа без поиска ненулевых элементов (рисунок 2.8)	650	146,65	23,71	0,421	0,099
	975	146,80	23,98	0,421	0,099
	1300	146,74	24,29	0,421	0,099
	1950	147,03	24,33	0,421	0,099
Программа с двумя дополнительными векторами (рисунок 2.11)	650	146,65	23,71	0,421	0,099
	975	146,80	23,98	0,421	0,099
	1300	146,74	24,29	0,421	0,099
	1950	147,03	24,33	0,421	0,099
[130]		159,81	20,81	0,422	0,100
Набор параметров 2					
Алгоритм 2.1	68	90,73	8,86	0,200	0,030
	220	90,64	9,11	0,198	0,030
	330	90,90	9,14	0,198	0,030
	1100	91,35	9,18	0,198	0,030
Программа с использованием формата CSR (рисунок 2.4)	68	90,73	8,86	0,200	0,030
	220	90,64	9,11	0,198	0,030
	330	90,90	9,14	0,198	0,030
	1100	91,35	9,18	0,198	0,030
Программа без поиска ненулевых элементов (рисунок 2.8)	68	90,73	8,86	0,200	0,030
	220	90,64	9,11	0,198	0,030
	330	90,90	9,14	0,198	0,030
	1100	91,35	9,18	0,198	0,030
Программа с двумя дополнительными векторами (рисунок 2.11)	68	90,73	8,86	0,200	0,030
	220	90,64	9,11	0,198	0,030
	330	90,90	9,14	0,198	0,030
	1100	91,35	9,18	0,198	0,030
[131]		92,36	8,49	0,198	0,030
[132]		92,24	8,50	0,198	0,030
[133]		91,89	7,02	0,186	0,024

Для набора 1 разница объяснима различными методами решения задачи: в работе [130] используется метод конформных отображений. Для набора 2 в работах [131–133] также используется метод моментов, поэтому

согласованность результатов выше, а разница объяснима разной сегментацией. Наибольшее расхождение наблюдается с результатами из работы [133], в которой использовано трехмерное моделирование и RWG базисные функции. Отличие результатов работ [131, 132] объяснимо различием используемых тестовых функций.

2.5 Основные результаты главы

Рассмотрена эффективность сжатия разреженных матриц за счет использования форматов хранения. Получены аналитические оценки коэффициентов сжатия. Проведен анализ границ эффективности форматов, т.е. такой плотности матрицы, выше которой происходит увеличение числа данных при использовании форматов хранения. Показано существование максимального коэффициента сжатия формата. На основе оценок для программной реализации выбран формат CSR в силу своей эффективности. Разработан алгоритм ILU(0)-разложения с использованием формата CSR. Для ускорения, формат модифицирован (добавлен вектор указателей на главную диагональ матрицы). С использованием формата CSR разработан алгоритм решения системы вида **My** = **z**, где **M** – матрица, хранящаяся в формате CSR.

Проведены вычислительные эксперименты, подтверждающие эффективность выполненных разработок. Приведено сравнение полученных результатов с результатами других авторов. Показано, что использование дополнительного вектора **Diag** ускоряет алгоритм ILU(0) до 1,23 раза (см. таблицу 2.4). Усовершенствования, предложенные в программе, использующей дополнительный вектор **Diag** (рисунок 2.6), позволили ускорить решение СЛАУ итерационным методом BiCGStab до 1,6 раза по сравнению с алгоритмом без использования формата CSR и до 4,4 раза по сравнению с прямым методом Гаусса. Программа с двумя дополнительными векторами (рисунок 2.11) показала дополнительное ускорение до 1,7 относительно программы, использующей дополнительный вектор **Diag** (рисунок 2.6).

3 УСКОРЕНИЕ МНОГОКРАТНОГО РЕШЕНИЯ СЛАУ ИТЕРАЦИОННЫМИ МЕТОДАМИ

3.1 Оценки ускорения по сравнению с прямым методом

В данном разделе представлены результаты использования итерационного метода для многократного решения СЛАУ с изменяющиеся матрицей. Раздел основан на работах [134 –139].

3.1.1 Аналитические оценки

Для многократного решения СЛАУ с изменяющиеся матрицей (см. разд. 1.5) использован итерационный метод с предобусловливателем, полученным из матрицы первой СЛАУ. Алгоритм для m-кратного решения СЛАУ имеет вид:

Алгоритм 3.1 – Многократное решение СЛАУ итерационным методом с предобусловливанием

1 Получить матрицу $\mathbf{A}_S$ из матрицы $\mathbf{A}_1$ с помощью предфильтрации
2 Вычислить матрицу $\mathbf{M}$ из матрицы $\mathbf{A}_S$
3 Для k от 1 до m
4 Итерационно вычислить $\mathbf{x}_i$ из уравнения $\mathbf{MA}_i\mathbf{x}_i = \mathbf{Mb}$ с заданной точностью Tol
5 Увеличить k

В данном алгоритме в качестве способа формирования матрицы предобусловливания использовано ILU(0)-разложение. Оценим возможное ускорение от применения итерационного метода. Если представить его отношением (β) общего времени решения m СЛАУ прямым методом (T_D) ко времени вычисления по алгоритму 3.1, то получим

$$\beta = \frac{m \cdot T_D}{T_{PR} + \sum_{k=1}^{m} T_k}, \tag{3.1}$$

где T_{PR} – время формирования матрицы предобусловливания; T_k – время итерационного вычисления $\mathbf{x}_k$ с заданной точностью. Предположим, что при решении отдельных СЛАУ T_k существенно не изменяется и, в среднем, равно T_A. Тогда из (3.1) получим усредненное ускорение

$$\beta_A = \frac{m \cdot T_D}{T_{PR} + m \cdot T_A} \tag{3.2}$$

и оценку максимального усредненного ускорения

$$\beta_A^{\max} = \lim_{m \to \infty} \left(\frac{m \cdot T_D}{T_{PR} + m \cdot T_A} \right) = \frac{T_D}{T_A}. \tag{3.3}$$

Формулы (3.1) и (3.3) имеют важные следствия.

Следствие 1. *Чем больше число решаемых СЛАУ, тем меньше ускорение зависит от времени построения предобусловливателя, а значит, от выбора вида предобусловливания, способа предфильтрации и оптимального значения допуска обнуления.*

Следствие 2. *Поскольку максимальное усредненное ускорение обратно пропорционально среднему времени итерационного процесса, то актуально уменьшение времени одной итерации и числа итераций.*

При использовании предобусловливания время итерационного процесса зависит от значения допуска обнуления τ, чем оно меньше, тем, вообще говоря, итерационный процесс быстрее. Это происходит за счет формирования более плотной матрицы **M**, что, как правило, ведет к уменьшению числа итераций [46]. Оно, очевидно, минимально при $\tau = 0$. Это следствие делает неэффективным использование форматов хранения разреженных матриц, т.к. отсутствует экономия памяти (см. гл. 2.1).

Еще одним параметром, влияющим на время итерационного процесса, является начальное приближение $\mathbf{x}^0$. Ясно что чем $\mathbf{x}^0$ ближе к решению системы, тем меньше итераций потребуется итерационному методу для его уточнения. Исходя из этого следует предположить, что для вектора начального приближения эффективно использовать вектор решения предыдущей СЛАУ. Для подтверждения этого далее сравнивались два варианта выбора $\mathbf{x}^0$: фиксированное начальное приближение (все элементы вектора равны некоторому значению, в данном случае, единице), используемое для решения каждой СЛАУ; вектор решения предыдущей СЛАУ.

Полезно оценить изменение элементов текущей матрицы СЛАУ относительно первой. Так как, в нашем случае, изменяющиеся элементы находятся на главной диагонали и равны между собой (см. раздел 1.2), рассматривались значения одного элемента матрицы. Зависимость изменения значений (Δa) от m приведена на рисунке 3.1. Матрицы получены для структуры, изображенной на рис. 2.7. Видно, что изменение значений элементов СЛАУ с увеличением m становится меньше и не превышает 12%. Таким образом, для данной задачи представляется более выгодным использовать вектор решения предыдущей СЛАУ.

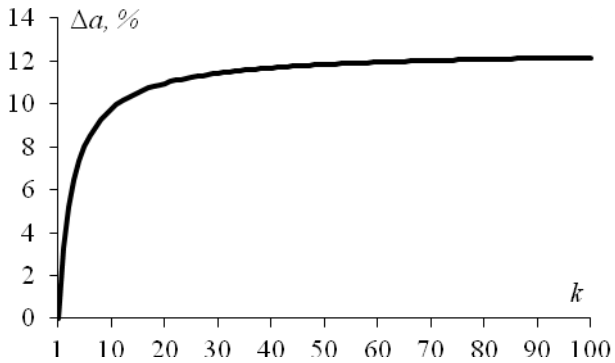


Рисунок 3.1 – Изменение значений элементов матрицы при m-кратном решении СЛАУ

Часто необходимо моделирование при изменении размеров структуры, что приводит к изменению элементов матрицы, расположенных в произвольных местах [18]. В данном случае алгоритм, использующий блочное LU-разложение, неприменим. Поэтому была рассмотрена возможность многократного решения СЛАУ итерационным методом (алгоритм 3.1).

Для ускорения итерационного процесса рассматривались два способа ускорения. Первый – это использование в качестве вектора начального приближения $\mathbf{x}^0$ вектора решения предыдущей СЛАУ, т.е. $\mathbf{x}^0_k = \mathbf{x}_{k-1}$, $k = 1, 2, \ldots m$ (для первой СЛАУ использовался единичный вектор). Второй способ – это использование матрицы предобусловливания $\mathbf{M}$, полученной при решении первой СЛАУ, т.е. $\mathbf{M}_k = \mathbf{M}_0$. Результаты приведены в следующем разделе.

3.1.2 Вычислительные оценки

Для вычислительного эксперимента использовался персональный компьютер (при вычислениях распараллеливание не использовалось, т.е. работало одно ядро процессора) со следующими параметрами: платформа – AMD FX(tm)-8320 Eight-Core Processor; частота процессора – 3,50 ГГц; объем ОЗУ – 16 Гбайт; число ядер – 8; операционная система – Windows 7x64.

В качестве исследуемой использована простая структура (рисунок 2.12), полученная в системе TALGAT. Целью моделирования являлась оценка временных затрат, необходимых для вычисления m емкостных матриц. В эксперименте использовалась одна структура, но посредством изменения сегментации границ проводников и диэлектрика были получены СЛАУ различных порядков (N = 708, 1416, 3540, 4425). В качестве итерационного метода использовался метод BiCGStab, который применялся в работах [118, 140] и хорошо зарекомендовал себя. Предобусловливатель **M** был получен с помощью LU-разложения исходной матрицы. Используемые матрицы были проанализированы, результаты анализа сведены в таблицу 3.1 (число обусловленности рассчитано по (1.16)). На рисунке 3.2 приведены портреты матриц, которые получены следующим образом: все элементы матрицы сортируются по модулю значений, затем разбиваются на четыре группы, каждой группе соответствует свой цвет. Как видно, портреты имеют схожий характер, поскольку матрицы были получены увеличением сегментации.

Таблица 3.1 – Характеристики матриц для структуры, представленной на рисунке 2.12

N	N_c	Плотность матрицы **A**, %	Плотность матрицы **M**,%	Число обусловленности
708	288	100	100	$1{,}75\cdot10^{8}$
1416	576	100	100	$5{,}78\cdot10^{8}$
3540	1440	100	100	$2{,}19\cdot10^{9}$
4425	1800	100	100	$3{,}12\cdot10^{9}$

При вычислениях использовался алгоритм 3.1 с двумя вариантами выбора начального приближения: фиксированное и равное вектору решения

предыдущей СЛАУ. Итерационный процесс продолжался, пока относительная норма вектора невязки была больше 10^{-6}.

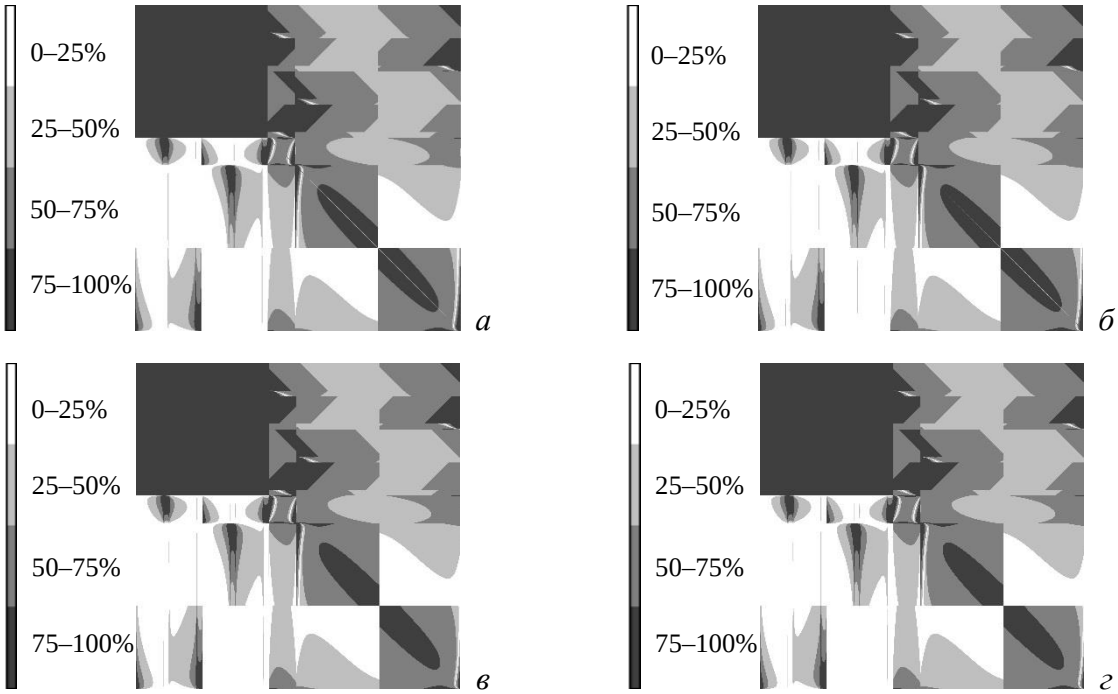


Рисунок 3.2 – Портреты матриц при разных N: 708 (*а*); 1416 (*б*); 3540 (*в*); 4425 (*г*)

Выбор критерия остановки 10^{-6} для данной задачи специально не исследовался. Однако примеры решения одиночных СЛАУ, возникающих в электродинамических задачах, как правило, гораздо хуже обусловленных, показывают, что относительная точность 10^{-6} по норме невязок обеспечивает 7–8 точных знаков после запятой в решении [128]. Как показали многочисленные эксперименты (на каждой из исследуемых матриц с учетом их изменения и обоих выборов начального приближения), максимальное абсолютное значение элементов разности векторов решения, полученных методом Гаусса и итерационным методом, не превышает 10^{-4}. Между тем, для практики, как правило, приемлема точность 1–2 знака [141]. Число итераций при многократном решении приведено на рисунке 3.3.

Как видно, использование решения предыдущей СЛАУ уменьшает число итераций. Далее проведены вычисления с целью оценки ускорения решения итерационным методом относительно метода Гаусса. Результаты для решения m СЛАУ (с разными N) с помощью алгоритма 3.1 сведены в таблице 3.2. Они показывают ускорение при $m>1$. С ростом m ускорение рас-

тет, достигая 49 при m = 1000 и N = 4425. В последней строке таблицы 1 приведены значения, полученные по (3.3). Видно, что к ним сходятся значения предыдущих строк, что подтверждает возможность аналитических оценок ускорений по (3.3) без вычислительного эксперимента. Также видно, что начальное приближение, равное вектору решения предыдущей СЛАУ, дает большее ускорение, чем фиксированное (в 1,4 раза при N = 4425).

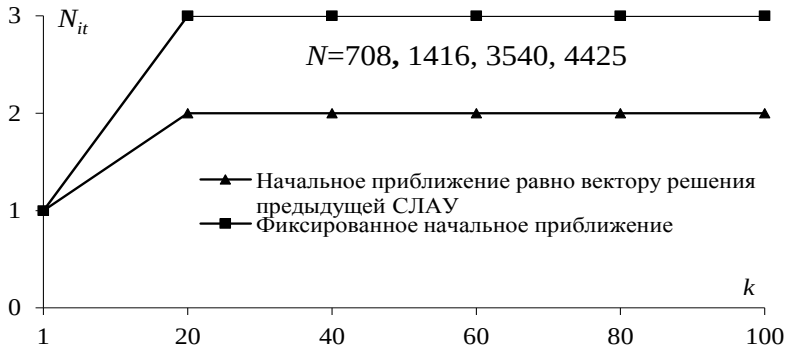


Рисунок 3.3 – Зависимости числа требуемых итераций на решение одной СЛАУ от m при разных вариантах начального приближения (совпадают для всех N)

Таблица 3.2 – Ускорение (относительно метода Гаусса) решения итерационным методом m СЛАУ

m	Фиксированное начальное приближение				Начальное приближение равно вектору решения предыдущей СЛАУ			
	N = 708	N = 1416	N = 3540	N = 4425	N = 708	N = 1416	N = 3540	N = 4425
1	0,36	0,51	0,52	0,31	0,36	0,51	0,52	0,31
5	1,37	2,15	2,43	1,52	1,53	2,27	2,49	1,53
10	2,12	3,65	4,53	2,93	2,52	4,00	4,74	2,98
100	4,12	9,58	20,53	18,09	5,70	12,74	24,82	20,42
1000	4,55	11,43	31,71	37,36	6,59	16,32	43,09	49,3
∞	4,61	11,69	33,78	42,43	6,71	16,86	46,96	58,59

Далее исследовалась структура из одного проводника на диэлектрической подложке над идеально проводящей плоскостью (рисунок 3.4). Целью эксперимента являлась оценка временных затрат для вычисления 100 емкостных матриц, полученных путем изменения одного из размеров структуры: высоты диэлектрика h (в диапазоне 12–112 мкм или на 933%); ширины проводника w (в диапазоне 18–118 мкм или на 656%); высоты проводника t (в диапазоне 6–106 мкм или на 1767%). Количество сегментов на каждом отрезке структуры не менялось, что обеспечивало постоянство порядка (1600)

матриц СЛАУ, необходимое для корректного сравнения. Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-8}. Для сравнения использовался метод исключения Гаусса.

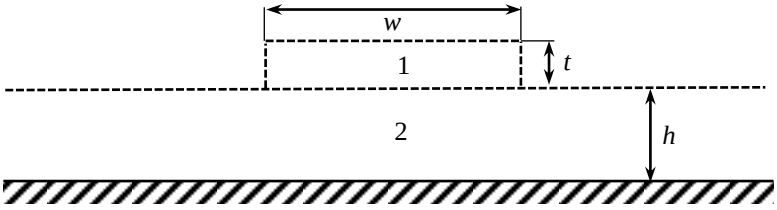


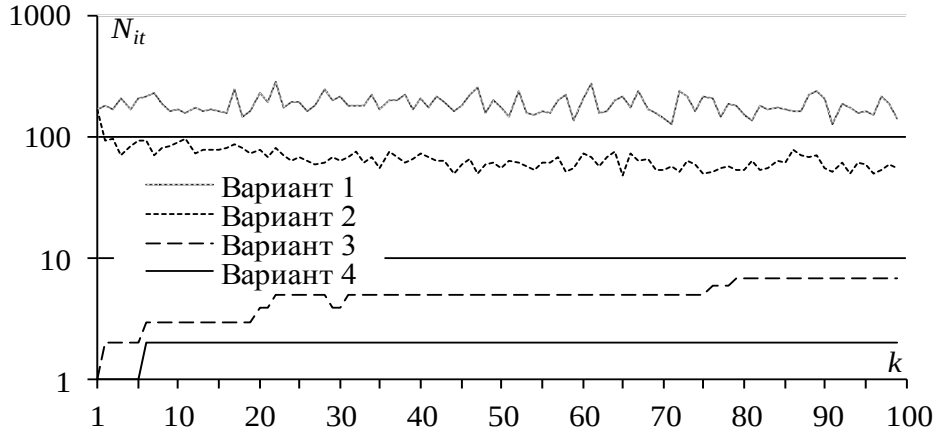
Рисунок 3.4 – Вид поперечного сечения исследуемой структуры в системе TALGAT (проводник 1 на диэлектрической подложке 2 над идеально проводящей плоскостью)

Для подтверждения эффективности способов ускорения сравнены следующие варианты вычислений. В исходном варианте 1 ускорение не использовано. В вариантах 2, 3 способы ускорений использованы отдельно, а в варианте 4 – совместно. На рисунке 3.5 приведено изменение количества итераций (N_{it}) с ростом количества решаемых СЛАУ (m) для каждого из вариантов.

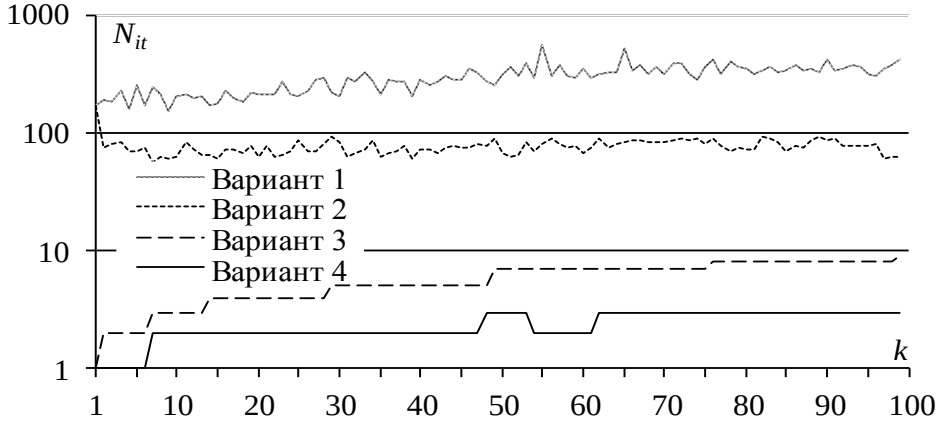
Анализ полученных данных позволяет сделать несколько выводов. В варианте 1 количество итераций велико и в среднем постоянно, поскольку не используются способы ускорения решения. В варианте 2, использующем в качестве начального приближения вектор решения предыдущей СЛАУ, наблюдается небольшое и постепенное уменьшение количества итераций и, следовательно, времени решения СЛАУ. В варианте 3 используется матрица предобусловливания, сформированная из матрицы первой СЛАУ. При решении первой СЛАУ количество итераций равно 1, а при решении последующих СЛАУ оно растет из-за изменений в матрице. В варианте 4 количество итераций минимально, что, соответственно, приводит к снижению общего времени решения всех СЛАУ и доказывает эффективность совместного использования способов ускорения. Также следует отметить, что при изменении разных размеров число итераций для решения СЛАУ разное. При изменении h и w количество итераций меньше, чем при изменении t. Это может быть связано с гораздо большими изменениями в матрице СЛАУ. Для оценки изменений в матрицах СЛАУ использовались нормы матриц измене-

ний $||\Delta\mathbf{A}_k||_1$ и $||\Delta\mathbf{A}_k||_\infty$, где $\Delta\mathbf{A}_k$ – матрица изменений ($\Delta\mathbf{A}_k = \mathbf{A}_k - \mathbf{A}_1$). Расчет норм выполнялся по формулам:

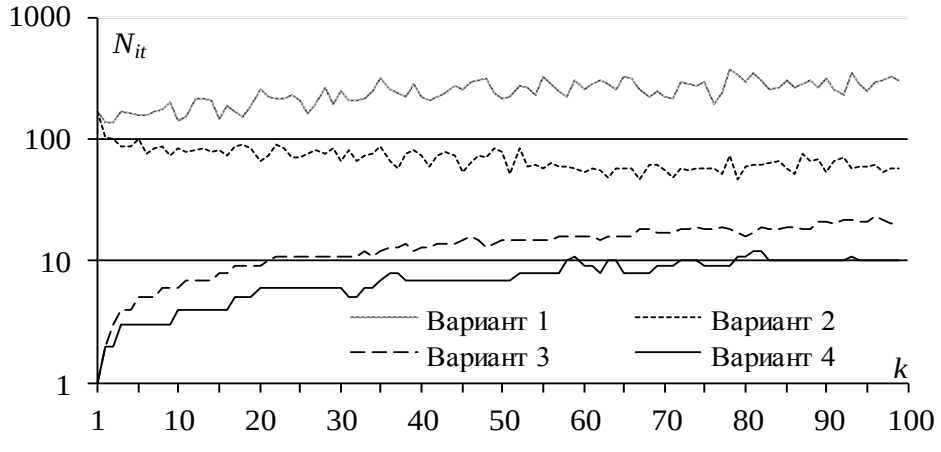
$$||\Delta\mathbf{A}_k||_1 = \max_i \sum_j |a_{ij}|, \quad ||\Delta\mathbf{A}_k||_\infty = \max_j \sum_i |a_{ij}|.$$



а



б



в

Рисунок 3.5 – Число итераций при решении *m*-й СЛАУ методом BiCGStab в зависимости от *m* для вариантов 1–4 при изменении: *а* – *h*; *б* – *w*; *в* – *t*

Результаты вычислений отношения норм матриц $||\Delta\mathbf{A}_m||_1/||\Delta\mathbf{A}_1||_1$, $||\Delta\mathbf{A}_m||_\infty/||\Delta\mathbf{A}_1||_\infty$ приведены на рисунке 3.6. Видно, что для t изменения больше, чем для h и w.

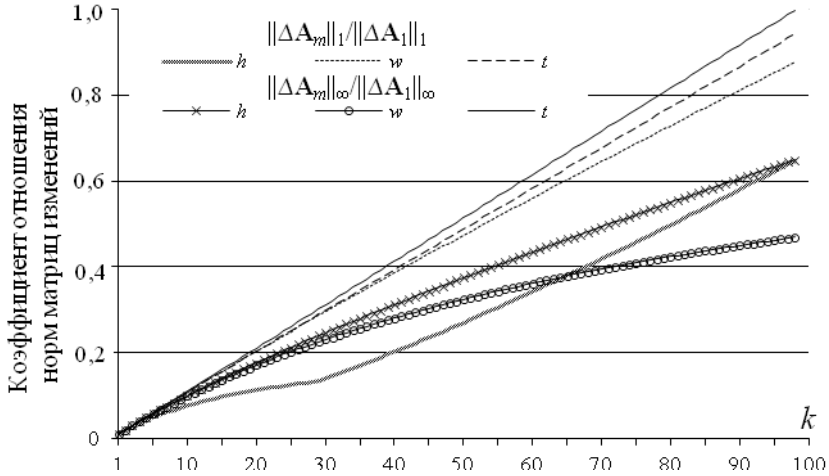


Рисунок 3.6 – Зависимости отношений норм матриц
от количества СЛАУ, полученных при изменении h, w, t

Полученная матрица была проанализирована, результаты анализа сведены в таблицу 3.3. На рисунке 3.7 приведен портрет матрицы при h = 12 мкм, w = 18 мкм, t = 6 мкм.

Таблица 3.3 – Характеристики матрицы для структуры, представленной на рисунке 3.4

N	N_c	Плотность матрицы **A**, %	Плотность матрицы **M**,%	Число обусловленности
1600	1200	99,23	100	$4{,}49\cdot10^3$

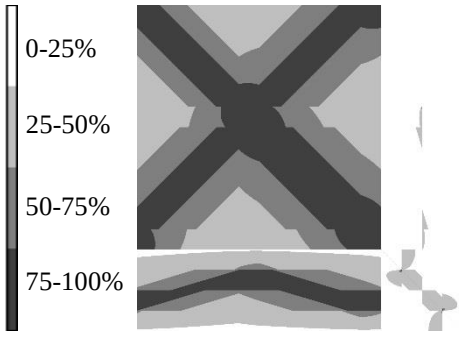


Рисунок 3.7 – Портрет матрицы при N = 1600 (h = 12 мкм, w = 18 мкм, t = 6 мкм)

Портреты матрицы $\Delta\mathbf{A}_k$ (k = 100), при изменении h, w, t, показаны на рисунке 3.8. Как видно, изменение каждого размера имеет свою специфику.

Рисунок 3.8 – Портрет матрицы $\Delta\mathbf{A}_k$ (k = 100) при изменении размеров: *а* – h; *б* – w; *в* – t

Отношение времени для решения k-й СЛАУ методом Гаусса (T_{GE}) к времени её решения методом BiCGStab ($T_{BiCGStab}$) в зависимости от k для вариантов 1–4 приведено на рисунке 3.9.

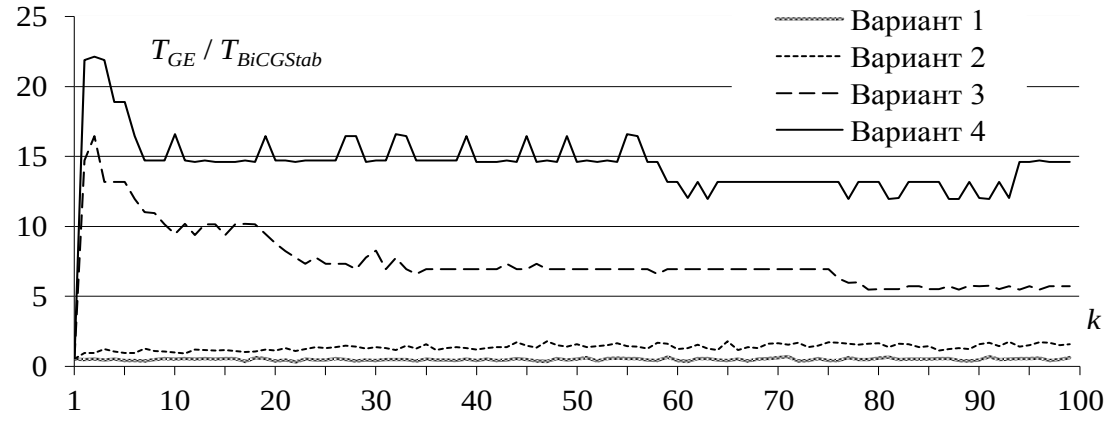
В проведенных тестах вариант 4 показал максимальное ускорение. При небольших изменениях (в пределах 100%) размеров можно получить довольно большое ускорение (10–30 раз). Снижение ускорения к концу многократного решения СЛАУ связано с ростом количества итераций (см. рисунок 3.5) из-за значительного изменения (до 1767%) размеров.

Отношения общего времени решения методом Гаусса к времени решения итерационным методом для 100 СЛАУ приведены в таблице 3.4.

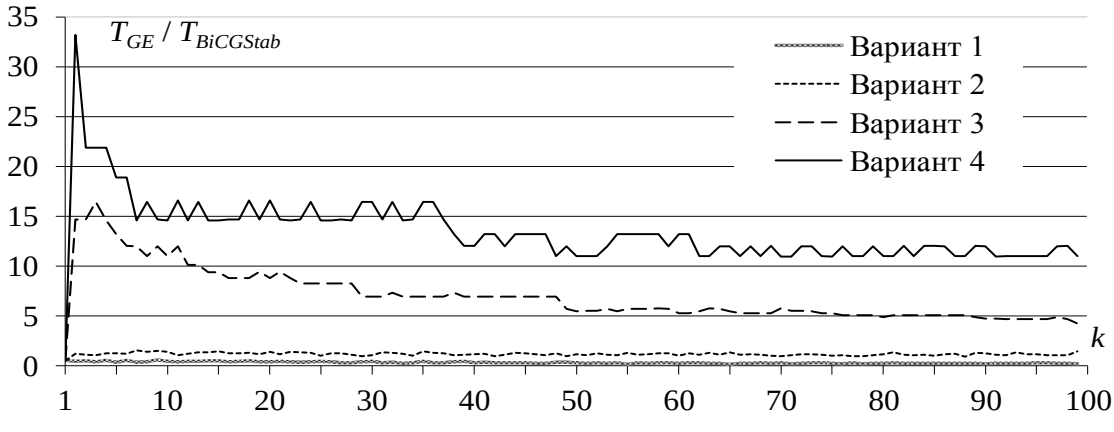
Таблица 3.4 – Ускорение решения 100 СЛАУ

Изменяемый параметр	Вариант 1	Вариант 2	Вариант 3	Вариант 4
h	0,48	1,32	6,49	11,77
w	0,31	1,15	5,87	10,98
t	0,37	1,28	2,87	4,92

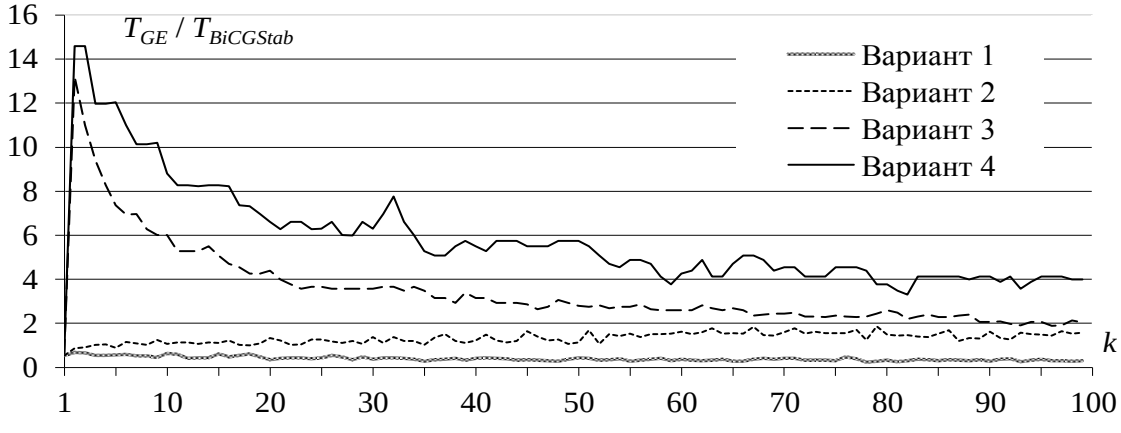
Из таблицы 3.4 видно, что лучшие результаты показал вариант 4 с совместным использованием двух способов ускорения. Вариант 1 показал замедление решения, поскольку в нем не использованы способы ускорения. Для варианта 4 при изменении t ускорение меньше, чем при изменении h и w. Это объясняется ростом количества итераций из-за накопления изменений значений элементов в матрице. При этом применяемые способы ускорения недостаточно эффективны.



а



б



в

Рисунок 3.9 – Отношение времени решения *k*-й СЛАУ методом Гаусса к времени её решения методом BiCGStab в зависимости от *k* для вариантов 1–4 при изменении: *а* – *h*; *б* – *w*; *в* – *t*

3.2 Выбор критерия переформирования предобусловливателя

В данном разделе рассмотрены условия переформирования предобусловливателя, позволяющие ускорить многократное решение СЛАУ с изменяющейся матрицей итерационным методом. Раздел основан на работах [142–144].

В разделе 3.1 показано, что эффективность многократного решения СЛАУ итерационным методом снижается, если увеличивается разница между первой и текущей матрицами СЛАУ. Это ведет к росту числа итераций и замедляет решение итерационным методом. В разд. 1.5 упоминалось о неэффективности корректировки предобусловливателя при неявном предобусловливании, поэтому такой подход далее не исследовался. Вместо корректировки, было предложено переформировывать матрицу **M** для получения ускорения, когда при решении текущей СЛАУ не обеспечивается быстрая сходимость метода.

3.2.1 Увеличение числа итераций выше заданного порога

Для формулировки условия переформирования матрицы предобусловливания можно воспользоваться различными критериями. Первоначально предложено использовать переформирование, когда текущее число итераций N_{it} становится выше заданного порога N_{it}^{MAX}. Далее приведен разработанный алгоритм итерационного решения m СЛАУ с использованием переформирования матрицы **M** по значению порога числа итераций N_{it}^{MAX}.

В данном алгоритме в качестве способа формирования матрицы предобусловливания использовано ILU(0)-разложение. Предфильтрация не использовалась, поскольку при большом числе СЛАУ алгоритм многократного решения эффективен при нулевом допуске обнуления (см. разд. 3.2.1). В начале (строка 1) происходит формирование матрицы предобусловливания **M** из матрицы первой СЛАУ. Если текущее число итераций превышает порог

(строка 4), то происходит переформирование матрицы **M** (строка 5) и дальнейшее ее использование при решении следующей системы.

Алгоритм 3.2 – Многократное решение СЛАУ с переформированием матрицы предобусловливания по порогу числа итераций

1 Вычислить матрицу **M** из матрицы $\mathbf{A}_1$
2 Положить $N_{it} = 0$
3 Для k от 1 до m
4 Если $N_{it} > N_{it}^{MAX}$ и $k > 1$, то
5 вычислить матрицу **M** из матрицы $\mathbf{A}_k$
6 Итерационно вычислить $\mathbf{x}_k$ из уравнения $\mathbf{MA}_k\mathbf{x}_k = \mathbf{Mb}$ с заданной точностью
7 Сохранить число итераций в N_{it}
8 Увеличить k

Рассмотрим влияние порога числа итераций на решение. Если порог слишком низок, то алгоритм будет вызывать много переформирований (чтобы поддерживать низкое число итераций), следовательно, общие затраты времени будут большими. При слишком высоком пороге, переформирований будет мало, но при этом число итераций при решении СЛАУ будет большим, следовательно, общие затраты времени также будут большими. Из приведенных предположений следует возможность существования оптимального порога числа итераций, при котором общие затраты времени минимальны. Для проверки этого предположения и для первого исследования предложенного алгоритма проведен вычислительный эксперимент.

Вычислительный эксперимент

Использовался персональный компьютер, параметры которого указаны в разд. 3.1.2. Исследовалась структура из одного проводника на диэлектрической подложке над идеально проводящей плоскостью (рисунок 3.4) в системе TALGAT. Вычислялись 100 емкостных матриц, полученных путем изменения высоты проводника (t) в диапазоне 6–106 мкм (на 1767%). Число сегментов на границах структуры не менялось, для сохранения порядка (1600) матриц СЛАУ. Анализ матриц приведен в разд. 3.1.2. В качестве итерационного метода использовался метод BiCGStab. Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-8}. В качестве началь-

ного приближения использовалось решение предыдущей системы (для первой использовался единичный вектор).

Были проведены вычисления с различными значениями порога переформирования. На рисунке 3.10 приведены изменения числа итераций при минимальном ($N_{it}^{MAX} = 2$) и максимальном ($N_{it}^{MAX} = 13$) значениях порога.

Из рисунка 3.10 видно, что значение порога сильно влияет на процесс многократного решения. Так, при минимальном значении порога число переформирований велико (12 – по числу минимумов на нижнем графике), а при максимальном оно равно нулю. График для $N_{it}^{MAX} = 2$ показателен частотой переформирований: она максимальна в начале решения, но уменьшается с ростом k, тем самым, отражая способность каждой вновь сформированной матрицы предобусловливания обеспечить решение с изменяющимися матрицами СЛАУ не более чем за 2 итерации. Для более детального анализа в таблице 3.5 приведены характеристики решения для каждого значения N_{it}^{MAX}.

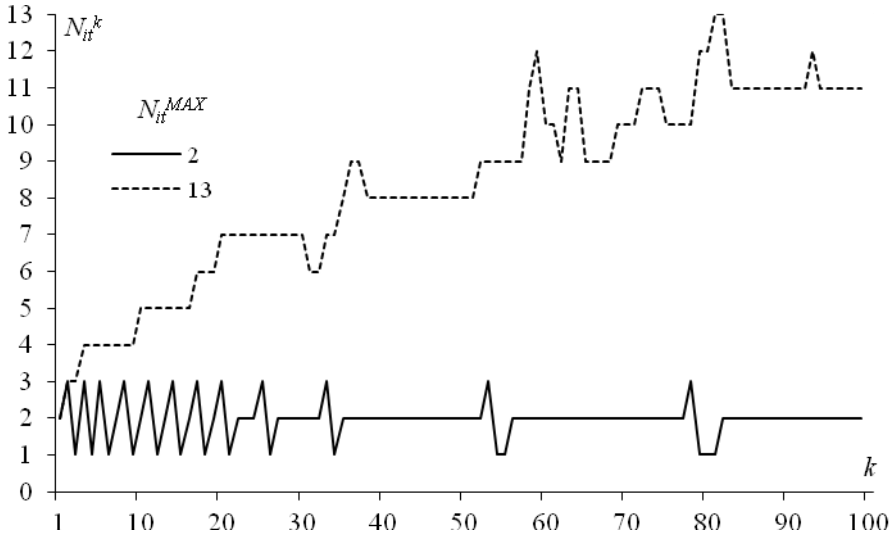


Рисунок 3.10 – Зависимости числа итераций при решении k-й СЛАУ от k при $N_{it}^{MAX} = 2$ и $N_{it}^{MAX} = 13$

Таблица 3.5 – Характеристики многократного решения СЛАУ с переформированием матрицы предобусловливания по порогу числа итераций при различных его значениях

Значение порога переформирования матрицы **M** (N_{it}^{MAX})	Число переформирований матрицы **M**	Сумма итераций решения всех СЛАУ	Время решения всех СЛАУ, мс	Ускорение относительно алгоритма без переформирования матрицы **M**
2	12	196	53673	0,77
3	5	249	30701	1,35
4	2	307	24039	1,72
5	2	415	29504	1,40
6	1	416	26040	1,59
7	1	387	24571	1,68
8	1	389	24978	1,66
9	1	487	28903	1,43
10	1	487	29018	1,43
11	1	496	29303	1,41
12	1	666	37122	1,11
13	0	825	41364	1,00

Проанализируем данные таблицы 3.5. При минимальном значении порога происходит много переформирований, которые увеличивают общее время решения, поскольку время одного переформирования значительно (около 3500 мс). Среднее число требуемых итераций на одно решение очень мало (около 2), поскольку ограничено порогом. Поэтому вклад итераций в общее время решения мал, а основная часть времени расходуется на переформирования. Примечательно, что они избыточны, поскольку увеличивают время решения относительно алгоритма без переформирований (ускорение 0,77 при $N_{it}^{MAX} = 2$). При максимальном значении порога, переформирований нет (ускорение 1,00 при $N_{it}^{MAX} = 13$), и общие затраты времени определяются затратами на итерации. Но эти затраты значительны, несмотря на малое время (около 50 мс) одной итерации, поскольку среднее число итераций увеличилось в 4 раза. Из анализа данных для промежуточных значений порога видно, что существуют его оптимумы: глобальный (4), при котором достигается максимальное ускорение (1,72), и локальный (7) с несколько меньшим ускорением (1,68). Такие ускорения представляются существенными, если учесть, что они получены лишь за счет переформирований матрицы предобусловливания (дополнительно к ускорениям за счет первого предобусловли-

вания, а также использования решения предыдущей СЛАУ в качестве начального приближения для последующей).

Однако у данного подхода, есть недостаток, который не позволяет широко использовать его на практике: определить значение оптимального порога до начала решения не представляется возможным. Поэтому актуален поиск условий переформирования, не обладающих таким недостатком.

3.2.2 Среднее арифметическое время решения

В алгоритмах, используемых далее, в качестве способа формирования матрицы предобусловливания использовано LU-разложение, т.к. предфильтрация не используется (см. разд. 3.1.1). Поскольку матрицы являются плотными, то нет необходимости использовать ILU(0), потому что при таких условиях ILU(0) полностью повторяет LU.

Эффективность итерационного метода при многократном решении СЛАУ определяется суммарным временем решения всех СЛАУ (T_Σ): чем меньше это время, тем эффективней метод [135]. Без переформирования матрицы **M** (аналогично (3.1))

$$T_\Sigma = T_{PR} + \sum_{k=1}^{m} T_k \,, \tag{3.4}$$

где T_{PR} – время формирования матрицы предобусловливания **M** из матрицы первой СЛАУ, T_k – время решения k-й СЛАУ, m – количество СЛАУ.

Используя (3.4), рассмотрим изменение среднего арифметического времени решения k СЛАУ:

$$\overline{T}(k) = \frac{T_\Sigma(k)}{k} = \frac{1}{k}\left(T_{PR} + \sum_{j=1}^{k} T_j \right). \tag{3.5}$$

Для удобства введем дополнительную величину s_k:

$$s_k = kT_{k+1} - \sum_{j=1}^{k} T_j \,. \tag{3.6}$$

Теорема 1. Если s_k монотонно возрастает при $k = 1, 2, \ldots, k_*, \ldots$ и выполнены условия $s_1 < T_{PR}$ и $s_{k_*} < T_{PR} \le s_{k_*+1}$ для некоторого $k_* \ge 1$, для этого k_* достигается единственный минимум функции $\overline{T}(k)$.

Доказательство. Рассмотрим разность $\overline{T}(k+1) - \overline{T}(k)$, тогда

$$\overline{T}(k+1) - \overline{T}(k) = \frac{1}{k+1}\left(T_{PR} + \sum_{j=1}^{k+1} T_j\right) - \frac{1}{k}\left(T_{PR} + \sum_{j=1}^{k} T_j\right) =$$

$$= \frac{1}{k(k+1)}\left(kT_{PR} - (k+1)T_{PR} + k\sum_{j=1}^{k+1} T_j - (k+1)\sum_{j=1}^{k} T_j\right) =$$

$$= \frac{1}{k(k+1)}\left(-T_{PR} + kT_{k+1} - \sum_{j=1}^{k} T_j\right) = \frac{1}{k(k+1)}\left(-T_{PR} + s_k\right).$$

Очевидно, что функция $\overline{T}(k)$ убывает, если $s_k < T_{PR}$, в противном случае ($s_k > T_{PR}$) функция $\overline{T}(k)$ возрастает. Для наличия минимума функции $\overline{T}(k)$ достаточно выполнения условий $s_1 < T_{PR}$ и, для некоторого конечного k_*, $s_{k_*} < T_{PR} \le s_{k_*+1}$. Теорема доказана.

Теорема 2. Если при $k = 1, 2, \ldots$ выполняются условия $T_k = t$ и $t < T_{PR}$, тогда функция $\overline{T}(k)$ убывает и стремится к t.

Доказательство. Пусть $T_k = t$, тогда согласно (3.6) $s_k = 0$, а

$$\overline{T}(k+1) - \overline{T}(k) = \frac{1}{k(k+1)}(-T_{PR}) < 0.$$

Из этого следует, что функция $\overline{T}(k)$ убывает для всех k и согласно (3.5)

$$\lim_{k\to\infty} \overline{T}(k) = \lim_{k\to\infty}\left(\frac{1}{k}(T_{PR} + kt)\right) = t.$$

Теорема доказана.

Условия, указанные в теореме 1, выполняются при многократном решении СЛАУ. Т.е. для монотонного увеличения s_k достаточно увеличения времени решения k-й СЛАУ (T_k), что и наблюдается (см. разд. 3.1.1). Как

правило, решение первой СЛАУ происходит значительно быстрее, чем время формирования матрицы предобусловливателя, т.е. выполняется условие $s_1 < T_{PR}$. Из этого следует, что при многократном решении СЛАУ будет существовать единственный минимум функции $\overline{T}(k)$, а это позволяет использовать его как условие для переформирования предобусловливателя с помощью LU-разложения (т.е. при выполнении условия $\overline{T}(k-1) < \overline{T}(k)$ или $\frac{T_\Sigma(k-1)}{(k-1)} < \frac{T_\Sigma(k)}{k}$). Из теоремы 2 следует, что при постоянном T_k, возможном только при постоянном и низком числе итераций (см. разд. 3.1.1), функция $\overline{T}(k)$ будет постоянно убывать, а значит, переформирование предобусловливателя не будет происходить, что и не требуется при низком числе итераций. С учетом вышеизложенного разработан алгоритм 3.3.

Алгоритм 3.3 – Многократное решение СЛАУ с переформированием матрицы предобусловливания по увеличению среднего арифметического времени решения

1 Вычислить матрицу предобусловливания **M** из матрицы $\mathbf{A}_1$
2 Сохранить время решения в T_{PR}
3 Найти $\mathbf{x}_1$ из уравнения $\mathbf{MA}_1\mathbf{x}_1 = \mathbf{Mb}$ с заданной точностью *Tol*
4 Сохранить время решения в T_1
5 $T_\Sigma = T_{PR} + T_1$
6 Для k от 2 до m
7 Найти $\mathbf{x}_k$ из уравнения $\mathbf{MA}_k\mathbf{x}_k = \mathbf{Mb}$ с заданной точностью *Tol*
8 Сохранить время решения в T_k
9 Если $T_\Sigma/(k–1) < (T_\Sigma + T_k)/k$ и $k \neq m$
10 Вычислить матрицу предобусловливания **M** из матрицы $\mathbf{A}_k$
11 Сохранить время вычисления в T_{PR}
12 $T_\Sigma = T_\Sigma + T_{PR}$
13 $T_\Sigma = T_\Sigma + T_k$
14 Увеличить k

В данном алгоритме в качестве способа формирования матрицы предобусловливания использовано LU-разложение. В строке 7 выполняется решение СЛАУ итерационным методом. Далее проверяется условие для переформирования матрицы предобусловливания (строка 9). Если среднее арифметическое значение времени решения на шаге k ($(T_\Sigma + T_k) / k$) больше,

чем на шаге k–1 ($T_\Sigma / (k–1)$), то происходит переформирование матрицы предобусловливания, которое реализуется в строке 10.

Вычислительный эксперимент

Для подтверждения сделанного предположения выполнен простой эксперимент по вычислению ёмкости проводника на диэлектрической подложке над идеально проводящей плоскостью (рисунок 3.4). Использовался персональный компьютер (без распараллеливания, т.е. работало одно ядро процессора) с параметрами: платформа – Intel(R) Core (TM) i7; частота процессора – 2,80 ГГц; объем ОЗУ – 12 Гбайт; число ядер – 8; операционная система – Windows 8x64. Использовался итерационный метод BiCGStab. Матрицы получены в системе TALGAT. Число сегментов на каждом отрезке структуры не менялось, для постоянства порядка матриц СЛАУ (N = 1600). В качестве способа формирования матрицы предобусловливания использовалось LU-разложение. Для всех СЛАУ вектор **b** – единичный. В качестве начального приближения использовалось решение предыдущей системы, для первой использовался единичный вектор. Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-8}. Зависимости T_k и $\overline{T}(k)$ приведены на рисунке 3.11, из которого видно наличие минимума в поведении среднего арифметического значения (при k = 52), что подтверждает сделанное предположение.

Далее для тестирования алгоритма сформированы 100 матриц, для двух рассмотренных структур. Для структуры 1 (рисунок 3.4), матрицы порядка N = 1600 получены путем изменения высоты проводника (t) в диапазоне 6, 7, … 106 мкм. Для структуры 2 (рисунок 3.12), представляющей собой модальный фильтр [145], матрицы порядка N = 2001 получены путем изменения зазоров (s) в диапазоне 100, 101, … 200 мкм. Далее учащением сегментации границ структуры 1 и 2 получены матрицы порядков 3200 и 3001, соответственно.

Рисунок 3.11 – Зависимости от k времени: решения k-й СЛАУ (T_k) и среднего арифметического ($\overline{T}(k)$)

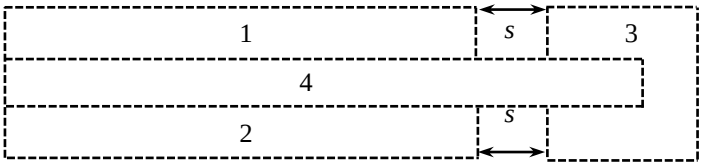


Рисунок 3.12 – Вид поперечного сечения структуры 2 (модальный фильтр): проводники 1, 2 и 3, между которыми помещен диэлектрик 4

Полученные матрицы проанализированы, результаты анализа сведены в таблицу 3.6. На рисунке 3.13 приведены портреты матриц при $k = 1$ (описание принципа построения портрета приведено в разд. 3.1.2).

Таблица 3.6 – Характеристики используемых матриц для структуры 1 (рисунок 3.4) и структуры 2 (рисунок 3.12)

Структура	N	N_c	Плотность матрицы **A**, %	Плотность матрицы **M**,%	Число обусловленности
1	1600	1200	99,23	100	$4{,}49 \cdot 10^3$
	3200	2400	99,22	100	$9{,}07 \cdot 10^3$
2	2001	1700	97,74	100	$2{,}93 \cdot 10^8$
	3001	2550	97,74	100	$6{,}39 \cdot 10^8$

Полученные ускорения решения 100 СЛАУ предложенным алгоритмом 3.3 для структур 1 и 2 по отношению к алгоритму без переформирования предобусловливателя сведены в таблицу 3.7. Также приведены результаты, полученные при переформировании предобусловливателя с помощью задания оптимального порога числа итераций (см. разд. 3.2.1).

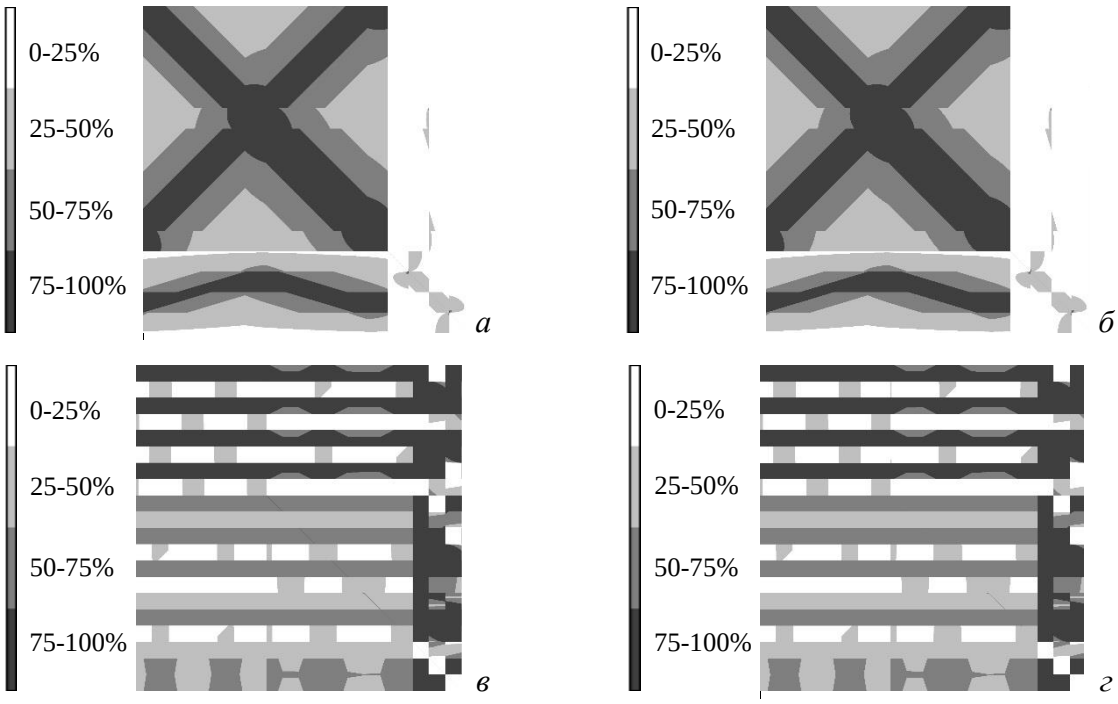


Рисунок 3.13 – Портреты матриц при разных *N*: 1600 (*а*); 3200 (*б*); 2001 (*в*); 3001 (*г*)

Таблица 3.7 – Ускорение решения 100 СЛАУ методом BiCGStab с переформированием предобусловливателя

Алгоритм	Ускорение			
	Структура 1		Структура 2	
	N = 1600	N = 3200	N = 2001	N = 3001
По оптимальному порогу числа итераций (N_{it}^{Opt})	1,51 (8)	1,24 (8)	1,62 (10)	1,58 (10)
По увеличению среднего арифметического времени	1,51	1,15	1,60	1,46

Максимальное ускорение обеспечивает алгоритм с заданием оптимального порога числа итераций. Однако этот алгоритм требует определения значения этого порога путем пошагового поиска (см. разд. 3.2.1). Алгоритм переформирования по увеличению $\overline{T}(k)$ показал ускорения близкие к оптимальным. Отличия составляют не более 7,6% (для структуры 2 при N = 3001).

Для детального анализа полученных результатов на рисунке 3.14 для обеих структур показано, как меняется число итераций при решении k-й СЛАУ с использованием обоих алгоритмов: по увеличению $\overline{T}(k)$ и по оптимальному порогу числа итераций (*Opt*).

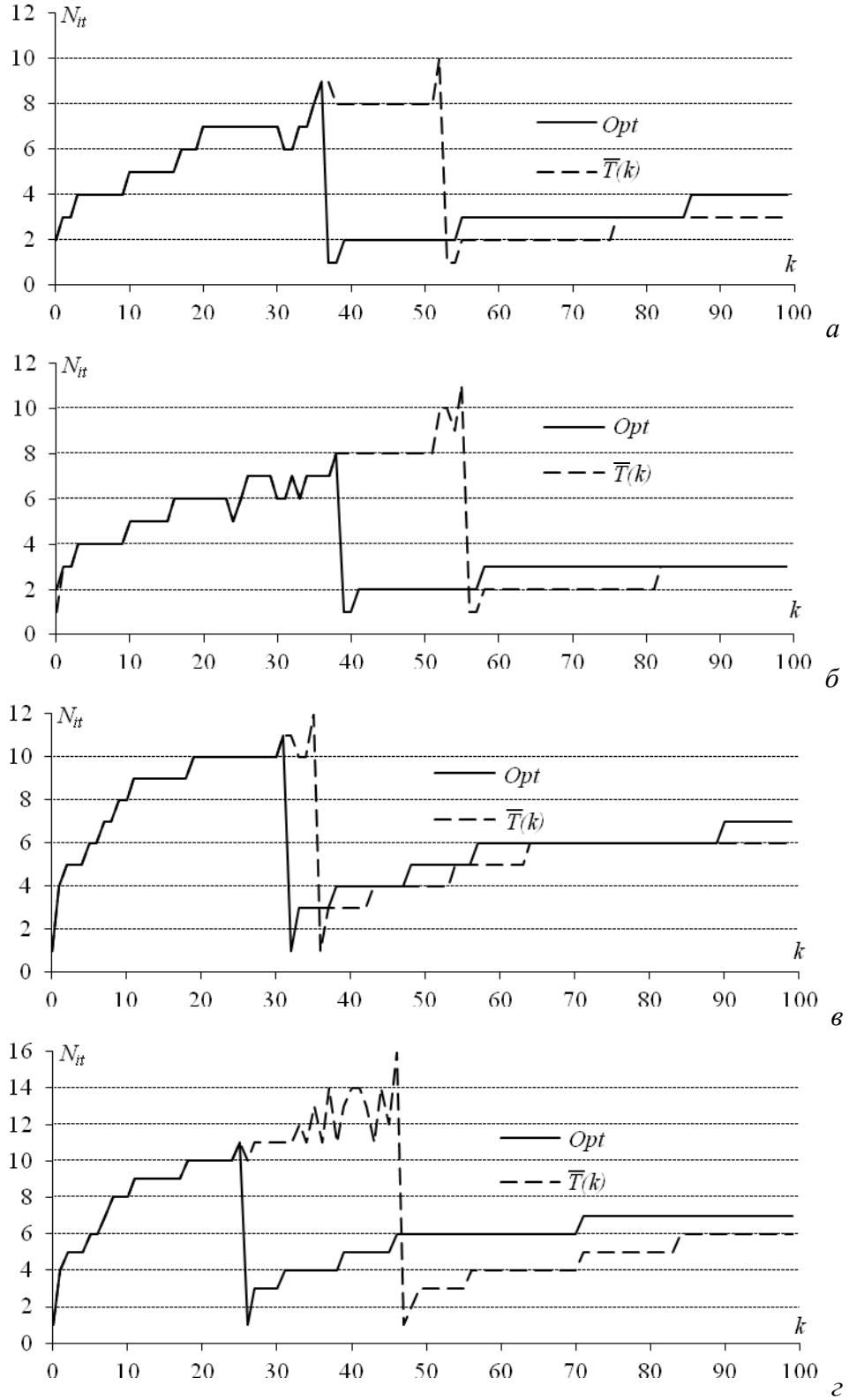


Рисунок 3.14 – Число требуемых итераций, в зависимости от k, при использовании алгоритма с переформированием предобусловливателя по увеличению $\overline{T}(k)$ и по заданному оптимальному порогу числа итераций (*Opt*) для структуры 1 при N = 1600 (*а*), 3200 (*б*) и структуры 2 при N = 2001 (*в*), 3001 (*г*)

Видно, что при более позднем переформировании предобусловливателя ускорение уменьшается. Например, для структуры 1 (рисунок 3.14*б*) алго-

ритм по увеличению $\overline{T}(k)$ (переформирование при $k = 56$) дает ускорение 1,15, а алгоритм по оптимальному порогу – 1,24 (переформирование при $k = 39$). Таким образом, алгоритм по увеличению $\overline{T}(k)$ адаптивно определяет момент переформирования матрицы предобусловливания при увеличении времени итерационного процесса, тем самым, минимизируя время решения всех СЛАУ (в независимости от загрузки центрального процессора рабочей станции посторонними задачами).

3.2.3 Средняя арифметическая сложность решения

Использование среднего арифметического времени решения имеет недостаток, обусловленный трудностью точного измерения времени [146]. Поэтому предложено перейти от измерений времени к расчету сложности алгоритмов [147].

Общую сложность решения (без переформирования предобусловливателя) m СЛАУ (F_Σ) можно выразить через среднюю арифметическую сложность решения одной СЛАУ ($\overline{F}$): $F_\Sigma = m\overline{F}$. Тогда вычислительные затраты многократного решения СЛАУ можно отслеживать, контролируя значение текущей средней арифметической сложности ($\overline{F}_k$) решения k СЛАУ: $\overline{F}_k = F_{\Sigma k} / k$. Для определения сложности предложено использовать арифметическую сложность (Q) [146] и O-нотацию.

Данные подходы могут быть использованы с любым итерационным методом. Для вычислительного эксперимента были выбраны два итерационных метода BiCGStab и CGS [81]. Алгоритмы этих итерационных методов приведены в разд. 1.3.3.3. Для получения матрицы предобусловливателя **M** использовалось LU-разложение, аналогично разд. 3.2.2.

При расчете сложности алгоритма BiCGStab учитывалось наличие двух возможных условий выхода из итерационного процесса (алгоритм 1.2, строки 17 и 24). Полученные оценки сложности для LU-разложения, BiCGStab и CGS приведены в таблице 3.8. Подробный расчет сложности алгоритмов

приведен в Приложении Б. Для верификаций полученных оценок проведен вычислительный эксперимент.

Таблица 3.8 – Оценки сложности алгоритмов

Алгоритм	**Арифметическая сложность (Q)**	**O-нотация**
LU-разложение	$f_{LU}(N) = (10N^3-3N^2+16N-18)/6$	$f_{LU}(N) = N^3/6$
BiCGStab условие 1 (выход в строке 17)	$f(N, N_{it}) = 5N^2+13N+3+N_{it}(10N^2+31N+16)+(N_{it}-1)(10N^2+33N+11)$	$f(N, N_{it}) = 4N^2+6N+4(N_{it}-1)(N^2+2N)$
BiCGStab условие 2 (выход в строке 24)	$f(N, N_{it}) = 5N^2+8N+2+N_{it}(20N^2+64N+37)$	$f(N, N_{it}) = 2N^2+2N+4N_{it}(N^2+2N)$
CGS	$f(N, N_{it}) = 10N^2+16N+1+N_{it}(20N^2+43N+15)+(N_{it}-1)(10N+4)$	$f(N, N_{it}) = 2N^2+N_{it}(8N^2+5N)$

Вычислительный эксперимент

Использовался персональный компьютер, параметры которого указаны в разд. 3.2.2. Матрицы получены в системе TALGAT. Для тестирования сформированы по 100 матриц для каждого рассматриваемого случая. Рассмотрены три структуры. Для структуры 1 (рисунок 3.4) матрицы с $N = 1600$ получены путем изменения высоты проводника (t) в диапазоне 6, 7, … 105 мкм. Для структуры 2 (рисунок 3.12) матрицы с $N = 2001$ получены путем изменения зазоров (s) в диапазоне 100, 101, … 199 мкм. Для структуры 3 (рисунок 3.15), представляющей собой зеркальный модальный фильтр [148], матрицы с $N = 1709$ получены путем изменения зазоров (s) в диапазоне 16,9; 16,8; … 7,1 мкм. Далее учащением сегментации получены матрицы порядков 3200, 3001 и 3109, соответственно. Для всех СЛАУ вектор **b** – единичный.

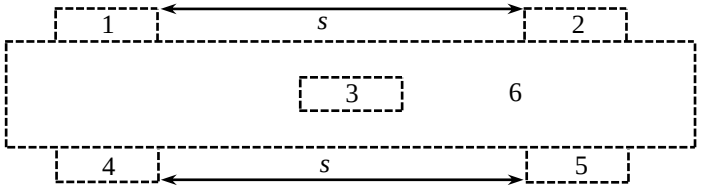


Рисунок 3.15 – Вид поперечного сечения
зеркального модального фильтра
(1–5 –проводники, 6 – диэлектрик)

Характеристики матриц для структур 1 и 2 приведены в таблице 3.6. Для структуры 3 результаты анализа сведены в таблицу 3.9. На рисунке 3.16 приведены портреты матриц структуры 3 (описание принципа построения портрета приведено в разд. 3.1.2).

Таблица 3.9 – Характеристики матриц для структуры 3 (рисунок 3.15)

N	N_c	Плотность матрицы **A**, %	Плотность матрицы **M**,%	Число обусловленности
1709	1220	95,9	100	$2{,}57 \cdot 10^7$
3109	2220	95,9	100	$8{,}29 \cdot 10^7$

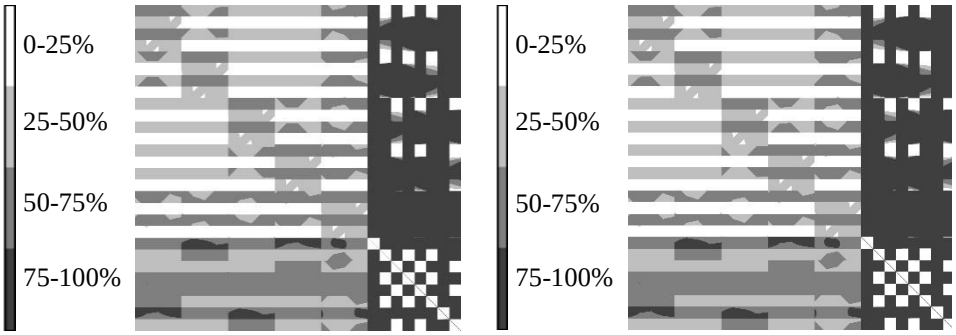


Рисунок 3.16 – Портреты матриц для структуры 3 при разных N: 1709 (*а*); 3109 (*б*)

Время вычисления и оценки сложности алгоритма LU-разложения, нормированные по максимальным значениям, представлены на рисунке 3.17. Видно, что характер графиков совпадает. На рисунке 3.18 приведены относительные отклонения нормированных значений оценок (O, Q) от нормированного измеренного времени (T), полученные по формулам:

$$\Delta O = \left| \frac{T - O}{T} \right| \times 100\%, \tag{3.7}$$

$$\Delta Q = \left| \frac{T - Q}{T} \right| \times 100\%. \tag{3.8}$$

Видно, что отклонения не превышают 9% и уменьшаются с ростом N. Таким образом, оценки сложности LU-разложения получены верно.

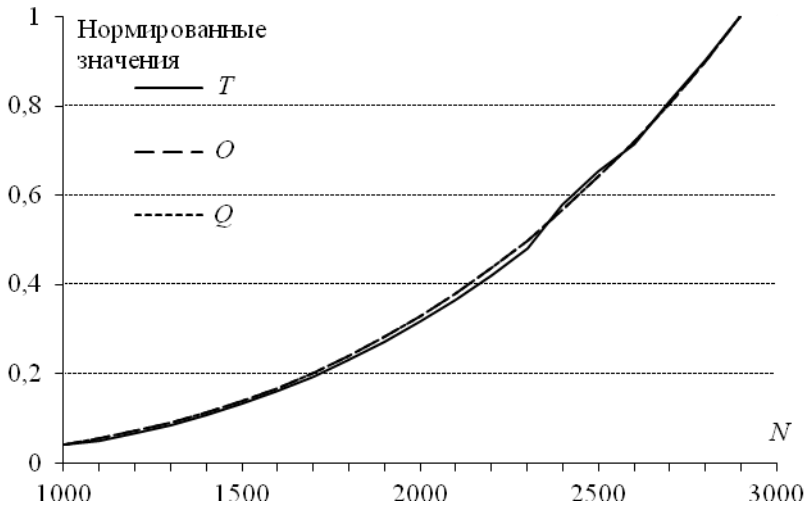


Рисунок 3.17 – Нормированные значения времени вычисления (T)
и оценки сложности с помощью O-нотации (O)
и арифметической сложности (Q) LU-разложения в зависимости от N

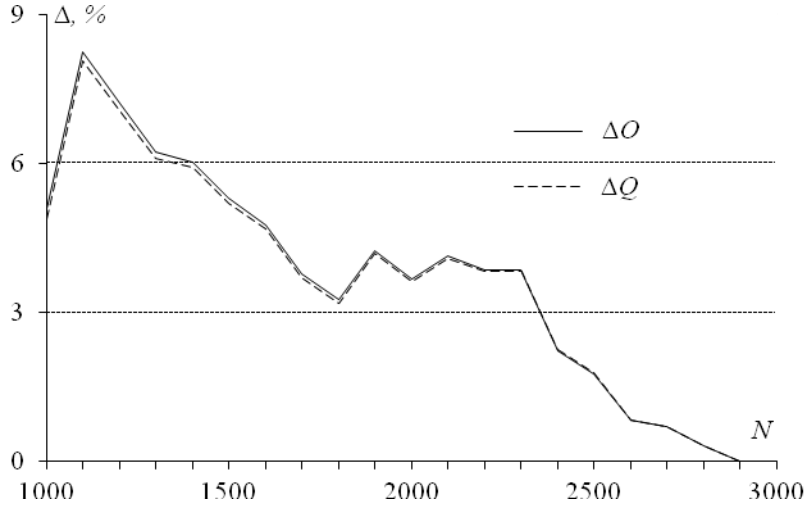


Рисунок 3.18 – Относительные отклонения оценок сложности (ΔO, ΔQ) LU-разложения
в зависимости от N

Для аналогичного сравнения оценок сложности алгоритмов методов BiCGStab и CGS использованы 100 СЛАУ. Взята структура 1 (рисунок 3.4) и матрицы порядка $N = 1600$. Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-8}. Для более корректной оценки искусственно увеличено время решения СЛАУ путем использования предобусловливателя Якоби (см. в разд. 1.3.3.2), в котором учитываются только диагональные элементы матриц. За счет этого увеличилось количество итераций при решении СЛАУ и, как следствие, измеряемое время. Время реше-

ния k-й СЛАУ методом BiCGStab (без переформирования предобусловливателя) и число требуемых итераций представлены на рисунке 3.19. Нормированные (по максимальному значению) время решения и оценки сложности метода BiCGStab приведены на рисунке 3.20.

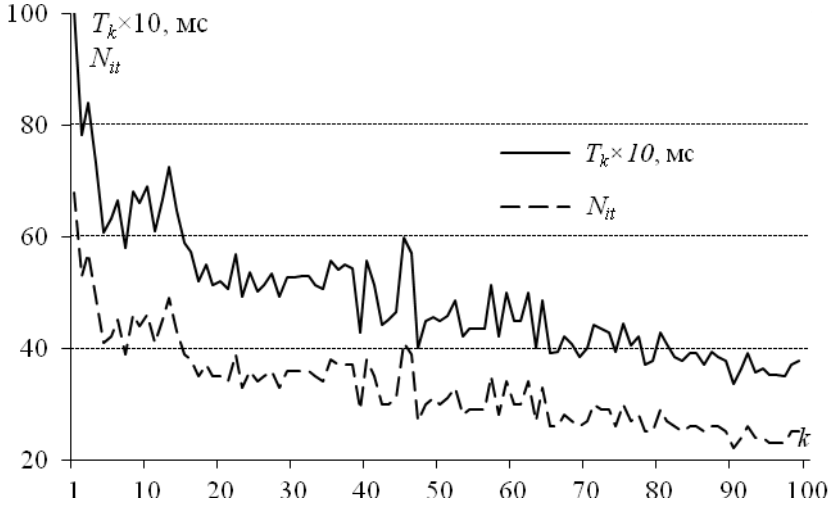


Рисунок 3.19 – Число итераций (N_{it}) и время решения (T_k) k-й СЛАУ методом BiCGStab с предобусловливателем Якоби в зависимости от k

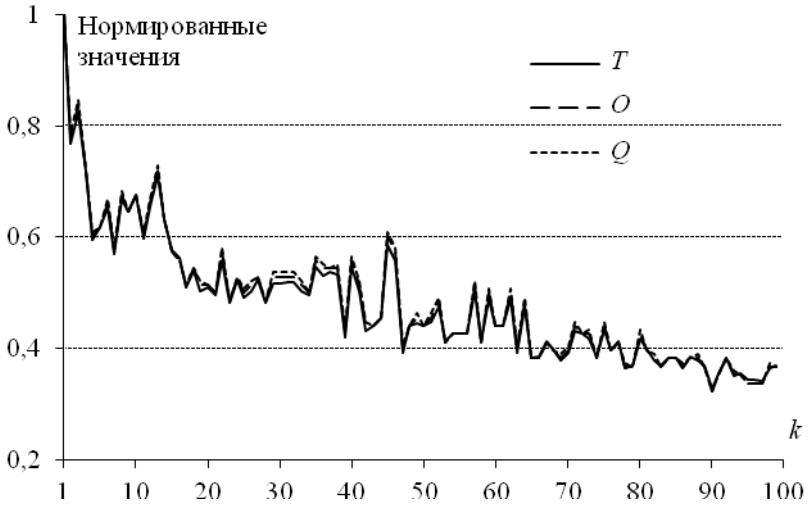


Рисунок 3.20 – Нормированные значения времени решения k-й СЛАУ (T) и нормированные оценки с помощью O-нотации и арифметической сложности Q метода BiCGStab с диагональным предобусловливателем в зависимости от k

Видно, что графики практически совпадают. Для более точного сравнения на рисунке 3.21 приведены отклонения нормированных значений оценок от нормированного измеренного времени (согласно (3.7) и (3.8)). Видно, что отклонения малы (максимальное – около 4 %), что подтверждает

верность оценок. Таким образом, аналитические оценки подтверждаются вычислительным экспериментом.

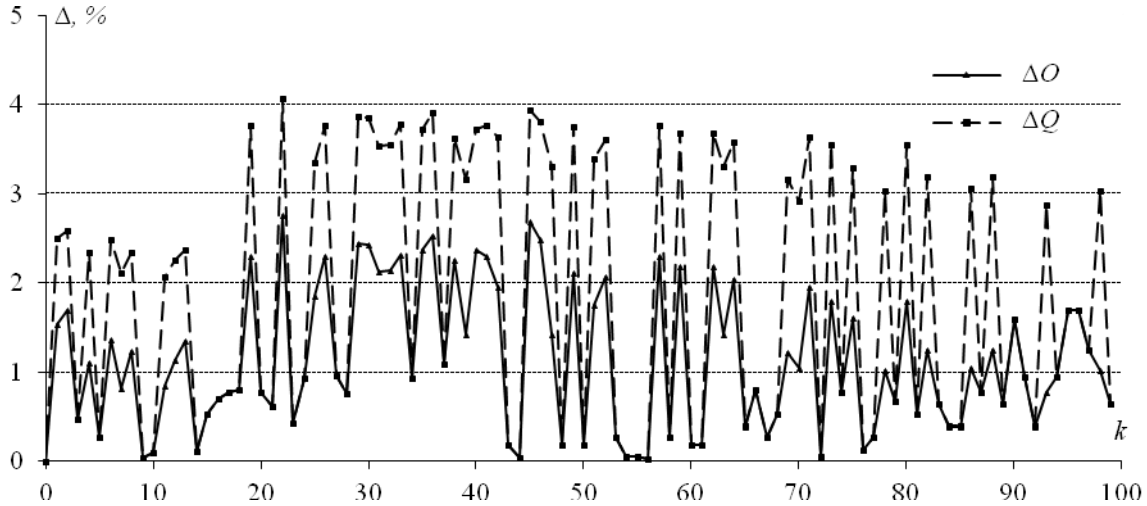


Рисунок 3.21 – Отклонения оценок (ΔO, ΔQ) для алгоритма BiCGStab в зависимости от k

Далее приведен разработанный алгоритм многократного решения СЛАУ с переформированием предобусловливателя, выполняемым на основе оценок из таблицы 3.8.

В данном алгоритме в качестве способа формирования матрицы предобусловливания использовано LU-разложение. В $F_{\sum}$ хранится значение общей сложности k решенных СЛАУ для отслеживания изменения средней сложности, при увеличении которой (строка 11) происходит переформирование матрицы предобусловливания **M** (строка 12). В противном случае решение продолжается без переформирования. В F_k хранится значение сложности итерационного процесса (строки 5 и 10). Для каждого итерационного метода F_k определяется согласно таблицы 3.8.

Далее выполнен вычислительный эксперимент по многократному решению СЛАУ с переформированием предобусловливателя. Полученные ускорения решения 100 СЛАУ для структур 1, 2 и 3 по отношению к алгоритму без переформирования предобусловливателя (для каждого итерационного метода отдельно) сведены в таблицу 3.10. Также приведены результаты, полученные при переформировании предобусловливателя с помощью задания оптимального порога числа итераций (*Opt*).

Алгоритм 3.4 – Многократное решение СЛАУ с переформированием предобусловливателя при увеличении средней сложности решения k СЛАУ

1 Вычислить матрицу **M** из матрицы $\mathbf{A}_1$
2 $F_\Sigma = f_{LU}(N)$
3 Найти $\mathbf{x}_1$ из уравнения $\mathbf{MA}_1\mathbf{x}_1 = \mathbf{Mb}$ с заданной точностью Tol
4 Сохранить количество итераций в N_{it}
5 $F_1 = f(N, N_{it})$
6 $F_\Sigma = F_\Sigma + F_1$
7 Для k от 2 до m
8 Найти $\mathbf{x}_k$ из уравнения $\mathbf{MA}_k\mathbf{x}_k = \mathbf{Mb}$ с заданной точностью Tol
9 Сохранить количество итераций в N_{it}
10 $F_k = f(N, N_{it})$
11 Если $F_\Sigma / (k–1) < (F_\Sigma + F_k) / k$ и $k \neq m$
12 Вычислить матрицу **M** из матрицы $\mathbf{A}_k$
13 $F_\Sigma = F_\Sigma + f_{LU}(N)$
14 $F_\Sigma = F_\Sigma + F_k$
15 Увеличить k

Таблица 3.10 – Ускорение решения 100 СЛАУ методами BiCGStab и CGS с переформированием предобусловливателя

Структура	N	По оптимальному порогу числа итераций (Opt)		По средней арифметической сложности решения (O-нотация)		По средней арифметической сложности решения (Q)	
		BiCGStab	CGS	BiCGStab	CGS	BiCGStab	CGS
1	1600	1,42 (8)	1,34 (7)	1,20	1,52	1,12	1,40
	3200	1,16 (8)	1,35 (7)	0,92	1,27	0,86	1,01
2	2001	1,62 (10)	1,44 (8)	1,60	0,94	1,52	1,05
	3001	1,58 (10)	1,31 (9)	1,55	0,87	1,44	0,95
3	1709	1,12 (9)	1,18 (8)	1,55	1,57	1,55	1,57
	3109	1,03 (10)	1,00(16)	1,59	1,33	1,59	1,33

Максимальное ускорение обеспечивает алгоритм с переформированием предобусловливателя с помощью задания оптимального порога числа итераций (исключение для структуры 1 при $N = 1600$ и использовании метода CGS и для всех тестов структуры 3). Ускорение многократного решения СЛАУ предложенными алгоритмами не постоянно. Например, метод BiCGStab показал отклонения значений ускорения для структуры 1 до 26%, для структуры 2 до 9%, для структуры 3 до 55%. При использовании метода CGS для структуры 1 отклонения составили до 28%, а для структуры 2 и 3 – до 35 и 34%, соответственно. Такие отклонения объясняются тем, что переформирование предобусловливателя происходит в разные моменты. Для детального анализа полученных результатов на рисунке 3.22 показано число итераций

при решении *k*-й СЛАУ с использованием приведенных алгоритмов для структур 1 и 2 при использовании методов BiCGStab и CGS.

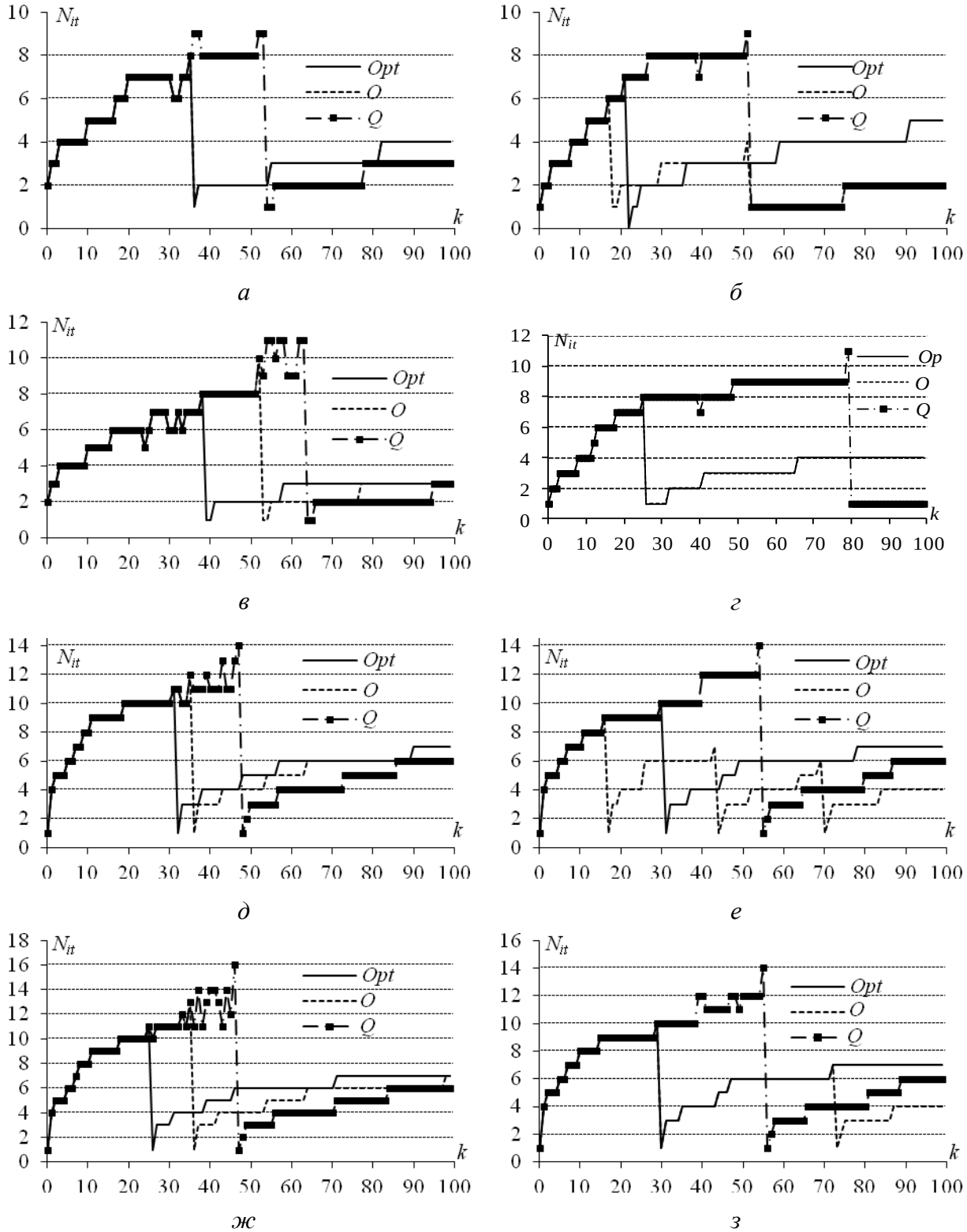


Рисунок 3.22 – Число требуемых итераций алгоритмами: с оценкой сложности алгоритма по *O*-нотации (*O*), по арифметической сложности (*Q*) и использующего оптимальный порог числа итераций (*Opt*) при использовании метода BiCGStab:
для структур 1 при *N* = 1600(*а*), 3200(*в*) и 2 при *N* = 2001(*д*), 3001(*ж*); CGS:
для структур 1 при *N* = 1600(*б*), 3200(*г*) и 2 при *N* = 2001(*е*), 3001(*з*)

Из анализа графиков можно сделать вывод, что при более позднем переформировании предобусловливателя ускорение уменьшается. Например, для структуры 1 (рис. 3.22*в*) при использовании арифметической сложности (Q) (переформирование при k_p = 64) ускорение отсутствует, а при использовании оптимального порога ускорение составляет 1,16 (переформирование при k_p = 39). В результате, можно сделать вывод, что на скорость решения СЛАУ сильное влияние оказывает момент переформирования, т.е. при каком k_p это происходит. При более позднем переформировании предобусловливателя ускорение уменьшается. Так же было, замечено что, чем ближе момент переформирования к m, тем меньшее ускорение будет получено.

3.3 Оптимальная очередность решения СЛАУ

В данном разделе рассмотрен подход к выбору оптимальной очередности решения СЛАУ. Раздел основан на работах [143, 147, 149].

Одним из ресурсов ускорения представляется использование определенной очередности решения СЛАУ. Действительно, эта очередность обычно определяется заданным изменением параметра структуры. Но если общее время решения всех СЛАУ зависит от того, какая именно СЛАУ будет решаться первой, какая – второй и т.д., то существует оптимальная очередность по критерию минимального времени решения. То, что такая зависимость существует, следует из самой сути многократного решения СЛАУ итерационным методом с предобусловливанием. Очевидно, что она определяется двумя факторами: выбором матрицы для вычисления предобусловливателя (то, какая именно из всех матриц будет выбрана, влияет на число итераций, требуемых для решения последующих СЛАУ), а также использованием решения предыдущей СЛАУ в качестве начального приближения текущей (чем ближе начальное приближение окажется к решению, тем меньше потребуется итераций) [135]. При многовариантном анализе используют несколько основных видов изменения параметра: линейное, логарифмическое, с заданными пользователем значениями. При оптимизации изменение может быть случайное, причем в любом направлении. Рассмотрим самое простое, но широко исполь-

зуемое изменение: линейное. При нем изменение очередности решения сводится к тривиальному выбору порядка решения, т.е. по возрастанию параметра (прямой порядок) или его убыванию (обратный порядок).

Отметим, что линейное изменение параметра вовсе не гарантирует монотонного изменения элементов матрицы СЛАУ или её нормы, но может быть частым на практике. В любом случае полезен анализ конкретных структур. Для оценки изменений матриц использованы нормы [24]: $\|\Delta\mathbf{A}_{i,j}\|_1$ и $\|\Delta\mathbf{A}_{i,j}\|_\infty$, где $\Delta\mathbf{A}_{i,j}$ – матрица изменений ($\Delta\mathbf{A}_{i,j}=\mathbf{A}_i-\mathbf{A}_j$), i и j – порядковые номера сравниваемых матриц ($i, j = 1\div m$, m – общее число СЛАУ). Анализируемые матрицы получены в системе TALGAT [17] по математическим моделям на основе метода моментов [18]. Использована структура (рисунок 3.12), представляющая собой симметричный модальный фильтр с лицевой связью [145]. Число сегментов на каждом отрезке структуры не менялось, для постоянства N. Матрицы с N=2001 получены путем изменения зазоров (s) в диапазоне. Изменения выполнялись в прямом (100, 101, … 199 мкм) и обратном (199, 198, …, 100 мкм) порядках.

Зависимости относительных норм матрицы изменений для прямого и обратного порядков от номера решаемой СЛАУ приведены на рисунке 3.23. Как видно, характер изменений для обеих норм одинаков, но для прямого порядка наблюдается монотонное нарастание с уменьшающейся скоростью (более выраженное в начале и менее – в конце диапазона), а для обратного порядка характер зависимостей противоположен. Как упоминалось, существует зависимость эффективности предобусловливателя от изменений матриц. Изменения в обратном порядке практически для всех i меньше чем в прямом. Поэтому при обратном порядке эффективность предобусловливателя увеличивается из-за меньших и более плавных изменений в матрице СЛАУ. Следовательно, для ускорения решения предпочтительно первоначально решать СЛАУ с меньшими изменениями между матрицами, а затем – с большими. Для этого необходимо для каждой матрицы определить близкую к ней матрицу с минимальными изменениями, т.е. найти $\min\limits_j \left\|\Delta A_{i,j}\right\|_1$ или

$\min_j \left\| \Delta A_{i,j} \right\|_\infty$ и далее выполнить сортировку по возрастанию изменений. Однако данный поиск трудоемок, поскольку требует нахождения матрицы изменений m^2 раз. Поэтому такой подход непрактичен.

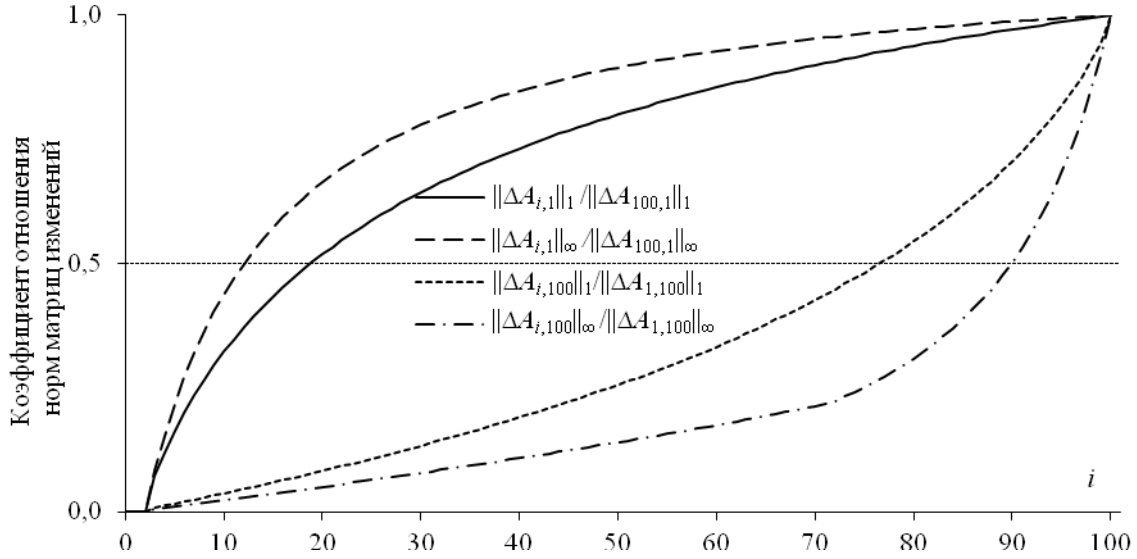


Рисунок 3.23 – Зависимости относительных норм матрицы изменений от номера решаемой СЛАУ для прямого и обратного порядков решения

Однако в нашем случае происходит линейное изменение параметра, что позволяет, меняя только порядок решения (прямой или обратный), выявить оптимальный порядок. Для вычислительного эксперимента по определению оптимального порядка многократного решения СЛАУ разработан общий алгоритм 3.5. В нём в качестве способа формирования матрицы предобусловливания использовано LU-разложение.

Алгоритм 3.5 – Многократное решение *m* СЛАУ с выбором очередности решения

1 Задать очередность решения СЛАУ
2 Вычислить матрицу предобусловливания **M** из матрицы $\mathbf{A}_1$
3 Для *k* от 1 до *m*
4 Найти $\mathbf{x}_k$ из уравнения $\mathbf{MA}_k\mathbf{x}_k = \mathbf{Mb}$ с заданной точностью *Tol*
5 Увеличить *k*

Вычислительный эксперимент

Для вычислительного эксперимента использовался персональный компьютер, параметры которого указаны в разд. 3.2.2. Использовались итерационные методы BiCGStab и CGS. Рассматривались: структура 1 (рисунок 3.4),

структура 2 (рисунок 3.12) и структура 3 (рисунок 3.15). Характеристики используемых матриц для структур 1 и 2 приведены в таблице 3.6, а для структуры 3 – в таблице 3.9. В таблицу 3.11 сведены полученные ускорения при использовании обратного порядка решения СЛАУ относительно прямого.

Таблица 3.11 – Ускорение решения 100 СЛАУ методами BiCGStab и CGS с использованием обратного порядка решения СЛАУ относительно прямого

Структура	N	BiCGStab	CGS
1	1600	1,76	1,73
	3200	1,63	1,66
2	2001	1,71	1,58
	3001	1,84	1,53
3	1709	1,82	1,59
	3109	1,83	1,32

Видно, что для всех структур наблюдается ускорение при использовании обратного порядка решения СЛАУ, причем оно получено для всех N и обоих итерационных методов. Ускорение достигается разницей в числе требуемых итераций при решении СЛАУ в прямом (N^{+}_{it}) и обратном (N^{-}_{it}) порядках, выражаемой в площади фигуры, ограниченной графиками числа итераций (для метода BiCGStab рисунок 3.24, для CGS – рисунок 3.25). Для всех структур при решении СЛАУ в прямом порядке число итераций больше чем в обратном. Основная причина этого заключается в разных матрицах СЛАУ, из которой получен предобусловливатель: в прямом порядке предобусловливатель получен из первой СЛАУ, а в обратном – из 100-й. На число требуемых итераций повлияла и различная степень изменений матрицы СЛАУ в начале (сильная) и конце (слабая) диапазона. Полученные результаты подтверждают предположение о влиянии очередности решения СЛАУ на эффективность решения.

Таким образом, оценено влияние очередности многократного решения СЛАУ итерационным методом на общее время решения всех СЛАУ. Полученные ускорения обратного порядка (до 1,84 раза) относительно вычислений в прямом порядке показывают перспективность контроля очередности решения СЛАУ.

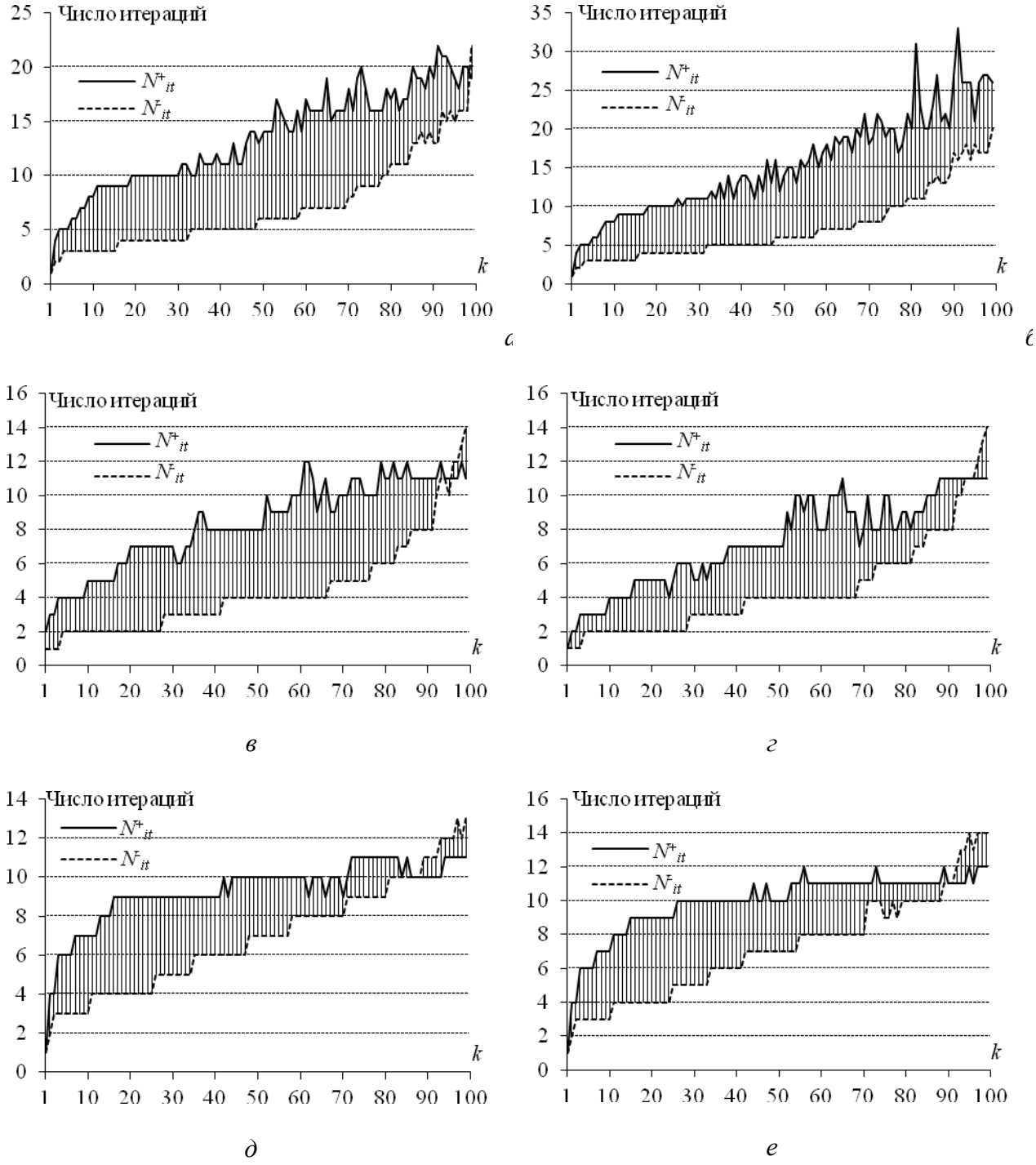


Рисунок 3.24 – Число итераций при многократном решении СЛАУ итерационным методом BiCGStab в прямом (N^+_{it}) и обратном (N^-_{it}) порядках для структур: 1 – при N = 1600 (*а*), 3200 (*б*); 2 – при N = 2001 (*в*), 3001 (*г*); 3 – при N = 1709 (*д*), 3109 (*е*)

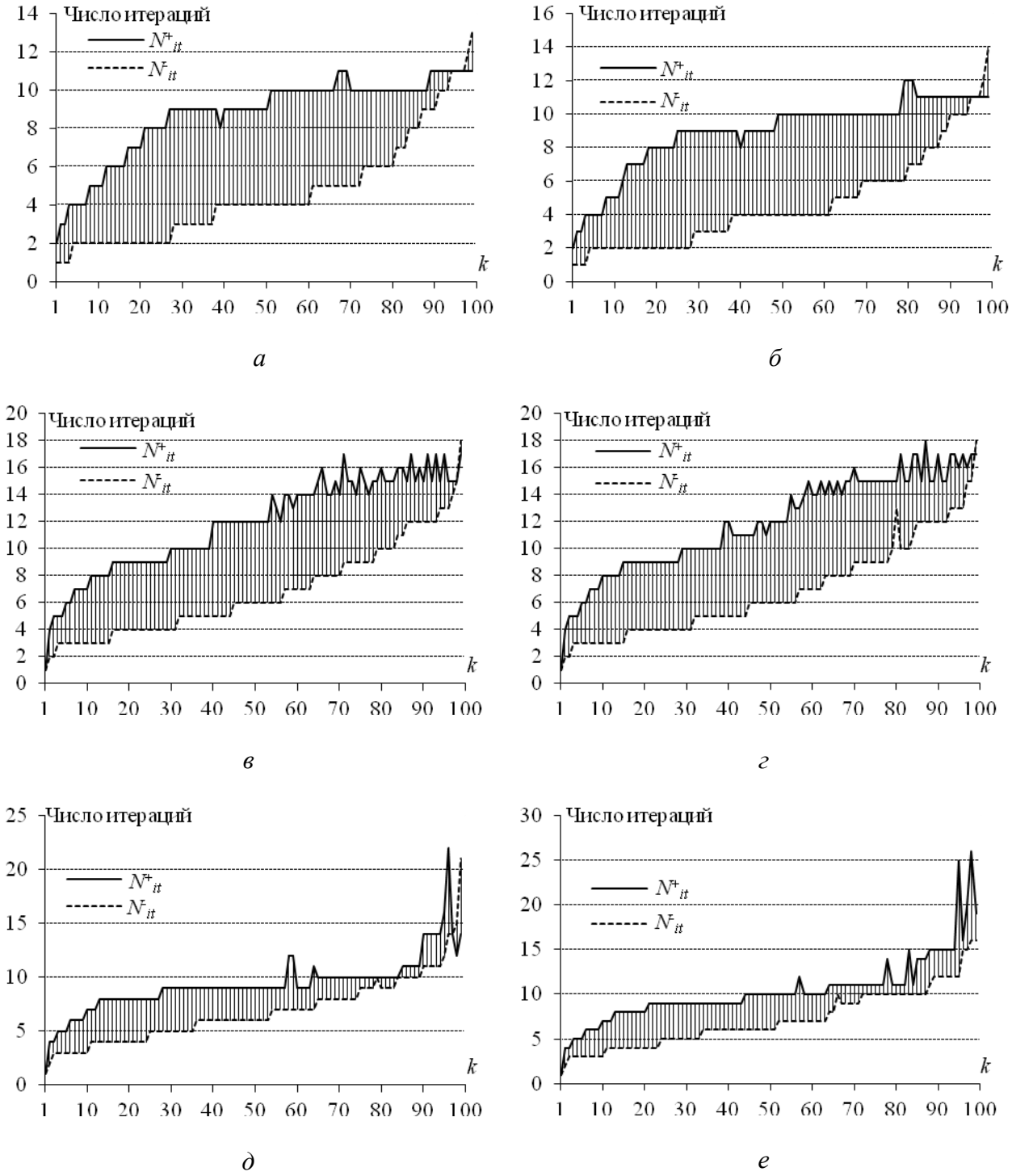


Рисунок 3.25 – Число итераций при многократном решении СЛАУ итерационным методом CGS в прямом (N^{+}_{it}) и обратном (N^{-}_{it}) порядках для структур: 1 – при N = 1600 (*а*), 3200 (*б*); 2 – при N = 2001 (*в*), 3001 (*г*); 3 – при N = 1709 (*д*), 3109 (*е*)

3.4 Выбор предобусловливателя

В данном разделе рассмотрен подход к выбору матрицы для формирования предобусловливателя. Раздел основан на работах [143, 147, 149].

Одним из факторов, влияющим на многократное решение СЛАУ, является выбор предобусловливателя. Очевидно, что данный выбор будет влиять на общее время многократного решения СЛАУ, т.к. предобусловливатель будет меняться. Как правило, на практике значения параметра структуры меняют в заданном диапазоне, что позволяет выбирать значение параметра для формирования предобусловливателя. Самый простой способ – выбрать среднее из крайних значений. На основе предложенного подхода разработан алгоритм 3.6. В нём в качестве способа формирования матрицы предобусловливания использовано LU-разложение.

Алгоритм 3.6 – Многократное решение СЛАУ с выбором матрицы для формирования предобусловливателя

1 Выбрать матрицу (i) для вычисления предобусловливателя, например, $i = m / 2$
2 Вычислить матрицу предобусловливания **M** из матрицы $\mathbf{A}_i$
3 Для k от 1 до m
4 Найти $\mathbf{x}_k$ из уравнения $\mathbf{MA}_k\mathbf{x}_k = \mathbf{Mb}$ с заданной точностью *Tol*
5 Увеличить k

Вычислительный эксперимент

Использовался персональный компьютер, параметры которого указаны в разд. 3.2.2 и итерационный метод BiCGStab. Рассматривались: структура 1 (рисунок 3.4), структура 2 (рисунок 3.12) и структура 3 (рисунок. 3.15). Характеристики используемых матриц приведены в таблицах 3.6 и таблице 3.9.

В качестве первого исследования проведен вычислительный эксперимент с выбором 50-й матрицы для предобусловливателя и порядка решения с 1-й по 100-ю СЛАУ. В таблице 3.12 приведены полученные ускорения относительно алгоритма с выбором 1-й матрицы для предобусловливания.

Таблица 3.12 – Ускорение решения 100 СЛАУ методам BiCGStab с выбором 50-й матрицы для предобусловливателя

Структура	*N*	BiCGStab
1	1600	2,07
	3200	2,21
2	2001	2,14
	3001	1,94
3	1709	1,81
	3109	1,86

Видно, что для всех структур и N наблюдается ускорение (до 2,21 для структуры 2 при $N = 3200$). Оно достигается снижением общего числа итераций при многократном решении СЛАУ. На рисунке 3.26 приведено число итераций при решении k-й СЛАУ для случаев с предобусловливателем, полученным из первой ($\mathbf{M}^1$) и из 50-й ($\mathbf{M}^{50}$) матриц СЛАУ. Видно, что число итераций при формировании предобусловливателя из 50-й матрицы СЛАУ значительно меньше, чем при формировании из 1-й, эта разница и объясняет полученное ускорение. Для оценки взаимного влияния алгоритмов 3.5 и 3.6 выполнен вычислительный эксперимент с использованием метода BiCGStab. В качестве матрицы для формирования предобусловливателя взята 50-я матрица, а для выбора оптимальной очередности использовалось 3 возможных варианта, представленных в таблице 3.13, где стрелками обозначено направление изменения номеров решаемых СЛАУ.

Таблица 3.13 – Варианты очередности решения 100 СЛАУ
с выбором 50-й матрицы для предобусловливателя

Вариант	Очередность решения СЛАУ	
	1–49	51–100
1	←	←
2	→	←
3	←	→

В таблице 3.14 приведены полученные значения ускорения относительно случая с выбором 1-й матрицы для предобусловливания при прямом порядке решения. Видно, что ускорения, при разных вариантах очередности вычисления для каждой структуры почти одинаковы. Максимальное различие 0,02 раза составляют 1 % относительно среднего ускорения, что можно объяснить погрешностью вычислений.

Таблица 3.14 – Ускорение решения 100 СЛАУ методом BiCGStab
с выбором 50-й матрицы для предобусловливания при разной очередности

Структура	N	Вариант очередности решения		
		1	2	3
1	1600	2,16	2,16	2,16
	3200	1,95	1,94	1,96
2	2001	2,05	2,07	2,07
	3001	2,19	2,21	2,20
3	1709	1,81	1,80	1,80
	3109	1,85	1,86	1,86

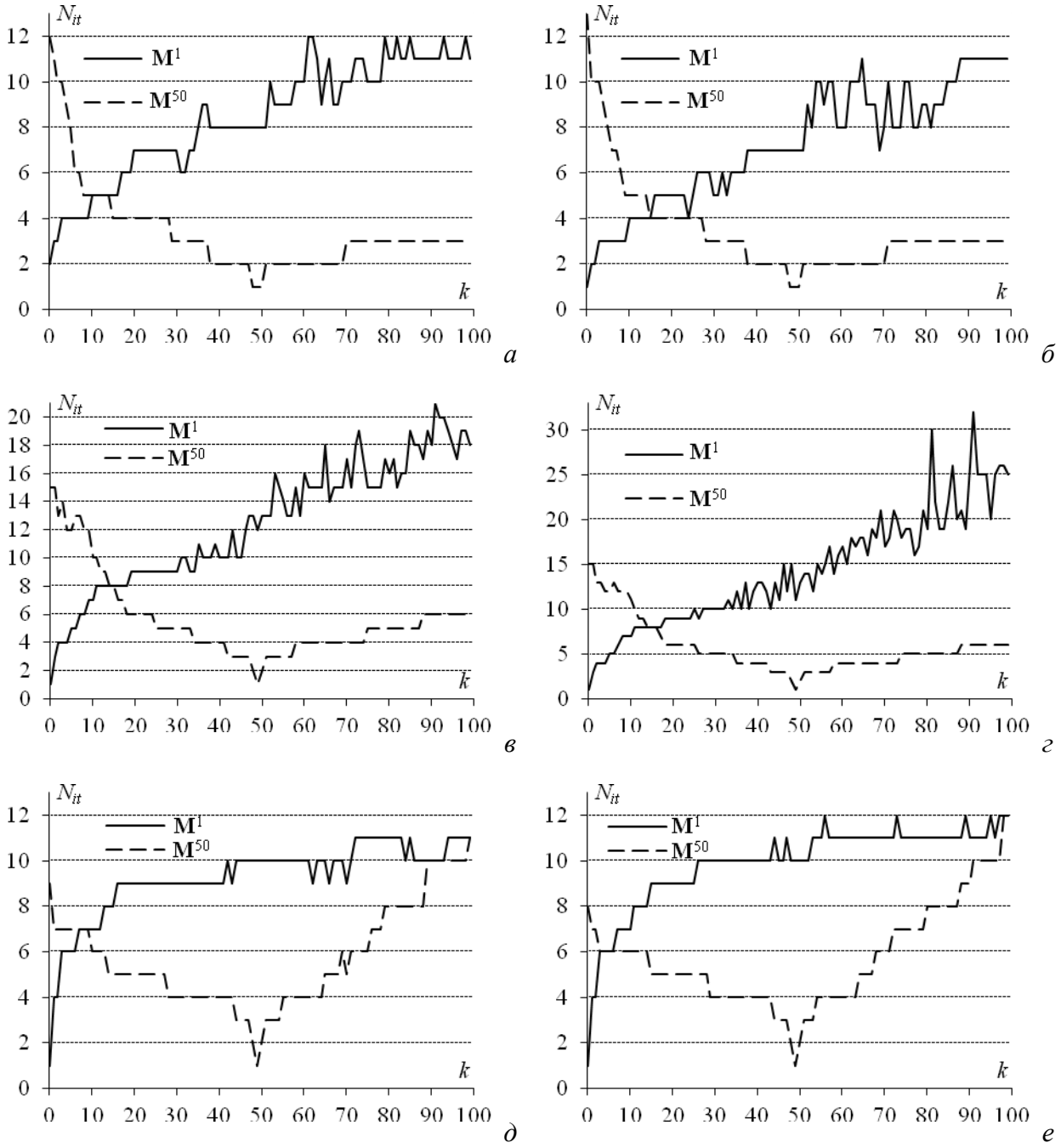


Рисунок 3.26 – Число требуемых итераций, в зависимости от k, при использовании предобусловливателя, полученного из первой ($\mathbf{M}^1$) и из 50-й ($\mathbf{M}^{50}$) матриц СЛАУ для вариантов: 1 – при N = 1600 (*а*), 3200 (*б*); 2 – при N = 2001 (*в*), 3001 (*г*); 3 – при N = 1709 (*д*), 3109 (*е*)

Далее выполнен вычислительный эксперимент по выбору i-й ($\mathbf{M}^i$) матрицы, используемой для предобусловливания, при решении всей последовательности СЛАУ. В качестве первого исследования взята структура 2, характеризующаяся худшей обусловленностью матрицы СЛАУ. На рисунке 3.27 приведены полученные ускорения относительно случая с выбором 1-й матрицы. Видно, что в диапазоне i от 30 до 60 ускорение почти постоянно.

Наибольшие значения ускорения составляют: 2.10 при i=51 и N=2001; 2.21 при i=50 и N=3001, что соответствует середине диапазона (i=50).

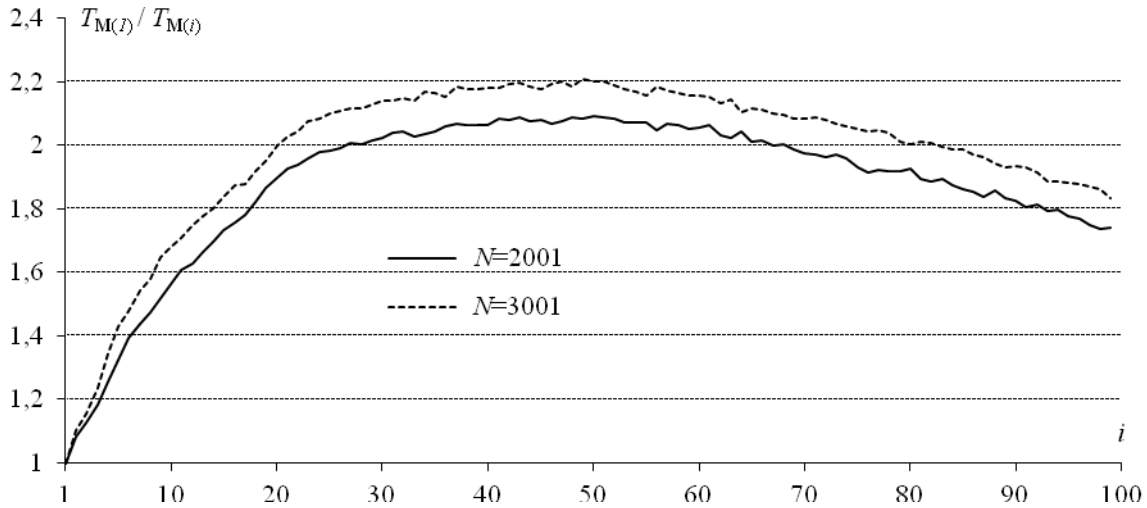


Рисунок 3.27 – Ускорение решения 100 СЛАУ методом BiCGStab с выбором $\mathbf{M}^i$ матрицы для предобусловливания относительно случая с выбором 1-й матрицы для структуры 2

3.5 Математическая модель с дополнительными параметрами

Исходная математическая модель многократного вычисления ёмкостных матриц методом моментов для произвольных двумерных структур проводников и диэлектриков в диапазоне изменения параметра, с использованием итерационного метода с предобусловливателем, полученным из первой СЛАУ, может быть представлена как

$$\mathbf{C}_i = F(x_i),\ i = 1, 2, \ldots, m, \qquad (3.9)$$

где $\mathbf{C}_i$ – ёмкостная матрица, соответствующая значению x_i, x_i – значение изменяющегося параметра, m – число изменений.

Исходя из результатов разделов 3.3 и 3.4, предложена математическая модель с дополнительными параметрами, позволяющая уменьшить время моделирования:

$$\mathbf{C}_i = F(x_i, O, N_M), \qquad (3.10)$$

где O – параметр, задающий порядок (прямой или обратный) вычисления ёмкостных матриц; N_M – номер матрицы СЛАУ для вычисления предобусловливателя $\mathbf{M}$. Добавленные параметры не изменяют итерационный метод и

метод вычисления ёмкостной матрицы, не оказывая существенного влияния на точность вычислений, поскольку она контролируется задаваемым значением *Tol*.

3.6 Основные результаты главы

Рассмотрено использование итерационных методов для многократного решения СЛАУ. Предложены способы ускорения: использование в качестве начального приближения вектора, полученного при решении предыдущей СЛАУ; использование для всех СЛАУ матрицы предобусловливания, полученной при решении первой СЛАУ. На основе предложенных способов предложен алгоритм многократного решения СЛАУ. Предложены условия переформирования предобусловливателя при многократном решении СЛАУ. Рассмотрены следующие условия: по порогу числа итераций, по увеличению среднего арифметического времени решения СЛАУ и средней арифметической сложности решения. Предложен алгоритм многократного решения СЛАУ с выбором очередности решения и выбором матрицы для формирования предобусловливателя.

По всем алгоритмам, разработанным в данной главе, проведены вычислительные эксперименты, подтверждающие их эффективность. Использование итерационного метода в многократном решении СЛАУ позволило при изменениях небольшого числа элементов в матрице ускорить вычисления до 49,3 раза относительно метода Гаусса при использовании задания начального приближения равного вектору решения предыдущей СЛАУ (см. разд. 3.1.2). При изменениях большого числа элементов в матрице, итерационный метод, использующий предложенные приёмы ускорений, показал ускорение до 11,77 раза относительно метода Гаусса.

Переформирование предобусловливателя при увеличении числа итераций выше заданного порога показало ускорение до 1,72 раза относительно способа решения без переформирования. Использование среднего арифметического времени решения позволило получить ускорение до 1,60 раза, а переформирование предобусловливателя с использованием средней

арифметической сложности решения – до 1,57 раза. Использование оптимального порядка решения дало ускорение до 1,85 раза, а использование выбора матрицы для формирования – до 2,21 раза. Показано, что выбор очередности решения, при формировании предобусловливателя из средней матрицы последовательности системы не оказывает существенного влияния на общее ускорение решения. Выявлено, что максимальное ускорение будет при выборе матрицы предобусловливателя в диапазоне от 30 до 60.

На основе полученных результатов предложена математическая модель с дополнительными параметрами (выбором очерёдности решения и матрицы предобусловливания), позволяющая уменьшить время моделирования полосковых структур методом моментов в диапазоне геометрических и электрофизических параметров структуры.

Таким образом, результаты работы показали эффективность использования итерационных методов с предобусловливанием для многократного решения СЛАУ с частично и полностью изменяющимся матрицами.

4 КОМПЛЕКС ПРОГРАММ ДЛЯ ИТЕРАЦИОННОГО РЕШЕНИЯ СЛАУ

Программная реализация алгоритмов решения СЛАУ выполнена с помощью консольных приложений, реализованных в среде Microsoft Visual C++. Исходные коды программ приведены в Приложении А.

Функциональная схема разработанного программного комплекса приведена на рисунке 4.1.

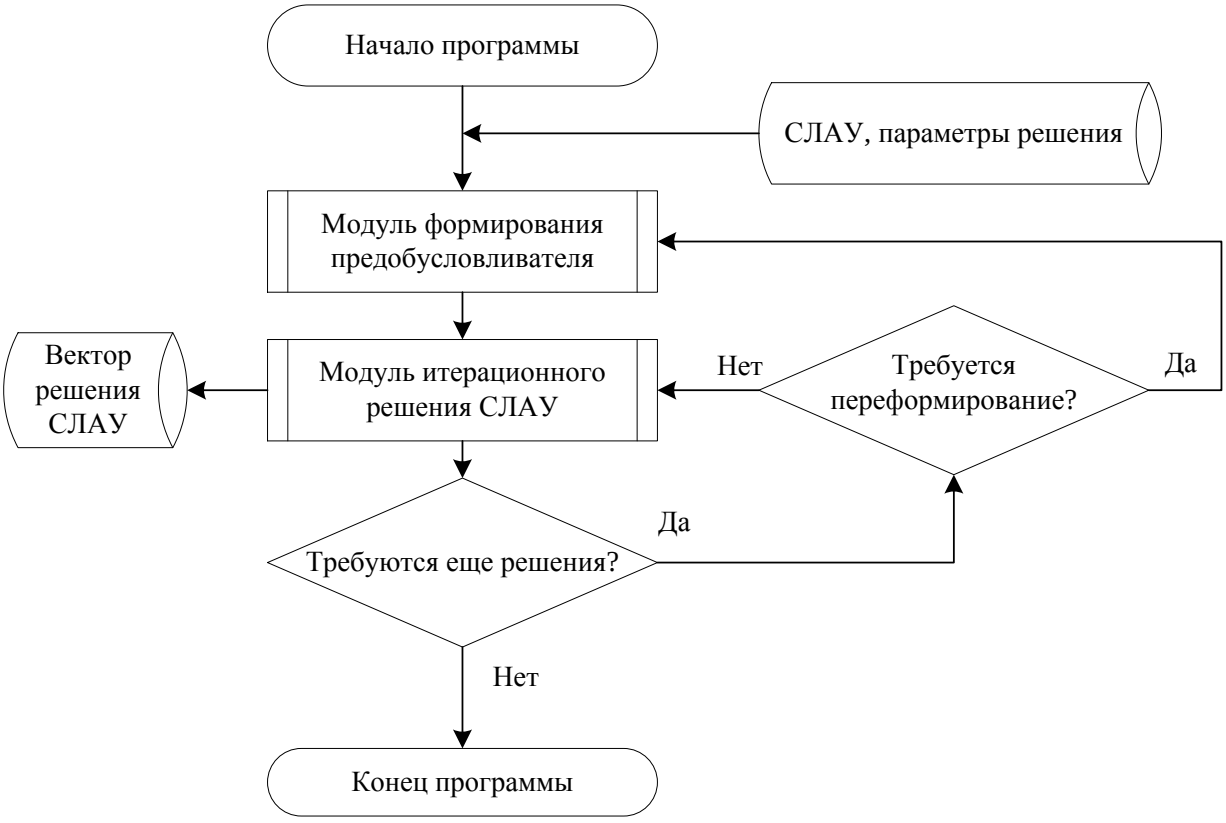


Рисунок 4.1 – Функциональная схема программного комплекса

Входные данные программного комплекса загружаются из внешних *.txt файлов с помощью процедуры LoadMatrix, с параметрами: *i* (тип int) – порядковый номер файла; **A** (тип double) – двухмерный массив, в котором хранится загруженная матрица; *n* (тип int) – порядок матрицы. Затем происходит формирование предобусловливателя из исходной матрицы. В случае многократного решения СЛАУ номер исходной матрицы определяется выбранным алгоритмом. Так при использовании алгоритма с выбором оптимальной очередности номер матрицы может быть первый или последний из последовательности, а при использовании алгоритма с выбором предобусловливателя номер матрицы будет средним из последовательности. Далее

выполняется итерационное решение СЛАУ. Если требуются решения еще, то следующим шагом производится загрузка очередной матрицы. Номер очередной матрицы также может зависеть от выбранного алгоритма. Затем, согласно выбранному алгоритму, определяется, требуется ли переформировать предобусловливатель.

Для реализации модуля формирования предобусловливателя использована программа с двумя дополнительными векторами. Его схема приведена на рисунке 2.11. Алгоритм реализован в виде процедуры со следующими аргументами: *N* (тип int) – порядок матрицы, хранящейся в формате CSR; **aelem_MatrM** – массив ненулевых элементов матрицы (одномерный массив типа double); **jptr_MatrM** – массив указателей для элементов из массива **aelem_MatrM** (тип int); **iptr_MatrM** – массив указателей (тип int) на положение (индексы) в **aelem_MatrM** первых элементов строк матрицы; **UdiagM** – массив указателей (тип int) на диагональные элементы в массиве **aelem_MatrM**. На вход передается исходная матрица, процедура возвращает матрицу после ILU(0)-разложения, которая для экономии оперативной памяти хранится на месте исходной.

Для реализации модуля итерационного решения СЛАУ использован итерационный метод BiCGStab (алгоритм 1.2) или CGS (алгоритм 1.3), выбор итерационного метода определяется пользователем до начала работы алгоритма. Итерационные методы реализованы в виде процедур со следующими аргументами: **A** (тип double) – двухмерный массив, в котором хранится исходная матрица; **ALU** (тип double) – двухмерный массив, хранящий исходную матрицу после ILU(0)-разложения; *N* (тип int) – порядок матрицы; *Tol* (тип double) – требуемая точность вычислений; *Nitmax* (тип int) – максимальное число итераций. Возвращаемыми данными являются: *NumbIter* (тип double) – число итераций, требуемое для решения; **x** – (одномерный массив типа double) – вектор решения.

Алгоритм, который используется в многократном решении СЛАУ, определяется пользователем. Далее приведены схемы алгоритмов много-

кратного решения СЛАУ: переформирование по порогу числа итераций (рисунок 4.2); переформирование при увеличении среднего арифметического времени решения (рисунок 4.3); переформирование при увеличении средней сложности решения (рисунок 4.4); выбор очередности решения (рисунок 4.5); выбор матрицы для формирования предобусловливателя (рисунок 4.6).

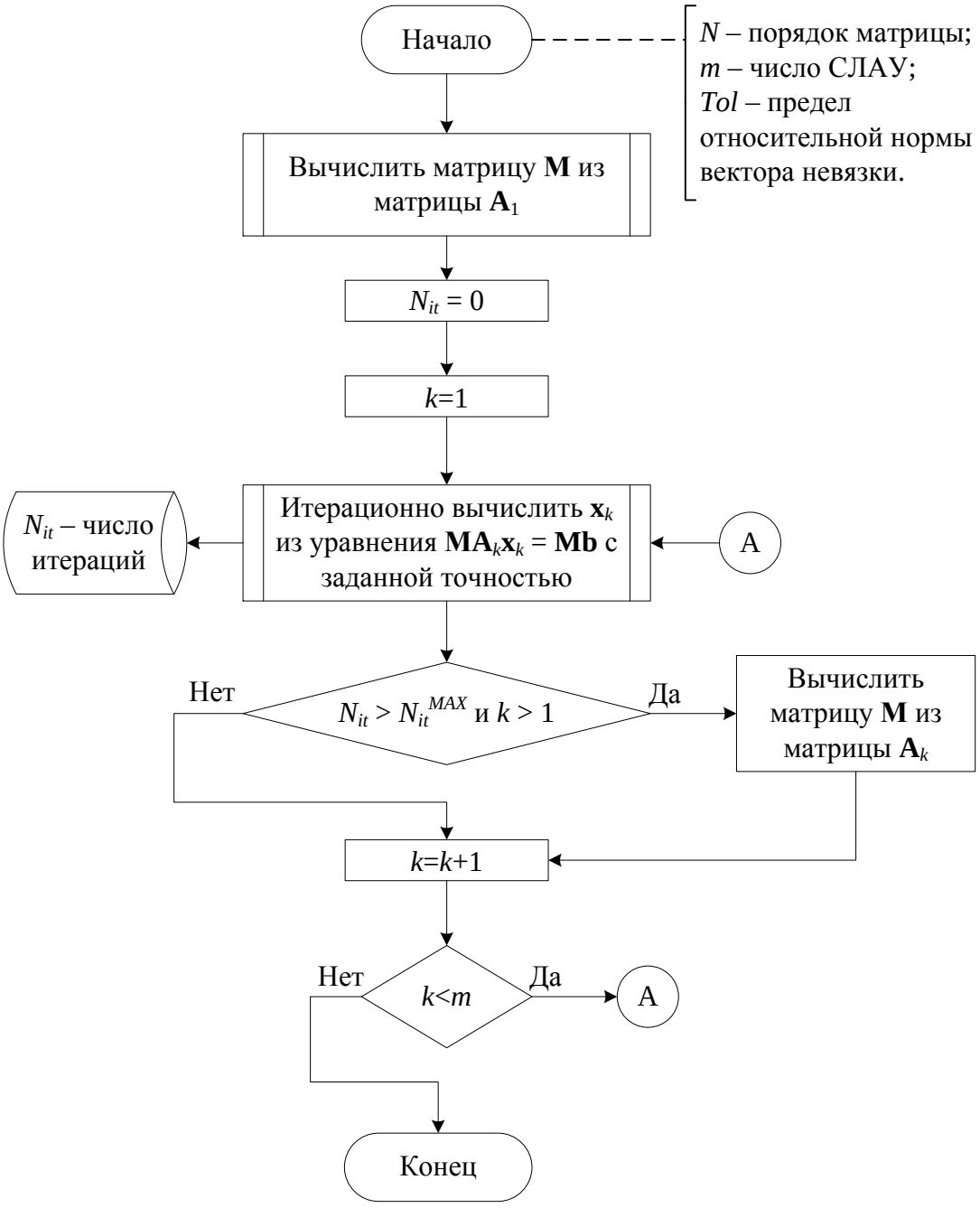


Рисунок 4.2 – Многократное решение СЛАУ
с переформированием матрицы предобусловливания по порогу числа итераций

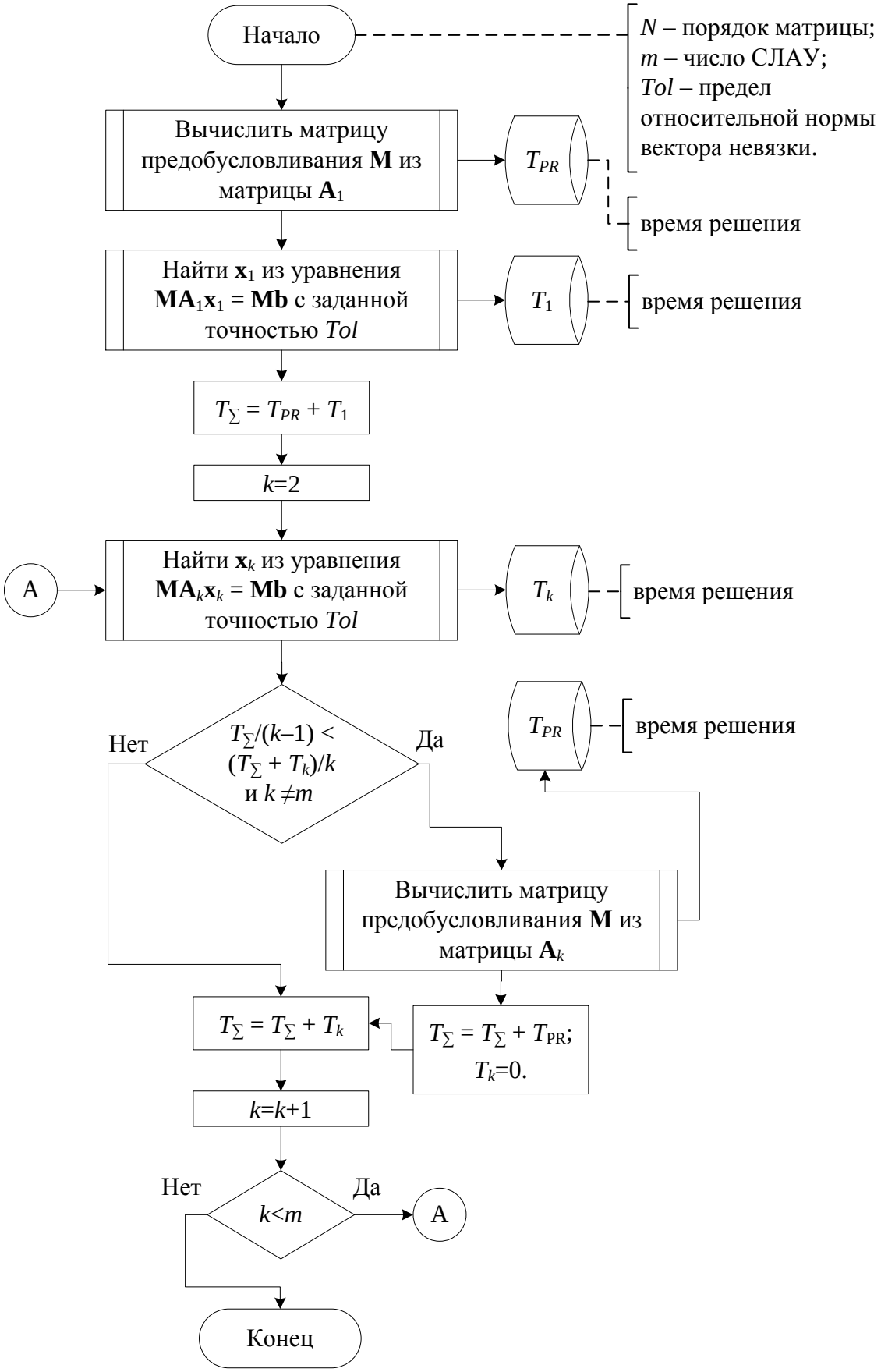


Рисунок 4.3 – Многократное решение СЛАУ с переформированием матрицы предобусловливания по увеличению среднего арифметического времени решения

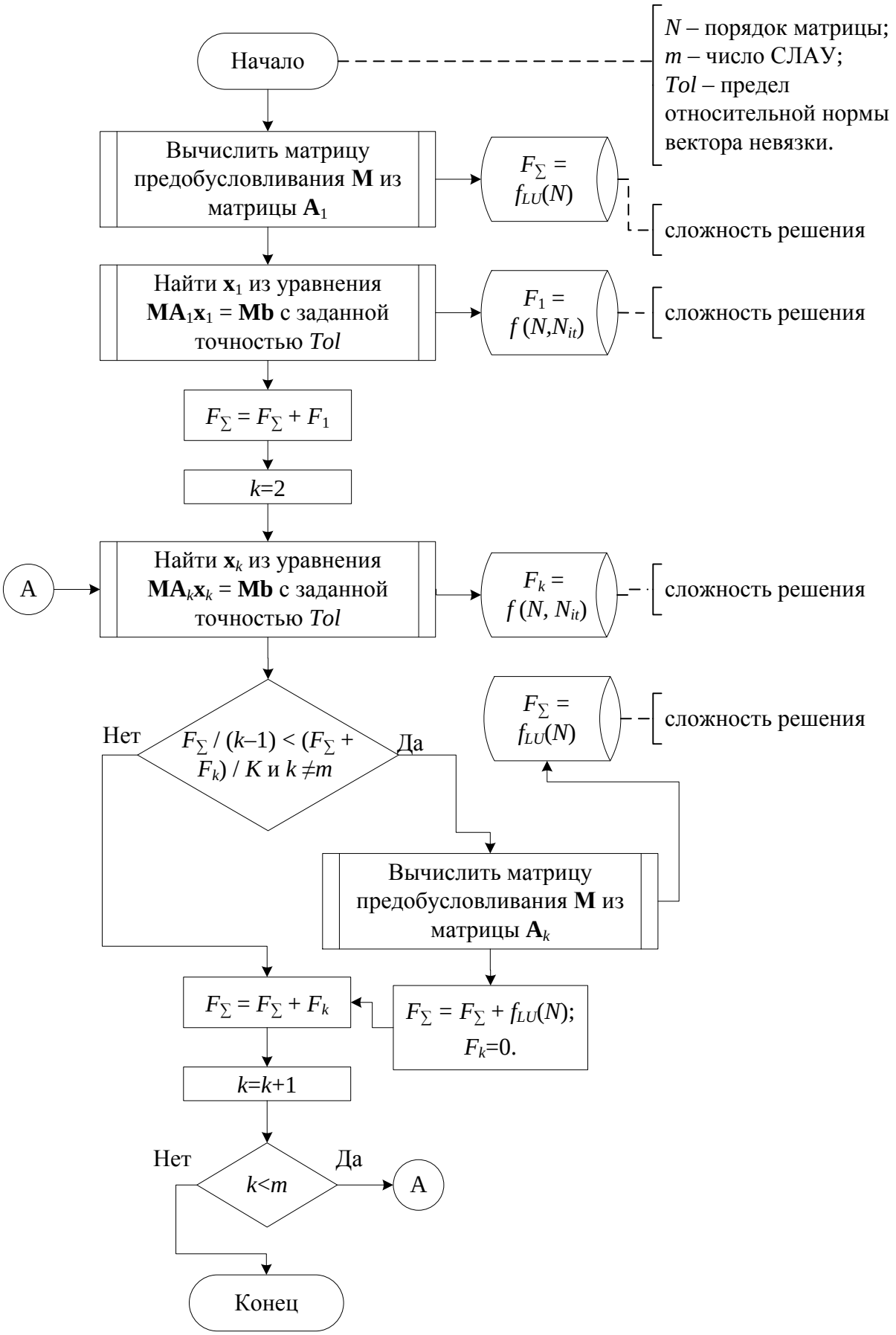


Рисунок 4.4 – Многократное решение СЛАУ с переформированием предобусловливателя при увеличении средней сложности решения k СЛАУ

Рисунок 4.5 – Многократное решение m СЛАУ с выбором очередности решения

Таким образом, разработан программный комплекс, предназначенный для решения СЛАУ итерационными методами BiCGStab и CGS. В нём также реализованы разработанные алгоритмы для ускорения как однократного решения СЛАУ, за счет модифицированного разреженного строчного формата хранения матриц (глава 2), так и многократного решения СЛАУ (глава 3). Практическая значимость разработки подтверждена 9 полученными свидетельствами о регистрации программ для ЭВМ [127, 150–163] и их внедрением.

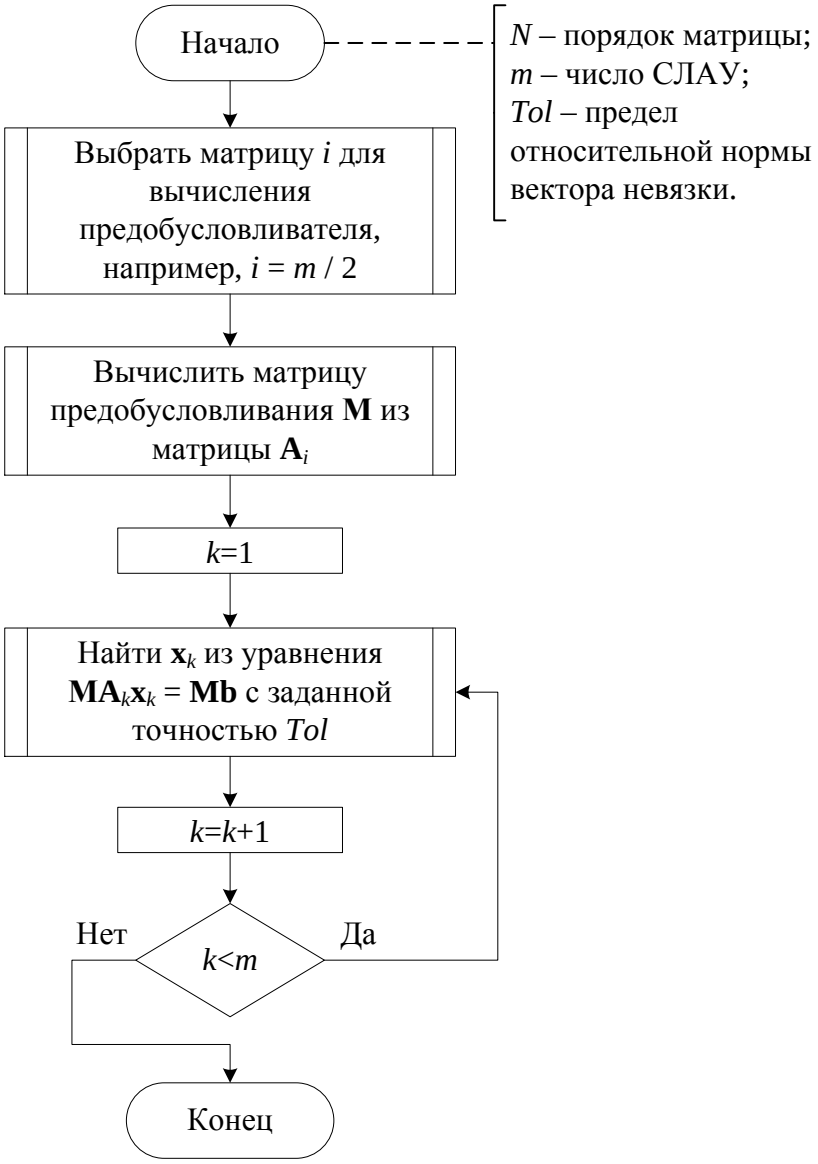


Рисунок 4.6 – Многократное решение СЛАУ
с выбором матрицы для формирования предобусловливателя

ЗАКЛЮЧЕНИЕ

При выполнении диссертационной работы получены следующие результаты [164]:

1. Получены оценки максимального значения коэффициента сжатия широко используемых форматов хранения разреженных матриц.

2. Усовершенствованы алгоритмы ILU(0)-разложения, за счет учета особенностей форматов хранения разреженных матриц.

3. Показано, что использование формата хранения разреженных матриц CSR позволяет уменьшить затраты оперативной памяти до 2-х раз, а время решения СЛАУ – до 2,5 раза.

4. Получены аналитические оценки максимально возможного ускорения многократного решения СЛАУ итерационным методом с предобусловливанием относительно метода исключения Гаусса.

5. Показана перспективность использования итерационных методов решения СЛАУ при многократном решении СЛАУ.

6. Разработаны алгоритмы многократного решения СЛАУ итерационным методом с переформированием предобусловливателя, основанные на задании числа итераций и оценке средних арифметических значений времени и сложности, учитывающие специфику изменения элементов матрицы при многократном решении СЛАУ.

7. Показано, что выбор вектора решения предыдущей СЛАУ в качестве начального приближения текущей позволяет сократить число требуемых итераций и время решения СЛАУ при использовании итерационного метода с предобусловливанием при анализе полосковых структур.

8. Разработан комплекс программ и выполнены вычислительные эксперименты, подтверждающие, что реализованные алгоритмы позволяют уменьшить вычислительные затраты при математическом моделировании методом моментов.

9. Показано, что использование адаптивных условий переформирования предобусловливателя при анализе полосковых структур позволяет уменьшить время решения ряда СЛАУ до 1,6 раза.

10. Выявлено, что время многократного решения СЛАУ итерационным методом с предобусловливанием при анализе полосковой структуры в диапазоне изменения её геометрического параметра может зависеть от выбора очередности решения СЛАУ, и получено уменьшение времени до 1,84 раза.

11. Показано, что выбор средней матрицы из диапазона для формирования предобусловливателя позволяет уменьшить время многократного решения СЛАУ итерационным методом до 2,21 раза.

По результатам диссертационной работы опубликовано 38 научных работ, в т.ч. 1 монография, 10 статей в журналах из перечня ВАК, 8 публикаций, индексируемых WoS и Scopus, 8 тезисов и докладов в трудах симпозиумов и конференций, 15 свидетельств о регистрации программы для ЭВМ.

Таким образом, цель работы достигнута, а полученные результаты имеют значение для технических наук в областях исследования «Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий», «Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента», «Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента» специальности 05.13.18 – Математическое моделирование, численные методы и комплексы программ.

СПИСОК ЛИТЕРАТУРЫ

1. Kane, Y. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media // IEEE Trans. Power Appar. Syst. – 1966. – V. 14. – № 3. – P. 302–307.

2. Харрингтон, Р.Ф. Применение матричных методов к задачам теории поля / Р.Ф. Харрингтон // ТИИЭР. – 1967. – № 2. – С. 5–19.

3. Silvester, P. Finite element solution of saturate magnetic field problems / P. Silvester, M. Chari // IEEE Trans. Power Appar. Syst. – 1970. – V. 89. – № 7. – P. 1642–1651.

4. Weiland, T. A discretization method for the solution of Maxwell's equations for six-component fields // Electronics and Communications AEUE. – 1977. – V. 31(3). – P. 116–120.

5. Johns, P.B. Numerical solution of 2-dimensional scattering problems using a transmission line matrix / P.B. Johns, R.L. Beurle // Proc. of the IEEE. – 1971. – V. 118(9). – P. 1203–1208.

6. Агапов, С.В. Электронные САПР для моделирования электромагнитных излучений от межсоединений печатных плат // Проблемы электромагнитной совместимости технических средств: Сб. докл. Всерос. симпозиума. М.: 2002. – С. 11–13.

7. Григорьев, А.Д. Методы вычислительной электродинамики – М.: Физматлит, 2013. – 430 с.

8. Газизов, Т.Р. Уменьшение искажений электрических сигналов в межсоединениях и влияний преднамеренных силовых электромагнитных воздействий: дис. … д-ра тех. наук: 05.12.07. – Томск, 2010. – 351 с.

9. Газизов, Т.Р. Вычисление ёмкостной матрицы двумерной конфигурации проводников и диэлектриков с ортогональными границами // Известия вузов. Физика. – 2004. – № 3. – С. 88–90.

10. Газизов, Т.Р. Матрица ёмкостных коэффициентов трехмерной системы проводников и диэлектриков // Известия вузов. Физика. – 1998. – № 3. – С. 123–125.

11. Газизов, Т.Р. Уменьшение искажений электрических сигналов в межсоединениях / Под ред. Н.Д. Малютина. – Томск: Изд-во НТЛ, 2003. – 212 с.

12. Djordjevich, A.R. Wideband frequency-domain characterization of FR-4 and time-domain causality / A.R. Djordjevich, R.M. Biljic, V.D. Likar-Smiljanic, T.K. Sarkar // IEEE Transactions on Electromagnetic Compatibility. – 2001. – Vol. 43, no. 4, November. – P. 662–666.

13. Куксенко, С.П. Совершенствование алгоритма вычисления методом моментов ёмкостных матриц структуры проводников и диэлектриков в диапазоне значений диэлектрической проницаемости / С.П. Куксенко, Т.Р. Газизов // Электромагнитные волны и электронные системы. – 2012. – №10. – С. 13–21.

14. Суровцев, Р. С. Исследование ускорения многократного решения СЛАУ с частично изменяющейся матрицей блочным методом / Р.С. Суровцев, В.К. Салов // Электромагнитные волны и электронные системы. – 2012. – Т.17(10). – С. 22–24.

15. Суровцев, Р. С. Вычисление матрицы емкостей произвольной системы проводников и диэлектриков методом моментов: зависимость ускорения за счет блочного LU-разложения от порядка матрицы СЛАУ / Р.С. Суровцев, С.П. Куксенко // Известия высших учебных заведений. Физика. – 2012. – Т. 55 (9/3). – С. 126–130.

16. Лемешко, К. А. Оценка эффективности алгоритма многократного вычисления емкостной матрицы полосковых структур на основе блочного LU-разложения / К.А. Лемешко, Р.Р. Ахунов, М.С. Танаева, О.К. Шпякина // Межд. науч.-техн. конф. студ., асп. и молодых учёных «Научная сессия ТУСУР-2017». – Томск. – 2017. – С. 64–67.

17. Куксенко, С.П. Новые возможности системы моделирования электромагнитной совместимости TALGAT / С.П. Куксенко, А.М. Заболоцкий, А.О. Мелкозеров, Т.Р. Газизов. – Докл. Томск. гос. ун-та систем упр. и радиоэлектроники – 2015. – № 2(36). – С. 45–50.

18. Gazizov, T.R. Analytic expressions for MOM calculation of capacitance matrix of two dimensional system of conductors and dielectrics having arbitrarily oriented boundaries. – Proc. of the 2001 IEEE EMC Symposium, Montreal, Canada, August 13–17. – 2001. – Vol. 1. – P. 151–155.

19. Salov, V.K. Convergence of multiple iterative solution of linear algebraic systems with a fully varying matrix using a single calculated initial preconditioner / V.K. Salov, T.R. Gazizov, O.A. Nikitina. – Innovative Information Technologies: Materials of the International Scientific-Practical Conference, Prague, Czech Republic, April 21–25. – 2014. – Part 2. – P. 452–457.

20. Баландин, М.Ю. Методы решения СЛАУ большой размерности / М.Ю. Баландин, Э.П. Шурина. – Новосибирск: НГТУ, 2000. – 65 с.

21. Турчак, Л.И. Основы численных методов: 2-е изд., перераб. и доп. / Л.И. Турчак, П.В. Плотников. – М.: ФИЗМАТЛИТ, 2002. – 304 с.

22. Вержбицкий, В.М. Численные методы (линейная алгебра и нелинейные уравнения): Учебное пособие. – М: Директ-Медиа, 2013. – 432 с.

23. Golub, Gene H. Matrix Computations: 3rd Ed. / Gene H. Golub, Charles F. Van Loan. – Baltimore, MD, USA: Johns Hopkins University Press, 1996. – 694 pp.

24. Амосов, А.А. Вычислительные методы для инженеров: Учебное пособие / А.А. Амосов, Ю.А. Дубинский, Н.В. Копченова. – М.: Высшая школа, 1994. – 544 с.

25. Райс, Дж. Матричные вычисления и математическое обеспечение: Пер. с англ. – М.: Мир, 1984. – 264 с.

26. Каханер, Д. Численные методы и программное обеспечение. / Д. Каханер, К. Моулер, С. Нэш. – М.: Мир, 1998. – 575 с.

27. Carpes, W.P. Analysis of the coupling of an incident wave with a wire inside a cavity using an FEM in frequency and time domains / W.P. Carpes, L. Pichon and A. Razek // IEEE Trans. on Electromagn. Compat. – August 2002. – V. 44(3). – P. 470–475.

28. The OpenMP API specification for parallel programming [Электронный ресурс]. – Режим доступа: http://openmp.org/.

29. Open MPI: Open Source High Performance Computing [Электронный ресурс]. – Режим доступа: http://www.open-mpi.org/.

30. Параллельные вычисления CUDA [Электронный ресурс]. – Режим доступа: http://www.nvidia.ru/object/cuda-parallel-computing-ru.html.

31. Использование видеокарт для вычислений [Электронный ресурс]. – Режим доступа: http://www.gpgpu.ru/.

32. International Organization for Standardization, Geneva. Information technology – Portable Operating System Interface (POSIX) – Part 1: System Application Program Interface (API) [C Language]. – December 1996.

33. Windows API [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/en-us/library/cc433218.

34. LAPACK – Linear Algebra PACKage [Электронный ресурс]. – Режим доступа: http://www.netlib.org/lapack/.

35. Automatically Tuned Linear Algebra Software (ATLAS) [Электронный ресурс]. – Режим доступа: http://math-atlas.sourceforge.net/.

36. Intel Math Kernel Library (Intel MKL) [Электронный ресурс]. – Режим доступа: https://software.intel.com/en-us/intel-mkl.

37. BLAS (Basic Linear Algebra Subprograms) [Электронный ресурс]. – Режим доступа: http://www.netlib.org/blas/.

38. Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms [Электронный ресурс]. – Режим доступа: http://eigen.tuxfamily.org/index.php.

39. Фаддеев, Д.К. Вычислительные методы линейной алгебры / Д.К. Фаддеев, В.Н. Фаддеева // Зап. научн. сем. ЛОМИ. – 1975. – Т. 54. – 1975. – С. 531–650.

40. Hoffman, Joe D. Numerical methods for engineers and scientists, Second Edition. – New York: CRC Press, 2001. – 823 p.

41. Higham, N.J. Gaussian elimination // Wiley Interdisciplinary Reviews: Computational Statistics. – 2011. – V. 3 (3). – P. 230–238.

42. Lam, L.-Y. Methods of solving linear equations in traditional China / Lam L.-Y. and S. Kangshen // Historia Mathematica. – 1989. – V. 16 (2). – P. 107–122.

43. Калиткин, Н.Н. Численные методы – М: Наука, 1978. – 512 с.

44. Форсайт, Дж. Численное решение систем линейных алгебраических уравнений / Дж. Форсайт, К. Молер – М.: Мир, 1969. – 167 с.

45. Богачев, К.Ю. Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений – М: Москва, 1998. – 137 с.

46. Saad, Y. Iterative methods for sparse linear systems (2nd ed.). Second edition. – Philadelphia, PA, USA: SIAM, 2003. – Pp. 547.

47. Левитин, А. Алгоритмы: введение в разработку и анализ: Пер. с англ. – М. : Издательский дом «Вильямс», 2006. – 576 с.

48. Yi, Q. Automatic Blocking of QR and LU Factorizations for Locality / Q. Yi, K. Kennedy, H. You, S. Keith, J. Dongarra. – Proceedings of the 2004 Workshop on Memory System Performance, New York, NY, USA. – 2004. – P. 12–22.

49. Li, N. Crout Versions of ILU for General Sparse Matrices / N. Li, Y. Saad, E. Chow // SIAM Journal on Scientific Computing. – 2003. – V. 25(2). – P. 716–728.

50. He, K. GPU-Accelerated Parallel Sparse LU Factorization Method for Fast Circuit Analysis / K. He, S.X. Tan, H. Wang, G. Shi, // IEEE Trans. Very Large Scale Integr. Syst. – 2016. – V. 24. – P. 1140–1150.

51. Pernice, M. A. Multigrid-Preconditioned Newton-Krylov Method for the Incompressible Navier-Stokes Equations / M. Pernice, M.D. Tocci // SIAM Journal on Scientific Computing. – 2001. – V. 23(2). – P. 398–418.

52. Крукиер, Л.А. Обзор методов подпространства Крылова / Л.А. Крукиер, О.А. Пичугина, Л.Г. Чикина. – XIV Международная конференция-школа с международным участием «Современные проблемы математическо-

го моделирования», Ростов-на-Дону: изд-во Южного федерального университета. – 2011. – С. 203–243.

53. Крылов, А.Н. О численном решении уравнения, которым в технических вопросах определяются частоты малых колебаний материальных систем. // Известия Академии наук СССР. VII серия. Отделение математических и естественных наук. – 1931. – №4. – С. 491–539.

54. Куксенко, С. П. Итерационные методы решения системы линейных алгебраических уравнений с плотной матрицей / С. П. Куксенко, Т.Р. Газизов. – Томск: Томский государственный университет, 2007. – 208 с.

55. Chow, E. Approximate inverse preconditioners via sparse-sparse iterations / E. Chow, Y. Saad // SIAM J. Sci. Comput. – 1998. – V.19. – P. 995–1023.

56. Kaporin, I.E. New convergence results and preconditioning strategies for the conjugate gradient method // Linear Algebra Appl. – 1994. – V. 1. – P. 179–210.

57. Grote, M.J. Thomas Parallel Preconditioning with Sparse Approximate Inverses / M.J. Grote, T. Huckle // SIAM J. Sci. Comput. – 1997. – V. 18(3). – P. 838–853.

58. Benzi, M. A sparse approximate inverse preconditioner for the conjugate gradient method / M. Benzi, C.D. Meyer, M. Tuma // SIAM J. Sci. Comput. – 1996. – V.17. – P. 1135–1149.

59. Benzi, M. A comparative study of sparse approximate inverse preconditioners / M. Benzi, M. Tuma // Appl. Numer. Math. – 1999. – V. 30. – P. 305–340.

60. Kolotilina, L.Yu. Factorized sparse approximate inverse preconditioning I. Theory / L. Yu. Kolotilina and A.Yu. Yeremin // SIAM J. Matrix Anal. Appl. – 1993. – V. 14. – P. 45–58.

61. Dehnavi, M.M. Parallel Sparse Approximate Inverse Preconditioning on Graphic Processing Units / M.M. Dehnavi, D.M. Fernandez, J. Gaudiot, D.D. Giannacopoulos // Parallel and Distributed Systems, IEEE Transactions on. – 2013. – V. 24(9). – P. 1852–1862.

62. Mori, H. An ILU(p)-Preconditoner Bi-CGStab Method for Power Flow Calculation / H. Mori, F. Iizuka // Power Tech, 2007 IEEE Lausanne. – 2007. – P. 1474–1479.

63. Колотилина, Л.Ю. Явно предобусловленные системы линейных алгебраических уравнений с плотной матрицей // Советская математика. – 1988. – № 43. – С. 2566–2573.

64. Куксенко, С.П. Совершенствование способов предфильтрации для решения СЛАУ с плотной матрицей итерационными методами с предобусловливанием в задачах вычислительной электродинамики / С.П. Куксенко, Т.Р. Газизов // Электромагнитные волны и электронные системы. – 2007. – № 9. – С. 12–17.

65. Gatsis, J. Preconditioning Techniques for a Newton–Krylov Algorithm for the Compressible Navier–Stokes Equations // The thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy Graduate. Department of Institute for Aerospace Studies University of Toronto. – 2013. – P. 144.

66. Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differentialand integral operators // J. Res. Nat. Bur. Stand. –1950. – Vol. 45. – P. 255–282.

67. Arnoldi, W. The principle of minimized iterations in the solution of the matrix eigenvalue problem // Quarterly of Applied Mathematics. – 1951. – V. 9. – P. 17–29.

68. Hestenes, M. Methods of conjugate gradients for solving linear systems / M. Hestenes, E. Stiefel // J. Res. Nat. Bur. Stand. – 1952. – Vol. 49. – P. 409–436.

69. Lanczos, C. Solution of systems of linear equations by minimized iterations // Journal of Research of the National Bureau of Standards. – 1952. – V. 49. – P. 33–53.

70. Paige, C. Solution of sparse indefinite systems of linear equations / C. Paige, M. Saunders // SIAM Journal on Numerical Analysis. –1975. – Vol. 12. –P. 617–629.

71. Fletcher, R. Conjugate gradient methods for indefinite systems – Proceedings of the Dundee Biennal Conference on Numerical Analysis 1974, edited by G. Watson, Springer Verlag, New York. – 1975. – P. 73–89.

72. Concus, P. A. Generalized conjugate gradient method for nonsymmetric systems of linear equations / P. Concus, G. Golub // Computer methods in Applied Sciences and Engineering, Second International Symposium, edited by R. Glowinski and J. Lions, Springer Verlag. – New York. – December 1976. – P. 56–65.

73. Vinsome, P. ORTHOMIN: an iterative method for solving sparse sets of simultaneous linear equations // Proceedings of the Fourth Symposium of Reservoir Simulation, Society of Petroleum Engineers of AIME. – 1976. – P. 149–159.

74. Meijerink, J.A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix / J.A. Meijerink, H.A. van der Vorst // Mathematics of Computation. – 1977. – V. 31(137). – P. 148–162.

75. Widlund, O. A Lanczos method for a class of non-symmetric systems of linear equations // SIAM J. Numer. Anal. – 1978. – V. 15. – P. 801–802.

76. Jea, K. Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods / K. Jea, D. Young // Linear Algebra and its Applications. – 1980. – V. 34. – P. 159–194.

77. Saad, Y. Krylov subspace methods for solving large unsymmetric linear systems // Mathematics of Computation. – 1981. – V. 37. – P. 105–126.

78. Paige, C.C. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares / C.C. Paige, M.A. Saunders // ACM Transactions on Mathematical. –1982. – V. 8(1). – P. 43–71.

79. Eisenstat, S.C. Variational iterative methods for nonsymmetric systems of linear equations / S.C. Eisenstat, H.C. Elman, M.H. Schultz // SIAM Journal on Numerical Analysis. – V. 20(2). – P. 345–357.

80. Saad, Y. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems / Y. Saad, M. H. Schultz // SIAM Journal on Scientific and Statistical Computing. – 1986. – P. 7:856–869.

81. Sonneveld, P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems // SIAM J. Sci. Statist. Comput. – 1982. –V. 10. – P. 36–52.

82. Freund, R.W. QMR: a quasi-minimal residual method for non-Hermitian linear systems / R.W. Freund, N.M. Nachtigal // Numer. Math. – 1991. – P. 60:315–339.

83. Van der Vorst, H.A. BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems // SIAM J. Sci. Stat. Comput. – 1992. – Vol. 13(2) – P. 631–644.

84. Gutknecht, M.H. Variants of BICGSTAB for matrices with complex spectrum // SIAM J. Sci. Comput. – 1993. – V. 14. – P. 1020–1033.

85. Sleijpen, G.L. BiCGStab(l) for linear equations involving unsymmetric matrices with complex spectrum / G.L.G. Sleijpen, D.R. Fokkema // Electronic Transactions on Numerical Analysis. – 1993 – P. 1:11–32.

86. Freund, R. W. An implementation of the QMR method based on coupled two-term recurrences / R.W. Freund, N.M. Nachtigal // SIAM Journal on Scientific Computing. – 1994. – Vol. 15. – P. 313–337

87. Weiss, R. Error-minimizing Krylov subspace methods // SIAM Journal on Scientific Computing. – 1994. – V. 15. – P. 511–527.

88. Chan, T. F. A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems / T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C.H. Tong // SIAM Journal on Scientific Computing. – 1994. – V. 15. – P. 338–347.

89. Kasenally, E.M. GMBACK: a generalized minimum backward error algorithm for nonsymmetric linear systems // SIAM J. Sci. Comput. – 1995. – V. 16. – P. 698–719.

90. Fokkema, D. R. Generalized conjugate gradient squared / D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, // Journal of Computational and Applied Mathematics. – 1996. – V. 71. – P. 125–146.

91. De Sturler, E. Truncation strategies for optimal Krylov subspace methods // SIAM Journal on Numerical Analysis. – 1999. – V. 36(3). – P. 864–889.

92. Писсанецки, С. Технология разреженных матриц: Пер. с англ. –М.: Мир, 1988. – 410 с.

93. Alvarado, F.L. A note on sorting sparse matrices // Proceedings of the IEEE. – 1979. – V.67(9). –P. 1362–1363.

94. Тьюарсон, Р. Разреженные матрицы: Перевод с английского Э. М. Пейсаховича, под редакцией Х. Д. Икрамова. – М.: Мир, 1977. – 172 с.

95. Knuth, D. The Art of Computer Programming: Fundamental algorithms. – Addison-Wesley, 1968. – 634 p.

96. Rheinboldt, W.C. Programs for the solution of large sparse matrix problems based on the arc-graph structure / W.C. Rheinboldt, C.K. Mesztenyi, Computer Science Center, University of Maryland, College Park MD. Technical Report TR-262. – 1973.

97. Larcombe, M. A list processing approach to the solution of large sparse sets of matrix equations and the factorization of the overall matrix // in Large Sparse Sets of Linear Equations, Reid, J. K., Ed., Academm Press, London. – 1971.

98. Sherman, A.H. On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations. Ph.D. dissertation, Department of Computer Science, Yale University, 1975.

99. Chang, A. Application of sparse matrix methods in electric power system analysis // Sparse Matrix Proceedings R. Willoughby, Ed., IBM Watson Research Center, RAl1707 (March, 1969) 113–122.

100. Gustavson, F.G. Some basic techniques for solving sparse systems of linear equations in Sparse Matrices and Their Applications // Rose, D. J. & Wi-Hougby, R. A.. Eds, New York, Plenum Press. – 1972. –P. 41–52.

101. Saad, Y. Numerical solution of large nonsymmetric eigenvalue problems // Comp. Phys. Comm. – 1989. – P. 53:71–90.

102. Dongarra, J.J. LINPACK Users' Guide / J.J. Dongarra, J. R. Bunch, C.B. Moler, G.W. Stewart. // SIAM, Philadelphia. – 1979.

103. Tinetti, F. Performance of Scientific Processing in NOW: Matrix Multiplication Example // Journal of Computer Science and Technology. – 2001. – V. 1(4). – P. 78–87.

104. Sparse matrix storage formats [Электронный ресурс]. – Режим доступа: http://www.netlib.org/utk/people/JackDongarra/etemplates/node372.html.

105. Jiang, J. The spatial relationship of DCT coefficients between a block and its sub-blocks / J. Jiang, G. Feng // Signal Processing, IEEE Transactions on. – 2002. – V. 50(5). – P. 1160–1169.

106. Vazquez, F. Improving the performance of the sparse matrix vector product with GPUs / F. Vazquez, G. Ortega, J.J. Fernandez, E.M. Garzon // Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. – 2010. – P. 1146–1151.

107. Kestur, S. Towards a universal FPGA matrix-vector multiplication architecture / S. Kestur, J.D. Davis, E.S. Chung // Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on. – 2012. – P. 9–16.

108. Karimi, S. A New Iterative Solution Method for Solving Multiple Linear Systems // Advances in Linear Algebra & Matrix Theory. – 2012. – V. 2(3). – P. 25–30.

109. Yu, F. Recursive least-squares algorithms with good numerical stability for multichannel active noise control / F. Yu, M. Bouchard // Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on. – 2001. – V.5. – P. 3221–3224.

110. Kress, R. Linear Integral Equations: Springer-Verlag. – New York. – 1989.

111. Jain, Anil K. Fundamentals of Digital Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1989.

112. Calgaro, C. Incremental incomplete LU factorizations with applications to time-dependent PDEs/ C. Calgaro, J.P. Chehab, Y. Saad // Numer. Lin. Algebra with Appl. – 2010. – No 17(5). – P. 811–837.

113. Simoncini, V. A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides / V. Simoncini and E. Gallopoulos // Journal of Computational and Applied Mathematics. – 1996. – V.66. –P. 457–469.

114. Toutounian, F. Global least squares method (Gl-LSQR) for solving general linear systems with several right-hand sides / F. Toutounian, S. Karimi // Applied Mathematics and Computation. – 2006. – V. 178(2). – P. 452–460.

115. Simoncini, V. An iterative method for nonsymmetric systems with multiple right-hand sides / V. Simoncini, E. Gallopoulos // SIAM J. Sci. Comput. – 1995. – V. 16(4). – P. 917–933.

116. Liu, Y. An efficient method MEGCR for solving systems with multiple right-hand sides in 3-D parasitic inductance extraction / Y. Liu, G. Xiaobo, W. Zeyi // Design Automation Conference, 2004. Proceedings of the ASP-DAC 2004. Asia and South Pacific. – 2004. – P. 702–706.

117. Суровцев, Р.С. Ускорение многократного решения СЛАУ с частично изменяющейся матрицей / Р.С. Суровцев, С.П. Куксенко, Т.Р. Газизов // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2011. – №2(24), часть 1. – С. 141–144.

118. Ахунов, Р.Р. Обзор форматов хранения разряженных матриц // Межд. науч.-техн. конф. студ., асп. и молодых учёных «Научная сессия ТУСУР-2010». – Томск. – 2010. – С. 137–139.

119. Ахунов, Р.Р. Форматы хранения разреженных матриц и ускорение решения СЛАУ с плотной матрицей итерационными методами / Р.Р. Ахунов, С.П. Куксенко, В.К. Салов, Т.Р. Газизов // Численные методы и вопросы организации вычислений. XXV. Зап. научн. семин. ПОМИ. – 2012. – Т. 405(25). – С. 24–39.

120. Ахунов, Р.Р. Усовершенствование алгоритма ILU(0)-разложения, использующего разреженный строчный формат / Р.Р. Ахунов, С.П. Куксенко, В.К. Салов, Т.Р. Газизов // Численные методы и вопросы организации вычислений. XXV. Зап. научн. семин. ПОМИ. – 2012. – Т. 405(25). – С. 40–53.

121. Салов, В.К. Ускорение вычислений в задачах моделирования ЭМС. / В.К. Салов, С.П. Куксенко, М.Е. Комнатнов, Р.Р. Ахунов, А.О. Мелкозеров, Р.И. Аширбакиев, Т.Р. Газизов // Труды 9-го Межд. Симп. по электромагнитной совместимости и электромагнитной экологии, г. Санкт-Петербург, 13–16 сентября 2011 г. – С. 269–272.

122. Газизов, Т.Р. Пути решения актуальных проблем проектирования радиоэлектронных средств с учетом электромагнитной совместимости / Т.Р. Газизов, А.М. Заболоцкий, А.О. Мелкозеров, С.П. Куксенко, П.Е. Орлов, В.К. Салов, И.Ф. Калимулин, Р.Р. Аширбакиев, Р.Р. Ахунов, Р.С. Суровцев, М.Е. Комнатнов // Техника радиосвязи. – 2014. – № 2 (22). – С. 11–22.

123. Gazizov, T. New results on EMC simulation for space projects of TUSUR University / T. Gazizov, A. Melkozerov, P. Orlov, V. Salov, R. Ashirbakiev, R. Akhunov, S. Kuksenko, I. Kalimulin // IEEE International Conference on Numerical Electromagnetic Modeling and Optimization for RF, Microwave, and Terahertz Applications. May 14–16, 2014, Pavia, Italy. P. 1–4.

124. Ахунов, Р.Р. Вычисление матрицы емкостей произвольной системы проводников и диэлектриков методом моментов: оценка использования разреженного строчного формата при решении СЛАУ методом Bicgstab / Р.Р. Ахунов, С.П. Куксенко // Известия высших учебных заведений. Физика. – 2012. – Т. 55 (7/2). – С. 27-30.

125. Лежнин, Е.В. Алгоритм ILU(0)-разложения с использованием OpenMP / Е.В. Лежнин, Р.Р. Ахунов, С.П. Куксенко // Докл. Томск. гос. ун-та систем упр. и радиоэлектроники. – 2015. – № 3(37). – С. 181–183.

126. Gibson, W.C. The method of moments in electromagnetics. – Boca Raton, FL: Taylor & Francis Group, 2007. – P. 272.

127. Свидетельство о государственной регистрации программы для ЭВМ №2015615730. Организация хранения плотной матрицы в модифицированном строчном разреженном формате после предфильтрации, основанной на максимальном элементе матрицы. Авторы: Ахунов Р.Р., Куксенко С.П.,

Газизов Т.Р. Заявка №2015612891. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.

128. Газизов, Т.Р. Оптимизация допуска обнуления при решении СЛАУ итерационными методами с предобусловливанием в задачах вычислительной электродинамики. / Т.Р. Газизов, С.П. Куксенко // Электромагнитные волны и электронные системы. – 2004. – №8. – С. 26–28.

129. Ахунов, Р.Р. Анализ полосковых структур радиоэлектронных устройств с уменьшенными вычислительными затратами // Докл. Томск. гос. ун-та систем упр. и радиоэлектроники. – 2017. – № 2 (20). – С. 96–99.

130. Sychev, A.N. Modeling of the coupled microstrip lines with using numerical conformal transformations / A.N. Sychev, M.A. Chekalin, V.A. Shestakov, S.M. Struchkov // European Microwave Week 2013 Conf. Proc. – Eur. MC 2013. – 6–11 Oct. 2013, Nuremberg, Germany. – P. 1107–1110.

131. Venkataraman, J. Analysis of arbitrarily oriented microstrip transmission lines in arbitrarily shaped dielectric media over a finite ground plane / J. Venkataraman, S.M. Rao, A.R. Djordjevic, T.K. Sarkar and Y. Naiheng // IEEE Tran. Microwave Theory Tech. – 1985. – V. 33 (10). – P. 952–960.

132. Weeks, W. T. Calculation of coefficients of capacitance of multiconductor transmission lines in the presence of a dielectric interface // IEEE Tran. Microwave Theory Tech. – 1970. – V. 18 (1). – P. 35–43.

133. Sarkar, T. K. The electrostatic field of conducting bodies in multiple dielectric media // T. K. Sarkar and R. F. Harrington // IEEE Tran. Microwave Theory Tech. – 1984. – V. 32 (11). – P. 1441–1448.

134. Ахунов, Р.Р. Многократное решение СЛАУ с частично изменяющейся матрицей итерационным методом / Р.Р. Ахунов, С.П. Куксенко, В.К. Салов, Т.Р. Газизов // Численные методы и вопросы организации вычислений. XXV. Зап. научн. семин. ПОМИ. – 2013. – Т. 419. – С. 16–25.

135. Ахунов, Р.Р. Ускорение многократного решения СЛАУ итерационным методом при вычислении емкости микрополосковой линии в широком диапазоне изменения ее размеров. / Р.Р. Ахунов, С.П. Куксенко,

Т.Р. Газизов // Численные методы и вопросы организации вычислений. XXVII, Зап. научн. сем. ПОМИ. – 2014. – Т. 428. – С. 32–41.

136. Ahunov, R.R. Multiple solution of linear algebraic systems by an iterative method with recomputed preconditioner in the analysis of microstrip structures / S.P. Kuksenko, T.R. Gazizov Proc. of the 13th Int. Conf. of Numerical Analysis and Applied Mathematics. Sept. 23–29, Rhodes, Greece – 2015. – P. 1–4.

137. Gazizov, T. Ensurance and simulation of electromagnetic compatibility: recent results in TUSUR University // T. Gazizov, A. Melkozerov, A. Zabolotsky, S. Kuksenko, P. Orlov, V. Salov, R. Akhunov, I. Kalimulin, R. Surovtsev, M. Komnatnov, A. Gazizov. International Conference on Applied Physics, Simulation and Computers, Austria, Vienna, 15–17 March 2015.

138. Kuksenko, S.P. New developments for improved simulation of interconnects based on method of moments / S.P. Kuksenko, T.R. Gazizov, A.M. Zabolotsky, R.R. Ahunov, R.S. Surovtsev, V.K. Salov, Eg.V. Lezhnin. Advances in Intelligent Systems Research. Proc. of the 2015 Int. Conf. on Modelling, Simulation and Applied Mathematics (MSAM2015). Phuket, Thailand. August 23–24. – 2015. – P. 1–8.

139. Kuksenko, S.P. Choosing order of operations to accelerate strip structure analysis in parameter range / S.P. Kuksenko, R.R. Ahunov, T.R. Gazizov. // Journal of Physics: Conf. Series. – 2018. Vol. 1015 (3). – Paper 032076. – P. 1–6.

140. Kuksenko, S.P. Dense linear system solution by preconditioned iterative methods in computational electromagnetics / S.P. Kuksenko, T.R. Gazizov. – 19th Int. Zurich Symp. Electromagn. Compatibility. – 2008. – P. 918–921.

141. Газизов, Т.Р. Основы автоматизации проектирования радиоэлектронных устройств: учебное пособие / Т.Р. Газизов. – Томск: ТМЦДО, 2005. – 243 с.

142. Ахунов, Р.Р. Многократное решение СЛАУ итерационным методом с переформированием матрицы предобусловливания / Р.Р. Ахунов, С.П. Куксенко, Т.Р. Газизов // Численные методы и вопросы организации вычислений. XXVII, Зап. научн. сем. ПОМИ. – 2014. – Т. 428. – С. 42–48.

143. Ахунов, Р.Р. Ускорение многократного решения СЛАУ с изменяющейся матрицей / Р.Р. Ахунов, С.П. Куксенко. – Международная конференция "Актуальные проблемы вычислительной и прикладной математики 2015", посвященную 90-летию со дня рождения академика Гурия Ивановича Марчука, Новосибирск, 19–23 октября. – 2015. – С. 84–90.

144. Ахунов, Р.Р. Многократное решение систем линейных алгебраических уравнений итерационным методом с адаптивным переформированием предобусловливателя / Р.Р. Ахунов, С.П. Куксенко, Т.Р. Газизов // Ж. вычисл. матем. и матем. физ. – 2016. – 56(8). – С. 1395–1400.

145. Zabolotsky, A.M. Improved design of modal filter for electronics protection / A.M. Zabolotsky, T.R. Gazizov, A.O. Melkozerov, P.E. Orlov, E.S. Dolganov. – Proc. of 31-th Int. conf. on lightning protection. Vienna, Austria. September 2–7. – 2012.

146. Вержбицкий, В.М. Основы численных методов / В.М. Вержбицкий. – М: Высшая школа, 2002. – 840 с.

147. Gazizov, T.R. Acceleration of multiple solution of linear systems for analyses of microstrip structures /T.R. Gazizov, S.P. Kuksenko, R.R. Akhunov // International journal of mathematical models and methods in applied sciences. – 2015. Vol. 9. – P. 721–726.

148. Заболоцкий, А.М. Использование зеркальной симметрии для совершенствования модальной фильтрации / А.М. Заболоцкий. – Докл. Томск. гос. ун-та систем упр. и радиоэлектроники. – 2015. – № 2(36). – С. 41–44.

149. Ахунов, Р.Р. Простой способ ускорения вычисления емкостных матриц полосковой структуры при изменении её геометрического параметра / Р.Р. Ахунов, С.П. Куксенко, Т.Р. Газизов // Докл. Томск. гос. ун-та систем упр. и радиоэлектроники. – 2015. – № 4. – С. 144–148.

150. Свидетельство о государственной регистрации программы для ЭВМ №2015615835. ILU(0)-разложение матрицы, хранимой в разреженном строчном формате, с последовательным перебором элементов. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612781. Дата поступления

09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 май 2015 г.

151. Свидетельство о государственной регистрации программы для ЭВМ №2015616205. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612783. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 03 июня 2015 г.

152. Свидетельство о государственной регистрации программы для ЭВМ №2015616124. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате, с использованием вспомогательного вектора. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612895. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 01 июня 2015 г.

153. Свидетельство о государственной регистрации программы для ЭВМ №2015616321. Решение СЛАУ с матрицей, полученной с помощью ILU(0)-разложения и хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612893. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.

154. Свидетельство о государственной регистрации программы для ЭВМ №2015616322. Многократное решение СЛАУ итерационным методом BiCGstab с использованием при решении текущей системы вектора решения предыдущей. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612894. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.

155. Свидетельство о государственной регистрации программы для ЭВМ №2015616320. Многократное решение СЛАУ итерационным методом BiCGStab с использованием матрицы предобусловливания, полученной при решении первой системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р.

Заявка №2015612892. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.

156. Свидетельство о государственной регистрации программы для ЭВМ №2015615837. Многократное решение СЛАУ итерационным методом BiCGStab с использованием переформирования матрицы предобусловливания по заданному порогу числа итераций. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612782. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.

157. Свидетельство о государственной регистрации программы для ЭВМ №2015615729. Многократное решение СЛАУ итерационным методом BiCGStab с переформированием матрицы предобусловливания при превышении среднего времени решения одной системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612890. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.

158. Свидетельство о государственной регистрации программы для ЭВМ №2013619615. TALGAT 2012. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Аширбакиев Р.И., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Газизов Р.Р., Ахунов Р.Р. Заявка №2013617773. Дата поступления 29 августа 2013 г. Зарегистрировано в Реестре программ для ЭВМ 11 октября 2013 г.

159. Свидетельство о государственной регистрации программы для ЭВМ №20156143. TALGAT 2013. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Русл.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Новикова Е.А. Заявка № 2015611288. Дата поступления 03 марта 2015 г. Зарегистрировано в Реестре программ для ЭВМ 16 апреля 2015 г.

160. Свидетельство о государственной регистрации программы для ЭВМ №2015617550. TALGAT 2014. Авторы: Газизов Т.Р., Мелкозеров А.О.,

Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Р.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Новикова Е.А., Газизов Руст.Р., Веселовский А.В. Заявка №2015614488. Дата поступления 27 мая 2015 г. Зарегистрировано в Реестре программ для ЭВМ 17 июля 2015 г.

161. Свидетельство о государственной регистрации программы для ЭВМ №2015660487. TALGAT 2015. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Русл.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Газизов Руст.Р., Веселовский А.В. Заявка №2015617580. Дата поступления 17 августа 2015 г. Зарегистрировано в Реестре программ для ЭВМ 01 октября 2015 г.

162. Свидетельство о государственной регистрации программы для ЭВМ №2016662520. TALGAT 2016. Заявка №20166619296. Дата поступления 01 сентября 2016 г. Зарегистрировано в Реестре программ для ЭВМ 14 ноября 2016 г. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Русл.Р., Салов В.К., Лежнин Е.В., Орлов П.Е., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Газизов Руст.Р., Газизов А.Т., Веселовский А.В., Квасников А.А., Носов А.В., Белоусов А.О., Буичкин Е.Н., Лесков А.Н., Демаков А.В., Лемешко К.А., Собко А.А., Осинцев А.В., Калимулин И.Ф. Заявка №20166619296. Дата поступления 01 сентября 2016 г. Зарегистрировано в Реестре программ для ЭВМ 14 ноября 2016 г.

163. Свидетельство о государственной регистрации программы для ЭВМ №2018611481. TALGAT 2017. Заявка №2017663209. Дата поступления 13 декабря 2017 г. Зарегистрировано в Реестре программ для ЭВМ 02 февраля 2018 г. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Русл.Р., Бусыгина А.В., Лежнин Е.В., Орлов П.Е., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Газизов Руст.Р., Газизов А.Т., Хажибеков Р.Р., Квасников А.А., Носов А.В., Белоусов А.О., Тер-

нов С.А., Сагиева И.Е., Демаков А.В., Осинцев А.В., Собко А.А. Заявка №2017663209. Дата поступления 13 декабря 2017 г. Зарегистрировано в Реестре программ для ЭВМ 02 февраля 2018 г.

164. Ахунов, Р.Р. Многократное решение систем линейных алгебраических уравнений итерационными методами с предобусловливанием в задачах электромагнитной совместимости. / Р.Р. Ахунов, С.П. Куксенко, Т.Р. Газизов, П.Е. Орлов // . – Томск: Изд-во Томск. гос. ун-та систем упр. и радиоэлектроники, 2015. – 152 с.

ПРИЛОЖЕНИЕ А

(справочное)

Комплекс программ для решения СЛАУ итерационными методами

В приложении приведены исходные коды программ, разработанных на основе алгоритмов, предложенных в главах 2 и 3. Алгоритмы реализованы в среде разработки Microsoft Visual C++ 2008.

Форматы хранения разреженных матриц

ILU(0)-разложение матрицы, хранимой в формате CSR

Далее приведен исходный код программы (рисунок 2.4).

```
void LU_Alg(int n ,double* &aelem_MatrM,int* &jptr_MatrM,
int* &iptr_MatrM)
{
  int k,i,j,s;
  int NachStroki, KonSroki, NachStroki1,KonSroki1, TekEl1;
  double diaronEl;
  bool Prodolgat,NashliElem;
  for (i = 1;i < n;i++)
  {
    for (k = 0;k <= (i–1);k++)
    {
      TekEl1 = –1;
      for (j = iptr_MatrM[k]; j <= iptr_MatrM[k+1];j++)
      {
        if (jptr_MatrM[k] == k)
        {
          TekEl1 = j;
          break;
        }
        if (jptr_MatrM[k] > k)
          break;
      }
      diaronEl = aelem_MatrM[TekEl1];
      NashliElem = false;
      NachStroki = iptr_MatrM[i];
      KonSroki = iptr_MatrM[i+1];
      NachStroki1 = iptr_MatrM[k];
      KonSroki1 = iptr_MatrM[k+1];
      for (s = NachStroki;s < KonSroki;s++)
      {
        if(jptr_MatrM[s] == k)
        {
          aelem_MatrM[s] = aelem_MatrM[s]/diaronEl;
          TekEl1 = s;
          NashliElem = true;
```

```
            NachStroki = s;
            break;
          }
          if (jptr_MatrM[s]>k)
            break;
        }
        if (NashliElem)
        {
          while (jptr_MatrM[NachStroki]<(k+1))
            NachStroki++;
          while (jptr_MatrM[NachStroki1]<(k+1))
            NachStroki1++;
          if (KonSroki1 <= NachStroki1)
            continue;
          if (KonSroki <= NachStroki)
            continue;
          Prodolgat = true;
          while (Prodolgat)
          {
            if (jptr_MatrM[NachStroki] == jptr_MatrM[NachStroki1])
            {
              aelem_MatrM[NachStroki] –=
                    aelem_MatrM[TekEl1] *    aelem_MatrM[NachStroki1];
              NachStroki++;
              NachStroki1++;
            }
            if (jptr_MatrM[NachStroki]>jptr_MatrM[NachStroki1])
              NachStroki1++;
            if (jptr_MatrM[NachStroki1]>jptr_MatrM[NachStroki])
              NachStroki++;
            if (NachStroki1 >= KonSroki1)
              Prodolgat = false;
            if (NachStroki >= KonSroki)
              Prodolgat = false;
          }
        }
      }
    }
  }
```

ILU(0)-разложение матрицы с использованием формата CSR с дополнительным вектором Diag

Далее приведен исходный код программы (рисунок 2.6).

```
void LU_Alg_Diag(int n ,double* &aelem_MatrM,int* &jptr_MatrM,
int* &iptr_MatrM,int* &UdialM)
```

```
{
  int k,i, s;
  int NachStroki,KonSroki, NachStroki1,KonSroki1, TekEl1;
  double diaronEl;
  bool Prodolgat,NashliElem;
  for (i = 1;i<n;i++)
  {
    for (k = 0;k <= (i–1);k++)
    {
      TekEl1 = UdialM[k];
      diaronEl = aelem_MatrM[TekEl1];
      NashliElem = false;
      NachStroki = iptr_MatrM[i];
      KonSroki = iptr_MatrM[i+1];
      NachStroki1 = iptr_MatrM[k];
      KonSroki1 = iptr_MatrM[k+1];
      for (s = NachStroki;s<KonSroki;s++)
      {
        if(jptr_MatrM[s] == k)
        {
          aelem_MatrM[s] = aelem_MatrM[s]/diaronEl;
          TekEl1 = s;
          NashliElem = true;
          NachStroki = s;
          break;
        }
        if (jptr_MatrM[s] > k)
          break;
      }
      if (NashliElem)
      {
        while (jptr_MatrM[NachStroki] < (k+1))
          NachStroki++;
        while (jptr_MatrM[NachStroki1] < (k+1))
          NachStroki1++;
        if (KonSroki1 <= NachStroki1)
          continue;
        if (KonSroki <= NachStroki)
          continue;
        Prodolgat = true;
        while (Prodolgat)
        {
          if (jptr_MatrM[NachStroki] == jptr_MatrM[NachStroki1])
          {
```

```
            aelem_MatrM[NachStroki] –=
                 aelem_MatrM[TekEl1] * aelem_MatrM[NachStroki1];
            NachStroki++;
            NachStroki1++;
          }
          if (jptr_MatrM[NachStroki]>jptr_MatrM[NachStroki1])
            NachStroki1++;
          if (jptr_MatrM[NachStroki1]>jptr_MatrM[NachStroki])
            NachStroki++;
          if (NachStroki1 >= KonSroki1)
            Prodolgat = false;
          if (NachStroki >= KonSroki)
            Prodolgat = false;
        }
      }
    }
  }
}
```

ILU(0)-разложение с использованием формата CSR
с двумя дополнительными векторами

Далее приведен исходный код программы (рисунок 2.11).

```
void LU_Alg_Diag_Modif(int n ,double* &aelem_MatrM,int* &jptr_MatrM,
int* &iptr_MatrM,int* &UdialM)
{
  int k,i,j;
  int NachStroki, KonSroki, NachStrokiOsn, NachStroki1, KonSroki1, TekEl1;
  bool ProdolgatOsn;
  bool *Line;
  int *Line1;
  Line = new bool [n];
  Line1 = new int [n];
  for (j = 0;j<n;j++)
  {
    Line[j] = false;
    Line1[j] = 0;
  }
  for (i = 1;i<n;i++)
  {
    NachStrokiOsn = iptr_MatrM[i];
    for (j = NachStrokiOsn;j < iptr_MatrM[i+1];j++){
      Line[jptr_MatrM[j]] = true;
      Line1[jptr_MatrM[j]] = j;
    }
    ProdolgatOsn = true;
```

```
    while (ProdolgatOsn)
    {
      k = jptr_MatrM[NachStrokiOsn];
      if (k >= i)
        break;
      TekEl1 = UdialM[k];
      aelem_MatrM[NachStrokiOsn] =
  aelem_MatrM[NachStrokiOsn] /aelem_MatrM[TekEl1];
      TekEl1 = NachStrokiOsn;
      NachStrokiOsn++;
      NachStroki = NachStrokiOsn;
      KonSroki = iptr_MatrM[i+1];
      NachStroki1 = UdialM[k];
      KonSroki1 = iptr_MatrM[k+1];
      NachStroki1++;
      if (KonSroki1 <= NachStroki1 || KonSroki <= NachStroki)
        continue;
      for (j = NachStroki1;j < KonSroki1;j++)
      {
        if (Line[jptr_MatrM[j]])
          aelem_MatrM[Line1[jptr_MatrM[j]]] –=
                aelem_MatrM[TekEl1]*aelem_MatrM[j];
      }
    }
    for (j = 0;j<n;j++)
      Line[j] = false;
  }
}
```

Решение СЛАУ с матрицей в модифицированном формате CSR

Далее приведен исходный код программы, реализующей алгоритм 2.3.

```
void Solve_Mb_x(int n, double* &VecB,double* &VecX,
double* &aelem_MatrM,int* &UdialM, int* &jptr_MatrM, int* &iptr_MatrM)
{
  double res1,res2,res3;
  int i,j,k,NachStroki,KonSroki;
  //Прямой ход
  double* VecXv;
  VecXv = new double[n];
  for (i = 0;i < n;i++)
    VecXv[i] = VecX[i];
  for (i = 0;i < n;i++)
  {
    NachStroki = iptr_MatrM[i];
    KonSroki = iptr_MatrM[i+1];
    VecB[i] = VecXv[i];
```

```
    for (j = NachStroki;j<KonSroki;j++)
    {
      k = jptr_MatrM[j];
      if (k >= i)
        continue;
      if (i != 0)
      {
        res1 = VecB[i];
        res2 = VecXv[k];
        res3 = aelem_MatrM[j];
        VecB[i] = res1–res2*res3;
      }
    }
    VecXv[i] = VecB[i];
  }
  //Обратный ход
  for (i = (n–1);i >= 0;i– –)
  {
    NachStroki = iptr_MatrM[i];
    KonSroki = iptr_MatrM[i+1];
    for (j = NachStroki;j<KonSroki;j++)
    {
      k = jptr_MatrM[j];
      if (k <= i)
        continue;
      VecB[i] = VecB[i]–VecXv[k]*aelem_MatrM[j];
    }
    res1 = aelem_MatrM[UdialM[i]];
    VecXv[i] = VecB[i]/res1;
  }
  for (i = 0;i<n;i++)
    VecB[i] = VecXv[i];
  delete [] VecXv;
}
```

Итерационные методы

BiCGStab

Далее приведен исходный код программы, реализующей алгоритм 1.2.

```
Alg_ BiCGStab(double **A, double **ALU,int &m, double &Tol, double Num-
bIter)
{
for (i = 0; i < n; i++)
{
  r[i] = 0;
  for (j = 0; j < n; j++)
  {
```

```
      r[i] += A[i][j]*x0[j]
    }
    r[i] = b[i] – r[i]
    rtilda[i] = r[i];
    p[i] = r[i];
  }
  for (iter = 1; iter < Nitmax; iter++)
  {
    ro = 0.0;
    for (uint i = 0; i < m; i++)
      ro += rtilda[i] * r[i];
    beta = (ro/ro_old)*(alpha/omega);
    for (uint i = 0; i < m; i++)
      p[i] = r[i] + beta * (p[i] – omega* v[i]);
    for (uint i = 0; i<m; i++)
    {
      ptilda[i] = p[i];
      for (uint j = 0; j < i; j++)
        ptilda[i] –= ptilda[j]*ALU[i][j];
    }
    ptilda[m–1] /= ALU[m–1][m–1];
    for (int i = m–2; i >= 0; i– –)
    {
      for (uint j = i+1; j < m; j++)
        ptilda[i] –= ALU[i][j]*ptilda[j];
      ptilda[i] /= ALU[i][i];
    }
    alpha = 0.0;
    for (uint i = 0; i < m; i++)
    {
      v[i] = 0.0;
      for (uint j = 0; j<m; j++)
        v[i] += A[i][j] * ptilda[j];
      alpha += rtilda[i]*v[i];
    }
    alpha = ro/alpha;
    norm_ri = 0.0;
    for (uint i = 0; i < m; i++)
    {
      s[i] = r[i] – alpha * v[i];
      norm_ri += s[i]*s[i];
    }
    if (sqrt(norm_ri/norm_r0) <= Tol)
    {
      for (uint i = 0; i<m; i++)
```

```
      x[i] += alpha*ptilda[i];
    break;
  }
  for (uint i = 0; i < m; i++)
  {
    t[i] = 0.0;
    for (uint j = 0; j<m; j++)
      t[i] += A[i][j] * stilda[j];
  }
  tt = ts = 0.0;
  for (uint i = 0; i < m; i++)
  {
    ts += t[i]*s[i];
    tt += t[i]*t[i];
  }
  omega = ts/tt;
  for (uint i = 0; i < m; i++)
    x[i] +=  alpha * ptilda[i] + omega*stilda[i];
  for (uint i = 0; i < m; i++)
    r[i] = s[i] – omega * t[i];
  norm_ri = 0.0;
  for (uint i = 0; i < m; i++)
    norm_ri += r[i]*r[i];
  if (sqrt(norm_ri/norm_r0) <= Tol)
    break;
  ro_old = ro;
}
```

CGS

Далее приведен исходный код программы, реализующей алгоритм 1.3.

```
Alg_CGS(double **A, double **ALU,int &m, double &Tol)
{
for (int i = 0; i<m; i++)
{
  r[i] = 0.0;
  for (int j = 0; j < m; j++)
    r[i] += A[i][j] * 1.0;
  r[i] = b[i] – r[i];
  norm_r0 += r[i]*r[i];
  r[i] = 0.0;
  for (int j = 0; j < m; j++)
    r[i] += A[i][j] * x[j];
  r[i] = b[i] – r[i];
  rtilda[i] = r[i];
```

```
  p[i] = r[i];
  u[i] = r[i];
}
for (iter = 1; iter < Nitmax; iter++)
{
  ro = 0.0;
  for (int i = 0; i < m; i++)
    ro += rtilda[i] * r[i];
  if (iter > 1)
  {
    beta = (ro/ro_old);
    for (int i = 0; i < m; i++)
    {
      u[i] = r[i] + beta * q[i];
      p[i] = u[i] + beta * (q[i] + beta * p[i]);
    }
  }
  for (int i = 0; i < m; i++)
  {
    ptilda[i] = p[i];
    for (int j = 0; j<i; j++)
      ptilda[i] –= ptilda[j] * ALU_Glob[i][j];
  }
  ptilda[m–1] /= ALU_Glob[m–1][m–1];
  for (int i = m–2; i >= 0; i– –)
  {
    for (int j = i+1; j < m; j++)
      ptilda[i] –= ALU_Glob[i][j]*ptilda[j];
    ptilda[i] /= ALU_Glob[i][i];
  }
  alpha = 0.0;
  for (int i = 0; i < m; i++)
  {
    v[i] = 0.0;
    for (int j = 0; j < m; j++)
      v[i] += A[i][j] * ptilda[j];
    alpha += rtilda[i]*v[i];
  }
  alpha = ro/alpha;
  for (int i = 0; i < m; i++)
  {
    q[i] = u[i] – alpha * v[i];
  }
  for (int i = 0; i < m; i++)
  {
```

```
    utilda[i] = u[i] + q[i];
    for (int j = 0; j < i; j++)
      utilda[i] –= utilda[j]*ALU_Glob[i][j];
  }
  utilda[m–1] /= ALU_Glob[m–1][m–1];
  for (int i = m–2; i >= 0; i– –)
  {
    for (int j = i+1; j < m; j++)
      utilda[i] –= ALU_Glob[i][j]*utilda[j];
    utilda[i] /= ALU_Glob[i][i];
  }
  for (int i = 0; i < m; i++)
    x[i] +=  alpha * utilda[i];
  for (int i = 0; i < m; i++)
  {
    qtilda[i] = 0.0;
    for (int j = 0; j < m; j++)
      qtilda[i] += A[i][j] * utilda[j];
  }
  for (int i = 0; i < m; i++)
    r[i] –= alpha * qtilda[i];
  norm_ri = 0.0;
  for (int i = 0; i < m; i++)
    norm_ri += r[i]*r[i];
  if (sqrt(norm_ri/norm_r0) <= Tol)
  {
    break;
  }
  ro_old = ro;
}
```

Многократное решение СЛАУ
Итерационный метод с предобусловливанием

Далее приведен исходный код программы, реализующей алгоритм 3.1.

```
void LU_Alg(double ** &A,int &m)
{
  for (int i = 1; i < m; i++)
    for (int k = 0; k < i; k++)
    {
      A[i][k] /= A[k][k];
      for (int j = k+1; j < m; j++)
        A[i][j] –= A[i][k]*A[k][j];
    }
}
```

```
void LoadMatrix(int i, double** &A, int &n)
{
  string fn_A = "test";
  fn_A += toString(i);
  fn_A += ".txt";
  ifstream fs_matrixA(fn_A.c_str(), ios::in);
  fs_matrixA >> n;
  A = new double *[n];
  for (int i = 0; i < n; i++)
    A[i] = new double [n];
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      fs_matrixA >> A[i][j];
  fs_matrixA.close();
}
int main(void)
{
  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
  LoadMatrix(1, A, n);
  double **ALU;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  LU_Alg(ALU, n);
  for (int i = 1;i <= 100;i++)
  {
    LoadMatrix(i, A, n);
    Alg_BiCGStab(A, ALU, n, Tol)
  }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

Переформирование матрицы предобусловливания
по порогу числа итераций

Далее приведен исходный код программы, реализующей алгоритм 3.2.

```
void LU_Alg(double ** &A,int &m)
```

```
{
  for (int i = 1; i < m; i++)
    for (int k = 0; k < i; k++)
    {
      A[i][k] /= A[k][k];
      for (int j = k+1; j < m; j++)
        A[i][j] –= A[i][k]*A[k][j];
    }
}
Void LoadMatrix(int i, double** &A, int &n)
{
  string fn_A = "test";
  fn_A += toString(i);
  fn_A += ".txt";
  ifstream fs_matrixA(fn_A.c_str(), ios::in);
  fs_matrixA >> n;
  A = new double *[n];
  for (int i = 0; i < n; i++)
    A[i] = new double [n];
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      fs_matrixA >> A[i][j];
  fs_matrixA.close();
}
int main(void)
{
  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
  LoadMatrix(1, A, n);
  double **ALU;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  LU_Alg(ALU, n);
  int MaxIt = 10;
  int &NumbIter = 1;
  for (int i = 1;i <= 100;i++)
  {
```

```
    if (MaxIt < NumbIter)
    {
      for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
          LU_Alg(ALU, n);
      LU_Alg(ALU, n);
    }
    LoadMatrix(i, A, n);
    Alg_BiCGStab (A, ALU, n, Tol, NumbIter)
  }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

Переформирование матрицы предобусловливания по увеличению среднего арифметического времени решения

Далее приведен исходный код программы, реализующей алгоритм 3.3.

```
int main(void)
{
  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
  LoadMatrix(1, A, n);
  double **ALU;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  DWORD begin, end, time01, SumTime = 0;
  begin = GetTickCount();
  LU_Alg(ALU, n);
  Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
  end = GetTickCount();
  SumTime = (end – begin);
  int &NumbIter = 1;
  for (int i = 2;i <= 100;i++)
  {
    LoadMatrix(i, A, n);
    begin = GetTickCount();
    Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
```

```
    end = GetTickCount();
    time01 = (end – begin);
    if ((SumTime + time01)/i > SumTime / (i–1))
    {
      for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
          LU_Alg(ALU, n);
      begin = GetTickCount();
      LU_Alg(ALU, n);
      end = GetTickCount();
      SumTime += time01;
      time01 = (end – begin);
      SumTime += time01;
    }
    else
      SumTime += time01;
  }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

Переформированием предобусловливателя
при увеличении средней сложности решения

Далее приведен исходный код программы, реализующей алгоритм 3.4.

```
int main(void)
{
  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
  LoadMatrix(1, A, n);
  double **ALU;
  int SumSl = 0;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  LU_Alg(ALU, n);
  Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
  SumSl += n * n * n /6;
  int TekSl = 0;
  int &NumbIter = 1;
```

```
  for (int i = 2;i <= 100;i++)
  {
    LoadMatrix(i, A, n);
    Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
    TekSl = 2*n*n + 2*n + NumbIter *(4*n*n+8*n)
    if ((SumSl + TekSl)/i > SumSl / (i–1))
    {
      for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
          LU_Alg(ALU, n);
      LU_Alg(ALU, n);
      SumSl += TekSl;
      SumSl += n * n * n /6;
    }
    else
      SumSl += TekSl;
  }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

Выбор очередности решения

Далее приведен исходный код программы, реализующей алгоритм 3.5.

```
int main(void)
{
  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
  LoadMatrix(1, A, n);
  double **ALU;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  int &NumbIter = 1;
  // Если выбран прямой порядок
  if (Pr == true)
    for (int i = 1;i <= 100;i++)
    {
      LoadMatrix(i, A, n);
      if (i == 1)
```

```
        LU_Alg(ALU, n);
      Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
    }
  else
    for (int i = 100;i >= 1;i– –)
    {
      LoadMatrix(i, A, n);
      if (i == 100)
        LU_Alg(ALU, n);
      Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
    }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

Выбор матрицы для формирования предобусловливателя

Далее приведен исходный код программы, реализующей алгоритм 3.6.

```
int main(void)
{  int n;
  double Tol = 1.0e–8;
  double tau = 0;
  double **A;
// Номер матрицы для формирования предобусловливателя
  int Form = 50;
  LoadMatrix(Form, A, n);
  double **ALU;
  ALU = new double *[m];
  for (int i = 0; i < m; i++)
    ALU [i] = new double [m];
  for (int i = 0; i < m; i++)
    for (int j = 0; j < m; j++)
      ALU [i][j] = A [i][j];
  LU_Alg(ALU, n);
  int &NumbIter = 1;
  for (int i = 1;i <= 100;i++)
  {
    LoadMatrix(i, A, n);
    Alg_BiCGStab(A, ALU, n, Tol, NumbIter)
  }
  for (int i = 0;i < m;i++)
    delete [] ALU[i];
  delete [] ALU;
}
```

ПРИЛОЖЕНИЕ Б

(справочное)

Расчет сложности алгоритмов

Далее приведен порядок расчета оценки сложности алгоритмов, приведенных в таблице 3.8.

Для расчета сложности использовался алгоритм LU-разложения *ikj*-версии (см. разд. 1.3.2.2).

Оценка по *O*-нотации.

$f_{LU}(N) = \sum_{i=2}^{N}\sum_{k=1}^{i-1}\sum_{j=k+1}^{N}1 = \frac{1}{6}N^3.$

Оценка по арифметической сложности. Далее приведен расчет числа элементарных операций алгоритма по их видам.

Деление:

$$f(N) = \sum_{i=2}^{N}\left(\sum_{k=1}^{i-1}(1)\right) = \sum_{i=1}^{N-1}(i-1+1) = \frac{N^2-N}{2}.$$

Умножение и сложение (вычитания) (в алгоритме их количество одинаково):

$$f(N) = \sum_{i=2}^{N}\left(\sum_{k=1}^{i-1}\left(\sum_{j=k+1}^{N}(1)\right)\right) = \sum_{i=2}^{N}\left(\sum_{k=1}^{i-1}\left(\sum_{j=1}^{N}(1)-\sum_{j=1}^{k}(1)\right)\right) = \sum_{i=2}^{N}\left(\sum_{k=1}^{i-1}(N-k)\right) =$$

$$= \sum_{i=2}^{N}\left(\sum_{k=1}^{i-1}N-\sum_{k=1}^{i-1}k\right) = \sum_{i=2}^{N}\left(N(i-1)-\frac{(i-1)i}{2}\right) = \sum_{i=2}^{N}\left(Ni-N-\frac{i^2-i}{2}\right) =$$

$$= \sum_{i=2}^{N}(Ni)-\sum_{i=2}^{N}(N)-\frac{1}{2}\sum_{i=2}^{N}(i^2)+\frac{1}{2}\sum_{i=2}^{N}(i) = N\sum_{i=1}^{N-1}(i+1)-N(N-1)-\frac{1}{2}\sum_{i=1}^{N-1}(i+1)^2+\frac{1}{2}\sum_{i=1}^{N-1}(i+1) =$$

$$= N\left(\sum_{i=1}^{N-1}(i)+N-1\right)-N^2+N-\frac{1}{2}\left(\sum_{i=1}^{N-1}(i^2+2i+1)\right)+\frac{1}{2}\left(\sum_{i=1}^{N-1}(i)+N-1\right) =$$

$$= N\left(\sum_{i=1}^{N-1}(i)+N-1\right)-N^2+N-\frac{1}{2}\left(\sum_{i=1}^{N-1}i^2+2\sum_{i=1}^{N-1}i+(N-1)\right)+\frac{1}{2}\left(\sum_{i=1}^{N-1}(i)+N-1\right) =$$

$$= N\left(\frac{N(N-1)}{2}+N-1\right)-N^2+N-\frac{1}{2}\left(\frac{(N-1)N(2(N-1)+1)}{6}+2\frac{N(N-1)}{2}+N-1\right)+$$

$$+\frac{1}{2}\left(\frac{N(N-1)}{2}+N-1\right) = N\frac{N^2-N+2N-2}{2}-N^2+N-$$

$$- \frac{1}{2}\left(\frac{(N^2-N)(2N-1)+6N^2-6N+6N-6}{6}\right)+\frac{N^2-N+2N-2}{4} =$$

$$= \frac{N^3+N^2-2N}{2} - N^2 + N - \frac{1}{2}\left(\frac{2N^3-N^2-2N^2+N+6N^2-6}{6}\right)+\frac{N^2+N-2}{4} =$$

$$= \frac{N^3+N^2-2N-2N^2+2N}{2} - \left(\frac{2N^3-N^2-2N^2+N+6N^2-6}{12}\right)+\frac{3N^2+3N-6}{12} =$$

$$= \frac{N^3-N^2}{2} - \left(\frac{2N^3+3N^2+N-6}{12}\right)+\frac{3N^2+3N-6}{12} =$$

$$= \left(\frac{6N^3-6N^2-2N^3-3N^2-N+6+3N^2+3N-6}{12}\right) =$$

$$= \left(\frac{6N^3-2N^3-6N^2-3N^2+3N^2-N+3N-6+6}{12}\right) = \frac{4N^3-6N^2+2N}{12} = \frac{2N^3-3N^2+N}{6}$$

Инкремент и сравнение (в алгоритме их количество одинаково):

$$f(N) = (N-1)+\sum_{i=2}^{N}\left((i-1)+\sum_{k=1}^{i-1}(N-k)\right) = (N-1)+\sum_{i=2}^{N}\left((i-1)+\sum_{k=1}^{i-1}(N)-\sum_{k=1}^{i-1}(k)\right) =$$

$$= (N-1)+\sum_{i=2}^{N}\left((i-1)+N(i-1)-\frac{(i-1)i}{2}\right) = (N-1)+\sum_{i=2}^{N}\left(\frac{2i-2+2Ni-2N-i^2+i}{2}\right) =$$

$$= (N-1)+\sum_{i=2}^{N}\left(\frac{2Ni-2N-i^2+3i-2}{2}\right) = (N-1)+\frac{1}{2}\left(\sum_{i=2}^{N}(2Ni)-\sum_{i=2}^{N}(2N)-\sum_{i=2}^{N}(i^2)+3\sum_{i=2}^{N}(i)-\sum_{i=2}^{N}(2)\right) =$$

$$= (N-1)+\frac{1}{2}\left(2N\sum_{i=1}^{N-1}(i+1)-2N\sum_{i=1}^{N-1}(1)-\sum_{i=1}^{N-1}((i+1)^2)+3\sum_{i=1}^{N-1}(i+1)-2\sum_{i=1}^{N-1}(1)\right) =$$

$$= (N-1)+\frac{1}{2}\left(2N\left(\frac{N-1}{2}N+N-1\right)-2N(N-1)-\sum_{i=1}^{N-1}(i^2+2i+1)+3\frac{N-1}{2}N+3N-3-2N+2\right) =$$

$$= (N-1)+\frac{1}{2}\left(2N\left(\frac{N^2-N+2N-2}{2}\right)-2N^2+2N-\right.$$

$$\left.-\left(\frac{(N-1)N(2N-1)}{6}+2\frac{N-1}{2}N+N-1\right)+\frac{3N^2-3N+2N-2}{2}\right) =$$

$$= (N-1)+\frac{1}{2}\left(\left(\frac{2N^3+2N^2-4N}{2}\right)-2N^2+2N-\left(\frac{(N^2-N)(2N-1)}{6}+\frac{2N^2-2N+2N-2}{2}\right)+\right.$$

$$\left.+\frac{3N^2-N-2}{2}\right) = (N-1)+\frac{1}{2}\left(\frac{2N^3+2N^2-4N-4N^2+4N}{2}-\right.$$

$$-\left(\frac{2N^3-N^2-2N^2+N}{6}+\frac{6N^2-6N+6N-6}{6}\right)+\frac{9N^2-3N-6}{6}=$$

$$=(N-1)+\frac{1}{2}\left(\left(\frac{2N^3-2N^2}{2}\right)-\left(\frac{2N^3-3N^2+N}{6}+\frac{6N^2-6}{6}\right)+\frac{9N^2-3N-6}{6}\right)=$$

$$=(N-1)+\frac{1}{2}\left(\left(\frac{6N^3-6N^2}{6}\right)-\left(\frac{2N^3+3N^2+N-6}{6}\right)+\frac{9N^2-3N-6}{6}\right)=$$

$$=(N-1)+\frac{1}{2}\left(\frac{6N^3-6N^2-2N^3-3N^2-N+6+9N^2-3N-6}{6}\right)=$$

$$=(N-1)+\left(\frac{4N^3-4N}{12}\right)=\frac{4N^3-4N+12N-12}{12}=\frac{4N^3+8N-12}{12}=\frac{N^3+2N-3}{3}$$

Присваивание:

$$f(N)=1+\sum_{i=2}^{N}\left(1+\sum_{k=1}^{i-1}(1+N-k)\right)=1+\sum_{i=2}^{N}\left(1+i-1+Ni-N-\frac{(i-1)i}{2}\right)=$$

$$=1+\sum_{i=2}^{N}\left(i+Ni-N-\frac{i^2-i}{2}\right)=1+\frac{1}{2}\sum_{i=2}^{N}\left(3i+2Ni-2N-i^2\right)=$$

$$=1+\frac{1}{2}\left(3\sum_{i=2}^{N}i+2N\sum_{i=2}^{N}i-2N\sum_{i=2}^{N}1-\sum_{i=2}^{N}i^2\right)=$$

$$=1+\frac{1}{2}\left(3\sum_{i=1}^{N-1}(i+1)+2N\sum_{i=1}^{N-1}(i+1)-2N(N-1)-\sum_{i=1}^{N-1}(i+1)^2\right)=$$

$$=1+\frac{1}{2}\left(3\frac{(N-1)N}{2}+3N-3+2N\frac{(N-1)N}{2}+2N^2-2N-2N(N-1)-\sum_{i=1}^{N-1}\left(i^2+2i+1\right)\right)=$$

$$=1+\frac{1}{2}\left(3\frac{N^2-N}{2}+3N-3+2N\frac{N^2-N}{2}+2N^2-2N-2N^2+2N-\sum_{i=1}^{N-1}i^2-2\sum_{i=1}^{N-1}i-\sum_{i=1}^{N-1}1\right)=$$

$$=1+\frac{1}{2}\left(\frac{3N^2-3N}{2}+3N-3+\frac{2N^3-2N^2}{2}-\frac{(N-1)N(2N-1)}{6}-2\frac{(N-1)N}{2}-N+1\right)=$$

$$=1+\frac{1}{2}\left(\frac{2N^3+N^2+3N-6}{2}-\frac{2N^3-3N^2+N}{6}-\frac{2N^2-2N-2N+2}{2}\right)=$$

$$=1+\frac{1}{2}\left(\frac{2N^3-N^2+7N-8}{2}-\frac{2N^3-3N^2+N}{6}\right)=$$

$$=1+\frac{1}{2}\left(\frac{6N^3-3N^2+21N-24}{6}-\frac{2N^3-3N^2+N}{6}\right)=1+\frac{1}{2}\left(\frac{4N^3+20N-24}{6}\right)=\frac{4N^3+20N-12}{12}=$$

$$=\frac{N^3+5N-3}{3}$$

В таблице А.1 приведено полученное количество элементарных операций алгоритма.

Таблица А.1 – Количество операций алгоритма LU-разложения

Операция	**Количество арифметических операций**
Деление	$\frac{N^2-N}{2}$
Умножение	$\frac{2N^3-3N^2+N}{6}$
Инкремент	$\frac{N^3+2N-3}{2}$
Сложение (вычитание)	$\frac{2N^3-3N^2+N}{6}$
Сравнение	$\frac{N^3+2N-3}{2}$
Операции присваивание	$\frac{N^3+5N-3}{3}$
Итого	$(10N^3-3N^2+16N-18)/6$

При расчете сложности стабилизированного метода бисопряженных градиентов (BiCGStab) использовался алгоритм 1.2. Учитывалось наличие двух возможных условий выхода из итерационного процесса (строки 17 и 24).

Оценка по *O*-нотации.

BiCGStab условие 1: $f(N, N_{it}) = 4N^2+6N+4(N_{it}-1)(N^2+2N)$.

BiCGStab условие 2: $f(N, N_{it}) = 2N^2+2N+4N_{it}(N^2+2N)$.

Оценка по арифметической сложности. В таблице А.2 приведено полученное количество элементарных операций алгоритма.

Таблица А.2 – Количество операций алгоритма BiCGStab

Операция	**Условие 1**	**Условие 2**
Деление	$N_{it} \cdot (4 + N) + (N_{it} - 1) \cdot (N + 2)$	$N_{it} \cdot (2N + 6)$
Умножение	$N^2 + N_{it} \cdot (N + 1 + 2N + N^2 - N + N^2 + N + 2N) + N + (N_{it} - 1) \cdot (N^2 - N + N^2 + 6N) = N^2 + N + N_{it} \cdot (2N^2 + 5N + 1) + (N_{it} - 1) \cdot (2N^2 + 5N)$	$N^2 + N_{it} \cdot (N + 1 + 2N + N^2 - N + N^2 + N + 2N + N^2 - N + N^2 + 6N) = N^2 + N_{it} \cdot (4N^2 + 10N + 1)$
Инкремент	$N^2 + N + N_{it} + N_{it} \cdot (2N + N^2 + N - 1 + N^2 + 2N) + N + (N_{it} - 1) \cdot (N^2 + N - 1 + N + N^2 + 4N) = N^2 + 2N + N_{it} \cdot (2N^2 + 5N) + (N_{it} - 1) \cdot (2N^2 + 6N - 1)$	$N^2 + N + N_{it} + N_{it} \cdot (2N + N^2 + N - 1 + N^2 + 2N + N^2 + N - 1 + N + N^2 + 4N) = N^2 + N + N_{it} \cdot (4N^2 + 11N - 1)$
Сложение (вычитание)	$N^2 + N + N_{it} \cdot (N + 2N + N^2 - N + N^2 + N + 2N) + N + (N_{it} - 1) \cdot (N^2 - N + N^2 + 2N + 4N) = N^2 + 2N + N_{it} \cdot (2N^2 + 5N) + (N_{it} - 1) \cdot (2N^2 + 5N)$	$N^2 + N + N_{it} \cdot (N + 2N + N^2 - N + N^2 + N + 2N + N^2 - N + N^2 + 2N + 4N) = N^2 + N + N_{it} \cdot (4N^2 + 10N)$
Сравнение	$N^2 + N + N_{it} + N_{it} \cdot (2N^2 + 5N - 1 + 1) + N + (N_{it} - 1) \cdot (2N^2 + 2N - 1 + 4N + 1) = N^2 + 2N + N_{it} \cdot (2N^2 + 5N + 1) + (N_{it} - 1) \cdot (2N^2 + 6N)$	$N^2 + N + N_{it} + N_{it} \cdot (2N^2 + 5N - 1 + 1 + 2N^2 + 2N - 1 + 4N + 1) = N^2 + N + N_{it} \cdot (4N^2 + 11N + 1)$
Операции присваивание	$N^2 + 5N + 2 + N_{it} \cdot (N + 2 + 2 + N + 2N^2 + 3N + 1 + 3N + 2 + 2N + 3) + 1 + N + (N_{it} - 1) \cdot (N^2 + 3N + 1 + 1 + 2N + N^2 + 3 + 2N + 1 + N + N + 1 + 2 + N + 1) = N^2 + 6N + 3 + N_{it} \cdot (2N^2 + 10N + 10) + (N_{it} - 1) \cdot (2N^2 + 10N + 10)$	$N^2 + 5N + 2 + N_{it} \cdot (N + 2 + 2 + N + 2N^2 + 3N + 1 + 3N + 2 + 2N + 3 + N^2 + 3N + 1 + 1 + 2N + N^2 + 3 + 2N + 1 + N + N + 1 + 2 + N + 1) = N^2 + 5N + 2 + N_{it} \cdot (4N^2 + 20N + 20)$
Итого	$5N^2 + 13N + 3 + N_{it} (10N^2 + 31N + 16) + (N_{it} - 1)(10N^2 + 33N + 11)$	$5N^2 + 8N + 2 + N_{it}(20N^2 + 64N + 37)$

При расчете сложности квадратичного метода сопряженных градиентов (CGS) использовался алгоритм 1.3.

Оценка по *O*-нотации.

$f(N, N_{it}) = 2N^2 + N_{it}(8N^2 + 5N)$.

Оценка по арифметической сложности. В таблице А.3 приведено полученное количество элементарных операций алгоритма.

Таблица А.3 – Количество операций алгоритма CGS

Операция	**Количество арифметических операций**
Деление	$N_{it} \cdot (2N + 2) + N_{it} - 1$
Умножение	$2N^2 + N + N_{it} \cdot (N + N^2 - N + N^2 + N + N + N^2 - N + 2N + N^2 + N) + (N_{it} - 1) \cdot (3N) = 2N^2 + N + N_{it} \cdot (4N^2 + 3N) + (N_{it} - 1) \cdot (3N)$
Инкремент	$2N^2 + N + N_{it} \cdot (1+N+N+ N^2 + N - 1 + N + N^2 + N^2 + N - 1 + 3N + N^2 + N) + (N_{it} - 1) \cdot (N) = 2N^2 + N + N_{it} \cdot (4N^2 + 9N - 1) + (N_{it} - 1) \cdot (N)$
Сложение (вычитание)	$2N^2 + 3N + N_{it} \cdot (N + N^2 - N + N + N^2 + N + N^2 - N + 2N + N^2 + N) + (N_{it} - 1) \cdot (3N) = 2N^2 + 3N + N_{it} \cdot (4N + 4N^2) + (N_{it} - 1) \cdot (3N)$
Сравнение	$2N^2 + N + N_{it} \cdot (1 + N + N^2 + N - 1 + N^2 + N + N + N^2 + N - 1 + 3N + N^2 + 1 + N) + (N_{it} - 1) \cdot (1 + N) = 2N^2 + N + N_{it} \cdot (9N + 4N^2) + (N_{it} - 1) \cdot (1 + N)$
Операции присваивание	$2N^2 + 10N + 1 + N_{it} \cdot (N + 2 + N^2 + 3N + 1 + 2 + 3N + N^2 + 2 + N + N^2 + 3N + 1 + 3 + 4N + N^2 + 3 + N) + (N_{it} - 1) \cdot (2 + 2N) = 2N^2 + 10N + 1 + N_{it} \cdot (4N^2 + 16N + 14) + (N_{it} - 1) \cdot (2N + 2)$
Итого	$10N^2 + 16N + 1 + N_{it} (20N^2 + 43N + 15) + (N_{it} - 1)(10N + 4)$

ПРИЛОЖЕНИЕ В

(справочное)

Копии свидетельств о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2013619615

TALGAT 2012

Правообладатель: *Общество с ограниченной ответственностью «ТУСУР - Модальные Технологии» (RU)*

Авторы: *см. на обороте*

Заявка № 2013617773
Дата поступления 29 августа 2013 г.
Дата государственной регистрации в Реестре программ для ЭВМ *11 октября 2013 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

Авторы: *Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Кукенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Аширбакиев Ренат Ихсанович (RU), Лежнин Евгений Владимирович (RU), Салов Василий Константинович (RU), Лежнин Егор Владимирович (RU), Орлов Павел Евгеньевич (RU), Калимуллин Илья Фидаильевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Газизов Руслан Рифатович (RU), Ахунов Роман Раисович (RU)*

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015614365

TALGAT 2013

Правообладатель: ***Общество с ограниченной ответственностью «ТАЛГАТ» (RU)***

Авторы: ***см. на обороте***

Заявка № **2015611288**

Дата поступления **03 марта 2015 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ ***16 апреля 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Авторы: ***Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Куксенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Газизов Руслан Рифатович (RU), Лежнин Евгений Владимирович (RU), Салов Василий Константинович (RU), Лежнин Егор Владимирович (RU), Орлов Павел Евгеньевич (RU), Калимулин Илья Фидаильевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Ахунов Роман Раисович (RU), Новикова Елена Александровна (RU)***

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015617550

TALGAT 2014

Правообладатель: ***Общество с ограниченной ответственностью «Твердь» (RU)***

Авторы: ***см. на обороте***

Заявка № **2015614488**

Дата поступления **27 мая 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***14 июля 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Авторы: ***Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Куксенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Газизов Руслан Рифатович (RU), Лежнин Евгений Владимирович (RU), Салов Василий Константинович (RU), Лежнин Егор Владимирович (RU), Орлов Павел Евгеньевич (RU), Калимулин Илья Фидаильевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Ахунов Роман Раисович (RU), Новикова Елена Александровна (RU), Газизов Рустам Рифатович (RU), Веселовский Александр Владимирович (RU)***

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015660487

TALGAT 2015

Правообладатель: ***Общество с ограниченной ответственностью «Твердь» (RU)***

Авторы: ***см. на обороте***

Заявка № **2015617580**

Дата поступления **17 августа 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***01 октября 2015 г.***

Заместитель руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Авторы: ***Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Куксенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Газизов Руслан Рифатович (RU), Лежнин Евгений Владимирович (RU), Салов Василий Константинович (RU), Лежнин Егор Владимирович (RU), Орлов Павел Евгеньевич (RU), Калимулин Илья Фидаильевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Ахунов Роман Рашович (RU), Газизов Рустам Рифатович (RU), Веселовский Александр Владимирович (RU)***

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016662520

TALGAT 2016

Правообладатель: ***Общество с ограниченной ответственностью «ТАЛГАТ» (RU)***

Авторы: ***см. на обороте***

Заявка № **2016619296**
Дата поступления **01 сентября 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***14 ноября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

Авторы: ***Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Куксенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Газизов Руслан Рифатович (RU), Салов Василий Константинович (RU), Лежнин Евгений Владимирович (RU), Орлов Павел Евгеньевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Ахунов Роман Раисович (RU), Газизов Рустам Рифатович (RU), Газизов Александр Тальгатович (RU), Веселовский Александр Владимирович (KZ), Квасников Алексей Андреевич (RU), Носов Александр Вячеславович (KZ), Белоусов Антон Олегович (KZ), Буичкин Евгений Николаевич (KZ), Лесков Андрей Николаевич (RU), Демаков Александр Витальевич (RU), Лемешко Ксения Андреевна (RU), Собко Александр Александрович (KZ), Осинцев Артем Викторович (KZ), Калимулин Илья Фидаильевич (RU)***

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018611481

TALGAT 2017

Правообладатель: ***Общество с ограниченной ответственностью «ТАЛГАТ» (RU)***

Авторы: ***см. на обороте***

Заявка № **2017663209**
Дата поступления **13 декабря 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***02 февраля 2018 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

Авторы: ***Газизов Тальгат Рашитович (RU), Мелкозеров Александр Олегович (RU), Газизов Тимур Тальгатович (RU), Куксенко Сергей Петрович (RU), Заболоцкий Александр Михайлович (RU), Газизов Руслан Рифатович (RU), Бусыгина Анна Владимировна (RU), Лежнин Евгений Владимирович (RU), Орлов Павел Евгеньевич (RU), Суровцев Роман Сергеевич (RU), Комнатнов Максим Евгеньевич (RU), Ахунов Роман Раисович (RU), Газизов Рустам Рифатович (RU), Газизов Александр Тальгатович (RU), Хажибеков Роман Русланович (KZ), Квасников Алексей Андреевич (RU), Носов Александр Вячеславович (KZ), Белоусов Антон Олегович (KZ), Тернов Станислав Анатольевич (KZ), Сагиева Индира Ериковна (KZ), Демаков Александр Витальевич (RU), Осинцев Артем Викторович (KZ), Собко Александр Александрович (KZ)***

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616322

Многократное решение СЛАУ итерационным методом BiCGstab с использованием при решении текущей системы вектора решения предыдущей.

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612894**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***05 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616321

Решение СЛАУ с матрицей, полученной с помощью ILU(0)-разложения и хранимой в модифицированном разреженном строчном формате.

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612893**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***05 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616320

Многократное решение СЛАУ итерационным методом BiCGStab с использованием матрицы предобусловливания, полученной при решении первой системы.

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612892**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***05 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616205

ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (ТУСУР) (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612783**
Дата поступления **09 апреля 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***03 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616124

ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате, с использованием вспомогательного вектора

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612895**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***01 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015615837

Многократное решение СЛАУ итерационным методом BiCGStab с использованием переформирования матрицы предобусловливания по заданному порогу числа итераций

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612782**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***26 мая 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015615835

ILU(0)-разложение матрицы, хранимой в разреженном строчном формате, с последовательным перебором элементов

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612781**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***26 мая 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015615730

Организация хранения плотной матрицы в модифицированном строчном разреженном формате после предфильтрации, основанной на максимальном элементе матрицы

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612891**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***22 мая 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015615729

Многократное решение СЛАУ итерационным методом BiCGStab с переформированием матрицы предобусловливания при превышении среднего времени решения одной системы

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» (RU)***

Авторы: ***Ахунов Роман Раисович (RU), Куксенко Сергей Петрович (RU), Газизов Тальгат Рашитович (RU)***

Заявка № **2015612890**

Дата поступления **09 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***22 мая 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

ПРИЛОЖЕНИЕ Г
(справочное)

Копии актов о внедрении (использовании) результатов диссертационной работы

УТВЕРЖДАЮ
Директор НИИ ПММ ТГУ,
д.ф-м.н., профессор
_____ А.А. Глазунов
«_09_» _ноября_ 2015 г.

АКТ
о внедрении (использовании) результатов диссертационной работы
Ахунова Романа Раисовича

Комиссия в составе:
зав. лаб. НИИ ПММ ТГУ, д.ф-м.н. С.В. Пономарёва,
учёного секретаря НИИ ПММ ТГУ, к.ф-м.н. И.В. Ерёмина
составила настоящий акт, подтверждающий факт использования в ОАО «ИСС» следующих результатов диссертационной работы Р.Р. Ахунова.

1. Результаты исследований использования разреженного строчного формата при решении систем линейных алгебраических уравнений (СЛАУ) методом BiCGStab с предобусловливанием с помощью ILU(0)-разложения в ходе вычисления матрицы ёмкостей произвольной системы проводников и диэлектриков методом моментов.
2. Усовершенствованный алгоритм ILU(0)-разложения с использованием разреженного строчного формата при решении СЛАУ с плотной матрицей итерационным методом с предобусловливанием.

Указанные результаты представлены в технических отчётах по опытно-конструкторской работе «Разработка комплекса программных и технических средств проектирования, изготовления и испытаний унифицированного ряда электронных модулей на основе технологии «система-на-кристалле» для систем управления и электропитания космических аппаратов (КА) связи, навигации и дистанционного зондирования Земли с длительным сроком активного существования» НИ ТГУ для ОАО «ИСС» в рамках реализации Постановления №218 Правительства РФ, тема «УЭМ», хоздоговор №2148 от 05.07.2010. В 2010–2012 гг. Р.Р. Ахунова являлся исполнителем четырёх этапов этой работы.

Зав. лаб. НИИ ПММ ТГУ д.ф-м.н.
_____ С.В. Пономарёв

Учёный секретарь НИИ ПММ ТГУ, к.ф-м.н.
_____ И.В. Ерёмин

Акционерное общество
«ИНФОРМАЦИОННЫЕ СПУТНИКОВЫЕ СИСТЕМЫ»
имени академика М.Ф. Решетнёва»

РЕШЕТНЕВ
АО «ИСС»

ул. Ленина, д. 52, г. Железногорск, ЗАТО Железногорск, Красноярский край, Российская Федерация, 662972
Тел. (3919) 76-40-02, 72-24-39, Факс (3919) 72-26-35, 75-61-46, e-mail: office@iss-reshetnev.ru, http: //www.iss-reshetnev.ru
ОГРН 1082452000290, ИНН 2452034898

УТВЕРЖДАЮ
Заместитель генерального конструктора по электрическому проектированию и системам управления КА акционерного общества «Информационные спутниковые систем» имени академика М.Ф. Решетнева» (АО «ИСС»), председатель секции №2 НТС, к.т.н., доцент

_____С.Г. Кочура

«___»_____2015

АКТ

о внедрении (использовании) результатов диссертационной работы
Ахунова Романа Раисовича

Комиссия в составе:
главного конструктора проектирования и испытаний РЭА ОАО «ИСС» В.Н. Школьного,
начальника отдела конструирования бортовой РЭА ОАО «ИСС», к.т.н. С.Б. Сунцова,
начальника группы ОАО «ИСС», к.ф.-м.н. А.А. Хвалько
составила настоящий акт, подтверждающий факт использования в ОАО «ИСС» следующих результатов диссертационной работы Ахунова Р.Р.:

1. Обзор методов решения систем линейных алгебраических уравнений (СЛАУ).
2. Алгоритмы, уменьшающие вычислительные затраты на решение СЛАУ при моделировании электромагнитной совместимости элементов системы автономной навигации.

Указанные результаты представлены в технических отчетах по опытно-конструкторской работе «Разработка принципов построения и элементов системы автономной навигации с применением отечественной специализированной элементной базы на основе наногетероструктурной технологии для космических аппаратов всех типов» ТУСУРа для АО «ИСС» в рамках реализации Постановления №218 Правительства РФ, тема «САН», хоздоговор 96/12 от 16.11.2012. В 2013-2015 гг. Ахунова Р.Р. являлся соисполнителем этапов этой работы.

Главный конструктор проектирования и испытаний РЭА АО «ИСС»
В.Н. Школьный

Начальник отдела конструирования бортовой РЭА АО «ИСС», к.т.н.
С.Б. Сунцов

Начальник группы ОАО «ИСС», к.ф.-м.н.
А.А. Хвалько

УТВЕРЖДАЮ
Проректор по НРиИ ТУСУР,
д.т.н., профессор
_____ Р.В. Мещеряков
«_____» _____2015 г.

АКТ

о внедрении (использовании) результатов диссертационной работы

Ахунова Романа Раисовича

Настоящим подтверждается факт использования в проекте №14-07-31267 мол_а РФФИ «Выявление, исследование и реализация новых возможностей уменьшения времени многократного решения СЛАУ», 2014–2015 гг. следующих результатов диссертационной работы работы Р.Р. Ахунова:

1. Аналитические оценки ускорения многократного решения СЛАУ итерационным методом.
2. Алгоритм переформирования матрицы предобусловливания при многократном решении СЛАУ по значению порога числа итераций.
3. Алгоритмы многократного решения СЛАУ итерационным методом с адаптивным переформированием предобусловливателя на основании оценок средних арифметических значений времени и сложности.
4. Комплекс программ для многократного решения СЛАУ итерационными методами с адаптивным переформированием предобусловливателя.

Указанные результаты представлены в отчётах по вышеуказанному проекту. Они позволили выполнить план работ, предусмотренных проектом. Р.Р. Ахунов являлся исполнителем всех этапов проекта.

Руководитель проекта, к.т.н.
_____ С.П. Куксенко

УТВЕРЖДАЮ
Проректор по НРиИ
_____ д.т.н., проф. Мещеряков Р.В.
18.12.2015

АКТ внедрения №2/2015 от 18.12.2015

Настоящим актом подтверждается факт внедрения (использования) в ОКР по теме «САН», хоздоговор 96/12 от 16.11.2012, выполняемой в ходе реализации Постановления 218 Правительства РФ, следующих результатов интеллектуальной деятельности:

1. Свидетельство о государственной регистрации программы для ЭВМ №2015611088. PCBsensor. Орлов П.Е., Газизов Т.Р., Заболоцкий А.М. – Заявка №2014661973. Дата поступления 25 ноября 2014 г. Зарегистрировано в Реестре программ для ЭВМ 23 января 2015 г.
2. Свидетельство о государственной регистрации программы для ЭВМ №2015614365. TALGAT 2013. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Р.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Новикова Е.А. Заявка №2015611288. Дата поступления 3 марта 2015 г. Зарегистрировано в Реестре программ для ЭВМ 16 апреля 2015 г.
3. Свидетельство о государственной регистрации программы для ЭВМ №2015615835. ILU(0)-разложение матрицы, хранимой в разреженном строчном формате, с последовательным перебором элементов. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612781. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 май 2015 г.
4. Свидетельство о государственной регистрации программы для ЭВМ №2015616205. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612783. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 03 июня 2015 г.
5. Свидетельство о государственной регистрации программы для ЭВМ №2015616124. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате, с использованием вспомогательного вектора. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612895. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 01 июня 2015 г.
6. Свидетельство о государственной регистрации программы для ЭВМ №2015615730. Организация хранения плотной матрицы в модифицированном строчном разреженном формате после предфильтрации, основанной на максимальном элементе матрицы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612891. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.
7. Свидетельство о государственной регистрации программы для ЭВМ №2015615837. Многократное решение СЛАУ итерационным методом BiCGStab с использованием переформирования матрицы предобусловливания по заданному порогу числа итераций. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов

Т.Р. Заявка №2015612782. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.

8. Свидетельство о государственной регистрации программы для ЭВМ №2015615729. Многократное решение СЛАУ итерационным методом BiCGStab с переформированием матрицы предобусловливания при превышении среднего времени решения одной системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612890. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.
9. Свидетельство о государственной регистрации программы для ЭВМ №2015615793. Многократное решение систем линейных алгебраических уравнений. Авторы: Суровцев Р.С., Куксенко С.П., Газизов Т.Р. Заявка №2015612566. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 25 мая 2015 г.
10. Свидетельство о государственной регистрации программы для ЭВМ №2015615987. Вычисление дифференциальной ёмкости, индуктивности и сопротивления для связанных обращено-подвешенных полосковых линий. Авторы: Заболоцкий А.М., Газизов Т.Р., Калимулин И.Ф. Заявка №2015612593. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 28 мая 2015 г.
11. Свидетельство о государственной регистрации программы для ЭВМ №2015616060. Вычисление дифференциальной ёмкости, индуктивности и сопротивления для связанных подвешенных полосковых линий. Авторы: Заболоцкий А.М., Газизов Т.Р., Калимулин И.Ф. Заявка №2015612715. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 29 мая 2015 г.
12. Свидетельство о государственной регистрации программы для ЭВМ №2015615988. Многорезонансная модель индуктивности Murata LQW2BHN33NJ03 для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612601. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 29 мая 2015 г.
13. Свидетельство о государственной регистрации программы для ЭВМ №2015616087. Многорезонансная модель конденсатора Murata GRM21BR71H224K для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612625. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 29 мая 2015 г.
14. Свидетельство о государственной регистрации программы для ЭВМ №2015615845. Модель LC-фильтра на базе компонентов Murata GRM21BR71H224K и LQW2BHN33NJ03 для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612624. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.
15. Свидетельство о государственной регистрации программы для ЭВМ №2015615589. Оптимизация параметров измерительных плат набора TRL Calkit for UTF3680 с учётом технологии производства. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612554. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 21 мая 2015 г.
16. Свидетельство о государственной регистрации программы для ЭВМ №2015615880. Вычисление напряжений и токов вдоль двухпроводной линии

передачи на основе квазистатического анализа / Р.Р. Газизов, А.М. Заболоцкий – Заявка №2015612556. Дата поступления 2 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.

17. Свидетельство о государственной регистрации программы для ЭВМ №2015615794. Вычисление временного отклика модального фильтра с лицевой связью / А.М. Заболоцкий, А.Т. Газизов, Т.Р. Газизов – Заявка №2015612568. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 25 мая 2015 г.
18. Свидетельство о государственной регистрации программы для ЭВМ №2015616321. Решение СЛАУ с матрицей, полученной с помощью ILU(0)-разложения и хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612893. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
19. Свидетельство о государственной регистрации программы для ЭВМ №2015616320. Многократное решение СЛАУ итерационным методом BiCGStab с использованием матрицы предобусловливания, полученной при решении первой системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612892. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
20. Свидетельство о государственной регистрации программы для ЭВМ №2015616322. Многократное решение СЛАУ итерационным методом BiCGstab с использованием при решении текущей системы вектора решения предыдущей. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612894. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
21. Свидетельство о государственной регистрации программы для ЭВМ №2015617207. Временной отклик последовательного интерфейса процессора. Авторы: Зырянова Н.А., Суровцев Р.С., Комнатнов М.Е, Газизов Т.Р. Заявка №2015614262. Дата поступления 22 мая 2015 г. Зарегистрировано в Реестре программ для ЭВМ 03 июля 2015 г.
22. Свидетельство о государственной регистрации программы для ЭВМ №2015617542. Вычисление дифференциальной ёмкости, индуктивности и сопротивления для связанных обращенных полосковых линий. Авторы: Заболоцкий А.М., Газизов Т.Р., Калимулин И.Ф. Заявка №2015612552. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 14 июля 2015 г.
23. Свидетельство о государственной регистрации программы для ЭВМ №2015618664. Вычисление комлексной емкостной матрицы многопроводной микрополосковой линии на подложке из FR-4 в диапазоне частот с помощью блочного LU-разложения. Авторы: Суровцев Р.С., Куксенко С.П., Газизов Т.Р. Заявка №2015612938. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 13 августа 2015 г.
24. Свидетельство о государственной регистрации программы для ЭВМ №2015617550. TALGAT 2014. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Р.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Новикова Е.А., Газизов Руст.Р., Веселовский А.В.

Заявка №2015614488. Дата поступления 27 мая 2015 г. Зарегистрировано в Реестре программ для ЭВМ 17 июля 2015 г.

25. Свидетельство о государственной регистрации программы для ЭВМ №2015660487. TALGAT 2015. Авторы: Газизов Т.Р., Мелкозеров А.О., Газизов Т.Т., Куксенко С.П., Заболоцкий А.М., Газизов Русл.Р., Лежнин Ев.В., Салов В.К., Лежнин Ег.В., Орлов П.Е., Калимулин И.Ф., Суровцев Р.С., Комнатнов М.Е., Ахунов Р.Р., Газизов Руст.Р., Веселовский А.В. Заявка №2015617580. Дата поступления 17 августа 2015 г. Зарегистрировано в Реестре программ для ЭВМ 1 октября 2015 г.
26. Свидетельство о государственной регистрации программы для ЭВМ №2015615988. Многорезонансная модель индуктивности Murata LQW2BHN33NJ03 для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612601. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 29 мая 2015 г.
27. Свидетельство о государственной регистрации программы для ЭВМ №2015616087. Многорезонансная модель конденсатора Murata GRM21BR71H224K для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612625. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 29 мая 2015 г.
28. Свидетельство о государственной регистрации программы для ЭВМ №2015615845. Модель LC-фильтра на базе компонентов Murata GRM21BR71H224K и LQW2BHN33NJ03 для диапазона частот 0,01–40 ГГц. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612624. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.
29. Свидетельство о государственной регистрации программы для ЭВМ №2015615589. Оптимизация параметров измерительных плат набора TRL Calkit for UTF3680 с учётом технологии производства. Авторы: Калимулин И.Ф., Заболоцкий А.М. Заявка №2015612554. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 21 мая 2015 г.
30. Свидетельство о государственной регистрации программы для ЭВМ №2015615880. Вычисление напряжений и токов вдоль двухпроводной линии передачи на основе квазистатического анализа / Р.Р. Газизов, А.М. Заболоцкий – Заявка №2015612556. Дата поступления 2 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.
31. Свидетельство о государственной регистрации программы для ЭВМ №2015617207. Временной отклик последовательного интерфейса процессора. Авторы: Зырянова Н.А., Суровцев Р.С., Комнатнов М.Е, Газизов Т.Р. Заявка №2015614262. Дата поступления 22 мая 2015 г. Зарегистрировано в Реестре программ для ЭВМ 03 июля 2015 г.

Указанные программы использовались для моделирования и обеспечения электромагнитной совместимости цепей электрических принципиальных схем, элементов печатных плат и конструкции блока системы автономной навигации космического аппарата. Результаты внедрения (использования) представлены в технических отчетах по теме «САН».

Руководитель темы «САН» директор НИИ космических технологий ТУСУРа

д.т.н. Ю.А. Шиняков

«УТВЕРЖДАЮ»
Проректор по учебной работе ТУСУР
к.ф.-м.н., доцент
_____ Боков Л.А.
_____ 2015 г.

АКТ
внедрения в учебный процесс результатов диссертационной работы
Ахунова Романа Раисовича

Мы, нижеподписавшиеся, заведующий кафедрой телевидения и управления (ТУ) ТУСУР, д.т.н., с.н.с. Газизов Т.Р., заместитель заведующего кафедрой ТУ по учебной работе, к.т.н. Булдаков А.Н. настоящим актом подтверждаем факт внедрения результатов диссертационной работы Ахунова Р.Р. в учебный процесс кафедры ТУ.

В ходе выполнении диссертационной работы Ахуновым Р.Р. разработаны алгоритмы усовершенствования ILU(0)-разложения и многократного решения СЛАУ итерационным методом с адаптивным переформированием предобусловливателя, комплекс программ, а также проведены исследования по их апробации при многовариантном анализе полосковых структур методом моментов. Разработанные алгоритмы позволяют уменьшить вычислительные затраты на анализ электромагнитной совместимости методом моментов.

Данные результаты применяются для моделирования различных реальных структур проводников и диэлектриков в ходе практических и лабораторных работ по дисциплинам «Математическое моделирование процессов электромагнитной совместимости радиоэлектронной аппаратуры» (бакалавриат), «Теория электромагнитной совместимости радиоэлектронных средств и систем» (магистранты), «Вычислительная электромагнитная совместимость» (магистранты), а также в рамках группового проектного обучения проекта ТУ-1502 «Вычислительная электромагнитная совместимость».

Указанные результаты показали свою работоспособность и полезность при практическом освоении сложных вопросов, связанных с обеспечением электромагнитной совместимости радиоэлектронной аппаратуры.

Заведующий каф. ТУ _____ Т.Р. Газизов

Заместитель заведующего каф. ТУ по учебной работе _____ А.Н. Булдаков

УТВЕРЖДАЮ
Проректор по НРиИ
_____ д.т.н., проф. Мещеряков Р.В.
21.12.2015

АКТ внедрения №3/2015 от 21.12.2015

Настоящим актом подтверждается факт внедрения (использования) в проекте №14-07-31267 мол_а РФФИ «Выявление, исследование и реализация новых возможностей уменьшения времени многократного решения СЛАУ», 2014–2015 гг., следующих результатов интеллектуальной деятельности:

1. Свидетельство о государственной регистрации программы для ЭВМ №2015615835. ILU(0)-разложение матрицы, хранимой в разреженном строчном формате, с последовательным перебором элементов. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612781. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 май 2015 г.
2. Свидетельство о государственной регистрации программы для ЭВМ №2015616205. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612783. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 03 июня 2015 г.
3. Свидетельство о государственной регистрации программы для ЭВМ №2015616124. ILU(0)-разложение матрицы, хранимой в модифицированном разреженном строчном формате, с использованием вспомогательного вектора. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612895. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 01 июня 2015 г.
4. Свидетельство о государственной регистрации программы для ЭВМ №2015615730. Организация хранения плотной матрицы в модифицированном строчном разреженном формате после предфильтрации, основанной на максимальном элементе матрицы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612891. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.
5. Свидетельство о государственной регистрации программы для ЭВМ №2015615837. Многократное решение СЛАУ итерационным методом BiCGStab с использованием переформирования матрицы предобусловливания по заданному порогу числа итераций. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612782. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 26 мая 2015 г.
6. Свидетельство о государственной регистрации программы для ЭВМ №2015615729. Многократное решение СЛАУ итерационным методом BiCGStab с переформированием матрицы предобусловливания при превышении среднего времени решения одной системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612890. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 22 мая 2015 г.
7. Свидетельство о государственной регистрации программы для ЭВМ №2015615793. Многократное решение систем линейных алгебраических уравнений. Авторы: Суровцев Р.С., Куксенко С.П., Газизов Т.Р. Заявка №2015612566. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 25 мая 2015 г.
8. Свидетельство о государственной регистрации программы для ЭВМ №2015616321. Решение СЛАУ с матрицей, полученной с помощью ILU(0)-разложения и хранимой в модифицированном разреженном строчном формате. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612893. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
9. Свидетельство о государственной регистрации программы для ЭВМ №2015616320. Многократное решение СЛАУ итерационным методом BiCGStab с использованием матрицы предобусловливания, полученной при решении первой системы. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612892. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
10. Свидетельство о государственной регистрации программы для ЭВМ №2015616322. Многократное решение СЛАУ итерационным методом BiCGstab с использованием при решении текущей системы вектора решения предыдущей. Авторы: Ахунов Р.Р., Куксенко С.П., Газизов Т.Р. Заявка №2015612894. Дата поступления 09 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 05 июня 2015 г.
11. Свидетельство о государственной регистрации программы для ЭВМ №2015618664. Вычисление комплексной емкостной матрицы многопроводной микрополосковой линии на подложке из FR-4 в диапазоне частот с помощью блочного LU-разложения. Авторы: Суровцев Р.С., Куксенко С.П., Газизов Т.Р. Заявка №2015612938. Дата поступления 02 апреля 2015 г. Зарегистрировано в Реестре программ для ЭВМ 13 августа 2015 г.

Указанные программы использовались для моделирования по вышеуказанному проекту. Результаты внедрения (использования) представлены в отчетах. Они позволили выполнить план работ, предусмотренных проектом.

Руководитель проекта РФФИ, к.т.н. С.П. Куксенко

«УТВЕРЖДАЮ»

Проректор по научной работе НИ ТГУ

д.ф.-м.н., проф. И.В.Ивонин

«1» 06 2017 г.

АКТ

о внедрении (использовании) в учебный процесс результатов диссертационной работы
Ахунова Романа Раисовича

Настоящим актом подтверждается внедрение в учебный процесс федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский Томский государственный университет» результатов диссертационной работы Ахунова Р.Р.

Материалы диссертационной работы использованы при преподавании дисциплины «Электромагнитная совместимость» для магистрантов физико-технического факультета по программе «Проектирование и конструирование промышленных космических систем» для предприятия «Газпром космические системы» (г. Королев) в весеннем семестре 2017 г.

Полученные в диссертационной работе результаты совершенствования анализа полосковых структур методом моментов, в части решения систем линейных алгебраических уравнений итерационными методами, позволили познакомить магистрантов с новыми алгоритмическим способами уменьшения вычислительных затрат на моделирование задач электромагнитной совместимости бортовой радиоэлектронной аппаратуры.

Декан ФТФ ТГУ _____ д.ф.-м.н., проф. Шрагер Э.Р.