ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ РОССИЙСКОЙ АКАДЕМИИ НАУК

На правах рукописи

Сушникова Дарья Алексеевна

УДК 519.6

Методы факторизации и решения линейных систем с блочно-малоранговыми матрицами

05.13.18 — Математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ

на соискание учёной степени

кандидата физико-математических наук

Научный руководитель

д.ф.-м.н. Оселедец И. В.

Москва 2017

Содержание

Введение

i.1. Актуальность темы исследования

Ряд плотных матриц возникающих в задачах электростатики, задаче многих тел, также матрицы, полученные при дискретизации сингулярных и гиперсингулярных интегральных уравнений, обладают блочно-малоранговой структурой. Под блочно-малоранговой структурой мы понимаем структуру блочной матрицы, с $M \times M$ блоками, при которой все блоки матрицы кроме $\mathcal{O}(M)$ имеют приближённо малый ранг. За данной структурой лежит следующий общий физический смысл: строки и столбцы таких матриц ассоциированы с некоторыми элементами в пространстве, задана некоторая функция взаимодействия этих элементов, если функция взаимодействия асимптотически гладкая, то взаимодействие разнесенных в пространстве групп элементов можно приблизить малым числом параметров [81] (критерий разделения). Таким образом, блоки, ассоциированные с хорошо разделёнными в пространстве группами элементов обладают приближенным малым рангом. Другой известный пример блочно-малоранговых матриц связан с матрицами полученными при дискретизации дифференциальных уравнений. Известно [9, 13], что если матрица A получена при конечно-элементной дискретизации дифференциального уравнения, удовлетворяющего некоторым ограничениям [9, 10, 13], то обратная к ней приближается блочно-малоранговой матрицей.

Блочно-малоранговые матрицы представляют из себя приближение с хорошей точностью плотных матриц в малопараметрическом формате. Блоки малого ранга представляются в виде произведения матриц меньшего размера. Это позволяет значительно экономить машинную память, так в отличие от плотной матрицы, которая требует n^2 ячеек памяти малопараметрическое представление, требует $\mathcal{O}(n(\log^{\alpha} n)(\log^{\beta} \varepsilon^{-1}))$ ячеек памяти (в зависимости от типа малопараметри-

ческого представления). Данная особенность позволяет хранить приближение к плотной матрице используя память, требуемую для хранения разреженной матрицы такого же размера. Другой характерной особенностью малопараметрического представления является быстрая процедура умножения такой матрицы на вектор ($\mathcal{O}(n \log n)$ или $\mathcal{O}(n)$ операций в зависимости от вида представления).

Быстрая процедура умножения матрицы на вектор позволяет эффективно применять к решению систем с малопараметрическими матрицами итерационные методы. Однако в случае плохой обусловленности, когда требуется решить систему прямым методом или приближенно для построения предобуславливателя матрицы в малопараметрическом представлении сталкиваются со значительными трудностями. Одной из основных трудностей является сложный формат хранения малопараметрических матриц: малопараметрические форматы, такие как $\mathcal{H}$ [39, 81], $\mathcal{H}^2$ [14, 42], HSS [17, 28] и т.д. рассчитаны на быстрое умножение матрицы на вектор, однако быстрое исключение строк и столбцов для таких матриц является трудоёмкой задачей. Данная работа посвящена методам прямого решения и приближенной факторизации систем с блочно-малоранговыми матрицами в малопараметрическом формате.

i.2. Историческая справка

- **Идеи, предшествовавшие блочно-малоранговым матрицам.** В 80-х годах появились алгоритмы работы с нелокальными операторами, использующие иерархическое разбиение области и приближение взаимодействия пространственно разнесённых блоков. Это метод Барнса-Хата [8] и быстрый мультипольный метод [19, 37, 38, 68]. Алгоритм Барнса-Хата использовался для решения гравитационной задачи многих тел, то есть фактически, умножения плотной матрицы на вектор, и требовал $\mathcal{O}(n \log n)$ ячеек памяти и операций, так как иерархически разбивались только источники. В отличие от него, быстрый мультипольный метод использовал иерархическое разбиение как источников так и приёмников, что позволяло получить линейную сложность хранения и выполнения операций. Оба этих метода не аппроксимировали и не хранили блочно-малоранговую матрицу явно, но опирались на похожие идеи.

- **Мозаично-скелетный метод и $\mathcal{H}$ матрицы.** По-видимому, первым алгебраическим методом аппроксимации блочно-малоранговых матриц был мозаично-скелетный метод [80,81], разработанный Е. Е. Тыртышниковым в 1993 году. В данном методе источники и приёмники иерархически разбиваются на кластеры, что приводит к блочно-иерархической структуре матрицы (мозаичное разбиение). Блоки матрицы, соответствующие хорошо разнесенным в пространстве кластерам, находящимся на одном уровне иерархии, аппроксимируются с помощью псевдо-скелетного разложения [34]. Для хранения мозаично-скелетонной матрицы требуется $\mathcal{O}(n \log n)$ ячеек памяти. В случае, если элементы матрицы заданы некоторой функцией для построении мозаично-скелетонной матрицы требуется $\mathcal{O}(n \log n)$ обращений к данной функции. Умножение такой матрицы на вектор может быть выполнено при помощи $\mathcal{O}(n \log n)$ операций. Позже, похожая идея иерархического разбиения и малорангового приближения дальних блоков также была представлена в работах [40,42].

- **$\mathcal{H}^2$ матрицы.** В работах [41,42] $\mathcal{H}$ матрицы были обобщены на случай вложенности базисов. Основная идея заключается в том, что базисные строки и столбцы блоков на нижних уровнях могут быть использованы в качестве базисных на более высоких уровнях иерархии. $\mathcal{H}^2$ матрицы являются матричным аналогом быстрого мультипольного метода. Для их хранения требуется $\mathcal{O}(n)$ ячеек памяти, а для умножения такой матрицы на вектор $\mathcal{O}(n)$ операций. Важной задачей является быстрое построение $\mathcal{H}^2$ аппроксимации матрицы, в работах А. Ю. Михалева [56,60] а также в его кандидатской диссертации [86] предложены методы построения такой аппроксимации по элементам матрицы.

  Кроме общего случая $\mathcal{H}^2$ матриц, идею вложенности базисов также используют HSS (Hierarchical Semi-Separable) [17, 28, 54, 73, 84, 85] матрицы, HODLR (Hierarchically Off-Diagonal Low-Rank) [5, 7] матрицы и др. $\mathcal{H}^2$ матрицы успешно применяются для различных практических приложений [7,25,59,61,88].

- **Расширение области применимости блочно-малоранговых матриц.** В работах [9, 10] было доказано, что матрицы, обратные к полученным

при конечно-элементной дискретизации дифференциальных уравнений, удовлетворяющих некоторым ограничениям [9, 10, 13], имеют блочно-малоранговую структуру. Данный факт приводит к идее малоранговой апроксимации заполнения, возникающего при LU факторизации разреженных матриц. Этой теме посвящена часть кандидатской диссертации.

- **Прямое решение систем с блочно-малоранговыми матрицами.** Поскольку методы аппроксимации блочно-малоранговых матриц предоставляют возможность быстрого матрично-векторного произведения, основной метод решения систем с ними итерационный. Для случаев плохой обусловленности требуется построить предобуславливатель, как, например, в работе [20]. Приближенная факторизация приводящая к предобуславливателям блочно-малоранговых матриц является предметом данной работы.

i.3. Общая характеристика кандидатской работы

i.3.1. Цель диссертационной работы

Целью диссертационной работы являлась разработка новых приближенных факторизаций блочно-малоранговых матриц и методов их построения.

i.3.2. Научная новизна

Предложен новый метод приближенной факторизации разреженных матриц (метод компрессии и исключения), также предложены два метода разреженной факторизации $\mathcal{H}^2$ матриц.

i.3.3. Практическая ценность

Предложенный в работе метод приближенной факторизации разреженных матриц может быть использован для приближенного решения, предобуславливания и приближенного вычисления определителя разреженных положительно определённых матриц, в частности, полученных при дискретизации дифференциальных уравнений.

Методы приближенной факторизации блочно-малоранговых матриц, в свою очередь, могут быть применены для приближенного решения и предобуславливания систем с плотными матрицами и для приближенного в вычисления определителя плотных матриц в задачах электростатики, аэро- и гидродинамики, а также в прикладной статистике. Данные методы показали свою эффективность в сравнении методами HODLR [5, 7] и IFMM [6, 20].

i.3.4. Положения, выносимые на защиту

На защиту выносятся следующие результаты и положения:

Основной результат работы состоит в том, что предложены новые приближенные факторизации блочно-малоранговых матриц и методы их построения, реализован программный комплекс, который применен к нескольким задачам математического моделирования. В частности:

1. Предложена факторизация разреженной симметричной положительно определённой матрицы в виде произведения матриц перестановки, блочно-диагональных унитарных и разреженных нижне-треугольных факторов (метод компрессии и исключения, compress and eliminate method, CE), на основе которой предложен прямой метод решения систем и метод построения предобуславливателя.

2. Предложены два типа разреженной факторизации $\mathcal{H}^2$ матриц: «расширенная» и «не-расширенная», которые позволяют ускорить решение линейных систем с $\mathcal{H}^2$ матрицами, в сравнении с итерационным методом BiCGStab [2] и прямыми методами HODLR [5, 7] и IFMM [6, 20].

3. Разработан комплекс программ реализующих представленные алгоритмы. Для факторизации CE проведено тестирование на системах, полученных при конечно-разностной дискретизации стационарного уравнения диффузии, а также системах, полученных при конечно-элементной дискретизации уравнения Пуассона и уравнения упругости. Проведено сравнение реализации метода CE с прямыми методами CHOLMOD [4, 24] и UMFPACK [22], а также итерационным методом BiCGStab с предобуславливателями ILU0 [70], ILUt [69] и ILU2 [46]. Для метода расширенной раз-

реженной факторизации проведено тестирование на задачах электростатики и гидромеханики, в ходе которого метод показал свою эффективность по времени в сравнении с непредобусловленным методом BiCGStab. Для не-расширенной разреженной факторизации проводилось тестирование на задаче моделирования гауссовских процессов для задачи регрессии. Метод показал свою эффективность по времени в сравнении с методами HODLR и IFMM.

i.3.5. Апробация работы и публикации

Результаты диссертационной работы докладывались автором и обсуждались на следующих научных семинарах и на конференциях:

- 56-я научная конференция МФТИ, 2013, Москва
- Научная конференция “Ломоносовские чтения”, 2013, Москва
- Fast Direct Solvers for Elliptic PDEs, Dartmouth College, 2014, USA
- Workshop: Low-rank Optimization and Applications, Hausdorff Center for Mathematics, Bonn, 2015, Germany
- 4th International Conference on Matrix Methods in Mathematics and Applications, Skolkovo Institute of Science and Technology, 2015, Moscow
- Scalable Hierarchical Algorithms for eXtreme Computing workshop, King Abdullah University of Science and Technology, 2016, Saudi Arabia
- Seminar of Extreme Computing Research Center, King Abdullah University of Science and Technology, 2016, Saudi Arabia
- 59-я научная конференция МФТИ, 2016, Москва
- Семинар имени К.И. Бабенко, ИПМ РАН, 2016, Москва
- Workshop on Fast Direct Solvers, Purdue CCAM, 2016, USA
- Объединённый семинар ИВМиМГ СО РАН и кафедры вычислительной математики НГУ, 2017, Новосибирск

- Сейсмический семинар ИНГиГ СО РАН, 2017, Новосибирск
- Научная конференция “Ломоносов”, 2017, Москва
- Семинар лаборатории ”Математическое моделирование нелинейных процессов в газовых средах”, МФТИ, 2017, Москва

Основные результаты кандидатской диссертации опубликованы в следующих работах:

1. Работы, опубликованные в изданиях, входящих в перечень рецензируемых научных изданий ВАК:
   (a) Sushnikova D. A., Oseledets I. V. Preconditioners for hierarchical matrices based on their extended sparse form //Russian Journal of Numerical Analysis and Mathematical Modelling. — 2016. — №. 1. — С. 29-40.
   (b) Сушникова Д. А., Приложение блочно-малоранговых матриц для задачи регрессии на основе гауссовских процессов //Вычислительные методы и программирование, — 2017. — Т. 18. — С. 214-220.
2. Статьи в журналах Web of Science
   (a) Ryzhakov G. V. , Mikhalev, A. Y., Sushnikova, D. A., Oseledets, I. V. Numerical solution of diffraction problems using large matrix compression //Antennas and Propagation (EuCAP), 2015 9th European Conference on. — IEEE, — 2015. – С. 1-3.
3. Работы, опубликованные в прочих изданиях:
   (a) Sushnikova D. A., Oseledets I. V. ” Compress and eliminate” solver for symmetric positive definite sparse matrices //arXiv preprint arXiv:1603.09133. – 2016.
   (b) Sushnikova D. A., Oseledets I. V. Simple non-extensive sparsification of the hierarchical matrices //arXiv preprint arXiv:1705.04601 – 2017.

i.3.6. Личный вклад

Результаты, описанные в главе 2, опубликованы в работе [3a], эта работа опубликована в соавторстве с И. В. Оселедцем. В работе [3a] основная идея метода разработана Д. А. Сушниковой. Также автору диссертации принадлежит программа ЭВМ и численные эксперименты. Оселедцу И.В. принадлежит постановка задачи.

Результаты, описанные в главе 3, опубликованы в работах [1a] и [3b], эти работы опубликованы в соавторстве с И. В. Оселедцем. В работах [1a] и [3b] Д. А. Сушниковой принадлежит основная идея метода, программа ЭВМ и численные эксперименты, Оселедцу И.В. принадлежит постановка задачи.

Результаты, описанные в главе 4, опубликованы в работе [1b], эта работа опубликована автором самостоятельно.

i.4. Содержание работы по главам

Диссертация состоит из введения, пяти глав («Блочно-малоранговые матрицы в задачах моделирования», «Метод компрессии и исключения», «Методы сведения малопараметрических матриц к разреженным», «Программный комплекс для факторизации и решения систем с блочно-малоранговыми матрицами», «Приложение для задачи регрессии на основе гауссовских процессов»), заключения и списка литературы. В главе «Блочно-малоранговые матрицы в задачах моделирования» приводится обзор задач математического моделирования которые эффективно решаются при помощи блочно-малоранговых методов, а также приводятся предварительные сведения об иерархических блочно-малоранговых матрицах. Кроме того, дается определение $\mathcal{H}^2$ матрицы, которое будут использоваться в тексте диссертации.

Глава «Метод компрессии и исключения» содержит метод приближенной факторизации разреженных матриц, разработанный автором диссертации, и оценку сложности предложенного алгоритма на основе анализа графа разреженности исходной матрицы.

В главе «Методы сведения малопараметрических матриц к разреженным» автором предложены два метода приведения $\mathcal{H}^2$ матрицы к разреженному виду:

расширенный, в котором размер полученной разреженной матрицы больше размера исходной $\mathcal{H}^2$ матрицы и не-расширенный, в котором размер разреженной и $\mathcal{H}^2$ матриц совпадает. Для не-расширенного метода приводится доказательство разреженности полученной факторизации.

Глава «Программный комплекс для факторизации и решения систем с блочно-малоранговыми матрицами» посвящена описанию программного кода, реализующего алгоритмы, приведённые в главах 2 и 3. Также в этой главе приводится сравнение предложенных автором программ с другими методами решения линейных систем для ряда задач.

Глава «Приложение для задачи регрессии на основе гауссовских процессов» посвящена применению блочно-малоранговых методов в задачах моделирования коррелированного шума.

i.5. Благодарности

Автор диссертации хотела бы поблагодарить в первую очередь своего научного руководителя, Ивана Валерьевича Оселедца, за научное руководство и постоянную поддержку в течении всех лет обучения в МГУ им. Ломоносова и ИВМ РАН им. Г.И.Марчука. Для меня и многих других студентов и аспирантов Иван Валерьевич является вдохновляющим примером исследователя с широким кругом научных интересов и глубоко разбирающегося в каждой исследуемой теме. Эти качества позволяют ему предлагать студентам задачи, в полной мере раскрывающие их потенциал.

Также автор выражает признательность коллективу ИВМ РАН им. Г.И.Марчука за профессиональный подход организации учебного процесса. Выбор актуальных тем современной вычислительной математики, в которой преподаватели являются признанными в мире специалистами, и блестящее качество авторских курсов позволяет институту выпускать специалистов по вычислительной математике и математическому моделированию, востребованных во всём мире. Основная часть моих теоретических и практических знаний, позволивших подготовить кандидатскую диссертацию, получена именно в ИВМ РАН им. Г.И.Марчука.

Хочу, кроме того, поблагодарить группу ”Научных вычислений” Сколковского Института Науки и Технологий, в которой автор работает в данное время, за творческую и плодотворную научную атмосферу, консультации и дискуссии по широкому кругу тем, касающихся данной диссертации и уходящих далеко за её пределы, а также за помощь в разнообразных технических вопросах, связанных с редактированием текста данной работы, поиском ошибок в программном коде и многих других.

Наконец, хочу поблагодарить свою семью: мужа и родителей за терпение и ежедневную поддержку во время написания данной работы.

Глава 1

Блочно-малоранговые матрицы в задачах моделирования

Данная глава является вводной и в ней приведён обзор блочно-малоранговых методов в приложении к задачам моделирования и показана целесообразность применения таких методов. Также в главе приведены предварительные сведения об иерархических блочно-малоранговых матрицах.

1.1. Обзор применения блочно-малоранговых методов в моделировании

В данном разделе анализируются задачи моделирования, при решении которых могут эффективно использоваться блочно-малоранговые матрицы. Также в данном разделе отмечаются основные преимущества блочно-малорангового подхода для таких задач.

1.1.1. Задачи моделирования, приводящие к системам с блочно-малоранговыми матрицами

Ряд задач моделирования приводит линейным системам, которые можно решать используя те или иные блочно-малоранговые методы. В основном, это так называемые «нелокальные» задачи. Как было отмечено в разделе i.1, если функция взаимодействия элементов асимптотически гладкая, то подматрицу, соответствующую взаимодействию разнесенных в пространстве группы элементов, можно приблизить малым числом параметров [81]. Такое свойство называют критерием разделения. Нелокальные задачи, как правило, приводят к плотным матрицам, обладающим блочно-малоранговой структурой. Приведём несколько примеров таких задач.

- **Задача многих тел** заключается в моделировании облака частиц под воздействием некоторых самопорождённых сил. Примером такой задачи может быть вычисление потенциалов облака заряженных частиц [88], в этом случае частицы взаимодействуют по закону Кулона $f(x,y) = \frac{1}{|x-y|}$. Другим важным примером задачи многих тел является гравитационная задача, частицы в ней взаимодействуют по закону $f_{ij} = \frac{Gm_i m_j (r_i - r_j)}{(|r_i - r_j|^2 + \epsilon^2)^{3/2}}$. Моделирование задачи многих тел широко распространено в астрофизике, начиная от моделирования систем типа Солнце-Земля-Луна до понимания эволюции крупномасштабных структур вселенной [79]. Следует отметить, что моделирование больших систем стало возможно именно благодаря блочно-малоранговым методам [8, 79].

- **Задачи, приводящие к интегральным уравнениям** при дискретизации часто сводятся к системам с блочно-малоранговыми матрицами. Приведём примеры несколько таких задач.

  - **Задача аэро- и гидродинамики.** Метод дискретных вихрей используется при моделировании аэродинамики самолётов и парашютов [51], при моделировании ветра в городской застройке [87], а также в моделировании ураганов [51]. Данный метод приводит к гиперсингулярному интегральному уравнению. Систему, полученную при дискретизации интегрального уравнения решают при помощи блочно-малоранговых

методов [74, 89]. Быстрый метод решения такой задачи приведён в главе 4, в параграфе 4.2.1 данной диссертации.

– **Задача электростатики.** Задача радиолокации часто решается при помощи гиперсингулярного уравнения электрического поля [59, 75, 76]. Один из методов решения данной задачи приведён в главе 4, в параграфе 4.2.1 данной диссертации.

– **Континуальная модель растворителя.** Задача вычисления полярной составляющей энергии сольватации молекулы является составной часть сложной системы моделирования и дизайна лекарств. Поляризованная модель континуума [21] это интегральное уравнение, которое эффективно может быть решено при помощи блочно-малоранговых матриц [86, 88].

- При помощи **гаусовских процессов** в метеорологии моделируется суточная норма осадков, температура и солнечная радиация [49, 67]. Также гауссовские процессы применяются в астрономии [29] и многих других областях. Одним из ключевых понятий в гауссовских процессах является матрица ковариации, это плотная матрица, обладающая блочно-малоранговыми свойствами, с которой в процессе моделирования требуется выполнять различные алгебраические операции (умножение на вектор, вычисление определителя, обращение), применение блочно-малоранговой аппроксимации позволяет значительно ускорить все вычисления. Задача моделирования при помощи гауссовских процессов подробно рассмотрена в главе 5.

Изначально, блочно-малоранговые матрицы применялись для задач математического моделирования с нелокальными связями, в частности, в интегральных уравнениях [14–16, 41, 54, 68]. Однако, позже, в работах [9, 10, 13, 35, 36] была предложена концепция использования блочно-малоранговых матриц для приближения обратной матрицы а также построения LU разложения и разложения Холецкого с факторами, представленными в блочно-малоранговых форматах. Идея казалась очень интересной, так как для класса задач, возникающих при дискретизации эллиптических дифференциальных уравнений второго порядка было доказано [9, 10], что соотвествующие матрицы имеют блочно-малоранговую структуру. В случаях, когда необходимо многократно решать линейную систему с од-

ной и той же матрицей, такой подход выглядит очень перспективно. Однако, возникла проблема, которая заключается в том, что несмотря на асимптотическую оптимальность предложенных алгоритмов, они часто проигрывали стандартным подходам (CHOLMOD, методы неполной факторизации). В недавних работах [63, 78, 82] эта идея получила новое развитие, связанное с построением новых приближенных факторизаций, основанных на блочно-малоранговых аппроксимациях, которые в ряде случаев становятся даже более эффективными, чем классические прямые методы решения разреженных линейных систем.

Решение задач диффузии и упругой деформации при помощи блочно-малоранговой аппроксимации факторов разложения Холецкого приводится в главе 4 в разделе 4.1 данной диссертации.

1.2. Иерархические блочно-малоранговые матрицы

В данном разделе сообщаются базовые понятия, которые будут использоваться в последующих главах кандидатской диссертации. Основным предметом диссертации являются блочно-малоранговые матрицы, ниже приводятся определения основных типов блочно-малоранговых матриц и сообщаются необходимые сведения о них.

1.2.1. Мозаично-скелетонные ($\mathcal{H}$) матрицы

Блочно-малоранговые матрицы это плотные матрицы обладающие специальной структурой: некоторые блоки данной матрицы можно приблизить блоками малого ранга. Важным случаем блочно-малоранговых матриц является мозаично-скелетонные [32, 80, 81] или $\mathcal{H}$ [42] матрицы. Первая ключевая идея мозаично-скелетонных матриц - это иерархическое разбиение матрицы на блоки, называемое мозаичным разбиением матрицы. Разбиение хранится в блочных кластерных деревьях строк и столбцов.

***Определение* 1.2.1** (Кластерное дерево)**.** *[14] Блочное кластерное дерево строк $\mathcal{T}_r$ (столбцов $\mathcal{T}_c$) матрицы A это дерево в котором:*

1. *Каждая вершина $i \in \mathcal{T}_r$ ($j \in \mathcal{T}_c$) ассоциирована с группой строк (столбцов).*

2. *Корневая вершина содержит все строки (столбцы).*

3. *Если вершина i является дочерней для вершины j, то строки, соответствующие i являются подмножеством строк, соответствующих j.*

Уровни будем считать от листьев к корню начиная с $l = 0$

Пусть $\mathcal{T}_c$ и $\mathcal{T}_r$ это столбцовые и строчные блочные кластерные деревья матрицы A. Для каждого уровня дерева рассмотрим все возможные пары вершин $p = (v, w)$ где $v \in \mathcal{T}_r, w \in \mathcal{T}_c$ на этом уровне. Отметим, что каждая пара p соответствует некоторому блоку матрицы A. Проведем следующую процедуру: начиная с корневого уровня, рассмотрим все пары p на этом уровне, и если блок матрицы A соответствующий p имеет приближенный малый ранг, то назовём его дальним и вычеркнем все дочерние вершины вершин v и w (которые находятся на уровень ниже) из дальнейшего рассмотрения. Те вершины, которые не имею приближенного малого ранга отметим как ближние. В ходе данной процедуры мы получим разбиение блоков матрицы A на «дальние» и «ближние». Пример такого разбиения приведен на Рисунке 1.1. Рисунки 1.1 и 1.2 предоставлены А. Ю. Михалевым.

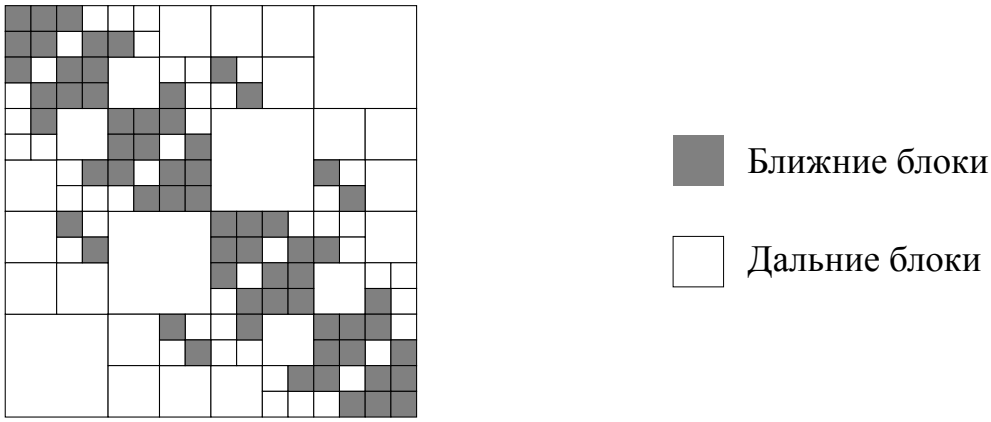


Рисунок 1.1: Иллюстрация мозаично-скелетонного разбиения

Отметим, что для произвольной плотной матрицы построение мозаичного разбиения является переборной задачей. Однако, если для матрицы известна дополнительная информация, такая, как сетка, если матрица получена при дискретизации интегрального уравнения, или положения частиц, если матрица порождена задачей многих тел, то мозаичное разбиение может быть построено на основе геометрической информации. Основная идея следующая: при наличии некоторой гладкости функции взаимодействия частиц, подматрица отвечающая гео-

метрически разнесённым частицам имеет приближенный малый ранг. Процедура построения мозаичного разбиения на основе геометрической информации аналогична процедуре её построения для матрицы, вместо кластерных деревьев строк и столбцов рассматриваются деревья соответствующих частиц,если группы частиц недостаточно разделены в пространстве, то они делятся на подгруппы, если группы частиц являются разнесёнными в пространстве - то биение прекращается.

Вторая ключевая идея мозаично-скелетонного метода - это аппроксимация дальних блоков, то есть блоков имеющих приближенный малый ранг, при помощи скелетного разложения. Скелетное разложение является методом малоранговой аппроксимации приближенно - малоранговых матриц. Пусть требуется приблизить матрицу $B \in \mathbb{R}^{M\times M}$, выберем r строк $R \in \mathbb{R}^{r\times N}$ и r столбцов $C \in \mathbb{R}^{N\times r}$, таких, что подматрица на их пересечении $\widehat{B}$ имеет максимальный модуль определителя (максимальный объем), тогда приближение

$$B \approx C\widehat{B}^{-1}R \tag{1.1}$$

является квазиоптимальным [33, 34].

Аппроксимация плотной матрицы при помощи мозаично-скелетонного метода приводит к уменьшению затрат на её хранение и,соответственно, ускоряет её умножение на вектор. Хранение $\mathcal{H}$ матрицы требует $\mathcal{O}(Nr\log(N))$, где N - размер матрицы A. Для умножения матрицы A на вектор требуется $\mathcal{O}(Nr\log(N))$ операций.

1.2.2. Блочно-малоранговые матрицы со вложенными базисами

Для уменьшения затрат по памяти используется дополнительные предположения относительно дальних блоков матрицы, описанные ниже. Пусть $\mathcal{T}_c$ и $\mathcal{T}_r$ это столбцовые и строчные блочные кластерные деревья матрицы A. Пусть для каждой пары вершин с $p = (v, w)$ где $v \in \mathcal{T}_r, w \in \mathcal{T}_c, v, w$ с одного уровня дерева, задана ближняя эта пара или дальняя. Тогда вводится такое понятие как кластерный базис.

***Определение* 1.2.2** (Кластерный базис)**.** *[14, стр. 54] Пусть $K = (K_i)_{i\in\mathcal{T}_I}$ это семейство конечных множеств индексов. Пусть $R = (R_i)_{i\in\mathcal{T}_r}$ это семейство*

матриц удовлетворяющих $R_i \in \mathbb{R}^{I \times K_t}$ для всех $i \in \mathcal{T}_r$. Тогда R называется кластерным базисом с распределением рангов K и матрицами R_i называются матрицами кластерного базиса.

Отметим, что ключевым свойством $\mathcal{H}^2$ структуры является свойство *вложенности базисов*.

***Определение* 1.2.3** (Вложенный кластерный базис.)**.** *[14, стр. 54] Пусть R это кластерный базис с распределением ранга K. R называется вложенной если существует семейство $(E_i)_{i \in \mathcal{T}_r}$ матриц удовлетворяющих следующим условиям:*

1. *Для всех $i \in \mathcal{T}_r$ и всех $i' \in \text{sons}(i)$, существуют $E_{i'} \in \mathbb{R}^{K_{i'} \times K_i}$.*
2. *Для всех $i \in \mathcal{T}_r$ таких, что $\text{sons}(i) \neq \varnothing$, выполняются следующие уравнения:*

$$R_i = \sum_{i' \in sons(i)} R_{i'} E_{i'}, \tag{1.2}$$

Матрицы E_i называются матрицами перехода.

***Определение* 1.2.4** (Список взаимодействия)**.** *Рассмотрим вершину i на уровне l кластерного дерева $\mathcal{T}_r$ и вершину j на уровне l кластерного дерева $\mathcal{T}_c$. если пара (i, j) дальняя, а пара $(\text{father}(i), \text{father}(j))$ ближняя, тогда блок (i, j) входит в список взаимодействия уровня l. Рассмотрим матрицу S_l всех блоков взаимодействия уровня l. Множество матриц взаимодействия на всех уровнях*

$$S = \{S_l\}_1^L$$

назовём списком взаимодействия.

***Определение* 1.2.5** (Ближняя матрица)**.** *Определим матрицу ближних блоков на нулевом уровне через "ближнюю матрицу" C.*

***Определение* 1.2.6** ($\mathcal{H}^2$ матрица)**.** *Если $\mathcal{T}_c$ и $\mathcal{T}_r$ это строчные и столбцовые кластерные деревья матрицы A, C - это её ближняя матрица. Если существует строчный набор матриц перехода R, столбцовый набор матриц перехода E и список взаимодействия S и если выполняется*

$$A = C + \sum_{i \in 0}^{L-1} \left(\prod_{j=0}^{i} E_j \right) S_{i+1} \left(\prod_{j=0}^{i} R_j^* \right),$$

тогда матрица $A \in \mathbb{R}^{N \times N}$ является аппроксимированной в $\mathcal{H}^2$ формате.

На Рисунке 1.2 привеен пример $\mathcal{H}^2$ матрицы, темно-серые блоки - это блоки матрицы C, засштрихованные блоки - это дальние блоки i-й строки.

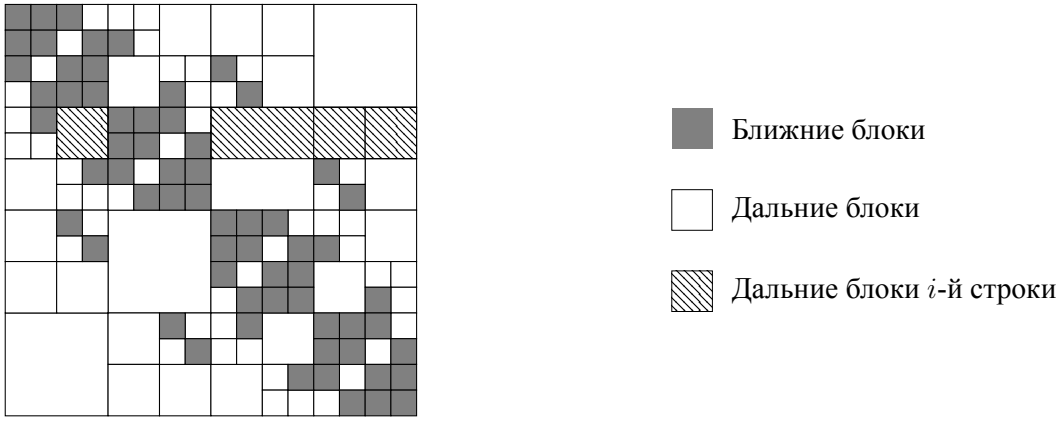


Рисунок 1.2: Иллюстрация матрицы с $\mathcal{H}^2$ структурой.

Понятие $\mathcal{H}^2$ матрицы является одним из ключевых понятий диссертации. Оно используется во всех последующих главах.

1.3. Выводы по главе

Данная глава является вводной и содержит краткий обзор задач математического моделирования к которым успешно применяются иерархические блочно-малоранговые методы. Также в данной главе вводится ряд ключевых для диссертации понятий, таких как «кластерное дерево», «блочно-малоранговая матрица», «$\mathcal{H}^2$ матрица». Данные понятия используются в следующих главах работы.

Глава 2

Метод компрессии и исключения

В данной Главе приводится метод приближенной факторизации разреженных симметричных положительно определённых матриц «Метод компрессии и исключения» (Compress and eliminate method, CE), разработанный автором диссертации.

2.1. CE алгоритм для симметричной положительно определенной матрицы

Рассмотрим линейную систему с разреженной симметричной положительно определённой матрицей $A \in \mathbb{R}^{N \times N}$. Перед началом описания алгоритма выберем перестановку P и будем рассматривать матрицу

$$A_0 = PAP^\top. \tag{2.1}$$

Выбор перестановки является важной частью алгоритма, он будет подробно описан далее в параграфе 2.2.2. Перестановка P и размер блока B определяют разбиение матрицы A_0 на блоки. Размер блока B это параметр алгоритма, как правило, это небольшое число. На Рисунке 2.1 приведён пример матрицы A_0.

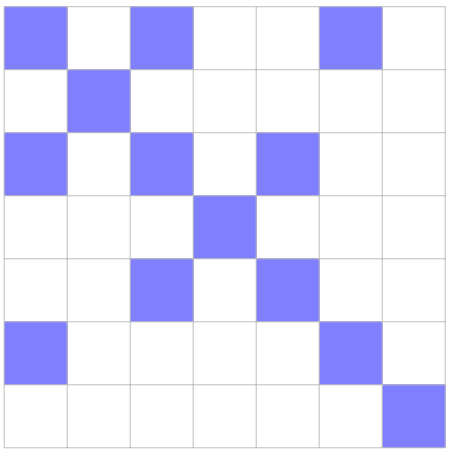


Рисунок 2.1: Матрица A_0

Пусть матрица A_0 задана в формате “сжатых блочных строк” (*compressed sparse block* (CSB) [69] format). Размер блока B - небольшое число, которое является параметром алгоритма. Исходные ненулевые блоки будем называть “ближними”. Ненулевые блоки, которые появятся при исключении будем называть “дальними”. Начнем с исключения первой блочной строки.

2.1.1. Исключение 1-й блочной строки

Рассмотрим блочную разреженную симметричную положительно определенную матрицу $A_0 \in \mathbb{R}^{N \times N}$ в следующем виде:

$$A_0 = \begin{bmatrix} D_1 & A_{1r} \\ A_{1r}^\top & A_{1*} \end{bmatrix},$$

где D_1 размера $B \times B$. Здесь и далее все диагональные блоки предполагаются невырожденными. Первая блочная строка $\widetilde{R}_1$ может быть переписана в виде

$$\widetilde{R}_1 = \begin{bmatrix} D_1 & A_{1r} \end{bmatrix} = \begin{bmatrix} D_1 & C_1 & 0 \end{bmatrix} P_1^{\text{col}},$$

где C_1 соответствует всем ближним блокам. Матрица P_1^{col} это матрица перестановки. Чтобы сделать первый шаг блочной факторизации Холецкого факторизуем диагональный блок:

$$D_1 = \widehat{L}_1 \widehat{L}_1^\top.$$

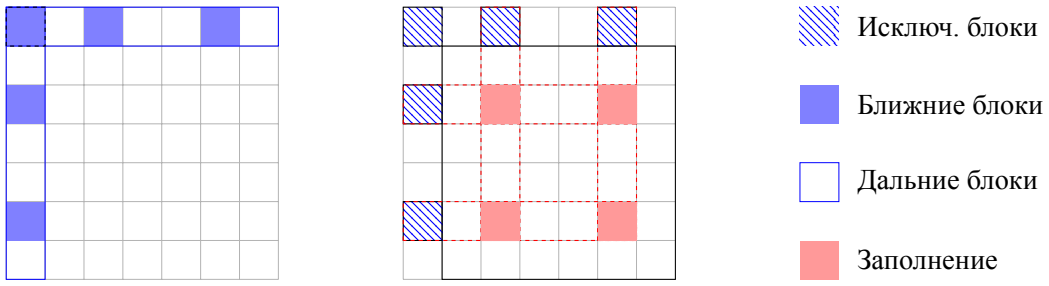
Затем применяем разложение Холецкого [4]:

$$A_0 = \begin{bmatrix} \widehat{L}_1 & 0 \\ A_{1r}^\top \widehat{L}_1^{-1} & 0 \end{bmatrix} \begin{bmatrix} \widehat{L}_1^\top & \widehat{L}_1^{-\top} A_{1r} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{A}_1^* \end{bmatrix} = \widetilde{L}_1 \widetilde{L}_1^\top + \widetilde{A}_1, \tag{2.2}$$

где блок $\widetilde{A}_1^* = A_{1*} - A_{1r}^\top D_1^{-1} A_{1r}$ это дополнение по Шуру. Матрица $\widetilde{A}_1^*$ имеет шаблон разреженности равный шаблону разреженности матрицы A_{1*} с добавлением матрицы заполнения

$$\widetilde{A}_{1F} = A_{1r}^\top D_1^{-1} A_{1r}. \tag{2.3}$$

Ненулевые блоки могут возникать только в позициях (i, j), где $i \in \mathcal{I}_1, j \in \mathcal{I}_1$, $\mathcal{I}_1$ - индексы ненулевых блоков в первой блочной строке $\widetilde{R}_1$, смотри Рисунок 2.2b.



(a) Первая блочная строка $\widetilde{R}_1$ и блочный столбец $\widetilde{R}_1^\top$ (b) Матрица заполнения $\widetilde{A}_{1F}$

Рисунок 2.2: Иллюстрация заполнения вызванного исключением 1-й блочной строки

Обратите внимание что число ненулевых блоков в новой матрице $\widetilde{A}_{1F}$ ограничено сверху числом $(\#\mathcal{I}_1)^2$, где $\#\mathcal{I}_1$ это число ненулевых блоков в первой блочной строке $\widetilde{R}_1$. Так как A это блочно-разреженная матрица, число ненулевых блоков находится в пределах $(\#\mathcal{I}_1)^2$.Расположение этих блоков заранее известно.

Если продолжить исключение число ненулевых блоков может возрасти. Чтобы избежать этого мы будем использовать дополнительную *процедуру компрессии.*

2.1.2. Сжатие и исключение i-й блочной строки

Предположим что мы уже исключили $(i-1)$ блочную строку и теперь работаем с i-й блочной строкой, смотри Рисунок 2.3.

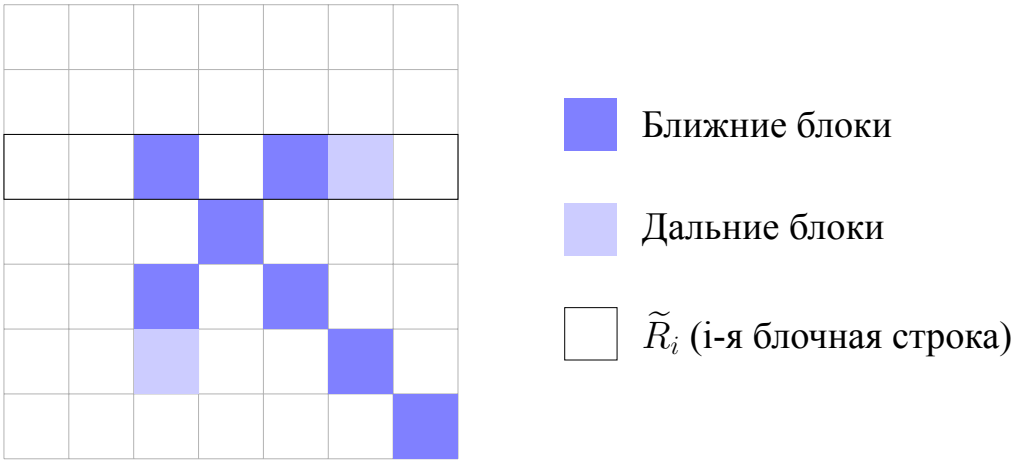


Рисунок 2.3: Матрица $\widetilde{A}_{i-1}$, перед исключением i-й строки (в данном примере $i = 3$)

Рассмотрим i-ю блочную строку матрицы $\widetilde{A}_{i-1}$ и попробуем её исключить. После исключения предыдущих блочных строк мы получим некоторое заполнение, смотри Рисунок 2.3. Переставим столбцы в блочной строке,

$$\widetilde{R}_i = \begin{bmatrix} D_i & C_i & F_i & 0 \end{bmatrix} P_i^{\text{col}},$$

где P_i^{col} это перестановка столбцов, $C_i = [C_i^{(1)} \dots C_i^{(n_i^C)}]$ это ближние блоки в i-й блочной строке $\widetilde{R}_i$, n_i^C это число ближних блоков в $\widetilde{R}_i$, $F_i = [F_i^{(1)} \dots F_i^{(n_i^F)}]$ это дальние блоки в $\widetilde{R}_i$, n_i^F это число дальних блоков в $\widetilde{R}_i$, смотри рисунок 2.4.

Ключевая идея заключается в аппроксимации матрицы $F_i \in \mathbb{R}^{B \times (Bn_i^F)}$ матрицей малого ранга:

$$\widetilde{U}_i^\top F_i = \begin{bmatrix} \hat{F}_i \\ E_i \end{bmatrix},$$

где $\widetilde{U}_i$ это унитарная матрица, $\|E_i\| < \varepsilon$ для некоторого ε и $\hat{F}_i \in \mathbb{R}^{r \times Bn_i^F}$.

***Замечание* 2.1.1.** *Мало-ранговая аппроксимация, представленная в уравнении (2.2) является важной деталью предложенного алгоритма. Рассматриваются две стратегии поиска ранга r:*

- *CE(ε): для заданной точности ε ищется минимальное r такое, что $\|E_i\| < \varepsilon$ (адаптивная аппроксимация).*
- *CE(r): ранг r фиксирован, обычно, $r = \frac{B}{2}$ аппроксимация фиксированного ранга).*

Алгоритм CE(ε) приводит к факторизации более высокой точности и поэтому может быть использован в качестве прямого метода решения систем. Основным недостатком алгоритма CE(ε) является то, что сжатие дальних блоков не гарантированно, что может привести к факторизации с большими затратами по памяти.

В то же время, алгоритм CE(r) напрямую контролирует использование памяти. Однако, в нём отсутствует контроль за точностью факторизации. Следовательно, CE(r) не подходит для прямого решения систем, но может оказаться хорошим предобуславливателем.

Пусть

$$\widetilde{Q}_i = \begin{bmatrix} I_{(i-1)B} & 0 & 0 \\ 0 & \widetilde{U}_i & 0 \\ 0 & 0 & I_{(M-i)B} \end{bmatrix}, \tag{2.4}$$

тогда матрица

$$\widehat{A}_{i-1} \approx \widetilde{Q}_i^\top \widetilde{A}_{i-1} \widetilde{Q}_i$$

имеет i-ю строку

$$\widetilde{R}_{i*} = \begin{bmatrix} \widetilde{U}_i^\top D_i \widetilde{U}_i & \widetilde{U}_i^\top C_i & \begin{bmatrix} \hat{F}_i \\ 0 \end{bmatrix} & 0 \end{bmatrix} P_i^{\text{col}},$$

смотри Рисунок 2.4.

Важная деталь алгоритма заключается в том, что мы исключаем только ту часть i-й блочной строки в преобразованной матрице $\widehat{A}_{i-1}$, которая обладает нулевой дальней зоной. Представим блочную подстроку $\widetilde{S}_i \in \mathbb{R}^{(B-r)\times n}$ (в красной прерывистой рамке на Рисунке 2.4):

$$\widetilde{S}_i = \begin{bmatrix} 0_{r\times r} & 0 \\ 0 & I_{(B-r)} \end{bmatrix} \widetilde{R}_{i*}.$$

Сейчас мы можем исключить блочную подстроку $\widetilde{S}_i$ матрицы $\widehat{A}_{i-1}$ используя блочную факторизацию Холецкого.

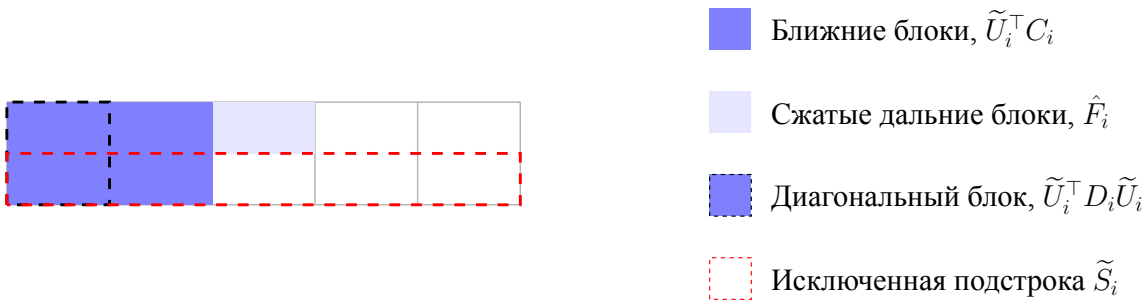


Рисунок 2.4: Шаг дожимания

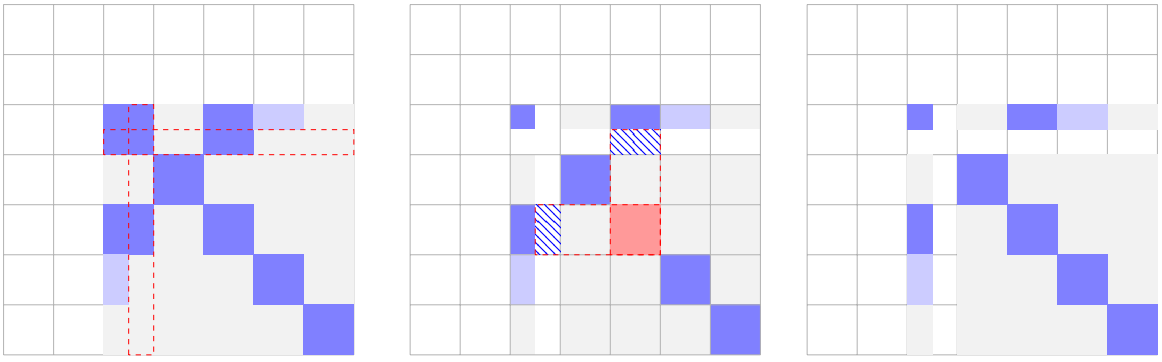
Обозначим диагональный блок блочной подстроки $\widetilde{S}_i$ за $\widetilde{S}_{ii}$, если разложение Холецкого диагонального блока это $\widetilde{S}_{ii} = \widehat{L}_i \widehat{L}_i^\top$ тогда

$$\widehat{A}_{i-1} = \widetilde{L}_i \widetilde{L}_i^\top + \widetilde{A}_i,$$

где матрица $\widetilde{A}_i \in \mathbb{R}^{n \times n}$ имеет нули на месте подстроки $\widetilde{S}_i$ и подстолбца $\widetilde{S}_i^\top$, и

$$\widetilde{L}_i = \widetilde{S}_i^\top \widehat{L}_i^{-\top} \tag{2.5}$$

Обратите внимание, что в данной схеме шаблон разреженности блочной подстроки $\widetilde{S}_i$ совпадает с блочным шаблоном разреженности оригинальной матрицы, таким образом после одного исключения в матрице $\widetilde{A}_i$ появится только константное число новых блоков. Также заметим что шаг компрессии не меняет расположения и размеров получаемого заполнения, смотри Рисунок 2.5b.



(a) Матрица $\widehat{A}_{i-1}$ (b) Заполнение $\widetilde{A}_{iF}$ (c) Матрица $\widetilde{A}_i$

Рисунок 2.5: Шаг исключения. Серый цвет обозначает не полностью нулевые строки и столбцы матрицы $\widetilde{A}_i$.

После шага компрессии-исключения матрица $\widetilde{A}_{i-1}$ приближается следующим об-

разом:

$$\widetilde{Q}_i \widetilde{A}_{i-1} \widetilde{Q}_i^\top \approx \widetilde{L}_i \widetilde{L}_i^\top + \widetilde{A}_i,$$

где $\widetilde{L}_i \in \mathbb{R}^{n \times (B-r)}$, а матрица $\widetilde{A}_i \in \mathbb{R}^{n \times n}$ имеет такой-же размер как исходная матрица A.

2.1.3. Полный проход алгоритма для одного уровня

Рассмотрим процедуру компрессии и исключения после обработки всех блочных строк. Мы храним факторы $\widetilde{L}_i$ умноженные на соответствующие матрицы $\widetilde{Q}_i$ как столбцы матрицы $\check{L}_1$:

$$\check{L}_1 = \left[\left(\prod_{j=M}^{2} \widetilde{Q}_j \right) \widetilde{L}_1 \quad \cdots \quad \left(\prod_{j=M}^{i+1} \widetilde{Q}_j \right) \widetilde{L}_i \quad \cdots \quad \widetilde{L}_M \right]. \tag{2.6}$$

Получаем следующее приближение:

$$\widetilde{Q}_M \dots \widetilde{Q}_1 A \widetilde{Q}_1^\top \dots \widetilde{Q}_M^\top \approx \check{L}_1 \check{L}_1^\top + \check{A}_1, \tag{2.7}$$

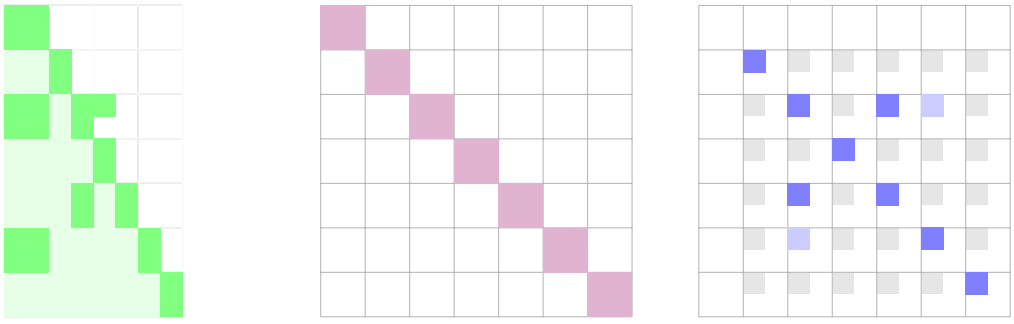
где матрица $\check{L}_1 \in \mathbb{R}^{n \times n_{L_1}}$, смотри Рисунок 2.6a, матрица $\check{A}_1 \in \mathbb{R}^{n \times n}$, смотри Рисунок 2.6c. Так как умножение на матрицу $\widetilde{Q}_i$ не меняет шаблон разреженности матриц $\check{L}_1$ и $\check{A}_1$ в ходе исключения, блочный шаблон разреженности $\check{L}_1$ Может быть легко получен из блочного шаблона разреженности исходной матрицы. Обозначим

$$Q_1 = \prod_{i=M}^{1} \widetilde{Q}_i.$$

Благодаря специальной структуре $\widetilde{Q}_i$ (3.10),

$$Q_1 = \operatorname{diag}(\widetilde{U}_1, \dots, \widetilde{U}_M),$$

то есть Q_1 это блочно диагональная матрица.



(a) Матрица $\check{L}_1$ (b) Матрица Q_1 (c) Матрица $\check{A}_1$

Рисунок 2.6: Матрицы $\check{L}_1$, Q_1 и $\check{A}_1$

Переставим блочные строки матрицы A_0 так чтобы исключенные стоки шли перед неисключенными:

$$P_1 Q_1 A_0 Q_1^\top P_1^\top \approx P_1(\check{L}_1 \check{L}_1^\top + \check{A}_1)P_1^\top = L_1 L_1^\top + \begin{bmatrix} 0_{n_{L_1} \times n_{L_1}} & 0 \\ 0 & A_1 \end{bmatrix},$$

где

$$L_1 = P_1 \check{L}_1.$$

Матрица $L_1 \in \mathbb{R}^{n \times n_{L_1}}$ это блочная разреженная нижне-треугольная матрица с размером бока $(B - r) \times (B - r)$, $A_1 \in \mathbb{R}^{n_{A_1} \times n_{A_1}}$ это блочная разреженная матрица $A_1 = P_1 \check{A}_1 P_1^\top$ с размером блока $r \times r$ и $n = n_{L_1} + n_{A_1}$, смотри Рисунок 2.7.

(a) Матрица L_1 (b) Матрица A_1

Рисунок 2.7: Матрицы L_1 и A_1 после полного прохода процедуры компрессии-исключения

Будем называть полный проход процедуры компрессии-исключения описанный выше “одним уровнем” исключения. Затем рассмотрим матрицу A_1 в качестве новой разреженной матрицы и применим к ней алгоритм компрессии-исключения (“второй уровень” исключения).

2.1.4. Многоуровневый алгоритм

Рассмотрим матрицу A_1 с блоками размера $Jr \times Jr$ и применим процедуру компрессии-исключения для поиска матриц Q_2, L_2 и P_2:

$$P_2 Q_2 A_1 Q_2^\top P_2^\top \approx L_2 L_2^\top + \begin{bmatrix} 0_{n_{L_2} \times n_{L_2}} & 0 \\ 0 & A_2 \end{bmatrix}.$$

Определим

$$\breve{Q}_2 = \begin{bmatrix} I_{n_{L_1}} & 0 \\ 0 & Q_2 \end{bmatrix}, \quad \breve{P}_2 = \begin{bmatrix} I_{n_{L_1}} & 0 \\ 0 & P_2 \end{bmatrix}, \quad \text{and} \quad \breve{L}_2 = \begin{bmatrix} 0_{n_{L_1} \times n_{L_2}} \\ L_2 \end{bmatrix}.$$

получим приближенную факторизацию второго уровня:

$$\breve{P}_2 \breve{Q}_2 \breve{P}_1 \breve{Q}_1 \, A_0 \, \breve{Q}_1^\top \breve{P}_1^\top \breve{Q}_2^\top \breve{P}_2^\top \approx \begin{bmatrix} L_1 & \breve{L}_2 \end{bmatrix} \begin{bmatrix} L_1 & \breve{L}_2 \end{bmatrix}^\top + \begin{bmatrix} 0 & 0 \\ 0 & A_2 \end{bmatrix}, \tag{2.8}$$

И так далее. Когда размер матрицы остатков достаточно мал мы применяем к ней разложение Холецкого.

Алгоритм компрессии и исключения (Compression-elimination (CE)) приводит к приближенной факторизации

$$A \approx QLL^\top Q^\top, \tag{2.9}$$

где Q это унитарная матрица равная произведению блочно-диагональных матриц и матриц перестановки, L это блочно-разреженная нижне-треугольная матрица.

2.1.5. Псевдокод алгоритма

В данном разделе обобщается алгоритм CE факторизации, и приводится псевдокод метода CE, описанного выше. Алгоритм 1 представляет собой общую схему алгоритма, приведенного в параграфе 2.1.4, он принимает на вход разреженную симметричную положительно определённую матрицу, параметр аппроксимации блоков заполнения (r или ε) и необходимое количество уровней, а возвращает разреженную CE факторизвцию (2.9).

Алгоритм 1: Алгоритм CE

Дано:

- $A \in \mathbb{R}^{N \times N}$ - разреженная матрица,
- $A_0 = PAP^\top$, $A_0 \in \mathbb{R}^{MB \times MB}$ - блочно-разреженная матрица,
- K - число уровней
- ε or r - параметр выбора ранга

Факторизация:

for $j = 1 \ldots K$ **do**

- $M_j = \frac{N}{BJ^{(j-1)}}$ - число блоков на текущем уровне
- **Исключение одного уровня: (см. Алгоритм 2)**
  - **Input** : A_{j-1}, M_j, r или ε
  - **Output** : $\breve{A}_j$, $\breve{L}_j$, Q_j
- $L_j = P_j^\top \breve{L}_j$,
- $A_j = P_j \breve{A}_j P_j^\top$ - остаток.
- Расширить Q_j единичной матрицей, получить $\breve{Q}_j$
- Расширить P_j единичной матрицей, получить $\breve{P}_j$
- Расширить L_j нулевой матрицей, получить $\breve{L}_j$

Факторизовать $A_K = L_{K+1} L_{K+1}^\top$,

$Q = \left(\prod_{j=1}^{K} \breve{P}_j \breve{Q}_j \right) P$,

$L = \begin{bmatrix} L_1 & \breve{L}_2 & \ldots & \breve{L}_{K+1} \end{bmatrix}$.

Вывод:

- L, Q (где $A \approx Q^\top L L^\top Q$).

Алгоритм 2 более подробно описывает процедуру исключения одного уровня, описанную в параграфе 2.1.3.

Алгоритм 2: Исключение одного уровня

Исключение j-го уровня

Дано:

A_{j-1}, M_j, ε or r

$\widehat{A}_0 = A_{j-1}$

for *i = 1 ...*M_j **do**

Шаг сжатия:

$\widetilde{R}_i = \begin{bmatrix} D_i & C_i & F_i & 0 \end{bmatrix} P_i^{\text{col}}$ - i-ая блочная строка матрицы $\widehat{A}_{i-1}$,

$F_i \approx \widetilde{U}_i \begin{bmatrix} \widehat{F}_i \\ 0 \end{bmatrix}$, $\widetilde{U}_i$ - унитарная матрица, $\widehat{F}_i \in \mathbb{R}^{r \times BN_i^F}$,

$\widetilde{R}_{i*} = \begin{bmatrix} \widetilde{U}_i^\top D_i \widetilde{U}_i & \widetilde{U}_i^\top C_i & \begin{bmatrix} \widehat{F}_i \\ 0 \end{bmatrix} & 0 \end{bmatrix} P_i^{\text{col}}$.

Шаг исключения:

$\widetilde{S}_i = \begin{bmatrix} 0_{r \times r} & 0 \\ 0 & I_{(B-r)} \end{bmatrix} \widetilde{R}_{i*}$,

Исключить $\widetilde{S}_i$ исбоьзуя блочное разложения Холецкого:

$\widehat{A}_{i-1} = \widetilde{L}_i \widetilde{L}_i^\top + \widetilde{A}_i$,

Добавить $\widetilde{L}_i$ в матрицу $\check{L}_j$.

$Q_j = \text{diag}(\widetilde{U}_1, \ldots, \widetilde{U}_{M_j})$.

$\check{A}_j = \widetilde{A}_{M_j}$

Output : $\check{A}_j$, $\check{L}_j$, Q_j

2.2. Оценка сложности CE алгоритма

2.2.1. Сложность CE алгоритма через блочный шаблон разреженности матрицы A

Изучим шаблон разреженности факторов L_1 и A_1 после исключения первого уровня. Для начала определим блочный шаблон разреженности блочной разреженной матрицы.

Определение* 2.2.1** (Блочный шаблон разреженности)**.** *Для матрицы A с M блочными строками, M блочными столбцами и размером блока $B \times B$ определим* ***bsp$(A)_{B\times B}^{M\times M}$ *(блочный шаблон разреженности) как функцию*

$$F_A(i,j) \to \{0,1\}, \forall i,j = \overline{1 \dots M}$$

которая возвращает 1 *когда блок* A_{ij} *ненулевой и* 0 *в противном случае.*

***Определение* 2.2.2.** *Через*

$$\#\textbf{\textit{bsp}}(A)_{B\times B}^{M\times M},$$

обозначим число ненулевых блоков $(B \times B)$ *матрицы* A*.*

***Теорема* 2.2.3.** *Алгоритм компрессии и исключения (CE) приводит к приближенной факторизации*

$$A \approx QLL^\top Q^\top,$$

где Q *- это унитарная матрица равная произведению блочно-диагональных матриц и матриц перестановки*

$$Q = \left(\prod_{j=1}^{K} \breve{P}_j \breve{Q}_j \right) P,$$

L *- это разреженная блочно нижне-треугольная матрица*

$$L = \begin{bmatrix} \begin{bmatrix} \\ L_1 \\ \\ \end{bmatrix} & 0 & 0 \\ & \begin{bmatrix} \\ L_2 \\ \\ \end{bmatrix} & \vdots \\ \dots & & 0 \\ & & \begin{bmatrix} L_{K+1} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} L_1 & \breve{L}_2 & \dots & \breve{L}_{K+1} \end{bmatrix},$$

в которой блок L_i *содержит*

$$\#L_j \leqslant 2\#\textbf{\textit{bsp}}(A_{j-1})_{rJ\times rJ}^{M/J^{j-1}\times M/J^{j-1}}$$

ненулевых блоков, и общее число ненулевых блоков в факторе L *равно*

$$\#L \leqslant 2\#\textbf{\textit{bsp}}(A_0)_{B\times B}^{M\times M} + 2\left(\sum_{j=1}^{K-1} \#\textbf{\textit{bsp}}(A_0^{2^j})_{BJ^j\times BJ^j}^{M/J^j\times M/J^j} \right) + \\ + \frac{1}{2}(M/J^K)^2 r^2. \qquad (2.10)$$

K - это число уровней в CE алгоритме, B это размер блока. Затраты по времени для приближенной факторизации CE составляют $\mathcal{O}(B^3(\#L_{K+1} - \#L_1) + B^2\#L)$. Затраты по памяти составляют $\mathcal{O}(B(B-r)\#L + NB)$.

Доказательство. Для простоты предположим что на первом уровне исключения на каждом шаге мы исключали $(B - r)$ строк. В данном доказательстве предполагается случай фиксированного ранга r и фиксированного размера блока B. Алгоритм с адаптивным выбором ранга и с неравномерным размером блока требует дополнительного исследования. Для матрицы $\check{L}_1$, В соответствии с уравнениями (2.5) и(2.6) получаем что

$$\mathbf{bsp}(\check{L}_1)^{M\times M}_{B\times(B-r)} \leqslant \mathbf{bsp}(A_0)^{M\times M}_{B\times B}.$$

Матрица перестановки P_1 не меняет количество ненулевых элементов, таким образом $L_1 = P_1\check{L}_1$ имеет такой же блочный шаблон разреженности как и A_0. Так как перестановка P_1 делит блоки матрицы L_1 на “исключенные” и “не исключенные” части,

$$\#\mathbf{bsp}(L_1)^{M\times M}_{(B-r)\times(B-r) \text{ or } r\times(B-r)} \leqslant 2\#\mathbf{bsp}(\check{L}_1)^{M\times M}_{B\times(B-r)} \leqslant 2\#\mathbf{bsp}(A_0)^{M\times M}_{B\times B}. \qquad (2.11)$$

В отличие от матрицы L_1, блочный шаблон разреженности матрицы A_1 изменяется более сложным образом. Так как матрица A_1 это стартовая точка для следующего уровня исключения, очень важно оценить её разреженность. Благодаря (2.3) можем сказать что

$$\mathbf{bsp}(A_1)^{M\times M}_{r\times r} \leqslant \mathbf{bsp}(A_0^2)^{M\times M}_{B\times B}.$$ [1]

Ключевая часть алгоритма это то, что для следующего уровня исключения мы соединяем блоки матрицы A_1 в супер-блоки размера $rJ \times rJ$ ($Jr \approx B$). Затем рассматриваем новую матрицу с блоками размера M/J (предположим что M делится на J). Число J является внутренним параметром алгоритма. Новая матрица рассматривается как матрица с $MJ \times MJ$ блоков. (Новый размер блока Jr.[2]) Получаем что

$$\mathbf{bsp}(A_1)^{(M/J)\times(M/J)}_{rJ\times Jr} \leqslant \mathbf{bsp}(A_0^2)^{(M/J)\times(M/J)}_{BJ\times BJ}. \qquad (2.12)$$

[1] Под $\mathbf{bsp}(A_1)^{M\times M}_{r\times r} \leqslant \mathbf{bsp}(A_0^2)^{M\times M}_{B\times B}$ мы понимаем следующее: $F_{A_1}(i,j) \leqslant F_{A^2}(i,j), \forall i,j = \overline{1\ldots M}$.

[2] Мы можем легко адаптировать алгоритм для переменного размера блоков, но для простоты считаем их равными.

В данный момент играет роль выбор изначальной перестановки. Исключение блоков увеличивает шаблон разреженности, процедура укрупнения в хорошо переставленной матрице может восстановит хорошую разреженность. Эта процедура подробно описана в параграфе 2.2.2.

После исключения K уровней,

$$\breve{P}_K\breve{Q}_K \dots \breve{P}_1\breve{Q}_1\, A_0\, \breve{Q}_1^\top \breve{P}_1^\top \dots \breve{Q}_K^\top \breve{P}_K^\top \approx \begin{bmatrix} L_1 & \breve{L}_2 & \dots & \breve{L}_K \end{bmatrix} + \begin{bmatrix} 0_{n_* \times n_*} & 0 \\ 0 & A_K \end{bmatrix}, \quad (2.13)$$

где $n_* = \sum_{j=1}^{K} n_{L_j}$. Учитывая Уравнение (2.1), обозначим

$$Q = \left(\prod_{j=1}^{K} \breve{P}_j \breve{Q}_j \right) P,$$

Остаток факторизуется точно как

$$A_K = L_{K+1} L_{K+1}^\top.$$

Обозначим

$$L = \begin{bmatrix} \begin{bmatrix} L_1 \\ \\ \end{bmatrix} & \begin{matrix} 0 \\ \begin{bmatrix} L_2 \\ \\ \end{bmatrix} \end{matrix} & \dots & \begin{matrix} 0 \\ \vdots \\ 0 \\ \begin{bmatrix} L_{K+1} \end{bmatrix} \end{matrix} \end{bmatrix} = \begin{bmatrix} L_1 & \breve{L}_2 & \dots & \breve{L}_{K+1} \end{bmatrix},$$

где L_j это прямоугольные блочно нижне-треугольные матрицы. Вычислим

$$\#L = \sum_{j=1}^{K+1} \#L_j.$$

Используя (2.11),

$$\#L_1 \leqslant 2\#\mathbf{bsp}(A_0)_{B\times B}^{M/J \times M/J},$$

для $j = 2 \dots K$:

$$\#L_j \leqslant 2\#\mathbf{bsp}(A_{j-1})_{rJ\times rJ}^{M/J^{j-1} \times M/J^{j-1}},$$

и для $j = K+1$:

$$\#L_{K+1} \leqslant \frac{1}{2}(M/J^K)^2 r^2.$$

Используя Уравнение (2.12) получаем что

$$\mathbf{bsp}(A_K)^{(M/J^K)\times(M/J^K)}_{r\times r} \leqslant \mathbf{bsp}(A^2_{K-1})^{(M/J^K)\times(M/J^K)}_{rJ\times rJ} \leqslant \dots$$

$$\dots \leqslant \mathbf{bsp}(A_0^{2^K})^{(M/J^K)\times(M/J^K)}_{BJ^K\times BJ^K}.$$

В итоге получаем уравнение (2.10).

В алгоритме CE требуется хранить факторы Q и L. Согласно уравнению (3.10), фактор Q это произведение K матриц перестановки $\breve{P}_1, \dots, \breve{P}_K$ и K блочно-диагональных матриц $\breve{Q}_1, \dots, \breve{Q}_K$ с размером блока B и числом блоков уменьшающемся на каждом шаге в J раз. Так как

$$\sum_{k=0}^{\infty} \frac{N}{J^k} B = NB\frac{J}{J-1}, \tag{2.14}$$

Матрица Q требует для хранения $\mathcal{O}(NB)$ ячеек памяти. Память, требуемая для фактора L составляет $\mathcal{O}(B(B-r)\#L)$ ячеек памяти. Следовательно, факторизация требует $\mathcal{O}(B(B-r)\#L + NB)$ ячеек памяти.

Оценим число операций, требуемых для CE факторизации. Факторизация CE выполняется за K проходов, каждый проход состоит из шагов компрессии и исключения. Вычислим сложность i-го прохода. Шаг компрессии требует факторизации SVD для дальних блоков и умножения блочных строк и столбцов на левый фактор SVD разложения. Матрица A_i имеет $(\#L_{i+2} - \#L_{i+1})$ дальних блоков размера $B \times B$, следовательно SVD разложение требует $\mathcal{O}(B^3(\#L_{i+2} - \#L_{i+1}))$ операций. Умножение матрицы A_i на унитарную блочно-диагональную с размером блока B (так как матрица A_i содержит $\#L_{i+1}$ блоков) требует $\mathcal{O}(B^2\#L_{i+1})$ операций. Процедура исключения для всех строк матрицы A_i требует $\mathcal{O}((B-r)^2\#L_{i+1})$ операций. Поэтому, i-ый проход алгоритма CE требует $\mathcal{O}\left((B)^3(\#L_{i+1} - \#L_i) + B^2\#L_{i+1}\right)$ операций. Общая вычислительная сложность K проходов алгоритма CE составляет

$$t = \sum_{i=0}^{K} \left(\mathcal{O}(B^3(\#L_{i+2} - \#L_{i+1}) + B^2\#L_{i+1})\right) =$$

$$\mathcal{O}\left(B^3(\#L_{K+1} - \#L_1)\right) + B^2 \sum_{i=0}^{K} \mathcal{O}(\#L_{i+1}),$$

или

$$t = \mathcal{O}\left(B^3(\#L_{K+1} - \#L_1) + B^2\#L\right).$$

□

***Теорема* 2.2.4.** *Система $Ax = b$ с вычисленной CE факторизацией матрицы A ($A = Q^\top LL^\top Q$) может быть решена за $\mathcal{O}((B-r)B\#L + NB)$ операций.*

Доказательство. Так как

$$Q^\top LL^\top Qx = b,$$

следовательно,

$$x = Q^\top L^{-\top} L^{-1} Qb.$$

Требуется вычислить произведение матриц Q и $Q^\top$ на вектор. В соответствии с (3.10), матрицы Q и $Q^\top$ являются произведениями K перестановок $\breve{P}_1, \ldots, \breve{P}_K$ и K блочно-диагональных матриц $\breve{Q}_1 \ldots \breve{Q}_K$ с размером блока B и числом блоков, уменьшающимся на каждом шаге в J раз. Учитывая уравнение (2.14), общая вычислительная сложность умножения матриц Q и $Q^\top$ на вектор составляет $\mathcal{O}(NB)$ операций.

Так как L это блочно-треугольная разреженная матрица с $\#L$ ненулевыми блоками размера $(B-r) \times B$, решение системы с матрицей L требует $\mathcal{O}((B-r)B\#L)$ операций. Аналогично, решение системы с матрицей $L^\top$ требует $\mathcal{O}((B-r)B\#L)$ операций. Общая вычислительная сложность решения системы $Q^\top LL^\top Qx = b$ составляет $\mathcal{O}((B-r)B\#L + NB)$ операций. □

2.2.2. Оценка сложности алгоритма CE на основе анализа графов

Рассмотрим ненаправленный граф $\mathcal{G}_{A_0}$ ассоциированный с блочным шаблоном разреженности симметричной матрицы A: i-я вершина графа соответствует i-ым блочной строке и блочному столбцу, если блок (i,j) ненулевой, тогда i-я и j-я вершины графа связаны ребром. В этом параграфе мы проясним связь между свойствами графа $\mathcal{G}_{A_0}$ и сложность алгоритма CE.

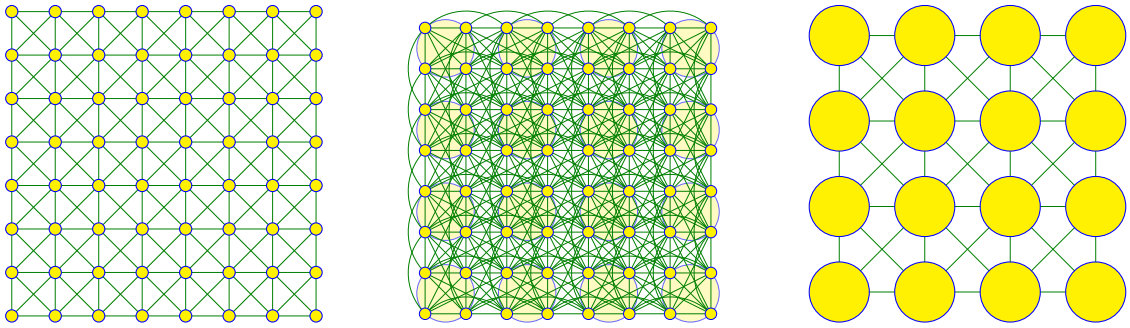
Рассмотрим, например, матрицу A с графом $\mathcal{G}_{A_0}$ изображенном на Рисунке 2.8a. Эта матрица могла быть получена из двумерного эллиптического уравне-

ния дискретизованного на равномерной квадратной сетке в области $[0,1]^2$ с помощью конечно-разностной схемы с девятиточечным шаблоном. Отметим, что каждая вершина графа соединена только с фиксированным числом ближайших пространственных соседей (будем называть это свойство *локальностью графа*.)

В соответствии с (2.12), один проход CE алгоритма приводит к возведению в квадрат блочного шаблона разреженности остатка. Возведение матрицы A в квадрат портит локальность графа $\mathcal{G}_{A_0}$: если две вершины были соединены с одной и той же вершиной в графе $\mathcal{G}_{A_0}$, они становятся соединены в графе $\mathcal{G}_{A^2}$ (смотри граф $\mathcal{G}_{A^2}$ на Рисунке 2.8b). Мы называем этот процесс «возведение в квадрат» $\mathcal{G}_{A^2} = \mathbf{Sq}(\mathcal{G}_{A_0})$). Отметим, что возведение в квадрат значительно увеличивает число вершин в графе $\mathcal{G}_{A^2}$.

Следующий шаг CE разложения это обьединение блочных строк и столбцов в супер-блоки по J штук. Получаем матрицу A_1 из уравнения (2.7). Для графа $\mathcal{G}_{A^2}$ это означает объединение вершин в супер-вершины по J штук. Мы называем этот шаг «укрупнением» $\mathcal{G}_{A_1} = \mathbf{Coars}(\mathcal{G}_{A^2})$. Укрупненный граф $\mathcal{G}_{A_1}$ обладает лучшей локальностью чем граф $\mathcal{G}_{A^2}$.

В данном конкретном примере структура графа для матрицы A и для возведенной в квадрат и укрупненной матрицы A_1 очень похожа, смотри Рисунок 2.8c. В ходе алгоритма CE мы получаем графы $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$, где $\mathcal{G}_{A_i} = \mathbf{Coars}(\mathbf{Sq}(\mathcal{G}_{A_{i-1}})), \quad \forall i = 1, \ldots, K.$



(a) Граф $\mathcal{G}_{A_0}$

(b) Граф $\mathcal{G}_{A_0^2}$ матрицы возведенной в квадрат

(c) Граф $\mathcal{G}_{A_1}$ укрупненной матрицы

Рисунок 2.8: Пример процедуры возведения в квадрат и укрупнения для графа $\mathcal{G}_{A_0}$

Отметим, что число ребер в графах $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$ связано с числом $\#L$, которое требуется оценить, смотри Утверждение 2.2.5.

***Утверждение* 2.2.5.** *Число $\#L$ ненулевых блоков в факторе L не превосходит общего числа рёбер в графах $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$:*

$$\#L \leqslant \sum_{i=0}^{K} \textbf{Edge\_num}(\mathcal{G}_{A_i}).$$

Доказательство. Число рёбер в графе $\mathcal{G}_{A_i}$ равно числу ненулевых блоков в матрице A_i $\forall i \in 0 \ldots K$ по определению. С учетом уравнения (2.10) получаем:

$$\begin{aligned} \#L \leqslant 2\#\textbf{bsp}(A)_{B\times B}^{M\times M} + 2\left(\sum_{j=1}^{K-1} \#\textbf{bsp}(A^{2^j})_{BJ^j\times BJ^j}^{M/J^j\times M/J^j}\right) + \\ +\frac{1}{2}(M/J^K)^2 r^2 = \textbf{Edge\_num}(\mathcal{G}_{A_0}) + \sum_{i=1}^{K} \textbf{Edge\_num}(\mathcal{G}_{A_i}). \end{aligned} \tag{2.15}$$

□

Таким образом, минимизация числа $\#L$ эквивалентна минимизации общего числа ребер в графах $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$.

Процедура укрупнения это ключ к уменьшению числа ребер. Отметим, что процедура укрупнения для всех графов $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$ может быть определена единственной перестановкой P и числом вершин объединяемый вместе J (соседи в перестановке P объединяются в группы по J). Также отметим, что с заданным J, процедура укрупнения зависит от изначальной перестановки P. Таким образом, чтобы минимизировать $\#L$, вместо оптимизации перестановки P можно оптимизировать процедуру укрупнения, что может быть гораздо более удобно.

Рассмотрим пример приведённый на Рисунке 2.8. Следующее утверждение показывает, что для данного примера хорошее укрупнение существует и является очень простым

***Утверждение* 2.2.6.** *Пусть граф $\mathcal{G}_{A_0}$ определен тензорной сеткой в $\mathbb{R}^2$, (точки сетки совпадают с вершинами графа $\mathcal{G}_{A_0}$, ребра сетки совпадают с ребрами графа. $\mathcal{G}_{A_0}$), как в примере на Рисунке 2.8. Если процедура укрупнения объединяет ближайшие вершины и каждая супер-вершине имеет минимум две вершины в каждом направлении, тогда каждый граф $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \ldots, \mathcal{G}_{A_K}$ имеет $\mathcal{O}(N)$ вершин.*

Доказательство. Рассмотрим граф $\mathcal{G}_{A_0}$ определенный тензорной сеткой в $\mathbb{R}^2$, пусть каждая вершина n этого графа имеет индекс (i, j). Пусть $s(e)$ это длинна вершины e которая соединяет вершины n_1 и n_2 с индексами (i_1, j_1) и (i_2, j_2) если

$$s(e) = \max(|i_1 - i_2|, |j_1 - j_2|).$$

Пусть $\tilde{s}(\mathcal{G}_{A_0})$ это максимальная длинна ребра в графе $\mathcal{G}_{A_0}$:

$$\tilde{s}(\mathcal{G}_{A_0}) = \max_{e_i \in \mathcal{G}_{A_0}} s(e_i).$$

Для графа $\mathcal{G}_{A_i}$матрицы A_i, $\forall i \in 0 \dots (K-1)$, $\tilde{s}(\mathcal{G}_{A_i}) = 1$. В соответствии с процедурой возведения в графа квадрат, $\mathcal{G}_{A_i^2} = \textbf{Sq}(\mathcal{G}_{A_l})$, $\tilde{s}(\mathcal{G}_{A_i^2}) = 2$. Докажем, что после процедуры укрупнения описанной в условии ($\mathcal{G}_{A_{i+1}} = \textbf{Coars}(\mathcal{G}_{A_i^2})$) мы получим $\tilde{s}(\mathcal{G}_{A_{i+1}}) = 1$. If $\tilde{s}(\mathcal{G}_{A_{i+1}}) > 1$, тогда, так как каждая супер-вершина имеет как минимум 2 вершины в каждом направлении, $\tilde{s}(\mathcal{G}_{A_i^2}) > 3$, это противоречие. Таким образом, $\tilde{s}(\mathcal{G}_{A_{i+1}}) = 1$.

Если $\tilde{s}(\mathcal{G}_{A_{i+1}}) = 1$, каждая вершина обладает константным числом присоединённых ребер. Тогда, число ребер в графе $\mathcal{G}_{A_i}, \forall i \in 0 \dots (K-1)$ это $\mathcal{O}(N)$. □

***Следствие* 2.2.7.** *Утверждение 2.2.6 также верно для* $\mathbb{R}^d, d > 2$.

Доказательство. Аналогично Утверждению 2.2.5. □

***Следствие* 2.2.8.** *Пусть матрица* $A_{\textbf{tpg}}$ *имеет граф порожденный тензорной сеткой в* $\mathbb{R}^d$. *Факторизация CE матрицы* $A_{\textbf{tpg}}$ *требует* $\mathcal{O}(B^3N + B^2NK)$ *операций и* $\mathcal{O}(B(B-r)NK)$ *ячеек памяти.*

Доказательство. В соответствии с Утверждением 2.2.6, каждый граф $\mathcal{G}_{A_0}, \mathcal{G}_{A_1}, \dots, \mathcal{G}_{A_K}$ имеет $\mathcal{O}(N)$ ребер. Тогда, по Утверждению 2.2.5, $\#L = \mathcal{O}(NK)$, $(\#L_{K+1} - \#L_1) = \mathcal{O}(N)$. Следовательно, по Утверждению 2.2.5, затраты по памяти для CE факторизации матрицы $A_{\textbf{tpg}}$ следующие:

$$\textbf{mem} = \mathcal{O}(B(B-r)\#L + NB) = \mathcal{O}(B(B-r)NK + NB) = \mathcal{O}(B(B-r)NK),$$

и вычислительная сложность:

$$\textbf{t} = \mathcal{O}(B^3(\#L_{K+1} - \#L_1) + B^2\#L) = \mathcal{O}(B^3N + B^2NK).$$

□

Отметим, что алгоритмы разбиения графа [62,64] и, в частности, процедура вложенного биения [30] может быть использована в для формирования хорошо локализованных супер-вершин в ходе процедуры укрупнения.

Обсудим выбор процедуры укрупнения в случае геометрических графов. Рассмотрим граф k-ближайших соседей [57]. Этот граф имеет геометрическую интерпретацию в которой каждая вершина имеет максимум k присоединенных ребер и имеет сферу которая содержит все присоединенные вершины и только их. Например, граф $\mathcal{G}_{A_0}$ на Рисунке 2.8b это граф 8-и ближайших соседей. Предлагаем следующую гипотезу.

***Гипотеза* 2.2.9.** *Пусть граф $\mathcal{G}_{A_0}$ это граф k-ближайших соседей; пусть $h_{\min}$ и $h_{\max}$ это максимальная и минимальная длины ребер; пусть B это максимальный размер блока. Пусть графы $\mathcal{G}_{A_1}$, ...$\mathcal{G}_{A_K}$ получены в ходу K шагов возведения в квадрат и укрупнения. Тогда существует процедура укрупнения и число*

$$k_1 = k_1\left(\frac{h_{\max}}{h_{\min}}, d, B\right),$$

такое что граф $\mathcal{G}_{A_1}$, ...$\mathcal{G}_{A_K}$ это граф k_1-ближайших соседей.

***Замечание* 2.2.10.** *Граф k-ближайших соседей имеет $\mathcal{O}(N)$ ребер если k не зависит от N. Таким образом, Гипотеза 2.2.9 может быть переформулирована следующим образом: если $\mathcal{G}_{A_0}$ это граф k-ближайших соседей, тогда существует процедура укрупнения такая, что каждый из графов $\mathcal{G}_{A_1}$, ...$\mathcal{G}_{A_K}$ имеет $\mathcal{O}(N)$ вершин. Таким образом, Аналогично Следствию 2.2.8, CE факторизация матрицы с графом k-ближайших соседей требует $\mathcal{O}(B^3N + B^2NK)$ операций и $\mathcal{O}(B(B-r)NK)$ ячеек памяти.*

***Замечание* 2.2.11.** *Програмная реализация, тестирование и сравнение данного алгоритма с другими методами решения разреженных линейных систем приведены в главе 4, в разделе 4.1.*

2.3. Выводы по главе

В данной главе автором предложена новая приближенная факторизация разреженных и блочно-малоранговых матриц (метод компрессии и исключения,

compress and eliminate method, CE метод), приводящая к приближенному прямому методу решения разреженных и блочно-малоранговых матриц а также к предобуславливателями для итерационных методов решения. Проведена оценка вычислительной сложности предложенного алгоритма. Также предложен способ улучшения скорости CE алгоритма путем поиска оптимальной изначальной перестановки матрицы.

Глава 3

Методы разреженной факторизации малопараметрических матриц

В данной главе автором предложены два метода разреженной факторизации $\mathcal{H}^2$ матриц. $\mathcal{H}^2$ матрицей называют малопараметрический формат хранения плотной матрицы (поэтому в названии главы она названа малопараметрической). $\mathcal{H}^2$ матрица обладает свойством быстрого $\mathcal{O}(N)$ умножения матрицы на вектор, что позволяет эффективно решать системы с ними при помощи итерационных методов, однако предобуславливание систем с $\mathcal{H}^2$ матрицами является сложной и нерешенной пока задачей. Переход от $\mathcal{H}^2$ матрицы к разреженной - это очень естественный шаг (так как обе формы требуют хранения $\mathcal{O}(N)$ элементов), который позволяет строить эффективные предобуславливатели применяя для решения поправочных систем классические методы решения систем с разреженными матрицами.

3.1. Метод построения расширенной разреженной матрицы

В данном разделе мы используем наблюдение, что классическое трехшаговое матрично-векторное произведение может быть переписано в виде боль-

шой линейной системы. Матрицу этой системы будем называть **разреженно-расширенная** (sparse extended, SE) матрица. Похожие идеи представлены в [18, 44, 66, 85]. На основе предложенной SE матрицы построен ряд методов решения систем с $\mathcal{H}^2$ матрицами.

3.1.1. Обозначения и базовые понятия

Коротко напомним основные свойства $\mathcal{H}^2$ матриц, которые будут необходимы в дальнейших рассуждениях. Подробные сведения можно найти в параграфе 1.2.2.

Прежде всего $\mathcal{H}^2$ матрица - это блочная матрица с размером блока B иерархическим укрупнением блоков, таким что на k-м уровне размер блока B_k равен

$$B_k = 2^k B, \quad k \in 0, \dots, L$$

На каждом уровне $k \in 0, \dots, L$ матрица представима в виде суммы непересекающихся **ближней** и **дальней** матриц:

$$A = C_k + F_k, \quad k \in 0, \dots, L$$

где $C_k \in \mathbb{R}^{N \times N}$ это блочно-разреженная матрица ближних блоков, $F_k \in \mathbb{R}^{N \times N}$ это блочная матрица дальних блоков такая, что каждая её блочная строка и каждый блочный столбец имеет малый ранг. Иерархическое разбиение строк и столбцов хранятся в виде кластерных деревьев. Матрицы дожимания строк и столбцов хранятся в виде матриц переходов между уровнями одного дерева, матрицы $(C_i - C_{i-1})$ хранятся в виде матриц перехода с дерева столбцов на дерево строк. Матрица C_0 хранится в виде отдельного оператора ближнего взаимодействия.

Ключевую роль для данного раздела играет алгоритм умножения $\mathcal{H}^2$ матрицы на вектор [14]. Рассмотрим вектор $x \in \mathbb{R}^{N \times 1}$ и $\mathcal{H}^2$ матрицу $A \in \mathbb{R}^{N \times N}$ и найдем их матрично-векторное произведение $y = Ax$. Напомним, что $\mathcal{H}^2$ матрица определяется кластерным деревом строк, кластерным деревом столбцов и следующими наборами матриц: $R = (R_i),\ i \in \mathcal{T}_c$ это переходы вверх по дереву столбцов, $C = (C_p),\ p = (i,j) \in \mathbb{P}_{close},\ i \in \mathcal{T}_c,\ j \in \mathcal{T}_r$ это ближние матрицы, $S = (S_p),\ p = \{(i,j) | i \in \mathcal{T}_c,\ j \in \mathcal{T}_r, (i,j) \in \mathbb{P}_{far}$ и $(\text{father}(i), \text{father}(j)) \in \mathcal{P}_{close}\}$ это переходы с дерева столбцов на дерево строк, $L = (L_j),\ j \in \mathcal{T}_r$, это переходы вниз по дереву строк. Формальное описание алгоритма приведено в 5.

Промежуточные переменные связанные с вершинами дерева столбцов будем называть $\hat{x} = (x_i), i \in \mathcal{T}_c$, переменные связанные с вершинами дерева строк будем называть $\hat{y} = (y_i), i \in \mathcal{T}_r$.

Удобно распределить вектор x между всеми вершинами кластерного столбцового дерева. Получившиеся в результате работы алгоритма промежуточные значения в листьевых вершинах дерева строк так же распределяются с листьев на вектор ответа y. Обратите внимание, что данные операции перехода от источников к листьям и от листьев к приёмникам рассматриваются как отдельные операторы E и D. Таким образом появляются два вспомогательных набора матриц $E = (E_i)$, $i \in$ **leaves**$(\mathcal{T}_c)$ это переходы с вектора источников x на листьевые вершины дерева столбцов, $D = (D_i)$, $i \in$ **leaves**$(\mathcal{T}_r)$ это переходы с листьевых вершин дерева строк на вектор приёмников y.

Алгоритм 3: Проход вверх по дереву столбцов

Проход вверх$(i, R, \hat{x})$

if sons$(i) \neq 0$ **then**

  $\hat{x}_{father(i)} := R_i \hat{x}_i$

else

  $\hat{x}_i := 0$

  for $j \in$ sons(i) **do**

    $\hat{x_j} :=$ **Проход вверх**$(j, R, \hat{x})$

    $\hat{x_i} := \hat{x_i} + R_j x_j$

Вернуть: $\hat{x}$

Алгоритм 4: Проход вниз по дереву строк

Проход вниз$(i, L, \hat{y})$

if sons$(i) \neq 0$ **then**

  $\hat{y}_{father(i)} := L_i \hat{y}_i$

else

  for $j \in$ sons(i) **do**

    $\hat{y_j} := \hat{y_j} + L y_i$

    $\hat{y} :=$ **Проход вниз**$(j, L, \hat{y})$

Вернуть: $\hat{y}$

Алгоритм 5: Умножение $\mathcal{H}^2$ матрицы на вектор

Дано: $\mathcal{H}^2$ матрица $A = \{\mathcal{T}_r, \mathcal{T}_c, \text{D, R, C, S, L, E}\}$, vector x

$\hat{x} = 0, \hat{y} = 0$

for $i \in \mathcal{T}_c$, $\text{sons}(i) = 0$ **do**
  $\hat{x}_i := E_i x_i$

$\hat{x} :=$ **Проход вверх**$(root(\mathcal{T}_c), R, \hat{x})$

for $i \in \mathcal{T}_c$ **do**
  for $j \in \mathcal{T}_r$ **do**
    if $(i, j) \in \mathcal{P}_{far}$ **then**
      if $(\text{father}(i), \text{father}(j)) \in \mathcal{P}_{close}$ **then**
        $\hat{y}_j += S_i \hat{x}_i$

$\hat{y} :=$ **Проход вниз**$(root(\mathcal{T}_c), L, \hat{y})$

for $i \in \mathcal{T}_c$, $\text{sons}(i) = 0$ **do**
  $y_i := D_i \hat{y}_i$

Вернуть: y

На Рисунке 3.1 приведена иллюстрация процесса матрично-векторного произведения.

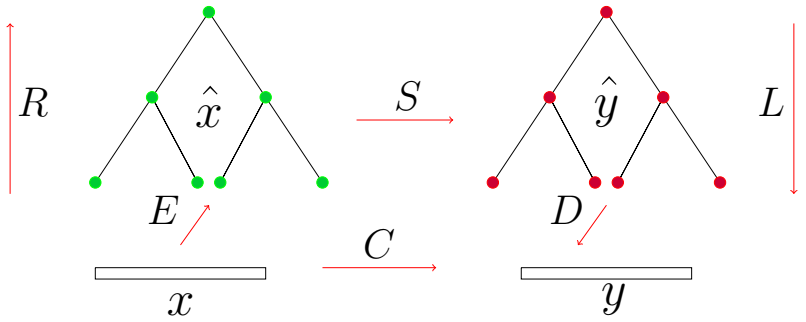


Рисунок 3.1: Умножение $\mathcal{H}^2$ матрицы на вектор

Отметим, что сложность данного алгоритма это $\mathcal{O}(N)$ и требуемая память также $\mathcal{O}(N)$.

3.1.2. Основная идея

Наше основное наблюдение заключается в том, что Алгоритм 5 может быть переписан как произведение разреженной матрицы на вектор (размер полученной разреженной матрицы больше размера исходной). Первый шаг алгоритма Алго-

ритма 5 может быть переписан в виде умножения разреженной матрицы на вектор

$$Dx = \hat{x}_l,$$

где $\hat{x}_l$ это промежуточные переменные в листьевых вершинах дерева столбцов,

$$D = \begin{bmatrix} D_1 & 0 & 0 & 0 \\ 0 & D_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & D_{N_{lc}} \end{bmatrix}, \tag{3.1}$$

где N_{lc} это количество листьевых вершин в дереве $\mathcal{T}_c$, нули представляют нулевые матрицы соответствующих размеров. Следующий шаг представим в виде

$$R\hat{x} = \hat{x}_n,$$

где $\hat{x}_n$ это промежуточные переменные в нелистьевых вершинах дерева столбцов,

$$R = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & R_{N_{nc}} \end{bmatrix}, \tag{3.2}$$

где N_{nc} это число нелистьевых вершин в дереве $\mathcal{T}_c$ нули представляют нулевые матрицы соответствующих размеров. Третий шаг матрично-векторного алгоритма эквивалентен уравнению

$$L\hat{y}_n + S\hat{x} = \hat{y},$$

где $\hat{y}_n$ это промежуточные переменные в нелистьевых вершинах дерева строк,

$$L = \begin{bmatrix} L_1 & 0 & 0 & 0 \\ 0 & L_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & L_{N_{nr}} \end{bmatrix}, \quad S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1N_c} \\ S_{21} & S_{22} & \cdots & S_{2N_c} \\ \vdots & \vdots & \ddots & \\ S_{N_r1} & S_{N_r2} & & S_{N_rN_c} \end{bmatrix}, \tag{3.3}$$

где N_r количество вершин в дереве $\mathcal{T}_r$, N_c количество вершин в дереве $\mathcal{T}_c$, N_{nr} это количество нелистьевых вершин в дереве $\mathcal{T}_r$, $S_{ij} = 0$ если $i \in \mathcal{T}_r$ $j \in \mathcal{T}$ $(i, j) \in \mathcal{P}_{close}$ или $(\text{father}(i), \text{father}(j)) \in \mathcal{P}_{far}$. Последний шаг переписываем в виде

$$y = E\hat{y}_l + Cx,$$

где $\hat{y}_l$ это промежуточные переменные в листьевых вершинах дерева строк,

$$E = \begin{bmatrix} E_1 & 0 & 0 & 0 \\ 0 & E_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & E_{N_{lr}} \end{bmatrix}, \tag{3.4}$$

где N_{lr} это количество листьевых вершин в дереве $\mathcal{T}_r$. Записывая все неравенства вместе:

$$\begin{cases} Dx = \hat{x}_l \\ R\hat{x} = \hat{x}_n \\ L\hat{y}_n + S\hat{x} = \hat{y} \\ E\hat{y}_l + Cx = y \end{cases}, \tag{3.5}$$

или в блочной форме:

$$\begin{bmatrix} C & 0 & 0 & E \\ 0 & S & L & 0 \\ 0 & R & 0 & 0 \\ D & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \\ \hat{y}_n \\ \hat{y}_l \end{bmatrix} = \begin{bmatrix} y \\ \hat{y} \\ \hat{x}_n \\ \hat{x}_l \end{bmatrix}, \tag{3.6}$$

Обозначим полученную матрицу H_0, и также напомним, что векторы

$$\hat{x} = \begin{bmatrix} \hat{x}_n \\ \hat{x}_l \end{bmatrix}, \quad \hat{y} = \begin{bmatrix} \hat{y}_n \\ \hat{y}_l \end{bmatrix}.$$

В итоге получаем:

$$H_0 \begin{bmatrix} x \\ \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} y \\ \hat{y} \\ \hat{x} \end{bmatrix}. \tag{3.7}$$

Выше описан алгоритм матрично-векторного умножения $y = Ax$, где при известном x требовалось найти y. Перейдем от этой задачи к задаче решения системы линейных уравнений, где при заданном y требуется найти x. Отметим, что в правую часть системы (3.7) в таком случае входят неизвестные переменные, преобразуем систему (3.7) к виду с полностью известной правой частью:

$$\left(H_0 + \begin{bmatrix} 0^{N \times N} & 0 & 0 \\ 0 & 0 & -I^{N_{\hat{y}} \times N_{\hat{y}}} \\ 0 & -I^{N_{\hat{x}} \times N_{\hat{x}}} & 0 \end{bmatrix} \right) \begin{bmatrix} x \\ \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix},$$

где $N_{\hat{x}} = len(\hat{x})$, $N_{\hat{y}} = len(\hat{y})$ и

$$H = H_0 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -I \\ 0 & -I & 0 \end{bmatrix}.$$

Итоговая система уравнений представляется в виде:

$$H \begin{bmatrix} x \\ \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix}. \tag{3.8}$$

Теперь правая часть системы (3.8) содержит только известные переменные и мы можем решить её и найти x, где x это решение $Ax = y$. Будем называть матрицу H **разреженно-расширенная** (sparse extended, SE) матрица.

3.1.3. Свойства SE матрицы

Важным свойством SE матрицы является то, что если исходная $\mathcal{H}^2$ матрица невырожденная, то и её SE матица также невырожденная.

***Теорема* 3.1.1.** *Если матрица $A \in \mathbb{R}^{N \times N}$ это невырожденная $\mathcal{H}^2$ матрица, тогда $H = \mathrm{SE}(A) \in \mathbb{R}^{N_H \times N_H}$ также невырожденная, кроме того $N_H < (2k+1)N$, где k это максимум из количества уровней в кластерных деревьях $\mathcal{T}_c$ и $\mathcal{T}_r$.*

Доказательство. Для начала докажем что матрица H квадратная. Матрица A квадратная, следовательно $\mathrm{len}(x) = \mathrm{len}(y) = N$. Пусть N_r это число сток в матрице H, а N_c это число столбцов. Тогда,

$$N_r = \mathrm{len}(x) + \mathrm{len}(\hat{x}) + \mathrm{len}(\hat{y}) = \mathrm{len}(y) + \mathrm{len}(\hat{y}) + \mathrm{len}(\hat{x}) = N_c$$

Таким образом H квадратная матрица. Обратите внимание, что $\mathrm{len}(\hat{x}) \leq k_1 N$, $\mathrm{len}(\hat{y}) \leq k_2 N$, где k_1 и k_2 это количества уровней в кластерных деревьях $\mathcal{T}_c$ и $\mathcal{T}_r$. Таким образом

$$N_H = \mathrm{len}(x) + \mathrm{len}(\hat{x}) + \mathrm{len}(\hat{y}) = N + k_1 N + k_2 N < (2k+1)N$$

Теперь докажем что H невырожденная. Предположим что $Hz = 0$ и докажем что $z = 0$. Из построения SE матрицы следует, что первая блочная компонента

вектора z удовлетворяет $Ax = 0$, значит $x = 0$. В соответствии с Алгоритмом 5 и уравнением (3.5), $x = 0$ означает что $\hat{x} = 0$ и $\hat{y} = 0$ следовательно $z = 0$ и ядро H вырожденное. □

Обратите внимание, что число обусловленности матрицы SE обычно намного больше числа обусловленности исходной матрицы, потому нужны некоторые специальные методы решения систем с ней.

3.1.4. Методы решения основанные на SE форме

Как можно использовать SE матрицу для решения систем с исходной $\mathcal{H}^2$ матрицей. Автор предлагает несколько методов, приведенных ниже.

1. (Прямой метод решения) Применим прямой разреженный решатель к SE(A) и при заданном y вычислим x.

2. (Итерационный метод решения для системы (3.8) с предобуславливателем) Сконструируем предобуславливатель для итерационного решателя системы (3.8) основанный на блочной структуре матрицы SE(A).

3. (Итерационный метод решения для системы с исходной матрицей A используя матрицу SE(A) в качестве предобуславливателя) Неточное прямое решение системы с SE матрицей (3.8) может быть использовано в качестве предобуславливателя для итерационного метода решения исходной системы

$$Ax = y.$$

Для получения неточного решения системы (3.8) мы используем несколько шагов итерационного метода. Также на основе (3.8) можно построить предобуславливатель ILUt.

Теперь опишем решатели в деталях.

Метод 1

Очень естественная идея применить быстрый прямой решатель к матрице SE(A). Однако, для больших N, потребность в памяти очень быстро растёт. Пре-

имущество данного метода это простота реализации как только SE матрица вычислена сразу можно вызвать прямой решатель и получить ответ.

Метод 2

Теперь рассмотрим возможность итерационного решения системы (3.8). В численных экспериментах установлено, что матрица $\mathrm{SE}(A)$ обладает достаточно большим числом обусловленности. Поэтому итерационному решателю нужен предобуславливатель. Автором предложен **блочный SE предобуславливатель**. мы вычисляем обращение «дальнего блока» матрицы $\mathrm{SE}(A)$ а все другие заменяем на единичные матрицы

$$B = \begin{bmatrix} I & 0 & 0 \\ 0 & P(S) & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{3.9}$$

где $P(S)$ это некоторый предобуславливатель блока S. Обратите внимание, что блок S может быть прямоугольным в таком случае мы рассматриваем предобуславливатель наименьшей квадратной подматрицы которая содержит S. В численных экспериментах видно, что плохая обусловленность блока S является причиной плохой обусловленности матрицы H.

Метод 3

Основным недостатком Метода 2 является медленное матрично-векторное произведение в сравнение с матрично-векторным произведением $\mathcal{H}^2$ матрицы. С другой стороны важное преимущества Метода 2 это возможность построения SE предобуславливателя. Данный метод использует итерационный метод решения для системы с $\mathcal{H}^2$ матрицей и предобуславливает его с помощью SE матрицы.

Матрица A может рассматриваться в качестве дополнения по Шуру матрицы $H = \mathrm{SE}(A)$ с исключенными строками соответствующими $\widehat{x}$ и $\widehat{y}$. Обычно, дополнение по Шуру используется как предобуславливатель; здесь мы используем предобуславливатель на основе *обратного дополнения по Шуру* (похожие идеи представлены в [66]). Дополнение по Шуру в данном методе используется в противоположном ключе: мы решаем систему с матрицей A итерационно и для решения поправочного уравнения мы переходим к расширенной системе, строим на её

основе предобуславливатель ILUt, решаем поправочную систему приближенно и достаем из большого решения соответствующую компоненту ответа. Более того, дополнительное ускорение может быть получено при использовании *дожимания* $\mathcal{H}^2$ матрицы, т.е.

$$A \approx B$$

где B имеет меньшие ранги. Описание эффективного алгоритма основано на идее представленной в [14], и реализовано в пакете h2tools. Затем используем $\mathrm{SE}(B)$ вместо $\mathrm{SE}(A)$ для построения SE матрицы, таким образом данный метод предобуславливания основан не на «обращением дополнения по Шуру», а на «приближенным обращением дополнения по Шуру». Мы обозначим данный предобуславливатель через *SVD-SE* (так как дожимается при помощи SVD разложения). Итоговый Алгоритм итерационного решения с SE-SVD предобуславливателем приведен в 6.

Алгоритм 6: Итерационный $\mathcal{H}^2$ решатель с SVD-SE предобуславливателем

Дано:

A - $\mathcal{H}^2$ матрица,

y - правая часть,

ε - точность основного итерационного метода,

Итерационный решатель:

$B = SVD\_compress(A)$

$H = \mathrm{SE}(B)$

$x = \mathrm{GMRES}(A, y, \mathrm{tol} = \varepsilon, \mathrm{prec} = \mathrm{RevSchur})$

Вернуть: x

***Предобуславливатель** RevSchur(y):*

$\hat{y} = \begin{pmatrix} y^\top & 0 & 0 \end{pmatrix}^\top$ - расширяем правую часть

$L, U = \mathrm{ILUt}(H)$

$\hat{x} = U^{-1}L^{-1}\hat{y}$

$x = \hat{x}[0 : \mathrm{size}(y)]$

Вернуть: x

Данный подход (Метод 3) оказался наиболее эффективным для рассмотренных нами задач.

***Замечание* 3.1.2.** *Програмная реализация, тестирование и сравнение приведённых выше алгоритмов с другими методами решения линейных систем с $\mathcal{H}^2$ матрицами приведены в главе 4, в параграфе 4.2.1.*

3.2. Не-расширенная разреженная факторизация $\mathcal{H}^2$ матрицы

3.2.1. Основная идея

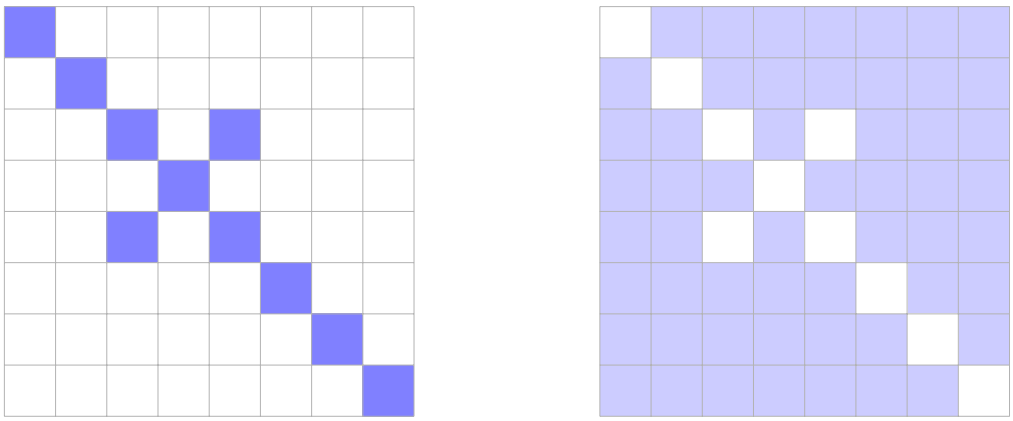
Рассмотрим плотную матрицу $A \in \mathbb{R}^{N\times N}$ с $\mathcal{H}^2$ структурой. Формальное определение $\mathcal{H}^2$ матриц представлено в главе 1, в параграфе 1.2.2, ниже приводятся некоторые факты об $\mathcal{H}^2$ матрицах, которые будут использоваться в данном разделе. Матрица A - это блочная матрица со следующими свойствами. Она состоит из «ближних» и «дальних» матриц:

$$A = C_l + F_l, \quad l \in 0, \ldots, L$$

для простоты рассуждений предположим, что размер блоков на нулевом уровне ($l = 0$) одинаковые и равны B, также предположим что блоки на более высоких уровнях обьеденяются по 2 (т.е. дерево источников и приёмников предполагаются бинарными). Таким образом, размер блока на l-м уровне B_l равен

$$B_l = 2^l B,$$

$C_l \in \mathbb{R}^{N\times N}$ - это блочно-разреженная матрица, состоящая из «ближних» блоков полного ранга, и $F_l \in \mathbb{R}^{N\times N}$ это блочная матрица, состоящая из «дальних» блоков, смотри Рисунок 3.2b. Блочные строки и столбцы матриц F_l обладают малым рангом. Также выполняется свойство вложенности базисов: базисные строки (столбцы) на уровне l являются подмножеством базисных строк на уровне $(l-1)$. Это используется при вычислениях следующих уровней, описанных в параграфе «Компрессия всех уровней».



(a) Матрица C_0 (b) Матрица F_0

Рисунок 3.2: Ближние и дальние блоки матрицы A на уровне $l = 0$

Компрессия дальних блоков на нулевом уровне

Сначала рассмотрим процедуру компрессии на нулевом блочном уровне ($l = 0$). Пусть число блоков на нулевом уровне равно M. Ненулевой блок $F_{ij} \in \mathbb{R}^{B\times B}$ матрицы F_0 имеет малый ранг:

$$F_{ij} \approx \widetilde{U}_i \widetilde{F}_{ij} \widetilde{V}_j^\top, \quad \forall i, j \in \overline{1 \dots M}$$

где $\widetilde{F}_{ij} \in \mathbb{R}^{B\times B}$ это сжатый дальний блок со следующей структурой:

$$\widetilde{F}_{ij} = \begin{bmatrix} \dot{F}_{ij} & 0 \\ 0 & 0 \end{bmatrix},$$

где $\dot{F}_{ij} \in \mathbb{R}^{r\times r}$. Матрицы $\widetilde{U}_i \in \mathbb{R}^{B\times B}$ и $\widetilde{V}_j \in \mathbb{R}^{B\times B}$ унитарные. Все блоки в i-й строке имеют одинаковую матрицу компрессии $\widetilde{U}_i$ и все блоки в j-м столбце также имеют одинаковую матрицу компрессии $\widetilde{V}_j^\top$.

Целью процедуры компрессии является спарсификация матрицы A путем получения сжатых блоков $\widetilde{F}_{ij}$ вместо плотных F_{ij}. Для этого необходимо найти компрессионные матрицы $\widetilde{U}_i$ и $\widetilde{V}_j$ и применить матрицу $\widetilde{U}_i^\top$ к i-й строке и $\widetilde{V}_j$ к j-му столбцу. Рассмотрим блочно-диагональную унитарную матрицу компрессии

для всей F_0:

$$U_0^\top = \begin{bmatrix} \widetilde{U}_1^\top & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \widetilde{U}_M^\top \end{bmatrix}. \tag{3.10}$$

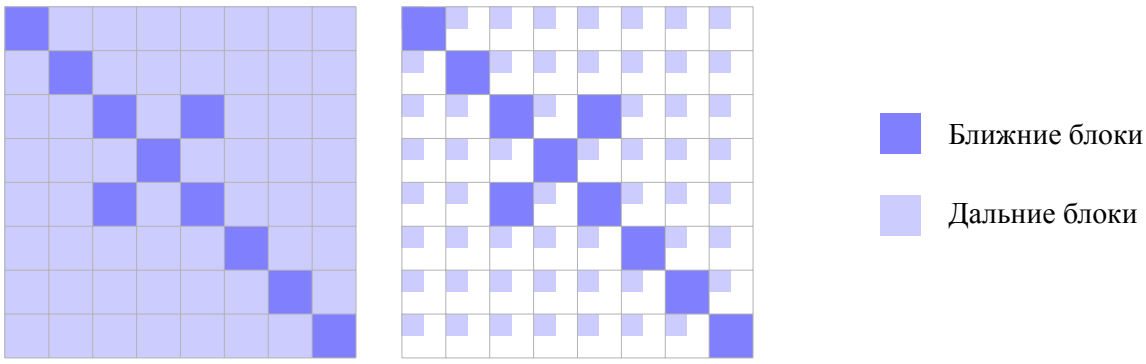
Аналогично, для блочных столбцов получаем блочно-диагональную унитарную компрессионную матрицу

$$V_0 = \begin{bmatrix} \widetilde{V}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \widetilde{V}_M \end{bmatrix}. \tag{3.11}$$

Применяя матрицы U_0 и V_0 к матрице A получаем матрицу A_1 с *дожатой* дальней матрицей:

$$A_1 = U_0^\top A V_0.$$

Данный процесс проиллюстрирован на Рисунке 3.3.



(a) Матрица A (b) Матрица A_1

Рисунок 3.3: Компрессия нулевого уровня

Получаем:

$$A_1 = U_0^\top (C_0 + F_0) V_0 = U_0^\top C_0 V_0 + \widetilde{F}_1,$$

где $\widetilde{F}_1$ это дожатая дальняя матрица которая состоит из блоков $\widetilde{F}_{ij}$. Отметим, что матрица C_0 хранится в $\mathcal{H}^2$ формате, это так называемая ”ближняя матрица”.

Компрессия первого уровня ($l = 1$)

Для каждой блочной строки матрицы A_1 обозначим строки с нулевой дальней зоной через **небазисные**, а остальные строки через **базисные первого уров-**

ня. Пусть каждая блочная строка (столбец) имеет r базисных строк (столбцов) и $(B-r)$ небазисных. Рассмотрим перестановку P_{r1} которая ставит не-базисные строки перед базисными сохраняя порядок и перестановку P_{c1} которая делает то же для столбцов, смотри Рисунок 3.5a. Для переставленной матрицы

$$\widetilde{A}_1 = P_{r1} A_1 P_{c1}$$

получим

$$\widetilde{A}_1 = \begin{bmatrix} A_{\mathbf{n_1 n_1}} & A_{\mathbf{n_1 b_1}} \\ A_{\mathbf{b_1 n_1}} & A_{\mathbf{b_1 b_1}} \end{bmatrix},$$

где $A_{\mathbf{n_1 n_1}} \in \mathbb{R}^{M(B-r)\times M(B-r)}$ это подматрица на пересечении небазисных строк и небазисных столбцов, $A_{\mathbf{b_1 n_1}} \in \mathbb{R}^{Mr\times M(B-r)}$ это блок на пересечении базисных строк и небазисных столбцов и так далее, смотри Рисунок 3.5a. Обозначим переставленную дальнюю матрицу через:

$$\widehat{F}_1 = (P_{r1} \widetilde{F}_1 P_{c1}).$$

Отметим, что перестановки P_{r1} и P_{c1} концентрируют все ненулевые блоки матрицы $\widetilde{F}_1$ внутри подматрицы $A_{\mathbf{b_1 b_1}}$. Обозначим переставленную ближнюю матрицу через

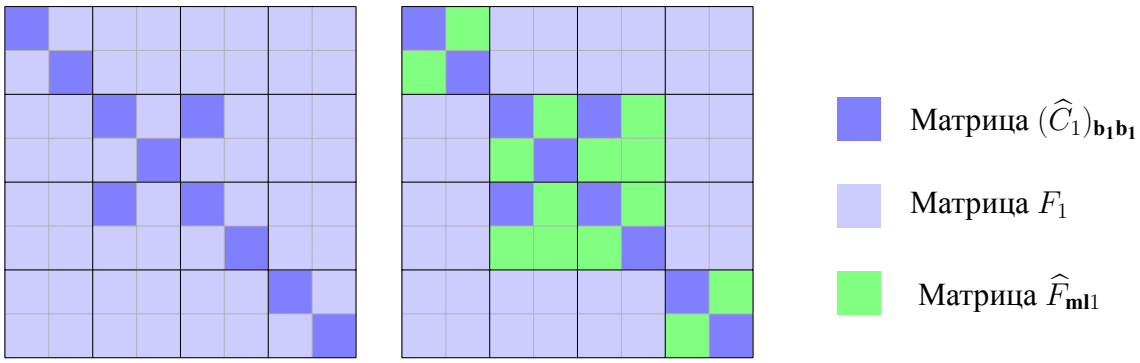
$$\widehat{C}_1 = (P_{r1} U_0^\top C_0 V_0 P_{c1}). \tag{3.12}$$

Рассмотрим подматрицу $A_{\mathbf{b_1 b_1}} \in \mathbb{R}^{Mr\times Mr}$, отметим, что эта матрица имеет в точности такую же блочную структуру как матрица A, но размер блока в $A_{\mathbf{b_1 b_1}}$ это r. Теперь объединим блочные строки и столбцы матрицы $A_{\mathbf{b_1 b_1}}$ в группы по J блоков (на Рисунке 3.4a $J = 2$). Предположим, что $Jr = B$. Будем называть новые сгруппированные блоки «большими блоками», если большой блок включает в себя только дальние блоки назовём его дальним, если большой блок включает в себя хотя бы один ближний блок - назовём его ближним. Дальние блоки дальней матрицы $\widehat{F}_1$ которые вошли в состав больших ближних блоков назовём $\widehat{F}_{\mathbf{ml}1}$, смотри Рисунок 3.4. Обозначим матрицу с большим ближними блоками через

$$C_1 = \widehat{C}_1 + \widehat{F}_{\mathbf{ml}1}. \tag{3.13}$$

Обозначим матрицу с большими дальними блоками через F_1. Рассмотрим данное обьединение блоков для матрицы $A_{\mathbf{b_1 b_1}}$:

$$A_{\mathbf{b_1 b_1}} = (\widehat{C}_1)_{\mathbf{b_1 b_1}} + \widehat{F}_1 = (\widehat{C}_1)_{\mathbf{b_1 b_1}} + \widehat{F}_{\mathbf{ml}1} + F_1 = (C_1)_{\mathbf{b_1 b_1}} + F_1.$$



(a) $A_{\mathbf{b_1 b_1}} = (\widehat{C}_1)_{\mathbf{b_1 b_1}} + \widehat{F}_1$ (b) $A_{\mathbf{b_1 b_1}} = (C_1)_{\mathbf{b_1 b_1}} + F_1$

Рисунок 3.4: Маленькие (размера r) и большие (размера B) дальние и ближние блоки матрицы $A_{\mathbf{b_1 b_1}}$

Это же уравнение для матрицы $\widetilde{A}_1$:

$$\widetilde{A}_1 = \widehat{C}_1 + \widehat{F}_1 = \widehat{C}_1 + \widehat{F}_{\mathbf{ml}1} + F_1 = C_1 + F_1.$$

Отметим, что большие блочные строки и столбцы матрицы F_1 имеют малый ранг, так как матрица A имеет $\mathcal{H}^2$ структуру. Аналогично (3.10) вычислим блочные матрицы компрессии $U_{\mathbf{b_1}}, V_{\mathbf{b_1}} \in \mathbb{R}^{Mr \times Mr}$ которые дожимают матрицу F_1.

Произведение матрицы F_1 на матрицы $U_{\mathbf{b_1}}$ и $V_{\mathbf{b_1}}$ приводит к сжатию:

$$\widetilde{F}_2 = U_{\mathbf{b_1}}^\top F_1 V_{\mathbf{b_1}}, \tag{3.14}$$

матрица $\widetilde{F}_2$ состоит из сжатых блоков.

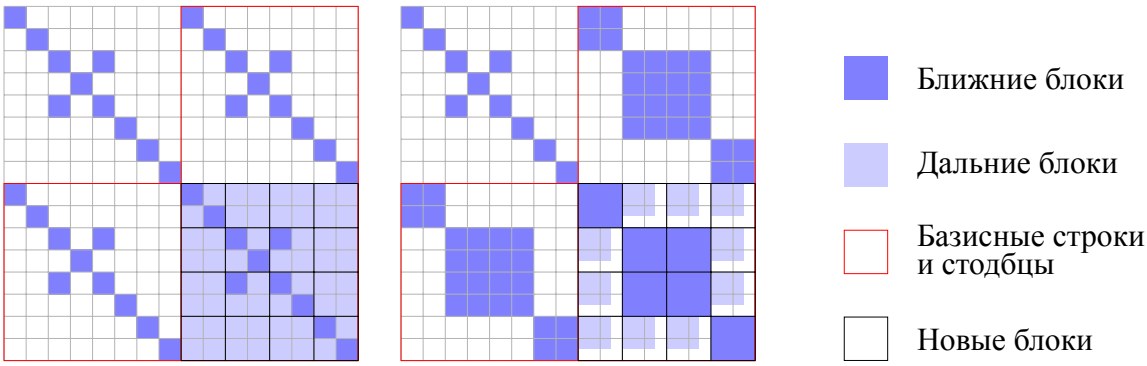
Рассмотрим расширенные матрицы $U_{\mathbf{b_1}}$ и $V_{\mathbf{b_1}}$ которые применим к матрице $\widehat{A}_1$:

$$U_1 = \begin{bmatrix} I_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & U_{\mathbf{b_1}} \end{bmatrix}, \quad V_1 = \begin{bmatrix} I_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & V_{\mathbf{b_1}} \end{bmatrix}. \tag{3.15}$$

Применяя матрицы U_1 и V_1 к матрице $\widetilde{A}_1$ получим матрицу с дожитым первым уровнем:

$$A_2 = U_1^\top \widetilde{A}_1 V_1.$$

Процесс компрессии первого уровня изображен на Рисунке 3.5b.



(a) Матрица $\widetilde{A}_1$ (b) Матрица $\widehat{A}_1$

Рисунок 3.5: Компрессия первого уровня

Для первого уровня получаем

$$A_2 = U_1^\top (C_1 + F_1) V_1 = U_1^\top C_1 V_1 + \widehat{F}_2.$$

Компрессия всех уровней

Затем применяем базисно-небазисную перестановку и компрессию L раз. Получаем:

$$\begin{aligned} A_1 &= U_0^\top A V_0 = U_0^\top C_0 V_0 + \widehat{F}_1 \\ A_2 &= U_1^\top U_0^\top A V_0 V_1 = U_1^\top C_1 V_1 + \widehat{F}_2 \\ &\vdots \\ A_L &= \left(\prod_{k=0}^{L} U_k^\top \right) A \left(\prod_{k=L}^{0} V_k \right) = U_{L-1}^\top C_{L-1} V_{L-1} + \widehat{F}_L = S, \end{aligned} \tag{3.16}$$

таким образом

$$A = \left(\prod_{k=L}^{0} U_k \right) S \left(\prod_{k=0}^{L} V_k^\top \right).$$

Если обозначим

$$U = \prod_{k=0}^{L} U_k, \quad V = \prod_{k=0}^{L} V_k, \tag{3.17}$$

тогда финальным результатом алгоритма будет приближенная разреженная факторизация

$$A = U S V^\top, \tag{3.18}$$

где S это разреженная матрица, которая имеет такой же размер как матрица A, U и V это унитарные матрицы являющиеся произведением матриц перестановки и блочно-диагональных унитарных матриц.

***Замечание* 3.2.1.** *Если A аппроксимирована в $\mathcal{H}^2$ формате, тогда матрицы S, U и V могут быть построены из параметров $\mathcal{H}^2$ представления, смотри детали в параграфе 3.2.3.*

***Замечание* 3.2.2.** *Матрицы U_i и V_i, $i \in \{1 \dots k\}$ при умножении увеличивают заполненность блоков $A_{\mathbf{n_i b_i}}$ и $A_{\mathbf{b_i n_i}}$. Поэтому, разреженность матрицы S это область отдельного исследования. В параграфе 3.2.2 исследуется разреженность фактора S.*

***Замечание* 3.2.3.** *Отметим, что предложенный алгоритм спарсификации также применим к таким частным случаям $\mathcal{H}^2$ матриц как HSS и HOLDR матрицы.*

Псевдокод алгоритма компрессии

В данном разделе приводится псевдокод алгоритма спарсификации

Алгоритм 7: Не-расширенная разреженная факторизация

Дано:

$A \in \mathbb{R}^{N \times N}$ - матрица с $\mathcal{H}^2$ структурой

L - число уровней

Компрессия:

C_0 известно из A

for *k = 0 ...L* **do**

M_k - число блоков на уровне M_k

$P_{rk}, \quad P_{ck}$ перестановка базисных и небазисных строк и столбцов

$\widetilde{A}_k = P_{rk} A_1 P_{ck}$

for *i = 1 ...*M_k **do**

U_{ki} SVD i-ой блочной строки матрицы $A_{\mathbf{b_k b_k}}$

V_{ki} SVD i-го блочного столбца матрицы $A_{\mathbf{b_k b_k}}$

$U_k = P_{rk}\, \mathbf{diag}(U_{k1}, \ldots, U_{kM_k})$

$V_k = P_{ck}\, \mathbf{diag}(V_{k1}, \ldots, {}_{kM_k})$

$A_{j+1} = U_j^\top A_j V_j P_j^\top$

Выход: *Факторизация* $A \approx U^\top S V$

$U = \left(\prod_{k=0}^{L} U_k\right),$

$V = \left(\prod_{k=0}^{L} V_k\right),$

$S = \left(\prod_{k=0}^{L} U_k^\top\right) A \left(\prod_{k=L}^{0} V_k\right)$

3.2.2. Разреженность матрицы S

Определим блочный шаблон разреженности разреженной матрицы. Для матрицы A с M_1 блочными строками, M_2 блочными столбцами и размером блока B определим $\mathbf{bsp}(A)_{B\times B}^{M_1\times M_2\times M_1\times M_2}$ (блочный шаблон разреженности) как функцию

$$\mathbf{bsp} : \mathbb{R}^{N\times N} \to \mathbb{B}^{M\times M}$$

которая возвращает 1 если блок A_{ij} ненулевой 0 в противном случае. Через $\#\mathbf{bsp}(A)_{B\times B}^{M\times M}$ обозначим ненулевых блоков в матрице A.

***Утверждение* 3.2.4.** *Если матрица A имеет $\mathcal{H}^2$ структуру, если алгоритм компрессии имеет L уровней, если размер блока на каждом уровне $l \in \overline{0,\dots,L}$ равен B, если A имеет ближнюю матрицу первого уровня C и если дальние блочные столбцы и строки всегда сжимаются в два раза $r = B/2$, тогда алгоритм компрессии для матрицы A приводит к факторизации:*

$$A = U^\top S V,$$

где U и V это унитарные матрицы равные произведению блочно-диагональных унитарных матриц компрессии и матриц перестановки

$$U = \left(\prod_{j=0}^{K} U_j P_j^\top\right), \quad V = \left(\prod_{j=0}^{K} V_j P_j^\top\right),$$

а матрица S это разреженная матрица которая имеет

$$\#S = \left(4L + 6(\frac{1}{2^L} - 1)\right) \#\boldsymbol{bsp}(C)_{B\times B}^{M\times M}$$

ненулевых блоков размера $(r \times r)$.

Доказательство. Рассмотрим матрицу S из (3.18). Благодаря базисно-небазисным перестановкам строк и столбцов P_{ri} и P_{ci} матрица S разбита на блоки S_{ij}, где $i, j \in 0,\dots,L$.

Число ненулевых блоков в матрице S равно сумме ненулевых блоков в матрицах S_{ij}:

$$\#S = \sum_{i=0}^{L}\sum_{j=0}^{L} S_{ij}.$$

Вычислим число ненулевых блоков в матрице S_{ij}:

$$\boldsymbol{bsp}(S_{ij})_{r\times r}^{M/2^j\times M/2^j} = \boldsymbol{bsp}(C)_{2^iB\times 2^jB}^{M/2^j\times M/2^j},$$

где $i, j \in \overline{0,\dots,L}$. Обозначим $\#\boldsymbol{bsp}(C)_{B\times B}^{M\times M}$ через ξ.

$$\boldsymbol{bsp}(C)_{2^jB}^{M/2^i\times M/2^j\times M/2^i\times M/2^j} = \frac{\xi}{2^{\min(i,j)}}$$

Тогда

$$\#S = \sum_{i=0}^{L}\sum_{j=0}^{L} \#\boldsymbol{bsp}(C)_{2^iB\times 2^jB}^{M/2^j\times M/2^j} = \sum_{i=0}^{L}\left(\frac{2(L-i)-1}{2^i}\right)\xi =$$

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

Или в изначальных обозначениях:

$$\#S = \left(4L + 6(\frac{1}{2^L} - 1)\right) \#\boldsymbol{bsp}(C)_{B\times B}^{M\times M}.$$

Таким образом, количество ненулевых блоков в матрице S в константу раз больше, чем с ближней матрице C. Так как C - разреженная матрица, то S тоже разреженная. □

3.2.3. Построение факторов разложения из параметров $\mathcal{H}^2$ матрицы

В данном разделе используется определение $\mathcal{H}^2$ матрицы, которое подробно приведено в параграфе 1.2.2. Все обозначения перенесены из параграфа 1.2.2.

Построение факторов U и V из параметров $\mathcal{H}^2$ матрицы

Сначала построим унитарные матрицы U и V из факторизации (3.18). В соответствии с (3.17)

$$U = \prod_{k=0}^{L} P_{rk} U_k,$$

и

$$V = \prod_{k=0}^{L} P_{ck} V_k$$

Отметим, что матрицы U_k и V_k, $k \in \overline{0 \dots L}$, очень близки по смыслу матрицам R_l и $E_l, l \in \overline{0 \dots L}$ из $\mathcal{H}^2$ факторизации: и те и другие это матрицы дожимания одного уровня. Основное различие между этими матрицами заключается в том, что матрицы U_k и V_k составлены из квадратных унитарных блоков, а R_l и E_l составлены из прямоугольных и не-унитарных блоков.

Таким образом, если ортогонализовать матрицы из которых составлены R_l и E_l, дополнить их до квадратных, то мы получим матрицы U_k и V_k. Алгоритм, который ортогонализовывает блоки матриц R_l и E_l известен как алгоритм компрессии $\mathcal{H}^2$ матриц и приведен, например, в работе [14]. Перестановки P_{rk} и P_{ck} могут быть построены из кластерных деревьев. Таким образом, матрицы U и V могут быть построены из параметров $\mathcal{H}^2$ матрицы.

Построение матрицы S из параметров $\mathcal{H}^2$ матрицы

По Уравнению (3.16), Уравнению (3.12) и Уравнению (3.13):

$$S = U_{L-1}^{\top} C_{L-1} V_{L-1} + \widehat{F}_L =$$

$$= U_{L-1}^{\top}(\dots(U_1^{\top}(U_0^{\top} C_0 V_0 + \widehat{F}_{\text{ml1}})V_1 + \widehat{F}_{\text{ml2}})\dots)V_{L-1} + \widehat{F}_L.$$

Построение матриц U_i и V_i приведено в предыдущем разделе, матрица C_0 хранится в $\mathcal{H}^2$ матрице как ближняя матрица C, матрицы $\widehat{F}_{\text{ml}i}$ это в точности матрицы S_i из листа взаимодействия, матрица $\widehat{F}_L$ это матрица S_L. Таким образом, матрица S может быть построена из параметров $\mathcal{H}^2$ матрицы.

***Замечание* 3.2.5.** *Программная реализация, тестирование и сравнение данного алгоритма с другими методами решения линейных систем с $\mathcal{H}^2$ матрицами приведены в главе 4, в параграфе 4.2.2.*

3.3. Выводы по главе

В данной главе автором предложено два метода приведения $\mathcal{H}^2$ матрицы к разреженному виду - экстенсивный и неэкстенсивный методы. Для более раннего экстенсивного метода приведена оценка размера получаемой разреженной матрицы и предложены несколько способов предобуславливания систем с $\mathcal{H}^2$ матрицами. Для не-экстенсивного метода приведена оценка количества ненулевых элементов в получаемой разреженной матрицы, также приводится алгоритм построения факторов разреженного разложения из параметров $\mathcal{H}^2$ матрицы. Для обоих методов приведена программная реализация. Для программной реализации проведено обширное тестирование и сравнение и близкими методами.

Глава 4

Программный комплекс для факторизации и решения систем с блочно-малоранговыми матрицами

В данной главе приводится описание программной реализации алгоритмов, описанных в главах 2 и 3. Приводится сравнение скорости работы и затрат по памяти представленных программ с другими методами решения применяемыми для соответствующих задач.

4.1. Метод компрессии и исключения

Приближенная факторизация разреженных матриц СЕ была реализована на языке программирования Fortran с использованием BLAS из Intel MKL [45]. Разработанный программный код имеет удобный интерфейс на языке программирования Python. Пример использования данного интерфейса приведен в параграфе 4.1.1. Код оформлен в виде библиотеки ce_solver [77] и находится в открытом доступе. Производилось сравнение следующих методов решения линейных си-

стем: CE(ε) факторизация и решение факторизованной системы, в качестве прямого метода решения, CE(ε) и CE(r) факторизации в качестве предобуславливателя для итерационных солверов MINRES [71] и BiCGStab [2], прямые солверы CHOLMOD [4, 24] и UMFPACK [22], а также итерационные солверы MINRES и BiCGStab с предобуславливливателями ILUt [69], ILU2 [46] и ILU0 [70].

4.1.1. Интерфейс программного кода

В Листинге 1 приведен пример простой программы, которая реализует CE($r = 2$) факторизацию трехмерной матрицы Лапласа.

```
# Import main package:
from ce.ce_wrap import h2_dir

# Import other tools:
from helpers.fd_tools import gen_3d_lap, make_coord
from helpers.prms import kdt_prm

# Set the matrix size:
p = 10
N = 2**p

# Generate random right-hand side:
rhs = np.random.randn(N)*1.

# Generate 3D Laplacian matrix:
A = gen_3d_lap(p)

# Set the block size for the permutation:
B = 16

# Build a permutation using KDTree from sklearn:
coord = make_coord(p)
prm, bl_ar = kdt_prm(coord, B=B)
A_prm = A[:, prm][prm, :]

# Apply CE factorization:
sol, info = h2_dir(A_prm, rhs, bl_ar=bl_ar, r=2, join=B)
time, mem, res, error = info
```

Листинг 1: Решение разреженной системы с помощью пакета ce_solver

4.1.2. Конечно-разностная дискретизация уравнения диффузии

Для тестов рассматривалась система полученная при равномерной кубической дискретизации трехмерного уравнения диффузии.

$$\begin{aligned} -\text{div}\,(\,k(x)\,\text{grad}\,u(x)) &= f(x), \quad x \in \Omega \\ u|_{\delta\Omega} &= 0 \end{aligned}, \tag{4.1}$$

где $\Omega = [0,1]^3$, тензор диффузии $k(x) = \text{diag}(x_1^2 + 0.5, x_2^2 + 0.5, x_3^2 + 0.5)$, правая часть $f(x) = 1$.

Для MINRES мы использовали Python библиотеку SciPy. Сравнение проводилось на сервере с 32 Intel® Xeon® E5-2640 v2 (20M Cache, 2.00 GHz) процессорами и с 256GB RAM. Тесты проводились в одно-процессорном режиме.

Факторизация CE(ε) в качестве прямого солвера

Факторизация, полученная в случае процедуры CE(r) недостаточно точна, чтобы использовать её в качестве прямого солвера. Рассмотрим факторизацию CE(ε) в качестве приближенного прямого солвера. В Таблице 4.14 мы приводим относительную точность η которая может быть получена при приближенном прямом решении системы, матрица которой факторизована с помощью CE.

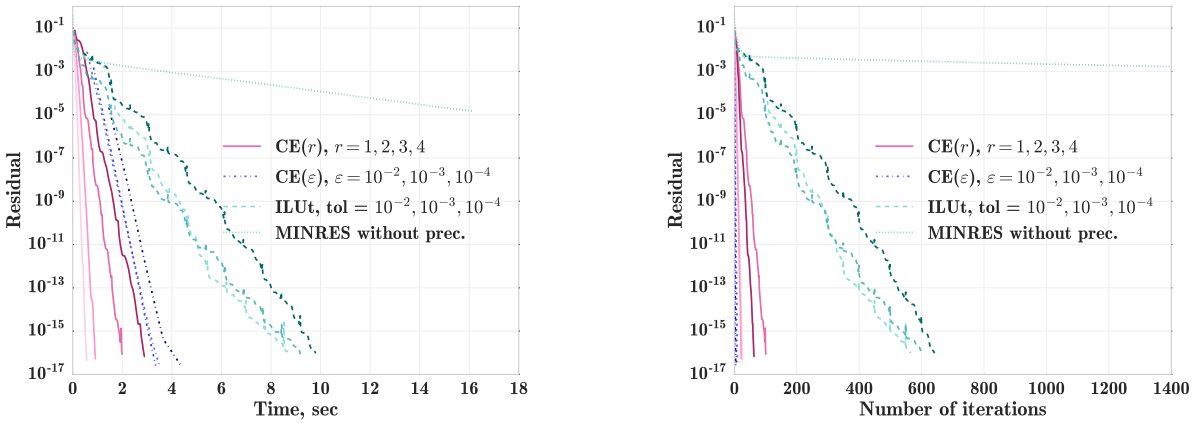
Параметр сжатия, ε	10^{-2}	10^{-4}	10^{-6}	10^{-8}
Относительная точность, η	4.0×10^{-1}	9.1×10^{-3}	1.2×10^{-5}	9.9×10^{-7}
Требуемая память, MB	553	1348	2494	2671
Время, sec	3.40107	8.28388	17.40455	29.65910

Таблица 4.1: Точность факторизации и требуемая точность, $N = 65536$. ТОчность η здесь вычисляется следующим образом: $\eta = \frac{|x_{CE} - x_*|}{|x_*|}$, где x_{CE} это полученное решение и x_* это точное решение.

Основная проблема с прямым солверов CE(ε) заключается в том, что он требует слишком много памяти. С другой стороны, CE(ε) с достаточно большим ε работает быстро и требует немного памяти. Эта факторизация, также как и CE(r) может быть использована как очень хороший предобуславливатель для итерационного солвера, например, MINRES.

Сходимость CE(ε), CE(r)

На Рисунке 4.1 представлена сходимость метода MINRES с CE(ε), CE(r), ILUt предобуславливателями, а также сходимость метода MINRES без предобуславливателя.



(a) Невязка/время

(b) Невязка/итерации

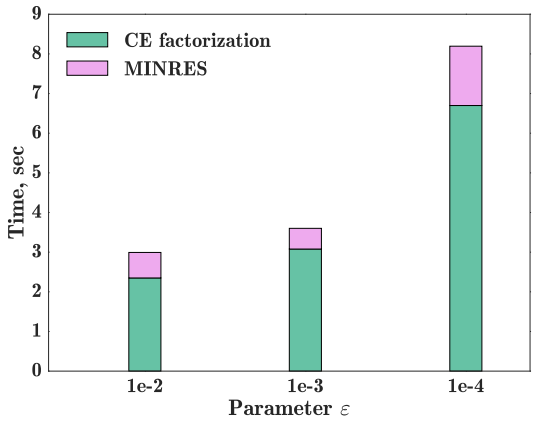
Рисунок 4.1: Сравнение сходимости солвера MINRES с разными предобуславливателями, $N = 32768$. Стиль линий определяет тип предобуславливателя, разные оттенки одного цвета обозначают разные параметры одного вида предобуславливателя.

***Замечание* 4.1.1.** *Отметим, что число итераций в MINRES без предобуславливателя, также как и число итераций с предобуславливателем ILUt растёт вместе с ростом сетки, так как обусловленность матрицы ухудшается. CE(ε) и CE(r) это предобуславливатели лучшего в том смысле, что для предобуславливателя CE(ε) с фиксированным параметром ε число итераций не растёт; для предобуславливателя CE(r) число итераций растёт медленно (Таблица 4.3).*

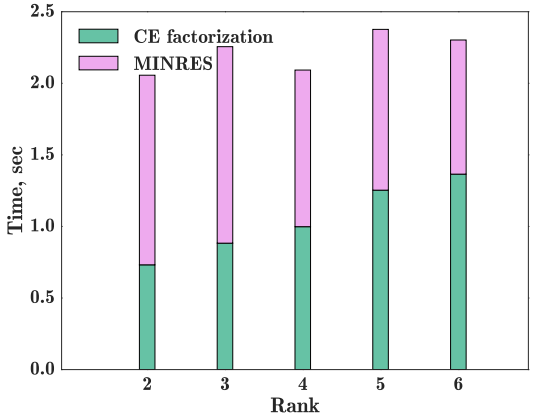
Matrix size, N	8192	16384	32768	65536	131072
Without preconditioner	147	2636	> 1000	> 1000	> 1000
ILUt, $t = 10^{-3}$	40	92	339	> 1000	> 1000
CE(r), $r = 4$	23	25	29	30	36
CE(ε), $\varepsilon = 10^{-3}$	4	5	6	5	6

Таблица 4.2: Число итераций требуемых MINRES чтобы сойтись до точности $\epsilon = 10^{-10}$ для разных предобуславливателей.

Наблюдается характерный компромисс: чем меньше ε, тем лучше сходимость, но тем больше требуется памяти для хранения. Данный факт иллюстрирован на Рисунке 5.13.



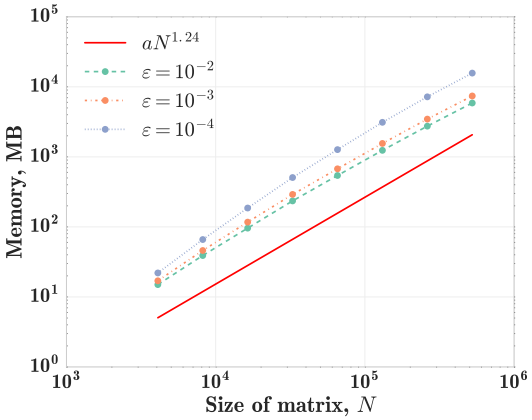
(a) Адаптивный предобуславливатель CE(ε)



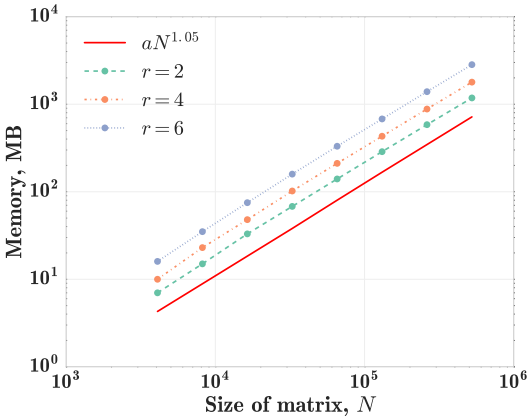
(b) Предобуславливатель с фиксированным рангом CE(r)

Рисунок 4.2: Время факторизации и итераций MINRES, $N = 32768$

Отметим, что на Рисунке 4.2b общее время для нескольких рангов одинаковое, однако, для $r = 2$ требуется меньше памяти. Также отметим что CE(ε) и CE(r) имеют сравнимое время решения, поэтому сравним требуемую память для разных N.



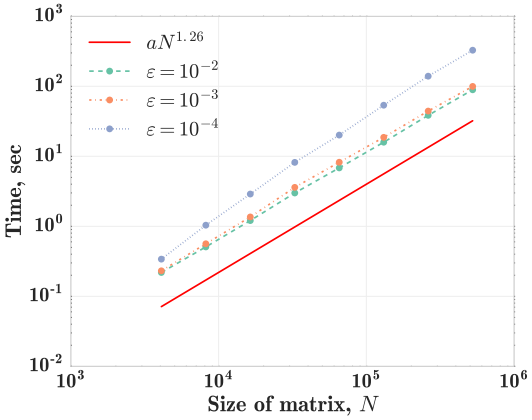
(a) Адаптивный предобуславливатель CE(ε)



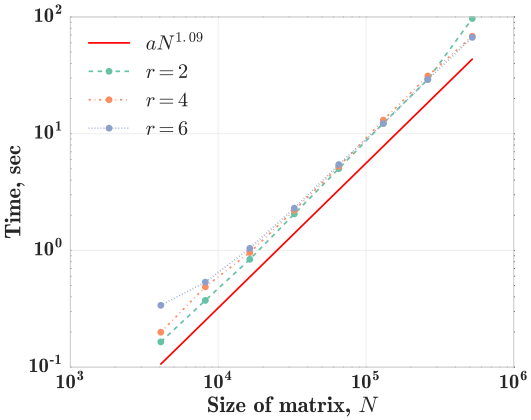
(b) Предобуславливатель с фиксированным рангом CE(r)

Рисунок 4.3: Память, требуемая для приближенной факторизации

Память, требуемая для факторизации с фиксированным рангом CE(r), предсказуемо оказалась ниже, чем для факторизации с адаптивным рангом CE(ε). На Рисунке 4.4 приведено общее время, требуемое для построения предобуславливателей CE(ε) и CE(r) и предобусловленных итераций MINRES для разных N.



(a) Адаптивный предобуславливатель CE(ε)



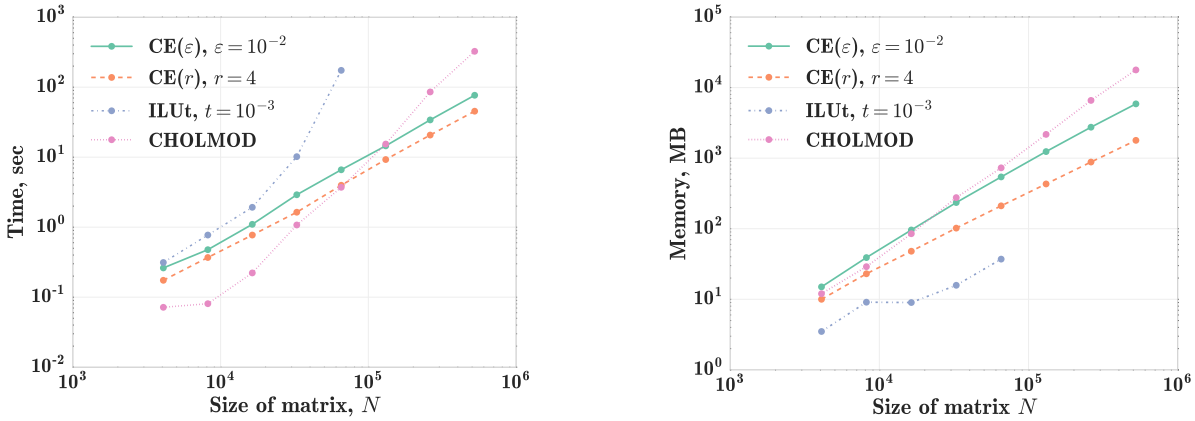
(b) Предобуславливатель с фиксированным рангом CE(r)

Рисунок 4.4: Общее время решения для MINRES с предобуславливателями CE(ε) и CE(r)

Мы установили экспериментально, что для предобуславливателя CE(r) $r = 4$ это обычно хороший выбор, отметим, что в данном случае $r = \frac{B}{2}$. Для предобуславливателя CE(ε) хорошим выбором является $\varepsilon = 10^{-2}$ (для данного примера).

Итоговое сравнение

Теперь сравним солвер CE(ε), солвер CE(r), CHOLMOD и MINRES с ILUt предобуславливателем. Так как MINRES с ILUt не сошелся за 1000 итераций для $N > 32768$, мы приводим его результаты только для тех размеров сетки, на которых он сошелся.



(a) Сравнение по времени.

(b) Сравнение по памяти

Рисунок 4.5: Итоговое сравнение солверов.

Отметим, что солвер CE(r) начинает выигрывать по памяти у метода CHOLMOD на $N \approx 5000$ и начинает работать быстрее на $N \approx 10000$. Для максимального N которое допускает кластер мы решаем систему в ~ 10 быстрее чем CHOLMOD, и требуем примерно в ~ 10 меньше памяти. Отметим, что точность решения CHOLMOD составляет 10^{-11}.

4.1.3. Конечно-элементная дискретизация уравнения Пуассона и уравнения упругой деформации

Двумерная задача Пуассона

Решается уравнение Пуассона

$$\begin{aligned} -\bigtriangledown^2 u(x) &= f(x), \quad x \in \Omega, \\ u(x) &= u_D(x), \quad x \in \delta\Omega. \end{aligned} \tag{4.2}$$

где $\Omega = K((0,0),1) - K((0.5,0),0.5)$, $K(c,r)$ - это круг с центром c и радиусом r, смотри Рисунок 4.6. Уравнение Пуассона (4.2) дискретизуется при помощи метода конечных элементов [83]. Дискретизация и построение матрицы проводится при помощи пакета FEniCS [26, 53].

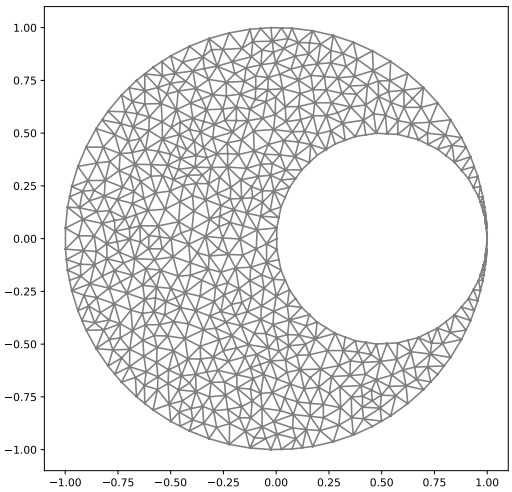


Рисунок 4.6: Область Ω

Полученная система решается методом BiCGStab [2] из пакета PyAMG [11] с использованием предобуславливателя CE($r = 2$), предобуславливателя ILUt с пороговым значением $\tau = 10^{-3}$, предобуславливателя ILU0 [70], предобуславливателя ILU2 [46] из пакета ANI2d [3], и прямого решателя CHOLMOD. Для метода CE предварительная перестановка строится при помощи пакета METIS [47, 48].

На Рисунке 4.7 приведено сравнение вклада времени построения предобуславливателя и времени итераций в общее время решения системы для предобу-

слабливателей ILU0, ILU2, ILUt и CE($r = 2$). Для метода BiCGStab требуется падение невязки до $\epsilon = 10^{-10}$.

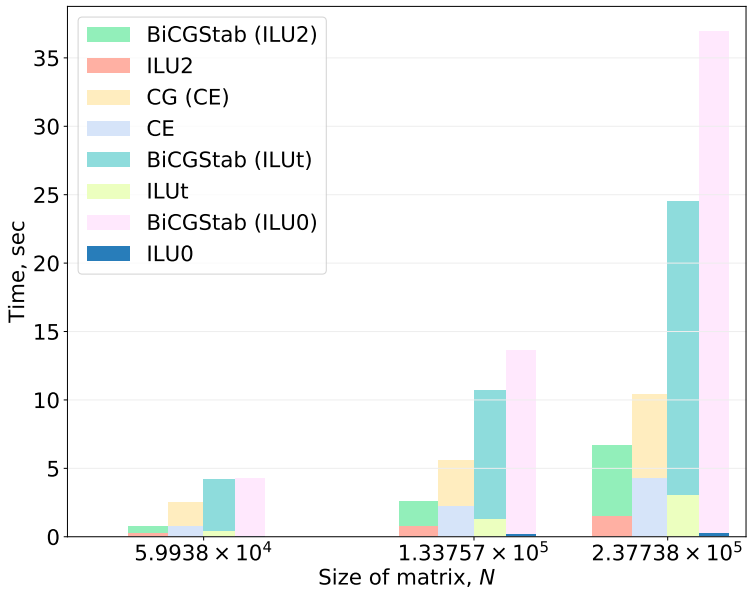
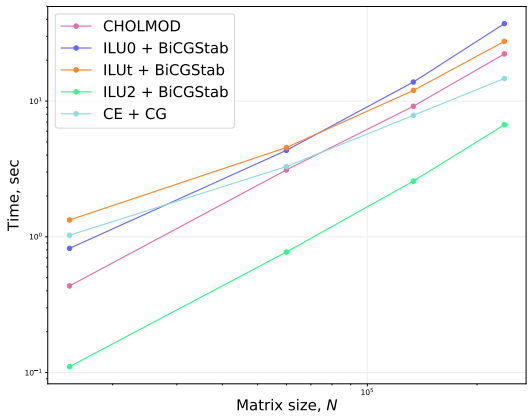
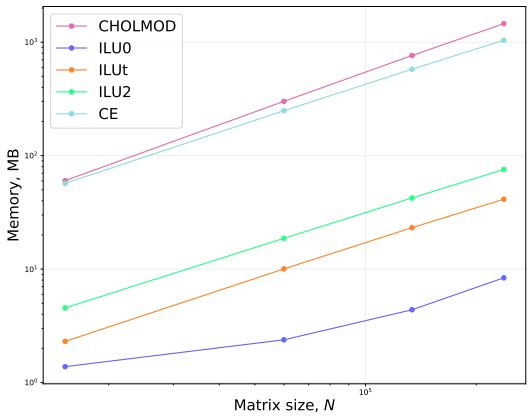


Рисунок 4.7: Сравнение предобуславливателей

Время построения CE($r = 2$) самое большое, однако, этот предобуславливатель значительно сокращает количество итераций метода BiCGStab. Сравним время и память требуемое на решение системы до точности $\varepsilon = 10^{-10}$ при помощи метода BiCGStab с различными предобуславливателями и метода CHOLMOD.



(a) Сравнение по времени.



(b) Сравнение по памяти

Рисунок 4.8: Итоговое сравнение солверов.

Приведем сравнение данных методов для различных требуемых точностей решения $\epsilon = 10^{-3}, 10^{-6}, 10^{-10}$.

Метод	CE(ε)	CE($r = 2$)+BiCGStab	ILUt+BiCGStab	ILU0+BiCGStab	ILU2+BiCGStab	CHOLMOD
$\epsilon = 10^{-3}$						
Число итераций	-	30	12	36	40	-
Время решения, сек	4.1	7.6(4.3+3.3)	4.63(3.0+1.63)	1.6(0.2+1.4)	3.9(1.6+2.3)	-
Память, MB	1042	1042	21.7	8.4	75.6	-
$\epsilon = 10^{-6}$						
Число итераций	-	53	114	278	66	-
Время решения, сек	-	9.5(4.3+5.2)	12.6(2.9+9.7)	19.1(0.2+18.9)	5.2(1.6+3.6)	-
Память, MB	-	1042	21.7	8.4	75.6	-
$\epsilon = 10^{-10}$						
Число итераций	-	76	195	500	95	-
Время решения, сек	-	11.8(4.2+7.6)	27.1(3.0+24.5)	37.1(0.2+36.9)	6.7(1.5+5.2)	22.2
Память, MB	-	1042	21.6	8.4	75.6	1457.8

Таблица 4.3: Сравнение данных методов для различных точностей решения, $N = 237738$

Трехмерная задача упругости

Рассматривается трехмерное уравнение упругой деформации для небольших смещений

$$\begin{gathered} -\bigtriangledown \sigma = f, \quad \in \Omega, \\ \sigma = \lambda \text{tr}(\varepsilon) I + 2\mu\varepsilon, \quad x \in \delta\Omega., \\ \varepsilon = \frac{1}{2}\left(\bigtriangledown u + (\bigtriangledown u)^{\top}\right) \end{gathered} \tag{4.3}$$

где σ - это тензор напряжений, f - это приложенная сила на единицу объема, λ и μ - это модули упругости Ламе для материала в области Ω,

$$\Omega = K((0,0,0),1) - K((0.5,0,0),0.48) - K((1,1,0),1) - K((1,-1,0),1),$$

$K(c,r)$ - это шар с центром c и радиусом r, смотри Рисунок 4.9. I это единичный тензор, tr - это оператор следа тензора, ε - это симметричный тензор скорости деформации, u - это векторное поле смещения. Здесь предполагаются изотропные условия упругости.

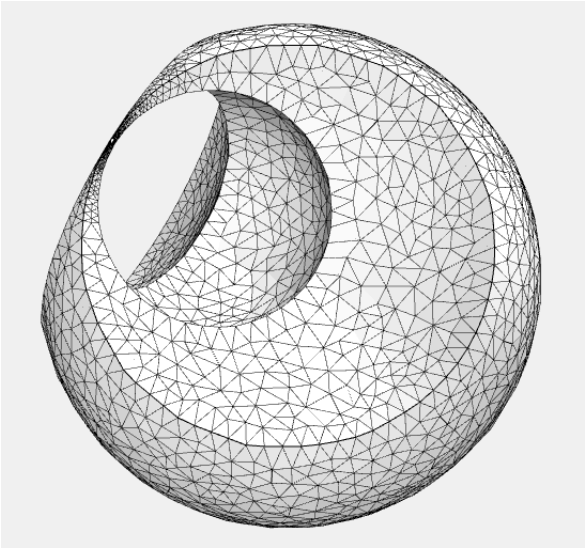


Рисунок 4.9: Область Ω

Дифференциальное уравнение (4.3) дискретизуется при помощи метода конечных элементов [83]. Полученная система решается методом BiCGStab с использованием предобуславливателя CE($r = 2$), предобуславливателя ILUt с пороговым значением $\tau = 10^{-3}$, предобуславливателя ILU0 [70], предобуславливателя ILU2 [46] из пакета ANI2d [3] и прямого решателя CHOLMOD. Для метода CE предварительная перестановка строится при помощи пакета METIS [47, 48].

На Рисунке 4.10 приведено сравнение вклада времени построения предобуславливателя и времени итераций в общее время решения системы для предобуславливателей ILU0, ILUt, ILU2 и CE($r = 2$). Для метода BiCGStab требуется падение невязки до $\epsilon = 10^{-10}$.

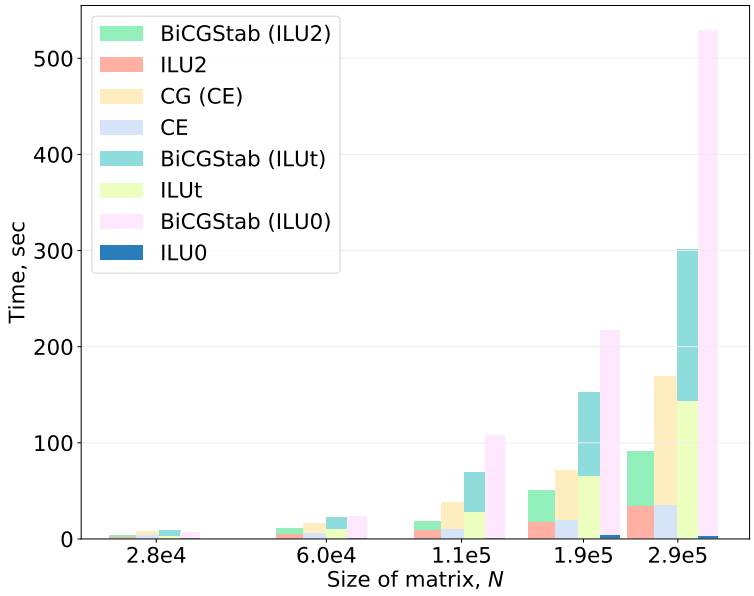
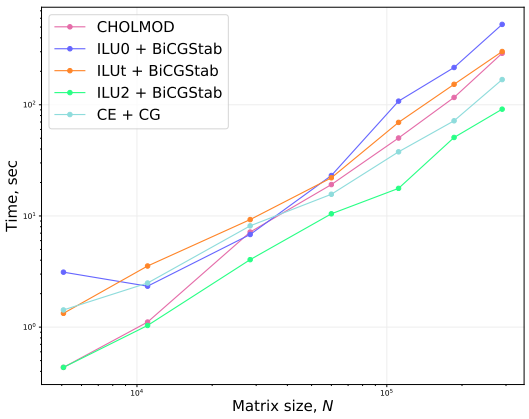
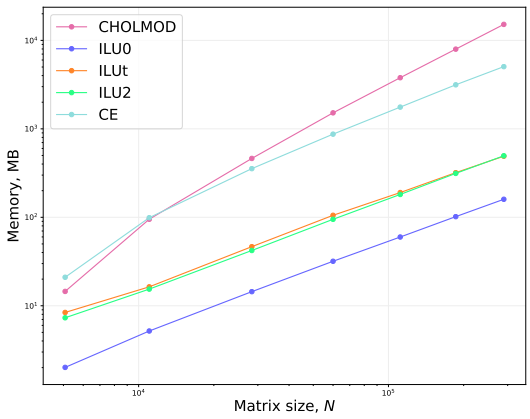


Рисунок 4.10: Сравнение предобуславливателей

Время построения CE($r = 2$) самое большое, однако,этот предобуславливатель значительно сокращает количество итераций метода BiCGStab. Сравним время и память требуемое на решение системы до точности $\varepsilon = 10^{-10}$ при помощи метода BiCGStab с различными предобуславливателями и метода CHOLMOD.



(a) Сравнение по времени.



(b) Сравнение по памяти

Рисунок 4.11: Итоговое сравнение солверов.

Приведем сравнение данных методов для различных точностей решения $\epsilon = 10^{-3}, 10^{-6}, 10^{-10}$.

Метод	CE(ε)	CE($r=2$)+BiCGStab	ILUt+BiCGStab	ILU0+BiCGStab	ILU2+BiCGStab	CHOLMOD
$\epsilon = 10^{-3}$						
Число итераций	-	128	118	218	1	-
Время решения, сек	38.6	71.5(34.1+ 37.4)	169.2(142.6+26.7)	66.9(2.5+64.4)	32.0(31.9 + 0.1)	-
Память, MB	5051	5051	487.4	159.8	496.0	-
$\epsilon = 10^{-6}$						
Число итераций	-	173	360	947	71	-
Время решения, сек	-	87.5(32.1+54.4)	201.8(136.1+65.6)	257.1(2.3+254.8)	48.3(30.8 + 17.5)	-
Память, MB	-	5051	479.2	159.8	496.0	-
$\epsilon = 10^{-10}$						
Число итераций	-	348	688	1868	220	-
Время решения, сек	-	168.4(35.1+133.2)	300.3(141.8+158.5)	528.7(2.4+526.3)	91.2(33.9 +57.3)	290.7
Память, MB	-	5051	491.2	159.8	496.0	15166.2

Таблица 4.4: Сравнение данных методов для различных точностей решения, $N = 2.8 \times 10^5$

4.1.4. Сравнение методов решения разреженных систем

В этом параграфе приводится сравнение скорости решения и затрат по памяти для четырёх больших симметричных положительно определённых разреженных матриц.

- Матрица M_1 (матрица s3dkt3m2 [1] из Florida Sparse Matrix Collection) размера $N = 90449$ содержит 1921955 ненулевых элементов.
- Матрица M_2 (матрица из Таблицы 4.3) размера $N = 237738$ содержит 237738 ненулевых элементов.
- Матрица M_3 (матрица из Таблицы 4.4) размера $N = 289566$ содержит 13117878 ненулевых элементов.
- Матрица M_4 (матрица af_1_k101 [23] из The SuiteSparse Matrix Collection) размера $N = 503625$ содержит 17550675 ненулевых элементов.

Использование тестовых матриц M_1 и M_4 предложено Ю. В. Василевским. Сравниваются следующие методы: итерациооный решатель BiCGStab с предобуславливателями CE($r = 2$), ILU2, а также прямые методы решения из библиотек CHOLMOD и UMFPACK [22].

В Таблице 4.5 приводятся результаты для методов решения с точностью $\epsilon = 10^{-3}$.

	M_1	M_2	M_3	M_4
ILU2+BiCGStab				
Время решения, сек	70.6	3.8	31.9	82.0
Память, MB	1063.3	75.6	496.0	2582.8
CE($r = 2$)+BiCGStab				
Время решения, сек	6.0	7.6	71.5	21.2
Память, MB	634	1042	5051	2146
UMFPACK				
Время решения, сек	3.0	-	40.8	-
Память, MB	311.9	-	6023.3	-

Таблица 4.5: Сравнение методов решения для различных тестовых матриц, $\epsilon = 10^{-3}$

В Таблице 4.6 приводятся результаты для методов решения с точностью $\epsilon = 10^{-6}$.

	M_1	M_2	M_3	M_4
ILU2+BiCGStab				
Время решения, сек	78.0	5.2	48.3	91.3
Память, MB	1063.3	75.6	496.0	2582.8
CE($r = 2$)+BiCGStab				
Время решения, сек	9.4	9.5	87.5	43.4
Память, MB	634	1042	5051	2146
UMFPACK				
Время решения, сек	-	-	-	30.3
Память, MB	-	-	-	2015.0

Таблица 4.6: Сравнение методов решения для различных тестовых матриц, $\epsilon = 10^{-6}$

В Таблице 4.7 приводятся результаты для методов решения с точностью $\epsilon = 10^{-10}$.

	M_1	M_2	M_3	M_4
ILU2+BiCGStab				
Время решения, сек	94.3	6.7	91.3	100.4
Память, MB	1063.3	75.6	496.0	2582.8
CE($r = 2$)+BiCGStab				
Время решения, сек	21.7	11.8	168.4	84.9
Память, MB	634	1042	5051	2146
UMFPACK				
Время решения, сек	-	2.2	-	-
Память, MB	-	180.6	-	-
CHOLMOD				
Время решения, сек	-	22.1	290.7	-
Память, MB	-	1457.8	15166.2	-

Таблица 4.7: Сравнение методов решения для различных тестовых матриц, $\epsilon = 10^{-10}$

Приведённые в Таблицах 4.5, 4.6, 4.7 результаты показывают, что для разных матриц и для решения систем с разной точностью могут оказаться эффективными разные методы решения, в том числе, и метод CE факторизации.

4.2. Разреженная факторизация блочно-малоранговых матриц

4.2.1. Расширенная разреженная факторизация

Электростатическая задача

В качестве модельной задачи рассмотрено граничное интегральное уравнение в следующей форме

$$\int_{\Omega} \frac{q(y)}{|x-y|} dy = f(x), \quad x \in \Omega, \tag{4.4}$$

где $\Omega = [0,1]^2$. Функция $f(x)$ задана и $q(y)$ требуется найти. Уравнение (4.4) дискретизировано с использованием метода коллокации с кусочно-постоянными базисными функциями на треугольной сетке Ω_N с N элементами (смотри Рисунок 4.13) Элементы матрицы могут быть вычислены аналитически [43]. Матрица

A аппроксимирована в $\mathcal{H}^2$ формате с использованием пакета h2tools [55]. SE форма матрицы A приведена на Рисунке 4.12 для $N = 1196$.

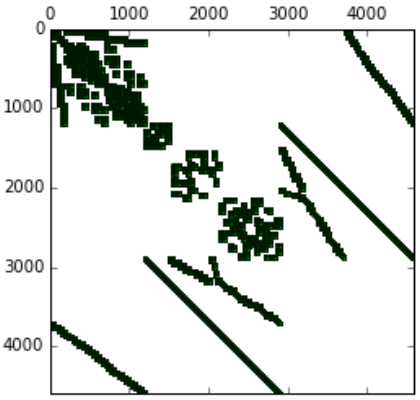


Рисунок 4.12: SE форма матрицы A

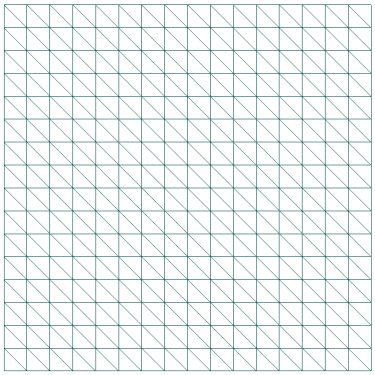


Рисунок 4.13: Треугольная сетка Ω_N

Метод 1. В Таблице 4.8 приведены результаты работы Метода 1 (прямой разреженный метод решения применён к SE(A)). Недостатком данного метода являются большие затраты по памяти.

N	time, (s)	Mem, (MB)
3928	2.585	376.85
13640	37.76	2527.7
28080	234	5012.1
59428	—	—
98936	—	—

Таблица 4.8: Время и память требуемые для работы Метода 1

Метод 2. Сходимость метода GMRES [71] с SE-ILUt предобуславливателем и блочным предобуславливателем приведена на Рисунке 4.14.

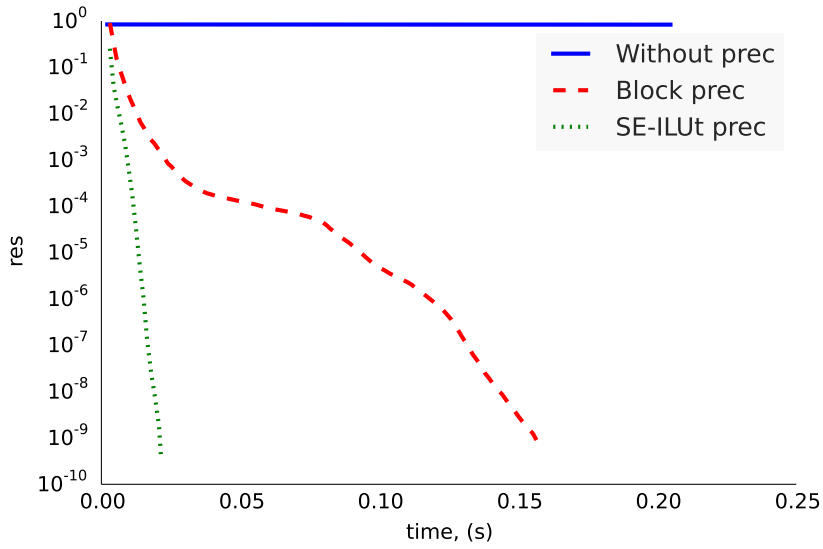


Рисунок 4.14: Метод 2: сходимость GMRES с различными предобуславливателями, $N = 1964$

Число итераций с предобуславливателем SE-ILUT значительно ниже, чем число итераций с блочным предобуславливателем, однако, вычисление блочного предобуславливателя имеет меньшую вычислительную сложность. Данный факт проиллюстрирован в Таблице 4.9, система решалась до точности $\epsilon = 10^{-10}$.

N	Block prec + GMRES, (s)	SE-ILUt prec + GMRES, (s)
3928	0.085 + 1.28	1.75 + 0.17
13640	0.23 + 5.6	9.17 + 0.52
28080	0.53 + 11.8	27.17 + 0.91
59428	1.34 + 34.8	75.02+ 3.13
98936	3.28 + 59.13	134.11+ 10.2

Таблица 4.9: Время построение предобуславливателя и GMRES итераций

Метод 3. Сходимость метода GMRES обратным предобуславливателем приведён на Рисунке 4.15, $N = 28080$.

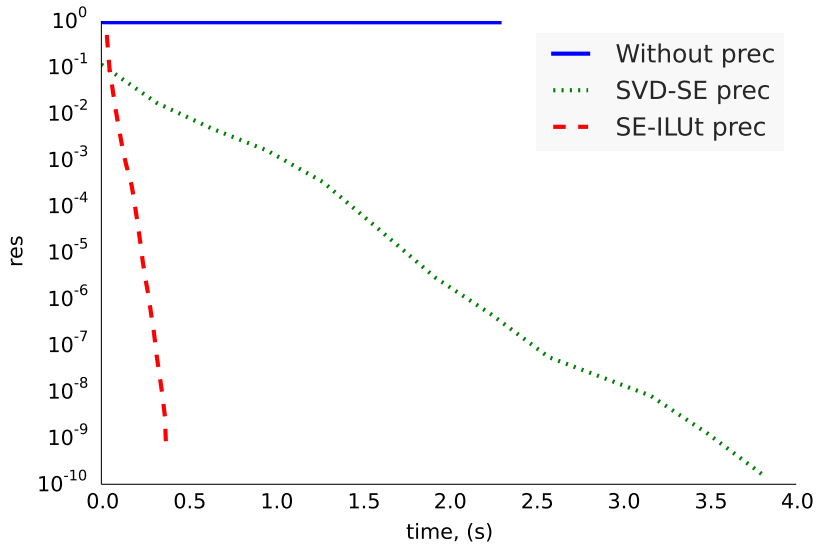


Рисунок 4.15: Метод 3: сходимость GMRES, $N = 28080$

В Таблице 4.10 приведена общая вычислительная сложность для различных обратных предобуславливателей Шура. Отметим, что предобуславливатель SE-ILUt во второй колонке это предобуславливатель из предыдущего пункта. Также отметим, что использование обратного предобуславливателя Шура для исходной системы оказывается более эффективным, чем решение системы с SE матрицей.

Компрессия $\mathcal{H}^2$ матрицы приводит к более эффективному предобуславливателю.

N	SE-ILUt prec + GMRES	SVD-SE prec + GMRES, (s)
3928	2.13 + 0.03	0.21 + 0.11
13640	8.84 + 0.11	1.34 + 1.32
28080	24.8 + 0.35	8.35 + 2.94
59428	69.1 + 1.33	19.7 + 6.13
98936	150.2 + 4.38	40.01 + 15.03

Таблица 4.10: Время работы Метода 3 с различными обратными предобуславливателями Шура

Итоговое сравнение В Таблице 4.11 приведено итоговое сравнение предложенных методов для различных N. Системы решались до точности $\epsilon = 10^{-10}$.

Для Методов 2 и 3 время решения складывается из времени построения предобуславливателя и времени GMRES итераций)

	Метод 1	Метод 2		Метод 3	
N	Dir. sol, (s)	Block, (s)	SE-ILUt, (s)	SE-ILUt, (s)	SVD-SE, (s)
3928	2.585	3.215	2.11	2.16	0.32
13640	37.76	10.83	9.69	8.95	2.66
28080	234	22.33	28.08	25.47	11.28
59428	—	53.14	78.15	70.37	25.83
98936	—	122.41	144.31	154.58	55.04

Таблица 4.11: Сравнение по времени предложенных методов

В Таблице 4.12 представлены затраты по памяти приведённых алгоритмов для различных N.

	Метод 1	Метод 2		Метод 3	
N	Dir. sol, (MB)	Block, (MB)	SE-ILUt, (MB)	SE-ILUt, (MB)	SVD-SE, (MB)
3928	424	24.2	31.4	15.7	< 10
13640	2606	251.2	279.8	31.4	< 10
28080	14702	675.3	2464.9	219.8	47.1
59428	—	1301.0	9720.6	810.0	128.3
98936	—	6340.1	—	2028.8	221.5

Таблица 4.12: Потребности по памяти предложенных методов для различных N, прочерки прочерки означают что алгоритм превысил допустимый лимит памяти

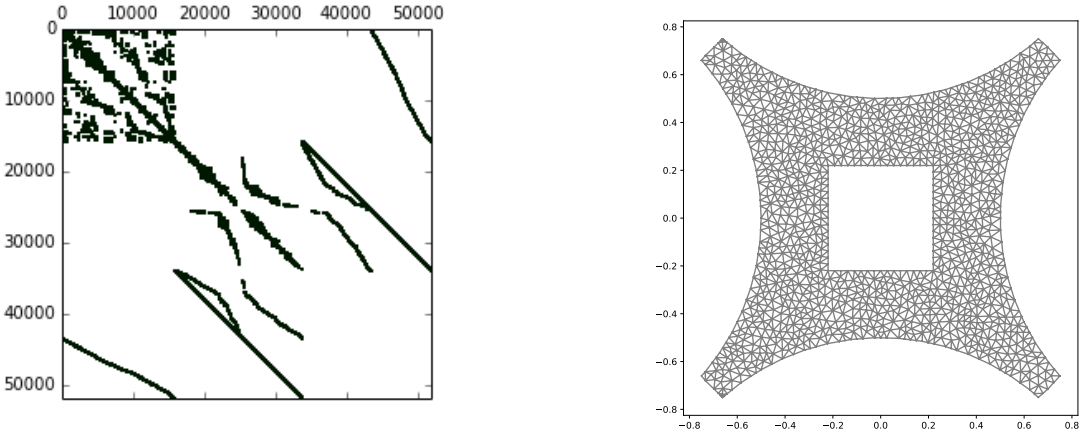
Метод 3 с SVD компрессией оказался самым быстрым методом и методом, требующим меньше всего памяти.

Гиперсингулярное интегральное уравнение для задачи обтекания

Вторая задача - это гиперсингулярное интегральное уравнение, к которому приводит задача обтекания части плоскости в трехмерном пространстве.

$$\int\limits_{\Omega} \frac{q(y)}{|x-y|^3}\, dy = f(x), \tag{4.5}$$

где Ω это часть плоскости в трехмерном пространстве, иллюстрация области Ω приведена на Рисунке 4.16b. Уравнение дискретизировано при помощи метода коллокации с кусочно-постоянными функциями на треугольной сетке, матрица аппроксимирована в $\mathcal{H}^2$ формате с помощью пакета h2tools. Иллюстрация SE формы матрицы A для $N = 15340$ приведена на Рисунке 4.16a.



(a) SE форма матрицы A

(b) Поверхность Ω

Рисунок 4.16: Поверхность Ω и SE форма матрицы A

В Таблице 4.13 представлено сравнение предложенных методов.

		Метод 1	Метод 2		Метод 3	
	N	Dir. sol	Block	SE-ILUt	SE-ILUt	SVD-SE
Память, (M)	7208	2806	115.8	84.1	30.2	< 10
Время, (сек)		30.2	-	-	1.4	0.9
Память, (M)	15340	-	420.1	214.2	101.2	52.5
Время, (сек)		-	-	-	7.2	3.4
Память, (M)	34112	-	980.1	756.5	379.1	107.3
Время, (сек)		-	-	-	10.2	6.4
Память, (M)	67101	-	2021.0	1501.3	623.7	248.3
Время, (сек)		-	-	-	28.3	13.4

Таблица 4.13: Время решения и память для предложенных методов для задачи (4.5)

4.2.2. Не-расширенная разреженная факторизация

Алгоритм спарсификации реализован на языке программирования Python в качестве дополнения к пакету h2tools [55]. Пример програмного кода приведен в параграфе 4.2.2. Все тесты проводились на MacBook Air с процессором 1.3GHz Intel Core i5 и 4GB 1600 MHz DDR3 RAM.

В начале Раздела приводится время построения разреженной факторизации и численно показывается, что матрица S в разреженной факторизации

$$A = USV^{\top}$$

действительно разреженная. Далее рассматривается разреженная факторизация $\mathcal{H}^2$ матрицы с применённым прямым методом решения в качестве метода решения систем с $\mathcal{H}^2$ матрицей и данный подход сравнивается с прямым методом решения HODLR. И наконец, рассматривается разреженная факторизация матрицы S факторизованной с помощью метода ILUt в качестве предобуславливателя для итерационного метода решения GMRES с $\mathcal{H}^2$ матрично-векторным произведением.

Интерфейс программного кода

В Листинге 2 приведен пример простой программы, которая реализует разреженную факторизацию $\mathcal{H}^2$ матрицы.

```
# Import main package:
import h2_to_sparse as h2ts

# Import other tools:
from helpers.gen_mat import h2_gen()

# Using h2tools generate H2 matrix:
matrix = h2_gen()

# Apply sparsification and obtain factors S, U, V
S,U,V = h2ts.convert_h2_to_sparse(matrix2)
```

Листинг 2: Разреженная факторизация $\mathcal{H}^2$ матрицы

Разреженность фактора S и время работы факторизации

В параграфе 3.2.2 аналитически изучалась разреженность матрицы S. Здесь приводится численная иллюстрация разреженности матрицы S. Тесты проводились на двух $\mathcal{H}^2$ матрицах.

***Пример* 4.2.1.**

$$A_{ij} = \begin{cases} \frac{1}{|r_i - r_j|} & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}, \tag{4.6}$$

где $r_i \in \mathbb{R}^2$ или $r_i \in \mathbb{R}^3$ это позиция i-го элемента. На Рисунке 4.17 эта $\mathcal{H}^2$ матрица обозначена «inv».

***Пример* 4.2.2.**

$$A_{ij} = 2\delta_{ij} + \exp(-|r_i - r_j|^2), \tag{4.7}$$

где $r_i \in \mathbb{R}^2$ или $r_i \in \mathbb{R}^3$ это позиция i-го элемента. На Рисунке 4.17 эта $\mathcal{H}^2$ обозначена «exp».

Для обоих примеров точность $\mathcal{H}^2$ аппроксимации составила $\epsilon = 10^{-6}$.

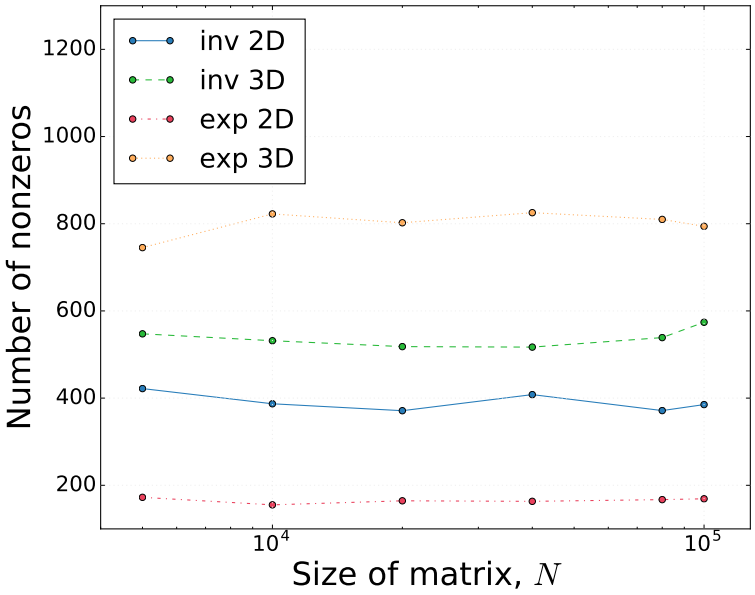


Рисунок 4.17: Число ненулевых элементов в строке матрицы S для Примеров 4.2.1 и 4.2.2 в 2D и 3D.

Число ненулевых элементов в строке не растёт с ростом размера матрицы, значит матрица S действительно разреженная.

На Рисунке 4.18 показано время факторизации $\mathcal{H}^2$ матриц из Примеров 4.2.1 и 4.2.2 в 2D и 3D обозначенные через ”inv” и ”exp” соответственно.

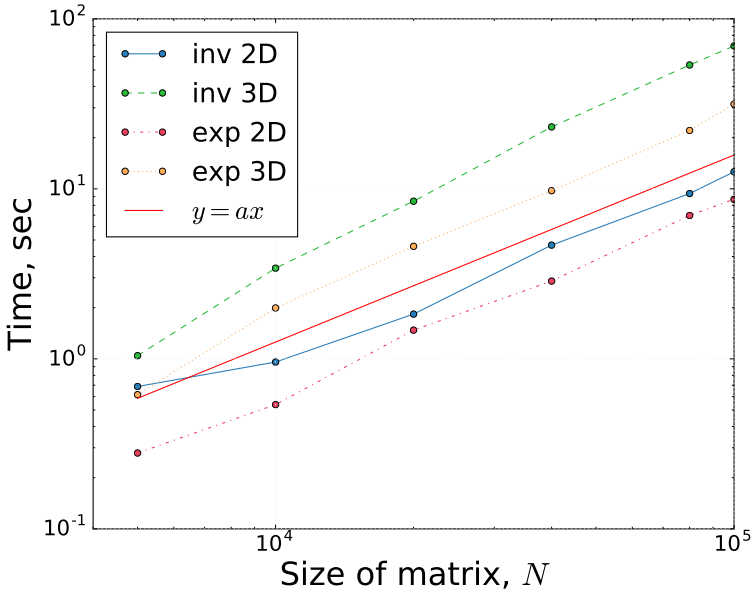


Рисунок 4.18: Время построения спарсификации для Примеров 4.2.1 и 4.2.2 в 2D и 3D.

Время спарсификации растёт почти линейно с ростом размера матрицы.

Сравнение с HODLR

Статья [7] рассматривает аппроксимацию HODLR плотной матрицы и факторизацию в качестве эффективного метода подсчета определителя плотной матрицы. Мы предлагаем аппроксимацию в $\mathcal{H}^2$ формате, спарсификацию и разреженную факторизацию в качестве альтернативы. Тесты проводились для трехмерных данных, для матрицы

$$A_{ij} = 2\delta_{ij} + \exp(-|r_i - r_j|^2),$$

где $r_i \in \mathbb{R}^3$ это положение i-го элемента. Матрица A - это матрица ковариации, получаемая при моделировании гауссовских процессов, их подробное описание находится в главе 5. На Рисунке 4.19 приведено сравнение по времени этих подходов.

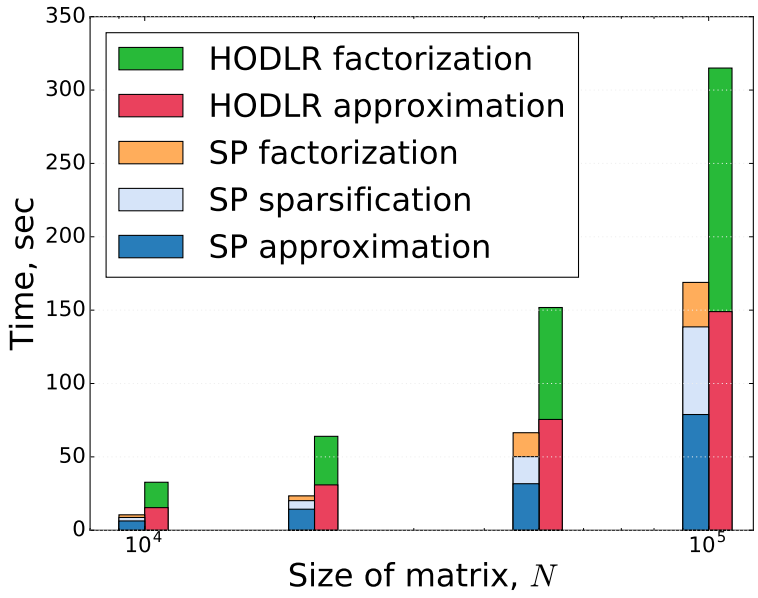


Рисунок 4.19: Сравнение подхода спарсификации $\mathcal{H}^2$ с HODLR методом решения в 3D.

Общее время решения приведено на Рисунке 4.20.

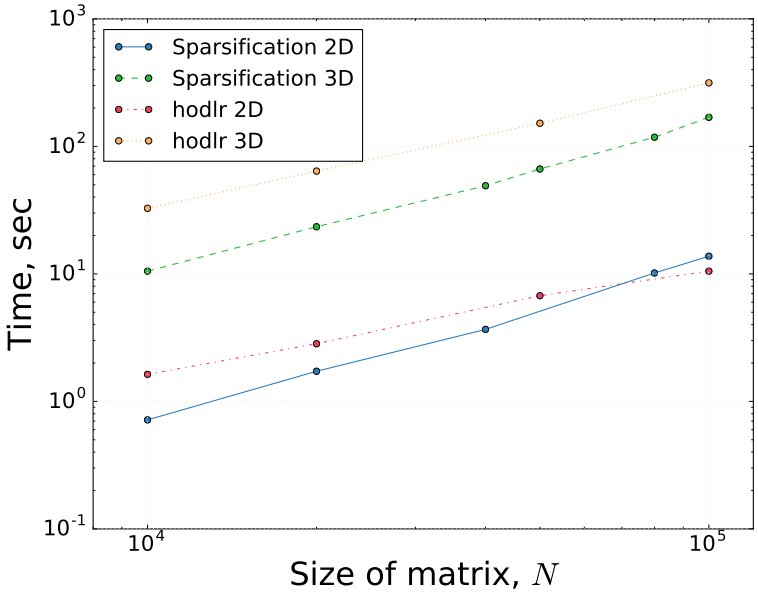


Рисунок 4.20: Сравнение по времени подхода со спарсификацией и методом решения HODLR в 2D и 3D.

Метод спарсификации в качестве предобуславливателя

$\mathcal{H}^2$ матрица это эффективный инструмент для умножения матрицы на вектор. Это позволяет применять итерационные методы, такие как GMRES для решения систем с $\mathcal{H}^2$ матрицами. Но предобуславливание является сложной задачей в связи с трудностями факторизации $\mathcal{H}^2$ матриц. Предлагается использовать приближенную факторизацию спарсификации $\mathcal{H}^2$ матрицы в качестве предобуславливателя для итерационных методов. Для тестов использовались случайно распределённые элементы в 3D со следующей матрицей взаимодействия.

***Пример* 4.2.3.**

$$A_{ij} = \begin{cases} 1 & \textit{if } i = j \\ \frac{|r_i - r_j|}{d} & \textit{if } 0 < |r_i - r_j| < d \\ \frac{d}{|r_i - r_j|}, & \textit{if } |r_i - r_j| \geqslant d \end{cases},$$

где $r_i \in \mathbb{R}^3$ *это позиция* i*-го элемента.*

Данный пример использовался для тестов в методе предобуславливания с помощью IFMM [20], поэтому для сравнения был выбран данный пример.

Данная матриц удобна для тестирования итерационных методов так как её число обусловленности значительно зависит от параметра d, чем больше d, тем больше число обусловленности.

Пример с хорошо обусловленной матрицей. Рассмотрим матрицу из Примера 4.2.3 с $d = 10^{-3}$, её число обусловленности $\mathrm{cond}(A) = 10$. Решаем эту систему используя итерационный метод решения GMRES для матрицы приближенной в $\mathcal{H}^2$ формате с точностью $\epsilon = 10^{-9}$ в качестве предобуславливателя используем $\mathcal{H}^2$ аппроксимацию матрицы A с точностью $\epsilon = 10^{-3}$, к которой применена спарсификация и факторизация ILUt. На Рисунке 4.21 приведена сходимость метода GMRES с разными пороговыми значениями ILUt. Требуемая невязка метода GMRES это $r = 10^{-10}$.

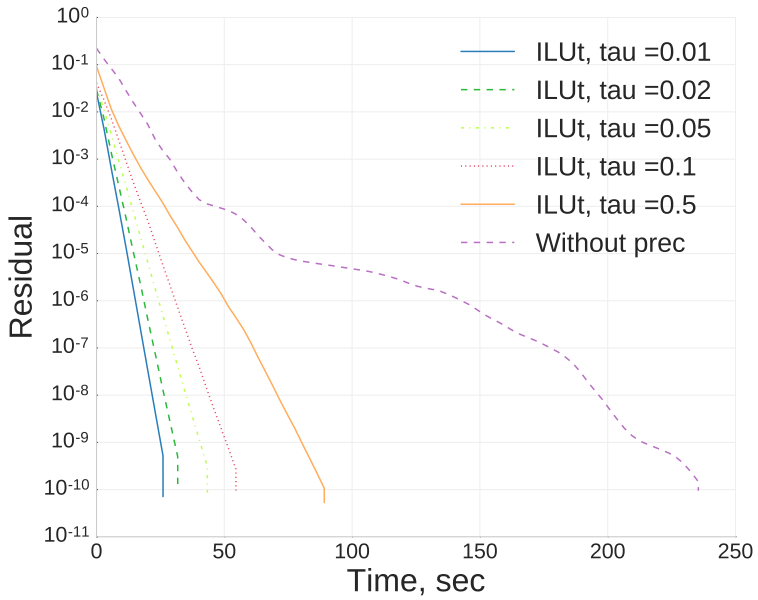


Рисунок 4.21: Сходимость GMRES с различными пороговыми значениями для факторизации ILUt

Обычный компромис: чем дольше строится предобуславливатель тем быстрее сходятся итерации. На Рисунке 4.22 показано полное время требуемое для решения системы (включая построение спарсификации).

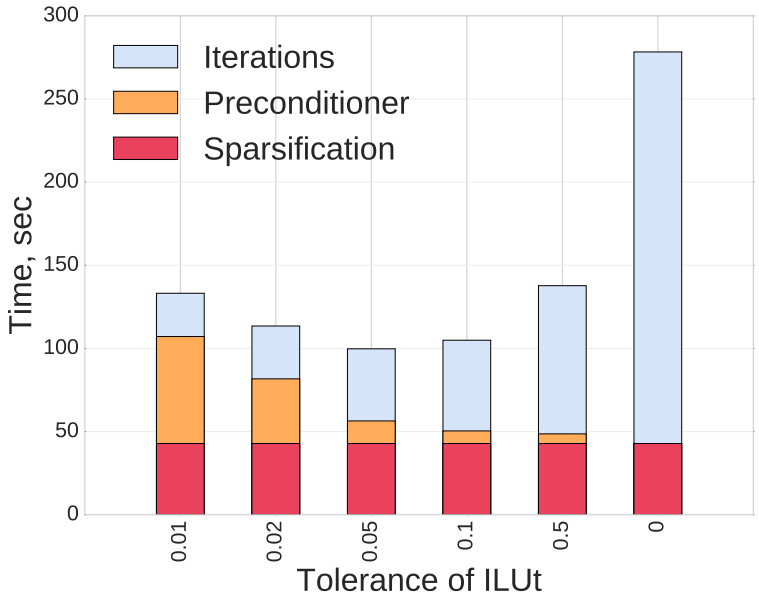
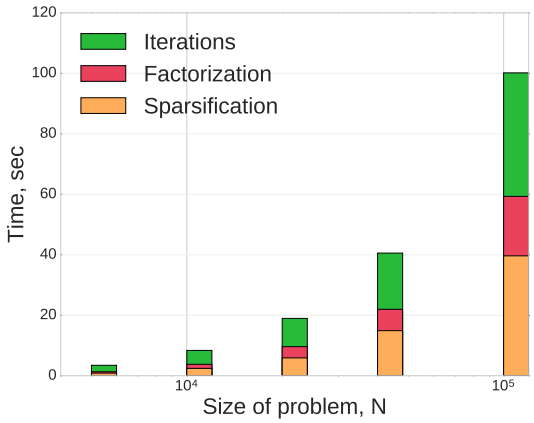
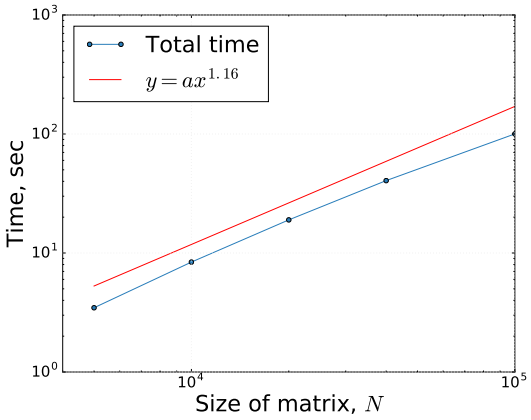


Рисунок 4.22: Вклад в общее время решения спарсификации, факторизации и итераций

На Рисунке 4.23a приводится общее время решения требуемое для спарсификации построения факторизации ILUt с пороговым значением $\tau = 2 \times 10^{-2}$ и временем итераций для $\mathcal{H}^2$ матрицы.



(a) Вклад в общее время решения спарсификации, факторизации и итераций



(b) Общее время решения

Рисунок 4.23: Общее время решения

Пример с плохо обусловленной матрицей. Рассмотрим матрицу из Примера 4.2.3 с $d = 10^{-2}$, её число обусловленности $\text{cond}(A) = 10^4$. На Рисунке 4.24 приведена сходимость до невязки $r = 10^{-10}$ для разных пороговых параметров факторизации ILUt.

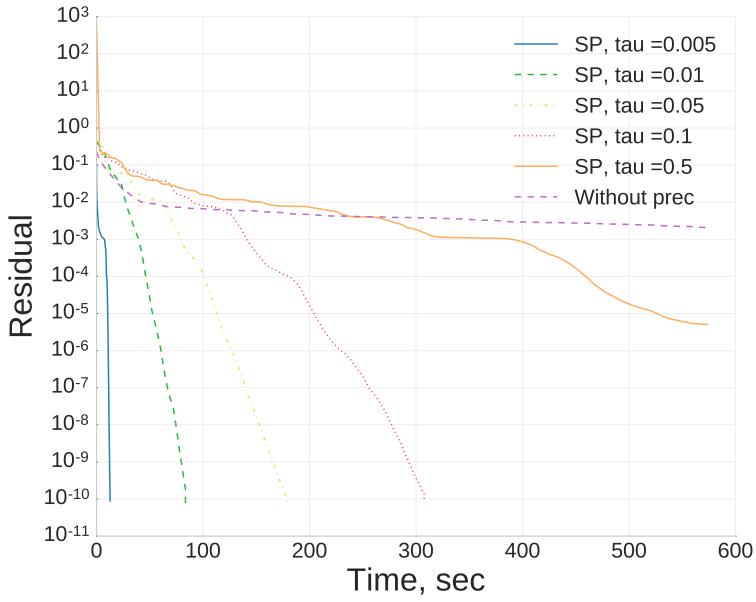


Рисунок 4.24: Сходимость GMRES при различных параметрах ILUt факторизации, $N = 10^5$

На Рисунке 4.25 приведено общее время решения для различных пороговых параметров ILUt.

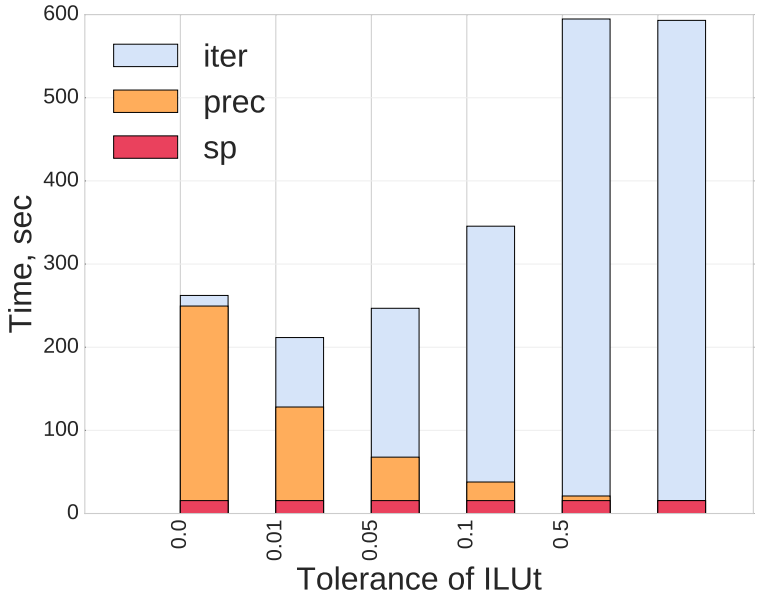
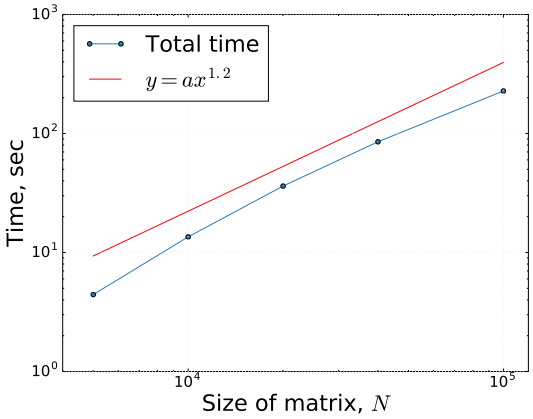


Рисунок 4.25: Вклад в общее время решения спарсификации, факторизации и итераций, $N = 10^5$

Заметим, что оптимальный пороговый параметр для факторизации ILUt для данной задачи это $\tau = 10^{-2}$. На Рисунке 4.26b приведено общее время требуемое для решения системы с оптимальным пороговым параметром факторизации ILUt.

(a) Вклад в общее время решения спарсификации, факторизации и итераций



(b) Общее время решения

Рисунок 4.26: Общее время решения

Для данной задачи и данного набора параметров общее время решения системы почти линейно.

Сравним полученные результаты (GMRES предобусловленный методом спарсификации) с результатами IFMM предобуславливателя [20] для одной задачи, одинаковых параметров и похожего вычислительного оборудования. Для тестов использовалась матрица из Примера 4.2.3 с элементами распределёнными в 3D, размером матрицы $N = 10^5$, итерационным методом GMRES и требуемой невязкой $r = 10^{-10}$, параметрами $d = 10^{-3}, 10^{-2}$ хорошо и плохо обусловленных систем. Мы представляем сравнение лучших временных результатов для двух методов.

	$d = 10^{-3}$	$d = 10^{-2}$
Время решения IFMM, sec	118	301
Время решения со спарсификацией, sec	96	218

Таблица 4.14: Сравнение с методом IFMM.

Отметим, что метод спарсификации реализован на языке программирования Python и может быть значительно ускорен переносом на другой язык (такой как C++ или Fortran).

4.3. Выводы по главе

В данной главе приводится описание программной реализации алгоритмов, описанных в главах 3 и 2. Библиотека, реализующая алгоритм компрессии и исключения тестируется на задачах Пуассона, диффузии и упругой деформации. Предложенная библиотека сравнивается по времени (и в некоторых случаях по памяти) на этих задачах с другими прямыми и итерационными методами решения систем. Программная реализация расширенной разреженной факторизации используется в качестве предобуславливателя при численном решении задач дискретных вихрей и электростатики, значительно сокращая число итераций, требуемых для решения линейной системы. Программная реализация не-расширенной разреженной факторизации показывает свою эффективность по времени в сравнении с методами HODLR и IFMM на задаче регрессии на основе гауссовских процессов.

Глава 5

Приложение для задачи регрессии на основе гауссовских процессов

5.1. Постановка задачи

Одной из актуальных задач прикладной статистики является задача регрессии, в которой набор данных имеет неизвестные корреляции в шуме. Эффект этого коррелированного шума часто трудно оценить, однако его влияние на окончательный ответ может быть велико. В данном разделе рассматривается искусственно сгенерированный набор данных с коррелированным шумом и простой нелинейной моделью. Ковариационная структура в данных моделируется с использованием гауссовского процесса.

В ходе применения гауссовского процесса к задаче регрессии требуется многократно вычислять значение функции правдоподобия для различных наборов параметров, а значит необходимо вычислять логарифм определителя матрицы ковариации и решать с ней линейные системы. Матрица ковариации, вообще говоря, является плотной, поэтому как вычисление логарифма её определителя так и решение системы с ней является трудоёмкой задачей.

Существует несколько методов аппроксимации регрессии на основе гаусовских процессов в машинном обучении [12, 72], в работах [50, 52], например, делается аппроксимация детерминанта с помощью методов Монте-Карло. В работе [7]

показано, что она обладает $\mathcal{H}^2$ структурой. Данное наблюдение может существенно ускорить вычисления. В данной главе предлагается аппроксимировать данную матрицу в $\mathcal{H}^2$ формате, затем приводить её к разреженному виду при помощи метода описанного в разделе 3.2, затем приближенно факторизовать их при помощи пакета CHOLMOD [24] и затем считать логарифм её определителя и решать с ней систему. В работе [7] определитель матрицы ковариации предлагалось считать с помощью аппроксимации в формате HODLR (Hierarchically Off-Diagonal Low-Rank) и её факторизации, в данном разделе будет показано, что предложенный автором метод подсчета детерминанта является более выгодным по времени и по памяти.

Приведем определение гауссовского процесса.

***Определение* 5.1.1.** *Гауссовский процесс (ГП) это совокупность случайных величин любое конечное число которых имеет совместное гауссово распределение. Случайные величины из ГП должны обладать свойством согласованности: если ГП задаёт*

$$y^{(1)}, y^{(2)} \sim \mathcal{N}(\mu, \Sigma),$$

тогда оно также задаёт

$$y^{(1)} \sim \mathcal{N}(\mu_1, \Sigma_{11}),$$

где

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.$$

ГП полностью определяется средней функцией и положительно определённой матрицей ковариации.

Подробное описание и приложение в задаче регрессии можно найти в работе [65].

5.2. Простой одномерный пример

Для моделирования коррелированного шума при помощи гауссовского процесса, а также для решения задачи регрессии в данной работе используется пакет

George [27]. Все вычисления проведены в одно-процессорном режиме на сервере с 32 Intel® Xeon® E5-2640 v2 (20M Cache, 2.00 GHz) процессорами и с 256GB RAM.

Наборы данных генерируются следующим образом: для N случайных точек $t_i, i \in \overline{1 \dots N}$ генерируются значения функции

$$y_i = f(t_i),$$

где

$$f = \alpha \exp(-\frac{[t-l]^2}{m}),$$

где $\alpha = -1$ это амплитуда, $l = 0.1$ это смещение, $m = 0.4$ это ширина. Шум генерируется при помощи гауссовского процесса с матрицей ковариации

$$K_{ij} = \sigma_i^2 \delta_{ij} + k(r),$$

где $\sigma_i^2 = 0.3$, а ядро $k(r)$ задаётся следующим образом:

$$k(r_{ij}) = \exp(-\frac{r^2}{2}),$$

где $r_{ij} = |t_i - t_j|$.

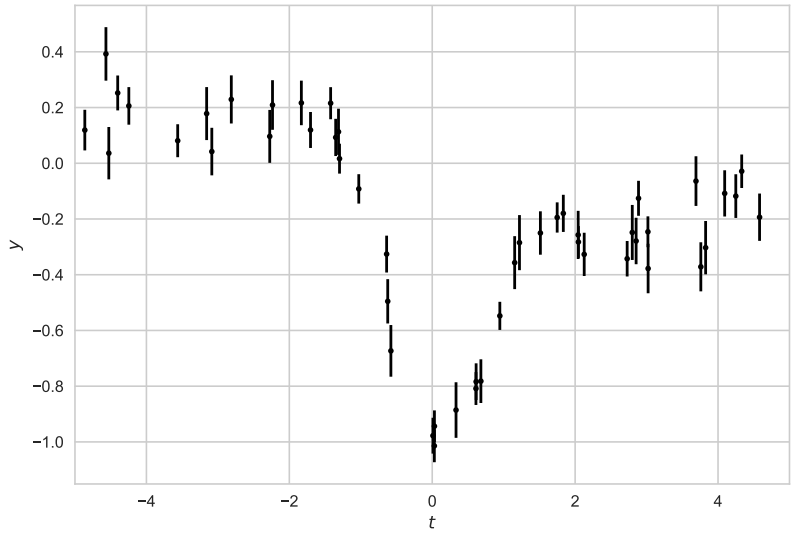


Рисунок 5.1: Пример набора данных для указанных выше параметров, $N = 50$

Для полученных искусственных данных ищется функция в следующем виде:

$$f_\theta = ct + d + \alpha \exp(-\frac{[t-l]^2}{m}). \tag{5.1}$$

В нашем примере шум предполагается коррелированным и моделируется при помощи гауссовского процесса со следующей матрицей ковариации:

$$K_{ij} = \sigma_i^2 \delta_{ij} + k(r), \tag{5.2}$$

с ядром

$$k(r_{ij}) = b^2 \left(1 + \frac{\sqrt{3}r}{\tau}\right) \exp\left(-\frac{\sqrt{3}r}{\tau}\right), \tag{5.3}$$

где $r_{ij} = |t_i - t_j|$, b^2 и τ это параметры модели. Таким образом, по исходным данным $t_i, y_i \quad i \in \overline{1 \dots N}$ требуется восстановить вектор параметров

$$\theta = \{c, d, \alpha, l, m, b, \tau\}. \tag{5.4}$$

Запишем функцию правдоподобия:

$$\ln p(\{y_i\}|\{t_i\}, \{\sigma_i^2\}, \theta) = -\frac{1}{2} e^\top K^{-1} e - \frac{1}{2} \ln \det K - \frac{N}{2} \ln 2\pi, \tag{5.5}$$

где

$$e = \begin{bmatrix} y_1 - f_\theta(t_1) \\ y_2 - f_\theta(t_2) \\ \vdots \\ y_N - f_\theta(t_N) \end{bmatrix}. \tag{5.6}$$

Параметры определяются при помощи метода Монте-Карло Марковских цепей (Markov chain Monte Carlo, MCMC) [31]. Данный метод приводит к следующим результатам:

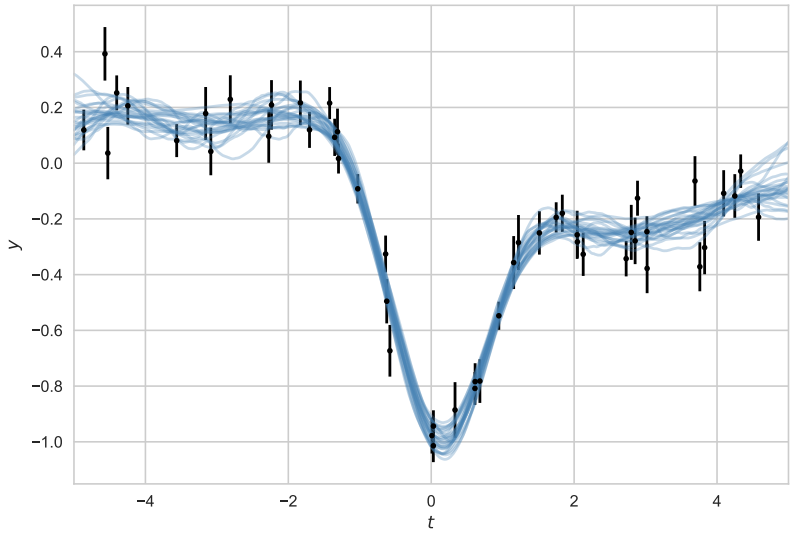


Рисунок 5.2: Результаты регрессии для корректированного шума

На Рисунке 5.2 приведены функции, сгенерированные при помощи 24 случайно выбранных наборов оцененных параметров. Для сравнения также рассмотрим случай, когда шум предполагается некоррелированным, тогда функция правдоподобия будет иметь следующий вид:

$$\ln p(\{y_i\}|\{t_i\},\{\sigma_i^2\},\theta) = -\frac{1}{2}\sum_{i=1}^{N}\frac{(y_i - f_\theta(t_i))^2}{\sigma_i^2},$$

Параметры в методе с некоррелированным шумом определяются также при помощи метода Монте-Карло Марковских цепей. Данный метод приводит к следующим результатам:

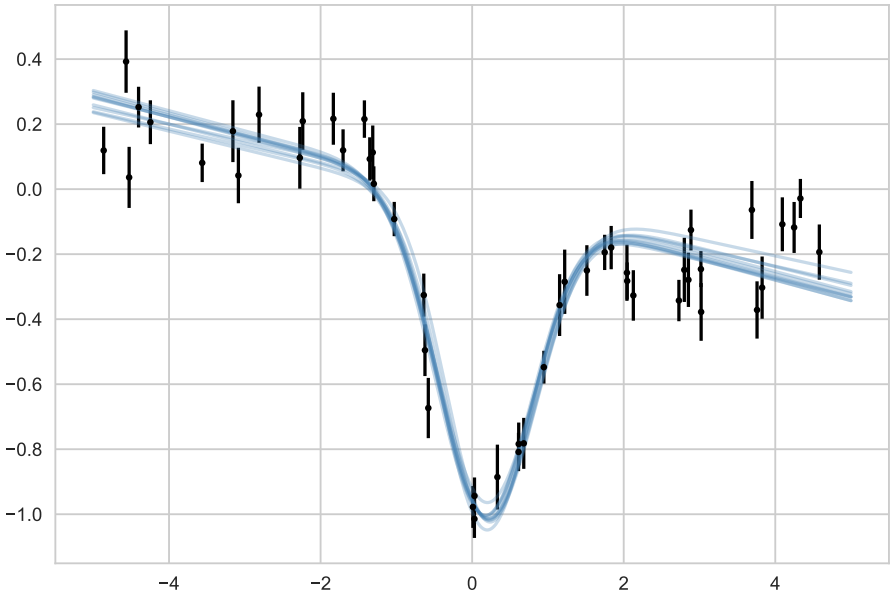
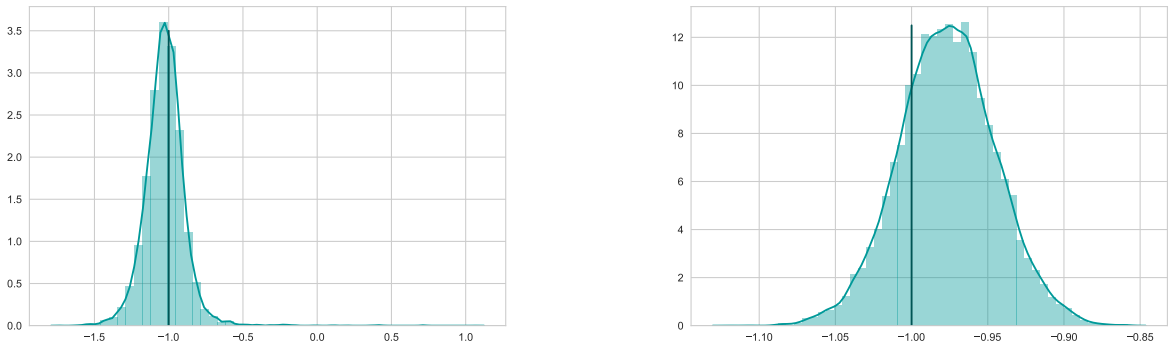


Рисунок 5.3: Результаты регрессии для некоррелированного шума

На Рисунке 5.3 приведены функции, сгенерированные при помощи 24 случайно выбранных наборов оцененных параметров для задачи, в которой шум предполагался некоррелированным. Метод, при котором шум предполагался некоррелированным, уступает в точности методу с коррелированным шумом, рассмотрим параметры, полученные в ходе регрессии двух этих случаях.

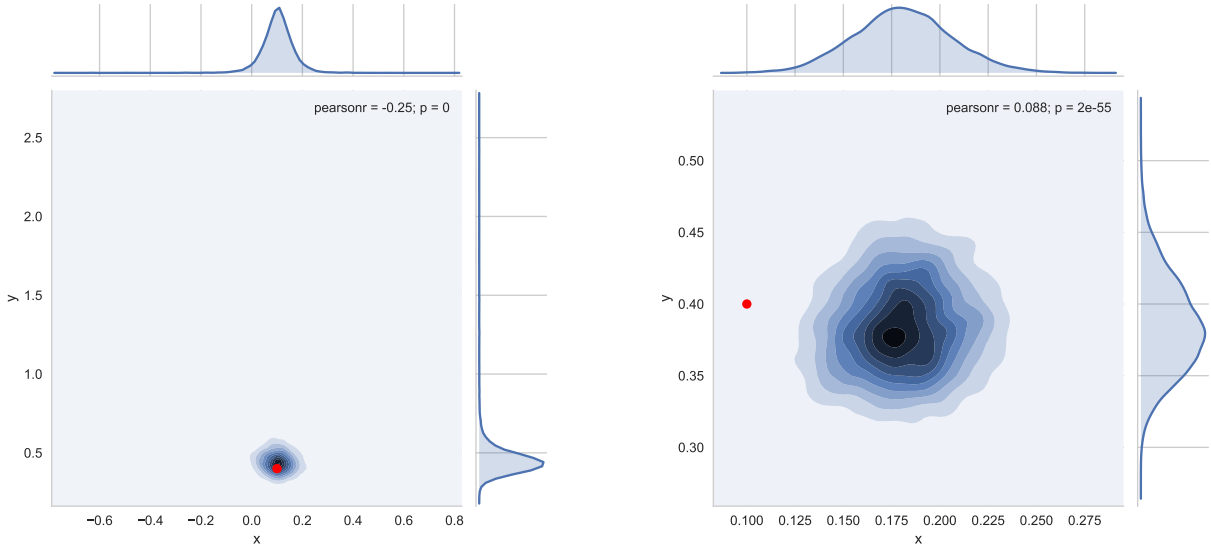


(a) коррелированный шум

(b) Некоррелированный шум

Рисунок 5.4: Распределение параметра α для разных типов моделирования шума

На Рисунке 5.4 прямая линия отмечает верное значение параметра α. Для некоррелированного шума параметр оценен с большой погрешностью. Рассмотрим два других параметра, l и σ.



(a) коррелированный шум

(b) Некоррелированный шум

Рисунок 5.5: Распределение параметров l и σ для разных типов моделирования шума

На Рисунке 5.15 красная точка отмечает верное значение параметров l и σ. Для некоррелированного шума параметры оценены с большой погрешностью. Графики, приведенные выше показывают, что метод с моделированием коррелированного шума (то есть моделирования шума при помощи гауссовского процесса) более эффективен для оценки параметров, чем метод, предполагающий белый шум. Оценим коэффициент детерминации двух рассмотренных выше моделей.

Коэффициент детерминации R^2 оценивает с какой вероятностью модель будет хорошо предсказывать будущие результаты измерений. Наилучший возможный результат это 1.0, коэффициент детерминации R^2 может быть и отрицательным. Константная модель, которая всегда предсказывает ожидаемое значение y вне зависимости от входных параметров, будет иметь $R^2 = 0.0$.

***Определение* 5.2.1** (Коэффициент детерминации)**.** *Если $\hat{y}_i$ это предсказанное значение i-х результатов измерений и y_i это соответствующее верное значение, тогда тогда параметр R^2 оцененный на n_s наборах измерений определяется как*

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_s-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_s-1}(y_i - \bar{y}_i)^2},$$

где $\bar{y}_i = \frac{1}{n_s}\sum_{i=0}^{n_s-1} y_i$

В Таблице 5.1 для разного числа измерений n_s приведены коэффициенты детерминации для моделей, рассмотренных выше.

n_s	50	100	200	400
Модель с кор. шумом,	$1 - 3.4 \times 10^{-6}$	-2.8×10^{-5}	$1 - 1.2 \times 10^{-5}$	$1 - 6.3 \times 10^{-6}$
Модель с некор. шумом,	$1 - 7.3 \times 10^{-2}$	$1 - 7.2 \times 10^{-2}$	$1 - 9.1 \times 10^{-2}$	$1 - 7.5 \times 10^{-2}$

Таблица 5.1: Коэфициент детерминации для моделей с коррелированнм шумом и без него.

Таким образом, модель с коррелированным шумом более эффективна для оценки параметров. Дальше, для 2D и 3D задач, рассматривается только данная модель. Отметим, что логарифм определителя матрицы ковариации и решение системы с матрицей ковариации требуется вычислять именно в этой модели.

5.3. Двумерная задача

Для двумерной задачи регрессии наборы данных генерируются аналогично одномерному случаю следующим образом: для N случайных точек $t_i \in \mathbb{R}^2, i \in \overline{1 \dots N}$ генерируются значения функции

$$y_i = f(t_i),$$

где

$$f = \alpha \exp(-\frac{[|t| - l]^2}{m}), \tag{5.7}$$

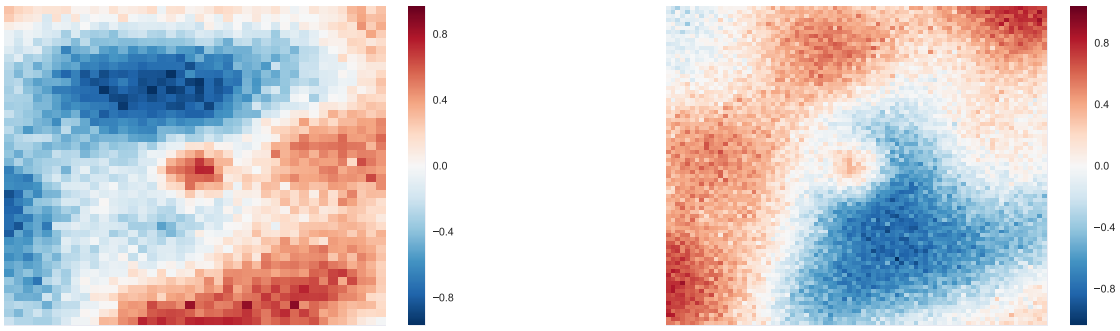
где $\alpha = -1$ это амплитуда, $l = 0.1$ это смещение, $m = 0.4$ это ширина. Шум как и в одномерном случае генерируется при помощи гауссовского процесса с матрицей ковариации

$$K_{ij} = \sigma_i^2 \delta_{ij} + k(r), \tag{5.8}$$

где $\sigma_i^2 = 0.3$, а ядро $k(r)$ задаётся следующим образом:

$$k(r_{ij}) = \exp(-\frac{r^2}{2}), \tag{5.9}$$

где $r_{ij} = |t_i - t_j|$. На Рисунке 5.6 приведены примеры двумерных функций сгенерированных при помощи процедуры, описанной выше.



(a) $N = 1600$ (b) $N = 6400$

Рисунок 5.6: Примеры наборов данных $y_i(t)$ на которых проводидись вычисления

Для полученных синтетических данных решается задача регрессии, функция ищется в следующем виде:

$$f_\theta = c \cdot t + d + \alpha \exp(-\frac{[|t| - l]^2}{m}), \tag{5.10}$$

где $c \in \mathbb{R}^2$, $d \in \mathbb{R}$.

Шум предполагается коррелированным и моделируется при помощи гауссовского процесса с матрицей ковариации такой же как и в одномерной задаче (5.2) с ядром (5.3). Таким образом, по исходным данным $t_i, y_i \quad i \in \overline{1 \dots N}$ требуется восстановить вектор параметров

$$\theta = \{c, d, \alpha, l, m, b, \tau\}. \tag{5.11}$$

Функция правдоподобия имеет вид (5.5), где e представляется в виде (5.6). Параметры определяются при помощи метода Монте-Карло Марковских цепей. В

процессе подсчета требуется считать логарифм определителя плотной матрицы K и решать систему с ней. Так как матрица ковариации симметричная и положительно определенная, для этого необходима посчитать её факторизацию Холецкого. По умолчанию в пакете George матрица факторизуется и при помощи пакета NumPy [58] с помощью инструментов для плотных матриц, это приводит к значительным затратам по памяти и по времени. Также в пакете встроен решатель HODLR, он использует блочно-малоранговые техники для быстрой факторизации, однако так как метод HODLR это аналог одномерных $\mathcal{H}^2$ матриц, для 2D и 3D задач он не эффективен. Мы предлагаем использовать для аппроксимации разреженное представление $\mathcal{H}^2$ матрицы, факторизованное с помощью разреженных инструментов, будем называть этот метод “спарсификация $\mathcal{H}^2$ ”. Сравним время подсчета логарифма детерминанта матрицы K для методов HODLR и спарсификации $\mathcal{H}^2$ для параметра точности аппроксимации $\varepsilon = 10^{-2}$ с учетом вклада во время подсчета метода спарсификация $\mathcal{H}^2$ разных этапов вычислений: аппроксимация в виде разреженной матрицы, подсчет факторизации и вычисление логарифма детерминанта.

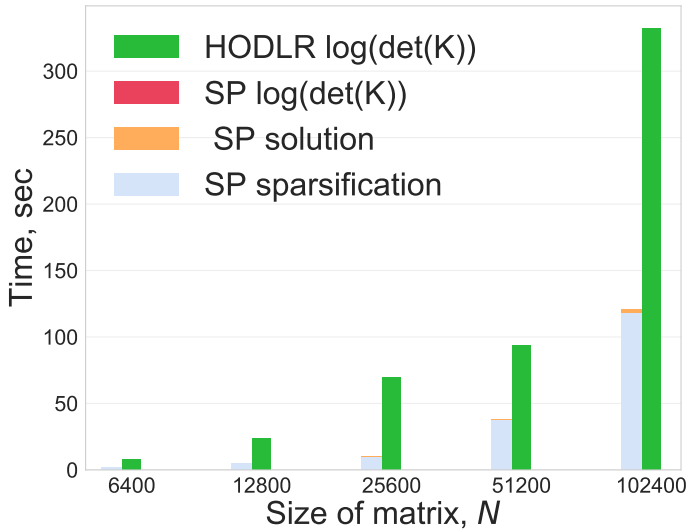


Рисунок 5.7: Время вычисления логарифма детерминанта матрицы K

Отметим, что кроме вычисления логарифма детерминанта необходимо вычислять и решение системы с матрицей K, однако после построения факторизации обе эти задачи решаются за незначительное время. Сравним общее время вычисления

функции правдоподобия для метода HODLR ($\varepsilon = 10^{-2}, 10^{-4}$), метода спарсификации $\mathcal{H}^2$ ($\varepsilon = 10^{-2}, 10^{-4}$) и плотного метода.

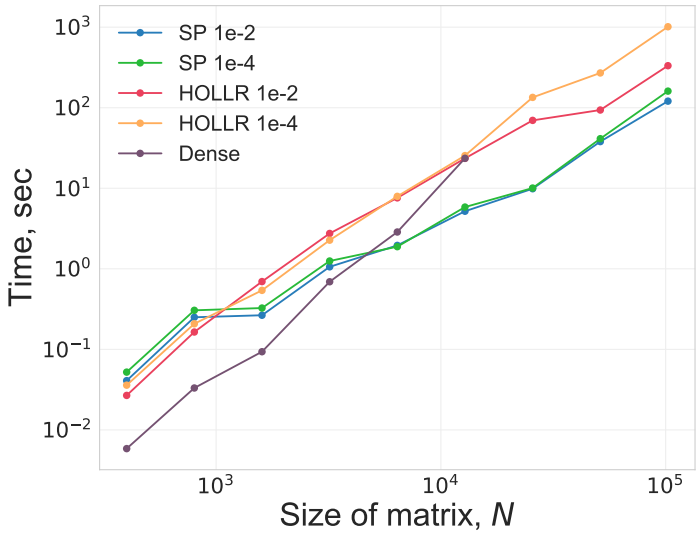
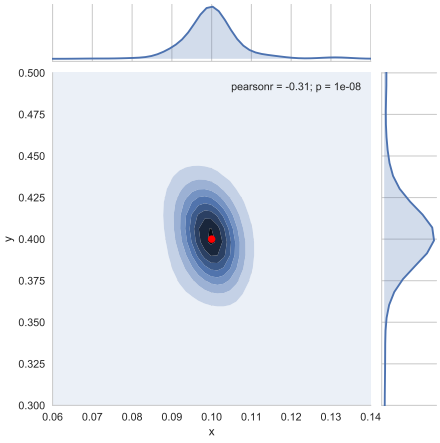
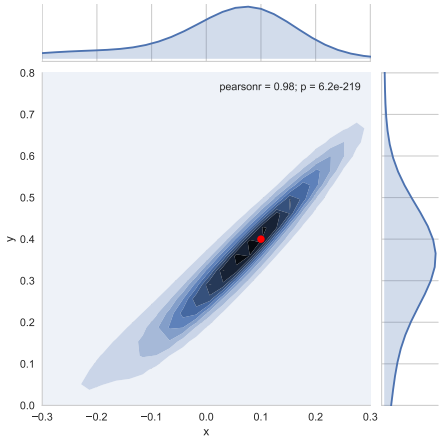


Рисунок 5.8: Время вычисления функции правдоподобия

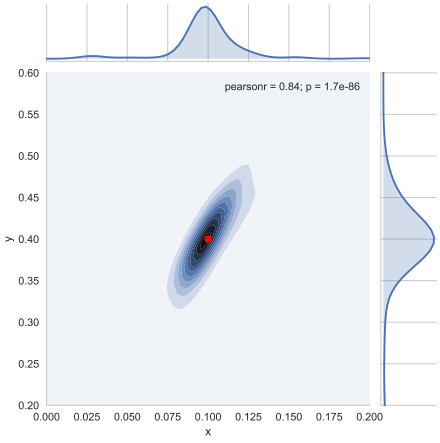
Отметим, что плотный метод требует слишком много памяти и не работает для матриц размером больше 10^4, однако, плотный метод решает системы с хорошей точностью. Покажем, что для данной задачи хорошая точность вычисления функции правдоподобия не требуется. Рассмотрим результаты работы метода Монте-Карло Марковских цепей и приведем распределение параметров l и σ^2 для различных методов факторизации матрицы K.



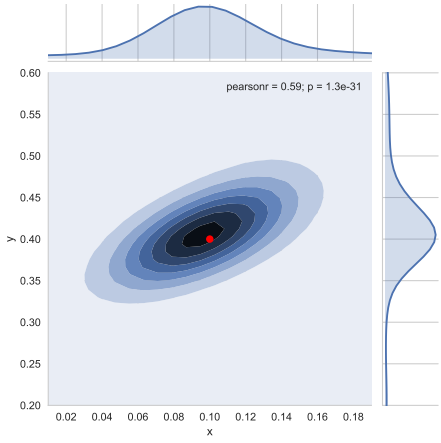
(a) Плотный метод



(b) HODLR, $\varepsilon = 1e-2$



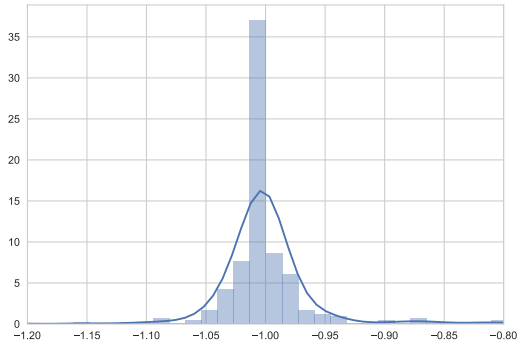
(c) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-2$



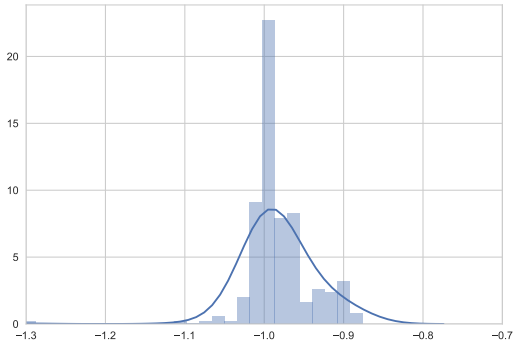
(d) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-4$

Рисунок 5.9: Распределение параметров l и σ^2

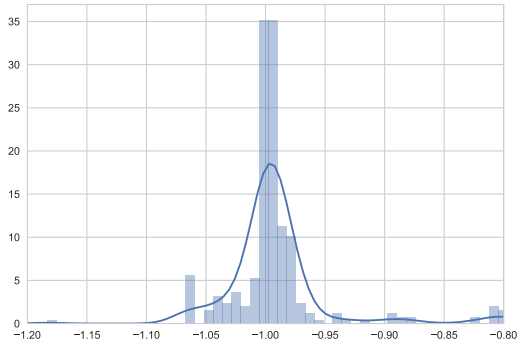
Также приведем распределение параметра α для различных методов факторизации матрицы K.



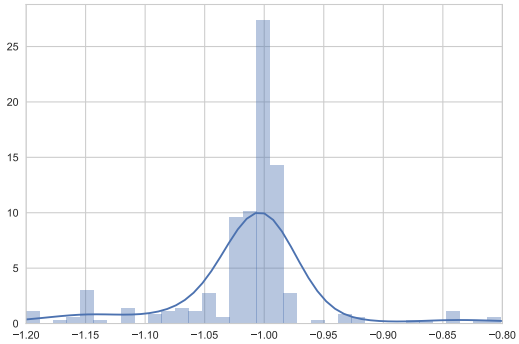
(a) Плотный метод



(b) HODLR, $\varepsilon = 1e-2$



(c) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-2$



(d) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-4$

Рисунок 5.10: Распределение параметра α

Распределение параметров при замене точного подсчета функции правдоподобия на приближенное изменяется незначительно. В Таблице 5.2 приведены коэффициенты детерминации для различных способов подсчета функции правдоподобия.

	R^2
HODLR, $\varepsilon = 1e-2$	$1 - 1.8 \times 10^{-3}$
HODLR, $\varepsilon = 1e-4$	$1 - 7.9 \times 10^{-5}$
Сп. $\mathcal{H}^2$, $\varepsilon = 1e-2$	$1 - 8.5 \times 10^{-5}$
Сп. $\mathcal{H}^2$, $\varepsilon = 1e-4$	$1 - 3.4 \times 10^{-5}$
Плотн.	$1 - 8.4 \times 10^{-6}$

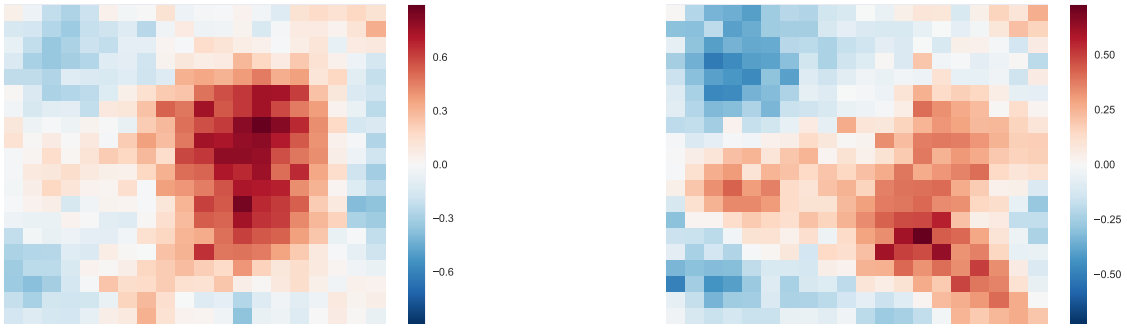
Таблица 5.2: Коэфициент детерминации для различных методов подсчета правдоподобия.

5.4. Трехмерная задача

Для трехмерной задачи регрессии наборы данных генерируются аналогично задаче в 2D. На Рисунке 5.11 приведен пример функции

$$y_i = f(t_i),$$

где $t_i \in \mathbb{R}^3, i \in \overline{1 \dots N}$, функция $f(t_i)$ задается формулой (5.7) параметры совпадают с 2D примером. Коррелированный шум задается при помощи гауцссовского процесса с матрицей ковариции (5.8) с ядром (5.9). Все параметры совпадают с двумерным случаем.



(a) $t^{(1)} = 3$ (b) $t^{(1)} = 5$

Рисунок 5.11: Примеры наборов данных на которых проводились вычисления, двух срезов трехмерной функции $y_i(t)$ по индексу $t^{(1)}$

Для полученных синтетических данных решается задача регрессии, функция ищется в следующем виде:

$$f_\theta = c \cdot t + d + \alpha \exp(-\frac{(|t| - l)^2}{m}), \tag{5.12}$$

где $c \in \mathbb{R}^3$, $d \in \mathbb{R}$. Шум моделируется при помощи гауссовского процесса матрицей ковариации (5.2) с ядром (5.3). Таким образом, по исходным данным $t_i, y_i \quad i \in \overline{1 \dots N}$ требуется восстановить вектор параметров

$$\theta = \{c, d, \alpha, l, m, b, \tau\}. \tag{5.13}$$

Функция правдоподобия имеет вид (5.5) где e представляется в виде (5.6). Параметры определяются при помощи метода Монте-Карло Марковских цепей. Для подсчета функции правдоподобия (5.5) как и в двумерном случае используются методы HODLR и спарсификация $\mathcal{H}^2$. Сравним время подсчета логарифма детерминанта матрицы K для методов HODLR и спарсификации $\mathcal{H}^2$ для параметра точности аппроксимации $\varepsilon = 10^{-2}$.

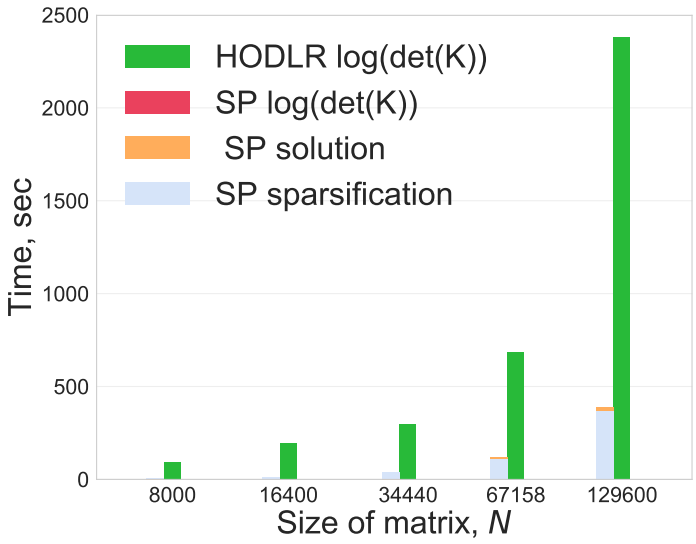


Рисунок 5.12

Также сравним общее время вычисления функции правдоподобия, самый трудоемкий процесс данного вычисления - факторизация плотной матрицы выполняется при помощи метода HODLR ($\varepsilon = 10^{-2}, 10^{-4}$), метода спарсификации $\mathcal{H}^2$ ($\varepsilon = 10^{-2}, 10^{-4}$) и плотного метода.

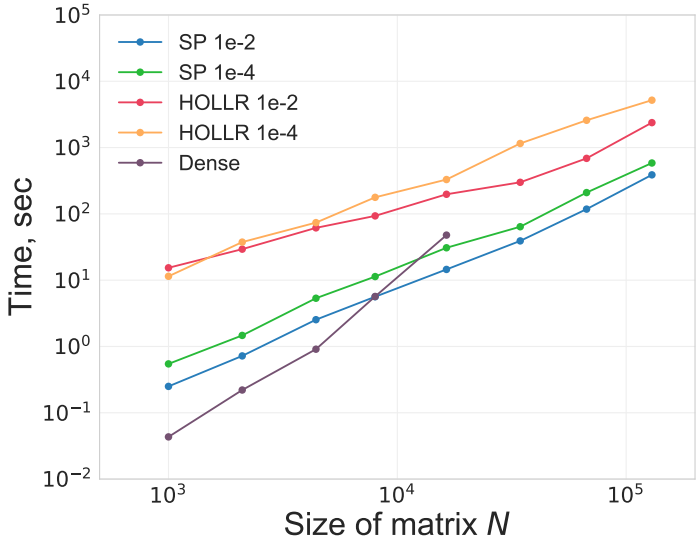
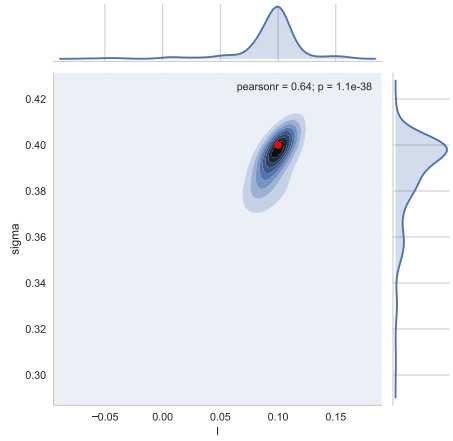
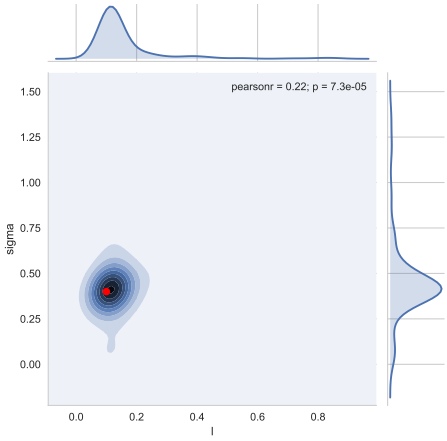


Рисунок 5.13: Время вычисления логарифма детерминнта матрицы K

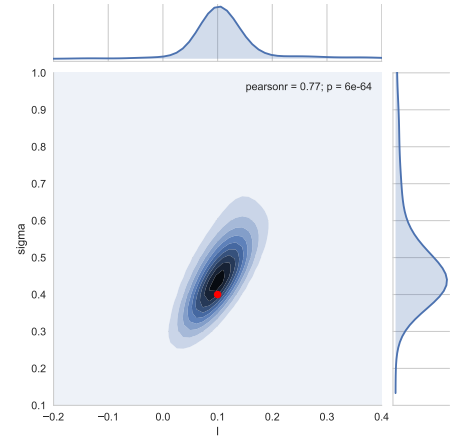
Отметим, что плотный метод требует слишком много памяти и не работает для матриц размаром больше 10^4, однако, плотный метод факторизует матрицу K с высокой точностью. Покажем, что для данной задаче высокая точность факторизации не требуется. Рассмотрим результаты работы метода Монте-Карло Марковских цепей и приведем распределение параметров l и σ^2 для различных методов факторизации матрицы K.



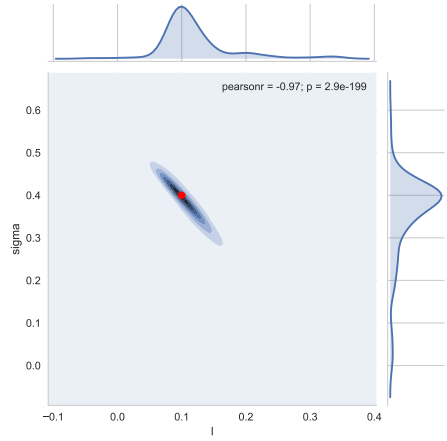
(a) Плотный метод



(b) HODLR, $\varepsilon = 1e-2$



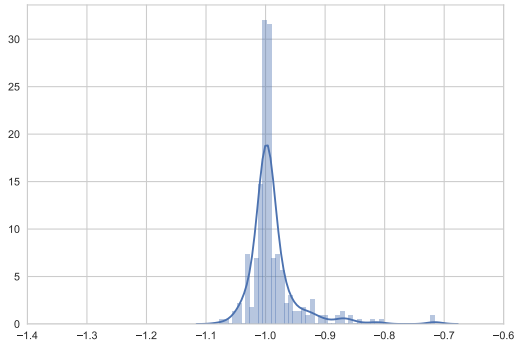
(c) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-2$



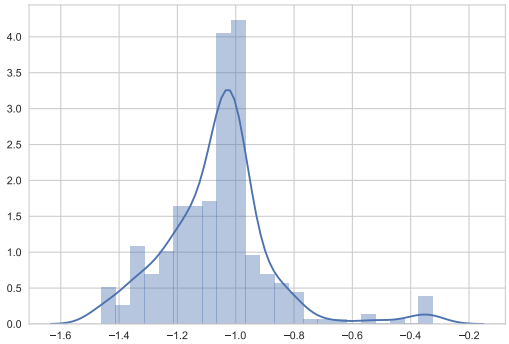
(d) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-4$

Рисунок 5.14: Распределение параметров l и σ^2

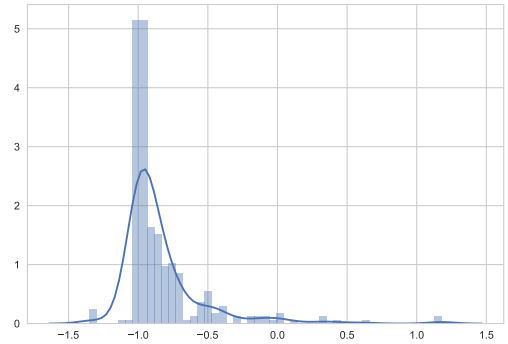
Также приведем распределение параметра α для различных методов факторизации матрицы K.



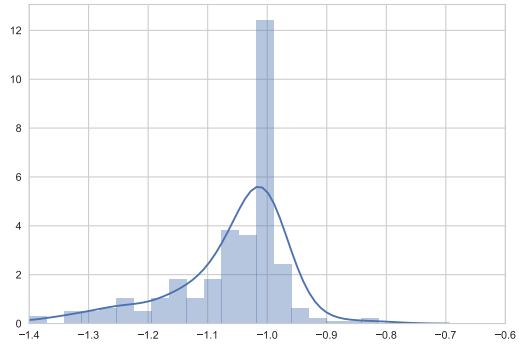
(a) Плотный метод



(b) HODLR, $\varepsilon = 1e-2$



(c) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-2$



(d) Спарсификация $\mathcal{H}^2$, $\varepsilon = 1e-4$

Рисунок 5.15: Распределение параметра α

В Таблице 5.3 приведены коэффициенты детерминации для рассмотренных выше моделей, для различных способов факторизации матрицы K.

	R^2
HODLR, $\varepsilon = 1e-2$	$1 - 2.0 \times 10^{-4}$
HODLR, $\varepsilon = 1e-4$	$1 - 1.2 \times 10^{-4}$
Сп. $\mathcal{H}^2$, $\varepsilon = 1e-2$	$1 - 4.2 \times 10^{-4}$
Сп. $\mathcal{H}^2$, $\varepsilon = 1e-4$	$1 - 4.6 \times 10^{-5}$
Плотн.	$1 - 2.6 \times 10^{-6}$

Таблица 5.3: Коефициент детерминации для модели с коррелированнм шумом и без него.

5.5. Выводы по главе

В данной главе предложено приложение факторизации иерархических матриц к задаче в гауссовских процессов для задачи регрессии. Логорифм определителя матрицы ковариации и решение системы с матрицей ковариации вычислялись при помощи аппроксимации в $\mathcal{H}^2$ формате и приближенной факторизации. Тестирование проводилось для двух- и трехмерной задач, для синтетически сгенерированных данных. Проводилось сравнение с методом HODLR. Предложенный метод значительно ускорил процесс вычисления правдоподобия при моделировании коррелированного шума в задаче регрессии.

Заключение

Диссертация посвящена приближенной факторизации блочно-малоранговых матриц. Основной результат работы состоит в том, что предложены новые приближенные факторизации блочно-малоранговых матриц и методы их построения, реализован программный комплекс, который применен к нескольким задачам математического моделирования. В частности:

1. Предложена факторизация разреженной симметричной положительно определённой матрицы в виде произведения матриц перестановки, блочно-диагональных унитарных и разреженных нижне-треугольных факторов (метод компрессии и исключения, compress and eliminate method, CE), на основе которой предложен прямой метод решения систем и метод построения предобуславливателя.

2. Предложены два типа разреженной факторизации $\mathcal{H}^2$ матриц: «расширенная» и «не-расширенная», которые позволяют ускорить решение линейных систем с $\mathcal{H}^2$ матрицами, в сравнении с итерационным методом BiCGStab [2] и прямыми методами HODLR [5, 7] и IFMM [6, 20].

3. Разработан комплекс программ реализующих представленные алгоритмы. Для факторизации CE проведено тестирование на системах, полученных при конечно-разностной дискретизации стационарного уравнения диффузии, а также системах, полученных при конечно-элементной дискретизации уравнения Пуассона и уравнения упругости. Проведено сравнение реализации метода CE с прямыми методами CHOLMOD [4, 24] и UMFPACK [22], а также итерационным методом BiCGStab с предобуславливателями ILU0 [70], ILUt [69] и ILU2 [46]. Для метода расширенной разреженной факторизации проведено тестирование на задачах электростати-

ки и гидромеханики, в ходе которого метод показал свою эффективность по времени в сравнении с непредобусловленным методом BiCGStab. Для не-расширенной разреженной факторизации проводилось тестирование на задаче моделирования гауссовских процессов для задачи регрессии. Метод показал свою эффективность по времени в сравнении с методами HODLR и IFMM.

Одним из ключевых объектов рассмотрения диссертации является разреженная матрица, полученная при дискретизации дифференциального уравнения. На основании идеи, что обратная к такой матрице является блочно-малоранговой матрицей построен метод компрессии и исключения, на ходу аппроксимирующий заполнение, которое возникает в матрице при исключении.

Также рассмотрен такой тип блочно-малоранговых матриц как $\mathcal{H}^2$ матрицы. Для них предложены два метода приведения к эквивалентным разреженным матрицам. Таким образом, трудная и важная задача приближенной факторизации $\mathcal{H}^2$ матриц сводится к более простой задаче приближенной факторизации разреженной матрицы.

Все предложенные методы приближенной факторизации могут быть рассмотрены в качестве приближенных методов решения и предобуславливателей для блочно-малоранговых матриц.

Для каждого предложенного в диссертации метода представлена его программная реализация. В качестве приложения предложенных методов, проводится моделирование коррелированного шума на основе гауссовсково процесса для задачи регрессии.

Литература

1. http://math.nist.gov/MatrixMarket/data/misc/cylshell/s3dkt3m2.html. — 1997.

2. Accelerating Wilson fermion matrix inversions by means of the stabilized biconjugate gradient algorithm / Andreas Frommer, Volker Hannemann, Bertold Nöckel et al. // Int. J. Mod. Phys. C. — 1994. — Vol. 5, no. 06. — P. 1073–1088.

3. Advanced Numerical Instruments 2D / K Lipnikov, Yu Vassilevski, A Danilov et al. — 1997.

4. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate / Yanqing Chen, Timothy A Davis, William W Hager, Sivasankaran Rajamanickam // ACM T. Math. Software. — 2008. — Vol. 35, no. 3. — P. 22.

5. Ambikasaran Sivaram, Darve Eric. An $\mathcal{O}(N \log N)$ Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices // J. Sci. Comput. — 2013. — Vol. 57, no. 3. — P. 477–501.

6. Ambikasaran Sivaram, Darve E. The Inverse Fast Multipole Method // arXiv preprint arXiv:1309.1773. — 2014.

7. Ambikasaran Sivaramand, Foreman-Mackey Daniel, Greengard Leslie. A fast direct methods for gaussian processes // arXiv preprint arXiv:1403.6015. — 2014.

8. Barnes J. Hut P. A hierarchical $\mathcal{O}(NlogN)$ force-calculation algorithm // Nature. — 1996. — Vol. 324, no. 4.

9. Bebendorf Mario. Why finite element discretizations can be factored by triangular hierarchical matrices // SIAM J. Numer. Anal. — 2007. — Vol. 45, no. 4. — P. 1472–1494.

10. Bebendorf Mario, Hackbusch Wolfgang. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients // Numer. Math. — 2003. — Vol. 95, no. 1. — P. 1–28.

11. Bell W. N., Olson L. N., Schroder J. B. PyAMG: Algebraic Multigrid Solvers in Python v3.0. — 2015. — Release 3.2. URL: https://github.com/pyamg/pyamg.

12. Belyaev M, Burnaev E, Kapushev Y. Computationally efficient algorithm for Gaussian Process regression in case of structured samples // Comp. Math. Math. Phys. — 2016. — Vol. 56, no. 4. — P. 499–513.

13. Börm Steffen. Approximation of solution operators of elliptic partial differential equations by-and-matrices // Numerische Mathematik. — 2010. — Vol. 115, no. 2. — P. 165–193.

14. Börm Steffen. Efficient numerical methods for non-local operators: $\mathcal{H}^2$-matrix compression, algorithms and analysis. — European Mathematical Society, 2010. — Vol. 14.

15. Borm Steffen, Grasedyck Lars, Hackbusch Wolfgang. Introduction to hierarchical matrices with applications // Eng. Anal. Bound Elem. — 2003. — Vol. 27, no. 5. — P. 405–422.

16. Börm Steffen, Grasedyck Lars, Hackbusch Wolfgang. Introduction to hierarchical matrices with applications // Eng. Anal. Bound. Elem. — 2003. — Vol. 27, no. 5. — P. 405–422.

17. Chandrasekaran S, Gu M, Lyons W. A fast adaptive solver for hierarchically semiseparable representations // Calcolo. — 2005. — Vol. 42, no. 3-4. — P. 171–185.

18. Chandrasekaran Shiv, Gu Ming, Pals Timothy. A fast ULV decomposition solver for hierarchically semiseparable representations // SIAM J. Matrix Anal. A. — 2006. — Vol. 28, no. 3. — P. 603–622.

19. Cheng Hongwei, Greengard Leslie, Rokhlin Vladimir. A fast adaptive multipole algorithm in three dimensions // J. Comput. Phys. — 1999. — Vol. 155, no. 2. — P. 468–498.

20. Coulier Pieter, Pouransari Hadi, Darve Eric. The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems // arXiv preprint arXiv:1508.01835. — 2015.

21. Cramer Christopher J, Truhlar Donald G. Implicit solvation models: equilibria, structure, spectra, and dynamics // Chem. Rev. — 1999. — Vol. 99, no. 8. — P. 2161–2200.

22. Davis Timothy A. Algorithm 832: UMFPACK V4. 3-an unsymmetric-pattern multifrontal method // ACM T. Math. Software. — 2004. — Vol. 30, no. 2. — P. 196–199.

23. Davis Timothy A. — https://www.cise.ufl.edu/research/sparse/matrices/Schenk_AFE/af_1_k101.html. — 2014.

24. Davis Timothy A, Hager William W. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves // ACM T. Math. Software. — 2009. — Vol. 35, no. 4. — P. 27.

25. Engineering optimization with the fast boundary element method / Igor Ostanin, Alexander Mikhalev, Denis Zorin, Ivan Oseledets // WIT Trans. Model. Sim. — 2015. — Vol. 61. — P. 175–181.

26. The FEniCS project version 1.5 / Martin Alnæs, Jan Blechta, Johan Hake et al. // Archive of Numerical Software. — 2015. — Vol. 3, no. 100. — P. 9–23.

27. Fast Direct Methods for Gaussian Processes and the Analysis of NASA Kepler Mission Data / S. Ambikasaran, D. Foreman-Mackey, L. Greengard et al. // http://arxiv.org/abs/1403.6015. — 2014. — URL: http://dan.iel.fm/george/current.

28. Fast algorithms for hierarchically semiseparable matrices / Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, Xiaoye S Li // Numer. Linear Algebr. — 2010. — Vol. 17, no. 6. — P. 953–976.

29. A Gaussian process framework for modelling instrumental systematics: application to transmission spectroscopy / NP Gibson, S Aigrain, S Roberts et al. // Mon. Not. R. Astron. Soc. — 2012. — Vol. 419, no. 3. — P. 2683–2694.

30. George Alan. Nested dissection of a regular finite element mesh // SIAM J. Numer. Anal. — 1973. — Vol. 10, no. 2. — P. 345–363.

31. Gilks Walter R, Richardson Sylvia, Spiegelhalter David. Markov chain Monte Carlo in practice. — CRC press, 1995.

32. Goreinov SA. Mosaic-skeleton approximations of matrices generated by asymptotically smooth and oscillatory kernels // Matrix methods and computations. — 1999. — P. 42–76.

33. Goreinov Sergei A, Tyrtyshnikov Eugene E. The maximal-volume concept in approximation by low-rank matrices // Contemp. Math. — 2001. — Vol. 280. — P. 47–52.

34. Goreinov Sergei A, Zamarashkin Nikolai Leonidovich, Tyrtyshnikov Evgenii Evgenevich. Pseudo-skeleton approximations by matrices of maximal volume // Math. Notes+. — 1997. — Vol. 62, no. 4. — P. 515–519.

35. Grasedyck Lars, Hackbusch Wolfgang, Kriemann Ronald. Performance of $\mathcal{H} - LU$ preconditioning for sparse matrices // Comput. Methods Appl. Math. — 2008. — Vol. 8, no. 4. — P. 336–349.

36. Grasedyck Lars, Kriemann Ronald, Le Borne Sabine. Domain decomposition based $mathcalH - LU$ preconditioning // Numer. Math. — 2009. — Vol. 112, no. 4. — P. 565–600.

37. Greengard L., Rokhlin V. A Fast Algorithm for Particle Simulations // J. Comput. Phys. — 1987. — Vol. 73, no. 2. — P. 325–348.

38. Grengard L, Rokhlin V. The rapid evaluation of potential fields in three dimensions. — Springer, 1988.

39. Hackbusch Wolfgang. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part i: Introduction to $\mathcal{H}$-matrices // Computing. — 1999. — Vol. 62, no. 2. — P. 89–108.

40. Hackbusch Wolfgang. Hierarchische Matrizen: Algorithmen und Analysis. — Springer-Verlag Berlin Heidelberg, 2009.

41. Hackbusch W., Borm S. $\mathcal{H}^2$-matrix approximation of integral operators by interpolation // Appl. Numer. Math. — 2002. — Vol. 43. — P. 129–143.

42. Hackbusch W., Khoromskij B.N., Sauter S. On $\mathcal{H}^2$-Matrices // H.-J. Bungartz, et al. (eds.), Lectures on Applied Mathematics. — Springer-Verlag, Berlin Heidelberg, 2000. — P. 9–30.

43. Calculation of non-lifting potential flow about arbitrary three-dimensional bodies : Rep. / DTIC Document ; Executor: John L Hess, AM Smith : 1962.

44. Ho Kenneth, Greengard Leslie. A fast direct solver for structured linear systems by recursive skeletonization // SIAM J. Sci. Comput. — 2012. — Vol. 34, no. 5. — P. A2507–A2532.

45. Intel Math Kernel Library. Reference Manual. — Santa Clara, USA : Intel Corporation, 2009. — ISBN 630813-054US.

46. Kaporin Igor E. High quality preconditioning of a general symmetric positive definite matrix based on its UTU + UTR + RTU-decomposition // Numer. Linear Algebr. — 1998. — Vol. 5, no. 6. — P. 483–509.

47. Karypis George, Kumar Vipin. METIS–unstructured graph partitioning and sparse matrix ordering system, version 2.0. — 1995.

48. Karypis George, Kumar Vipin. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM J. Sci. Comput. — 1998. — Vol. 20, no. 1. — P. 359–392.

49. Kleiber William, Katz Richard W, Rajagopalan Balaji. Daily spatiotemporal precipitation simulation using latent and transformed Gaussian processes // Water Resour. Res. — 2012. — Vol. 48, no. 1.

50. Leithead WE, Zhang Yunong, Leith DJ. Efficient Gaussian process based on BFGS updating and logdet approximation // IFAC P. Vol. — 2005. — Vol. 38, no. 1. — P. 1305–1310.

51. Leonard Anthony. Vortex methods for flow simulation // J. Comput. Phys. — 1980. — Vol. 37, no. 3. — P. 289–335.

52. Log-det approximation based on uniformly distributed seeds and its application to Gaussian process regression / Yunong Zhang, WE Leithead, DJ Leith, L Walshe // J. Comput. Appl. Math. — 2008. — Vol. 220, no. 1. — P. 198–214.

53. Logg Anders, Mardal Kent-Andre, Wells Garth. Automated solution of differential equations by the finite element method: The FEniCS book. — Springer Science & Business Media, 2012. — Vol. 84.

54. Martinsson Per-Gunnar, Rokhlin Vladimir. A fast direct solver for boundary integral equations in two dimensions // J. Comput. Phys. — 2005. — Vol. 205, no. 1. — P. 1–23.

55. Mikhalev A.Yu. — https://bitbucket.org/muxas/h2tools. — 2013.

56. Mikhalev A Yu, Oseledets Ivan V. Iterative representing set selection for nested cross approximation // Numer. Linear Algebr. — 2016. — Vol. 23, no. 2. — P. 230–248.

57. Miller Gary L, Teng S-H, Vavasis Stephen A. A unified geometric approach to graph separators // Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on / IEEE. — 1991. — P. 538–547.

58. NumPy. — 2013–. — URL: http://www.numpy.org/.

59. Numerical solution of diffraction problems using large matrix compression / GV Ryzhakov, A Yu Mikhalev, DA Sushnikova, IV Oseledets // The 9th European Conference on Antennas and Propagation (EuCAP) / IEEE. — 2015. — P. 1–3.

60. Oseledets IV, Mikhalev A Yu. Representation of quasiseparable matrices using excluded sums and equivalent charges // Linear Algebra Appl. — 2012. — Vol. 436, no. 3. — P. 699–708.

61. Ostanin Igor, Zorin Denis, Oseledets Ivan. Toward fast topological-shape optimization with boundary elements // arXiv preprint arXiv:1503.02383. — 2015.

62. Parallel hypergraph partitioning for scientific computing / Karen D Devine, Erik G Boman, Robert T Heaphy et al. // Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International / IEEE. — 2006. — P. 10–pp.

63. Pouransari Hadi, Coulier Pieter, Darve Eric. Fast hierarchical solvers for sparse matrices using extended sparsification and low-rank approximation // SIAM Journal on Scientific Computing. — 2017. — Vol. 39, no. 3. — P. A797–A830.

64. Rajamanickam Sivasankaran, Boman Erik G. Parallel partitioning with Zoltan: Is hypergraph partitioning worth it // Contemp. Math. — 2012. — Vol. 588. — P. 37–52.

65. Rasmussen Carl Edward. Gaussian processes for machine learning. — 2006.

66. “Reverse-Schur” Approach to Optimization with Linear PDE Constraints: Application to Biomolecule Analysis and Design / Jaydeep Bardhan, Michael Altman, Bruce Tidor, Jacob White // J. Chem. Theory Comput. — 2009. — Vol. 5, no. 12. — P. 3260–3278.

67. Richardson Clarence W. Stochastic simulation of daily precipitation, temperature, and solar radiation // Water Resour. Res. — 1981. — Vol. 17, no. 1. — P. 182–190.

68. Rokhlin Vladimir. Rapid solution of integral equations of classical potential theory // J. Comput. Phys. — 1985. — Vol. 60, no. 2. — P. 187–207.

69. Saad Yousef. ILUT: A dual threshold incomplete LU factorization // Numer. Linear Algebr. — 1994. — Vol. 1, no. 4. — P. 387–402.

70. Saad Yousef. Iterative methods for sparse linear systems. — Siam, 2003.

71. Saad Youcef, Schultz Martin. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems // SIAM J. Sci. Stat. Comp. — 1986. — Vol. 7, no. 3. — P. 856–869.

72. Snelson Edward, Ghahramani Zoubin. Sparse Gaussian processes using pseudo-inputs // Adv. Neur. In. — 2006. — Vol. 18. — P. 1257.

73. Solovyev Sergey. Multifrontal Hierarchically Solver for 3D Discretized Elliptic Equations // International Conference on Finite Difference Methods / Springer. — 2014. — P. 371–378.

74. Stavtsev SL. Application of the method of incomplete cross approximation to a non-stationary problem of vortex rings dynamics // Russ. J. Numer. Anal. M. — 2012. — Vol. 27, no. 3. — P. 303–320.

75. Stavtsev SL, RAS INM. Block LU Preconditioner for the Electric Field Integral Equation // Session 2P7 SC1: Novel Mathematical Methods in Electromagnetics. — 2015. — P. 1028.

76. Stavtsev SL, Tyrtyshnikov EE, RAS INM. Application of mosaic-skeleton approximations for solving EFIE // PIERS proceedings. — 2009. — P. 1752–1755.

77. Sushnikova Daria. Compress and Eliminate solver. — https://github.com/dsushnikova/ce_solver. — 2017.

78. Sushnikova Daria A, Oseledets Ivan V. ” Compress and eliminate” solver for symmetric positive definite sparse matrices // arXiv preprint arXiv:1603.09133. — 2016.

79. Trenti Michele, Hut Piet. N-body simulations (gravitational) // Scholarpedia. — 2008. — Vol. 3, no. 5. — P. 3930.

80. Tyrtyshnikov Eugene. Incomplete cross approximation in the mosaic-skeleton method // Computing. — 2000. — Vol. 64, no. 4. — P. 367–380.

81. Tyrtyshnikov E. E. Mosaic-skeleton approximations // Calcolo. — 1996. — Vol. 33, no. 1. — P. 47–57.

82. Yang Kai, Pouransari Hadi, Darve Eric. Sparse Hierarchical Solvers with Guaranteed Convergence // arXiv preprint arXiv:1611.03189. — 2016.

83. Zienkiewicz Olgierd Cecil, Taylor Robert Leroy, Taylor Robert Lee. The finite element method. — McGraw-hill London, 1977. — Vol. 3.

84. A distributed-memory package for dense Hierarchically Semi-Separable matrix computations using randomization / Francois-Henry Rouet, Xiaoye S Li, Pieter Ghysels, Artem Napov // arXiv preprint arXiv:1503.05464. — 2015.

85. A fast solver for HSS representations via sparse matrices / Shiv Chandrasekaran, Patrick Dewilde, Ming Gu et al. // SIAM J. Matrix Anal. A. — 2006. — Vol. 29, no. 1. — P. 67–81.

86. Михалев Александр. Метод построения блочно-малоранговой аппроксимации матрицы по её элементам : Ph. D. thesis / Александр Михалев ; Московский государственный университет имени М.В.Ломоносова. — 2014.

87. О численном решении двумерного гиперсингулярного интегрального уравнения и о распространении звука в городской застройке / В А Гутников, В Ю Кирякин, Иван Кузьмич Лифанов et al. // Журнал вычислительной математики и математической физики. — 2007. — Vol. 47, no. 12. — P. 2088–2100.

88. Применение мультизарядового приближения больших плотных матриц в рамках модели поляризуемого континуума для растворителя / Александр Юрьевич Михалев, Игорь Владимирович Офёркин, Иван Валерьевич Оседец et al. // Вычислительные методы и программирование. — 2014. — Vol. 15, no. 1. — P. 9–21.

89. Ставцев Станислав Леонидович. Применение метода неполной крестовой аппроксимации к решению задач аэродинамики методом дискретных вихрей // Научный вестник Московского государственного технического университета гражданской авиации. — 2013. — no. 2 (188).