Российская академия наук
Институт вычислительной математики

На правах рукописи

Савостьянов Дмитрий Валериевич

УДК 519.6

Быстрая полилинейная аппроксимация матриц и интегральные уравнения

01.01.07 — Вычислительная математика

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
чл.-корр. РАН, проф. Тыртышников Е. Е.

Москва 2006

Содержание

Введение **2**
i.1 Быстрые методы решения интегральных уравнений . . . 3
i.2 Историческая справка 7
i.3 Основной результат данной работы 11
i.4 Содержание работы по главам 16

Глава 1. Мозаично-скелетонный метод **20**
1.1 Описание и развитие мозаично-скелетонного метода . . . 20
1.1.1 Описание метода. 20
1.1.2 Методы дожимания (переаппроксимации). 24
1.1.3 Численные эксперименты. 25
1.2 Параллельная версия метода 26
1.2.1 Особенности кластерных станций. 27
1.2.2 Модель параллелизации и распределение нагрузки. 28
1.3 Выводы 29

Глава 2. Метод трехмерной крестовой аппроксимации **31**
2.1 Введение 31
2.2 Теорема существования 35
2.3 Трехмерный крестовый метод 38
2.3.1 Как нельзя построить этот алгоритм. 38
2.3.2 Как можно построить этот алгоритм. 40
2.3.3 Как добиться почти линейной сложности. 40
2.3.4 Как сделать алгоритм эффективным. 45
2.3.5 Сложность полученного метода. 51
2.4 Важные детали 54
2.4.1 Как найти подматрицу максимального объема. . . 54
2.4.2 Как найти наибольший элемент в срезке. 57
2.4.3 Как добавить векторы в ортогональный набор. . . 60
2.4.4 Как проверять точность аппроксимации. 63
2.4.5 Какова точность «нулевого приближения» 64
2.5 Модельные численные эксперименты 66
2.6 Асимптотика предложенного метода 67
2.6.1 Теоретические оценки. 67
2.6.2 Практические значения ранга. 72
2.6.3 Время работы алгоритма. 73
2.7 Выводы 83

Глава 3. Приложения к численному решению уравнений **84**
3.1 Применение мозаично-скелетонного метода к задаче Дирихле для уравнения Гельмгольца 84
3.1.1 Некоторые факты из теории потенциала. 84
3.1.2 Интегральное уравнение и дискретизация. . . . 86
3.1.3 Численные эксперименты. 89
3.2 Применение мозаично-скелетонного метода к задаче гидроакустики 92
3.2.1 Постановка задачи. 92
3.2.2 Метод замкнутых дискретных вихревых рамок. . 95
3.2.3 Вычисление элементов матриц. 96
3.2.4 Численные эксперименты. 97
3.3 Применение тензорных аппроксимаций к решению простейшего интегрального уравнения 98
3.3.1 Постановка задачи. 98
3.3.2 Дискретизация задачи. 98
3.3.3 Сжатие матрицы. 99
3.3.4 Структурированные векторы. 100
3.3.5 Предобусловливание тензорных матриц. 101
3.3.6 Численные результаты. 105
3.4 Выводы 106

Глава 4. Специфика матриц в одной задаче электродинамики **108**
4.1 Постановка задачи 108
4.1.1 Физическая постановка. 108
4.1.2 Интегральное уравнение. 110
4.2 Метод дискретизации 112
4.3 Специфика полученной матрицы 113
4.4 Параллельный алгоритм 115
4.5 Численные эксперименты 119
4.6 Пример решения обратной задачи 128
4.6.1 Приближение Борна. 128
4.6.2 Горизонтальное зондирование. 130
4.6.3 Двумерное зондирование. 131
4.7 Применение трилинейной крестовой аппроксимации для сжатия данных 131
4.8 Выводы 134

Заключение **135**

Литература **137**

Введение

i.1. Быстрые методы решения интегральных уравнений

В последние годы при решении сложных инженерных задач (геологоразведки, акустики моря, ЯМР в квантовой химии) все чаще используются интегральные уравнения. Интерес к ним обусловлен несколькими причинами.

- Интегральные уравнения позволяют понизить размерность задачи, сводя, например, трехмерную задачу к интегральным уравнениям записанным на границе области — двумерном многообразии.

- Интегральные уравнения позволяют сводить краевые задачи в бесконечных областях к задачам на ограниченном носителе, как например метод объемного интегрального уравнения для задачи рассеяния.

Серьезным фактором, затрудняющим применение интегральных уравнений в практических расчетах, являются затраты на вычисление, хранение и умножение на матрицу возникающей линейной системы. В подавляющем большинстве случаев она плотная, а значит при числе неизвестных n в общем случае на ее хранение требуется n^2 ячеек памяти. Долгое время именно за счет этого интегральные уравнения проигрывали в глазах инженеров-вычислителей подходам, основанным на дифференциальных уравнениях. Последние при дискретизации приводят к *разреженным* матрицам, на хранение которых требуется лишь $\mathcal{O}(n)$ элементов памяти. Однако возникают другие сложности: например, прямое решение такой системы приводит к заполнению, а итерационное требует построения предобусловливателя, так как возникающая матрица обычно очень плохо обусловлена.

Если мы хотим использовать методы интегральных уравнений так же эффективно, как и дифференциальные постановки, требуется предложить быстрые алгоритмы решения возникающих линейных систем. Называя метод «быстрым», обычно имеют в виду, что его сложность составляет $o(n^2)$, где n — размерность линейной системы; в последнее время этот термин все чаще относят к алгоритмам *почти линейной* по n сложности, то есть сложности $\mathcal{O}(n \log^a n)$, с некоторым $a > 0$. Для *приближенных* методов чаще всего отдельно выделяют зависимость сложности от параметра точности ε, обычно тоже логарифмическую, указывая $\mathcal{O}(n \log^a n \log^b \varepsilon^{-1})$. При этом для

сохранения аппроксимации параметр точности ε естественно выбрать порядка точности дискретизации. При этом он оказывается связан с n степенной зависимостью (например $\varepsilon \sim n^{-1}$ или $\varepsilon \sim n^{-2}$), что не нарушает почти линейную оценку сложности.

Основным результатом нашей работы является развитие метода, сложность которого асимптотически *меньше*, чем размерность линейной системы, т.е. $o(n)$. Аналогичные алгоритмы с такой асимптотикой сложности нам не известны.

Естественно, возможность построения быстрого метода основана в первую очередь на обнаружении и использовании некоторой особенности возникающих плотных матриц. В основном обсуждается два вида специфики.

Первый вид связан с наличием у ядра трансляционной симметрии. При использовании равномерных сеток это приводит к возникновению в матрице теплицевых и ганкелевых структур. При этом затраты памяти снижаются до $\mathcal{O}(n)$, а сложность получаемых методов составляет $\mathcal{O}(n \log n)$. В одномерном случае для решения возникающей системы можно предложить *прямые* методы, однако их обобщение на случай большей размерности невозможно. Это, кстати, является частой проблемой предлагаемых «быстрых» методов: они эффективны лишь для одномерных задач. Для теплицевых структур обобщение на многомерный случай, впрочем, возможно — при использовании итерационных методов решения они приводят к значительному ускорению вычислений; однако дополнительно встает вопрос об эффективном предобусловливании. Однако в ряде случаев использование равномерных декартовых сеток представляется невозможным (например в областях сложной формы) или невыгодным (когда выгодно сгустить сетку к некоторой особенности). Подобный пример для решения гиперсингулярного интегрального уравнения Прандтля в квадрате предложен в работе [62].

Второй вид специфики связан с выделением в большой плотной матрице *блоков,* для которых может быть построено малоранговое приближение. Таким образом, речь идет о приближении матрицы A другой матрицей $\tilde{A}$, такой, что:

- матрично-векторное умножение $y = \tilde{A}x$ выполняется быстро;
- объем памяти, необходимый для хранения $\tilde{A}$, мал;
- погрешность в решении, допускаемая при замене A на $\tilde{A}$, сравнима с погрешностью дискретизации.

Под «быстро» и «мал» мы имеем в виду $\mathcal{O}(n \log^a n \log^b \varepsilon^{-1})$ арифметических операций и ячеек памяти. В качестве критерия допустимой погрешности мы понимаем условие $\|A - \tilde{A}\|_F \leqslant \varepsilon \|A\|_F$. При таком подходе для решения линейной системы должен использоваться какой-либо итерационный метод (возможно, с предобусловливанием), основной операцией которого является умножение на матрицу $\tilde{A}$. Эта операция совершается быстро за счет выявления той или иной неявной структуры в матрице, позволяющей приближенно описывать ее почти линейным числом параметров.

Именно так с матричной точки зрения можно трактовать следующие известные методы построения быстрых алгоритмов:

- мультипольные разложения [30, 31, 32, 1];
- кластеризация граничных элементов [15];
- интерполяция на регулярную (иерархическую) сетку [60, 4];
- локальные волны (wavelets) [3, 29].

Исторически, однако, эти методы развивались вовсе не на основе наблюдения за неявной структурой возникающей матрицы. Первые три из них являются вообще говоря *безматричными,* или *операторными,* то есть исходная матрица в явном виде не участвует в операции умножения. Операторные методы (обзор см., например, в [66]), как правило формулируются на аналитическом языке для конкретного ядра интегрального оператора ($\log|x-y|$, $1/|x-y|$, $e^{i\kappa|x-y|}/|x-y|$). При этом действие интегрального оператора рассматривается в терминах взаимодействия *источников* и *приемников*, причем при описании такого взаимодействия для каждого источника (или группы источников) определяются *зоны близкодействия,* куда попадает небольшое число приемников и *дальнодействия,* куда попадают все остальные элементы. Поскольку в зоне дальнодействия потенциал взаимодействия уже не содержит сингулярности, интегральный оператор может быть приближен суммой нескольких операторов с разделенными переменными на основе какого-либо разложения ядра. Погрешность разложения при этом определяет точность аппроксимации. С матричной точки зрения это означает, что элементы матрицы разделяются на блоки, и для блоков, отвечающих геометрически хорошо отделенным друг от друга группам элементов, строится малоранговое приближение.

Поскольку операторные методы построения аппроксимаций основаны на специальном представлении исходного оператора, для их

практического применения в инженерных расчетах требуется переработка всех этапов алгоритма, начиная от процедуры дискретизации и заканчивая обработкой результата решения линейной системы. Вычислительная процедура может быть очень сложной и многоэтапной, и при практической реализации эффективного метода инженер заинтересован подвергать модификации как можно меньшее количество разработанных блоков программы. Именно этим особенно привлекательны методы, использующие непосредственно элементы блоков аппроксимируемой матрицы, например, *мозаично-скелетонный метод* [37, 51, 38, 11, 39] или же метод иерархических матриц [16]. Для их применения надо изменить лишь процедуру построения матрицы системы и операцию матрично-векторного умножения. Подчеркнем, что способ дискретизации системы и процедура вычисления элементов матрицы остаются прежними, меняется лишь техника вычисления блока матрицы — если в исходном алгоритме он строился полностью и хранится в плотном виде, то в быстром методе для него ищется малоранговое приближение. При определенных предположениях относительно свойств гладкости ядра интегрального оператора (точный вид которого знать не требуется) для построения такого приближения достаточно вычислить лишь небольшое число элементов блока. Это означает, что матрица присутствует при работе алгоритма лишь *виртуально*, в виде процедуры вычисления отдельных элементов. Полностью она не вычисляется и не хранится, что позволяет снизить сложность метода до почти линейной не только на этапе решения, но и на этапе генерации матрицы.

Основной результат нашей работы основан на еще одном, не обсужденном выше, виде специфики плотных матриц. Речь идет о возможности представить многоуровневую матрицу, порожденную интегральным уравнением, в виде суммы прямых или *тензорных* произведений [67, 40, 17]. На операторном уровне это отвечает геометрическому расщеплению ядра *по координатам.* При этом для трехуровневых матриц при определенных предположениях касательно ядра (которым удовлетворяют все основные типы интегральных уравнений) достигается сверх-линейное сжатие данных, а возникающие методы решения имеют сложность, *асимптотически меньшую* числа неизвестных. Предлагаемый в нашей работе алгоритм вычисления тензорных аппроксимаций может считаться развитием (нетривиальным) метода крестовой аппроксимации и следует основным его идеям: в частности, рассматривает матрицу системы как виртуальный объект, заданный процедурой вычисления его элементов, и строит аппроксимацию на основе лишь небольшого числа элементов матрицы. За счет этого

сверх-линейной скоростью обладают как этап решения системы, так и этап генерации аппроксиманта.

Мы считаем, что до презентации основного результата и детального его обсуждения, стоит рассказать об истории развития алгоритмов аппроксимации матриц, этапом которой, мы надеемся, является и данная работа.

i.2. Историческая справка

Еще в 1969 году Н. С. Бахваловым была сформулирована следующая гипотеза [47], относящаяся к корректно поставленным краевым задачам математической физики:

Пусть известно, что правая часть $f(x)$ принадлежит некоторому компактному множеству, а n_ε — минимальное число вычислений правой части $f(x_i)$, достаточное для того, чтобы при любой правой части $f(x)$ по этим значениям можно было восстановить с точностью ε решение $u(x)$. Тогда для решения корректной задачи математической физики достаточно использовать $\mathcal{O}(n_\varepsilon)$ значений f_i и дополнительно произвести $\mathcal{O}(n_\varepsilon \log^a n_\varepsilon \log^b \varepsilon^{-1})$ математических действий.

Возможно, первый широко известный алгоритм с такой асимптотикой сложности был предложен в 1985 году В. Рохлиным [30] и получил название *мультипольного метода*. Весьма близок ему появившийся в 1989 году метод кластеризации граничных элементов (*panel clustering*), предложенный В. Хакбушем и З. П. Новаком [15]. В этих работах элементы исходной матрицы разделялись на две группы, отвечающие *близкодействию* и *дальнодействию*. Выбор этих зон определяется видом оператора исходной задачи и особенностями метода. Поскольку сингулярность содержится только в зоне близкодействия, интегральный оператор в дальней зоне является достаточно гладким и может быть приближен оператором с разделенными переменными с помощью разложения в ряд Тейлора (как в методе кластеризации) или разложения по сферическим функциям (как в мультипольном методе). Остаточный член разложения определяет погрешность приближения, а количество членов в разложении определяет ранг аппроксимации.

Приведенные методы формулировались на геометрическом и операторном уровне, в терминах взаимодействия источников и приемников. Возможность получить малопараметрическое описание линейной системы (и тем самым построить быстрый метод ее решения) описывалась через объекты, предшествующие самой системе: оператор взаимодействия исходной задачи и геометрию расчетной области. Однако

с позиций инженерной практики куда более привлекательной выглядит возможность работать на матричном уровне, то есть строить малопараметрическую аппроксимацию на основе элементов имеющейся матрицы, а не исходного оператора. В 1992 году Е. Е. Тыртышников предложил матричную трактовку существующих методов. Понятие зон близко– и дальнодействия оператора выразилось на матричном языке в виде техники иерархического разбиения матрицы на блоки. При этом блоки, отвечающие хорошо разделенным узлам сетки (зоне дальнодействия), допускали малоранговое приближение. Чтобы еще больше сблизить операторное и матричное описание, аппроксимация интегрального оператора в области дальнодействия производилась на основе интерполяционного многочлена. Если для дискретизации интегрального уравнения применяется простейший метод коллокации, то значения интегрального оператора в точках сетки совпадают с элементами матрицы, а значит их интерполяция эквивалентна аппроксимации блока матрицы. Эти результаты, полученные совместно с М. В. Мягчиловым, были опубликованы в 1992 году в работе [27]. На почве этих исследований в это же время происходило сотрудничество группы Е. Е. Тыртышникова с компанией Elegant Mathematics, результаты которого были отражены в работе [37] (1993 год). В частности, было предложено аппроксимировать блоки с помощью алгоритма типа Ланцоша (это быстрее, чем применяемое традиционно сингулярное разложение) и закреплена идея иерархической структуры блоков, идентичная концепции $\mathcal{H}$-матриц, предложенной В. Хакбушем в 1999 году в работе [16].

Для предложенного метода быстро нашлось практическое применение. Уже в 1993 году он использовался при решении серьезных промышленных задач, например для исследования рассеяния электромагнитной волны на гладкой поверхности (для фирмы Cray Research). С тех пор моделирование электромагнитных процессов было постоянной (но совсем не единственной) областью приложения быстрых алгоритмов, развиваемых в группе Е. Е. Тыртышникова. В 1995 году в работах [20, 21] рассматривались трехмерные интегральные уравнения, возникающие в задачах распространения электромагнитной волны. В связи с плотной структурой матрицы в таких задачах, ее вычисление и хранение в полном виде не представлялось возможным. Тогда для снижения арифметических затрат была использована имеющаяся блочно-теплицевая структура матрицы — это оказалось выгоднее, чем строить малопараметрические аппроксимации. Стоит отметить, что в данной работе мы предлагаем алгоритм, превосходящий по эффективности метод использования теплицевых структур. Еще более важным

является то, что наш метод может быть *скомбинирован* с имеющейся блочно-теплицевой (или любой другой свойственной матрице) структурой, что приведет к дополнительному сжатию данных.

Начиная с 1993 года одновременно с разработкой приложений происходило и развитие основного алгоритма. В 1995 году в Докладах РАН была опубликована работа [51], в которой показывалось возможность построения малоранговой аппроксимации для блоков матрицы лишь по малой части ее элементов — нескольким столбцам и строкам. Было доказано, что такие «опорные» столбцы и строки существуют, но оставался открытым вопрос о конструктивном критерии их выбора. В том же году для дальнейшего развития метода в фонд Volkswagen-Stiftung был подан совместный российско-германский проект, руководителями которого стали Е. Е. Тыртышников (Россия) и С. Рязанов (ФРГ). В состав проекта с российской стороны входили также С. А. Горейнов, Н. Л. Замарашкин, И. В. Ибрагимов и М. С. Мартынов, с немецкой стороны М. Бебендорф. Идеи представления матрицы в виде блоков, допускающих малоранговую аппроксимацию, представлялись на различных конференциях по вычислительной математике и приложениям: Болгария, 1995; Enumath, Париж, 1995; Руссэ, 1996; Картона, 1996. Результатом последних выступлений стала публикация в итальянском журнале «Calcolo» [38], в это же время вышли статьи о псевдоскелетном методе в журнале Linear Algebra and Applications [11, 12]. Все это способствовало популяризации метода.

В 1998 году состоялся заключительный рабочий семинар проекта Volkswagen-Stiftung, на котором Е. Е. Тыртышников впервые продемонстрировал численные результаты аппроксимации матриц на основе неполной информации о ее элементах. Алгоритм аппроксимации стал полностью автоматическим, в нем столбцы и строки аппроксимируемого блока вычислялись адаптивно, при этом на каждом шаге вычислялись те строки (столбцы), позиции которых совпадали с расположением подматрицы *наибольшего объема* (модуля детерминанта) в уже вычисленных столбцах (строках). Эта техника, названная неполной крестовой аппроксимацией, была затем опубликована в работе [39].

По всей видимости, этот результат стал наиболее полезным для дальнейшего развития алгоритма. Именно тогда стало понятно, что аппроксимация блока может быть вычислена без использования всех его элементов. Техника поиска при этом могла иметь различные варианты. Например, в 1999 году М. Бебендорф предложил в своей диссертации другой алгоритм поиска опорного креста, позже включенный в статью [2]. Алгоритм Бебендорфа можно охарактеризовать

как «жадный» до вычислений матричных элементов — на каждом шаге к имеющемуся кресту добавлялись только одна строка и столбец, причем решение о том, какой крест должен быть вычислен на следующем шаге, принималось только по уже имеющейся информации. Конкретнее, если на p-том шаге метода аппроксимация блока $A \approx \tilde{A}_{p-1} = \sum_{\alpha=1}^{p-1} u_\alpha v_\alpha^\top$ была уточнена за счет вычисления столбца u_p и строки v_p, то следующий столбец выбирается среди еще не вычисленных на той позиции, где погрешность приближения строки v_p аппроксимантом $\tilde{A}_{p-1}$ оказалась максимальной. Аналогичный выбор происходит и для строки. Точность аппроксимации оценивалась методами теории интерполяции, основанными на результатах работы [27]. При этом в оценках точности сохранялась не до конца исследованная зависимость от выбранной сетки.

Позже Е. Е. Тыртышниковым было замечено, что фактически предложенный М. Бебендорфом метод есть ни что иное, как метод разложения Гаусса с выбором ведущего элемента по столбцу и с определенным образом зафиксированной техникой выбора столбцов в активной подматрице. В алгоритмах определения ранга метода Гаусса применялся уже давно. Если ранг матрицы равен r, то после r шагов метода Гаусса ведущая подматрица оказывается в точности нулевой, что позволяет определять ее ранг. Если же матрица A приближена матрицей ранга r с некоторой точностью ε, число шагов метода Гаусса зависит от техники выбора ведущих элементов, выбора первого ведущего столбца и в значительной степени определяется свойствами самой матрицы. Таким образом, по существу в работе [2] для задачи определения ранга вычисляемых блоков предложен метод Гаусса с определенной стратегией выбора ведущих элементов и эмпирически показана его применимость для некоторых типовых матриц, возникающих при решении интегральных уравнений.

Следует отметить, что выбор ведущего элемента по столбцу является стандартным способом избежать роста погрешности в методе Гаусса, но выбор новых столбцов в ведущей подматрице, вообще говоря, может быть осуществлен произвольно, лишь бы получившийся набор сохранял линейную независимость. Трактовка метода крестовой аппроксимации как варианта метода Гаусса позволила окончательно прояснить его связь с концепцией поиска подматриц наибольшего объема. В 2000 году в работах [39, 13] было доказано, что среди всех возможных аппроксимаций блока, построенных по его строкам и столбцам, наилучшей является та, строки и столбцы в которой образуют на пересечении подматрицу максимального объема. Быстрого метода поиска этой подматрицы не существует, однако можно искать близкие к

ней подматрицы на основе тех или иных адаптивных стратегий. Таким образом было определено направление дальнейшей работы — эвристические техники поиска подматрицы большого объема. При этом возникала определенная вариативность метода поиска такой подматрицы, что позволяло достаточно несложным образом улучшать его вычислительные свойства. Известны примеры, в которых метод М. Бебендорфа «не сходится» , то есть неоправданно увеличивает размер вычисляемой аппроксимации без заметного улучшения ее точности. Этот недостаток устраняется, если при выборе ведущих столбцов несколько расширить список элементов, среди которых выбирается элемент с наибольшей погрешностью аппроксимации. Например, в работе [9] в этот список кроме элементов текущей строки добавлена еще диагональ аппроксимируемой матрицы, а в работе [69] рассматривается алгоритм, содержащий дополнительную проверку точности аппроксимации на некотором случайном множестве элементов. На численных примерах была показана высокая надежность и эффективность полученного метода. В работе [69] также была предложена версия скелетного метода для параллельных кластерных платформ. Эти результаты частично включены в данную диссертацию.

Связь «принципиальных» строк и столбцов матрицы с подматрицей максимального объема активно используется и при построении трехмерного аналога метода крестовой аппроксимации для тензоров.

Необходимо также отметить, что одновременно с развитием алгоритмической части метода постоянно продолжалось расширение класса задач, для которых теоретически обосновывалось существование малоранговых аппроксимаций. Для асимптотически гладких и осцилляционных ядер оценки были даны С. Горейновым в [52]. Они же позже вошли в его диссертационную работу [53]. Таким образом, в 2001 году метод был в высокой степени теоретически, алгоритмически и технологически развит.

i.3. Основной результат данной работы

Основные результаты предлагаемой работы относятся к быстрому построению малоранговой аппроксимации для трехмерных массивов. Задача формулируется так: для заданного массива $\mathcal{A} = [a_{ijk}]$ найти с заданной точностью представление в виде

$$a_{ijk} \approx \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}$$

с возможно меньшим значением ранга r. В применении к решению линейных систем, возникающих при дискретизации интегральных уравнений, эта задача равносильна поиску аппроксимации матрицы системы суммой прямых (кронекеровых, тензорных) произведений. Как способ представления данных тензорные аппроксимации дают *сверхлинейное* сжатие, что делает их крайне привлекательными при решении интегральных уравнений, когда размер возникающих массивов очень велик. Именно эта область применения алгоритма обсуждается в данной работе. Однако сама по себе трилинейная аппроксимация имеет массу других приложений:

- при обработке данных эксперимента в физике,
- в задаче о ядерном магнитном резонансе в химии,
- при обработке сигнала в инженерной практике,
- при слепом разделении сигналов в MIMO (multiple-input multiple-output) системах передачи информации,
- при томографии мозга и внутренних органов в медицине,
- при обработке статистического материала в факторном анализе, социологии и экономике,
- в мультимедийных приложениях для обработки больших массивов графической и видео-информации,
- в задачах идентификации голоса или образа,
- в теории сложности при поиске оптимального алгоритма.

Читателю, интересующимися приложениями, мы рекомендуем замечательный обзор [7].

Вообще говоря, задача аппроксимации трехмерных массивов есть частный случай задачи приближенного построения канонического (или мультилинейного) разложения для тензора произвольной размерности. Однако, сужение рассматриваемой проблемы на трехмерный случай вполне оправдано, поскольку

- именно разложение трехмерных массивов является ключевым этапом практически во всех указанных приложениях,

- алгоритм, предлагаемый для трехмерного случая, тривиально обобщается для случая большей размерности. При этом следует отметить, что сам по себе он не является тривиальным обобщением двумерного алгоритма.

Первая работа по разложению многомерных массивов принадлежит Л. Р. Таккеру, в 1966 году предложившему использовать трилинейное разложение в факторном анализе [36]. Оказалось, что разложение многомерных массивов существенным образом отличается от малорангового разложения матриц. Например, ранг трилинейного разложения (*тензорный ранг*) может превосходить размеры массива, причем до сих пор не получены точные верхние оценки на его значение. Тензорный ранг зависит от числового поля, в котором строится аппроксимация. Если для матриц $n \times n$ множество матриц ранга меньше n имеет меру ноль, то для тензоров это не так: среди тензоров $2 \times 2 \times 2$ ранг 2 имеет 79% тензоров, а ранг 3 — 21%. Эти экспериментальные результаты были получены в 1977 Крускалом. Кроме указанных результатов, в работе [23] он сформулировал и условия единственности такого разложения.

Первоначально для конструктивного построения канонического разложения использовался только метод переменных направлений [18, 5]. Этот метод весьма просто запрограммировать, стоимость одной итерации крайне невысока, и невязка при итерациях не возрастает, однако он может сойтись в некоторый локальный минимум. Достоинства и недостатки этого метода в сравнении с другими минимизационными методами поиска трилинейного разложения обсуждаются нами в работе [81]. Скачок в развитии этой задачи произошел в 1997 году, когда Ливен де Латауэр предложил в работе [24] использовать для задач обработки сигналов технику многомерного SVD-разложения. Это фактически разделило рассматриваемую проблему на две независимые части:

- поиск промежуточного разложения, называемого разложением Таккера

$$a_{ijk} \approx \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'}, \qquad (*)$$

  который осуществляется на основе сингулярного разложения, и

- поиск трилинейного разложения для ядра $\mathcal{G} = [g_{ijk}]$.

Поскольку размеры ядра $\mathcal{G}$ в практических приложениях оказываются меньше размеров исходного массива (обычно они совпадают с

его тензорным рангом), трилинейное разложение для $\mathcal{G}$ строится существенно быстрее, чем для $\mathcal{A}$. Кроме того, при $r_1, r_2, r_3 \ll n$ уже разложение Таккера дает значительное сжатие данных.

Впервые задача обобщенного сингулярного разложения серии матриц — формальный аналог задачи о трилинейном разложении трехмерного массива — рассматривалась группе Е. Е. Тыртышникова в 1999 году И. В. Ибрагимовым [56], в приложении к проблеме люминесцентной спектроскопии. При этом сразу же возникло желание построить быстрый алгоритм для разложения тензоров на основе имеющегося опыта работы с матрицами. Однако тривиальные обобщения матричных подходов не приводили к успеху. Стандартные методы, основанные на SVD высокого порядка (HOSVD) [25, 26], обладают слишком большой асимптотикой сложности, и применимы поэтому только при небольших размерах тензоров, что характерно для задач обработки сигналов или статистических приложений. Но при решении интегральных уравнений возникающие тензоры очень большие. Например для уравнения на сетке $n \times n \times n$ матрица имеет n^6 ненулевых элементов, и сложность HOSVD порядка $\mathcal{O}(n^8)$. Даже полное вычисление такого массива и хранение его в оперативной памяти при сколь-либо существенных размерах сетки невозможно — оперативной памяти размером $2\mathrm{GB}$ оказывается достаточно лишь для матрицы, отвечающей неравномерной сетке $25 \times 25 \times 25$. При использовании равномерной сетки этот порог поднимается до 645, но время вычисления все равно слишком велико.

Первый известный нам быстрый алгоритм трилинейного разложения был предложен в 2002 году И. В. Ибрагимовым, работавшим тогда в университете земли Саар (Германия). В его работе [19] приведен метод, который строит для матрицы, порожденной интегральным уравнением на сетке $n \times n \times n$, тензорную аппроксимацию за $\mathcal{O}(n^4)$ действий. Матрично-векторное умножение выполняется с той же сложностью. Алгоритм использует информацию лишь о *малой части* элементов матрицы, однако в нем необходимо указать некоторый набор «важных элементов» матрицы, на множестве которых происходит оптимизация. Автоматической процедуры для их определения не предлагается, что является, на наш взгляд, недостатком метода.

Другой подход к построению быстрых алгоритмов для такого рода плотных систем в это же время был предложен в работе [9]: он состоит в комбинировании тензорного разложения с эффективным представлением факторов на основе вейвлет-спарсификации. Одновременно с этим Е. Е. Тыртышников изучал вопрос *существования* тензорной аппроксимации и показал [67, 40], что для матриц, порож-

денных асимптотически гладкими функциями (а в этот класс входят дискретные аналоги всех известных интегральных уравнений) существует тензорная аппроксимация, на ранг которой можно дать весьма оптимистичные верхние оценки: $r \leqslant c \log^a n$.

После того, как существование аппроксимации доказано, снова встает вопрос о максимально эффективном его вычислении. По аналогии с методом крестовой аппроксимации, хотелось бы отыскать способ нахождения аппроксиманта по малой части элементов исходного массива. Здесь принципиальны два вопроса.

- Существует ли разложение $(*)$, в котором матрицы U, V и W состояли бы из элементов исходного массива?
- Если да, то есть ли способ найти эти матрицы «быстро» ?

На оба эти вопроса в нашей работе дан положительный ответ, что составляет ее основной результат. Мы показали, что если массив $\mathcal{A}$ может быть приближен разложением Таккера с погрешностью порядка ε, то существует другое разложение Таккера, в котором факторы U, V, W состоят из столбцов, строк и «распорок» массива $\mathcal{A}$, и оно приближает массив $\mathcal{A}$ с точностью $c\varepsilon$, где коэффициент c зависит только от модовых рангов r_1, r_2, r_3. Для эффективного вычисления такого разложения мы предлагаем версию крестового алгоритма, обобщенную на многомерный случай. Это обобщение не тривиально. Оно базируется на рекуррентном применения крестового метода: сначала для разложения матрицы $A = [a_{(ij)k}]$, а затем для вычисления всех требуемых срезок $A_k = [a^k_{ij}]$. Однако для того, чтобы сделать метод по-настоящему эффективным, необходимо было предложить новый, отличный от матричного случая, формат представления данных, а также решить целый ряд задач, касающихся эффективной работы с матрицами в малоранговом представлении. Алгоритм действует адаптивно, последовательно вычисляя столбцы, строки и распорки массива $\mathcal{A}$ и за счет этого наращивая размер опорных базисов U, V, W. Принципиальную роль при выборе новых вычисляемых элементов играют позиции подматриц максимального объема в текущих факторах U, V, W. Если алгоритм применяется к массиву, у которого существует представление тензорного ранга r с точностью ε, то после r шагов вносимая коррекция будет иметь порядок ε. На этом факте основан критерий остановки для метода.

Для массива $\mathcal{A}$ размером $n^2 \times n^2 \times n^2$, отвечающему дискретизации интегрального оператора на сетке $n \times n \times n$, сложность нашего метода составит $\mathcal{O}(n^2 r^3)$ операций. При $r \sim \log^a n$ это на два

порядка лучше, чем сложность метода, предложенного И. Ибрагимовым, и на шесть порядков лучше, чем сложность метода, основанного на непосредственном вычислении HOSVD. Метод включает эвристические техники выбора ведущих элементов, однако его надежность не ниже, чем у крестового метода для матриц. В серии проведенных численных экспериментов не было замечено отклонений от требуемой точности. На тестовых массивах $\mathcal{A}$ размера $m \times m \times m$, размером от $m = 10^3$ до $m = 10^5$, скорость метода отвечает теоретической асимптотике, а на некоторых примерах быстрее нее. Применение этого метода к матрице, порожденной объемным интегральным уравнением, позволило представить данные, которые при полном хранении требовали бы $128\text{PB} = 128 \cdot 2^{50}\text{B}$ в объеме порядка 100MB. Вычисление потребовало всего 30 минут. Применение метода к аппроксимации матриц, порожденных объемными интегральными уравнениями, позволяет достичь сверхлинейного сжатия данных. Применяя тот же метод для сжатия *векторов* итерационного процесса, мы получаем метод решения сложности порядка $\mathcal{O}(n^2 \log^b n)$. При использовании равномерных сеток или дополнительной технологии эффективного представления факторов тензорного разложения (например, вейвлет-спарсификации), мы можем получить алгоритм *почти линейной* по n сложности.

i.4. Содержание работы по главам

Первая глава посвящена мозаично-скелетонному методу аппроксимации матриц и является своеобразной предысторией метода крестовой аппроксимации для тензоров. Ее первый раздел содержит краткое описание основных понятий и структуры мозаично-скелетонного метода. Там же приводится алгоритм построения крестовой аппроксимации блока на основе информации о малой части его элементов. Алгоритм основан на последовательном вычислении строк и столбцов с выбором ведущего элемента в духе метода определения ранга с помощью разложения Гаусса и с контролем точности по случайному множеству элементов [69, 71]. Приводится способ эффективного вычисления критерия остановки, не требующий проверки точности аппроксимации для всех элементов блока. Определенная эмпирика, присущая этому критерию, на практике приводит к тому, что полученная аппроксимация блока имеет несколько завышенный ранг. В том же разделе рассматриваются алгоритмы снижения ранга вычисленной аппроксимации, которые мы называем методами «дожимания» или переаппроксимации. Раздел завершают численные эксперименты, демонстрирующие выгоду от применения дожимателей.

В разделе 1.2 предложена параллельная версия мозаично-скелетонного метода. Поскольку после вычисления матричного разбиения все блоки могут быть вычислены и аппроксимированы независимо, задача состоит лишь в сбалансированном их распределении по процессорам. Предложена оценочная функция сложности и основанная на ней методика распределения блоков по процессорам, позволяющая достичь практически полной загруженности процессоров на этапе генерации матрицы. На этапе решения также достигается линейное ускорение при числе процессоров до 16.

Вторая глава посвящена методу трехмерной крестовой аппроксимации. В первом разделе содержится введение в задачу, показана выгода от применения тензорных аппроксимаций, объяснена связь тензорного разложения матриц с каноническим разложением массива, дан краткий обзор существующих методов построения трилинейного разложения, указаны их основные достоинства, ограничения и недостатки. Определено разложение Таккера, приведен алгоритм его вычисления на основе SVD. Поставлена задача о вычислении разложения Таккера с почти линейной сложностью.

Содержанием раздела 2.2 является теорема о существовании разложения Таккера $(*)$, в котором факторы U, V и W состоят из элементов массива. Это дает обоснование адаптивному методу построения разложения Таккера, последовательно наращивающему опорные базисы за счет вычисления новых строк, столбцов и распорок массива.

В разделе 2.3 дано описание алгоритма трехмерной крестовой аппроксимации. При этом изложение ведется последовательно в виде цепочки постепенно усложняемых алгоритмов. Начиная с простейшего метода, мы добавляем в него ускоряющие вычисление процедуры и рассматриваем способы быстрой реализации отдельных компонент, приходя в итоге к эффективному, но довольно объемному и сложному алгоритму. Таким образом мы демонстрируем значение всех частей метода, стараясь одновременно с этим не затуманить его основной идеи. В заключение раздела приведена оценка сложности нашего метода.

Раздел 2.4 посвящен деталям, которые необходимы для эффективной реализации трехмерного крестового метода. Обсуждается, как с почти линейной сложностью найти подматрицу прямоугольной матрицы, имеющую максимальный, или близкий к нему, объем. Использованный для этой цели алгоритм заимствован из диссертационной работы С. А. Горейнова [53]. Показано, как эффективно пересчитывать эту подматрицу при добавлении векторов в базис. Доказывается теорема, оценивающая максимальный элемент всей матрицы че-

рез максимальный элемент в ее подматрице максимального объема. На основе этой теоремы с почти линейной сложностью решается задача определения максимального (или близкого к нему) элемента в малоранговой матрице, заданной произведением факторов. Обсуждается технология добавления векторов в опорный набор, основанная на применении реортогонализации. Указана необходимость контроля над точностью аппроксимации на основе дополнительного, например, случайного, набора элементов. Предложена стратегия построения «нулевого приближения» к очередной вычисляемой срезке, значительно ускоряющая сходимость алгоритма после совершения нескольких первых итераций. Дана оценка точности полученного приближения.

Раздел 2.5 содержит численные эксперименты по сжатию модельных числовых массивов, подобных ядрам типовых интегральных операторов. Здесь мы приводим точную формулировку вычислительного метода (метод трилинейной аппроксимации является его ключевым этапом, но некоторый простор для реализации все же остается). Численные эксперименты проводятся с апостериорной проверкой, на основе которой продемонстрирована надежность метода. Приведены результаты сжатия данных, показана почти линейная скорость работы.

В разделе 2.6 приведено более систематическое экспериментальное исследование зависимости асимптотики полученного алгоритма от размера массива, от требуемой точности и от вида ядра. Показано, что практическая скорость работы метода полностью согласуется с теоретической оценкой, а в некоторых случаях оказывается даже лучше нее.

Третья глава содержит различные приложения описанных методов аппроксимации матриц для быстрого решения интегральных уравнений. Раздел 3.1 посвящен применению мозаично-скелетонного метода для решения задачи Дирихле для уравнения Гельмгольца на гладком контуре. В нем сформулированы задачи и приведены классические определения и факты из теории потенциала. Затем обсуждается метод дискретизации и способ вычисления матричных элементов. Эксперименты, проведенные с использованием параллельной вычислительной платформы, демонстрируют хорошее ускорение метода, а также значимую выгоду от применения алгоритмов переаппроксимации. Раздел 3.2 посвящен задаче гидроакустики мелкого моря, сформулированную в терминах теории потенциала. Применение для ее решения малоранговых аппроксимации позволило снизить время генерации матрицы и полного решения задачи от дней до минут. В разделе 3.3 приведены эксперименты по применению метода тензорных аппроксимаций для решения простейшего объемного интегрального уравнения. Для по-

строения эффективного итерационного метода предлагается использовать *структурированные векторы,* аппроксимируя их в формате разложения Таккера. Предлагается способ построения тензорного предобусловливателя заданного ранга. Демонстрируется сходимость полученного алгоритма и высокая скорость его работы. Однако исследование концепции структурированных векторов и полученных на ее основе вариантов итерационных методов, равно как и техники предобусловливания, конечно же, является предметом отдельной и весьма обширной работы.

В четвертой главе описывается одна из особенно интересных для нас практических задач, сводимых к решению интегрального уравнения. Речь идет о распространении электромагнитной волны в неоднородной трехмерной среде с затуханием. Для того, чтобы избежать сложностей с правильным выбором неотражающего краевого условия, задача формируется в виде объемного интегрального уравнения. В силу специального вида его ядра, на равномерной сетке матрица задачи является суммой трехуровневой теплицевой и трехуровневой теплиц-ганкель-теплицевой матрицы. Использование этих структур позволило сократить затраты памяти на хранение этой плотной матрицы с n^6 до $\mathcal{O}(n^3)$ элементов, но тем не менее, оперативная память персональной станции позволяет использовать только сетки размером до $64 \times 64 \times 64$. Для практических вычислений этого было недостаточно, так как в принципиально важном случае — приближении источника к зоне неоднородности — возникающие дискретизационные шумы не позволяли достичь требуемого разрешения. Размер сетки был увеличен за счет использования многопроцессорных систем. Предложен алгоритм распределенного хранения и умножения на теплицеву матрицу, учитывающий ее структуру и особенность кластерных систем. Его применение позволило производить расчет на сетках вплоть до $256 \times 256 \times 256$, используя 256 процессоров. Результаты этого расчета были использованы для решения обратной задачи, то есть зондирования неоднородности, и привели к хорошему качеству восстановления искомых параметров.

Используя метод трилинейной крестовой аппроксимации мы можем эффективно решить ту же задачу на одном процессоре, причем в том числе и на неравномерной сетке. Предварительные эксперименты показывают, что для матрицы обсуждаемой задачи эффективно строится малоранговое тензорное приближение. Это дает основание для еще более эффективных методов решения электродинамической задачи.

Глава 1. Мозаично-скелетонный метод

1.1. Описание и развитие мозаично-скелетонного метода

Детальное описание мозаично-скелетонного метода дано в работах [37, 51, 38, 11, 39]. Поскольку сам по себе мозаично-скелетонный метод не является центральным объектом изучения в данной работе, мы ограничимся лишь его конспектным изложением. В деталях мы опишем только алгоритм построения скелетонного разложения, послуживший прототипом алгоритма крестового разложения для тензоров. Также в этот раздел включены результаты по развитию метода — техника переаппроксимации и параллельная версия мозаично-скелетонного алгоритма, предложенные в [69].

1.1.1. Описание метода. Будем считать, что решаемая задача сформулирована в виде интегрального уравнения на границе некоторой области, и проведена дискретизация, порождающая линейную систему. Элементы матрицы этой линейной системы и использованная для дискретизации сетка являются входными данными для мозаично-скелетонного алгоритма. Основные этапы алгоритма и используемая в процессе информация приведены ниже в таблице.

построение дерева кластеров	геометрическая информация (сетка)
построение списка блоков	геометрическая информация (сетка) + асимптотические свойства ядра (общее представление о типе оператора)
аппроксимация блоков	малая часть элементов матрицы (процедура генерации элементов)

На этапе построения дерева кластеров используется информация о сетках, на которых проведена дискретизация исходного уравнения. Сетки рассматриваются как набор точек, которые объединяются по степени геометрической близости в *кластеры.* Кластеры в свою очередь формируют бинарное дерево, в корне которого лежит кластер, содержащий все узлы сетки, и при переходе на новый уровень дерева текущий кластер разбивается на два дочерних. На рисунке 1.1 показана сетка на эллиптическом контуре, разбитая на четыре кластера (второй уровень дерева кластеров).

Рис. 1.1. Дерево кластеров

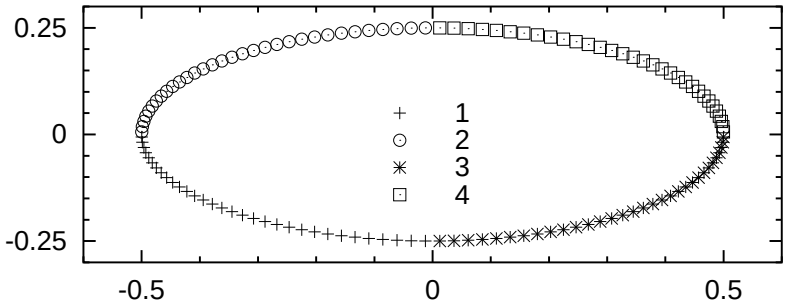
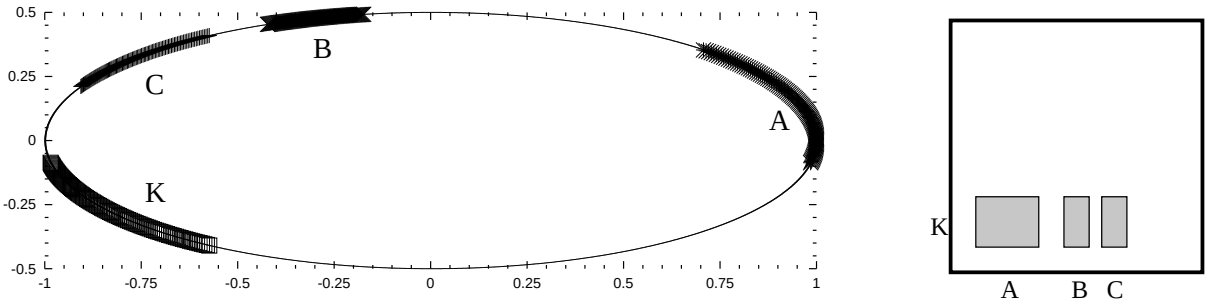


Рис. 1.2. Блоки в матрице



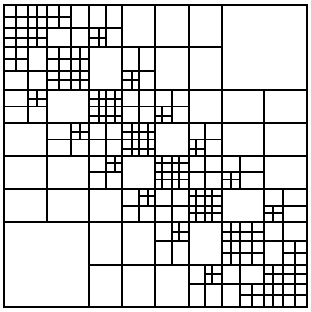
Имея два дерева кластеров, отвечающих паре сеток, на которых проведена дискретизация, мы разбиваем матрицу на иерархический набор блоков разного размера, каждый из которых отвечает взаимодействию группы *источников* с группой *приемников* (кластеров точек на первой и второй сетках). Блоки, отвечающие взаимодействию достаточно удаленных друг от друга кластеров точек, могут (при определенных предположениях относительно гладкости ядра интегрального оператора) быть приближены матрицей малого ранга. На рис. 1.2 сетки источников и приемников совпадают, при этом блок матрицы, отвечающий взаимодействию удаленных друг от друга кластеров K и A допускает малоранговую аппроксимацию. Блоки, отвечающие взаимодействию K с B и K с C, такой аппроксимации не допускают, поэтому на этапе построения блочного разбиения матрицы мы разделяем их на более мелкие, то есть спускаемся на следующий уровень дерева кластеров. Блочное разбиение, получившееся в результате действия этого алгоритма, представлено на рисунке 1.3.

Для каждого блока $A \in \mathbb{R}^{m\times n}$ решается задача поиска малоранговой аппроксимации

$$\|A - UV^T\| \leqslant \varepsilon, \tag{1.1}$$

причем ранг матрицы UV^T должен быть существенно меньшим, чем ее размеры. Для хранения блока в малоранговом формате нужна

Рис. 1.3. Мозаичное разбиение



память $\mathtt{mem}(UV^T) = (m + n)r$, тогда как полный блок занимал бы $\mathtt{mem}(A) = mn$ ячеек. С помощью скелетонной аппроксимации мы снижаем затраты на хранение каждого блока, и следовательно, матрицы в целом. Можно ввести так называемый *мозаичный ранг* матрицы $\mathbf{A} \in R^{M\times N}$ по формуле

$$\mathtt{mr}(\mathbf{A}) = \frac{\mathtt{mem}(\mathbf{A})}{M+N}, \qquad \mathtt{mem}(\mathbf{A}) = \sum \mathtt{mem}(UV^T) = \sum (m_i + n_i)r_i, \quad (1.2)$$

где суммирование ведется по всем блокам мозаичного разбиения. Чем ниже мозаичный ранг, тем больше эффект сжатия и ускорения от применения аппроксимации.

Метод быстрого умножения, основанный на приближении матрицы матрицей малого ранга, был впервые предложен в [37]. Малоранговую аппроксимацию можно искать, например, на основе сингулярного разложения матрицы A или аналогичных ему методов типа Ланцоша. Они используют все элементы матрицы A, а кроме того еще и довольно большое количество арифметических действий, поэтому для практических целей они малоприменимы.

В работах [38, 13, 39] задача малоранговой аппроксимации была связана с отысканием в матрице A столбцов и строк, на пересечении которых располагается подматрица *максимальъного объема* (максимального модуля детерминанта) среди всех подматриц размера r. В силу сложности данной задачи в условиях неполной информации выбор «хороших» столбцов и строк реализуется с помощью различных эвристических алгоритмов.

Наиболее просто и эффективно это делается на основе метода Гаусса с выбором ведущего элемента по некоторому шаблону [9]. В работе [2] в качестве такого шаблона выбираются строка и столбец. Аналогичным образом построен и метод, предложенный в работах [69, 71], который мы для полноты изложения повторим здесь.

Алгоритм 1 *(Cross2D).* Пусть задана матрица A размера $n \times n$ и требуется получить ее приближение матрицей $\tilde{A}_r$, являющейся суммой r одноранговых матриц $u_p v_p^T$ (называемых также *скелетонами*).

0 Пусть p — номер текущего шага. На первом шаге ($p = 1$) в матрице A выбирается некоторый столбец j_1.

1 В выбранном столбце j_p вычисляются все элементы, причем из вычисляемых элементов матрицы вычитаются значения всех предыдущих скелетонов в этих позициях. В полученном векторе находится наибольший по модулю элемент. Пусть он стоит на строке i_p.

2 Вычисляется строка i_p, из нее также вычитаются значения текущего аппроксиманта. Затем находится наибольший по модулю элемент в строке, причем элемент из столбца j_p повторно выбран быть не может. Пусть максимальный элемент стоит в столбце j_{p+1}.

3 По кресту с центром (i_p, j_p) строится скелетон, то есть матрица $u_p v_p^T$ ранга один, так, чтобы значения получившейся матрицы

$$\tilde{A}_p = \sum_{\alpha=1}^{p} u_\alpha v_\alpha^T \tag{1.3}$$

совпадали с точными значениями исходной матрицы на позициях p вычисленных столбцов $j_1, \ldots, j_p$ и p вычисленных строк $i_1, \ldots, i_p$.

4 Проверяется критерий остановки, и если он не удовлетворен, устанавливается $p := p + 1$ и метод повторяется с шага 1.

Насчитанная на p-м шаге работы алгоритма аппроксимация (1.3) считается достаточно точной, если выполняется критерий

$$\|A - \tilde{A}_p\| < \varepsilon \|A\|_F \approx \varepsilon \|\tilde{A}_p\|_F. \tag{1.4}$$

Проблема состоит в том, что для точной проверки этого критерия нам требуется вычислить все элементы матрицы, то есть совершить работу порядка n^2 действий, что слишком долго. Поэтому практический критерий остановки формулируется следующим образом:

$$(n - p)\|u_p v_p^T\|_F \leqslant \varepsilon \|\tilde{A}\|_F. \tag{1.5}$$

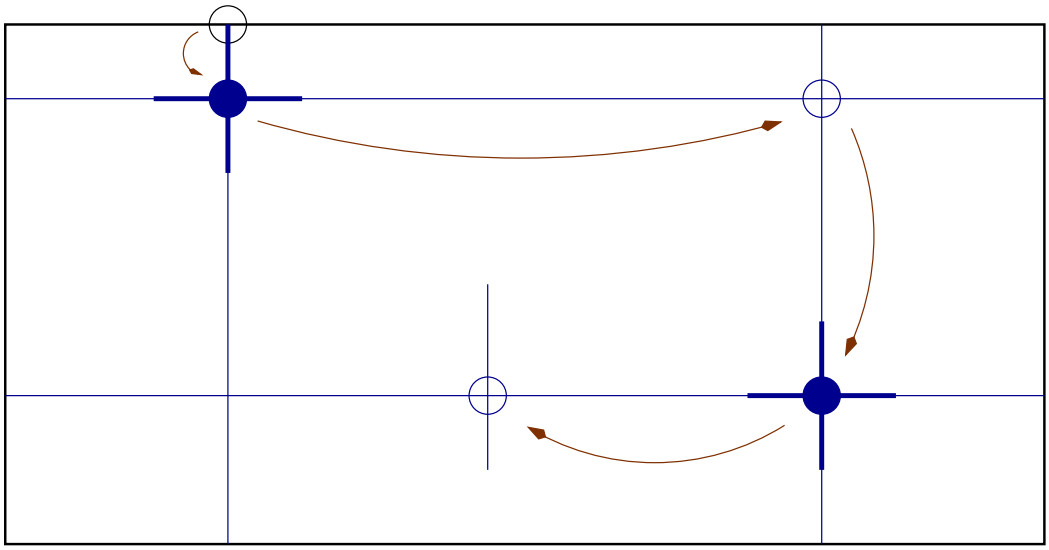


Рис. 1.4. Схема работы крестового метода

Выражение в левой части является верхней оценкой (обычно довольно грубой) для величины погрешности $\|A - \tilde{A}\|_F$ на p–том шаге, выражение в правой части аппроксимирует $\varepsilon\|A\|_F$. Для проверки критерия (1.5) не требуется знать всех элементов матрицы A, а достаточно только вычисленных p столбцов и строк.

1.1.2. Методы дожимания (переаппроксимации). Так как критерий остановки метода (1.5) более жесткий, чем (1.4), ранг построенного в результате работы метода крестовой аппроксимации аппроксиманта часто может быть снижен без значительного ущерба для точности. Рассмотрим алгоритмы, реализующие такое дожимание (алгоритмы переаппроксимации). Они основаны на вычислении сингулярного разложения для имеющейся малоранговой матрицы и снижении ранга за счет отбрасывания младших сингулярных векторов, с контролем точности по величине младших сингулярных значений.

Сингулярное разложение для матрицы может быть построено на основе итерационного метода типа Ланцоша, примененного к матрице

$$A = UV^T, \qquad U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r} \qquad r \quad \text{— ранг аппроксимации} \tag{1.6}$$

Умножение на A в малоранговом представлении требует $nr + mr$ действий, поэтому при $r \ll n, m$ скорость алгоритма на порядок выше, чем для плотной матрицы.

Второй способ основан на неявном построении сингулярного разложения для матрицы A, представленной в виде (1.6). Ортогонализуем

факторы U и V, например, с помощью QR–алгоритма.

$$U = Q_U R_U, \quad V = Q_V R_V, \qquad Q_U \in \mathbb{R}^{m\times r},\ Q_V \in \mathbb{R}^{n\times r},\ R_U, R_V \in \mathbb{R}^{r\times r}.$$

Тогда

$$A = Q_U R_U R_V^T Q_V^T$$

Построим сингулярное разложение для матрицы

$$B = R_U R_V^T =: U_B \Sigma V_B^T.$$

Тогда

$$A = Q_U U_B \Sigma V_B^T Q_V^T$$

Обозначив $U_A = Q_U U_B$, а $V_A = Q_V V_B$, имеем

$$A = U_A \Sigma V_A^T, \qquad U_A \in \mathbb{R}^{m\times r}, \quad V_A \in \mathbb{R}^{n\times r}, \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$$

Полученное разложение есть с точностью до нулевых сингулярных чисел сингулярное разложение матрицы A, вычисленное без явного ее формирования. Если погрешность в аппроксимации составляет $\tilde{\varepsilon}$, а мы допускаем погрешность $\varepsilon \geqslant \tilde{\varepsilon}$, то можно отбросить младшие сингулярные числа, руководствуясь условием

$$\left(\sum_{s=\tilde{r}+1}^{r} \sigma_s^2 \right)^{1/2} \leqslant \varepsilon - \tilde{\varepsilon},$$

где $\tilde{r}$ — новое, уменьшенное число скелетонов, аппроксимирующих A с точностью ε. Тем самым мы снизим ранг A, контролируя внесенную погрешность. Этот метод дожимания имеет сложность $\mathcal{O}((m+n)r^2)$, основные затраты идут на ортогонализацию факторов.

1.1.3. Численные эксперименты. Проводились тесты по аппроксимации мозаично–скелетонным методом матрицы, порожденной функцией

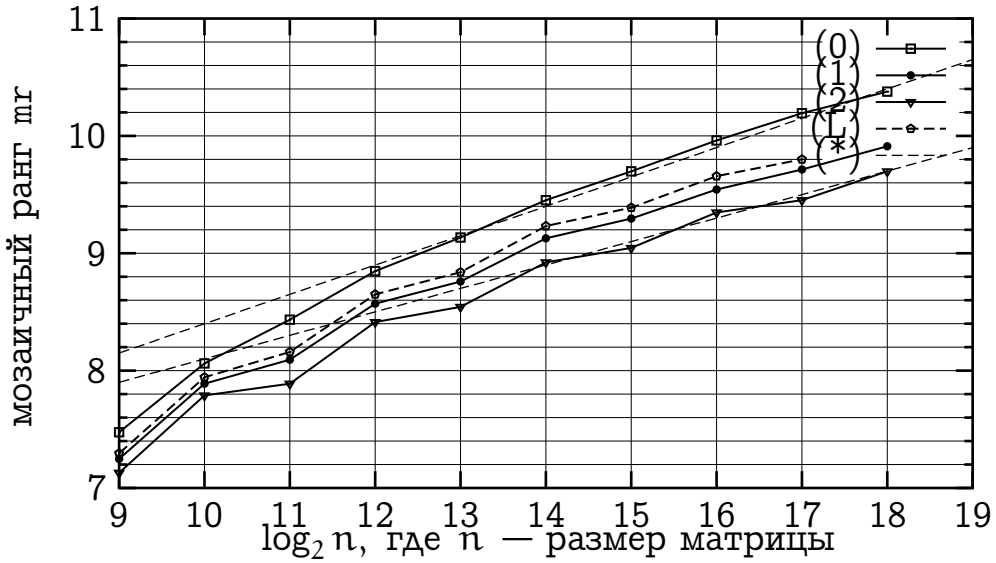
$$A_{ij} = f(x_i, y_j) = \frac{1}{\rho(x_i, y_j)} = \frac{1}{\|(x_i - y_j)\|_2}$$

на равномерной сетке в квадрате

$$\Omega = \{[0;1] \otimes [0;1]\} \subset \mathbb{R}^2.$$

Точность аппроксимации составляла $\varepsilon = 10^{-4}$. Аппроксимация блоков проводилась методом Cross2D. В качестве переаппроксиматора брался алгоритм, построенный по методу Ланцоша и метод, основанный на SVD. Проводился и тест без переаппроксиматора.

Рис. 1.5. Эффект от применения алгоритмов дожимания



Результаты теста аппроксимации представлены на картинке 1.5. На графике представлена зависимость мозаичного ранга от размера матрицы. Кривые отвечают разным алгоритмам построения аппроксимации.

(0) Метод Cross2D без переаппроксимации;

(1) Cross2D, переаппроксимация методом Ланцоша;

(2) Cross2D, переаппроксимация SVD;

(L) Метод Ланцоша применяется к полному блоку;

(*) Прямая, задающая наклон графиков.

Видно, что использование переаппроксиматоров снижает мозаичный ранг, как по абсолютной величине, так и по асимптотике. А именно, если наклон кривой в логарифмическом масштабе составлял без переаппроксимации 0.25, то использование переаппроксиматоров снизило его до 0.20. Абсолютное значение мозаичного ранга уменьшается примерно в полтора раза. Лучшее сжатие предоставляет метод с переаппроксимацией на основе неявно получаемого SVD.

1.2. Параллельная версия метода

Мозаично–скелетонный метод ориентирован на работу с плотными матрицами больших и очень больших размеров. Объем памяти, зани-

маемый такими матрицами, очень велик, и даже с учетом мозаично-скелетонного сжатия данных, может превысить размер оперативной памяти отдельной рабочей станции. Поэтому для его практического применения к реальным задачам может потребоваться использование многопроцессорной платформы. Впервые такая попытка была проделана в работе [59], где алгоритм адаптировался к использованию на параллельном компьютере МВС-100. В результате было продемонстрировано хорошее распараллеливание метода. Нами предложена параллельная версия алгоритма, позволяющая эффективно применить метод на многопроцессорной платформе с распределенной памятью, реализованная на основе стандарта `MPI` и ориентированная на использование на вычислительных кластерах.

1.2.1. Особенности кластерных станций. В наибольшей степени мы ориентировались на *кластерные станции*, которые представляют собой вычислительную систему на основе обычных рабочих станций, соединенных высокоскоростной сетью. Такой принцип построения вычислительных систем сравнительно недорог, но кластерные станции обладают рядом особенностей:

- скорость межпроцессорных соединений невысока, много медленнее скорости операций внутри одного вычислительного узла;
- время передачи массива данных складывается из фиксированного *времени инициализации,* которое может быть весьма велико, и *времени пересылки,* пропорциональное длине массива;
- скорость обмена данными между соседними узлами выше, чем скорость обмена данными между случайной парой узлов.

Исходя из этих особенностей вычислительной платформы, при разработке параллельного алгоритма по-возможности следует:

- уменьшать число обменов данными между вычислительными узлами, возможно даже за счет увеличения вычислительной сложности на каждом узле;
- производить передачу данных крупными блоками;
- производить обмен данными между соседними вычислительными узлами, пользуясь процедурами глобального обмена данными только по необходимости;

- организовать алгоритм так, чтобы минимизировать «простой» вычислительного узла, связанный с ожиданием прибытия необходимых данных с других узлов. Этого можно достигнуть следующими способами:
  - хранить на каждом узле необходимый запас данных для проведения расчета. При этом каждый вычислительный узел сам отслеживает количество оставшихся у него, своевременно информирует корневой узел о необходимости прислать дополнительные и получает их раньше, чем закончит работу с текущими данными;
  - производить операции пересылки одновременно с операциями, требующими существенных расчетов на вычислительном узле. Например, обмениваться частями распределенного вектора в то же время, когда происходит умножение этого вектора на локальные части матриц.

Более подробно специфика кластерных станций изучена при выполнении работы [70]. Указанные технологические особенности были учтены при выборе модели параллелизации.

1.2.2. Модель параллелизации и распределение нагрузки. В большинстве практических задач основную часть времени при больших размерах матрицы занимает вычисление ее элементов, особенно если они выражаются в виде многомерных интегралов от ядра интегрального уравнения и используемых при дискретизации базисных (и пробных) функций. Поэтому при параллельной реализации основное внимание следует уделить блоку построения аппроксиманта. Так как после построения блочного разбиения каждый блок матрицы аппроксимируется независимо от других, естественным шагом к построению параллельного метода является распределение блоков по процессорам. Каждый процессор получает список закрепленных за ним блоков матрицы, независимо вычисляет требуемые матричные элементы и строит для них аппроксимацию. Взаимодействие процессоров происходит только при умножении на матрицу — каждый процессор умножает на вектор те блоки, которые у него имеются, после чего результат суммируется по всем процессорам. Это хорошо согласуется с технологическими особенностями кластерных систем, так как количество обменов между процессорами невелико и хорошо локализовано.

При распределении блоков (а значит и вычислительной нагрузки) по процессорам мы можем следовать одной из двух стратегий:

- «планирование» : вся работа сразу распределяется между процессорами по какому–либо правилу.

- «куча» : работа распределяется между процессорами лишь частично, оставшаяся часть запасается в «куче» . Выполнив свою порцию работы, процессор забирает новое задание из «кучи» .

Второй вариант требует дополнительного обмена данными между корневым и рабочими процессорами, что усложняет логику программы, вносит некоторую задержку на этапе исполнения и иногда требует снять с корневого процессора собственно расчетную нагрузку, освободив его только для решения коммуникативных задач. Вообще говоря, стратегию «кучи» необходимо применять в случае, когда вычислительная сложность отдельных этапов алгоритма не может быть оценена заранее. При реализации мозаично-скелетонного метода время решения отдельной подзадачи (аппроксимации отдельного блока) вполне предсказуемо и составляет $(m+n)r$ вычислений элементов блока и $\mathcal{O}((m+n)r^2)$ дополнительных арифметических действий. Размеры блоков известны до начала их аппроксимации, а значения рангов оцениваются на основе той же информации о ядре, которая была использована при построении мозаичного разбиения матрицы.

Итак, схема распределения блоков по процессорам следующая:

- блоки упорядочиваются, начиная с самых больших, затем последовательно распределяются по процессорам;

- очередной блок закрепляется за тем процессором, суммарный «вес» блоков на котором в данный момент наименьший.

Вес блока размером $m \times n$ оценивается как $(m+n)r(m,n)$, где $r(m,n)$ — оценка для ранга блока вида $r \sim \log^{\gamma}(m+n)$, в котором γ определяется типом ядра и размерностью пространства. При числе процессоров вплоть до 32 с помощью этого метода мы получали распределение нагрузки, при котором время ожидания процессора не превышало 10% от времени его работы.

1.3. Выводы

В этой главе мы рассмотрели метод неполной крестовой (билинейной) аппроксимации матриц. Он станет нашей базой для развития метода крестовой (трилинейной) аппроксимации для тензоров. Матрица

присутствует в этом алгоритме лишь *виртуально* и задана процедурой вычисления любого элемента; полностью она никогда не вычисляется и не хранится. Поэтому для построения аппроксимации достаточно почти линейного по размеру матрицы числа операций. Мы рассмотрели способ эффективного вычисления критерия остановки, не требующий проверки точности аппроксимации для всех элементов блока. Определенная эмпирика, присущая этому критерию, на практике приводит к тому, что полученная аппроксимация блока имеет несколько завышенный ранг. Мы предложили алгоритмы снижения ранга вычисленной аппроксимации, которые мы называем методами «дожимания» или переаппроксимации. Численные эксперименты демонстрируют, что применение переаппроксимации приводит к существенному снижению затрат на хранение матрицы.

Мы также предложили параллельную версию мозаично-скелетонного метода. Численные эксперименты, демонстрирующие ускорение метода на параллельной платформе, мы продемонстрируем в разделе 3.1.3 на примере решения конкретной задачи.

Глава 2.
Метод трехмерной крестовой аппроксимации

2.1. Введение

Не так давно предложено искать аппроксимации матрицы A в виде суммы тензорных произведений [17, 40, 67]

$$A \approx \tilde{A}_r = \sum_{\alpha=1}^{r} A_\alpha^1 \times A_\alpha^2 \times \ldots \times A_\alpha^m.$$

Символ „$\times$“ означает прямое (кронекерово) произведение матриц: если $P = [p_{ij}]$ имеет размер $m_p \times n_p$ и $Q = [q_{ij}]$ имеет размер $m_q \times n_q$, то их прямое произведение есть матрица, имеющая размеры $(m_p m_q) \times (n_p n_q)$ и заданная формулой

$$P \times Q = [p_{ij} Q].$$

Выгода от применения тензорных аппроксимаций достаточно очевидна (см., например, [19]). Пусть сомножители A_α^s, $s = 1, \cdots, m$, являются квадратными матрицами размера p, тогда матрица A тоже квадратная и имеет размер p^m. На ее хранение в плотном формате требуется p^{2m} ячеек памяти, ее умножение на вектор производится за p^{2m} операций. Общее число элементов в матрицах, задающих факторы тензорного разложения, равняется rmp^2, умножение $\tilde{A}_r$ на вектор требует $rmp^2 p^{m-1} = rmp^{m+1}$ операций. Если определить *коэффициент сжатия* как отношение объема памяти, требуемого для хранения аппроксимации, к объему памяти, требуемому для хранения всей матрицы, то можно утверждать, что тензорные аппроксимации как метод сжатия данных обладают коэффициентом сжатия, равным $rmp^{2-2m} = rmn^{2/m-2}$, где $n = p^m$ — линейный размер матрицы A. При $m \geqslant 3$, таким образом, тензорные аппроксимации производят *сверхлинейное сжатие* данных: коэффициент сжатия стремится к нулю быстрее, чем $1/n$. К тому же, если многоуровневая матрица обладает дополнительной структурой (например, ее уровни теплицевы), то факторы тензорного разложения наследуют структуру соответствующих уровней исходной матрицы, что приводит к еще большему

сжатию и (что даже важнее) более быстрому умножению получаемого аппроксиманта на вектор (см., например, [28]).

В большинстве случаев число кронекеровских сомножителей m выбирается равным размерности физического пространства, в нашей работе $m = 3$. Если элементы матрицы a_{ij} порождаются значениями некоторой *асимптотически гладкой* функции $f(x, y)$, вычисленными в точках

$$x \in \{x_i\}_{i=1}^{n}, \qquad y \in \{y_j\}_{j=1}^{n},$$

где наборы $\{x_i\}$ и $\{y_j\}$ суть узлы сеток, полученных декартовым произведением трех одномерных, то индексы i, j матрицы A можно выразить с помощью мультииндексов (многомерных индексов)

$$i = (i_1, i_2, i_3), \qquad j = (j_1, j_2, j_3).$$

Здесь i_1, i_2 и i_3 — индексы узла x_i по одномерным сеткам, а j_1, j_2 и j_3 — аналогичные индексы узла y_j. Иногда такое представление индексов элемента матрицы с помощью мультииндекса вводят более искусственно. В любом случае, как только элементы матрицы можно задать с помощью мультииндексов, мы можем расщепить и перегруппировать компоненты мультииндексов следующим образом:

$$a_{ij} = a_{(i_1,i_2,i_3)(j_1,j_2,j_3)} = a_{(i_1,j_1)(i_2,j_2)(i_3,j_3)} = a_{ijk},$$

вводя новые, парные, индексы $i = (i_1, j_1)$, $j = (i_2, j_2)$, $k = (i_3, j_3)$. Теперь видно, что для того, чтобы матрица

$$A = [a_{(i_1,i_2,i_3)(j_1,j_2,j_3)}]$$

могла быть представлена в виде суммы r тензорных произведений

$$A = \sum_{\alpha=1}^{r} U_\alpha \times V_\alpha \times W_\alpha, \tag{2.1}$$

$$U_\alpha = [u_{(i_1,j_1)\alpha}], \quad V_\alpha = [v_{(i_2,j_2)\alpha}], \quad W_\alpha = [w_{(i_3,j_3)\alpha}],$$

массив a_{ijk} должен обладать представлением в виде

$$a_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha},$$

называемым еще *трилинейным разложением.*

Далее мы будем обсуждать методы построения разложений (точных или приближенных) трехмерного массива $\mathcal{A} = [a_{ijk}]$, обозначая

его элементы a_{ijk} и не обращая внимание на то, что его индексы являются парными мультииндексами. Для простоты будем считать, что размеры массива по всем трем направлениям равны n.

Предлагаемые в настоящий момент подходы к поиску трилинейного разложения можно условно разделить на «минимизационные» и «матричные» . В первом случае задачу нахождения трилинейного разложения ранга r

$$a_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha} \tag{2.2}$$

рассматривают как задачу минимизации нормы отклонения

$$\min_{u,v,w} \sum_{i,j,k} \left(\sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha} - a_{ijk} \right)^2 . \tag{2.3}$$

Для минимизации этого функционала могут применяться различные стандартные итерационные методы минимизации [8, 75, 79, 81], чаще всего они требуют выполнения порядка n^3 операций на каждой итерации. Основной проблемой минимизационных методов для этой задачи является их весьма медленная сходимость, одной из причин возникновения которой является неединственность решения задачи минимизации. Другая сложность — необходимость априорно знать ранг трилинейной аппроксимации, что в большинстве случаев невозможно, или последовательно решать серию задач минимизации (2.3) с разными r.

Под «матричными» методами получения разложения (2.2) мы понимаем алгоритмы, основанные на *одновременной редукции* матриц–срезок массива $\mathcal{A}$, то есть вычисления для матриц $A_k = [(a_k)_{ij}]$ одновременного представления в виде

$$A_k = U B_k V^T,$$

где B_k имеет специальный вид (треугольная или диагональная). Матричные методы развиты в основном для случая, когда $r = n$ (см. [75, 77]), они обладают сложностью порядка n^4, однако условие $r = n$ достаточно сильно ограничивает их применение, поскольку на практике вообще говоря трилинейное разложение ранга $r = n$ может не приближать массива $\mathcal{A}$ с заданной точностью. Нам не удалось обнаружить в литературе упоминания о матричных методах, применимых для случая $r > n$. Для $n < r < 2n - 3$ мы предлагаем такой метод в [78], его сложность $\mathcal{O}(n^4)$. Суммируя сказанное и учитывая оценки сложности всех упомянутых методов, можно заключить, что как для

минимизационных, так и для матричных методов область применимости ограничена небольшими размерами n, на практике в основном не превышающими $n = 128$. При решении интегральных уравнений нам требуется рассматривать массивы куда больших размеров.

В качестве промежуточного (впрочем, иногда и достаточного) шага при поиске трилинейного разложения рассматривают поиск разложения Таккера [36]

$$a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'}, \tag{2.4}$$

ядро которого $\mathcal{G} = [g_{i'j'k'}]$ является трехмерным массивом размера $r_1 \times r_2 \times r_3$, а факторы U, V, W являются ортогональными матрицами. Практический интерес представляет задача построения разложения Таккера, размеры ядра которого много меньше, чем размеры исходного массива. Затем к этому ядру применяют упомянутые выше методы поиска трилинейного разложения. Если трилинейное разложение для $\mathcal{G}$ найдено, то тем самым найдено трилинейное разложение того же ранга для исходного массива $\mathcal{A}$.

Для нахождения разложения Таккера известен метод, основанный на вычислениях сингулярных разложений для трех *разверток* исходного массива:

$$\begin{aligned} A^{(1)} &= [a^1_{i(jk)}] = [a_{ijk}], \\ A^{(2)} &= [a^2_{j(ki)}] = [a_{ijk}], \\ A^{(3)} &= [a^3_{k(ij)}] = [a_{ijk}], \end{aligned} \tag{2.5}$$

то есть матриц, полученных переупорядочением элементов исходного массива и имеющих размер $n \times n^2$. Левые («короткие») сингулярные векторы разложений

$$A^{(1)} = U\Sigma_1\Psi^T, \qquad A^{(2)} = V\Sigma_2\Phi^T, \qquad A^{(3)} = W\Sigma_3\Upsilon^T \tag{2.6}$$

дают матрицы-факторы U, V, W разложения Таккера, ядро же вычисляется как *свертка* массива $\mathcal{A}$ с матрицами U, V, W :

$$g_{i'j'k'} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_{ijk} u_{ii'} v_{jj'} w_{kk'}. \tag{2.7}$$

Сложность SVD-разложений составляет $\mathcal{O}(n^4)$, поэтому применять метод (2.6) непосредственно к массивам $\mathcal{A}$ при $n > 128$ крайне затруднительно. Более того, даже простое вычисление и хранение самого массива $\mathcal{A}$ (то есть выполнение $\mathcal{O}(n^3)$ операций) может оказаться слишком дорогостоящей задачей при n порядка тысячи.

Для решения этой проблемы предлагается метод, который использует идею алгоритма *крестовой аппроксимации* [38, 39], и не требует вычисления всех элементов исходной матрицы, а лишь использует процедуру их генерации, вычисляя порядка $n \log^{\beta} n \log^{\gamma} \varepsilon^{-1}$ элементов. Положительные числа β и γ зависят от свойств матрицы A, а ε определяет требуемую точность аппроксимации. Таким образом, предлагаемый алгоритм обладает той же, почти линейной по n оценкой сложности, то есть строит аппроксимацию $\tilde{A}_r$ массива $\mathcal{A}$ за $\mathcal{O}(n \log^{\beta} n \log^{\gamma} \varepsilon^{-1})$ операций.

Естественно, для того чтобы по предлагаемому нами алгоритму можно было найти для массива $\mathcal{A}$ трилинейное разложение малого ранга r, точно или с определенной погрешностью ε, оно должно существовать. Вопросы существования такого разложения, возникающие требования к массивам $\mathcal{A}$ и теоретические оценки на ранг такого разложения, в зависимости от свойств массива, предложены, например, в работах [40, 67]. В следующем разделе мы покажем, что факторы U, V и W такого разложения можно составить из некоторых строк, столбцов и распорок массива $\mathcal{A}$. В дальнейшем при описании алгоритма мы всегда полагаем, что искомая аппроксимация с заданной точностью у рассматриваемого массива существует, и останавливаемся на способе ее поиска. Здесь и далее возникающая в оценках сложности метода величина r будет означать ранг (ε-ранг) трилинейного разложения для исходного массива, а об оценках вида $\mathcal{O}(nr^d)$ мы будем говорить, как о *почти линейных* по n.

2.2. Теорема существования

Итак, пусть задан трехмерный массив $\mathcal{A} = [a_{ijk}]$ размером $n_1 \times n_2 \times n_3$. Пусть на основе априорной информации о свойствах массива нам известно, что для него существует аппроксимация в виде разложения Таккера с рангами (r_1, r_2, r_3), погрешность которой составляет ε

$$a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'} + \varepsilon_{ijk}, \qquad |\varepsilon_{ijk}| \leqslant \varepsilon. \tag{2.8}$$

Возникает вопрос, можно ли в в качестве U, V и W взять некоторые наборы строк, столбцов и распорок исходного массива? Положительный ответ на него дает следующая

Теорема 1 Пусть равенство (2.8) выполняется для некоторых U, V, W

и $\mathcal{G}$. Тогда существует аппроксимация

$$a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g'_{i'j'k'} u'_{ii'} v'_{jj'} w'_{kk'} + \varepsilon'_{ijk}, \qquad |\varepsilon'_{ijk}| \leqslant \varepsilon', \tag{2.9}$$

в которой факторы U', V', W' состоят соответственно из r_1 столбцов, r_2 строк и r_3 распорок массива $\mathcal{A}$, а для вычисления ядра $\mathcal{G}'$ требуются лишь элементы факторов. Погрешность такой аппроксимации не превосходит

$$\varepsilon' \leqslant ((r_1 + 1) + r_1(r_2 + 1) + r_1 r_2(r_3 + 1))\,\varepsilon. \tag{2.10}$$

Доказательство. Поскольку массив $\mathcal{A}$ может быть представлен разложением Таккера ранга (r_1, r_2, r_3) с точностью ε, его развертка $A^{(1)} = [a_{i(jk)}]$ может быть приближена матрицей ранга r_1 с той же точностью ε. Матрица малого ранга может быть представлена в виде *скелетонного разложения*

$$A^{(1)} = U B_*^{-1} B + \mathcal{E}_1,$$

где матрица U составлена из r_1 столбцов $A^{(1)}$ (то есть из r_1 столбцов массива $\mathcal{A}$), матрица B из r_1 ее строк, а невырожденная матрица B_* лежит на пересечении U и B. В работе [13] показано, что если в качестве B_* выбрать подматрицу $B_\square$ максимального объема среди всех подматриц такого размера в $A^{(1)}$, точность полученного разложения

$$A^{(1)} = U B_\square^{-1} B + \mathcal{E}_1$$

оценивается поэлементно как

$$\operatorname{maxel}(\mathcal{E}_1) \leqslant (r_1 + 1)\varepsilon$$

(здесь maxel означает максимальный по модулю элемент матрицы). Кроме того, в [13] показано, что если $B_\square$ — подматрица максимального объема, то все элементы $U B_\square^{-1}$ не превосходят по модулю единицы. Итак,

$$\left| a_{ijk} - \sum_{s=1}^{r_1} \sum_{s'=1}^{r_1} u_{is} \varphi_{ss'} b_{s'jk} \right| \leqslant (r_1 + 1)\varepsilon, \tag{2.11}$$

$$\left| \sum_{s=1}^{r_1} u_{is} \varphi_{ss'} \right| = |\tilde{u}_{is'}| \leqslant 1, \qquad i = 1, \ldots, n_1, \quad s' = 1, \ldots, r_1.$$

Отметим, что так как подматрица $B_\square$ расположена в столбцах U, то для вычисления значений $B_\square^{-1} = [\varphi_{ss'}]$ нам не требуется получать новых элементов $\mathcal{A}$.

Рассмотрим элементы матрицы B. Они представляют собой выборку некоторых срезок массива $\mathcal{A}$. Переупорядочив их, развернем этот массив по индексу j: $\mathcal{B} = [b_{j(ks')}]$. В силу существования аппроксимации (2.8), ε-ранг этого массива не превосходит r_2. Записывая скелетонное разложение для $\mathcal{B}$, и пользуясь результатами [13], получаем

$$\left| b_{jks'} - \sum_{t=1}^{r_2} \sum_{t'=1}^{r_2} v_{jt} \psi_{tt'} c_{t'ks'} \right| \leqslant (r_2 + 1)\varepsilon, \tag{2.12}$$

где матрица $V = [v_{jt}]$ состоит из r_2 строк массива $\mathcal{A}$, массив $\mathcal{C} = [c_{k(s't')}]$ является выборкой распорок массива $\mathcal{A}$, а для вычисления коэффициентов $\varphi_{tt'}$ не требуется новых элементов $\mathcal{A}$.

Массив $\mathcal{C}$ также может быть представлен скелетонным разложением с оценкой погрешности

$$\left| c_{ks't'} - \sum_{p=1}^{r_3} \sum_{p'=1}^{r_3} w_{kp} \gamma_{pp'} d_{p's't'} \right| \leqslant (r_3 + 1)\varepsilon, \tag{2.13}$$

где матрица $W = [w_{kp}]$ состоит из r_3 распорок $\mathcal{A}$, а для вычисления коэффициентов $\gamma_{pp'}$ и элементов $d_{p's't'}$, составляющих при правильном упорядочивании ядро $\tilde{G}' = [g_{s't'p'}] = [d_{p's't'}]$, не требуются новые элементы $\mathcal{A}$.

Окончательно погрешность приближения элемента a_{ijk} оценивается как

$$\left| a_{ijk} - \sum_{s=1}^{r_1} \sum_{s'=1}^{r_1} \sum_{t=1}^{r_2} \sum_{t'=1}^{r_2} \sum_{p=1}^{r_3} \sum_{p'=1}^{r_3} u_{is} \varphi_{ss'} v_{jt} \psi_{tt'} w_{kp} \gamma_{pp'} g_{s't'p'} \right| \leqslant$$

$$\leqslant \left| a_{ijk} - \sum_{s=1}^{r_1} \sum_{s'=1}^{r_1} u_{is} \varphi_{ss'} b_{s'jk} \right| + \sum_{s'=1}^{r_1} \left| b_{jks'} - \sum_{t=1}^{r_2} \sum_{t'=1}^{r_2} v_{jt} \psi_{tt'} c_{t'ks'} \right| +$$

$$+ \sum_{s'=1}^{r_1} \sum_{t'=1}^{r_2} \left| c_{ks't'} - \sum_{p=1}^{r_3} \sum_{p'=1}^{r_3} w_{kp} \gamma_{pp'} d_{p's't'} \right|,$$

откуда с учетом (2.11),(2.12) и (2.13), имеем

$$\left| a_{ijk} - \sum_{s'=1}^{r_1} \sum_{t'=1}^{r_2} \sum_{p'=1}^{r_3} u_{is} v_{jt} w_{kp} g'_{s't'p'} \right| \leqslant \left((r_1 + 1) + r_1 (r_2 + 1) + r_1 r_2 (r_3 + 1) \right) \varepsilon.$$

В последнем равенстве элемент ядра $\mathcal{G}'$ определяется как

$$g'_{s't'p'} = \sum_{s=1}^{r_1} \sum_{t=1}^{r_2} \sum_{p=1}^{r_3} \varphi_{ss'} \psi_{tt'} \gamma_{pp'} g_{stp}.$$

Таким образом, существование аппроксимации (2.9) и оценка точности (2.10) доказаны. □

При $r_1 = r_2 = r_3 = r$ коэффициент роста погрешности ограничен величиной $c \leqslant (r+1)^3$.

В общем случае наша оценка не является симметричной относительно перестановки модовых рангов r_1, r_2 и r_3, и по всей видимости, получение симметричного результата требует иной техники доказательства. Такая оценка (с меньшим значением константы c) получена С. Горейновым.

2.3. Трехмерный крестовый метод

Нам представляется более правильным давать изложение предлагаемого алгоритма в виде цепочки методов, начиная с простых для восприятия и переходя ко все более эффективным (а вместе с тем более детальным и сложным) версиям алгоритма. При этом на каждом шаге мы поясняем, какую цель преследуют вводимые в алгоритм новые элементы, то есть изложение как бы следует истории разработки метода. Для удобства в этом разделе мы полагаем $n_1 = n_2 = n_3 = n$ и $r_1 = r_2 = r_3 = r$.

2.3.1. Как нельзя построить этот алгоритм. Работая с матрицей, крестовый алгоритм последовательно выбирает максимальные элементы в строках и столбцах остатка, то есть разности между исходной матрицей и насчитанной на текущем шаге аппроксимацией, и добавляет к уже насчитанной аппроксимации *скелетоны,* то есть матрицы ранга один, вычисленные по столбцу и строке, на пересечении которых стоит очередной максимальный элемент. Попытки наивно построить таким же образом алгоритм для крестовой аппроксимации трехмерного массива не приводят к успеху. В самом деле, если, например, последовательно выбирать максимальный индекс в столбце, строке и *распорке* (пусть так называется вектор, полученный выборкой вдоль третьего направления массива) и вычислять триплет $u_p \otimes v_p \otimes w_p$ так, чтобы на вычисленных позициях элементы аппроксиманта и исходного массива совпадали, то такой метод не будет с разумной скоростью сходиться к трилинейной аппроксимации небольшого ранга, как в общем случае, так и для матриц, характерных для практических задач.

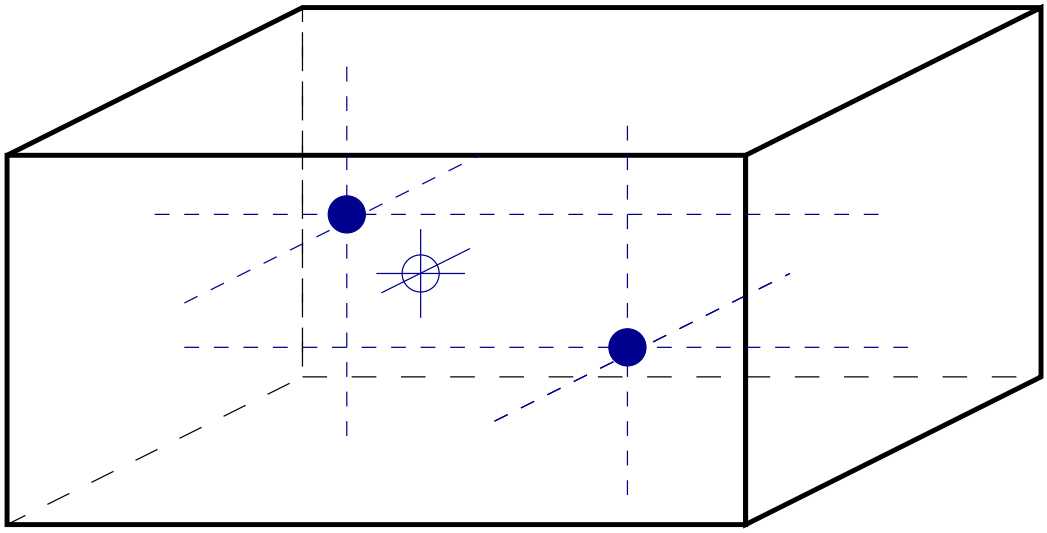


Рис. 2.1. Прямое обобщение крестового метода для тензора

Более того, если даже выбирать на каждом шаге максимальный по модулю элемент $\mathcal{A}-\tilde{\mathcal{A}}_p$ среди всех n^3 элементов этой разности (полагая их известными), то и такой метод не будет сходиться к трилинейной аппроксимации малого ранга.

Здесь мы определяем *триплет*

$$\mathcal{A} = u \otimes v \otimes w$$

как трехиндексный массив $\mathcal{A} = [a_{ijk}]$ с элементами

$$a_{ijk} = u_i v_j w_k.$$

Мы будем также пользоваться в дальнейшем символом $\otimes$ и в более общем смысле, например, обозначая с помощью

$$\mathcal{A} = A \otimes w$$

трехиндексный массив $\mathcal{A} = [a_{ijk}]$ с элементами

$$a_{ijk} = A_{ij} w_k.$$

Вообще, если $\mathcal{A}$ является p–индексным массивом с элементами $a_{i_1,i_2,\ldots,i_p}$, а $\mathcal{B}$ — q–индексным массивом с элементами $b_{j_1,j_2,\ldots,j_p}$, то $\mathcal{C} = \mathcal{A} \otimes \mathcal{B}$ является $(p+q)$–индексным массивом с элементами

$$c_{i_1,\ldots,i_p,j_1,\ldots,j_q} = a_{i_1,\ldots,i_p} b_{j_1,\ldots,j_q}.$$

Эта операция носит в литературе название *внешнего произведения* (outer product).

2.3.2. Как можно построить этот алгоритм. Рассмотрим развертки массива $\mathcal{A}$, определенные формулой (2.5), как прямоугольные матрицы размера $n \times n^2$ и применим к ним алгоритм поиска крестовой аппроксимации. Если у массива $\mathcal{A}$ существует трилинейное разложение ранга r, то для разверток $A^{(1)}$, $A^{(2)}$, $A^{(3)}$ существует аппроксимация

$$\tilde{A}_r^{(1)} = U\Phi^T \qquad \tilde{A}_r^{(2)} = V\Psi^T \qquad \tilde{A}_r^{(3)} = W\Upsilon^T,$$

где U, V, W имеют размер $n \times r$, а матрицы Φ, Ψ, Υ имеют размер $n^2 \times r$. Базисы, заданные матрицами U, V, W, формируют факторы разложения Таккера (достаточно найти ортогональные базисы $\tilde{U}$, $\tilde{V}$, $\tilde{W}$ тех же подпространств). Ядро разложения Таккера вычисляется с помощью свертки вида (2.7), где вместо точных значений a_{ijk} используются их приближенные значения, представленные с помощью любого из полученных крестовых разложений. Например, пользуясь разложением для развертки по первому направлению, $\tilde{A}_r^{(1)} = U\Phi^T$, имеем

$$a_{ijk} \approx \tilde{a}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha}\varphi_{jk\alpha}.$$

Подставив это выражение в формулу свертки (2.7), получаем

$$\begin{aligned} g_{i'j'k'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r} u_{i\alpha}\varphi_{jk\alpha}\right)\tilde{u}_{ii'}\tilde{v}_{jj'}\tilde{w}_{kk'} = \\ &= \sum_{\alpha=1}^{r}(u_\alpha, \tilde{u}_{i'})\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\tilde{v}_{jj'}\tilde{w}_{kk'}\varphi_{jk\alpha}\right). \end{aligned} \tag{2.14}$$

Оценим сложность приведенного метода. Крестовый алгоритм, примененный к матрице размера $n \times n^2$, требует $\mathcal{O}(n^2 r)$ вычислений элемента массива $\mathcal{A}$ и $\mathcal{O}(n^2 r^2)$ операций. Оценка $\mathcal{O}(n^2)$ асимптотически является малой по сравнению с полным числом элементов массива $\mathcal{A}$, однако она все равно не является для нас удовлетворительной, поскольку уже при n порядка двух–трех тысяч такой алгоритм будет строить аппроксимацию очень медленно.

2.3.3. Как добиться почти линейной сложности. Мы намерены снизить оценку сложности метода до почти линейной по n. Для этого надо избавиться от вычисления векторов длины n^2, то есть оптимизировать каким-то образом вычисление строк матриц разверток (2.5). Вспомнив, что эти строки есть не что иное, как срезки массива $\mathcal{A}$, то есть векторизованные матрицы размера $n \times n$, применим и для их вычисления крестовый алгоритм. Поскольку массив $\mathcal{A}$ может быть по

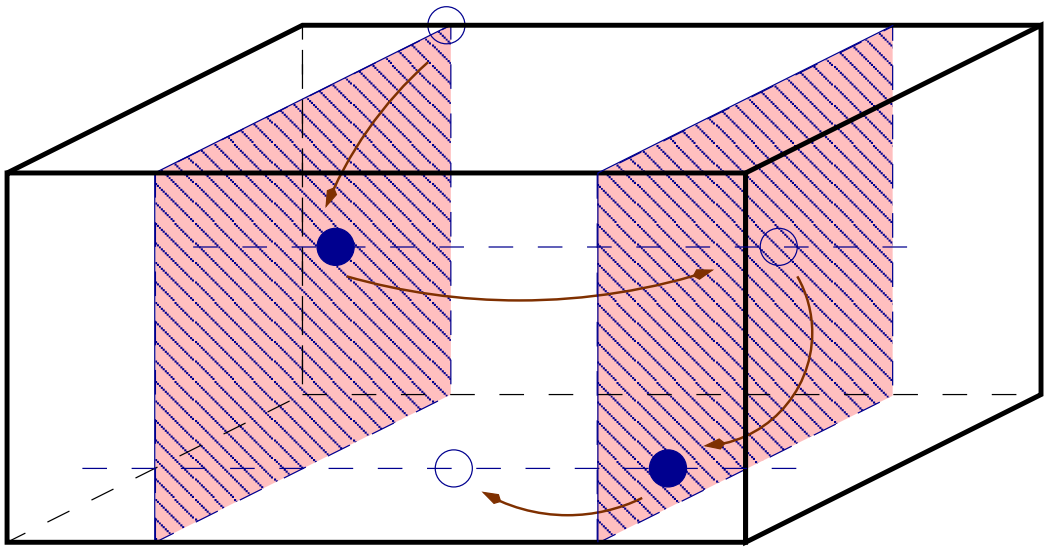


Рис. 2.2. Схема работы трехмерного крестового метода

предположению аппроксимирован с точностью ε трилинейным разложением, то каждая срезка $A_k = [(a_k)_{ij}]$ (тут мы рассматриваем множество элементов $[a_{ijk}]$ как семейство матриц A_k с индексами i, j) имеет аппроксимацию матрицей ранга r с точностью ε, она будет обнаружена при применении крестового метода. В дальнейшем мы будем хранить вычисленную «длинную» строку матрицы-развертки в малоранговом формате и работать с ней только через ее представление суммой скелетонов. Сам же массив $\mathcal{A}$ в ходе работы алгоритма будет аппроксимирован как трилинейное разложение вида (2.2), или сумма некоторого числа триплетов, число которых, однако, окажется порядка r^2.

Алгоритм 2 Пусть задан трехмерный массив $\mathcal{A}$ размера $n \times n \times n$ и требуется получить его аппроксимацию за число операций, почти линейное по n (то есть за $\mathcal{O}(nr^d)$ операций).

Выберем один из индексов i, j, k (назовем его *направляющим;* в приводимом ниже алгоритме таковым будет индекс k) и построим соответствующую ему матрицу-развертку размера $n \times n^2$. Для ее аппроксимации применим крестовый метод. Столбцы такой матрицы (короткие, то есть векторы длины n) вычисляются обычным порядком, для вычисления же строк матрицы, то есть длинных векторов, мы вновь применяем крестовый метод, получая их аппроксимации за почти линейное по n число вычислений.

0 Пусть p — номер текущего шага. На первом шаге ($p = 1$) в

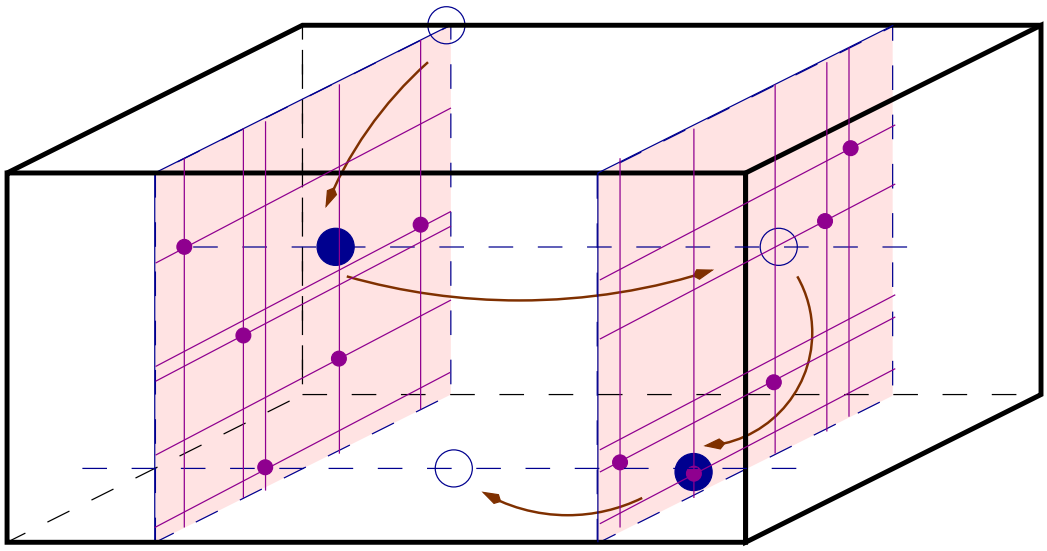


Рис. 2.3. Схема работы эффективного трехмерного крестового метода

массиве $\mathcal{A}$ выбирается некоторая срезка $A_{k_1} = [(a_{k_1})_{ij}]$. Устанавливается $\tilde{\mathcal{A}} = 0$.

1a Для вычисления срезки A_{k_p} как матрицы размером $n \times n$ применяется крестовый метод, причем в ходе его работы из вычисляемых элементов массива $\mathcal{A}$ вычитаются значения всех предыдущих триплетов в этих позициях. Результат сохраняется в виде

$$\tilde{A}_{k_p} = \sum_{q=1}^{r} u_q^{\{p\}} (v_q^{\{p\}})^T.$$

1b В полученной срезке находится наибольший по модулю элемент. Пусть его мультииндекс (i_p, j_p).

2 Вычисляется строка $w^{\{p\}}$, отвечающая мультииндексу (i_p, j_p), из нее также вычитаются значения всех предыдущих триплетов. Затем находится наибольший по модулю элемент в строке, причем элемент на позиции k_p вновь выбран быть не может. Пусть максимальный элемент стоит в позиции k_{p+1}. Вектор $w^{\{p\}}$ нормируется так, чтобы его элемент в позиции k_p равнялся единице:

$$w^{\{p\}} := w^{\{p\}} / w^{\{p\}}_{k_p}.$$

3 К уже вычисленной аппроксимации добавляется r триплетов

$$\tilde{A}_{k_p} \otimes w^{\{p\}} = \left(\sum_{q=1}^{r} u_q^{\{p\}} (v_q^{\{p\}})^T \right) \otimes w^{\{p\}} = \sum_{q=1}^{r} u_q^{\{p\}} \otimes v_q^{\{p\}} \otimes w^{\{p\}}.$$

Теперь значения получившейся аппроксимации $\tilde{A}$ совпадают с точностью до ε с точными значениями исходной матрицы на позициях p вычисленных срезок с номерами $k_1, \ldots, k_p$ и в p вычисленных коротких столбцах $(i_1, j_1), \ldots, (i_p, j_p)$.

4 Проверяется критерий остановки, и, если он не удовлетворен, устанавливается $p := p + 1$, и метод повторяется с шага 1.

В результате работы этого метода массив $\mathcal{A}$ аппроксимируется в виде $\tilde{\mathcal{A}} = [\tilde{a}_{ijk}]$, где

$$\tilde{a}_{ijk} = \sum_{p=1}^{r} \left(\sum_{q=1}^{r} u_{iq}^{\{p\}} v_{jq}^{\{p\}} \right) w_k^{\{p\}} = \sum_{\alpha=1}^{r^2} u_{i\alpha} v_{j\alpha} w_{k\alpha}, \tag{2.15}$$

то есть в виде трилинейного разложения ранга r^2, в котором все r^2 векторов u_α, v_α, вообще говоря, различные, а среди векторов w_α различными являются только r штук.

При реализации этого метода возникает необходимость найти способ быстро (то есть с асимптотикой, линейной по n) и достаточно точно решать следующие задачи:

- находить максимальный элемент в срезке, приближенной матрицей малого ранга (то есть выполнять шаг **1**b);
- оценивать точность вычисленной аппроксимации

  $$\|\mathcal{A} - \tilde{\mathcal{A}}\|_F \leqslant \varepsilon \|\mathcal{A}\|_F \approx \varepsilon \|\tilde{\mathcal{A}}\|_F,$$

  то есть проверять выполнение критерия остановки (шаг **4**).

Подход к решению первой задачи, основанный на поиске в вычисленной срезке подматрицы наибольшего (или близкого к наибольшему) объема, описан в нашей работе отдельно, в разделе 2.4.2. Он основан на алгоритме поиска подматриц максимального объема в факторах малорангового разложения, описаном в разделе 2.4.1. Его сложность составляет $\mathcal{O}(cnr)$ операций, где c — число внутренних итераций

метода. На всех рассмотренных нами примерах метод демонстрирует быструю сходимость и свою высокую надежность (определяемый с его помощью максимальный элемент отличается от истинного на величину не более 10%).

Проверка критерия точности аппроксимации происходит в нашем методе полностью аналогично двумерному случаю. Норма $\|\tilde{\mathcal{A}}\|_F$ вычисляется с помощью представления

$$\|\tilde{\mathcal{A}}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r^2} u_{i\alpha}v_{j\alpha}w_{k\alpha}\right)^2 = \\ = \sum_{\alpha=1}^{r^2}\sum_{\alpha'=1}^{r^2}(u_\alpha, u_{\alpha'})(v_\alpha, v_{\alpha'})(w_\alpha, w_{\alpha'}),$$

то есть для ее вычисления достаточно затратить $\mathcal{O}(nr^4)$ операций на подсчет скалярных произведений, и ее можно обновлять на каждом шаге за $\mathcal{O}(nr^3)$ операций. Оценка величины $\|\mathcal{A} - \tilde{\mathcal{A}}\|_F$ на p-м шаге производится по величине норм срезки A_{k_p} и вектора w_p, что приводит, например, к критерию вида

$$(n-p)\|w_p\|_2\|A_{k_p}\|_F < \varepsilon\|\tilde{A}\|_F.$$

Довольно большое количество дополнительных действий при работе алгоритма требуется для вычитания из получаемых элементов массива $\mathcal{A}$ значений уже полученного аппроксиманта. На p-м шаге на это необходимо затратить $\mathcal{O}(npr^2)$ действий, что приводит к суммарной оценке $\mathcal{O}(nr^4)$. Таким образом, общая сложность полученного метода составляет $\mathcal{O}(nr^2)$ вычислений элементов массива $\mathcal{A}$ и $\mathcal{O}(nr^4)$ дополнительных операций, то есть предложенный алгоритм является почти линейным по n.

Завершая обсуждение предложенного алгоритма, заметим, что хотя размер полученной трилинейной аппроксимации формально равен r^2, мы можем снизить его до r. Для этого можно, например, трижды применить алгоритм 2, установив последовательно в качестве направляющего индексы i, j и k. В первом случае мы получим трилинейную аппроксимацию размера r^2, в которой будет только r различных векторов u_p, во втором случае — аппроксимацию с r различными векторами v_p, в третьем — с r векторами w_p. Составив из этих векторов матрицы U, V, W размера $n\times r$ и найдя ортогональные базисы в полученных подпространствах, например, с помощью QR–разложения

$$U = \tilde{U}R_u, \qquad V = \tilde{V}R_v, \qquad W = \tilde{W}R_w,$$

используем $\tilde{U}, \tilde{V}, \tilde{W}$ как факторы разложения Таккера. Ядро разложения можно вычислить с помощью свертки (2.7), где элементы массива $\mathcal{A}$ представлены с помощью одного из имеющихся трилинейных разложений, выражающих a_{ijk} в виде (2.15). В результате имеем

$$\begin{aligned} g_{i'j'k'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r^2} u_{i\alpha}v_{j\alpha}w_{k\alpha}\right)\tilde{u}_{ii'}\tilde{v}_{jj'}\tilde{w}_{kk'} = \\ &= \sum_{\alpha=1}^{r^2}(u_\alpha, \tilde{u}_{i'})(v_\alpha, \tilde{v}_{j'})(w_\alpha, \tilde{w}_{k'}). \end{aligned} \tag{2.16}$$

Для вычисления использованных в формуле (2.16) скалярных произведений потребуется $\mathcal{O}(nr^3)$ операций. Полученное в результате разложение Таккера с ядром размера $r \times r \times r$ можно приводить к трилинейному, применяя к ядру $\mathcal{G}$ матричные алгоритмы поиска трилинейного разложения, ранг которого совпадает с размером массива.

2.3.4. Как сделать алгоритм эффективным. Для того, чтобы дополнительно снизить затраты на хранение факторов вычисляемого разложения, введем *единые* базисы U, V, W для хранения всех столбцов, строк и распорок и на каждом шаге при вычислении новых u_p, v_p, w_p будем раскладывать их по имеющимся базисам U, V, W, а остаток (если им нельзя пренебречь) использовать для расширения этих базисов. Итак, наше предложение состоит в том, чтобы аппроксимировать насчитанные срезки A_{k_p} в виде

$$A_{k_p} = UB_pV^T, \tag{2.17}$$

где матрицы U, V — ортогональные размера $n \times r$ (мы будем называть их *опорными*), а ядра B_p имеют размер $r \times r$. На хранение p посчитанных срезок в таком формате требуется $2nr + pr^2$ ячеек памяти, что асимптотически равно $\mathcal{O}(nr)$, то есть значительно меньше, чем для алгоритма 2. Базисы U, V, способные обеспечить выполнение равенства (2.17) одновременно для всех срезок, по предположению существуют — ими являются U, V факторы искомого трилинейного разложения, или же разложения Таккера. Само же построение такой одновременной редукции матриц–срезок A_{k_p} равносильно построению разложения Таккера для трехмерного массива

$$\mathcal{A}' = [A_{k_1} \dots A_{k_p}],$$

размер которого $n \times n \times p$. В самом деле, если удается представить

$$a_{ijk_p} = \sum_{i'=1}^{r} \sum_{j'=1}^{r} \sum_{p'=1}^{r} g_{i'j'p'} u_{ii'} v_{jj'} w_{pp'},$$

то обозначив

$$\sum_{p'=1}^{r} g_{i'j'p'} w_{pp'} = b_{i'j'p} = (b_p)_{i'j'},$$

немедленно получаем (2.17).

Алгоритм 3 *(Cross3D).* Пусть задан трехмерный массив $\mathcal{A}$ размера $n \times n \times n$ и требуется получить его аппроксимацию в виде разложения Таккера. Выберем один из индексов i, j, k (назовем его *направляющим;* в приводимом ниже алгоритме таковым будет индекс k) и построим соответствующую ему матрицу-развертку размера $n \times n^2$. Для ее аппроксимации применим крестовый метод. Столбцы такой матрицы вычисляются обычным порядком, для вычисления же строк, то есть «длинных» векторов, то есть срезок A_{k_p} мы применяем более эффективный прием. На каждом шаге для них вычисляется «нулевое приближение» на основе уже имеющихся опорных базисов и небольшой информации об элементах A_{k_p}. Далее значение срезки уточняется с помощью крестового метода, и соответствующим образом расширяются опорные базисы. Вся аппроксимация строится за почти линейное по n число вычислений.

Устанавливается $\tilde{\mathcal{A}} = 0$.

0 На первом шаге ($p = 1$) строится аппроксимация для некоторой срезки $A_k = [(a_k)_{ij}]$, скажем, первой ($k_1 = 1$).

0.1 Для вычисления срезки A_{k_1} как матрицы размером $n \times n$ применяется крестовый метод:

$$\tilde{A}_{k_1} = \sum_{q=1}^{r} u_q^{\{1\}} (v_q^{\{1\}})^T.$$

0.2 В полученной срезке находится наибольший по модулю (или близкий к нему) элемент. Пусть его мультииндекс (i_1, j_1).

0.3 Вычисляется «короткая» строка w_1, отвечающая мультииндексу (i_1, j_1).

0.4 Теперь

$$\tilde{A} = \left(\sum_{q=1}^{r} u_q^{\{1\}} (v_q^{\{1\}})^T \right) \otimes w_1 = \sum_{q=1}^{r} u_q^{\{1\}} \otimes v_q^{\{1\}} \otimes w_1.$$

Найдем ортогональные опорные матрицы U, V с помощью двух QR-разложений:

$$U_1 = [u_q^{\{1\}}] =: UR_u, \qquad V_1 = [v_q^{\{1\}}] =: VR_v.$$

Тогда $\mathcal{A}$ представляется в виде

$$\tilde{\mathcal{A}} = (UR_u R_v^T V^T) \otimes w_1 = (UB_1 V^T) \otimes (w_1 / \|w_1\|_2),$$

$$B_1 = R_u R_v^T \|w_1\|_2.$$

Далее отнормируем $w_1 := w_1/\|w_1\|_2$ (далее мы будем ортонормировать все векторы, составляющие опорные базисы U, V, W). В векторе w_1 вычислим максимальный элемент, пусть его индекс k_2.

0.5 Аппроксимант $\tilde{A}$ приведен к виду (2.17). Находим подматрицы максимального объема $U_\square$ и $V_\square$ в опорных матрицах U, V. Отметим, что они невырождены, иначе число столбцов в матрицах U и V можно было бы сократить, не изменив аппроксиманта.

1 Далее для каждой новой срезки аппроксимация строится по более эффективному алгоритму.

1.0 Вычисляем «нулевое приближение» для аппроксимации срезки A_{k_p} в виде

$$A_{k_p} \approx \tilde{A}^0_{k_p} = UB^0_p V^T.$$

Мы предполагаем, что уже имеющиеся опорные факторы U и V предоставляют достаточно хорошую аппроксимацию для столбцов и строк A и осталось найти только ядро B^0_p. Для нахождения B^0_p не требуется вычислять всю матрицу A_p, достаточно лишь ее подматрицы размера $r \times r$, в качестве которой мы возьмем подматрицу $A^\square_{k_p}$, стоящую на пересечении строк и столбцов, отвечающих подматрицам максимального объема в опорных факторах U и V. Выписав

$$A^\square_{k_p} = U_\square B^0_p V_\square,$$

обратим $U_\square$ и $V_\square$ и найдем B^0_p.

Оценка точности «нулевого приближения» дана в разделе 2.4.5. В практических вычислениях за счет этого шага можно значительно (нередко вплоть до нуля) уменьшить число строк и столбцов, вычисляемых крестовым методом на последующем этапе «уточнения» (шагах **1.1 — 1.4**).

1.1 С помощью крестового метода уточняем срезку A_{k_p}, аппроксимируя ее в виде

$$A_{k_p} \approx \tilde{A}_{k_p} = UB_p^0V^T + \sum_{q=1}^{r_p} u_q^{\{p\}}(v_q^{\{p\}})^T.$$

1.2 Текущие опорные матрицы U, V расширяются добавлением матриц $U_p = [u_q^{\{p\}}]$, $V_p = [v_q^{\{p\}}]$, $q = 1, \ldots, r_p$, и полученные новые наборы $[UU_p]$, $[VV_p]$ ортогонализуются

$$[UU_p] = [U\hat{U}_p]\begin{bmatrix} I & M_u \\ 0 & R_u \end{bmatrix}, \quad [VV_p] = [V\hat{V}_p]\begin{bmatrix} I & M_v \\ 0 & R_v \end{bmatrix},$$

$$U^T\hat{U}_p = 0 \qquad V^T\hat{V}_p = 0.$$

1.3 Вычисляется новое ядро B_p размером $(r + r_p) \times (r + r_p)$

$$B_p = \begin{bmatrix} B_p^0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} M_u \\ R_u \end{bmatrix}\left[M_v^T R_v^T\right] - \sum_{q=1}^{p-1} w_{k_p q}\begin{bmatrix} B_q & 0 \\ 0 & 0 \end{bmatrix}.$$

Теперь срезка

$$A_{k_p} = [U\hat{U}_p]B_p[V\hat{V}_p]^T$$

содержит элементы, равные разности значений массива $\mathcal{A}$ на этих позициях и значений полученной за $p - 1$ шаг аппроксимации $\tilde{\mathcal{A}}_{p-1}$. Остальные срезки представлены в новом базисе в виде

$$A_{k_q} = [U\hat{U}_p]\begin{bmatrix} B_q & 0 \\ 0 & 0 \end{bmatrix}[V\hat{V}_p]^T, \qquad q = 1, \ldots, p-1.$$

Таким образом, аппроксимант $\tilde{\mathcal{A}}_{p-1}$ представлен в виде

$$\tilde{\mathcal{A}}_{p-1} = \sum_{q=1}^{p-1}\left(U'B_q'V'^T\right) \otimes w_q,$$

$$U' = [U\hat{U}_p], \quad V' = [V\hat{V}_p], \qquad B'_q = \begin{bmatrix} B_q & 0 \\ 0 & 0 \end{bmatrix}, \quad q = 1, \ldots, p-1.$$

Положим также $B'_p = B_p$.

1.4 В матрицах U' и V' уточняется расположение подматрицы максимального объема (см. пункт 2.4.3).

1.5 В полученной срезке A_{k_p} отыскивается максимальный (или близкий к нему) элемент (см. пункт 2.4.2). Пусть он стоит на позиции (i_p, j_p).

2.1 Вычисляется распорка w_p, отвечающая мультииндексу (i_p, j_p), из нее вычитаются значения $\tilde{\mathcal{A}}_{p-1}$.

2.2 Вектор w_p ортогонализуется к уже имеющемуся набору векторов $W = [w_1, \ldots, w_{p-1}]$

$$w_p = \sum_{q=1}^{p-1} c_q w_q + \hat{w}_p, \qquad w_q^T \hat{w}_p = 0, \quad q = 1, \ldots, p-1.$$

Ядра старых срезок B'_q, $q = 1, \ldots p-1$, модифицируются

$$B''_q = B'_q + c_q B'_p, \qquad q = 1, \ldots, p-1,$$

вектор $\hat{w}_p$ нормируется

$$w_p := \hat{w}_p / \|\hat{w}_p\|_2, \qquad B''_p = \|\hat{w}_p\|_2 B'_p.$$

3 Аппроксимант $\tilde{\mathcal{A}}_p$ представлен в виде

$$\tilde{\mathcal{A}}_p = \sum_{q=1}^{p} \left(U' B''_q V'^T\right) \otimes w_q. \tag{2.18}$$

Он совпадает с точными значениями массива $\mathcal{A}$ на позициях p вычисленных срезок и p вычисленных векторов–распорок. Размеры матриц U' и V' равны $n \times (r + r_p)$, размер ядер B''_q равен $(r + r_p) \times (r + r_p)$. Чтобы вернуться к прежнему размеру опорных матриц и ядер, применим метод редукции Таккера.

3.1 Составляется трехмерный массив $\mathcal{B}'' = [B''_1 \ldots B''_p]$ размера $(r + r_p) \times (r + r_p) \times p$ и для него строится разложение Таккера (2.4) с помощью SVD-алгоритма, описанного на с. 34 формулами (2.5),

(2.6), (2.7). Пусть факторы найденного разложения $U_\clubsuit, V_\clubsuit, W_\clubsuit$, ядро $G_\clubsuit$, тогда

$$b''_{ijk} = \sum_{i'=1}^{r}\sum_{j'=1}^{r}\sum_{k'=1}^{r} g^{\clubsuit}_{i'j'k'} u^{\clubsuit}_{ii'} v^{\clubsuit}_{jj'} w^{\clubsuit}_{kk'}.$$

Обозначив

$$\sum_{k'=1}^{r} g^{\clubsuit}_{i'j'k'} w^{\clubsuit}_{kk'} = b^{\clubsuit}_{i'j'k} = (b^{\clubsuit}_{k})_{i'j'},$$

имеем

$$B''_k = U_\clubsuit B_{\clubsuit k} V_\clubsuit^{\mathsf{T}}, \qquad k = 1, \ldots, p. \tag{2.19}$$

где ортогональные матрицы $U_\clubsuit$ и $V_\clubsuit$ имеют размер $(r + r_p) \times r$, ядра $B_{\clubsuit k}$ имеют размер $r \times r$.

3.2 Подставив (2.19) в (2.18), получаем следующее представление для аппроксиманта:

$$\tilde{\mathcal{A}}_p = \sum_{k=1}^{p} \left((U'U_\clubsuit) B_{\clubsuit k} (V'V_\clubsuit)^{\mathsf{T}}\right) \otimes w_k = \sum_{k=1}^{p} \left(U B_k V^{\mathsf{T}}\right) \otimes w_k, \tag{2.20}$$

где матрицы $U = U'U_\clubsuit$, $V = V'V_\clubsuit$ ортогональны, а ядра $B_k \stackrel{\text{def}}{=} B_{\clubsuit k}$ имеют размер $r \times r$. Таким образом, формат (2.17) хранения срезок аппроксиманта восстановлен.

3.3 В новых опорных матрицах U и V уточняется положение подматрицы максимального объема.

4 Проверяется критерий остановки, и, если он не удовлетворен, устанавливается $p := p + 1$, и метод повторяется с шага **1.1**. Критерий остановки метода формулируется аналогично алгоритму 2 в виде

$$(n - p)\|w_p\|_2 \|A_{k_p}\|_F < \varepsilon \|\tilde{A}\|_F,$$

однако поскольку векторы w_p в нашем методе нормированы, а нормы срезок A_{k_p} равны норме их ядер B_p, он переписывается в более простом виде:

$$(n - p)\|B_{\clubsuit p}\|_F < \varepsilon \|\mathcal{B}_\clubsuit\|_F. \tag{2.21}$$

При реализации этого метода возникает необходимость найти способ быстро (то есть с асимптотикой, линейной по n и не сильно растущей с рангом) и достаточно точно решать следующие задачи:

- находить подматрицу максимального объема в матрице-факторе размера $n \times r$;
- пересчитывать ее при расширении матрицы-фактора r_p столбцами;
- пересчитывать ее при сужении матрицы-фактора;
- находить максимальный элемент в срезке, приближенной матрицей малого ранга.

Алгоритмам, эффективно решающим эти задачи, посвящен раздел 2.4.

2.3.5. Сложность полученного метода. Сначала определим сложность метода в случае, если для массива $\mathcal{A} = [a_{ijk}]$ существует точное разложение Таккера (2.4), с одинаковыми модовыми рангами (r, r, r). В этом случае ранг первой срезки, вычисляемой на шаге **0** алгоритма Cross3D, равен $r_1 \leqslant r$. Вообще говоря, это не означает, что для вычисления срезки будет достаточно найти r_1 столбец и r_1 строку матрицы-срезки A_{k_1} — эффективность здесь напрямую зависит от «качества» эвристического алгоритма, выбирающего новые вычисляемые столбцы, и конечно же, от структуры самой вычисляемой матрицы. Примером «хорошей» матрицы является, например, $A = [a_{ij}] = [i + j]$ — ее ранг равен двум и она восстанавливается по любым двум вычисленным строкам и столбцам, поскольку любая ее подматрица размера 2×2 невырождена. Примером «плохой» матрицы является матрица, содержащая только два ненулевых элемента: хотя ее ранг тоже не превосходит двух, подавляющее большинство вычисляемых столбцов — нулевые и не расширяют информацию об опорных базисах. Поэтому для определения ненулевых элементов скорее всего потребуется вычислить матрицу полностью. Опыт решения практических задач, возникающих в связи с решением интегральных уравнений, однако, свидетельствует о том, что хорошие матрицы встречаются чаще плохих. Тем не менее качество эвристического метода выбора следующего столбца остается наиболее принципиальным фактором для успешности применения алгоритма в целом. Качественный алгоритм должен предугадывать расположение плохо аппроксимированных элементов массива $\mathcal{A}$ и вычислять следующий столбец, строку так, чтобы они *не лежали полностью* в пространстве, заданном уже имеющимися базисами U, V, W. В противном случае вычисление n элементов массива $\mathcal{A}$ не приводит к расширению базисов, и тем самым как бы

оказывается проведено впустую. Мы будем называть такие вычисления *ложными*. Конечно, полностью избежать ложных вычислений невозможно, хотя бы потому что именно они являются сигналом для критерия остановки. Однако количество ложных вычислений должно быть по-возможности небольшим, не зависящем существенно от свойств массива и не растущим с n. Во всех дальнейших оценках сложности мы пренебрегаем наличием ложных вычислений, считая что их не более $\mathcal{O}(r)$.

Если при аппроксимации первой срезки размер накопленных базисов составил $r_1 = r$ векторов, то в дальнейшем «нулевое приближение» для всех срезок будет в точности совпадать с их значениями (см. теорему 4 в пункте 2.4.5), и вычислять дополнительные столбцы и строки массива $\mathcal{A}$ при аппроксимации новой срезки больше не потребуется. Если же $r_1 < r$, то какие-то срезки не могут быть аппроксимированы имеющимися базисами U и V, а значит вычислении этой срезки нулевое приближение к ней не совпадет со значениями ее элементов и базисы будут расширены. Однако число r_p вычисленных на этапе уточнения скелетонов не превосходит $r_p < r - r_1$. Рассуждая аналогично, видим, что суммарно в массиве $\mathcal{A}$ потребуется вычислить ровно rn строк, столбов и распорок. Через r шагов алгоритма размер базиса распорок также станет равен n, и аппроксимант $\tilde{\mathcal{A}}$ в точности совпадет со значениями массива $\mathcal{A}$.

Итоговая сложность метода в этом случае составит $\mathcal{O}(rn)$ вычислений элементов массива $\mathcal{A}$ и $\mathcal{O}(r^2 n)$ дополнительных действий.

Более детально обсудим сложность алгоритма Cross3D в случае, когда разложение Таккера для массива $\mathcal{A}$ существует с некоторой точностью ε. Как и прежде, вступительный шаг **0** требует порядка $\mathcal{O}(nr)$ вычислений элемента массива $\mathcal{A}$ и $\mathcal{O}(nr^2)$ дополнительных операций при определении срезки, еще $\mathcal{O}(nr^2)$ операций требуется на ортогонализацию полученных расширенных матриц, $\mathcal{O}(crn)$ операций на поиск максимального элемента (где c — число итераций в алгоритме поиска подматрицы максимального объема), число же операций, требующихся на вычисление ядра B_1 и нахождение по нему нормы аппроксиманта (пользуясь тем, что $\|\tilde{\mathcal{A}}_1\|_F = \|B_1\|_F$), не зависит от n, то есть является пренебрежимо малым.

Далее, на каждом шаге метода (рассмотрим шаг с номером p) требуется затратить следующее число действий:

1.0 «Нулевое приближение» для срезки A_{k_p} строится на основе r^2 вычислений элементов массива $\mathcal{A}$ и требует $\mathcal{O}(r^3)$ дополнительных действий. Это число не зависит от n и поэтому пренебре-

жимо мало.

1.1 Уточнение срезки A_{k_p} требует $\mathcal{O}(r_p n)$ вычислений элемента массива $\mathcal{A}$ и $\mathcal{O}(r_p(r+r_p)n)$ дополнительных операций. Мы полагаем, что в большинстве случаев $r_p \ll r$; в практических расчетах для уточнения срезки часто достаточно вычислить один-два (или даже ни одного) дополнительных вектора. Однако мы не имеем гарантированной верхней оценки на значение r_p кроме очевидной $r_p \leqslant r$.

1.2 Ортогонализация расширенной матрицы требует $\mathcal{O}((rr_p + r_p^2)n)$ действий, методы по повышению эффективности ее реализации описаны в пункте 2.4.3.

1.3 Вычисление нового ядра B_p имеет сложность $\mathcal{O}(r^2 p)$, то есть не зависящую от n.

1.4 Пересчет подматриц максимального объема требует $\mathcal{O}(nrr_p(r + r_p)\log(r + r_p))$ дополнительных действий (эта оценка получена в разделе 2.4.3). После этого максимальный элемент в новой срезке определяется за $\mathcal{O}((r + r_p)^3)$ операций (алгоритм описан в разделе 2.4.2). Для еще более быстрой реализации этого шага можно предварительно «дожать» срезку, снизив число векторов в ее малоранговом разложении с $r+r_p$ до r (или до еще меньшего значения с привнесением дополнительной погрешности). Способ вычисления такого быстрого дожимания и замечания по его реализации предложены при описании мозаично-скелетонного метода в пункте 1.1.2.

2.1 Чтобы вычислить распорку w_p, требуется вычислить $\mathcal{O}(n)$ элементов массива $\mathcal{A}$ и произвести $\mathcal{O}(prn)$ дополнительных действий для вычитания из него значений уже имеющегося аппроксиманта.

2.2 Ортогонализация вектора w_p к W требует $\mathcal{O}(rn)$ действий, пересчет ядер B_q выполняется за время, не зависящее от n.

3.1 Вычисление разложение Таккера для массива $\mathcal{B}''$ на основе трех SVD-разложений требует $\mathcal{O}(r^4)$ действий.

3.2 Получение новых опорных матриц U, V требует $\mathcal{O}(r^2 n)$ действий.

4 Норма аппроксиманта массива $\tilde{A}$ в точности равна норме ядра разложения Таккера $\mathcal{B}_{\clubsuit} = [B_{\clubsuit_1}, \ldots, B_{\clubsuit_p}]$, поэтому сложность ее вычисления не зависит от n.

Таким образом, на каждом шаге наш метод требует $\mathcal{O}(rn)$ вычислений массива $\mathcal{A}$ и $\mathcal{O}(r^2 n)$ дополнительных действий (здесь мы полагаем, что влиянием r_p и $\log r$ можно пренебречь, считая их ограниченными константой), полностью же сложность алгоритма составляет $\mathcal{O}(r^a n), 1 \leqslant a \leqslant 2$, вычислений элемента $\mathcal{A}$ и $\mathcal{O}(r^3 n)$ дополнительных действий.

2.4. Важные детали

Автор благодарит Н. Л. Замарашкина за советы и ценные обсуждения результатов этого раздела.

Без эффективной реализации перечисленных в этой главе процедур алгоритм трехмерной крестовой аппроксимации не мог бы называться эффективным.

2.4.1. Как найти подматрицу максимального объема. В ходе работы алгоритма Cross3D возникает необходимость находить в матрице $n \times r$ подматрицу максимального объема (максимального модуля детерминанта) размера $r \times r$. Решение этой задачи за разумное время представляется весьма затруднительным, и нам не известны алгоритмы, решающие задачу в такой постановке. Поэтому в нашей работе вместо подматрицы максимального объема мы используем подматрицу, которую мы будем называть *доминирующей*.

Определение. Пусть матрица A имеет размер $n \times r$, $n > r$. Будем называть ее подматрицу $A_\square$ размера $r \times r$ *доминирующей*, если все элементы матрицы $AA_\square^{-1}$ не превосходят по модулю единицы.

Алгоритм поиска доминирующей подматрицы предложен в работе [53]. Для удобства читателя приведем его здесь.

Алгоритм 4 *(maxvol)* Требуется найти доминирующую подматрицу $A_\square$ размера $r \times r$ в матрице A размера $n \times r$.

0 Пусть $A_\square$ — ведущая подматрица. В начале работы алгоритма в качестве $A_\square$ выберем любую невырожденную $r \times r$ подматрицу A и переставим строки матрицы так, чтобы $A_\square$ разместилась в первых r строках.

1 Вычислим

$$AA_{\square}^{-1} = \begin{bmatrix} I_{r\times r} \\ Z \end{bmatrix} = B.$$

2 Найдем максимальный по модулю элемент $|z_{ij}|$ в подматрице Z.

3 **Если** $\gamma = |z_{ij}| > 1$, **то**

переставим в B строки $r+i$ и j. Верхняя подматрица в B после перестановки имеет вид

$$\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ * & * & \gamma & * & * & \\ & & & \ddots & & \\ & & & & 1 & \end{pmatrix}$$

и ее определитель равен $\gamma \geqslant 1+\varepsilon$, то есть в результате перестановки строк он увеличился. Значит, и определитель верхней подматрицы в A при такой перестановке увеличился. Обозначим теперь через $A_{\square}$ новую подматрицу в первых r строках матрицы A и вернемся на шаг **1**.

Если же $\gamma \leqslant 1$, то завершим алгоритм.

На практике, чтобы избежать слишком долгого перебора, не приводящего к существенному увеличению объема ведущей подматрицы $A_{\square}$, критерий в шаге **3** алгоритма проверяется в ослабленном виде $|z_{ij}| > 1+\delta$, где δ — небольшой параметр.

Для вычисления произведения $AA_{\square}^{-1}$ на шаге **1** не нужно каждый раз обращать новую матрицу $A_{\square}$. Достаточно заметить, что на каждом шаге с ней совершается элементарное преобразование ранга один, а именно, если максимальный элемент в подматрице Z был найден на позиции i^*, j^*, то в $A_{\square}$ модифицируется строка с номером j^*:

$$A_{\square} := A_{\square} + uv^{T},$$

$$u_i = e_{ij_*}, \quad v_j = a_{i^*j} - e_{j^*j}, \qquad i,j = 1,\ldots,r, \tag{2.22}$$

$$e_{pq} = \begin{cases} 1, & p = q, \\ 0 & p \neq q. \end{cases}.$$

Обратная матрица в этом случае тоже модифицируется преобразованием ранга один (формула Шермана-Вудбери-Моррисона):

$$A_{\square}^{-1} := A_{\square}^{-1} - A_{\square}^{-1}u\left(1 + v^{T}A_{\square}^{-1}u\right)^{-1}v^{T}A_{\square}^{-1},$$

а значит, подматрица B также модифицируется

$$B := B + Bu\left(1 + v^T A_\square^{-1} u\right)^{-1} v^T A_\square^{-1}.$$

С учетом вида u, v (2.22) получаем формулу преобразования для нижней подматрицы Z

$$Z := Z + a\,(1+\gamma)^{-1}\, b^T,$$

$$a_i = z_{ij^*}, \quad b_j = z_{i^*j} - e_{j^*j}, \qquad i = 1, \ldots, n-r, \quad j = 1, \ldots, r.$$

Таким образом, пересчет матрицы B на шаге **1** алгоритма есть простое преобразование ранга один, которое выполняется за nr операций. Поиск наибольшего элемента в подматрице Z (шаг **2**) также требует nr операций, таким образом сложность итерационной части алгоритма составляет $2cnr$, где c — необходимое число итераций метода. Чтобы оценить значение c, заметим, что на каждой итерации объем ведущей подматрицы увеличивается не менее чем в γ раз, причем $\gamma \geqslant 1 + \delta$. Таким образом через k шагов алгоритма

$$|\det A_\square^{\{k\}}| \geqslant |\det A_\square^{\{0\}}|(1+\delta)^k,$$

где $A_\square^{\{k\}}$ — ведущая подматрица на k-том шаге. Отсюда получаем оценку на число итераций

$$c \log(1+\delta) \leqslant \log \frac{|\det A_\square^{\max}|}{|\det A_\square^{\{0\}}|}, \tag{2.23}$$

где $A_\square^{\max}$ — подматрица максимального объема в A.

Приведенная оценка показывает, что выбор хорошей ведущей подматрицы на нулевом шаге алгоритма может существенно снизить количество необходимых итераций.

Замечание. Насколько же существенно для нашего метода отличие подматрицы максимального объема от доминирующей подматрицы?

Если $A_\square$ — доминирующая подматрица, то есть все элементы в нижней части подматрицы B не превосходят по модулю единицы, то пользуясь неравенством Адамара [48], можно установить, что объем любой подматрицы размера $r \times r$ не превосходит произведения норм ее строк, то есть величины $r^{r/2}$. Эта оценка достижима, например, если на текущем шаге матрица $AA_\square^{-1}$ имеет вид

$$B = \begin{bmatrix} I_{r\times r} \\ F_r \\ * \end{bmatrix},$$

где F_r — матрица Фурье, все элементы которой по модулю равны единице, а объем которой равен $r^{r/2}$ (в действительной арифметике примером могут являться например матрицы Уолша или Адамара) [46]. В этом случае объем найденной подматрицы в точности в $r^{r/2}$ раз меньше объема подматрицы, расположенной в строках F_r.

Итак, объем доминирующей подматрицы может отличаться от максимально возможного на коэффициент $r^{r/2}$. Однако в большинстве случаев оценки, сформулированные для подматрицы наибольшего объема, можно получить и для доминирующей подматрицы. Например, если при аппроксимации матрицы крестовым методом нам удалось выбрать «правильные» столбцы (то есть столбцы, которые содержат подматрицу максимального объема), то строки в них могут быть найдены с использованием алгоритма 4, и оценка точности аппроксимации, приведенная в работе [13], не изменится, так как по существу в доказательстве используется только свойство доминирования. По этой причине мы считаем допустимым смешивать термины «подматрица максимального объема» и «доминирующая подматрица», и оговаривать разницу, только когда различие действительно принципиально.

2.4.2. Как найти наибольший элемент в срезке. В алгоритме трехмерного крестового разложения существенную важность представляет процедура поиска максимального по модулю (или близкого к нему) элемента в срезке, то есть в матрице A ранга r и размера $n \times n$, представленной в виде скелетонного разложения $A = UV^T$, где U, V — матрицы $n \times r$. Решая эту задачу простым сравнением всех элементов матрицы A, мы затратим $\mathcal{O}(rn^2)$ действий, что противоречит желанию построить алгоритм почти линейной сложности. Значит требуется каким-то образом указать *подмножество элементов* матрицы A, число которых мало по сравнению с n^2 и среди которых находится максимальный или достаточно большой по модулю элемент A. В качестве такого подмножества мы будем использовать подматрицу максимального объема. Обоснованием нашего выбора является следующая

Теорема 2 Пусть B — подматрица максимального объема среди всех таких подматриц размера $r \times r$ в матрице A, тогда

$$\operatorname{maxel}(B) \geqslant \operatorname{maxel}(A)/(2r^2 + r), \tag{2.24}$$

где $\operatorname{maxel}(A)$ — максимальное значение модуля элемента матрицы A. Если при этом матрица A имеет ранг r, то

$$\operatorname{maxel}(B) \geqslant \operatorname{maxel}(A)/r^2. \tag{2.25}$$

Доказательство. Если $\mathrm{maxel}(A)$ находится в подматрице B, то утверждения теоремы тривиальны. Поэтому достаточно рассмотреть случай, когда $\mathrm{maxel}(A)$ лежит в окаймлении подматрицы B. Рассмотрим подматрицу A размера $(k+1)\times(k+1)$, содержащую этот максимальный элемент.

$$\hat{A} = \begin{bmatrix} B & c \\ d^T & b \end{bmatrix}.$$

Элементы векторов c и d оцениваются как

$$\mathrm{maxel}(c) \leqslant r\,\mathrm{maxel}(B), \qquad \mathrm{maxel}(d) \leqslant r\,\mathrm{maxel}(B).$$

В самом деле,

$$c = B(B^{-1}c) = B\tilde{c},$$

и так как B — подматрица наибольшего объема, то все элементы вектора $\tilde{c}$ не превосходят по модулю единицы. Отсюда очевидно следует

$$\mathrm{maxel}(c) \leqslant \max_{i=1,\ldots,r} \left| \sum_{j=1}^{r} b_{ij}\tilde{c}_j \right| \leqslant \mathrm{maxel}(B)r.$$

Для элементов вектора d доказательство проводится аналогично.

Осталось оценить значение модуля элемента b. В работе [13] показано, что в условиях теоремы

$$\left|b - d^T B^{-1} c\right| \leqslant (r+1)\sigma_{r+1}(\hat{A}), \tag{2.26}$$

где $\sigma_1(\hat{A}) \geqslant \sigma_2(\hat{A}) \geqslant \ldots \geqslant \sigma_{r+1}(\hat{A})$ — сингулярные числа $\hat{A}$. Отсюда

$$|b| \leqslant (r+1)\sigma_{r+1}(\hat{A}) + |d^T B^{-1} c| \leqslant (r+1)\sigma_{r+1}(\hat{A}) + |d^T \tilde{c}|$$

Так как все компоненты $\tilde{c}$ не превосходят по модулю единицы, имеем

$$|b| \leqslant (r+1)\sigma_{r+1}(\hat{A}) + \mathrm{maxel}(d)r \leqslant (r+1)\sigma_{r+1}(\hat{A}) + \mathrm{maxel}(B)r^2. \tag{2.27}$$

Если матрица A имеет ранг r, то $\sigma_{r+1}(A) = \sigma_{r+1}(\hat{A}) = 0$, а значит оценка (2.25) получена. Для матрицы произвольного ранга необходимо еще оценить $\sigma_{r+1}(A)$ через значения ее элементов.

Для этого заметим, что

$$\hat{A}^T\hat{A} = \begin{bmatrix} B^T & d \\ c^T & b \end{bmatrix} \begin{bmatrix} B & c \\ d^T & b \end{bmatrix} = \begin{bmatrix} B^TB + dd^T & B^Tc + bd \\ c^TB + bd^T & c^Tc + b^2 \end{bmatrix}.$$

Отсюда по теореме о перемежении

$$\sigma_r(B^TB + dd^T) \geqslant \sigma^2_{r+1}(\hat{A}),$$

и в силу той же теоремы при $r > 1$

$$\sigma_{r-1}(B^T B) \geqslant \sigma_r(B^T B + dd^T) \geqslant \sigma^2_{r+1}(\hat{A}),$$

и окончательно

$$\sigma_1(B) \geqslant \sigma_{r+1}(A).$$

Для максимального по модулю элемента справедливо

$$\operatorname{maxel}(B) \geqslant \sigma_1(B)/r,$$

что позволяет записать оценку (2.27) в виде

$$|b| \leqslant (r+1)r \operatorname{maxel}(B) + r^2 \operatorname{maxel}(B) = (2r^2 + r) \operatorname{maxel}(B),$$

откуда сразу следует (2.24). □

Таким образом, максимальный элемент в подматрице максимального объема не сильно отличается от максимального элемента всей матрицы A.

Замечание. В доказательстве данной теоремы для случая матрицы ранга r по существу используется только *доминирование* подматрицы B в строках и столбцах матрицы A.

Гипотеза. Мы полагаем, что в условиях теоремы справедлива и более сильная оценка

$$\operatorname{maxel}(B) \geqslant \operatorname{maxel}(A)/r.$$

Подматрица $r \times r$ максимального объема в A лежит на пересечении строк $i_1, \ldots, i_r$, на которых расположена подматрица максимального объема в U, и столбцов $j_1, \ldots, j_r$, на которых расположена подматрица максимального объема в V^T. Расположение этих подматриц может быть определено с помощью алгоритма 4 и уточняться на каждом шаге, как это показано в пункте 2.4.3. Вычисление всех элементов указанной подматрицы A определение среди них максимального требует $\mathcal{O}(r^3)$ действий. Этот элемент совпадает или весьма близок к максимальному элементу всей матрицы, то есть может быть использован в качестве хорошего ведущего элемента для крестового метода.

Заметим, что мы можем снизить ранг скелетонного разложения рассматриваемой матрицы A, и следовательно, затраты на поиск максимального элемента, если воспользуемся алгоритмом «дожимания» , описанным в разделе 1.1.2. При этом мы внесем в элементы матрицы дополнительную погрешность, но если предполагать, что ее величина

примерно одинакова на всех элементах, то относительное изменение в искомом наибольшем по модулю элементе будет минимальным. Практические расчеты показывают, что использование «дожимателя» с мягким критерием $\tilde{\varepsilon}$ позволяет значительно снизить ранг аппроксимации и тем самым ускорить процедуру поиска максимального элемента, так, что затраты на само дожимание оказываются несущественными.

2.4.3. Как добавить векторы в ортогональный набор. Обсудим эффективную реализацию шага **1.2** алгоритма 3. На этом шаге требуется ортогонализовать r_p полученных новых векторов U_p к r ортогональным векторам U, то есть вычислить QR-разложение

$$[UU_p] = [UQ] \begin{bmatrix} I & M \\ & R \end{bmatrix}, \qquad U^TQ = 0.$$

Для второго блочного столбца матричного равенства мы получаем $U_p = UM + QR$, откуда, используя $U^TQ = 0$, заключаем

$$M = U^TU_p, \qquad QR = U_p - UU^TU_p = V.$$

Итак, для ортогонализации U_p к пространству U мы используем блочную версию алгоритма Грама-Шмидта, а для ортогонализации векторов V друг к другу и вычисления матриц Q, R можно применить стандартный алгоритм QR-разложения. Известно, что при использовании метода Грама-Шмидта в случае, когда вектор u_p в значительной степени лежит в пространстве U, реальная ортогональность полученного v к U в условиях машинной арифметики может значительно нарушаться, поскольку число достоверных знаков в разности $v = (I - UU^T)u$ может оказаться весьма малым. Обычно в таком случае рекомендуют сравнивать нормы u и v и, если норма v упала достаточно сильно по сравнению с нормой u, производить *реортогонализацию,* то есть повторную ортогонализацию v к U. Термин «достаточно сильно» можно понимать в широких пределах, в практических случаях обычно проверяют критерий $\|v\|_2 \leqslant \frac{1}{2}\|u\|_2$. Описанный метод («двойной цены достаточно») и анализ принадлежат Кахану, детальное его описание можно найти в [63].

Для наших целей мы используем блочную версию этого алгоритма.

0 Матрица U состоит из r ортонормированных столбцов, требуется ортогонализовать к ним r_p столбцов матрицы U_p.

1 $M = U^TU_p, \quad V = U_p - UM.$

2 если $\|v_j\|_2 < \frac{1}{2}\|u_j\|_2$ для какого-то $j = 1, \dots r$, **найти**

$$S = U^T V, \qquad M := M + S, \quad V := V - US.$$

3 Вычислить QR-разложение $V =: QR$.

Применение реортогонализации является для нашей задачи абсолютно необходимым, поскольку получаемые при расчете новой срезки векторы u_p с каждым шагом работы алгоритма 3 все в большей степени попадают в пространство, заданное столбцами опорной матрицы U. Использование приведенного алгоритма позволяет ускорить процедуру ортогонализации примерно в 4 раза по сравнению с применением стандартного QR-разложения к матрице $[UU_p]$. Асимптотическая сложность этого шага составляет $\mathcal{O}(rr_p n + r_p^2 n)$ операций.

При добавлении векторов в ортогональный набор мы должны также решить еще одну не менее важную задачу: пересчитать положение подматрицы наибольшего объема в расширенном базисе. Эта информация используется затем при нахождении максимального элемента в срезке (шаг **1.5** алгоритма Cross3D). Итак, пусть в матрице U известно расположение доминирующей подматрицы $U_\square$, и мы расширяем ее добавлением линейно независимых столбцов U_p. Не ограничивая общности, считаем, что подматрица $U_\square$ располагается в верхних строках матрицы U, и запишем для расширенной подматрицы блочное LU-разложение

$$\begin{bmatrix} U & U_p \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} U_\square & C \\ U_{\text{off}} & D \end{bmatrix} =: \begin{bmatrix} I & \\ U_{\text{off}} U_\square^{-1} & \hat{D} \end{bmatrix} \begin{bmatrix} U_\square & C \\ & I \end{bmatrix}, \tag{2.28}$$

где $\hat{D} = D - U_{\text{off}} U_\square^{-1} C$. В качестве ведущей подматрицы в $[UU_p]$ возьмем подматрицу размера $(r + r_p) \times (r + r_p)$, первые r строк которой совпадают с положением доминирующей подматрицы $U_\square$, а оставшиеся r_p строк — с положением доминирующей подматрицы $\hat{D}_\square$ в $\hat{D}$. Будем называть подматрицу, выбранную таким образом, *надстройкой* подматрицы $U_\square$.

Если расположение $U_\square$ известно, то для определения расположения надстройки нужно затратить nrr_p операций для вычисления $\hat{D}$ и cnr_p операций для определения $\hat{D}_\square$, где c — количество итераций алгоритма 4. Поскольку обычно $r_p \ll r$, эти затраты имеют тот же порядок, что и затраты на ортогонализацию U_p к U. Целью дальнейшего будет показать, надстройка обладает свойствами, близкими к свойствам подматрицы максимального объема в $[UU_p]$, и использование ее в качестве ведущей подматрицы приведет к быстрой сходимости алгоритма 4 для расширенной матрицы.

Теорема 3 Пусть в матрице U известно положение доминирующей подматрицы $U_\square$, и на ее основе в расширенной матрице $\mathbf{U} = [U U_p]$ построена подматрица-надстройка $\mathbf{U}_\boxdot$ размера $(r + r_p) \times (r + r_p)$, как это показано выше. Тогда

1. все элементы в матрице $\mathbf{U}\mathbf{U}_\boxdot^{-1}$ по модулю не превосходят $r_p + 1$,
2. объем подматрицы $\mathbf{U}_\boxdot$ связан с максимально возможным среди всех подматриц такого размера неравенством

$$|\det \mathbf{U}_\boxdot| \geqslant |\det \mathbf{U}_{\max}| / (r + r_p)^{(r+r_p)/2}. \tag{2.29}$$

Доказательство. Не ограничивая общности, можно считать, что доминирующая подматрица $U_\square$ находится в первых r строках матрицы U, а ее надстройка получается за счет добавления следующих r_p строк матрицы $\mathbf{U}$. Рассмотрим блочное представление

$$Q = \begin{bmatrix} Q_\square \\ Q_{\text{off}} \end{bmatrix}, \qquad Q' = QX = \begin{bmatrix} Q_\square X \\ Q_{\text{off}} X \end{bmatrix} = \begin{bmatrix} Q'_\square \\ Q_{\text{off}} \end{bmatrix},$$

и отметим, что домножая матрицу Q справа на любую невырожденную X, мы не меняем соотношения между определителями любых двух подматриц размера $r \times r$ в Q (они одинаково домножаются на $\det X$). Кроме того, $Q'_{\text{off}}(Q'_\square)^{-1} = Q_{\text{off}} Q_\square^{-1}$, что позволяет домножать матрицу справа на невырожденные, не меняя интересующих нас оценок.

Таким образом, совершив разложение (2.28), мы можем перейти к рассмотрению только матриц вида

$$\begin{bmatrix} I & \\ C & D \end{bmatrix},$$

причем так как $U_\square$ — доминирующая подматрица, то все элементы C не превосходят по модулю единицы. Для таких матриц дополнение Шура $\hat{D} = D$, а значит $\hat{D}_\square = D_\square$. Выделив явно расположение $D_\square$ в D, запишем рассматриваемую матрицу в виде

$$\begin{bmatrix} I & \\ C_1 & D_\square \\ C_2 & D_{\text{off}} \end{bmatrix} = \begin{bmatrix} I & \\ C_1 & I \\ C_2 & D_{\text{off}} D_\square^{-1} \end{bmatrix} \begin{bmatrix} I & \\ & D_\square \end{bmatrix}.$$

Итак, мы свели задачу к рассмотрению матриц вида

$$A = \begin{bmatrix} I & \\ C_1 & I \\ C_2 & D \end{bmatrix},$$

где все элементы C_1, C_2 и D не превосходят по модулю единицы. Определитель выбранной нами матрицы-надстройки

$$\det A_{\boxdot} = \det \begin{bmatrix} I & \\ C_1 & I \end{bmatrix} = 1.$$

При этом согласно теореме Адамара определитель любой другой подматрицы размера $(r + r_p) \times (r + r_p)$ в A не превосходит произведения норм строк, то есть величины $(r + r_p)^{(r+r_p)/2}$. Тем самым доказано второе утверждение теоремы. Теперь запишем

$$AA_{\boxdot}^{-1} = \begin{bmatrix} I & \\ C_1 & I \\ C_2 & D \end{bmatrix} \begin{bmatrix} I & \\ -C_1 & I \end{bmatrix} = \begin{bmatrix} I & \\ & I \\ C_2 - DC_1 & D \end{bmatrix}.$$

Поскольку элементы матрицы D размера $(n - r - r_p) \times r_p$ и элементы матрицы C_1 размера $r_p \times r$ не превосходят по модулю единицы, то элементы DC_1 не превосходят по модулю r_p, откуда сразу следует первое утверждение теоремы. $\square$

Замечание. Для надстройки $\mathbf{U}_{\boxdot}$ оценка во втором утверждении теоремы 3 имеет тот же вид, что и для доминирующей матрицы $\mathbf{U}_{\square}$.

Следствие. Подстановкой неравенства (2.29) в оценку (2.23) можно показать, что если алгоритм поиска доминирующей подматрицы (алгоритм 4) в качестве ведущей подматрицы на нулевом шаге использует подматрицу-надстройку $\mathbf{U}_{\boxdot}$, то число его итераций не превосходит

$$c = \frac{r + r_p}{2} \log(r + r_p) \log^{-1}(1 + \delta).$$

С учетом этой оценки сложность пересчета доминирующей подматрицы на каждом шаге алгоритма Cross3D составляет

$$\mathcal{O}(nrr_p(r + r_p) \log(r + r_p)).$$

2.4.4. Как проверять точность аппроксимации. Как бы ни был хорош критерий остановки (2.21), он не может гарантировать, что полученный аппроксимант $\tilde{\mathcal{A}}$ действительно близок к исходному массиву $\mathcal{A}$, как и любой другой критерий, не вычисляющий всех элементов разности. Чтобы повысить надежность нашего метода, после срабатывания критерия остановки мы дополнительно вычисляем $\mathcal{O}(n)$ случайно выбранных элементов массива $\mathcal{A}$ и проверяем на них точность аппроксимации. В случае, если на этой выборке индексов $p = (i_p, j_p, k_p)$, $p = 1, \ldots, n$, выполняется

$$\|a_p - \tilde{a}_p\|_2 \leqslant \|a_p\|_2,$$

аппроксимация признается в самом деле точной, если же приведенный критерий не выполняется, номер шага увеличивается на 1, и алгоритм 3 выполняется снова с пункта **1.1**, причем номер новой вычисляемой срезки k_p устанавливается равным индексу k_p того элемента из множества проверенных, ошибка в приближении которого оказалась максимальна.

2.4.5. Какова точность «нулевого приближения». В этом разделе мы обсудим шаг построения «нулевого приближения» для вычисляемой срезки, то есть шаг **1.0** алгоритма Cross3D. Его эффективность основана на том предположении, что опорные базисы, заданные матрицами U и V, уже после нескольких шагов алгоритма становятся достаточно близки к искомым, и в дальнейшем уточнению подлежит в основном фактор W и ядро $\mathcal{G}$. Это факт наблюдался нами на всех рассмотренных тестовых тензорах $\mathcal{A}$ и вообще говоря естественен, учитывая «несимметричность» метода относительно индексов массива: уже на нулевом шаге при построении аппроксимации для первой рассматриваемой срезки A_{k_1} требуется вычислить r столбцов и r строк массива $\mathcal{A}$, но только одну распорку w_1. За счет этого накопление информации об опорных базисах U и V происходит существенно быстрее.

Используя эту «избыточную» информацию, мы на следующих шагах алгоритма можем сократить время и количество элементов массива, требуемое для аппроксимации вычисляемых срезок A_{k_p}. Для этого мы рассматриваем доминирующие подматрицы $U_\square$ и $V_\square$ в факторах U и V и находим в A_{k_p} подматрицу $A^\square_{k_p}$ на пересечении строк, отвечающих положению $U_\square$ и столбцов, задающих положение $V_\square$. По имеющимся данным срезка A_{k_p} оценивается как

$$A^0_{k_p} = UB_0V^T, \qquad B_0 = U_\square^{-1}A^\square_{k_p}V_\square^{-1}. \tag{2.30}$$

Точность этой оценки позволяет установить следующая теорема.

Теорема 4 Пусть для матрицы A размера $n_1 \times n_2$ существует малоранговое приближение вида

$$A = UBV^T + E, \qquad \|E\|_F = \varepsilon,$$

где унитарные факторы U и V имеют размеры $n_1 \times r_1$ и $n_2 \times r_2$ соответственно. Тогда точность аппроксимация с ядром B_0, полученным согласно формуле (2.30), оценивается как

$$\|A - UB_0V^T\|_F \leqslant \sqrt{n_1n_2r_1r_2 + 1}\,\varepsilon.$$

Доказательство. Рассмотрев в матрице $A = UBV^T + E$ строки и столбцы, отвечающие положению доминирующих подматриц $U_\square$ и $V_\square$, получим

$$A_\square = U_\square B V_\square + E_\square,$$

где подматрицы $A_\square$ и $E_\square$ лежат на пересечении этого креста. Так как B_0 определяется из равенства

$$A_\square = U_\square B_0 V_\square,$$

справедливо

$$U_\square (B - B_0) V_\square + E_\square = 0,$$

то есть

$$\|B - B_0\|_F \leqslant \|U_\square^{-1}\|_F \|E_\square\|_F \|V_\square^{-1}\|_F.$$

В силу унитарности факторов U и V справедливо

$$\|U_\square^{-1}\|_F = \|UU_\square^{-1}\|_F, \qquad \|V_\square^{-1}\|_F = \|VV_\square^{-1}\|_F.$$

По определению доминирующих подматриц, все элементы $UU_\square^{-1}$ и $VV_\square^{-1}$ не превосходят по модулю единицы. Отсюда

$$\|B - B_0\|_F \leqslant \sqrt{n_1 r_1}\sqrt{n_2 r_2}\|E_\square\|.$$

Оценивая норму разности $\|A - UB_0V^T\|$, рассмотрим отдельно элементы подматрицы $A_\square$ и все остальные, которые мы обозначим A_{off}. Естественно,

$$\|A - UB_0V^T\|_F^2 = \|(A - UB_0V^T)_\square\|_F^2 + \|(A - UB_0V^T)_{\text{off}}\|_F^2,$$

причем первый член в правой части равен нулю в силу выбора B_0. Теперь с учетом унитарности U и V можно записать

$$\begin{aligned}\|A - UB_0V^T\|_F = \|(UBV^T + E - UB_0V^T)_{\text{off}}\|_F \leqslant \|U(B - B_0)V^T\|_F + \|E_{\text{off}}\|_F = \\ = \|B - B_0\|_F + \|E_{\text{off}}\|_F \leqslant \sqrt{n_1 n_2 r_1 r_2}\|E_\square\|_F + \|E_{\text{off}}\|_F.\end{aligned}$$

Максимум функции $\alpha a + b$ при $\alpha > 0$ и ограничении $a^2 + b^2 = 1$ составляет $\sqrt{\alpha^2 + 1}$, следовательно

$$\sqrt{n_1 n_2 r_1 r_2}\|E_\square\|_F + \|E_{\text{off}}\|_F \leqslant \sqrt{n_1 n_2 r_1 r_2 + 1}\|E\|_F,$$

что завершает доказательство теоремы. $\square$

2.5. Модельные численные эксперименты

В этом разделе мы приведем результаты применения описанного нами алгоритма для разложения двух трехмерных массивов:

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n,$$

$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2+j^2+k^2}}, \qquad 1 \leqslant i,j,k \leqslant n.$$

Аппроксимация строилась следующим образом:

1. Трижды применялся алгоритм 3, в котором в качестве направляющего последовательно брались индексы i, j и k. В результате получались аппроксимации вида

$$\begin{aligned} \mathcal{A} &\approx \tilde{\mathcal{A}}^{\{1\}} = \sum_{k=1}^{r_1} \left(V^{\{1\}} B_k^{\{1\}} W^{\{1\}T} \right) \otimes u_k^{\{1\}}, \\ \mathcal{A} &\approx \tilde{\mathcal{A}}^{\{2\}} = \sum_{k=1}^{r_2} \left(W^{\{2\}} B_k^{\{2\}} U^{\{2\}T} \right) \otimes v_k^{\{2\}}, \\ \mathcal{A} &\approx \tilde{\mathcal{A}}^{\{3\}} = \sum_{k=1}^{r_3} \left(U^{\{3\}} B_k^{\{3\}} V^{\{3\}T} \right) \otimes w_k^{\{3\}}. \end{aligned} \tag{2.31}$$

   При выполнении алгоритма, вследствие машинных погрешностей округления и использования стохастических методов при проверке критерия точности, ранги r_1, r_2, r_3 полученных аппроксимаций могли немного отличаться друг от друга.

2. Ортогональные столбцы $U^{\{1\}}$, $V^{\{2\}}$, $W^{\{3\}}$ использовались в качестве факторов U, V, W разложения Таккера (2.4). Ядро разложения вычислялось с помощью свертки (2.7), где вместо массива $\mathcal{A}$ использовалась одна из полученных аппроксимаций, что в данном случае приводит к формуле

$$g_{i'j'k'} = \left(V^{\{1\}} V^{\{2\}T} B_{i'}^{\{1\}} W^{\{3\}} W^{\{1\}T} \right)_{j'k'}.$$

   Таким образом строилось разложение Таккера для трехмерного массива с ядром $\mathcal{G}$ размера $r_1 \times r_2 \times r_3$. Однако вследствие того что критерий остановки в алгоритме 3 может быть слишком жестким, размеры этого $\mathcal{G}$ могут оказаться несколько большими, чем это в самом деле необходимо.

3. Чтобы снизить размеры ядра $\mathcal{G}$, применяем к нему метод редукции Таккера (2.5)-(2.7). Итоговый размер ядра приведен в таблицах 2.1,2.2.

Табл. 2.1 демонстрирует значения рангов, точности и размера полученной аппроксимации для массива $\mathcal{A}$, табл. 2.2 — для массива $\mathcal{B}$. Точность аппроксимации вычислялась оценочно по выборке элементов массива, столь большой, что удвоение выборки сохраняло величину ошибки с точностью 10%. Видим, что аппроксимация всегда удовлетворяет требованиям точности (то есть метод надежен) и приводит к очень значительной экономии памяти — например, данные, требующие при полном хранении двух петабайт ($2 \cdot 2^{50}$ байт) памяти, можно сжать до размера порядка 100 MB.

Алгоритм позволяет работать с массивами таких размеров даже на обычной персональной вычислительной станции. Время работы алгоритма на персональной машине с процессором `Pentium-4` с частотой 3.4 Ghz показано на рис. 2.4. Видим, что сложность алгоритма действительно является почти линейной по времени, более того, время практической работы алгоритма невелико — для построения аппроксимации массива $\mathcal{B}$ с полным размером два петабайта требуется всего около часа. Это показывает высокую практическую эффективность алгоритма и дает основания для проверки его «в реальных условиях», то есть при решении трехмерных интегральных уравнений, сводимых дискретизацией на тензорных сетках к линейным системам с плотными матрицами больших и сверхбольших размеров. Пример решения такой задачи приведен нами в разделе 3.3.

2.6. Асимптотика предложенного метода

Вопрос, обсуждаемый в этом разделе, можно сформулировать так: *Как значение ранга трехмерного массива ведет себя асимптотически при увеличении его размера? Согласуется ли предсказанная асимптотика для ранга с асимптотикой рангов разложений, практически вычисляемых трехмерным крестовым методом?*

Мы ответим на этот вопрос для типовых модельных массивов, соответствующих матрицам, порожденным сингулярными и гиперсингулярными интегральными уравнениями.

2.6.1. Теоретические оценки. В работе [67] теоретически обосновано наличие у матриц, порожденных асимптотически гладкими функциями $a_{ij} = F(x_i - y_j)$, тензорного разложения и приведены оценки на ранг и

Таблица 2.1. Результаты модельных экспериментов, массив $\mathcal{A}$

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n$$

Ранг и точность построенного разложения Таккера

ε	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	5	$2.57_{10}-4$	8	$2.33_{10}-6$	10	$1.46_{10}-8$	12	$3.09_{10}-10$
128	6	$6.86_{10}-4$	8	$4.47_{10}-6$	11	$3.19_{10}-8$	13	$6.4_{10}-10$
256	6	$8.84_{10}-4$	9	$3.97_{10}-6$	12	$5.49_{10}-8$	15	$3.03_{10}-10$
512	7	$7.49_{10}-4$	10	$1.41_{10}-6$	13	$6.84_{10}-8$	16	$5.2_{10}-10$
1024	7	$5.71_{10}-4$	11	$4.83_{10}-6$	14	$4.09_{10}-8$	18	$2.74_{10}-10$
2048	7	$6.63_{10}-4$	12	$2.08_{10}-6$	16	$4.01_{10}-8$	19	$3.97_{10}-10$
4096	8	$3.23_{10}-4$	12	$6.32_{10}-6$	17	$3.44_{10}-8$	21	$3.47_{10}-10$
8192	8	$6.36_{10}-4$	13	$3.36_{10}-6$	18	$1.93_{10}-8$	22	$4.56_{10}-10$
16384	9	$7.95_{10}-4$	14	$3.52_{10}-6$	19	$7.21_{10}-8$	24	$5.64_{10}-10$
32768	9	$6.4_{10}-4$	14	$8.86_{10}-6$	20	$5.27_{10}-8$	25	$3.79_{10}-10$
65536	9	$4.07_{10}-4$	15	$6.31_{10}-6$	21	$2.51_{10}-8$	26	$5.25_{10}-10$

Ранг и размер (MB) построенного разложения Таккера.
(Размеры менее 1MB в таблице не указаны)

ε		$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2MB	5		8		10		12	
128	16MB	6		8		11		13	
256	128MB	6		9		12		15	
512	1GB	7		10		13		16	
1024	8GB	7		11		14		18	
2048	64GB	7		12		16		19	
4096	512GB	8	<1	12	1.1	17	1.6	21	2
8192	4TB	8	1.5	13	2.5	18	3.5	22	4.2
16384	32TB	9	3.4	14	5.25	19	7.2	24	9
32768	256TB	9	6.75	14	10.5	20	15	25	19
65536	2PB	9	13.5	15	22	21	31	26	39

Таблица 2.2. Результаты модельных экспериментов, массив $\mathcal{B}$

$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2 + j^2 + k^2}}, \qquad 1 \leqslant i, j, k \leqslant n$$

Ранг и точность построенного разложения Таккера

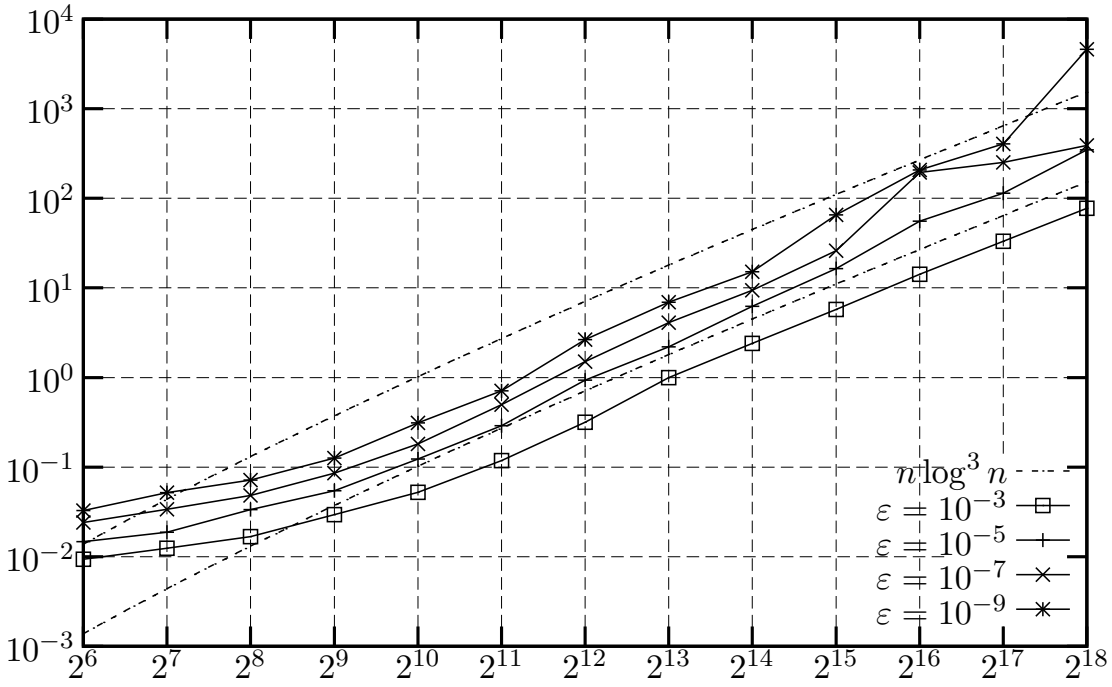
ε	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	7	$3.77_{10}-4$	11	$3.91_{10}-6$	14	$5.7_{10}-8$	18	$2.21_{10}-10$
128	8	$5.19_{10}-4$	12	$5.92_{10}-6$	17	$2._{10}-8$	20	$5.63_{10}-10$
256	9	$4.11_{10}-4$	14	$6.4_{10}-6$	19	$3.46_{10}-8$	23	$4.5_{10}-10$
512	10	$4.93_{10}-4$	15	$6.67_{10}-6$	21	$2.92_{10}-8$	26	$3.27_{10}-10$
1024	10	$5.47_{10}-4$	17	$3.21_{10}-6$	23	$3.95_{10}-8$	29	$4.73_{10}-10$
2048	11	$4.98_{10}-4$	18	$5.26_{10}-6$	25	$6.83_{10}-8$	31	$5.94_{10}-10$
4096	12	$8.4_{10}-4$	19	$4.25_{10}-6$	27	$3.56_{10}-8$	34	$3.38_{10}-10$
8192	12	$6.8_{10}-4$	20	$6._{10}-6$	28	$5.8_{10}-8$	36	$3.66_{10}-10$
16384	13	$2.69_{10}-4$	22	$4.78_{10}-6$	31	$5.65_{10}-8$	39	$2.67_{10}-10$
32768	13	$8.52_{10}-4$	23	$6.09_{10}-6$	32	$7.16_{10}-8$	41	$5.51_{10}-10$
65536	14	$6.27_{10}-4$	24	$6.52_{10}-6$	34	$7.89_{10}-8$	44	$1.41_{10}-9$

Ранг и размер (MB) построенного разложения Таккера.
(Размеры менее 1MB в таблице не указаны)

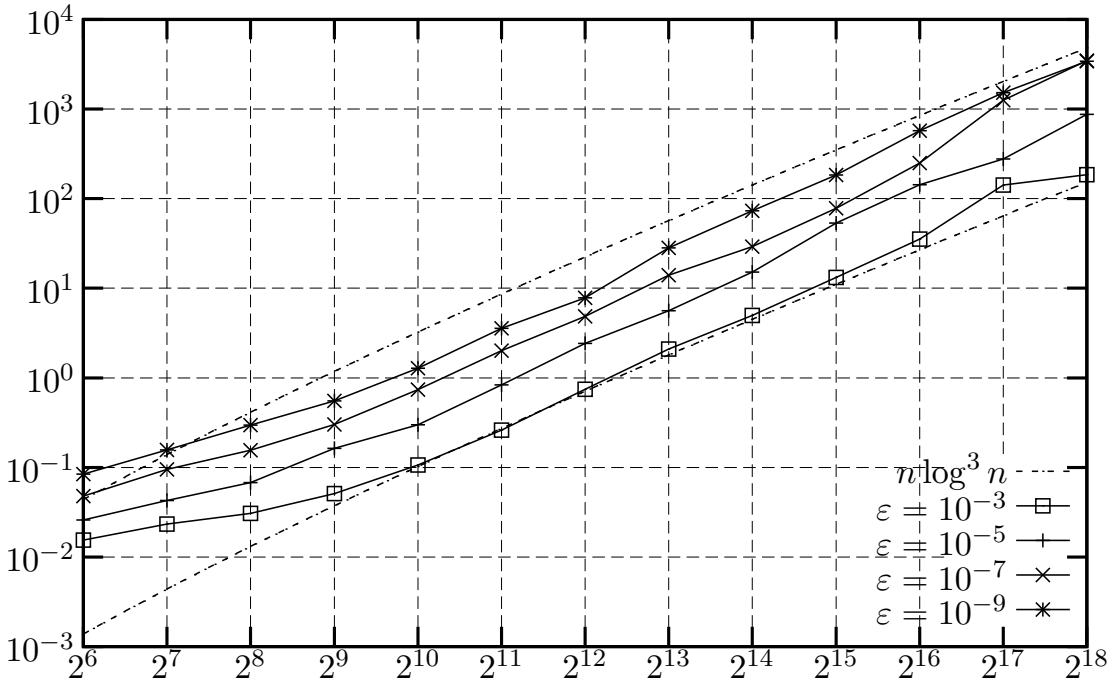
ε		$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2MB	7		11		14		18	
128	16MB	8		12		17		20	
256	128MB	9		14		19		23	
512	1GB	10		15		21		26	
1024	8GB	10		17		23		29	
2048	64GB	11	<1	18	<1	25	1.18	31	1.46
4096	512GB	12	1.15	19	1.78	27	2.54	34	3.2
8192	4TB	12	2.25	20	3.75	28	5.3	36	6.8
16384	32TB	13	4.9	22	8.25	31	11.7	39	14.7
32768	256TB	13	9.75	23	17.25	32	24	41	31
65536	2PB	14	21	24	36	34	51	44	66

Рис. 2.4. Время на построение аппроксимации, с.

$$\mathcal{A} = [a_{ijk}], \qquad a_{ijk} = \frac{1}{i+j+k}, \qquad 1 \leqslant i,j,k \leqslant n$$



$$\mathcal{B} = [b_{ijk}], \qquad b_{ijk} = \frac{1}{\sqrt{i^2+j^2+k^2}}, \qquad 1 \leqslant i,j,k \leqslant n$$



точность такого разложения в зависимости от свойств асимптотической гладкости порождающей функции. Оценки представлены в виде

$$\begin{gathered} r \leqslant \left(c_0 + c_1 \log h^{-1}\right) p^{m-1} + \tau, \\ |\{A - \tilde{A}_r\}_{ij}| \leqslant c_2 \gamma^p \|x_i - y_j\|^g. \end{gathered} \tag{2.32}$$

Здесь $m = 3$ — размерность пространства, параметры c_0, c_1, c_2 и τ зависят от размерности пространства и сеток x_i, y_j, h — минимальное расстояние от узлов сетки до сингулярности, величина γ меньше единицы, параметр g описывает асимптотическую гладкость функции F:

$$|D^p F(v)| \leqslant c d^p p! \|v\|^{g-p}, \qquad \forall p \geqslant 0. \tag{2.33}$$

В последнем уравнении $p = (p_1, \ldots, p_m)$, $p = p_1 + \ldots + p_m$,

$$D^p = \frac{\partial^{p_1} \ldots \partial^{p_m}}{(\partial v_1)^{p_1} \ldots (\partial v_m)^{p_m}}.$$

Рассматривая второе из уравнений (2.32) как оценку абсолютной погрешности аппроксимации

$$\varepsilon_{abs} = c_2 \gamma^p \|v\|^g$$

и учитывая, что

$$\|v\|^g = D^0 F(v) = F(v) = F(x - y) = a_{ij},$$

перепишем его в виде оценки на относительную погрешность

$$\varepsilon = \varepsilon_{rel} = c_2 \gamma^p,$$

поскольку именно в терминах относительной погрешности сформулирован критерий остановки в алгоритме 3. Отсюда $p \sim c_3 \log \varepsilon^{-1} + c_4$, что приводит к следующей оценке на ранг аппроксимации (для $m = 3$)

$$r \leqslant \left(c_0 + c_1 \log h^{-1}\right) \left(c_3 \log \varepsilon^{-1} + c_4\right)^2 + \tau.$$

На равномерных сетках $h^{-1} \sim n$, и оценка на асимптотическое поведение ранга может быть выписана в виде

$$r \leqslant c \log n \log^2 \varepsilon^{-1}. \tag{2.34}$$

Экспериментальной проверке этой асимптотической оценки посвящена наша работа [80]. В следующем разделе мы отразим лишь самые главные ее результаты.

2.6.2. Практические значения ранга. Мы рассматривали массивы

$$\mathcal{B}_{ijk} = \frac{1}{\sqrt{i^2 + j^2 + k^2}} = \frac{1}{\rho},$$

$$\mathcal{D}_{ijk} = \frac{1}{(i^2 + j^2 + k^2)^{3/2}} = \frac{1}{\rho^3}.$$

Вычисления происходили следующим образом.

- Сначала с помощью трехмерного метода крестовой аппроксимации мы получали разложение Таккера для этих массивов, как это показано в разделе 2.5.

- Затем к получившемуся ядру разложения Таккера применялись матричные [75, 78] методы поиска трилинейного разложения, ранг которого совпадает с размером ядра. Обычно такие методы дают хорошее начальное приближение. Если этого не происходило, мы увеличивали ранг искомого трилинейного разложения на единицу, то есть искали *переопределенное разложение* [77]. Этого оказывалось достаточно, чтобы получить начальное приближение с хорошей точностью.

- Затем использовалась комбинация минимизационных [75, 79, 81] методов поиска трилинейного разложения, являющихся весьма эффективными при наличии хорошего начального приближения. Получив трилинейное разложение ядра, мы восстанавливали трилинейное разложение всего массива.

Графики 2.5,2.6 содержат зависимости ранга от размера массива, где для каждой отдельно взятой точности ε показана прямая, наилучшим образом (в смысле задачи наименьших квадратов), приближающая указанное множество точек.

Графики 2.7,2.8 содержат зависимости ранга от точности, где для каждого отдельно взятого размера n показаны прямая и парабола, наилучшим образом приближающие указанное множество точек. Они показывают, что старший коэффициент в экспериментальной зависимости полученного ранга точности аппроксимации весьма невелик, и с высокой точностью можно описывать зависимость r от $\log n$ как линейную. Таким образом, можно считать экспериментально подтвержденным (для рассматриваемых массивов) оценку

$$r \sim \log n \log \varepsilon^{-1},$$

которая по ε лучше теоретической (2.34). Для массива $\mathcal{B}$ это можно обосновать теоретически, поскольку функция $1/\sqrt{x^2+y^2+z^2}$ является гармонической. Однако для этого потребуется специальная техника доказательства, отличная от представленной в [67]. Для массива $\mathcal{D}$ этот факт является чисто экспериментальным.

2.6.3. Время работы алгоритма. Очень важной практической характеристикой алгоритма является время его работы: даже самые точные и надежные алгоритмы становятся малополезными, если время их выполнения слишком велико; напротив, даже приближенные методы вычислений могут оказаться весьма применимыми, если они представляют результат, пусть и приближенный, достаточно быстро. Поэтому мы приведем здесь асимптотику времени работы алгоритма, вычисляющего трилинейное разложение для рассмотренных массивов. Мы указываем только время работы трехмерного крестового метода (Cross3D), поскольку на практике оно на один-два порядка больше, чем время поиска трилинейного разложения в ядре, а значит, именно его вклад в общую асимптотику сложности оказывается самым существенным.

На графиках 2.9,2.10 показана зависимость времени работы трехмерного крестового метода от размера массива; на графиках 2.11,2.12 — зависимость времени от точности. На первой серии графиков полученная зависимость (время исполнения программы от размера массива) аппроксимирована с помощью двух различных кривых, отвечающих линейной гипотезе

$$t = cn^{\alpha}, \qquad \log_{10} t = (\alpha \log_{10} 2)\log_2 n + \beta$$

и *логарифмической* гипотезе

$$t = cn\log_2^{\gamma} n, \qquad \log_{10} t = \gamma \log_{10}\log_2 n + \log_{10} 2 \cdot \log_2 n + \beta.$$

Здесь термины *линейный* и *логарифмический* описывают зависимость между логарифмами времени и размера (как это показано на графике). Зависимость между самим временем и размером, конечно, в первом случае степенная, а во втором — почти линейная. Интересно отметить, что коэффициент γ перед логарифмом, обнаруженный по экспериментальным данным, с высокой точностью оказывается равен трем, что отвечает теоретической оценке сложности крестового метода $\mathcal{O}(nr^3)$ при учете $r \sim \log n$. Таким же образом построены графики для зависимости времени работы программы от точности аппроксимации. Интересно отметить, что экспериментальное значение коэффициента

γ в логарифмической модели

$$t \sim \log^{\gamma} \varepsilon^{-1}$$

оказывается на практике меньше теоретического $\gamma = 3$.

Итак, при рассмотрении практических рангов трилинейных разложений, вычисляемых с помощью трехмерного крестового метода и комплекса минимизационных методов, оказалось, что их асимптотическое поведение для рассмотренных массивов можно описать зависимостью

$$r \sim \log n \log \varepsilon^{-1},$$

которая лучше, чем теоретическая оценка

$$r \leqslant \log n \log^2 \varepsilon^{-1}.$$

При этом время, которое требуется для сжатия больших массивов асимптотически растет почти линейно по размеру массива

$$t \sim n \log^3 n \log^{\gamma} \varepsilon^{-1},$$

где коэффициент γ на практике не превосходит трех.

Рис. 2.5. Зависимость ранга массива $\mathcal{B}$ от его размера

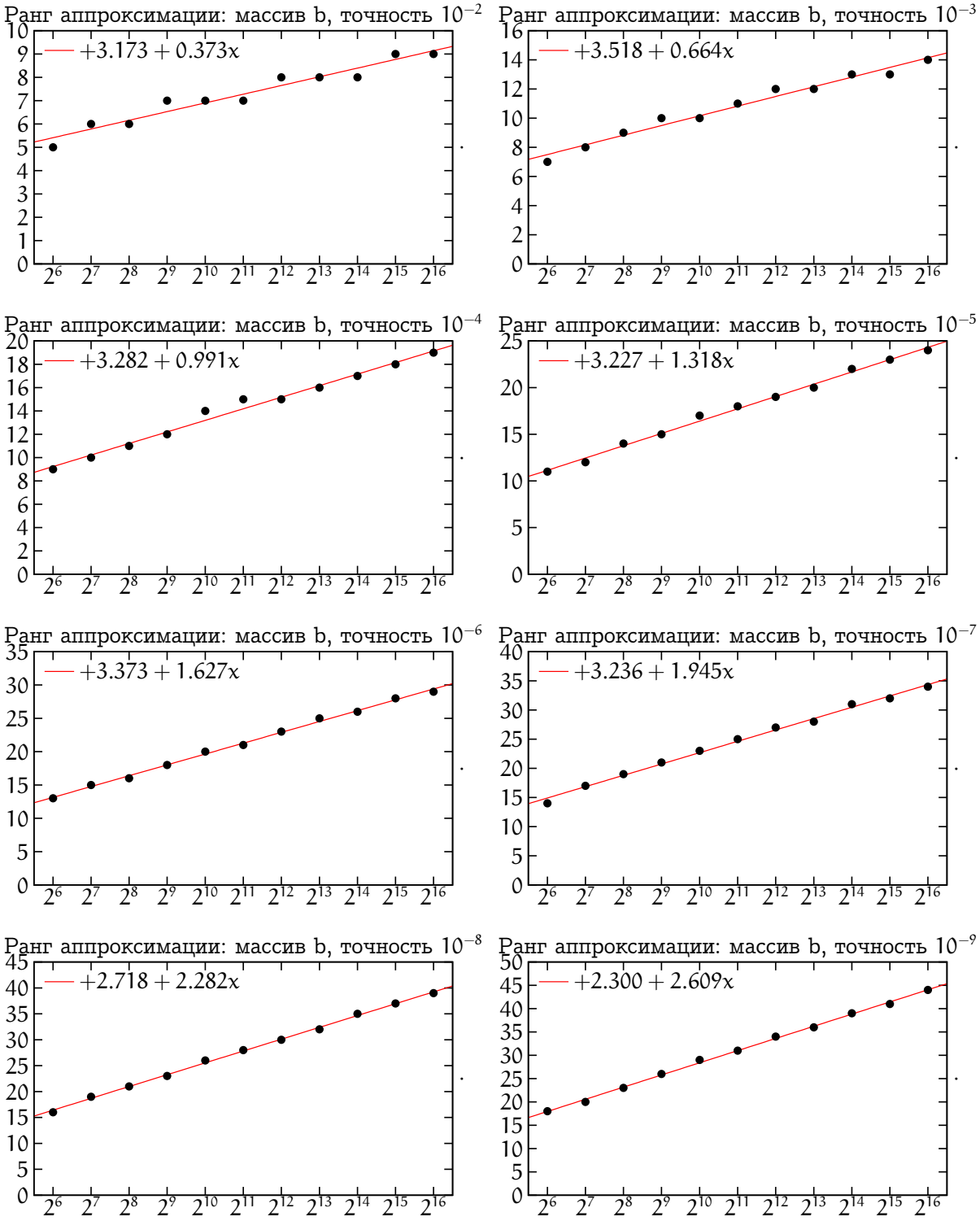


Рис. 2.6. Зависимость ранга массива $\mathcal{D}$ от его размера

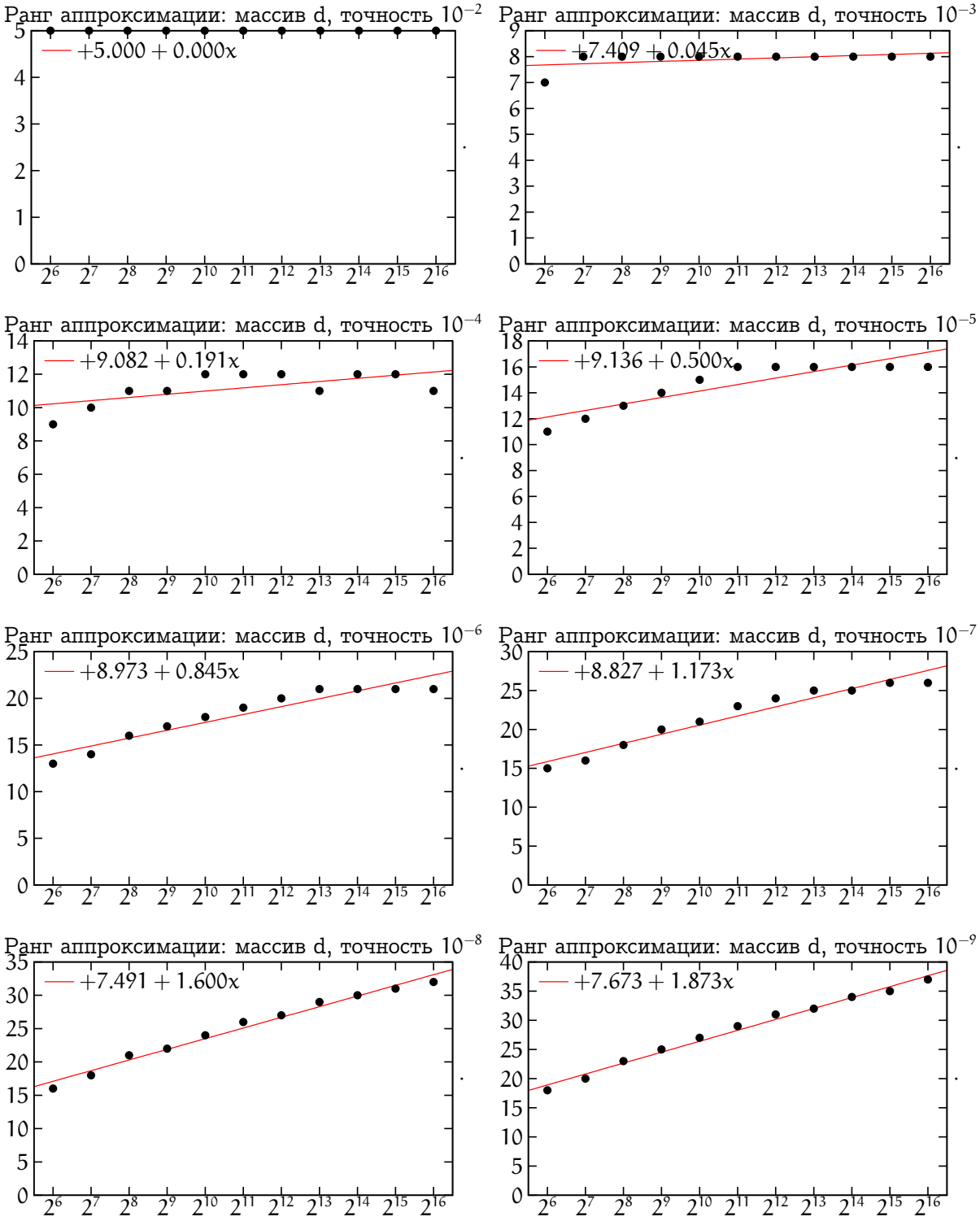


Рис. 2.7. Зависимость ранга массива $\mathcal{B}$ от точности

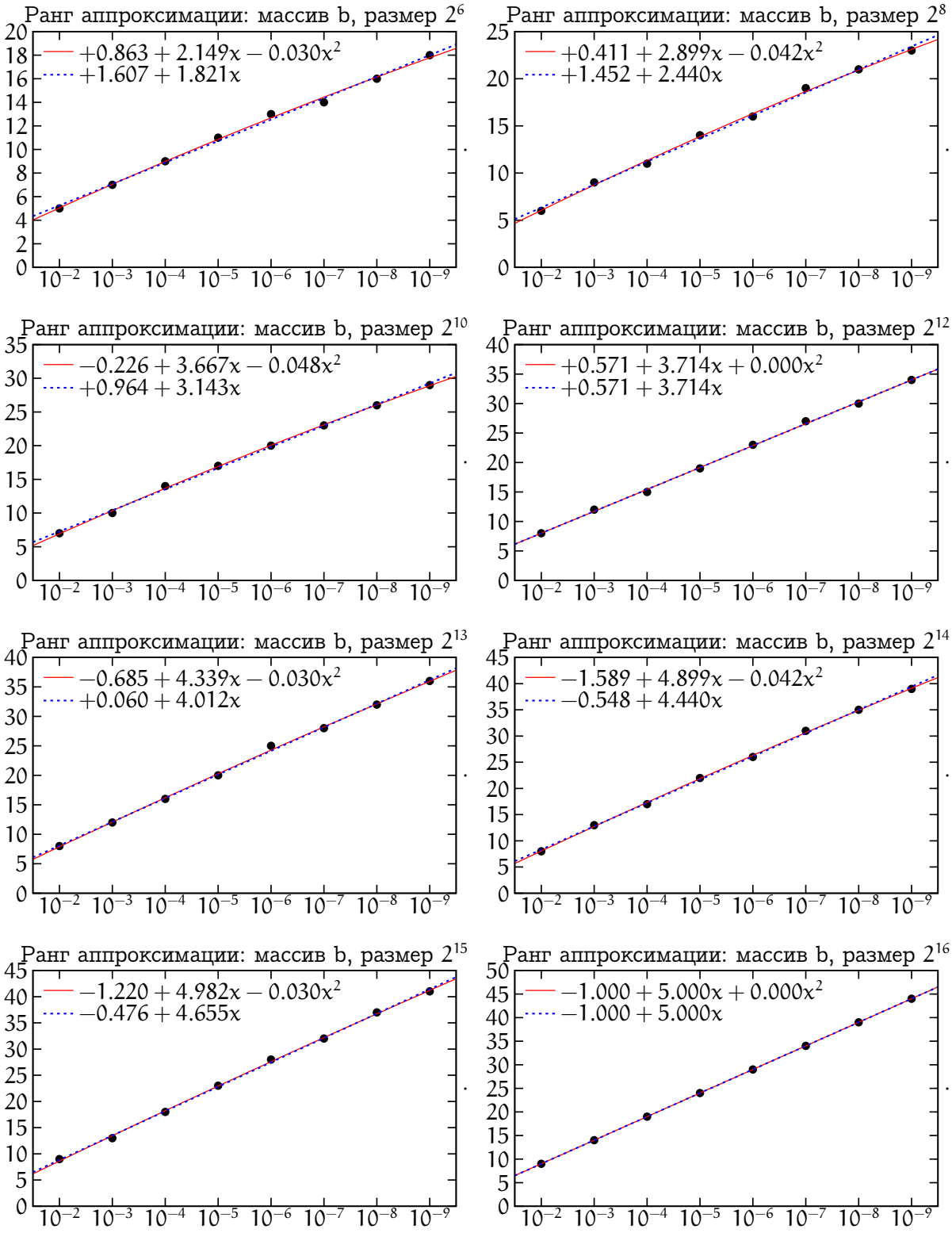


Рис. 2.8. Зависимость ранга массива $\mathcal{D}$ от точности

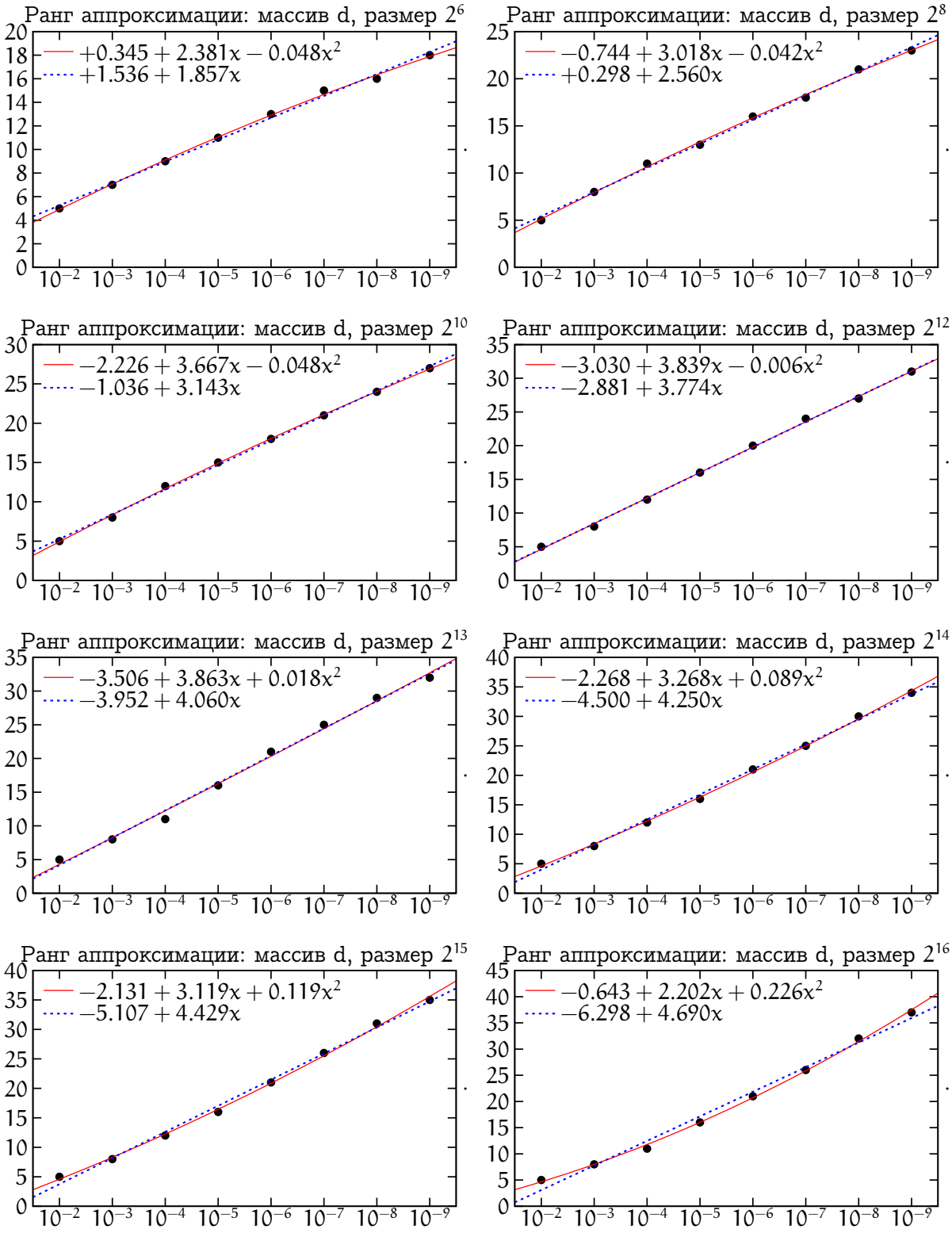


Рис. 2.9. Зависимость времени аппроксимации массива $\mathcal{B}$ от размера

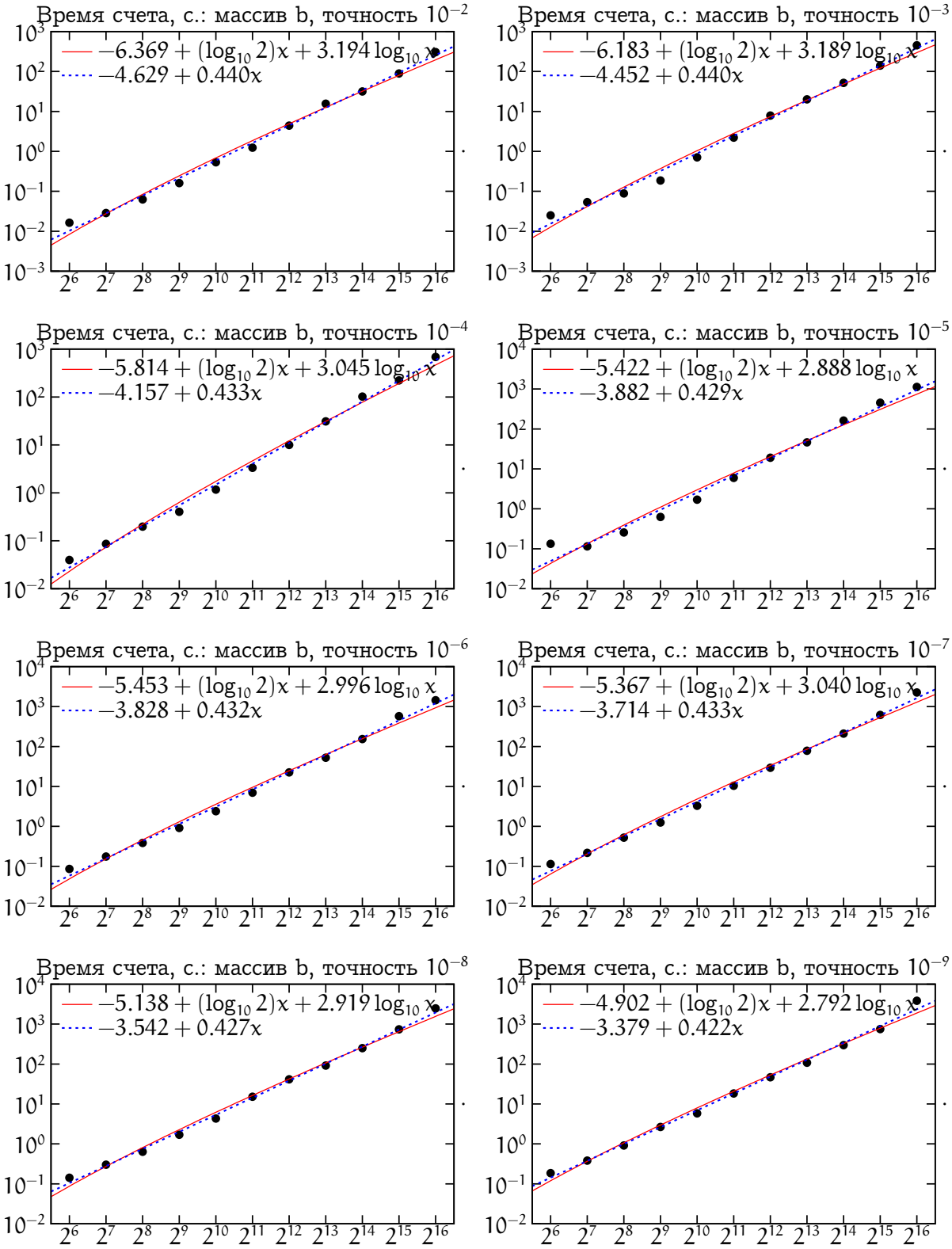


Рис. 2.10. Зависимость времени аппроксимации массива $\mathcal{D}$ от размера

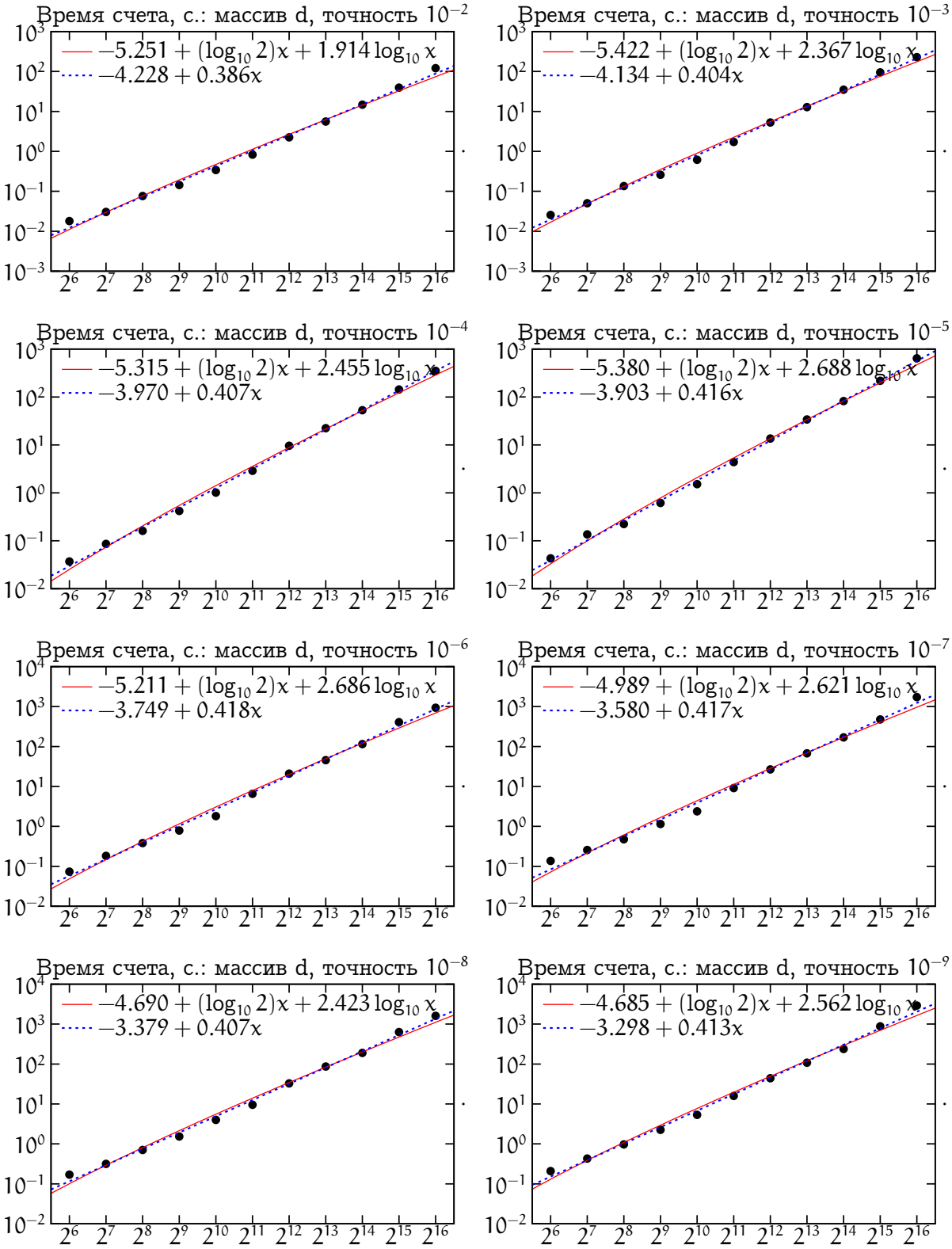


Рис. 2.11. Зависимость времени аппроксимации массива $\mathcal{B}$ от точности

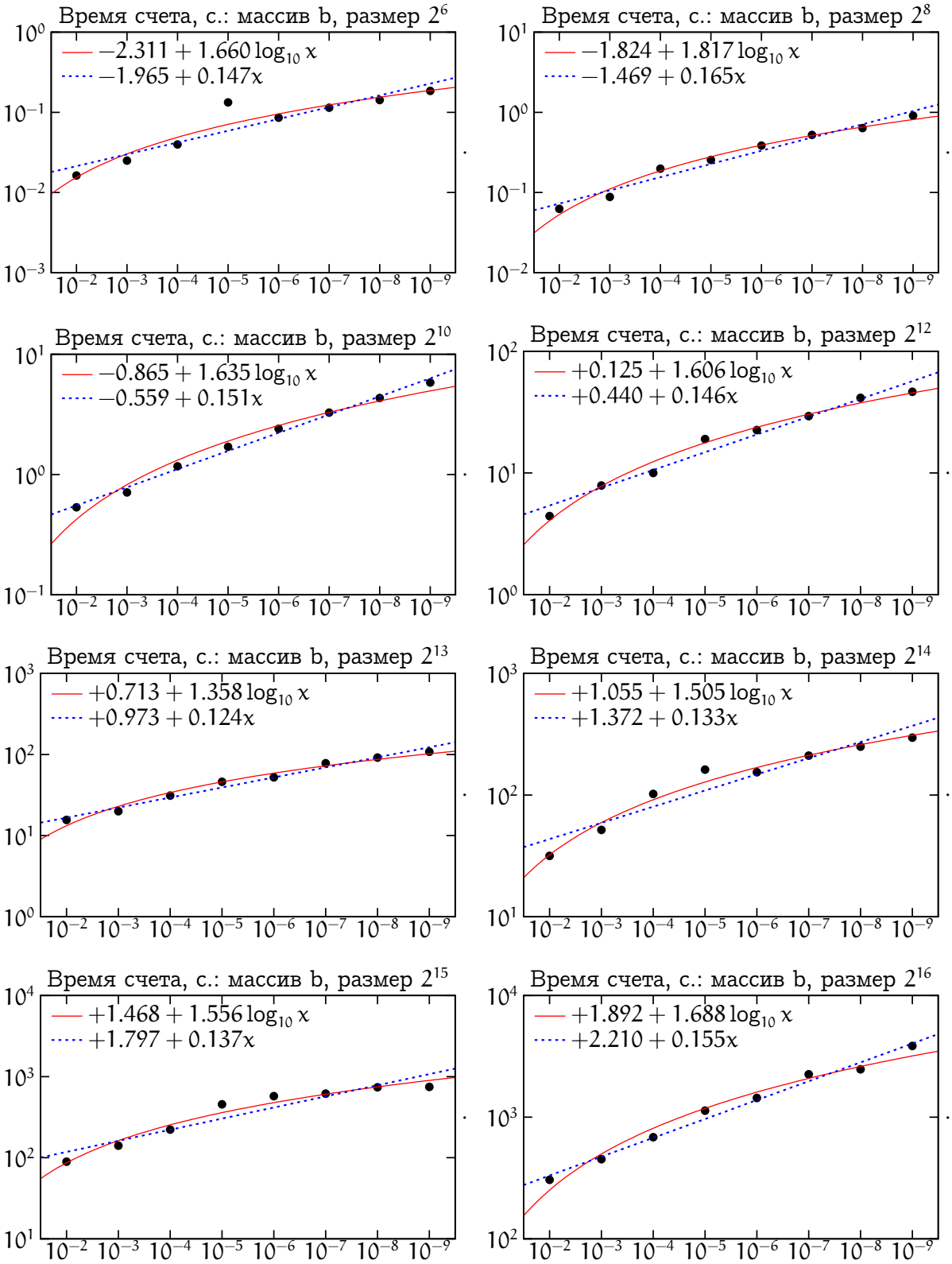
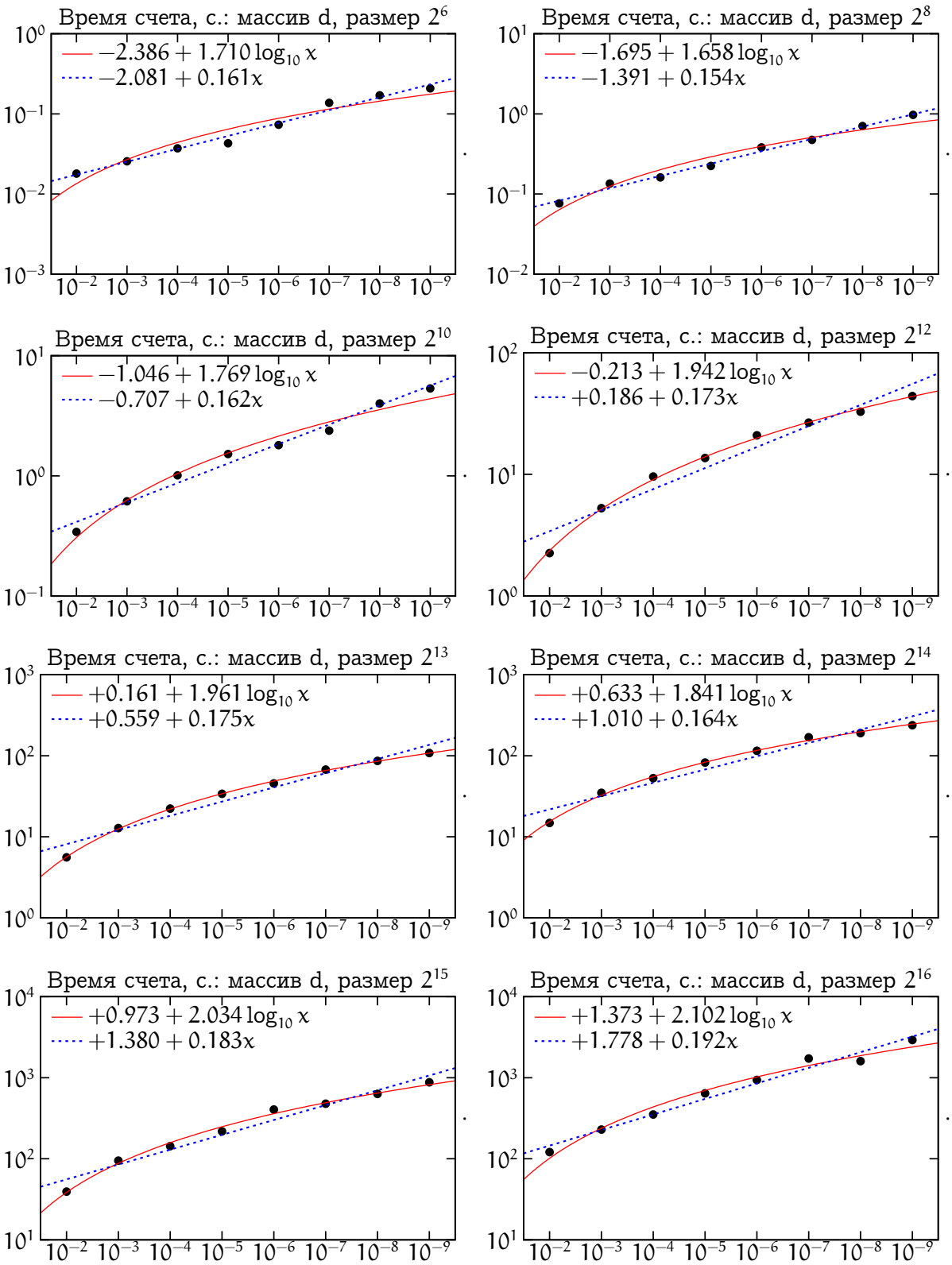


Рис. 2.12. Зависимость времени аппроксимации массива $\mathcal{D}$ от точности



2.7. Выводы

В этом разделе получены основные результаты диссертации:

- Теорема существования трехмерной крестовой аппроксимации, которая строится по небольшому числу столбцов, строк и распорок исходного массива.

- Алгоритм трехмерной неполной крестовой аппроксимации. В качестве входного параметра алгоритм использует процедуру вычисления любого элемента массива $[a_{ijk}]$, однако этот массив не вычисляется и не хранится полностью. Для построения аппроксимации $[a_{ijk}]$ в виде разложения Таккера достаточно вычислить порядка $\mathcal{O}(nr^a)$ $(r = \max(r_1, r_2, r_3),\ 1 \leqslant a \leqslant 2)$ элементов массива и совершить порядка $\mathcal{O}(nr^3)$ дополнительных действий. Алгоритм следует основным идеям, используемым при построении билинейной аппроксимации в методе неполной крестовой аппроксимации для матриц, и сохраняет главные достоинства этого метода — надежность, высокую эффективность, гибкость в применении к различным типам задач и простоту в практическом использовании.

- Проведено тестирование алгоритма трехмерной неполной крестовой аппроксимации на модельных массивах, отвечающих ядрам типовых интегральных уравнений. Экспериментально показана надежность метода, почти линейная скорость его работы. Проведено сравнение теоретической сложности алгоритма с практической асимптотикой его скорости, показано, что поведение алгоритма не хуже, а иногда лучше теоретического.

В следующем разделе рассмотренные нами методы би– и трилинейной аппроксимации будут применены к решению практических интегральных уравнений.

Глава 3. Приложения к численному решению уравнений

3.1. Применение мозаично-скелетонного метода к задаче Дирихле для уравнения Гельмгольца

3.1.1. Некоторые факты из теории потенциала. В качестве первого примера задачи, сводящейся к решению интегрального уравнения, рассмотрим задачу Дирихле для уравнения Лапласа, применив для ее решения методы теории потенциалов.

Пусть в некоторой области Ω задана функция u, удовлетворяющая внутри области дифференциальному уравнению

$$\mathcal{L}u = 0, \qquad \text{в} \quad \Omega \tag{3.1}$$

Пусть область $\Omega \subset \mathbb{R}^m$ ограничена, ее граница $\Gamma \stackrel{\text{def}}{=} \partial\Omega$ обладает классом гладкости C^2. Задача Дирихле для этого уравнения состоит в том, чтобы найти функцию $u \in C^2(\Omega) \bigcap C(\bar{\Omega})$, удовлетворяющую уравнению (3.1) и граничному условию

$$u|_\Gamma = f, \tag{3.2}$$

где f — заданная непрерывная функция.

Существует несколько подходов к решению этой задачи. Дифференциальный подход состоит в том, чтобы построить какую–нибудь сетку внутри Ω, аппроксимировать на ней дифференциальный оператор и граничное условие, получить алгебраическую дискретизацию задачи $Au = f$ с разреженной матрицей A и решить ее одним из большого числа методов, ориентируясь на полученную картину разреженности. Главные трудности, возникающие при таком подходе, состоят в следующем.

- Необходимо ввести в рассмотрение сетку, содержащую число узлов, растущее как $\mathcal{O}((a/h)^m)$, где a — характерный размер области Ω, h — характерный шаг сетки. Таким образом, затраты памяти на хранение сетки, вектора неизвестных и решения будут расти как $\mathcal{O}((a/h)^m) = \mathcal{O}(n^m)$, где n — порядковое число узлов сетки вдоль каждого направления. При решении задач в двух- и

особенно трехмерном пространстве такой рост затрат становится достаточно отягощающим.

- В случае неограниченной области Ω (например, при решении внешней задачи Дирихле), требуется ввести дополнительно внешнюю границу и разумно определить граничное условие на ней, что не всегда можно сделать достаточно обоснованно.

- Техника построения дискретного аналога задачи часто накладывает дополнительные условия на сетку (например, требование конформности), которое надо сочетать с хорошей аппроксимацией границы, сгущением к заданным множествам (например, к границам раздела сред) и по-возможности, разгущением в остальных областях. Вопросы построения сеток, отвечающих таким требованиям, до сих пор представляют существенные трудности.

Чтобы избавиться от проблем с сеткой в Ω, можно применить другой подход к задаче, сформулировав ее в виде граничного интегрального уравнения. Для этого требуется ввести функцию Грина, удовлетворяющую уравнению

$$\mathcal{L}G(x,y) = \delta(x-y), \qquad x,y \in \Omega$$

и дополнительным граничным условиям. Тогда значение неизвестной функции внутри области может быть представлено потенциалами простого или двойного слоя, которые определяются соответственно равенствами

$$u(x) \stackrel{\text{def}}{=} \int_\Gamma \varphi(y) G(x,y)\, ds(y), \qquad x \in \mathbb{R}^m \setminus \Gamma \tag{3.3}$$

$$v(x) \stackrel{\text{def}}{=} \int_\Gamma \varphi(y) \frac{\partial G(x,y)}{\partial n(y)}\, ds(y), \qquad x \in \mathbb{R}^m \setminus \Gamma \tag{3.4}$$

При решении задач Дирихле или Неймана для уравнений Лапласа или Гельмгольца функция Грина совпадает с фундаментальным решением, которое определяется формулами [22]

$$\Phi(x,y) = \begin{cases} \dfrac{1}{2\pi} \ln \dfrac{1}{\|x-y\|} & \text{для } m = 2, \\ \dfrac{1}{4\pi} \dfrac{1}{\|x-y\|} & \text{для } m = 3; \end{cases} \tag{3.5}$$

для уравнения Лапласа и

$$\Phi(x,y) = \begin{cases} \dfrac{i}{4}H_0^{(1)}(\kappa|x-y|) & \text{для } m=2, \\ \dfrac{1}{4\pi}\dfrac{e^{i\kappa|x-y|}}{\|x-y\|} & \text{для } m=3; \end{cases} \tag{3.6}$$

для уравнения Гельмгольца. Здесь $\|\cdot\|$ обозначает евклидову норму в пространстве $\mathbb{R}^m$.

Непосредственной проверкой устанавливается, что

- потенциал простого слоя с непрерывной плотностью φ является решением внутренней задачи Дирихле, если φ является решением интегрального уравнения первого рода

$$\int_\Gamma \varphi(y)\Phi(x,y)ds(y) = f(x), \qquad x \in \Gamma \tag{3.7}$$

- потенциал двойного слоя с непрерывной плотностью ψ является решением внутренней задачи Дирихле, если ψ является решением интегрального уравнения второго рода

$$\psi(x) - 2\int_\Gamma \psi(y)\frac{\partial\Phi(x,y)}{\partial n(y)}\,ds(y) = -2f(x), \qquad x \in \Gamma, \tag{3.8}$$

Известно также ([22]), что внутренняя задача Дирихле имеет единственное решение.

Для нахождения решения, мы можем теперь представить его в виде потенциала простого или двойного слоя и свести задачу к решению интегрального уравнения соответственно первого или второго рода.

3.1.2. Интегральное уравнение и дискретизация. Рассмотрим двумерное *уравнение Гельмгольца* с волновым числом κ, описывающее, например, рассеяние электромагнитной волны в среде, однородной вдоль некоторого направления

$$\begin{aligned} &\Delta u(x) + \kappa^2 u(x) = 0, && x \in \Omega \\ &u(x) = f(x), && x \in \Gamma, \quad \Gamma = \partial\Omega. \end{aligned}$$

Выражая неизвестную функцию $u(x)$ в виде потенциала простого слоя (3.3) на границе Γ, получаем интегральное уравнение

$$\int_\Gamma \Phi(x,y)\varphi(y)ds(y) = f(x), \qquad x \in \Gamma, \quad x,y \in \mathbb{R}^2 \tag{3.9}$$

с ядром *Ганкеля*

$$\Phi(x,y) = \frac{i}{4}H_0^{(1)}(\kappa|x-y|) \tag{3.10}$$

Теперь неизвестной является *плотность потенциала* $\varphi(y)$, определенная на границе подобластей. Введем параметризацию контура $z(t)$, $0 \leqslant t \leqslant 2\pi$, тогда точкам контура x, y будут соответствовать параметры τ и t. Отождествив в записи $\varphi(y) = \varphi(z(t))$ и $\varphi(t)$, запишем задачу в следующем виде

$$\int_0^{2\pi} \frac{i}{4}H_0^{(1)}(\kappa|z(t)-z(\tau)|)\varphi(t)|z'(t)|dt = f(\tau) \tag{3.11}$$

Для дискретизации полученного уравнения воспользуемся методом Галеркина. В качестве базисных функций $\{\psi_i(t)\}_{i=1}^n$ мы будем использовать кусочно-постоянные или кусочно–линейные базисные функции. Точки t_i, определяют сетку $\{t_i\}_{i=0}^n$, $t_0 = t_n$, введенную на круге $[0, 2\pi]$. Каждую базисную функцию можно ассоциировать с определенной точкой сетки, например, поиндексно. Для удобства дальнейших построений сделаем замену неизвестных

$$\chi(t) \stackrel{\text{def}}{=} \varphi(t)|z'(t)| \tag{3.12}$$

и будем искать приближенное решение $\chi^h(t)$ в виде линейной комбинации базисных функций

$$\chi^h(t) \stackrel{\text{def}}{=} \sum_{j=1}^{n} x_j\psi_j(t) \tag{3.13}$$

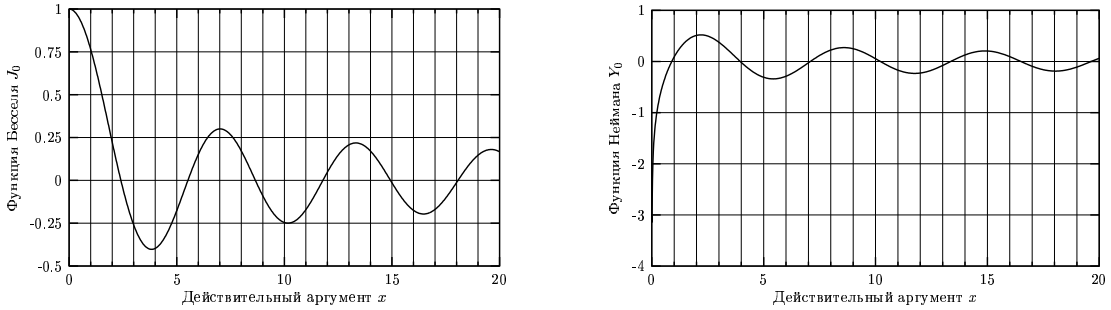
Следуя методу Галеркина, заменяем в задаче

$$\int_0^{2\pi} \frac{i}{4}H_0^{(1)}(\kappa|z(t)-z(\tau)|)\chi(t)dt = f(\tau) \tag{3.14}$$

точное решение на приближенное и домножаем обе части уравнения скалярно (в смысле $L_2[0, 2\pi]$) на набор тестовых функций. В качестве тестовых функций возьмем базисные функции $\psi_i(\tau)$. Таким образом мы переходим к алгебраической системе

$$\int_0^{2\pi}\int_0^{2\pi} \Phi(z(t), z(\tau)) \sum_{j=1}^{n}(x_j\psi_j(t))\psi_i(\tau)d\tau dt = \int_0^{2\pi} f(\tau)\psi_i(\tau)d\tau, \quad i = 1, \ldots, n \tag{3.15}$$

Рис. 3.1. Поведение функций Бесселя и Неймана нулевого порядка



то есть алгебраической задаче $Ax = b$ с матрицей

$$A = [a_{ij}], \qquad \text{где} \quad a_{ij} = \int_0^{2\pi}\int_0^{2\pi} \frac{i}{4} H_0^{(1)}(\kappa|z(t) - z(\tau)|)\psi_i(\tau)\psi_j(t)\,d\tau dt, \quad (3.16)$$

правой частью

$$b = [b_i], \qquad \text{где} \quad b_i = \int_0^{2\pi} f(\tau)\psi_i(\tau)\,d\tau, \quad (3.17)$$

и вектором неизвестных

$$x = [x_i]. \quad (3.18)$$

При численной реализации метода встает проблема вычисления матричных элементов, связанная с необходимость вычислять интегралы от ядра $H_0^{(1)}(z)$, которое неограниченно возрастает при $z \to 0$. Функция Ганкеля первого рода нулевого порядка является комбинацией функций Бесселя и Неймана нулевого порядка (см. рис. 3.1)

$$H_0^{(1)}(z) = J_0(z) + iY_0(z),$$

для которых можно написать разложение в ряды [43, 61]

$$J_0(z) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k!)^2}\left(\frac{z}{2}\right)^{2k}, \quad (3.19)$$

$$Y_0(z) = \frac{2}{\pi}J_0(z)\left(\ln\frac{z}{2} + \gamma\right) - \frac{1}{\pi}\sum_{k=1}^{\infty}\frac{(-1)^k}{k!(k-1)!}\left(\frac{z}{2}\right)^{2k}\left(\sum_{p=1}^{k}\frac{2}{p}\right),$$

где $\gamma \approx 0.57721566498606065120900824024 3$ называется константой Эйлера. Таким образом при малых значениях аргумента z

$$\frac{i}{4}H_0^{(1)} = -\frac{1}{2\pi}\ln\frac{z}{2} - \frac{\gamma}{2\pi} + \frac{i}{4} + o(1).$$

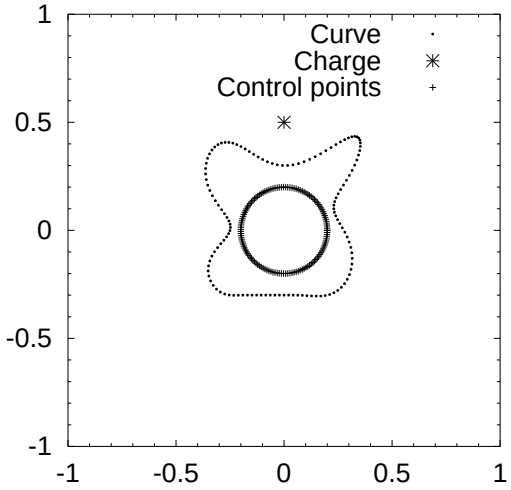


Рис. 3.2. Геометрия задачи

Интегралы от главных (быстрорастущих) частей разложения вычисляются аналитически, а остаток является гладкой функцией и может быть с хорошей точностью проинтегрирован численно на весьма небольшом числе точек. На этом основано эффективное вычисление матричных элементов, детальное описание которого дано в работе [71].

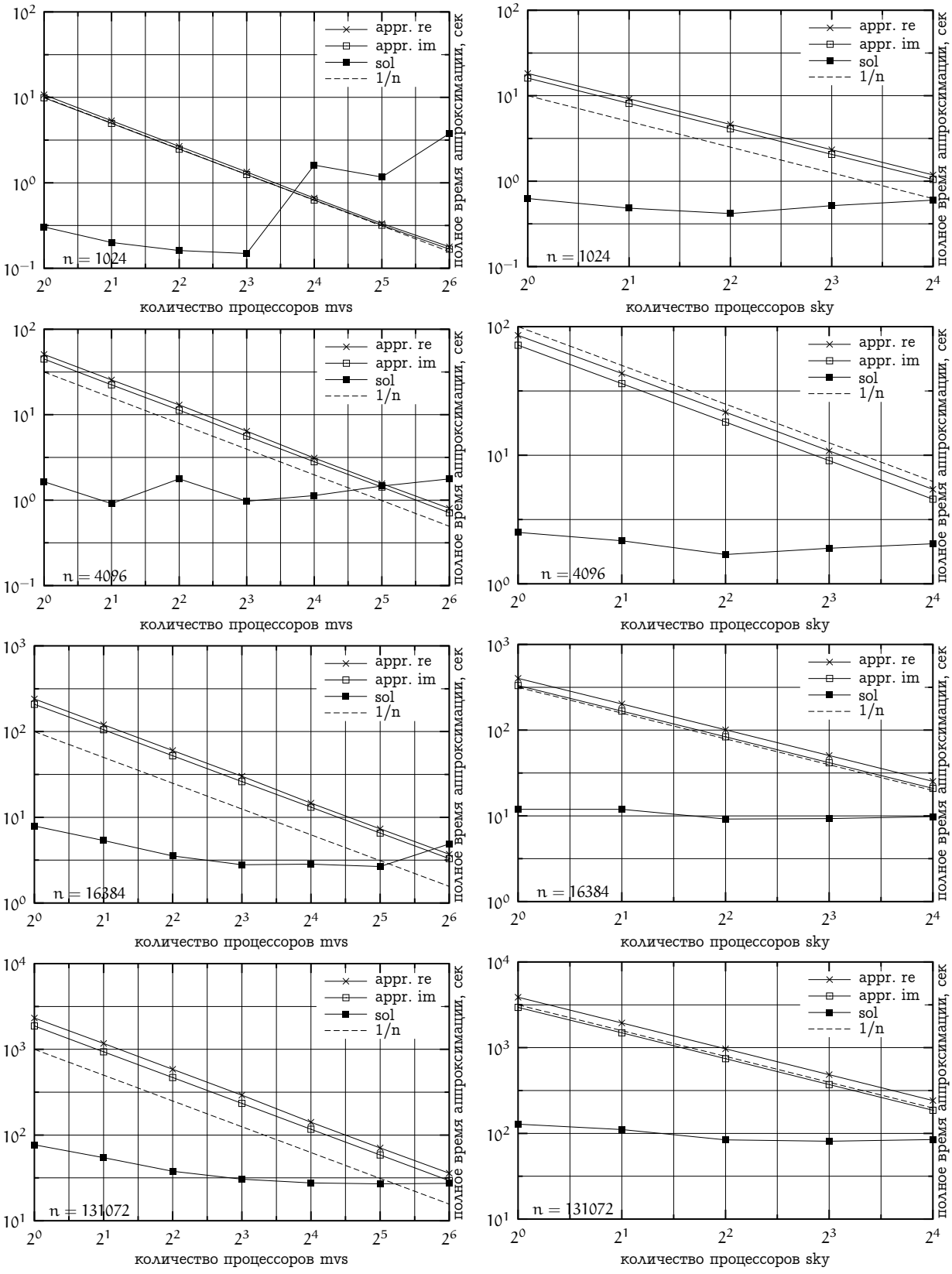
3.1.3. Численные эксперименты. Даже с учетом эффективной реализации процедуры вычисления матричных элементов, генерация полной матрицы все равно может занимать более 90% машинного времени системы. Отличительной особенностью мозаично-скелетонного метода является то, что он сокращает сложность вычисления матрицы и умножения на нее до *почти линейной* по размеру матрицы. Приведем некоторые результаты тестирования параллельной версии мозаично-скелетонного на двух платформах:

- МВС-1000М в Межведомственном Суперкомпьютерном Центре.
  Узлы на основе процессора Alpha21264A, 667 MHz, сеть Myrinet 2 Gbit/s.
  (`http://www.jscc.ru`)

- Кластер рабочих станций в НИВЦ МГУ (“sky”).
  Узлы на основе Pentium-III, 850 MHz, коммутатор Fast Ethernet.
  (`http://www.parallel.ru`)

Решалась задача (3.15) на гладком контуре (см. рис. 3.2). Использовались кусочно-постоянные базисные функции. Волновое число взято равным $\kappa = 10$. Параметр мозаичной аппроксимации для блоков составлял $\varepsilon = 10^{-4}$, матрица Галеркина полагалась симметричной.

Рис. 3.3. Ускорение мозаично-скелетонного метода



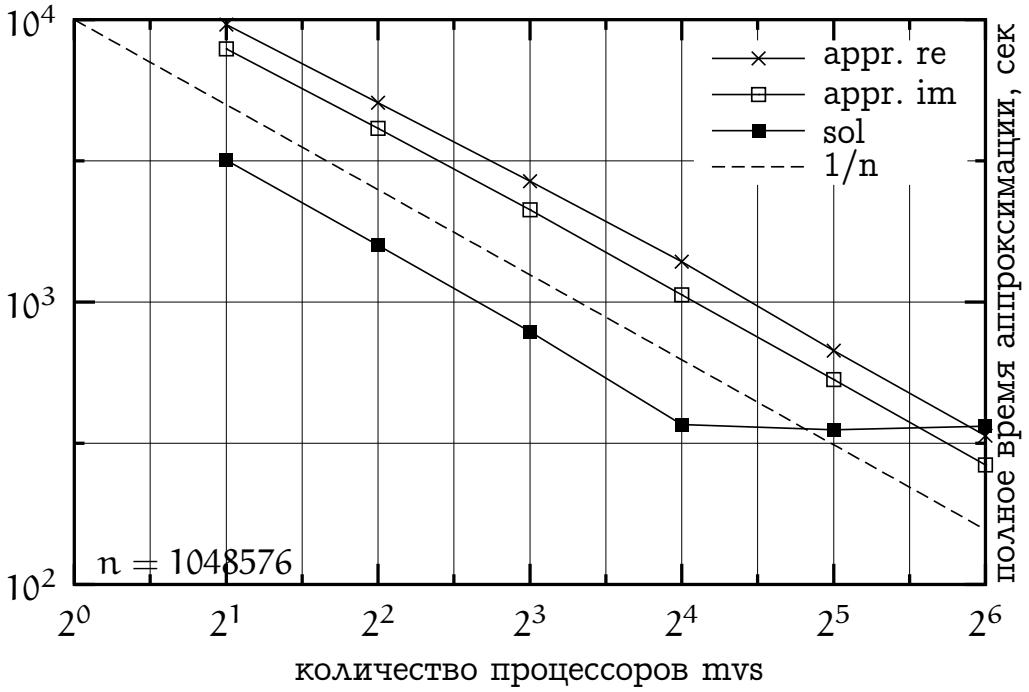


Рис. 3.4. Ускорение мозаично-скелетонного метода

Решение задачи с аппроксимированной матрицей производилось методом QMR (квази-минимальной невязки) предобусловленным с помощью циркулянта [6]. Критерием остановки являлось уменьшение невязки в 10^8 раз. Число итераций в результате составляло 17 и не менялось от размера матрицы или числа процессоров.

В разделе 1.2 мы представляли параллельную версию алгоритма крестовой аппроксимации. Продемонстрируем, какое ускорение алгоритма происходит при решении данной задачи на нескольких процессорах. На рис. 3.3 показана зависимость скорости работы от числа процессоров. Для широкого набора размеров матрицы мы наблюдаем практически одинаковую картину: затраты на аппроксимацию падают линейно с ростом числа процессоров при числе процессоров вплоть до 64 (и видимо, эта тенденция сохранилась бы и далее), а процедура умножения ускоряется довольно слабо, или же вообще не ускоряется. Причина понятна — операция умножения на локальную часть матрицы происходит достаточно быстро, и все расчетное время уходит на установление соединения в группе процессоров для суммирования локальных результатов, что существенно медленнее, чем локальный расчет. Впрочем, так как вычисление матрицы занимает во много раз больше времени, чем решение системы, алгоритм решения в целом ускоряется хорошо.

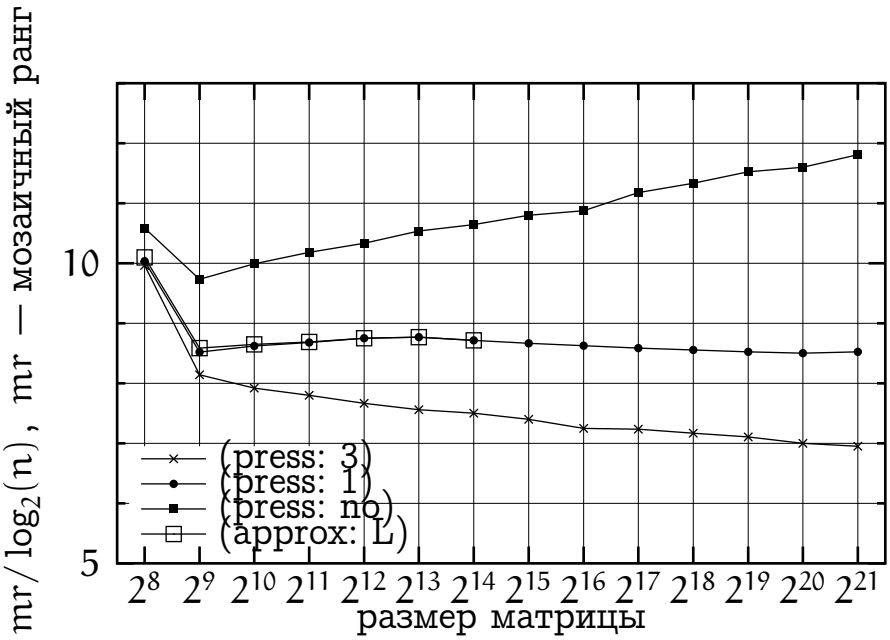


Рис. 3.5. Мозаичный ранг при использовании дожимателей и без них

Отдельно продемонстрируем результаты, полученные для матрицы размером более миллиона (рис. 3.4). В этом случае провести тест на одном процессоре невозможно, так как аппроксимант требует для сохранения 2.5 GB памяти. Разумное ускорение шага умножения мы получаем при числе процессоров вплоть до 16. Построение аппроксимации ускоряется линейно вплоть до 64 процессоров.

Продемонстрируем также результаты применения дожимателей (алгоритмов, описанных в пункте 1.1.2) для этого примера. На рис. 3.5 показаны значения мозаичного ранга (1.2) для матрицы (3.16) при применении различных алгоритмов построения аппроксимации:

(press: 3) Алгоритм 1 (Cross2D), переаппроксимация SVD;

(press: 1) Алгоритм 1, переаппроксимация методом Ланцоша;

(press: no) Алгоритм 1, без переаппроксимации;

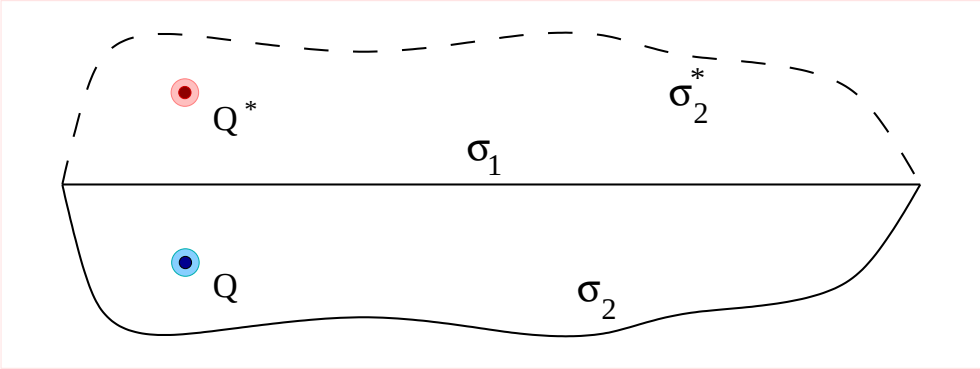
(approx: L) Метод Ланцоша, примененный к полному блоку.

Видим, наилучшее сжатие осуществляется методом с использованием Cross2D и дожимания на основе неявного SVD.

3.2. Применение мозаично-скелетонного метода к задаче гидроакустики

3.2.1. Постановка задачи. Процесс распространения звука в воде, в том числе и в «мелком море» , довольно сложен. Поэтому для его изучения

Рис. 3.6. Геометрия задачи о распространении звука в мелком море



построено множество моделей. Рассмотрим самую простую модель, когда скорость распространения звука в воде постоянна и равна c. Пусть слой воды занимает область D, границы которой — плоскость поверхности воды σ_1 и поверхность σ_2 — абсолютно отражающее звук морское дно.

Введем систему координат $Oxyz$ таким образом, чтобы плоскость Oxy совпадала с плоскостью σ_1, а источник лежал на положительной полуоси z. Распространение звука в воде от тонального источника Q с частотой ν и координатами $\mathbf{y} = (0, 0, z_q)$, описывается уравнением Гельмгольца с волновым числом $\kappa = \frac{2\pi\nu}{c}$:

$$\Delta P(\mathbf{x}) + \kappa^2 P(\mathbf{x}) = -\frac{Q}{4\pi}\delta(\mathbf{x}-\mathbf{y}), \quad \mathbf{x} \in D \tag{3.20}$$

и следующими граничным условиями

$$P|_{\sigma_1} = 0, \qquad \left.\frac{\partial P}{\partial n}\right|_{\sigma_2} = 0. \tag{3.21}$$

Для установления единственности решения будем искать такие $P(\mathbf{x})$, которые удовлетворяют условию излучения на бесконечности

$$\left(\frac{\mathbf{x}}{|\mathbf{x}|}, \operatorname{grad} P(\mathbf{x})\right) - i\kappa P(\mathbf{x}) = \mathcal{O}\left(\frac{1}{|\mathbf{x}-\mathbf{y}|}\right)$$

при $|\mathbf{x}-\mathbf{y}| \to \infty$.

Первичное поле звукового давления от источника Q определяется формулой

$$p_Q(\mathbf{x}) = \frac{Q}{4\pi}\frac{e^{i\kappa|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|}, \tag{3.22}$$

где Q — мощность источника. Выражая полное давление как сумму первичного и возмущенного давления

$$P(\mathbf{x}) = p_Q(\mathbf{x}) + P_1(\mathbf{x}),$$

получаем задачу определения возмущенного давления P_1, удовлетворяющего уравнению

$$\Delta P_1(\mathbf{x}) + \kappa^2 P(\mathbf{x}) = 0, \quad \mathbf{x} \in D, \tag{3.23}$$

и граничным условиям

$$P_1|_{\sigma_1} = -p_Q|_{\sigma_1}, \qquad \left.\frac{\partial P}{\partial n}\right|_{\sigma_2} = -\left.\frac{\partial p_Q}{\partial n}\right|_{\sigma_2}.$$

Для выполнения граничного условия Дирихле воспользуемся методом отражений. Рассмотрим поверхность σ_2^* симметричную поверхности σ_2 относительно плоскости $z = 0$. Обозначим через D' область, ограниченную поверхностями σ_2 и σ_2^*. Симметрично источнику расположим сток той же интенсивности, то есть

$$p_Q^*(\mathbf{x}) = -\frac{Q}{4\pi}\frac{e^{i\kappa|\mathbf{x}-\mathbf{y}^*|}}{|\mathbf{x}-\mathbf{y}^*|}, \tag{3.24}$$

где $\mathbf{y}^* = (0, 0, -z_Q)$ — «отражение» точки $\mathbf{y}$.

Будем искать давление в виде $P_1(M) = P_2(M) + p_Q^*(M)$, вводя новую неизвестную $P_2(M)$. Теперь условие Дирихле на границе σ_1 выполняется автоматически, а на поверхностях σ_2 и σ_2^* граничные условия преобразуются к виду

$$\left.\frac{\partial P_2}{\partial n}\right|_{\sigma} = -\left.\frac{\partial (p_Q + p_Q^*)}{\partial n}\right|_{\sigma}, \qquad \sigma = \sigma_2 \cup \sigma_2^*. \tag{3.25}$$

Таким образом, задача сводится к нахождению функции P_2, удовлетворяющей в области $D \cup D^*$ однородному уравнению Гельмгольца, а на ее границе условию Неймана (3.25). Решение задачи Неймана будем искать в виде потенциала двойного слоя, заданного на поверхности σ

$$P_2(\mathbf{x}) = \int\limits_{\sigma_2} \frac{\partial F(\mathbf{x}, \mathbf{y})}{\partial n} g(\mathbf{y}) d\sigma_2 + \int\limits_{\sigma_2^*} \frac{\partial F(\mathbf{x}, \mathbf{y})}{\partial n} g^*(\mathbf{y}) d\sigma_2^*,$$

где

$$F(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi}\frac{e^{i\kappa|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|}.$$

В силу симметрии задачи $g^*(x, y, -h(x, y)) = -g(x, y, h(x, y))$, поэтому для $P_2(\mathbf{x})$ выполняется (3.25), если $g(\mathbf{y})$ удовлетворяет интегральному уравнению на поверхности σ_2 :

$$\int\limits_{\sigma_2} \frac{\partial}{\partial n}\frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial n} g(\mathbf{y}) d\sigma_2 = -\frac{\partial (p_Q(\mathbf{x}) + p_Q^*(\mathbf{x}))}{\partial n}, \qquad \mathbf{x} \in \sigma_2, \tag{3.26}$$

где $G(x,y) = F(x,y) - F(x,y^*)$.

3.2.2. Метод замкнутых дискретных вихревых рамок. Для численного решения гиперсингулярного интегрального уравнения (3.26) используется метод замкнутых дискретных вихревых рамок [54, 58]. Поверхность σ параметризуется в виде $\sigma = \sigma(\xi,\zeta)$, где ξ и ζ — параметры, заданные на единичном квадрате

$$\Pi = \{0 \le \xi < 1, 0 \le \zeta < 1\},$$

и делится равномерно по параметрам на $s_1 s_2$ частей вдоль координатных линий $\xi_i = \text{const}$, $i = 1,\dots,s_1$, и $\zeta_i = \text{const}$, $i = 1,\dots,s_2$. Точки пересечения координатных линий обозначим через M^0_{ij} Таким образом поверхность σ состоит из $s_1 s_2$ криволинейных ячеек, которые имеют вершины $M^0_{i-1\,j-1}, M^0_{i\,j-1} M^0_{ij}, M^0_{i-1\,j}$. Расчетную точку y_{ij} поместим на пересечении отрезков, соединяющих середины противоположных боковых сторон соответствующей ячейки. Функция $g(y)$ аппроксимируется кусочно-постоянной функцией, и для определения неизвестных коэффициентов $g_{ij} = g(y_{ij})$ применяется метод коллокации на множестве точек x_{lm}, совпадающем с y_{ij}.

$$\sum_{i=1}^{s_1}\sum_{j=1}^{s_2} g_{ij}\Phi_{ij}(x_{lm}) = -f(x_{lm}), \qquad l = 1,\dots,s_1, \quad m = 1,\dots,s_2, \quad (3.27)$$

где $f(x)$ — правая часть уравнения (3.26), а $\Phi_{ij}(x)$ вычисляется по формуле [58], являющейся следствием формулы Стокса

$$\Phi_{ij}(x) = -\int\limits_{\partial\sigma_{ij}} n_x \cdot \left(\overrightarrow{dl_y} \times \operatorname{grad}_y G(x,y)\right) + \kappa^2 \int\limits_{\sigma_{ij}} G(x,y) n_x n_y \, d\sigma_y, \quad (3.28)$$

где $\partial\sigma_{ij}$ — граница ячейки σ_{ij}.

Предварительные исследования показывают, что искомая функция является сильно осциллирующей функцией, а значит для ее аппроксимации требуется порядка 10 ячеек на период. Это означает, что для решения задачи при частоте $\nu = 4\,\text{Гц}$, глубине источника $z_Q = 50\,\text{м}$, и скорости звука $c = 1450\,\text{м/с}$ необходимо решить систему порядка 400 уравнений. При увеличении частоты до $\nu = 30\,\text{Гц}$ возникает необходимость в решении $3 \cdot 10^4$ уравнений. В реальной акустике моря требуется исследовать поля с еще более высокими частотами, вплоть до 1000 - 2000 Гц.

3.2.3. Вычисление элементов матриц. Как вычислять $\Phi_{ij}(x_{lm})$? Как видно из выражения (3.28), эта величина представляется в виде суммы контурного интеграла и интеграла по площадке. Контурный интеграл будем вычислять по формуле прямоугольников. Так как площадка плоская, то $n_x \cdot n_y = \text{const}$ и интегралы по площадке имеют вид

$$\int\limits_{\sigma_{ij}} \frac{e^{i\kappa|x-y|}}{|x-y|}d\sigma = \int\limits_{\sigma_{ij}} \frac{\cos\kappa|x-y|}{|x-y|}d\sigma + i\int\limits_{\sigma_{ij}} \frac{\sin\kappa|x-y|}{|x-y|}d\sigma.$$

Функция $\sin\kappa r/r$ при малых $r = |x-y|$ является гладкой. Для вычисления интеграла от нее заменим интегрирование по площадке σ_{ij} на интегрирование по двум треугольникам с вершинами в точках $(M^0_{i-1\,j-1}, M^0_{i-1\,j}, M^0_{ij})$ и $(M^0_{i-1\,j}, M^0_{ij}, M^0_{ij-1})$. Для вычисления интегралов по треугольнику воспользуемся квадратурными формулами типа Гаусса порядка p. В наших вычислениях p бралось равным 8, и число точек для каждого треугольника равнялось 52.

Интеграл от $\cos\kappa r/r$ запишем в виде

$$\int\limits_{\sigma_{ij}} \frac{\cos\kappa r}{r}d\sigma = \int\limits_{\sigma_{ij}} \frac{\cos\kappa r - 1}{r}d\sigma + \int\limits_{\sigma_{ij}} \frac{1}{r}d\sigma, \tag{3.29}$$

где в первом интеграле стоит гладкая при малых r функция, интеграл от которой вычисляется по квадратурной формуле типа Гаусса, а второй интеграл берется аналитически.

Выделение главной части $(1/r)$ и использование квадратурных формул высокого порядка позволяет вычислять элементы матрицы с достаточной точностью. Это необходимо по следующим причинам.

- Матрица A системы оказывается плохо обусловленной, поэтому ошибка в элементах матрицы увеличивает ошибку в решении в $\text{cond}(A)$ раз, и при больших n исчезает внутренняя сходимость, решение «разбалтывается».

- Для вычисления интегралов типа (3.29) по простейшим квадратурным формулам для достижения необходимой точности нужно брать много точек (при использовании формулы прямоугольников порядка 100^2 вблизи особенности). Это очень сильно замедляет вычисление элементов матрицы.

Большой порядок используемых матриц и большая стоимость вычисления матричного элемента увеличивают вычислительную сложность этого алгоритма, поэтому весьма привлекательным является

Таблица 3.1. Результаты применения мозаично-скелетонного метода к задаче гидроакустики

Время генерации матрицы

Сетка	20×20	40×40	80×80	150×150
Стандартный	7 м.	2 часа	$>$ 1 дня	–
Мозаично-скелетонный	20с.	2.5 м.	13.3 м.	1 ч.

Время решения системы

n	20×20	40×40	80×80	150×150
Стандартный	6 с.	3 м. 7 с.	$\sim$ 1 ч.	–
Мозаично-скелетонный	7 с.	50 с.	4.3 м.	3.8 ч.

Мозаичные ранги для различных волновых чисел

κ	0.01	0.02	0.04	0.08	0.1
mr_{re}	627	740	950	1430	1660
$\mathrm{mr}_{\mathrm{im}}$	322	506	775	1320	1590

использование быстрых приближенных методов, таких как мозаично-скелетонная аппроксимация, позволяющих сократить затраты до почти линейных по размеру используемой сетки.

3.2.4. Численные эксперименты. Поскольку для применения мозаичного метода необходима лишь информация о сетке и процедура вычисления матричных элементов, то собственно его применение к задаче гидроакустики ничем не отличается от применения задаче для волнового уравнения, описанной в пункте 3.1. Поэтому мы сразу приведем результаты численных экспериментов.

В таблице 3.1 указано время генерации матрицы и время решения системы стандартным и мозаично-скелетонным методом. Видно, что использование мозаично-скелетонного метода ускоряет этап генерации более чем в сто раз, а этап решения более чем в десять раз.

Экспериментально исследуем зависимость мозаичного ранга от волнового числа κ. Матрицы аппроксимировались с точностью $\varepsilon = 10^{-4}$. В таблице 3.1 представлены также величины mr_{re} и $\mathrm{mr}_{\mathrm{im}}$ — мозаичные ранги действительной и мнимой частей соответственно. Вычисления проводились на сетке 250×250. Видно, что с увеличением частоты мозаичный ранг существенно увеличивается. Это связано с ухудшением констант асимптотической гладкости ядер типа $e^{\mathrm{i}\kappa r}/r$ и ограничивает применимость метода при больших значениях κ.

Результаты применения для той же задачи метода двумерной тензорной аппроксимации и дополнительные эксперименты представлены в работе [73].

3.3. Применение тензорных аппроксимаций к решению простейшего интегрального уравнения

Результаты раздела 3.3.5 получены совместно с И. В. Оселедцем.

3.3.1. Постановка задачи. Рассмотрим простейшее модельное трехмерное интегральное уравнение на кубе

$$\int_K \frac{u(y)}{|x-y|} dy = f(x), \qquad K = [0:1]^3. \tag{3.30}$$

Пусть дискретизация выполнена методом коллокации на неравномерной сетке, являющейся тензорным произведением трех сеток размера n. Количество неизвестных составляет $N = n^3$, количество ненулевых элементов матрицы равно $N^2 = n^6$, умножение матрицы на вектор требует n^6 действий, то есть для решения требуется затратить не менее $\mathcal{O}(n^6)$ операций. В этом разделе мы хотим показать, что с использованием техники построения тензорных аппроксимаций на основе алгоритма Cross3D мы можем решить возникающую линейную систему за $\mathcal{O}(n^2 \log^a n)$ действий при каком-то $a > 0$. Принципиально, что эта оценка имеет вид $o(N)$, то есть сложность метода асимптотически *меньше,* чем число неизвестных.

Конечно же, требуется сформулировать определенные предположения на класс задач, для которых такое решение существует. Кроме этого, специальная версия итерационного метода, применяемая нами для решения линейной системы, требует более детального рассмотрения и обоснования. Наконец, необходимо изучить свойства применяемого нами переобусловливателя. Все эти вопросы представляются нам крайне интересными и заслуживающими отдельного детального рассмотрения. В этой работе мы ограничимся лишь тем, что продемонстрируем *возможность* решения модельной задачи и опишем используемые для этого приемы.

3.3.2. Дискретизация задачи. Введем некоторые неравномерные сетки

$$\begin{aligned}
&\{x_i\}, \quad i = 1, \ldots, n+1, \qquad x_1 = 0, \quad x_{n+1} = 1;\\
&\{y_j\}, \quad j = 1, \ldots, n+1, \qquad y_1 = 0, \quad y_{n+1} = 1;\\
&\{z_k\}, \quad k = 1, \ldots, n+1, \qquad z_1 = 0, \quad z_{n+1} = 1;
\end{aligned}$$

и применим метод коллокации, выбрав в качестве базисных функций $\varphi_{ijk}(y)$ кусочно-постоянные элементы

$$\varphi_{ijk}(y) = \begin{cases} 1, & y \in \kappa_{ijk}, \\ 0, & \text{иначе}, \end{cases}$$

с носителем на ячейках

$$\kappa_{ijk} = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}].$$

В качестве точек наблюдения возьмем какие-то точки внутри этих ячеек

$$x_{ijk} = (x'_i, y'_j, z'_k) \in \kappa_{ijk}, \qquad i, j, k = 1, \ldots, n.$$

Получим линейную систем

$$Au = f, \qquad u = u_{i'j'k'}, \quad f = [f_{ijk}], \quad A = [a_{(ijk)(i'j'k')}], \tag{3.31}$$

$$u(y) = \sum_{i'j'k'} u_{i'j'k'} \varphi_{i'j'k'}(y), \quad f_{ijk} = f(x_{ijk}), \quad a_{(ijk)(i'j'k')} = \int\limits_{\kappa_{i'j'k'}} \frac{dy}{|x_{ijk} - y|}.$$

Для вычисления матричных элементов несложно получить аналитические формулы.

3.3.3. Сжатие матрицы. Для матрицы линейной системы (3.31) мы получали тензорную аппроксимацию (2.1) следующим образом.

1. К трехмерному тензору $\mathcal{A} = [a_{ijk}]$, $i = (i, i')$, $j = (j, j')$, $k = (k, k')$, трижды применялся алгоритм крестовой трехмерной аппроксимации, как это объяснено в пункте 2.5.

2. Для ядра полученного разложения Таккера строилось трилинейное разложение с помощью алгоритмов, описанных в работах [77, 78, 79]. Таким образом мы получали трилинейное разложение массива $\mathcal{A}$, то есть тензорную аппроксимацию матрицы A.

Результаты по сжатию матрицы линейной системы (3.31) приведены в таблице 3.2. Стоит заметить, что уже при размере сетки $m = 32$ матрица в полном виде не может быть сохранена в памяти рабочей станции, тогда как ее аппроксимация даже на сетке $m = 512$ занимает всего лишь 100MB памяти. Время получения тензорной аппроксимации также очень невелико и составляет всего лишь 30 минут для сетки $m = 512$.

Таблица 3.2. Результаты сжатия матрицы модельного трехмерного интегрального уравнения. Точность аппроксимации $\varepsilon = 10^{-5}$.

Размер сетки m	16	32	64	128	256	512
Полная матрица	128MB	8GB	512GB	32TB	2PB	128PB
Тензорная аппр.	74KB	320KB	1.1MB	6MB	25MB	114MB
Ранг	11	13	15	16	17	19
Время	0.6 с.	1.5 с.	8.4 с.	54 с.	5.5 мин.	30 мин.

3.3.4. Структурированные векторы. Обсудим теперь процедуру матрично-векторного умножения. Матрица A, представленная в виде суммы r тензорных произведений

$$A \approx \tilde{A} = \sum_{\alpha=1}^{r} U_\alpha \times V_\alpha \times W_\alpha, \qquad A \in \mathbb{R}^{m^3 \times m^3}, \quad U, V, W \in \mathbb{R}^{m \times m},$$

может быть умножена на вектор $b \in \mathbb{R}^{m^3}$ за $\mathcal{O}(rm^4) = \mathcal{O}(N^{4/3})$ операций. Это асимптотически быстрее, чем сложность умножения $\mathcal{O}(N^2)$ неструктурированной матрицы, но значительно медленнее, чем время построения тензорной аппроксимации, составляющее $\mathcal{O}(m^2 \log^a m) = o(N)$. Для того, чтобы сохранить эту асимптотику сложности и при решении интегрального уравнения, мы предлагаем использовать в итерационном методе *структурированные векторы*.

Суть идеи в следующем. При определенных условиях на гладкость правой части мы можем аппроксимировать ее в виде разложения Таккера

$$f_{ijk} \approx \tilde{f}_{ijk} = \sum_{i'j'k'} g^{(f)}_{i'j'k'} u^{(f)}_{ii'} v^{(f)}_{jj'} w^{(f)}_{kk'}$$

с некоторыми значениями модовых рангов (r_1, r_2, r_3), которые мы для простоты опускаем. Теперь вычисление матрично-векторного произведения $p = \tilde{A}\tilde{f}$ осуществляется за $\mathcal{O}(m^2 r(r_1 + r_2 + r_3))$ операций следующим образом

$$\begin{aligned} p_{i^*j^*k^*} &= \sum_{ijk} \left(\sum_\alpha (U_\alpha)_{i^*i} (V_\alpha)_{j^*j} (W_\alpha)_{k^*k} \right) \left(\sum_{i'j'k'} g^{(f)}_{i'j'k'} u^{(f)}_{ii'} v^{(f)}_{jj'} w^{(f)}_{kk'} \right) = \\ &= \sum_\alpha \sum_{i'j'k'} g_{i'j'k'} (U_\alpha u^{(f)})_{i^*i'} (V_\alpha v^{(f)})_{j^*j'} (W_\alpha w^{(f)})_{k^*k'}. \end{aligned}$$

Формально это ни что иное, как разложение Таккера для p с модовыми рангами (rr_1, rr_2, rr_3). Применив метод редукции Таккера к ядру этого разложения, мы можем с контролируемой погрешностью «дожать» его, снизив значения модовых рангов. При этом сложность на этапе дожимания не зависит от m и поэтому является пренебрежимо малой (при разумно невысоких значениях ранга).

Таким образом, если для решения уравнения $\tilde{A}\tilde{u} = \tilde{f}$ применяется итерационный метод, основанный на вычислении векторов пространства Крылова $v_k = A^k(f - Au^{\{0\}})$, мы можем вычислять и хранить эти векторы в таккеровском формате, пользуясь тем, что *умножение тензорной матрицы на таккеровский вектор дает таккеровский вектор.*

Предварительные эксперименты показывают, что модовые ранги $(r_1^{\{p\}}, r_2^{\{p\}}, r_3^{\{p\}})$ векторов Крыловского базиса на каждом шаге p итерационного метода несколько увеличиваются. Поэтому крайне важно добиться быстрой сходимости итерационного процесса — иначе из-за больших значений модовых рангов сложность вычисления матрично-векторного произведения становится слишком большой. Если же значения модовых рангов принудительно ограничить сверху, дожимая векторы до требуемого размера, то итерационный метод расходится. Таким образом, необходим выбор хорошего переобусловливателя.

3.3.5. Предобусловливание тензорных матриц. При построении переобусловливателя для матрицы, представленной в виде суммы тензорных произведений, нам требуется сразу же искать его в не менее эффективном виде, иначе сложность предобусловливания значительно превысит сложность матрично-векторного умножения. В этом разделе мы рассмотрим задачу об отыскании для матрицы

$$A = \sum_{\alpha=1}^{r} A_\alpha \times B_\alpha \times C_\alpha, \qquad A \in \mathbb{R}^{m^3 \times m^3}, \quad A_\alpha, B_\alpha, C_\alpha \in \mathbb{R}^{m \times m} \tag{3.32}$$

наилучшего предобусловливателя

$$P = X \times Y \times Z, \qquad P \in \mathbb{R}^{m^3 \times m^3}, \quad X, Y, Z \in \mathbb{R}^{m \times m} \tag{3.33}$$

имеющего вид тензорного произведения ранга один. Сформулируем ее как поиск минимума функционала $\|AP - I\|_F$. Предобусловливатели, полученные таким образом, называются супероптимальными [42]. Если предобусловливатель ищется в классе циркулянтных матриц, супероптимальный предобусловливатель обычно работает не слишком хорошо, но для P вида (3.33) наши результаты оказываются вполне удовлетворительными.

Решим эту задачу в более общем виде (это будет полезно нам в дальнейшем): для произвольной правой части F, представленной в виде разложения Таккера

$$F = \sum_{\varphi=1}^{r_1}\sum_{\psi=1}^{r_2}\sum_{\upsilon=1}^{r_3} g_{\varphi\psi\upsilon} U_\varphi \times V_\psi \times W_\upsilon, \qquad F \in \mathbb{R}^{m^3\times m^3}, \quad U, V, W \in \mathbb{R}^{m\times m} \tag{3.34}$$

найти X, Y, Z, доставляющие минимум функционала $\|AP - F\|_F$, где A и P имеют вышеуказанный вид.

Воспользуемся для решения задачи минимизации техникой *попеременных направлений* (alternating least squares, ALS). Этот метод часто используется в задаче построения трилинейного разложения (см., например [81]). Схематически он описывается так.

Алгоритм 5 *(ALS Prec-1)* При известных A вида (3.32) и F вида (3.33) требуется минимизировать $\|AP - F\|_F$ по всем P вида (3.33).

(0) Пусть дано некоторое приближение $P = X \times Y \times Z$ к решению задачи минимизации.

(1.x) Будем считать, что Y и Z «заморожены» , и найдем по ним новое значение $\hat{X}$ из решения задачи наименьших квадратов

$$\hat{X} = \arg\min_X \|A(X \times Y \times Z) - F\|_F . \tag{3.35}$$

Положим $X := \hat{X}$.

(1.y) Затем аналогично замораживая факторы X и Z, найдем новое $Y := \hat{Y}$, решив задачу

$$\hat{Y} = \arg\min_Y \|A(X \times Y \times Z) - F\|_F . \tag{3.36}$$

(1.z) Наконец, заморозив X и Y, найдем

$$\hat{Z} = \arg\min_Z \|A(X \times Y \times Z) - F\|_F , \tag{3.37}$$

после чего положим $Z := \hat{Z}$ и перейдем к следующей итерации на шаг **1.x**.

Для сходимости алгоритма обычно достаточно 10-20 итераций.

Осталось объяснить, как решать задачи (3.35)-(3.37). Рассмотрим только первую задачу, остальные решаются аналогично. Заметим, что фробениусова норма порождается скалярным произведением, определяемым для матриц $A, B \in \mathbb{R}^{m\times m}$ по формуле

$$\langle A,\ B\rangle = \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}b_{ij}, \qquad \langle A,\ A\rangle = \|A\|_F^2.$$

Задача минимизации квадратичной формы $\|AP - F\|_F$ равносильна выполнению условия

$$\langle AP - F,\ A\,\delta P\rangle = 0, \qquad \text{для всех} \quad \delta P.$$

Таким образом, задача (3.35) сводится к системе уравнений

$$\langle A(X\times Y\times Z) - F,\ A(\delta X\times Y\times Z)\rangle = 0, \qquad \text{для всех} \quad \delta X. \tag{3.38}$$

Учитывая вид A (3.32), имеем

$$A(X\times Y\times Z) = \sum_{\alpha=1}^{r} A_\alpha X\times B_\alpha Y\times C_\alpha Z = \sum_{\alpha=1}^{r} A_\alpha X\times B_\alpha^\bullet\times C_\alpha^\bullet,$$

где $B_\alpha^\bullet = B_\alpha Y$, $C_\alpha^\bullet = C_\alpha Z$ — известные матрицы. Система (3.38) принимает вид

$$\left\langle \sum_{\alpha=1}^{r}(A_\alpha X\times B_\alpha^\bullet\times C_\alpha^\bullet) - F,\ \sum_{\beta=1}^{r}\left(A_\beta\,\delta X\times B_\beta^\bullet\times C_\beta^\bullet\right)\right\rangle = 0.$$

Отсюда получаем

$$\sum_{\alpha\beta}\left\langle A_\alpha X\times B_\alpha^\bullet\times C_\alpha^\bullet,\ A_\beta\,\delta X\times B_\beta^\bullet\times C_\beta^\bullet\right\rangle = \sum_{\beta}\left\langle F,\ A_\beta\,\delta X\times B_\beta^\bullet\times C_\beta^\bullet\right\rangle. \tag{3.39}$$

Упростим правую часть равенства (3.39).

$$\left\langle A_\alpha X\times B_\alpha^\bullet\times C_\alpha^\bullet,\ A_\beta\,\delta X\times B_\beta^\bullet\times C_\beta^\bullet\right\rangle = \langle A_\alpha X,\ A_\beta\,\delta X\rangle\left\langle B_\alpha^\bullet,\ B_\beta^\bullet\right\rangle\left\langle C_\alpha^\bullet,\ C_\beta^\bullet\right\rangle.$$

Легко проверить, что $\langle A,\ B\rangle = \mathrm{Tr}(B^TA)$, где $\mathrm{Tr}(\,\cdot\,)$ означает след матрицы. Поэтому

$$\langle A_\alpha X,\ A_\beta\,\delta X\rangle = \mathrm{Tr}\left((A_\beta\,\delta X)^T(A_\alpha X)\right) = \left\langle A_\beta^TA_\alpha X,\ \delta X\right\rangle$$

и левая часть (3.39) имеет вид

$$\sum_{\alpha\beta} \left\langle A_\beta^{\mathsf{T}} A_\alpha X,\ \delta X \right\rangle p_{\alpha\beta}, \qquad p_{\alpha\beta} = \left\langle B_\alpha^\bullet,\ B_\beta^\bullet \right\rangle \left\langle C_\alpha^\bullet,\ C_\beta^\bullet \right\rangle. \tag{3.40}$$

Правая часть (3.39) приводится с учетом (3.34) к виду

$$\sum_{\beta\varphi\psi\upsilon} g_{\varphi\psi\upsilon} \left\langle A_\beta^{\mathsf{T}} U_\varphi,\ \delta X \right\rangle \left\langle V_\psi,\ B_\beta^\bullet \right\rangle \left\langle W_\upsilon,\ C_\beta^\bullet \right\rangle = \sum_{\varphi\beta} \left\langle A_\beta^{\mathsf{T}} U_\varphi,\ \delta X \right\rangle q_{\varphi\beta}, \tag{3.41}$$

где $q_{\varphi\beta} = \sum_{\psi\upsilon} g_{\varphi\psi\upsilon} \left\langle V_\psi,\ B_\beta^\bullet \right\rangle \left\langle W_\upsilon,\ C_\beta^\bullet \right\rangle$.

Таким образом, система (3.39) принимает вид

$$\sum_{\alpha\beta} \left\langle A_\beta^{\mathsf{T}} A_\alpha X,\ \delta X \right\rangle p_{\alpha\beta} = \sum_{\varphi\beta} \left\langle A_\beta^{\mathsf{T}} U_\varphi,\ \delta X \right\rangle q_{\varphi\beta}. \tag{3.42}$$

Взяв последовательно $\delta X = E_{ij}$ (матрица E_{ij} имеет единственный ненулевой элемент в позиции i,j), мы получаем линейную систему

$$\sum_{\alpha\beta} (A_\beta^{\mathsf{T}} A_\alpha X)_{ij} p_{\alpha\beta} = \sum_{\varphi\beta} (A_\beta^{\mathsf{T}} U_\varphi)_{ij} q_{\varphi\beta}, \qquad i,j = 1,\ldots,m,$$

из m^2 уравнений для определения m^2 элементов матрицы X. При этом X может быть вычислена эффективно, поскольку в матричном виде

$$\sum_{\alpha\beta} p_{\alpha\beta} A_\beta^{\mathsf{T}} A_\alpha X = \sum_{\varphi\beta} q_{\varphi\beta} A_\beta^{\mathsf{T}} U_\varphi$$

для этого достаточно лишь обратить матрицу $\sum_{\alpha\beta} p_{\alpha\beta} A_\beta^{\mathsf{T}} A_\alpha$ размера $m \times m$.

Таким образом способ решения задачи (3.35) полностью описан, а значит описан и алгоритм построения тензорного переобусловливателя единичного ранга. На его основе мы можем предложить алгоритм построения тензорного предобусловливателя ранга r.

Алгоритм 6 *(ALS Prec-r)*

(0) Положить $k = 1, F_1 = I$.

(1) С помощью алгоритма 5 найти

$$P_k = \arg\min_P \|AP - F_k\|.$$

(2) Вычислить $F_{k+1} = F_k - AP_k$ использовать метод редукции Таккера для более эффективного хранения F_{k+1}.

(3) Установить $k := k + 1$ и если $k \leqslant r$, вернуться на шаг **1**.

Окончательно $P(r) = \sum_{k=1}^{r} P_k$.

Если $r_2 > r_1$, то $\|AP(r_2) - I\|_F \leqslant \|AP(r_1) - I\|_F$. Поэтому мы склонны ожидать, что с переобусловливателем $P(r_2)$ итерационный метод сойдется быстрее. Однако на практике это не так: использование переобусловливателя $P(r)$ не всегда приводит к лучшим результатам, чем использование $P(1)$. Это происходит по следующим причинам:

- умножение $P(r)$ на вектор требует больше времени, чем умножение $P(1)$, поэтому формально снижая число итераций, мы можем увеличить общее время решения;
- умножение на предобусловливатель высокого ранга приводит к более быстрому росту модовых рангов векторов в итерационном процессе. За счет этого вычисления становятся более сложными и требуют больше памяти. Иногда это вынуждает нас принудительно ограничивать модовые ранги, что может привести к расходимости алгоритма.

С учетом сделанных замечаний ясно, что выбор оптимального ранга переобусловливателя $P(r)$ требует дополнительного изучения.

3.3.6. Численные результаты. Приведем результаты решения задачи (3.31). Для построения метода коллокации мы используем по каждому направлению пару чебышевских сеток размера $m = 64$. Вектор правой части f_{ijk} формируется случайным образом. Тензорная аппроксимация матрицы строится с точностью $\varepsilon = 10^{-5}$. Для решения использовался метод BCGstab. На рисунке 3.7 показано поведение итерационного метода при использовании предобусловливателей $P(r)$ с различными значениями r. Без предобусловливания сходимость метода не достигалась.

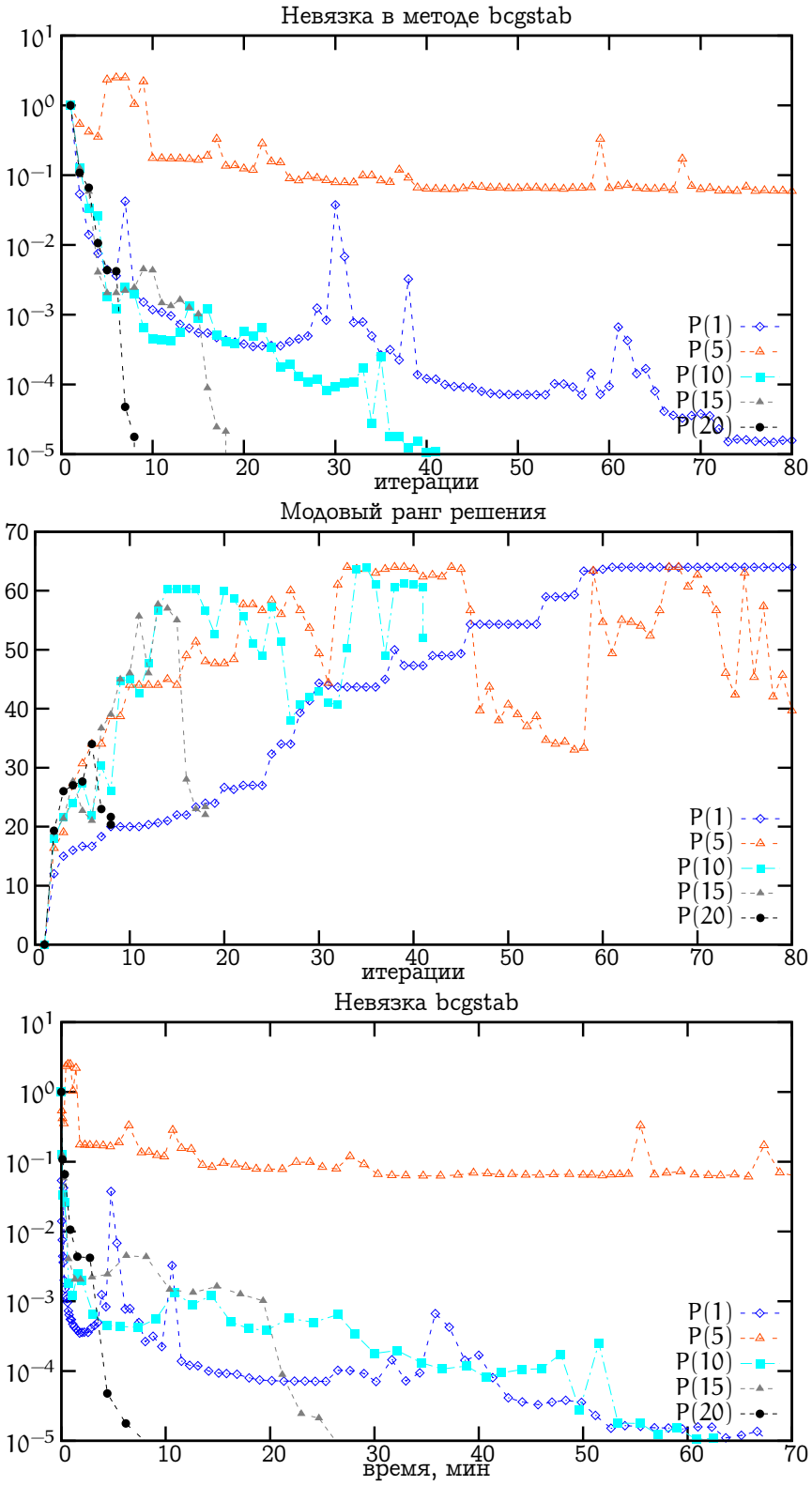
На первой картинке продемонстрировано убывание невязки в ходе работы итерационного метода. Заметим, что никакой «монотонной» зависимости скорости убывания невязки от от ранга r выбранного предобусловливателя $P(r)$ не наблюдается. Например, методы с $P(1)$ и $P(10)$ сходятся хорошо, а метод с $P(5)$ не сходится. Причина, видимо, состоит в росте модовых рангов (они продемонстрированы на второй картинке) и принудительном сжатии векторов до модового ранга

64. На третей картинке представлено значение невязки в зависимости от реального времени расчета. На этом графике мы можем сравнивать чистую скорость методов с различными $P(r)$. Заметим, что хотя метод, предобусловленный матрицей $P(10)$ сошелся за меньшее число итераций, реальной затраченное на вычисление время практически то же, что и при использовании $P(1)$. При больших значениях r результаты становятся более предсказуемы: метод с $P(20)$ оказался быстрее метода с $P(15)$, а тот свою очередь быстрее метода с $P(10)$, как в смысле итераций, так и по чистому времени.

3.4. Выводы

В этом разделе на примере решения модельных и практических задач, сведенных к решению интегральных уравнений, мы продемонстрировали эффективность методов полилинейной аппроксимации. Показано, что методы, основанные на билинейной аппроксимации (мозаично-скелетонный метод) приводят к алгоритмам решения почти линейной (по числу неизвестных) сложности, а методы трилинейной аппроксимации при решении объемных интегральный уравнений в трехмерном пространстве позволяют строить алгоритмы сублинейной сложности.

Рис. 3.7. Результаты решения модельного трехмерного интегрального уравнения



Глава 4.
Специфика матриц в одной задаче электродинамики

4.1. Постановка задачи

4.1.1. Физическая постановка. Технологический процесс геологической разведки часто включает в себя метод электромагнитного каротажа, или же *зондирования* области, с целью определить электромагнитные параметры среды, расположенной в недоступной области за границей шахты. Численная обработка полученных данных измерений основана на решении обратной задачи электромагнитного рассеяния, то есть на определении электромагнитных параметров среды по известной информации об источниках и значениях электромагнитного поля в точках измерения. Решение обратной задачи в существенной степени основано на эффективном алгоритме решения *прямой задачи* электромагнитного рассеяния, то есть в определении значений электрического и магнитного полей в заданных точках при известной конфигурации источников и заданной геометрии и параметрах среды. Решение прямой задачи может включаться как элемент в алгоритм решения обратной задачи; кроме того, алгоритм решения прямой задачи рассеяния, необходим на этапе отладки алгоритма решения обратной задачи, поскольку позволяет получать данные измерений методом численного, а не физического (более дорогого, менее точного и не всегда возможного) эксперимента.

Прямая задача электромагнитного рассеяния в некотором модельном приближении может быть поставлена так. Рассмотрим распространение электромагнитной волны в полупространстве, ограниченном плоскостью, которая представляет собой идеальный проводник. Будем считать, что зависимость полей от времени гармоническая $e^{-i\omega t}$. Электрические параметры среды будем считать переменными, но ограничимся моделью *локально неоднородной* среды, согласно которой среда представляет из себя однородное пространство в котором располагается конечных размеров *неоднородность* $\mathcal{V}$ [65, 64]. Размеры неоднородности предполагаются столь большими, чтобы их увеличение не оказывало существенного влияния на поведение полей в области, где производятся измерения. Таким образом, тот факт, что неоднородное пространство сведено к неоднородности конечных размеров,

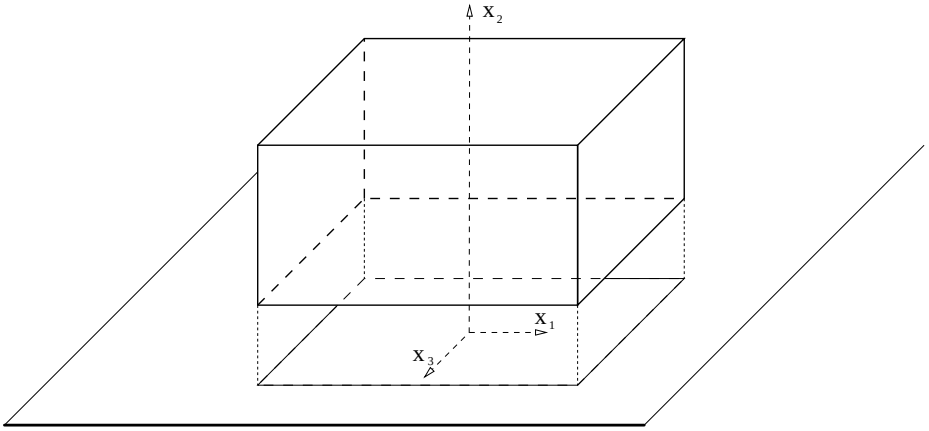


Рис. 4.1. Геометрия задачи

не отражается на значениях измеряемых величин.

Мы ограничим рассмотрение случаем, когда область $\mathcal{V}$ по форме является параллелепипедом. Введем прямоугольную систему координат так, чтобы идеальный проводник содержался в плоскости $x_2 = 0$, неоднородность располагалась в области $x_2 \geqslant 0$, а ось x_2 проходила через ее центр. При таком представлении неоднородности следует выбирать ее размеры так, чтобы значения электрического и магнитного полей на боковых и верхней стенке параллелепипеда достаточно сильно затухали в рамках выбранной точности. Геометрия задачи представлена на рисунке 4.1.

Будем считать, что сама неоднородность $\mathcal{V}$ представляет собой набор подобластей, в каждой из которых проводимость и диэлектрическая проницаемость среды постоянна (но различна в разных подобластях). Магнитная проницаемость внутри неоднородности совпадает с магнитной проницаемостью внешней среды. Внутри каждой подобласти справедлива система уравнений Максвелла, имеющая в гармоническом случае вид

$$\begin{aligned} \operatorname{rot} \mathbf{H} &= -i\omega\varepsilon\mathbf{E} + \mathbf{j}^0, \\ \operatorname{rot} \mathbf{E} &= i\omega\mu\mathbf{H}, \end{aligned} \tag{4.1}$$

где комплекснозначная ε определена равенством

$$\varepsilon = \varepsilon_d - i\sigma/\omega.$$

Источником электромагнитного поля является магнитный диполь, направленный вдоль оси x_2 и расположенный вне неоднородности в точке $(0, h, 0)$. Магнитное поле измеряется в нескольких точках в плоскости $x_2 = h$. Типичная частота источника 100 МГц, сопротивление внешней среды 10^3 Ом $\cdot$ м.

4.1.2. Интегральное уравнение. Определим *первичное поле* $\mathbf{E}^0, \mathbf{H}^0$ равенством

$$\begin{aligned} \operatorname{rot} \mathbf{H}^0 &= -i\omega\varepsilon_0 \mathbf{E}^0 + \mathbf{j}^0, \\ \operatorname{rot} \mathbf{E}^0 &= i\omega\mu \mathbf{H}^0, \end{aligned} \tag{4.2}$$

где ε_0 — диэлектрическая проницаемость внешней среды. Представим полное поле как сумму первичного и *аномальноого* поля

$$\begin{aligned} \mathbf{H} &= \mathbf{H}^0 + \mathbf{H}^s, \\ \mathbf{E} &= \mathbf{E}^0 + \mathbf{E}^s. \end{aligned} \tag{4.3}$$

Для аномального поля $\mathbf{E}^s, \mathbf{H}^s$ из (4.1), (4.2) и (4.3) имеем

$$\begin{aligned} \operatorname{rot} \mathbf{H}^s &= -i\omega\varepsilon_0 \mathbf{E}^s + i\omega(\varepsilon_0 - \varepsilon)\mathbf{E}, \\ \operatorname{rot} \mathbf{E}^s &= i\omega\mu \mathbf{H}^s. \end{aligned} \tag{4.4}$$

Поскольку $\operatorname{div} \mathbf{H}^s = 0$, то

$$\mathbf{H}^s = \operatorname{rot} \mathbf{A},$$

что вместе с предыдущим уравнением дает

$$\mathbf{E}^s = i\omega\mu\mathbf{A} + \operatorname{grad} \Phi.$$

$\mathbf{A}$ и Φ называются *векторным* и *скалярным потенциалом.* теперь первое уравнение из (4.4) переписывается в виде

$$\operatorname{rot}\operatorname{rot} \mathbf{A} = \kappa_0^2 \mathbf{A} - i\omega\varepsilon_0 \operatorname{grad} \Phi + i\omega\varepsilon_0 \left(1 - \frac{\varepsilon}{\varepsilon_0}\right) \mathbf{E},$$

где $\kappa_0 = \sqrt{\varepsilon_0\mu\omega^2}$.

В декартовых координатах $\operatorname{rot}\operatorname{rot} = \operatorname{grad}\operatorname{div} - \Delta$. Учитывая это, получаем

$$\operatorname{grad}(\operatorname{div} \mathbf{A} + i\omega\varepsilon_0\Phi) = \left(\Delta + \kappa_0^2\right) \mathbf{A} + i\omega\varepsilon_0 \left(1 - \frac{\varepsilon}{\varepsilon_0}\right) \mathbf{E}.$$

Учитывая условие калибровки (Лоренца)

$$\Phi = -\frac{1}{i\omega\varepsilon_0} \operatorname{div} \mathbf{A},$$

имеем

$$\mathbf{E}^s = \frac{1}{-i\omega\varepsilon_0} \left(\kappa_0^2 + \operatorname{grad}\operatorname{div}\right) \mathbf{A},$$

после чего векторный потенциал $\mathbf{A}$ должен удовлетворять векторному уравнению Лапласа

$$\left(\Delta + \kappa_0^2\right) \mathbf{A} = i\omega\varepsilon_0 \mathbf{J}, \tag{4.5}$$

где $\mathbf{J}$ определяется равенством

$$\mathbf{J} = \gamma \mathbf{E}, \qquad \gamma = \frac{\varepsilon}{\varepsilon_0} - 1.$$

Отсюда заключаем

$$\mathbf{A}(x) = -i\omega\varepsilon_0 \int_{\mathcal{V}} G(x,y)\mathbf{J}(y)\, dy,$$

где $G(x,y)$ — тензорная функция Грина. Для аномальных полей таким образом справедливы формулы

$$\mathbf{E}^s(x) = \left(\kappa_0^2 + \operatorname{grad}\operatorname{div}\right) \int_{\mathcal{V}} G(x,y)\mathbf{J}(y)\, dy, \tag{4.6}$$

$$\mathbf{H}^s(x) = -i\omega\varepsilon_0 \operatorname{rot} \int_{\mathcal{V}} G(x,y)\mathbf{J}(y)\, dy. \tag{4.7}$$

Для определения тока $\mathbf{J} = \gamma(\mathbf{E}^0 + \mathbf{E}^s)$ перепишем уравнение (4.6) в виде *объемного интегрального уравнения* [34]

$$\gamma^{-1}\mathbf{J}(x) - \left(\kappa_0^2 + \operatorname{grad}\operatorname{div}\right) \int\limits_{\mathcal{V}} G(x,y)\mathbf{J}(y)\, dy = \mathbf{E}^0, \qquad x \in \mathcal{V}. \tag{4.8}$$

Функция Грина для однородной среды без идеально проводящей плоскости имеет вид

$$G(x,y) = \begin{bmatrix} g(x,y) & & \\ & g(x,y) & \\ & & g(x,y) \end{bmatrix}, \qquad g(x,y) = \frac{e^{i\kappa_0\|x-y\|}}{4\pi\|x-y\|}. \tag{4.9}$$

В случае, когда в плоскости $x_2 = 0$ размещен идеальный проводник, функция Грина принимает вид

$$G(x,y) = \begin{bmatrix} g_1(x,y) & & \\ & g_2(x,y) & \\ & & g_3(x,y) \end{bmatrix}, \tag{4.10}$$

$$g_1(x,y) = g_3(x,y) = \frac{e^{i\kappa_0\|x-y\|}}{4\pi\|x-y\|} - \frac{e^{i\kappa_0\|x-y^*\|}}{4\pi\|x-y^*\|},$$

$$g_2(x,y) = \frac{e^{i\kappa_0\|x-y\|}}{4\pi\|x-y\|} + \frac{e^{i\kappa_0\|x-y^*\|}}{4\pi\|x-y^*\|},$$

где точка y^* получается "отражением" точки y относительно идеально проводящей плоскости

$$y^* = (y_1, y_2, y_3)^* = (y_1, -y_2, y_3).$$

4.2. Метод дискретизации

Для дискретизации интегрального уравнения (4.8) применяется метод Галеркина. В качестве базисных функций используются конечные элементы, кусочно-линейные по одному из направлений и кусочно-постоянные по двум другим (см. [34]).

Введем на области $\mathcal{V} = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ сетку, построенную декартовым произведением трех равномерных одномерных сеток:

$$\begin{array}{rcll} x_1^{j_1} & = & a_1 + (j_1 - 1)h_1, & h_1 = (b_1 - a_1)/n_1, \\ x_2^{j_2} & = & a_2 + (j_2 - 1)h_2, & h_2 = (b_2 - a_2)/n_2, \\ x_3^{j_3} & = & a_3 + (j_3 - 1)h_3, & h_3 = (b_3 - a_3)/n_3. \end{array} \tag{4.11}$$

Неизвестную функцию $\mathbf{J}$ будем приближенно искать в виде следующей комбинации

$$\begin{aligned} \mathbf{J}(x_1, x_2, x_3) \;=\; & \sum_{j_1=1}^{n_1-1} \sum_{j_2=1}^{n_2} \sum_{j_3=1}^{n_3} u^1_{j_1 j_2 j_3} \mathbf{F}^1_{j_1 j_2 j_3}(x_1, x_2, x_3) + \\ & \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2-1} \sum_{j_3=1}^{n_3} u^2_{j_1 j_2 j_3} \mathbf{F}^2_{j_1 j_2 j_3}(x_1, x_2, x_3) + \\ & \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \sum_{j_3=1}^{n_3-1} u^3_{j_1 j_2 j_3} \mathbf{F}^3_{j_1 j_2 j_3}(x_1, x_2, x_3) \end{aligned} \tag{4.12}$$

$$\mathbf{F}^1_{j_1 j_2 j_3} = \begin{bmatrix} \Phi^1_{j_1} \Psi^2_{j_2} \Psi^3_{j_3} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{F}^2_{j_1 j_2 j_3} = \begin{bmatrix} 0 \\ \Psi^1_{j_1} \Phi^2_{j_2} \Psi^3_{j_3} \\ 0 \end{bmatrix}, \quad \mathbf{F}^3_{j_1 j_2 j_3} = \begin{bmatrix} 0 \\ 0 \\ \Psi^1_{j_1} \Psi^2_{j_2} \Phi^3_{j_3} \end{bmatrix},$$

где $\Phi^k_{j_k}$ $(j_k = 1, \ldots, n_k - 1)$ — кусочно-линейные функции

$$\Phi^k_{j_k}(x_k) = \begin{cases} 1 - |x_k - x_k^{j_k+1}|/h_k, & |x_k - x_k^{j_k+1}| \leqslant h_k, \\ 0, & \text{при прочих } x_k, \end{cases}$$

а $\Psi^{j_k}_k$ $(j_k = 1, \ldots, n_k)$ — кусочно-постоянные функции

$$\Psi^k_{j_k}(x_k) = \begin{cases} 1, & x_k^{j_k} \leqslant x_k \leqslant x_k^{j_k+1}, \\ 0, & \text{при прочих } x_k. \end{cases} \qquad (k = 1, 2, 3)$$

Неизвестные коэффициенты из разложения (4.12) определяются, со-

гласно методу Галеркина, из следующей системы линейных уравнений

$$\int\limits_{\mathcal{V}}\left(\gamma^{-1}\mathbf{J}(x),\mathbf{F}^{k}_{i_1i_2i_3}(x)\right)dx- \\ -\int\limits_{\mathcal{V}}\left((\kappa_0^2+\operatorname{grad}\operatorname{div})\int\limits_{\mathcal{V}}G(x,y)\mathbf{J}(y)dy,\mathbf{F}^{k}_{i_1i_2i_3}(x)\right)dx= \\ =\int\limits_{\mathcal{V}}\left(\mathbf{E}^{0}(x),\mathbf{F}^{k}_{i_1i_2i_3}(x)\right)dx. \tag{4.13}$$

Полученная линейная система имеет вид

$$\left(\begin{bmatrix} D_1 & & \\ & D_2 & \\ & & D_3 \end{bmatrix}+\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}\right)\begin{bmatrix} u^1 \\ u^2 \\ u^3 \end{bmatrix}=\begin{bmatrix} f^1 \\ f^2 \\ f^3 \end{bmatrix}. \tag{4.14}$$

Блочно-диагональная часть матрицы соответствует вне-интегральному слагаемому из (4.13) и определяется равенством

$$\left(D_k\right)^{j_1j_2j_3}_{i_1i_2i_3}=\int\limits_{\mathcal{V}}\gamma^{-1}\left(\mathbf{F}^{k}_{i_1i_2i_3}(x),\mathbf{F}^{k}_{j_1j_2j_3}(x)\right)dx. \tag{4.15}$$

Плотная часть матрицы, отвечающая интегральному слагаемому из (4.13), определяется формулой

$$\left(A_{kl}\right)^{j_1j_2j_3}_{i_1i_2i_3}=-\int\limits_{\mathcal{V}}\left(\left((\kappa_0^2+\operatorname{grad}\operatorname{div})\int\limits_{\mathcal{V}}G(x,y)\mathbf{F}^{l}_{j_1j_2j_3}(y)dy\right),\mathbf{F}^{k}_{j_1j_2j_3}(x)\right)dx. \tag{4.16}$$

Метод эффективного вычисления подобных многомерных интегралов и некоторые смежные вопросы освещены в работе [41].

4.3. Специфика полученной матрицы

Именно действия с плотной частью матрицы требуют основных затрат для вычисления элементов и проведения операции умножения матрицы на вектор. Поэтому вопрос о ее эффективном хранении и умножении представляет наибольший практический интерес. Интегральная часть уравнения (4.13), представленная матрицей (4.16), обладает определенной спецификой, учитывая которую, мы можем значительно снизить затраты на вычисление, хранение и умножение на эту матрицу.

В случае, когда в плоскости $x_2 = 0$ размещен сверхпроводник, функция Грина (4.10), представима в виде суммы функции, зависящей от разности координат двух точек и функции, зависящей от разности и суммы координат двух точек. Для матрицы задачи это означает, что

$$\left(A_{kl}\right)_{i_1 i_2 i_3}^{j_1 j_2 j_3} = A_{kl}^{(1)}(i_1 - j_1, i_2 - j_2, i_3 - j_3) + A_{kl}^{(2)}(i_1 - j_1, i_2 + j_2, i_3 - j_3). \quad (4.17)$$

Матрица, элементы которой зависят только от разности номера строки и столбца, называется *теплицевой.* Матрица, элементы которой зависят от суммы индекса строки и столбца, называется *ганкелевой.* Итак, в нашем случае матрица задачи представляется в виде суммы двух блочных 3×3 матриц, каждый блок которых обладает следующими свойствами:

- имеет трехуровневую структуру;
- на верхнем уровне блоки являются блочно-теплицевыми
- на втором уровне для матрицы $A^{(1)}$ блоки блочно-теплицевы, для матрицы $A^{(2)}$ блочно-ганкелевы;
- на нижнем уровне случаях блоки являются теплицевыми матрицами.

Матрицы, обладающие подобными структурами, рассматривались в [34, 20, 21, 55].

Как известно, для хранения теплицевой матрицы размера n необходимо $O(n)$ ячеек памяти, а умножение на нее можно произвести за $O(n \log(n))$ арифметических действий [49]. То же самое справедливо и для ганкелевой матрицы, так как перестановкой строк и столбцов ее можно привести к теплицевой. Таким образом, матрицу нашей задачи можно хранить в $O(n_1 n_2 n_3)$ ячеек памяти и умножать на вектор за $O(n_1 n_2 n_3 \log(n_1 n_2 n_3))$ арифметических операций, пользуясь процедурами быстрого умножения Фурье (см. [45]).

Имея оптимизированную процедуру умножения на матрицу, мы можем найти решение линейной системы (4.14) с помощью какого-нибудь итерационного метода (в нашей работе применяется метод минимальных невязок). По найденному решению $\mathbf{J}$ строятся поля в точках наблюдения, согласно формулам (4.6) и (4.7).

Таблица 4.1. Количество необходимой памяти, в зависимости от размера сетки. Данные приведены в предположении, что память, необходимая на хранение одного элемента составляет $\mathtt{mem}(1) = 16\mathrm{b}$. Матрица $A^{(1)}$ занимает порядка $9 \cdot 8 \cdot N$ ячеек памяти.

n	16	32	64	128	256
$N = n^3/2$	2^{11}	2^{14}	2^{17}	2^{20}	2^{23}
$\mathtt{mem}(N)$	32 Kb	256 Kb	2 Mb	16 Mb	128 Mb
$\mathtt{mem}(A^{(1)})$	2.3 Mb	18.4 Mb	144 Mb	1.1 Gb	9.2 Gb

4.4. Параллельный алгоритм

Предварительные расчеты, проведенные по указанному алгоритму, показывают, что точность вычисления величин полей заметно ухудшается в случае, когда плоскость, содержащая источник и точки наблюдения, приближается к неоднородности. В этом случае для поддержания определенной точности ответа, приходится увеличивать количество элементов сеток (4.11), что приводит к резкому росту памяти, необходимого для хранения матриц и векторов задачи и арифметических операций, необходимых для решения линейной системы.

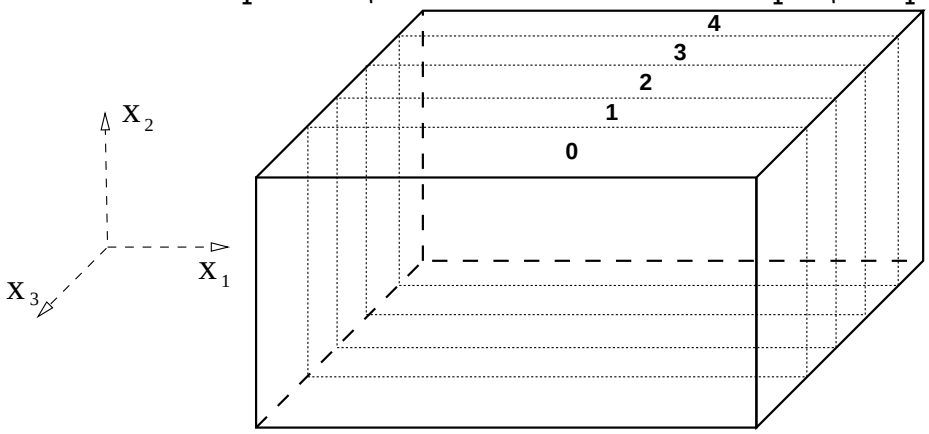
Рассмотрим в качестве модельного примера неоднородность $\mathcal{V}$ как параллелепипед $[-a, a] \times [H, H + a/2] \times [-a, a]$. Сетку для дискретизации выберем размером $n \times n/2 \times n$. В таблице 4.1 приведена зависимость количества памяти, необходимой для хранения векторов и матриц, использованных при расчете, от размера выбранной сетки.

На персональных компьютерах, доступных в текущее время, количество оперативной памяти не превышает 2 Гб. Таким образом, провести интересующие нас расчеты на персональной станции возможно только при числе точек сетки $n = 16, 32, 64$. Для расчетов с большими значениями числа точек n необходимо привлекать многопроцессорные станции, обладающие большими ресурсами оперативной памяти и большей вычислительной мощностью. Мы предлагаем параллельную версию описанного алгоритма, по-прежнему, ориентируясь на использование *кластерных станций*, основные особенности которых мы описывали в разделе 1.2.1.

Для параллельной реализации описанного метода необходимо распределить по np процессорам следующие объекты:

- векторы размера $3 \cdot n_1 n_2 n_3$, предназначенные для хранения
  - правой части уравнения (4.14)

Рис. 4.2. Схема размещения неизвестных по процессорам



  - вектора неизвестных
  - вспомогательных векторов итерационного метода

- составные части матрицы A (см. (4.17))
  - $A^{(1)}$ состоит из 3×3 трехуровневых трижды теплицевых
  - $A^{(2)}$ состоит из 3×3 трехуровневых теплиц-ганкель-теплицевых

  В силу указанной специфики количество памяти, необходимой для хранения каждой из матриц, пропорционально их размеру и составляет

$$3 \cdot 3 \cdot (2n_1 - 1)(2n_2 - 1)(2n_3 - 1) \sim 72n_1n_2n_3$$

  для каждой.

Векторы распределяются по процессорам путем "разрезания" множества неизвестных на np частей вдоль оси x_3 (рис. 4.2). Конкретнее, определим числа

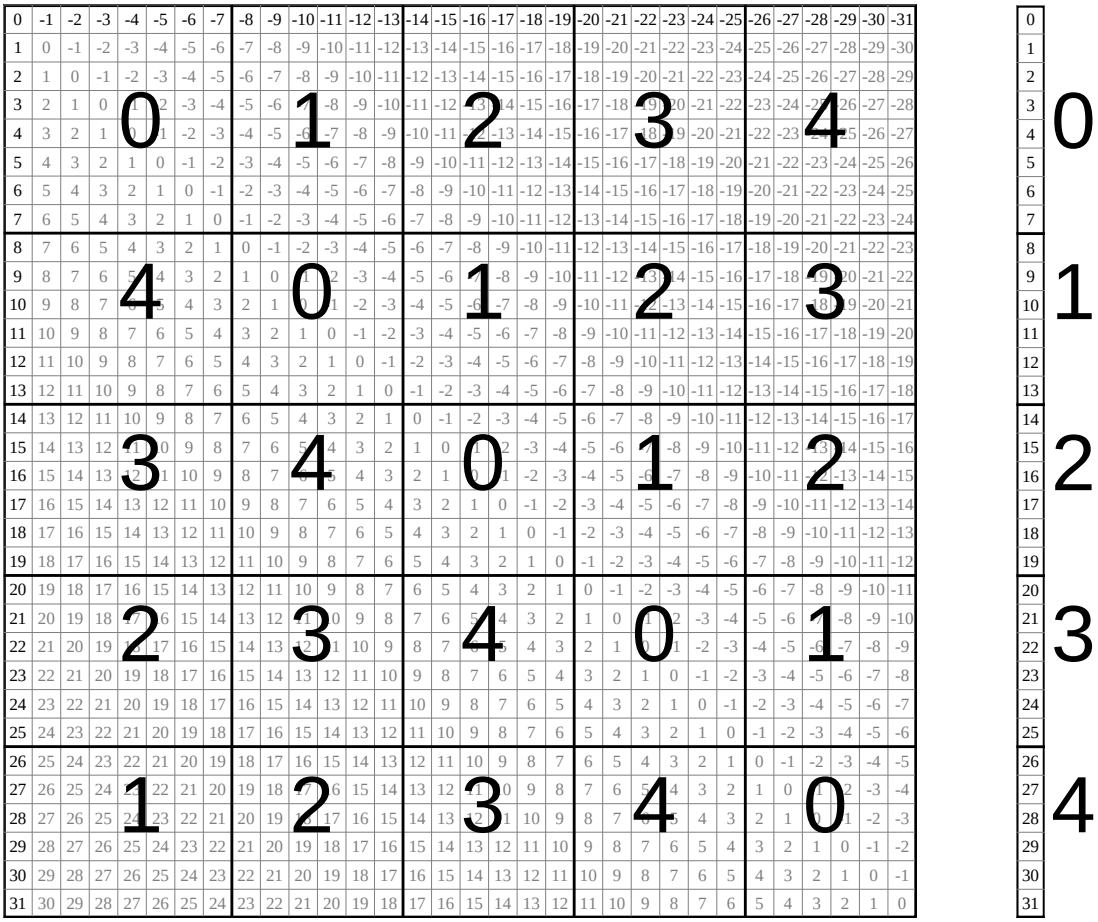
$$k_3 = \lfloor n_3/np \rfloor , \qquad r_3 = n_3 - k_3n_3$$

и разобьем сетку вдоль оси x_3 на np частей, в каждую из которых войдет $n_3^{\{p\}}$ точек:

$$n_3^{\{0\}} = k_3 + r_3, \qquad n_3^{\{p\}} = k_3, \qquad \text{для прочих} \quad p = 1, \ldots, np-1.$$

Таким образом, "корневой" процессор с индексом 0 берет на себя большую из частей сетки x_3, если ее невозможно распределить равномерно. На каждом из процессоров хранится, с учетом трехуровневой структуры $3n_1n_2n_3^{\{p\}}$ элементов вектора.

Рис. 4.3. Распределение элементов матрицы и вектора по процессорам



Параметры $np = 5$, $n_3 = 32$, $k_3 = 6$, $r_3 = 2$, $n_3^{\{0\}} = 8$, $n_3^{\{p\}} = 6$, $p \geqslant 1$.

Определим теперь схему размещения на процессорах элементов матрицы. Поскольку матрица обладает трехуровневой структурой, а разрезание множества неизвестных происходит только вдоль оси x_3, можно рассматривать только разрезание верхнего уровня. На верхнем уровне, отвечающем сетке вдоль x_3, все 3×3 блока матриц $A^{(1)}$ и $A^{(2)}$ являются теплицевыми, поэтому можно поставить вопрос об эффективном распределении по процессорам элементов теплицевой матрицы, не учитывая тот факт, что ее элементы сами по себе являются двухуровневыми матрицами специальной структуры. Поскольку для теплицевой матрицы элементы зависят только от разности индексов строки и столбца $a_{ij} = a_{i-j}$, реально мы распределяем по процессорам вектор длины $2n - 1$, однако для эффективного умножения на распределенную матрицу структура распределения этого вектора должна быть согласована с распределением элементов вектора неизвестных и

правой части, описанном выше. Проиллюстрируем это распределение картинкой, см. рис. 4.3.

Корневой процессор размещает у себя один сегмент матрицы, все прочие по два. Индексы $i_3^{\{0\}} = i - j$ элементов a_s, размещенных на корневом процессоре, следующие

$$i_3^{\{0\}} = -n_3^{\{0\}} + 1, \ldots, n_3^{\{0\}} - 1.$$

Прочие процессоры размещают у себя по два сегмента элементов, отвечающие индексам

$$i_3^{\{p\}} = (-p-1)k_3 - r_3 + 1, \cdots, (-p+1)k_3 - 1,$$
$$i_3^{\{p\}} = (np-p-1)k_3 + 1, \cdots (np-p+1)k_3 + r_3 - 1.$$

Приведем теперь умножения на распределенную матрицу.

Алгоритм 7 $y := A \cdot x$.

Каждому процессору присвоен уникальный номер p, лежащий в пределах от 0 до $np-1$. Алгоритм представлен для процессора с номером p, хотя выполняется, конечно же, на всех процессорах одновременно.

do $ip = 0, np-1$ //расчет локальной части вектора y

1. [irecv] предоставить локальный вектор $\tilde{x}$ для приема локальной части вектора x с соседнего процессора;
2. [isend] $x \to \tilde{x}$ выслать локальную часть вектора x процессору с индексом $p+1 \bmod np$ (“вниз”);
3. [A*x] $\tilde{y} := A \cdot x$ произвести умножение подходящей локальной части матрицы на локальную часть вектора x (подробнее ниже);
4. [allreduce] $y^{\{ip\}} := \operatorname{sum}\limits_{all} \tilde{y}$ просуммировать локально вычисленные векторы $\tilde{y}$ на процессор ip;
5. [wait isend] дождаться завершения операции 2, если необходимо;
6. [wait irecv] дождаться завершения операции 1, если необходимо.

enddo

геометрия	
размер неоднородности	$(0.8, 0.2, 0.8)$
уровень неоднородности	$H = 0.06$
положение источника	$(0, h, 0)$
тип источника	y–диполь
частота источника	100 МГц

электрические параметры		
параметр	среда	неоднор.
сопротивление	10^3	50
магн. прониц.	0	
диэл. прониц.	1	

точки наблюдения				
1	2	3	4	5
$(0.02, h, 0.0)$	$(0.04, h, 0.0)$	$(0.06, h, 0.0)$	$(0.08, h, 0.0)$	$(0.10, h, 0.0)$

Таблица 4.2. Параметры теста

Поскольку локальные части матриц, хранимые на каждом из процессоров, как и вся матрица, обладают свойством $a_{ij} = a_{i-j}$, следовательно, являются теплицевыми матрицами (для процессора $p = 0$) или могут быть расширены до теплицевой матрицы (для прочих процессоров), для локального умножения мы также применяем алгоритм почти линейной сложности. Подробнее процедура локального умножения описана в работе [72].

Таким образом, мы описали процедуры умножения для всех процессоров на все хранимые ими части исходной матрицы. Тем самым, пункт 3 из алгоритма умножения на распределенную матрицу, а вместе с ним и сам алгоритм полностью описан.

Построение параллельного алгоритма для умножения на блочно-диагональную часть матрицы (4.15) не представляет труда. Таким образом, умножение на матрицу задачи полностью параллелизовано. На основе параллельного алгоритма умножения на матрицу производится решение задачи (4.14) с помощью какого-либо итерационного метода. В нашей работе мы использовали метод минимальных невязок [33].

4.5. Численные эксперименты

На нескольких примерах продемонстрируем, в какой степени применение параллельной версии метода позволило улучшить получаемые результаты. Все приводимые далее примеры так или иначе моделируют процесс «разведывания» среды геологическим зондом, который излучает электромагнитное поле, и измеряет его в нескольких точках наблюдения и по полученным данным должен определить параметры среды.

В данной работе мы приводим достаточно краткое и качественное

описание проведенных экспериментов. Более детально они представлены в работе в работе [72].

Суть первого эксперимента в следующем. Измеряются значения магнитных полей в 5 точках наблюдения, расположенных на одном уровне с источником. Высота этого уровня изменяется в наших экспериментах в пределах от $h = 1\,\text{см}$ до $h = 5.9\,\text{см}$. В измеренных полях рассматриваемой величиной является мнимая часть проекции поля, направленной вдоль оси диполя-источника, то есть вдоль оси x_2 (эту ось мы в дальнейшем будем обозначать также y.) Нас интересует точность измерения этой компоненты, причем так как первичное поле вычисляется, в принципе, достаточно просто и надежно, мы будет исследовать точность измерения аномального поля $\mathfrak{I}H^s_y$. В качестве меры точности принимается критерий *внутренней сходимости,* выраженный через два измерения одной и той же величины, произведенные при двух значениях числа точек дискретизации n, отличающихся вдвое.

$$\varepsilon_{\text{im}}(n) = \frac{|\mathfrak{I}H^s_y(n) - \mathfrak{I}H^s_y(n/2)|}{\left(|\mathfrak{I}H^s_y(n)| + |\mathfrak{I}H^s_y(n/2)|\right)/2}$$

Численные параметры эксперимента приведены в таблице 4.2.

Из таблицы 4.1 следует, что в том случае, когда расчет по обсуждаемому методу происходит на стандартной персональной вычислительной платформе с количеством памяти до $2\,\text{Гб}$, количество элементов сетки ограничено величиной $n = 64$. На картинке приведены результаты следующего эксперимента: мы фиксируем две сетки с количеством узлов $n = 32$ и $n = 64$ и, приближая плоскость, содержащую источник и точки наблюдения, к неоднородности, измеряем точность вычисления интересующих нас компонент полей. Рисунок 4.4 показывает, что при приближении к неоднородности на расстояние ближе $10\,\text{мм}$, то есть при $h \geqslant 50\,\text{мм}$, происходит резкое ухудшение точности измерения компонент полей практически для всех точек наблюдения. Для наших практических вычислений необходимо поддерживать точность вычисления компонент полей хотя бы на уровне 10%. Для того, чтобы получить возможность производить расчеты с такой точностью в областях, более близких к неоднородности, нам приходится увеличивать количество точек сетки n, что вызывает необходимость использовать многопроцессорные вычислительные системы.

Увеличение числа точек сетки приводит к повышению точности расчета, что проиллюстрировано экспериментами (рис. 4.5).

Возможность увеличить количество узлов сетки, конечно же, не отменяет ухудшения точности расчета при приближении плоскости из-

Рис. 4.4. Падение точности вычисления поля при приближении плоскости измерения к неоднородности

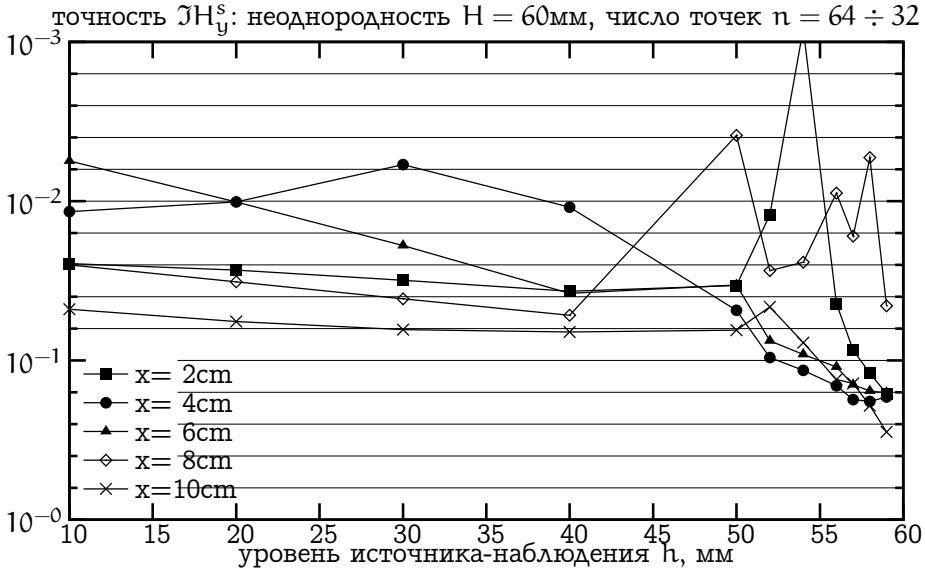


Рис. 4.5. Повышение точности расчета при измельчении сетки

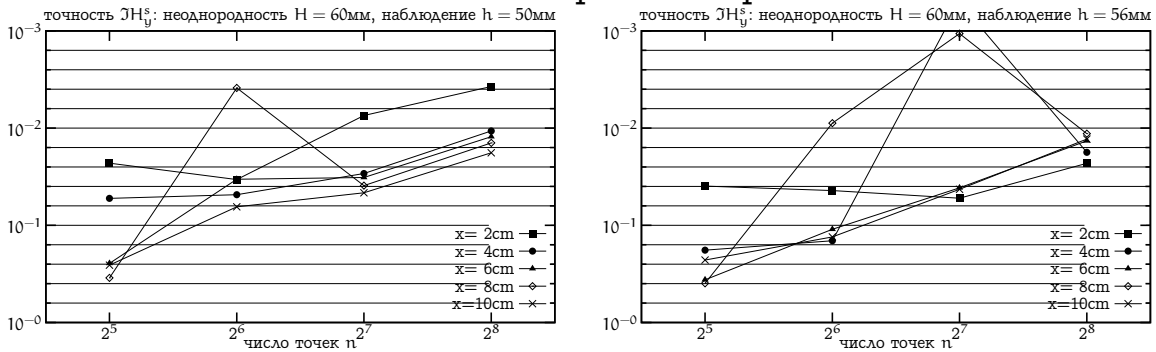


Рис. 4.6. Улучшенные значения точности для вычисляемых полей

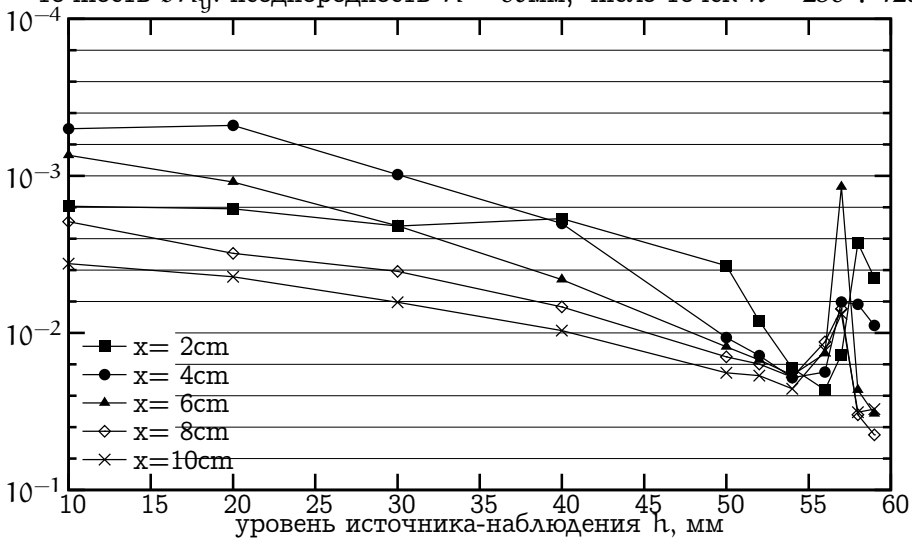
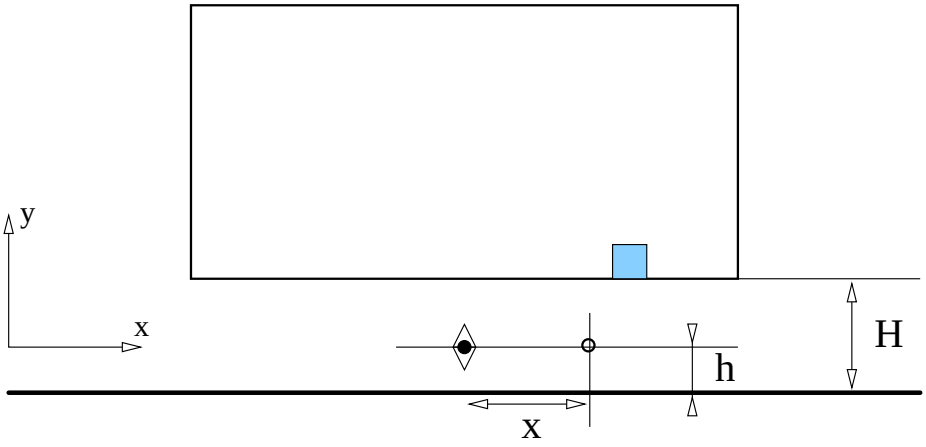


Таблица 4.3. Число итераций, необходимых для решения задачи при разных значениях числа узлов n. Параметр остановки $\varepsilon = 0.05$

h, мм	10	20	30	40	50	52	54	56	57	58	59
$n = 64$	2	2	2	2	4	6	6	7	8	8	8
$n = 256$	2	2	2	2	2	2	2	2	3	5	7

Рис. 4.7. Неоднородность с небольшим вкраплением



мерения к неоднородности, но заметно смягчает ее, делая возможным проведения измерений с требуемой точностью на расстояниях порядка 1 мм от неоднородности (рис. 4.6).

Число итераций, необходимых для обращения матрицы задачи, заметно увеличивается при приближении плоскости источников-приемников к неоднородности. При этом этот рост смягчается на больших размерах сетки, что проиллюстрировано таблицей 4.3.

Второй эксперимент мы посвятим исследованию разрешающей способности прибора. Рассмотрим случай, когда неоднородность по существу является таковой, то есть содержит области с различными электрическими свойствами. Для наглядности остановимся на следу-

электрические параметры			
параметр	внеш. среда	неоднор.	вкрапление
сопротивление, Ом · м	10^3	50	2
магн. проницаемость	0	0	0
диэл. проницаемость	1	1	1

Таблица 4.4. Электрические параметры теста

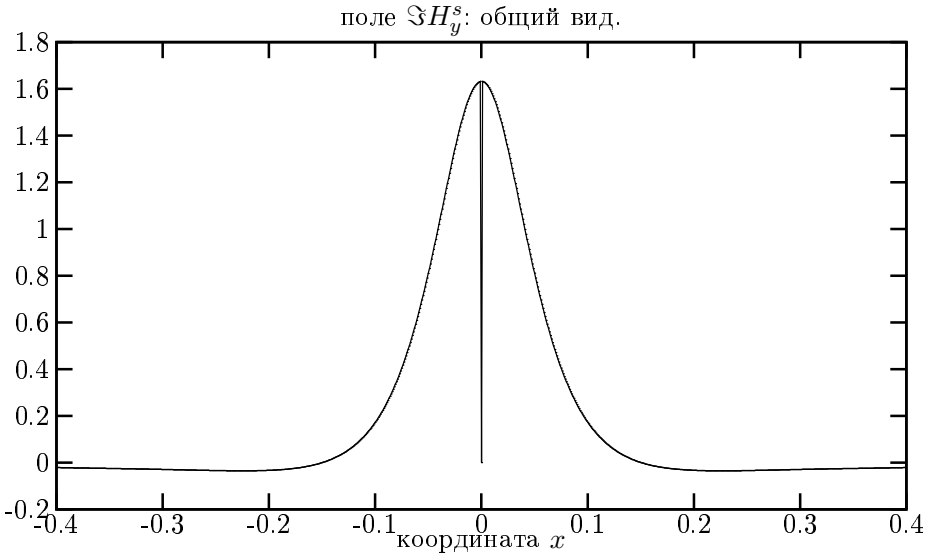


Рис. 4.8. Общий вид зависимости измеряемой компоненты аномального поля от координаты x

ющем случае: неоднородность представляет собой (как и в прошлом эксперименте) параллелепипед с небольшим «вкраплением» $\tilde{\mathcal{V}}$ (см. рис. 4.7) Соотношения электрических параметров приведены в таблице 4.4.

Цель проводимых измерений состоит в том, чтобы с помощью величин магнитных полей, измеренных в удачно для этого выбранной плоскости наблюдения, «почувствовать» наличие вкрапления и максимально точно определить его месторасположение. Пример, рассматриваемый нами, является в определенном смысле модельным, но позволяет качественно понять проблему, для решения которой нам необходимо увеличивать точность дискретизации.

Итак, при зафиксированной геометрии, проведем измерения величин интересующих нас компонент магнитных полей (как и прежде, это мнимая часть компоненты аномального поля $\Im H_y^s$) в плоскости измерения $x_2 = h$ на отрезке $x = [-0.4, 0.4], y = h, z = 0$. Общий вид аномального поля представлен на картинке 4.8

Измеряемая компонента поля весьма сильно изменяется на рассматриваемом отрезке, поэтому для рассмотрения поведения полей вблизи вкрапления мы будем приводить графики с подходящим выбором рассматриваемого диапазона значений поля. Рассмотрим графики на рис. 4.9-4.11. На них изображено поведение исследуемой компоненты на рассматриваемом отрезке, пунктиром отмечена область, которая соответствует вкраплению $\tilde{\mathcal{V}}$.

По первой серии графиков, сделанной при количестве узлов сетки $n = 64$ с использованием обычной однопроцессорной рабочей станции,

Рис. 4.9. Точность и чувствительность полей, сетка $n = 64$

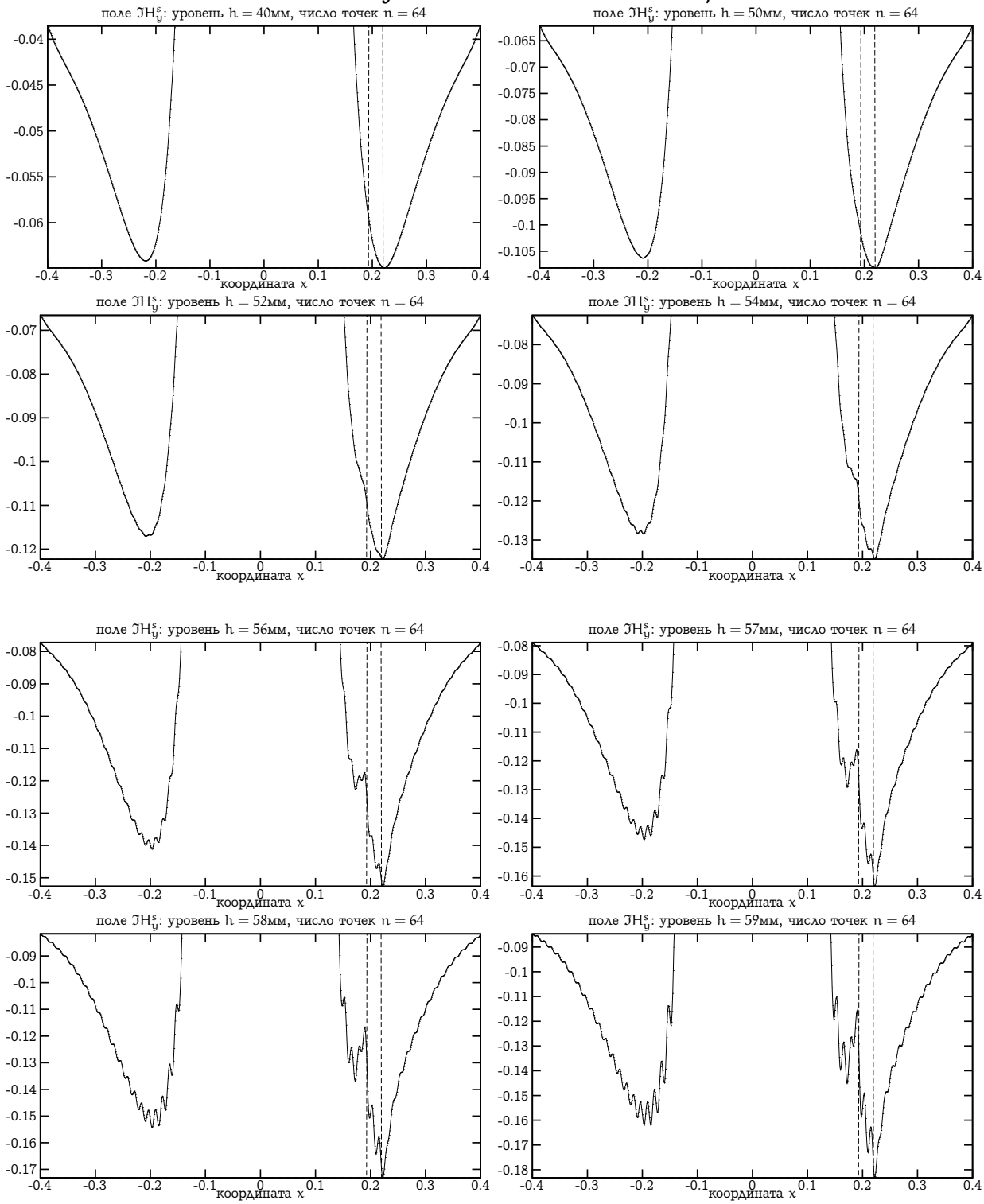


Рис. 4.10. Точность и чувствительность полей, сетка $n = 128$

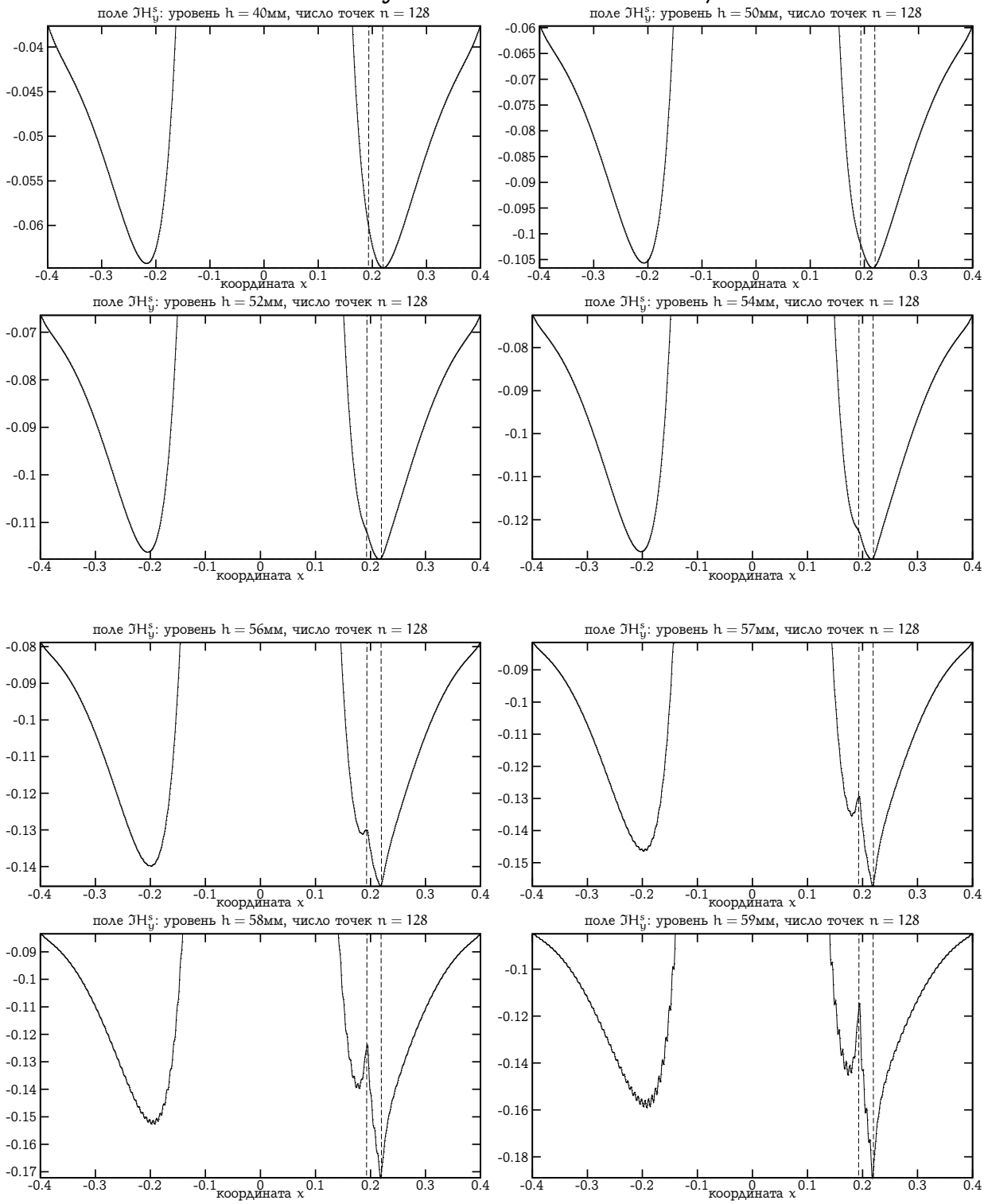
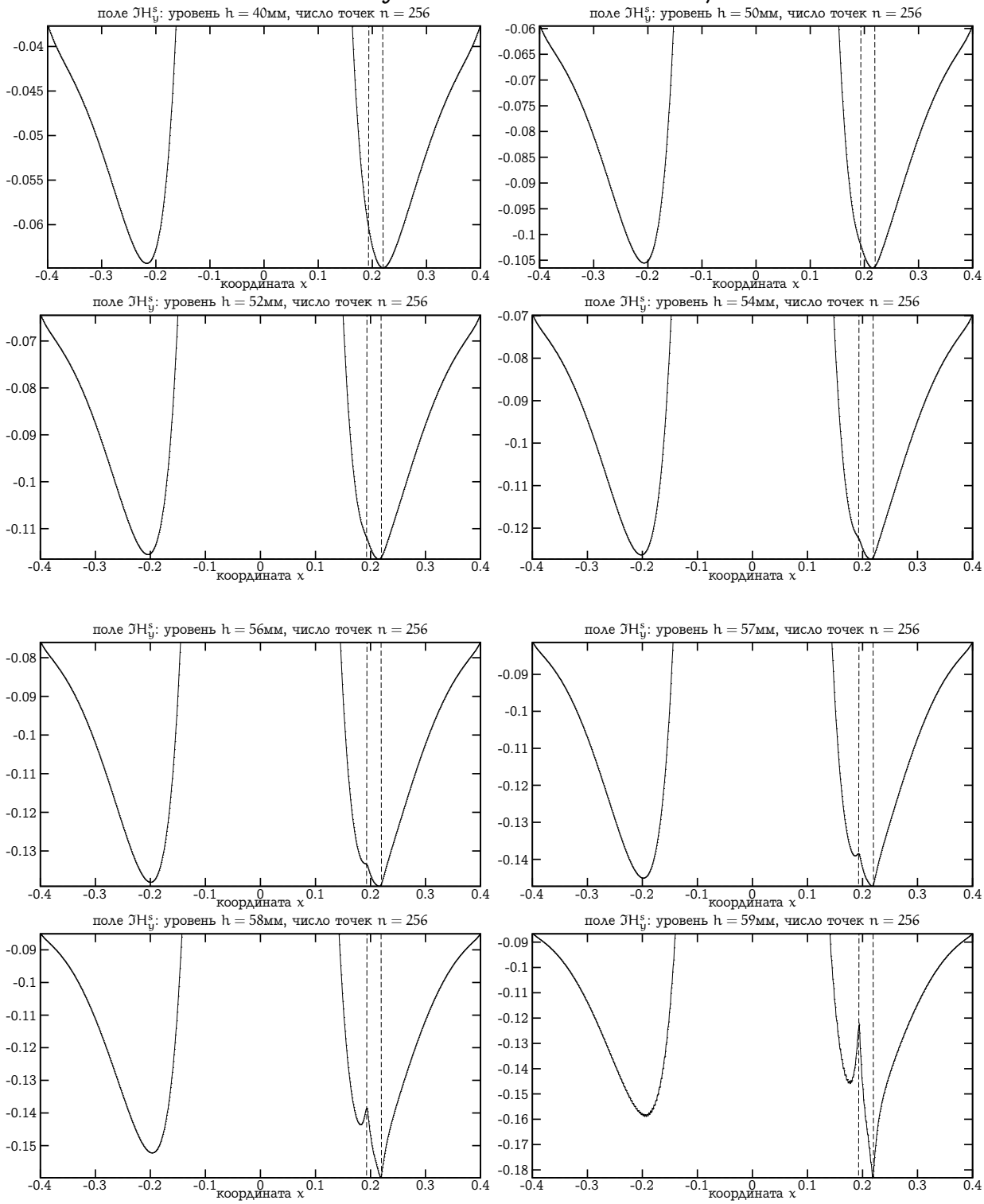


Рис. 4.11. Точность и чувствительность полей, сетка $n = 256$



можно сделать следующие наблюдения.

- На существенном расстоянии плоскости измерения от неоднородности ($h = 40$ мм, то есть в 20 мм от неоднородности) обнаружить по измеренным полям наличие вкрапления практически невозможно.

- При приближении плоскости измерения к неоднородности график измеряемой компоненты от координаты x начинает претерпевать изменения в районе, соответствующем положению вкрапления. Эти изменения слегка заметны при $h = 50$ мм (плоскость измерения в 10 мм от неоднородности), и становятся все более заметны при дальнейшем приближении к неоднородности.

- Уже при $h = 52 - 54$ мм и особенно при $h = 56$ мм изменения поведения графика в районе вкрапления становятся достаточно заметны, но измерения содержат уже достаточно сильные погрешности, что выражается в существенных их биениях, дискретизационных шумах, которые могут быть восприняты как наличие других вкраплений и тем иным способом внести ошибки в процедуру локации неоднородности. Возникающие биения делают невозможным построить сколь-либо точные предположения относительно положения вкрапления $\tilde{V}$.

- Дальнейшее приближение плоскости измерения к неоднородности ($h = 58 - 59$ мм) приводит к тому, что результаты измерений вследствие возникающих погрешностей становятся неприменимыми для анализа.

Сделанные наблюдения приводят к выводу о необходимости увеличивать точность измерений с целью подавить биения и добиться более аккуратных измерений полей вблизи неоднородности. Для достижения этой цели приходится увеличивать количество узлов сетки дискретизации, что уже возможно реализовать только на параллельной платформе. Вторую серию экспериментов проведем с количеством узлов $n = 128$ (рис. 4.10).

Видим, что определение местоположения включения $\tilde{V}$ по-прежнему затруднено ввиду наличия дискретизационных «шумов» , но с определенной точностью все же возможно при выборе положения плоскости измерения на уровне $h = 56\text{-}57$ мм от неоднородности. Проблема, однако, состоит в том, что положение плоскости измерения требуется указать до проведения реальных экспериментов, поэтому хотелось бы

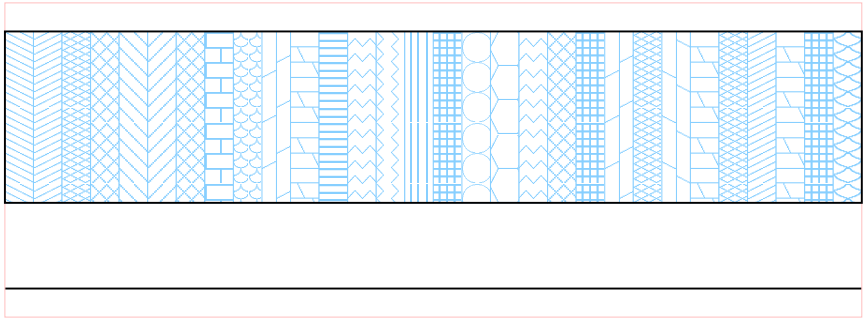


Рис. 4.12. Область зондирования

расширить диапазон возможных значений h, при которых можно полученные результаты измерений могут считаться достаточно точными и вместе с тем, достаточно чувствительными для решения обратной задачи.

Рассматривая результаты третьей серии экспериментов ($n = 265$, рис. 4.11), можно заметить, что положение неоднородности $\tilde{V}$, хорошо просматривается по графикам измеренных компонент полей при положении плоскости источников на уровне $h = 56 - 59$ мм . Это означает, что точность решения прямой задачи, то есть определения величин полей при заданной геометрии неоднородности, при таком значении параметров уже достаточно высока для того, чтобы на основе алгоритма, реализующего прямую задачу, строить алгоритм для решения обратной задачи, т.е. определения геометрии и электрических параметров среды внутри неоднородности.

4.6. Пример решения обратной задачи

Рассмотрим обратную задачу (задачу *зондирования*): по известным измерениям величин аномальных магнитных полей (показаниям измеряющего прибора, или же *зонда*) восстановить электрические параметры среды. Для простоты будем считать, что магнитная проницаемость среды равно 0, диэлектрическая равна 1, и вопрос сводится только к определению профиля сопротивления в неоднородности.

4.6.1. Приближение Борна. Методы решения задачи зондирования часто основываются на интегральных уравнениях (4.6),(4.7), в которых значения полей $\mathbf{E}^s$ и $\mathbf{H}^s$ рассматриваются как известные данные, а неизвестная $\gamma(y) = \varepsilon(y)/\varepsilon_0 - 1$ содержит информацию о распределении сопротивления в неоднородности. Перепишем уравнения (4.6),(4.7) в более простом виде

$$\begin{aligned} \mathbf{H}^s(x) &= G_H(\gamma(\mathbf{E}^0 + \mathbf{E}^s)), \\ \mathbf{E}^s(x) &= G_E(\gamma(\mathbf{E}^0 + \mathbf{E}^s)). \end{aligned} \tag{4.18}$$

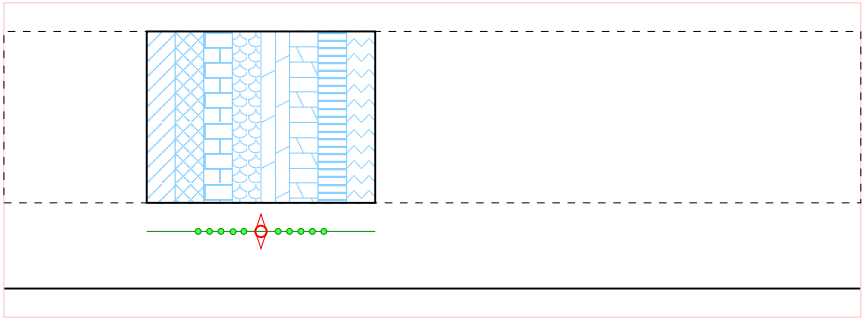


Рис. 4.13. Расчетная область для прямой задачи

Поскольку для непосредственного решения этой задачи требуются чрезвычайно высокие вычислительные ресурсы, обычно используются какие-либо упрощенные аналоги этих же интегральных уравнений, позволяющие находить γ без решения дополнительных прямых задач. Аппроксимация Борна [10] основывается на предположении

$$|\mathbf{E}^s(y)| \ll |\mathbf{E}^0(y)|, \qquad y \in \mathcal{V},$$

то есть на пренебрежимой малости рассеянного поля по сравнению с первичным, что достаточно хорошо выполняется в нашей задаче. Таким образом, значениями $\mathbf{E}^s$ в правой части равенств (4.18) можно пренебречь, вследствие чего мы получаем формулы

$$\mathbf{H}^s(x) = G_H(\gamma \mathbf{E}^0), \qquad \mathbf{E}^s(x) = G_E(\gamma \mathbf{E}^0), \tag{4.19}$$

непосредственно определяющие $\mathbf{E}^s$ и $\mathbf{H}^s$ как линейные функции от γ.

Теперь неизвестную γ можно определить, решив всего одну задачу наименьших квадратов

$$\min_{\gamma} \left\| G_H(\mathrm{obs}_i)\mathbf{E}^0(\mathrm{src}_j)\gamma - (\mathbf{H}^s)(\mathrm{obs}_i, \mathrm{src}_j) \right\|_2, \tag{4.20}$$

где src_j определяет всевозможные положения источника, obs_i определяет положения всех точек измерения поля, а норма $\| \circ_{ij} \|_2$ является обычной нормой вектора данных, заданного двойным индексом i, j. Общее число данных, определяющее число строк матрицы задачи наименьших квадратов, равняется числу различных положений источника (зонда), умноженному на число измерений значений магнитных полей, проводимых зондом в различных точках. Общее число неизвестных, определяющее число столбцов матрицы задачи наименьших квадратов, равняется числу ячеек, на которые разбита неоднородность, и внутри которых среда полагается однородной.

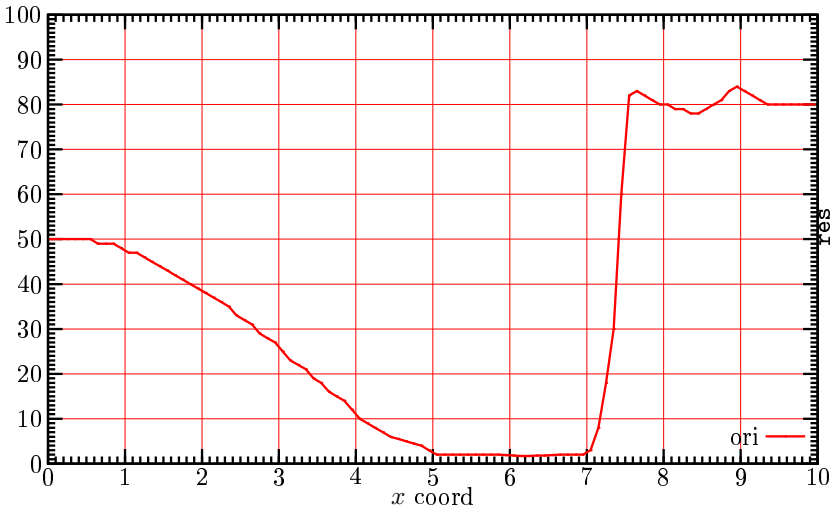


Рис. 4.14. Реальный профиль сопротивления

4.6.2. Горизонтальное зондирование. *Горизонтальным зондированием* мы будем называть решение обратной задачи, то есть определение сопротивления среды, в случае, когда оно меняется только вдоль одной (например, горизонтальной) оси (рис. 4.12). Рассматриваемый участок среды в реальных задачах может оказаться достаточно протяженным. Решив большое количество прямых задач, отвечающих разным положениям источника и измеряющей плоскости, мы можем смоделировать измерения, проведенные прибором, движущемся вдоль среды. При решении каждой прямой задачи мы пользуемся описанным выше методом, основанном на модели локальной неоднородности, полагая, таким образом, что удаленные участки среды не влияют на значение измеряемых полей (рис. 4.13).

Как и прежде, при решении прямой и обратной задачи мы считаем, что плоскость $y = 0$ содержит идеальный проводник, а источником является y-диполь. Заданный точный профиль сопротивления среды показан на рисунке 4.14.

При рассмотрении среды, сопротивление которой изменяется только вдоль горизонтальной оси, число неизвестных есть просто число вертикальных «слоев» , в которых среда полагается однородной. Количество этих «слоев» и другие параметры эксперимента представлены в таблице 4.5. Число шагов измеряющего зонда согласовано с числом слоев с неизвестными сопротивлениями.

Результаты численного решения задачи горизонтального зондирования представлены на рис. 4.15. Мы можем с удовлетворением констатировать, что как при положении измеряющей плоскости $h = 55$ мм, так и при положении $h = 59$ мм решение обратной задачи, основанное на На том же рисунку приведен график точности зондиро-

полная длина рассеивателя	$x_{len} = 10\,\text{m}$
шаг изменения сопротивления	$0.1\,\text{m}$
шаг измеряющего зонда	$0.1\,\text{m}$
число точек наблюдения	40
шаг между точками наблюдения	0.01

Таблица 4.5. Параметры задачи горизонтального зондирования

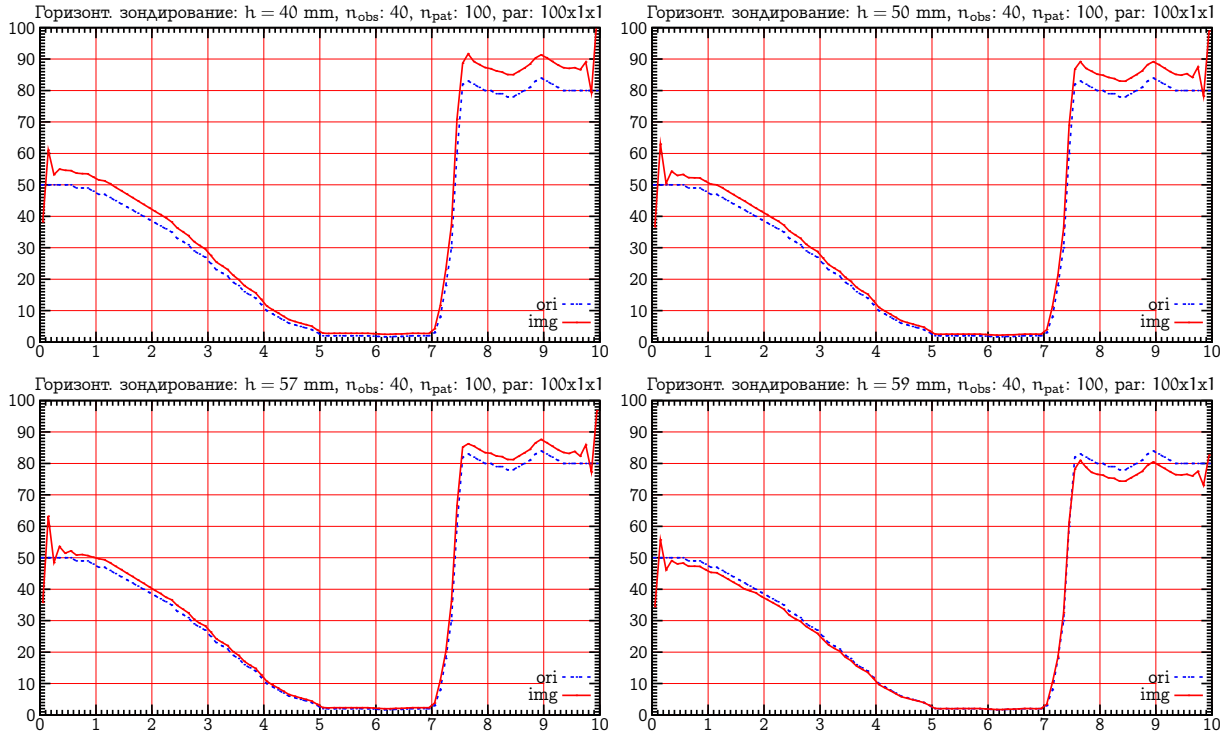
вания для всех проведенных экспериментов. Следует отметить, что в областях с высоким сопротивлением точность определения сопротивления среды достаточно хорошая при различных значениях h, в то же время для качественного (с точностью порядка 10%) определения сопротивления в областях, где оно мало, требуется максимально приблизить измеряющую плоскость к неоднородности, вплоть до уровня $h = 59\,\text{мм}$.

4.6.3. Двумерное зондирование. *Двумерным* зондированием мы назовем определение сопротивления среды в предположении, что оно меняется по вертикали и горизонтали. При этом число неизвестных резко возрастает, что затрудняет расчет. Кроме того, значения электрических параметров в точках, расположенных далеко от плоскости измерения, мало влияют на значения наблюдаемых компонент и поэтому восстанавливаются с меньшей точностью. Эксперименты по двумерному зондированию полностью описаны нами в работе [74]. Здесь мы ограничимся лишь одним графиком 4.16.

4.7. Применение трилинейной крестовой аппроксимации для сжатия данных

Предварительные опыты, продемонстрированные в разделе 3.3 для модельного интегрального уравнения на неравномерной сетке в кубе, показывают, что применение тензорных аппроксимаций к этой задаче может быть крайне эффективным. В таблице 4.6 мы приводим степень сжатия блоков матрицы электродинамической задачи (4.16). Можно заметить, что использование тензорных аппроксимаций даже на неравномерной сетке требует значительно меньших памяти и вычислительных ресурсов, чем использование специальной структуры матрицы при решении задачи на равномерной сетке. Работа в этом направлении, конечно же, не завершена — еще требуются серьезные усилия для того, чтобы изучить свойства возникающих итерационных методов и технику построения эффективных переобусловливате-

Рис. 4.15. Результаты и точность горизонтального зондирования



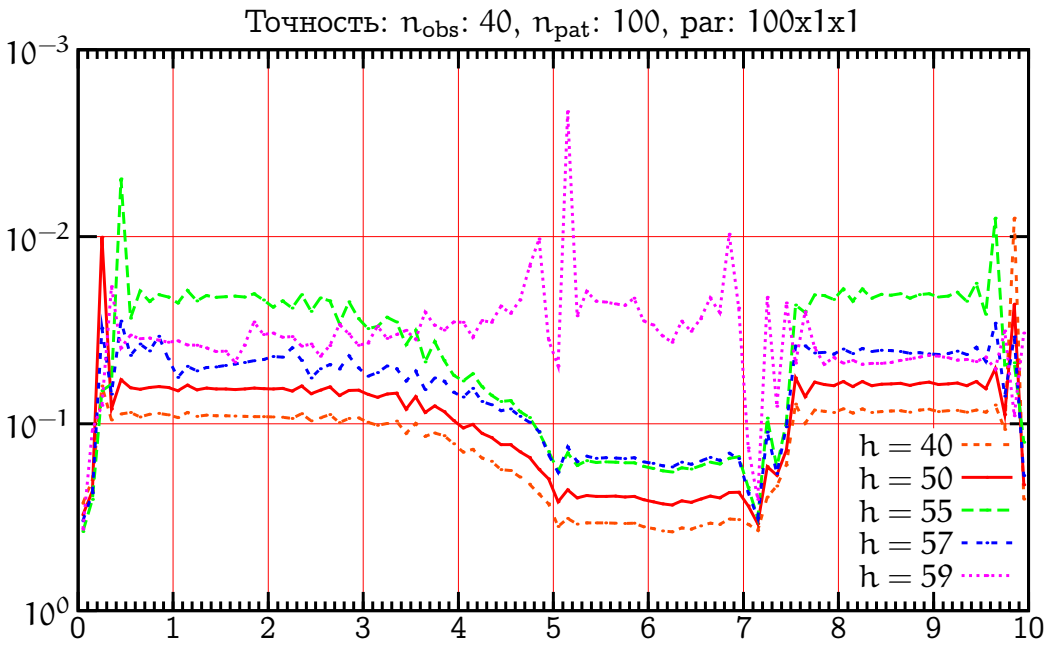
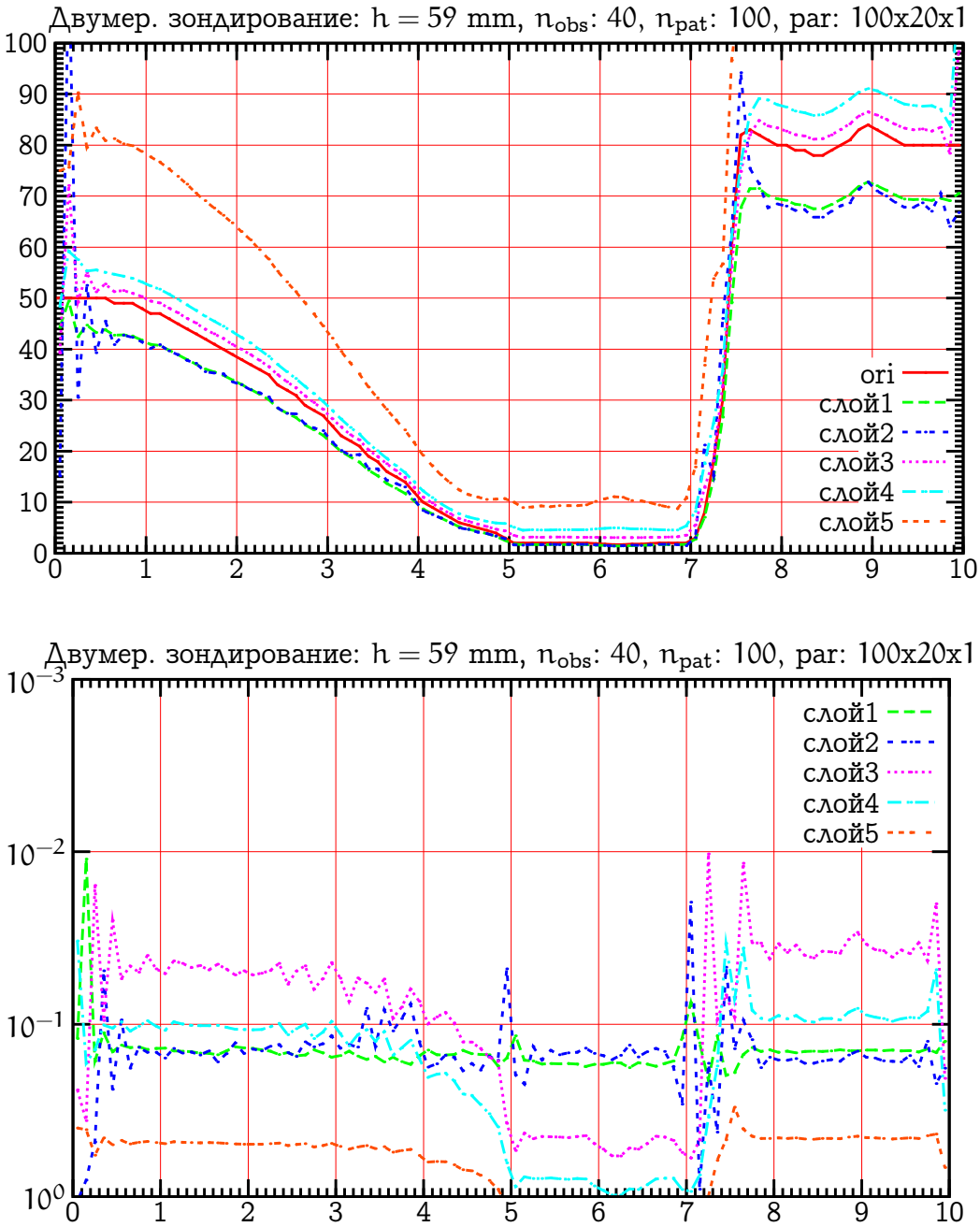


Рис. 4.16. Численные результаты и точность двумерного зондирования



лей. Однако опыт решения модельной задачи показывает, что решение этих задач вполне возможно и является лишь вопросом времени.

4.8. Выводы

В этой главе описана одна из особенно интересных для нас практических задач, сводимых к решению интегрального уравнения — задача о распространении электромагнитной волны в неоднородной трехмерной среде с затуханием. В силу специального вида ядра возникающего интегрального уравнения на равномерной сетке матрица задачи является суммой трехуровневой теплицевой и трехуровневой теплиц-ганкель-теплицевой матрицы. Использование этих структур позволило нам сократить затраты памяти на хранение этой плотной матрицы с n^6 до $\mathcal{O}(n^3)$ элементов, но тем не менее, оперативная память персональной станции позволяет использовать только сетки размером до $64 \times 64 \times 64$. Для практических вычислений этого было недостаточно, так как в принципиально важном случае — приближении источника к зоне неоднородности — возникающие дискретизационные шумы не позволяли достичь требуемого разрешения. Размер сетки был увеличен за счет использования многопроцессорных систем. Предложен алгоритм распределенного хранения и умножения на теплицеву мат-

Таблица 4.6. Сжатие блоков A^{kl} матрицы (4.16), точность $\varepsilon = 10^{-3}$.

n		16	32	64	128
$\mathtt{mem}(A)$		1.1GB	72GB	4.6PB	288PB
$\mathtt{mem}(A)$, равн. сетка		5MB	37MB	288MB	2.2GB
$\Re A^{11}$	mem	60Kb	280Kb	1.2Mb	6.2Mb
	(r_1, r_2, r_3)	$(11, 11, 8)$	$(13, 11, 11)$	$(15, 13, 13)$	$(18, 16, 14)$
$\Im A^{11}$	mem	60Kb	250Kb	1.0Mb	4.4Mb
	(r_1, r_2, r_3)	$(10, 10, 8)$	$(11, 11, 9)$	$(12, 10, 9)$	$(13, 11, 10)$
$\Re A^{12}$	mem	72Kb	350Kb	1.5Mb	7.1Mb
	(r_1, r_2, r_3)	$(12, 14, 10)$	$(16, 16, 13)$	$(18, 17, 15)$	$(20, 19, 16)$
$\Im A^{12}$	mem	74Kb	312Kb	1.2Mb	6.0Mb
	(r_1, r_2, r_3)	$(13, 13, 11)$	$(14, 13, 12)$	$(15, 14, 13)$	$(16, 15, 15)$
$\Re A^{13}$	mem	66Kb	288Kb	1.2Mb	6.0Mb
	(r_1, r_2, r_3)	$(11, 11, 11)$	$(13, 10, 13)$	$(14, 11, 15)$	$(16, 13, 17)$
$\Im A^{13}$	mem	64Kb	280Kb	1.0Mb	4.4Mb
	(r_1, r_2, r_3)	$(11, 10, 11)$	$(12, 11, 11)$	$(12, 10, 12)$	$(12, 10, 12)$

рицу, учитывающий ее структуру и особенность кластерных систем. Его применение позволило производить расчет на сетках вплоть до $256 \times 256 \times 256$, используя 256 процессоров. Результаты этого расчета были использованы для решения обратной задачи, то есть зондирования неоднородности, и привели к хорошему качеству восстановления искомых параметров.

Используя метод трехмерной крестовой аппроксимации, мы можем эффективно решить ту же задачу на одном процессоре, причем в том числе и на неравномерной сетке. Эксперименты демонстрируют сверхлинейное сжатие матрицы обсуждаемой задачи, что служит основанием для развития еще более эффективных методов ее решения. Как показано на модельном примере в разделе 3.3, использование трилинейной аппроксимации может привести к алгоритмам решения сублинейной сложности. В этом случае можно будет констатировать, что для этой задачи методы трилинейной аппроксимации даже на неравномерной сетке оказываются быстрее методов, основанных на использовании равномерных сеток и теплицевых структур в матрице. Кроме того, при использовании равномерных сеток теплицевая структура факторов в тензорном разложении матрицы будет сохраняться, то есть метод тензорной аппроксимации можно скомбинировать с использованием теплицевых структур. Это приведет к еще большему ускорению алгоритма решения.

Заключение

В заключение диссертации сформулируем ее основные результаты.

1. Доказана теорема существования трехмерной крестовой аппроксимации, которая строится по небольшому числу столбцов, строк и распорок исходного массива.

2. Предложен алгоритм трехмерной неполной крестовой аппроксимации. В качестве входного параметра алгоритм использует процедуру вычисления любого элемента массива $[a_{ijk}]$, однако этот массив не вычисляется и не хранится полностью. Для построения аппроксимации $[a_{ijk}]$ в виде разложения Таккера с модовыми рангами (r_1, r_2, r_3) достаточно вычислить порядка $\mathcal{O}(nr^a)$ $(r = \max(r_1, r_2, r_3),\ 1 \leqslant a \leqslant 2)$ элементов массива и совершить порядка $\mathcal{O}(nr^3)$ дополнительных действий.

3. Проведено тестирование алгоритма трехмерной неполной крестовой аппроксимации на модельном объемном интегральном уравнении и показано, что на основе этого метода можно построить алгоритмы сублинейной по числу неизвестных сложности.

Кроме указанных результатов в диссертации предложен метод дожимания для блоков, возникающих в ходе работы мозаично-скелетонного метода, предложена параллельная версия мозаично-скелетонного алгоритма и проведено ее тестирование на практических примерах, продемонстрировано применение мозаично-скелетонного метода к решению задачи Дирихле для двумерного уравнения Гельмгольца и к задаче гидроакустики. Рассмотрена задача распространения электромагнитной волны в трехмерной среде с затуханием, сформулированная в виде интегрального уравнения. Предложен параллельный алгоритм решения этой задачи, использующий специфику возникающих структурированных матриц. Алгоритм применен для решения прямой и обратной задач электродинамики. Показано, что применение трехмерного крестового метода может приводить к значительному сжатию матрицы системы и в перспективе позволит решать ту же задачу на неравномерных сетках без использования многопроцессорных систем.

Литература

[1] Anderson C. R. An implementation of the fast multipole method without multipoles. // *SIAM J. Sci. Stat. Comp.* 1992. V. 13. P. 923-947.

[2] Bebendorf M. Approximation of boundary element matrices // *Numer. Math.* 2000. V. 86, No. 4. P. 565-589.

[3] Beylkin G., Coifman R., Rokhlin V. Fast Wavelet Transforms and Numerical Algorithm I // *Yale University Preprint.* 1990.

[4] Brandt A., Lubrecht A. Multilevel matrix multiplication and fast solution of integral equations. // *J. Comp. Phys.* 1990. V. 90. P. 348-370.

[5] Caroll J. D., Chang J. J. Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart-Young decomposition // *Psychometrica.* 1970. V.35 p. 283-319.

[6] Chan R. H., Tyrtyshnikov E. E. Spectral Equivalence and Proper Clusters for Boundary Element Method Matrices // *Internat. J. Numer. Methods Engrg.* 2000. V. 49. No. 9. P. 1211–1224.

[7] Comon P. Tensor decomposition: State in the Art and Applications // *In IMA Conf. mathematics in Signal Processing*, Warwick, UK, Dec. 18-20,2000.

[8] Dennis J. E., Schabel R. B. Numerical methods for unconstrained optimization and nonlinear equations — Plentice-Hall: Englewood Clis, 1983.

[9] Ford J. M., Tyrtyshnikov E. E. Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related systems // *SIAM J. Sci. Comp.* 2003. V. 25, No. 3. P. 961-981

[10] Gao G., Fang S., Torres-Verdin C. A new approximation for 3D electromagnetic scattering in the presence of anisotropic conductive media // 3DEMIII Workshop, 2003, Adelaide.

[11] Goreinov S. A., Tyrtyshnikov E. E., Yeremin A. Y. Matrix-free iteration solution strategies for large dense linear systems // *Numer. Linear Algebra Appl.* 1996. V. 4 (5). P. 1-22.

[12] Goreinov S. A., Tyrtyshnikov E. E., Zamarashkin N. L. A theory of Pseudo–Skeleton Approximations // *Linear Alebra Appl.* 1997. V. 261. P. 1-21.

[13] Goreinov S. A., Tyrtyshnikov E. E. The maximal-volume concept in approximation by low-rank matrices // *Contemporary Mathematics.* 2001. V. 208. P. 47-51.

[14] Greengard L., Rokhlin V. The rapid evaluation of potential fields in three dimentions // *Lecture Notes in Mathematics.* 1988. V. 1360. P. 121-141.

[15] Hackbusch W., Nowak Z. P. On the fast matrix multiplication in the boundary elements method by panel clustering // *Numer. math.* 1989. V. 54 (4). P. 463-491.

[16] Hackbusch W. A sparce matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices // *Computing.* 1999. V. 62. P. 89-108.

[17] Hackbush W., Khoromskij B. N., Tyrtyshnikov E. E. Hierarchical Kronecker tensor-product approximations. // *J. Numer. Math.* 2005. V. 13. P. 119-156.

[18] Harshman R. A. Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis // *UCLA Working Papers in Phonetics.* 1970. V. 16. P. 1-84.

[19] Ibraghimov I. Application of the three-way decomposition for matrix compression // *Numer. Linear Algebra Appl.* 2002. V. 9, No. 6-7. P. 551-565.

[20] Ivakhnenko V. I., Kukk A. V., Tyrtyshnikov E. E. Application of 3D volume integral equations to solution of electromagnetic wave scattering problems. Report EM-RR 22/95. *Elegant Mathematics,* 1995.

[21] Ivahnenko V.I., Tyrtyshnikov E. E. Block-Toeplitz-Structure-based solution strategies for CEM problems. // Proceedings of 11th Annual Review of Progress in Applied Comp. Electromagnetics — Monterey, CA, March 20-25, 1995. P. 181-188.

[22] Kress R. Linear Integral Equations. // Appl. Math. Sci., 1982.

[23] Kruskal J. B. Three-way arrays: Rank and uniqueness for 3-way and n-way arrays // *Linear Algebra Appl.* 1977. V. 18. P. 95-138.

[24] De Lathauwer L. Signal processing based on multilinear algebra. Ph.D. Thesis. — Katholieke Univ.: Leuven, 1997.

[25] De Lathauwer L., de Moor B., Vandewalle J. A multilinear singular value decomposition *// SIAM J. Matrix Anal. Appl.* 2000. V.21. P. 1324-1342.

[26] De Lathauwer L., de Moor B., Vandewalle J. On best rank-1 and rank-(R1,R2,...,RN) approximation of high-order tensors *// SIAM J. Matrix Anal. Appl.* 2000. V. 21. P. 1324-1342.

[27] Myagchilov M. V., Tyrtyshnikov E. E. A fast matrix-vector multiplier in discrete vortex method *// Russian Journal of Numer. Anal. and Math. Modelling.* 1992. V. 7 (4). P. 325-342.

[28] Olshevsky V., Oseledets I. V., Tyrtyshnikov E.E. Tensor properties of multilevel Toeplitz and related matrices. *// Lin. Algebra Appl.* 2006. V. 1. P. 1-21.

[29] Petersdorff T., Schwab C., Schneider R. Multiwavelets for second kind integral equations. *// Preprint of University of Maryland.* 1991.

[30] Rokhlin V. Rapid solution of integral equations of classical potential theory *// J. Comput. Physics.* 1985. V. 60. P. 187-207.

[31] Rokhlin V. A fast algorithm for particle simulations. *// J. Comput. Physics.* 1987. V. 73. P. 325-348.

[32] Rokhlin V. Rapid solution of integral equations of scattering theory in two dimensions *// J. Comput. Physics.* 1990. V. 86. P. 414-439.

[33] Saad Y., Schultz M. H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM F. Scientific and Stat. Comp.* 7:856-869, 1986.

[34] Smirnov Yu. G., Tsupack A. A. Volume singular integral equations for solving diffraction problem of electromagnetic waves in mocrowave oven. *// Proc. of European Symp. on Numer. Meth. in Electromagnetics* March 6-8, 2002, Tolouse, France. P. 172-176.

[35] Sun X., Pitsianis N. P. A matrix version of the fast multipole methos *// SIAM Review.* 2001. V. 43, No. 2. P. 289-300.

[36] Tucker L. R. Some mathematical notes on three-mode factor analysis. *// Psychometrika.* 1966. V. 31. P. 279-311.

[37] Tyrtyshnikov E. E. Matrix approximations and cost-effective matrix-vector multiplication — М.: ИВМ РАН, 1993.

[38] Tyrtyshnikov E. E. Mosaic–skeleton approximations // *Calcolo.* 1996. V. 33 (1-2). P. 47-57.

[39] Tyrtyshnikov E. E. Incomplete cross approximation in the mosaic–skeleton method // *Computing.* 2000. V. 4. P. 367-380.

[40] Tyrtyshnikov E. E. Kronecker-product approximations for some function-related matrices // *Linear Algebra Appl.* 2004. V. 379. P. 423-437.

[41] Tyrstyshnikov E. E. Fast computation of Toeplitz forms and some multidimentional integrals. *Rus. J. Numer. Anal.* 2005. V. 20, No. 4. P. 383-380

[42] Tyrtyshnikov E. E. Optimal and Super-optimal Circulant Preconditioners // *SIAM J. Matrix Anal. Appl.,* 1992. V. 13. №2, p. 459-473.

[43] Watson G. N. A Threatise on the Theory of Bessel Functions. — Cambridge University Press, 1996.

[44] Zhang T., Golub G.H. Rank-one approximation to high-order tensors // *SIAM J. Matrix Anal. Appl.* 2001. V. 23. P. 534-550.

[45] Zwamborn A. P. M., Van der Berg. The three-dimensional weak form of the conjugate gradient FFT method for solving scattering problems. *IEEE Trans. Microwave Theory Tech.,* 1992, MTT-40, 9:1757-1765.

[46] Ахмед Н., Рао К. Ортогональные преобразования при обработке цифровых сигналов. М.: Связь, 1980.

[47] Бахвалов Н. С. Об оптимальных методах решения задач // *Appl. Mat.* 1969. V. 13. P. 27-43.

[48] Беккенбах Э., Беллман Р. Неравенства. М.: Мир, 1965.

[49] Воеводин В. В., Тыртышников Е. Е., Вычислительные процессы с теплицевыми матрицами. — М.: Наука, 1987.

[50] Голуб Дж., Ван Лоун Ч. Матричные вычисления / Пер. с. англ. — М.: Мир, 1999.

[51] Горейнов С. А., Замарашкин Н. Л., Тыртышников Е. Е. Псевдоскелетные аппроксимации матриц // *Доклады Российской академии наук.* 1995. V. 343 (2). P. 151-152.

[52] Горейнов С. А. Мозаично-скелетонные аппроксимации матриц, порожденных асимптотически гладкими и осцилляционными ядрами // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 42-76.

[53] Горейнов С. А. Псевдоскелетные аппроксимации для блочных матриц, порожденных асимптотически гладкими ядрами. Диссертация на соискание ученой степени кандидата физико-математических наук. — М.: ИВМ РАН, 2001.

[54] Довгий С.А., Лифанов И.К. Методы решения интегральных уравнений. – Киев: Наукова думка, 2002. – 344 с.

[55] Еремин Ю. А., Ивахненко В. И. Строгие и приближенные модели царапины на основе метода интегральных уравнений // *Дифф. уравнения.* Т. 37, №10. 2001. С. 1386-1394.

[56] Ибрагимов И. В. Новый подход к решению задачи обобщённого сингулярного разложения // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 193–201.

[57] Колтон Д., Кресс Р. Методы интегральных уравнений в теории рассеивания. – М.: Мир, 1987. – 311с.

[58] Лифанов И.К. Метод сингулярных интегральных уравнений и численный эксперимент. М.: ТОО Янус, 1995 – 520с.

[59] Мартынов М. С. Результаты применения мозаично–скелетонного метода для решения интегральных уравнений на параллельном компьютере МВС-100 // *Численный анализ и математическое моделирование* — М.: ИВМ РАН, 1999. С. 109-115.

[60] Нечепуренко Ю. М. Быстрые численно устойчивые алгоритмы для широкого класса линейных дискретных преобразований. — М.: 1985. Препринт ОВМ АН СССР №98.

[61] Никифоров А. Ф., Уваров В. Б. Специальные функции математической физики — М.: Наука, 1984.

[62] Оселедец И. В., Тыртышников Е. Е. Приближенное обращение матриц при решении гиперсингулярного интегрального уравнения // *ЖВМиМФ,* 2005. Т. 45, №2. С. 315-326.

[63] Парлетт Б. Симметричная проблема собственных значений. Численные методы / Пер. с англ. — М.: Мир, 1983.

[64] Самохин А. Б. Интегральные уравнения и итерационные методы в электромагнитном рассеянии — М.: Радио и связь, 1998.

[65] Самохин А. Б. Исследование задач дифракции электромагнитных волн в локально-неоднородных средах // *ЖВМиМФ.* 1990. Т. 30. С. 107-121.

[66] Тыртышников Е. Е. Методы быстрого умножения и решение уравнений // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 4-41.

[67] Тыртышников Е. Е. Тензорные аппроксимации матриц, порожденных асимптотически гладкими функциями // *Матем. сб.* 2003. Т. 194, 6. С. 147-160.

Публикации по теме диссертации:

[68] Oseledets I. V., Savostianov D. V., Tyrtyshnikov E. E. Tucker dimensionality reduction of three-dimensional arrays in linear time // *SIAM J. Matr. Anal. Appl.,* принято к публикации.

[69] Савостьянов Д. В. Мозаично–скелетонные аппроксимации. Выпускная квалификационная работа на степень бакалавра — М.: ИВМ РАН, 2001.

[70] Тыртышников Е. Е., Горейнов С. А., Чугунов В. Н., Святский Д. А., Савостьянов Д. В. Математическое обеспечение для решения задач на многопроцессорных вычислительных кластерах. Отчет по ФЦП «Интеграция», инв. номер 02.200.203461, 2001г.

[71] Тыртышников Е. Е., Горейнов С. А., Савостьянов Д. В. Математическое и программное обеспечение для задач электродинамики на вычислительном кластере. Задачи электродинамики для распределенных вычислительных систем. Отчет по ФЦП «Интеграция», инв. номер 02.20.0303023, 2002г.

[72] Савостьянов Д. В. Параллельная реализация метода решения объемного интегрального уравнения электродинамики на основе многоуровневых теплицевых матриц // *Методы и технологии решения больших задач* — ИВМ РАН. 2004. С. 139–170.

[73] Оселедец И. В., Савостьянов Д. В., Ставцев С. Л. Применение нелинейных методов аппроксимации для быстрого решения задачи распространения звука в мелком море // *Методы и технологии решения больших задач* — ИВМ РАН. 2004. С. 139–170.

[74] Савостьянов Д. В., Тыртышников Е. Е. Применение многоуровневых матриц специального вида для решения прямых и обратных задач электродинамики // *Вычислительные методы и программирование.* 2006. Т. 7. С. 1–16.

[75] Оселедец И. В., Савостьянов Д. В. Методы разложения тензора // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 51–64.

[76] Оселедец И. В., Савостьянов Д. В. Трехмерный аналог алгоритма крестовой аппроксимации и его эффективная реализация // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 65–100.

[77] Оселедец И. В., Савостьянов Д. В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 101–116.

[78] Оселедец И. В., Савостьянов Д. В. Об одном алгоритме построения трилинейного разложения // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 117–130.

[79] Оселедец И. В., Савостьянов Д. В. Минимизационные методы аппроксимации тензоров и их сравнение // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 131–146.

[80] Оселедец И. В., Савостьянов Д. В. Тензорные ранги сверхбольших трехмерных матриц // *Матричные методы и технологии решения больших задач* — М.: ИВМ РАН, 2005. С. 147–174.

[81] Оселедец И. В., Савостьянов Д. В. Минимизационные методы аппроксимации тензоров и их сравнение // *ЖВМиМФ.* 2006. Т. 46, №10. С. 1641-1650.