МАТРИЧНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ РЕШЕНИЯ БОЛЬШИХ ЗАДАЧ

МОСКВА 2005

Российская академия наук
Институт вычислительной математики

МАТРИЧНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ РЕШЕНИЯ БОЛЬШИХ ЗАДАЧ

Сборник научных трудов
под редакцией Е. Е. Тыртышникова

Москва 2005

УДК 519.6
ББК 22.20
М 65

Матричные методы и технологии решения больших задач: Сборник научных трудов / Под ред. Е. Е. Тыртышникова. — М.: Институт вычислительной математики РАН, 2005. — 176 с. ISBN 5-901854-09-8

В сборнике представлены результаты по развитию новых математических методов и технологий для решения задач со сверхбольшим числом неизвестных. Все работы выполнены в рамках программы приоритетных фундаментальных исследований Отделения математических наук Российской академии наук «Вычислительные и информационные технологии решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях». Для научных работников, аспирантов и студентов, специализирующихся в области численного анализа и разработки современного программного обеспечения.

The book presents a collection of results developing new mathematical methods and technologies for problems with super-large number of unknowns. All the papers are performed under the support of the priority basic research programme "Computational and informational technologies for large-scale problems" of the Mathematical Department of the Russian Academy of Sciences, within the project "Matrix methods in integral and differential equations". The book is addressed to researchers, post graduates and students majorizing in numerical analysis and modern software development.

Печатается по решению Ученого совета
Института вычислительной математики
Российской академии наук

ISBN 5-901854-09-8

ПРЕДИСЛОВИЕ

При решении трехмерных задач в инженерной и научно-исследовательской практике необходимо уметь работать с огромными массивами числовых данных. Например, в задачах расчета летательных аппаратов областью определения неизвестной функции является поверхность летательного аппарата, в задачах дифракции это может быть поверхность рассеивателя (например, того же летательного аппарата) или антенны, в задачах электромагнитного каротажа это может быть область неоднородности (нефтяной пласт и т. п.) или ее граница.

Основным инструментом моделирования во всех этих случаях являются системы интегральных уравнений, заданные на поверхностях или в областях сложной формы. При этом значение неизвестной функции в любой точке зависит от значений во всех других точках — это означает, что после дискретизации все или почти все коэффициенты для всех пар узлов из области сложной формы отличны от нуля и должны участвовать в расчете. Если общее число узлов есть величина порядка 10^6, то общее число ненулевых числовых коэффициентов будет порядка 10^{12}. Если для хранения одного числа используется 8 байт, то для всего массива коэффициентов потребуется более 7 терабайт. Это не так уж мало. Но более серьезной проблемой является то, что традиционные методы решения систем уравнений имеют время работы, пропорциональное кубу числа узлов. Поэтому если допустить, что для 10^3 узлов требуется время порядка 0.1 секунды, то для 10^6 потребуется уже около 1200 суток, то есть более 3 лет.

Для задач такого рода необходимы не только высокопроизводительные компьютеры, но и специальные математические методы, позволяющие получать сжатое представление огромного массива числовых данных с помощью относительно малого числа параметров. Существенным является также то, что это должно быть представление с определенной структурой данных, допускающей эффективные методы решения соответствующих приближенных систем уравнений. Такого типа подходы могут базироваться на современных методах нелинейной аппроксимации.

Схожие проблемы возникают и в тех случаях, когда связи носят локальный характер, например при дискретизации дифференциальных уравнений. В таких задачах первостепенное значение имеет развитие технологий построения адаптивных и анизотропных сеток с минимально возможным числом степеней свободы.

По данным направлениям исследования в Институте вычислительной математики РАН велись с момента основания института. В последние три года они были существенным образом поддержаны проектом «Матричные методы в интегральных и дифференциальных уравнениях» в рамках программы приоритетных исследований ОМН РАН «Вычислительные и информационные технологии решения больших задач». К настоящему времени можно уже говорить об очень успешном продвижении в развитии всего данного направления — можно утверждать, что в ИВМ РАН созданы не только основы уникальных эффективных технологий, но и сделаны серьезные шаги по их внедрению в решение практических больших задач. Направление развивается настолько активно, что работы, публикуемые в данном сборнике, являются не только итогом проведенных исследований, но в большей степени базой для их продолжения и превращения в хорошо отлаженные технологии и комплексы программ для вычислительных систем разных типов, в том числе и для вычислительных кластеров.

Особенно следует отметить успехи в области построения тензорных аппроксимаций на основе неполной информации об исходных данных. Получен быстрый метод одновременного приближённого приведения семейства матриц к треугольному виду. На основе этого метода разработан алгоритм построения трилинейной аппроксимации трёхмерных массивов, позволяющий находить разложения массива с размерами $128\times128\times128$ и тензорным рангом 128 за несколько минут. Разработаны эффективные методы трилинейной аппроксимации для трёхмерных массивов (тензоров) с рангом, превышающим размеры тензора. Построен эффективный алгоритм неполной крестовой аппроксимации для многомерных массивов, позволяющий находить разложения Таккера для кубического массива размера n в кубе с почти линейной асимптотикой сложности по n. Созданный на основе разработанных методов программный комплекс позволяет строить трилинейное разложение в случае $n = 65536$ за время порядка часа на обычной персональной ЭВМ и тем самым снижать затраты на его хранение с двух петабайт (2^{50} байт) до нескольких десятков мегабайт.

Эти результаты открывают принципиально новые возможности в развитии вычислительных технологий. Конечно, это еще потребует значительных усилий. Но можно утверждать, что все основы для очень заметного успеха в данном направлении, безусловно, созданы.

Е. Е. Тыртышников

О применении метода выделения уравнения для давления при решении задачи многофазовой фильтрации[a]

Ю. В. Василевский, В. Н. Чугунов

Исследуются особенности применения метода выделения уравнения для давления при решении задачи многофазной фильтрации. Проводится анализ численных экспериментов.

1. Введение

При моделировании нелинейных процессов фильтрации в пористых средах в последнее время большое распространение получают полностью неявные схемы, которые являются абсолютно устойчивыми, хотя и вычислительно дорогими. Каждый временной шаг этих схем требует решения системы нелинейных уравнений, которое осуществляется неточным методом Ньютона. При использовании этого метода необходимо решать системы линейных уравнений с большой разреженной несимметричной и плохо обусловленной матрицей. Поэтому возникает потребность в конструировании качественных предобусловливателей для этих матриц.

Метод выделения уравнения для давления строит эффективный предобусловливатель, основанный на физических, а не на геометрических свойствах матрицы. Известно, что для модели нелетучей нефти [1, 3, 7] уравнение для давления параболично или эллиптично, а уравнения для концентраций гиперболичны или параболичны с доминирующим транспортом. Такие же свойства наследуются композиционными моделями [9]. В связи с этим естественно по-

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

разному предобусловливать различные блоки матрицы, которые, однако, связаны между собой. Метод выделения уравнения для давления [8, 5, 6] основан на преобразовании матрицы с целью ее расщепления на независимые блоки и их дальнейшем предобусловливании. Результаты сравнения данного подхода с методами неполного LU-разложения свидетельствуют о существенных преимуществах метода выделения уравнения для давления [2].

В данной работе исследуются особенности применения метода выделения уравнения для давления при решении задачи многофазной фильтрации. Поставленные нами численные эксперименты свидетельствуют о большом выигрыше в скорости сходимости итерационного процесса и в объеме памяти в результате использования метода при решении системы, в которой отсутствуют неизвестные, связанные со скважинами. Однако при наличии неизвестных, связанных со скважинами, исследуемый метод не дает преимуществ, открывая перспективы дальнейших исследований.

Структура статьи следующая. В разделе 2 описано в общих чертах, как возникает система линейных алгебраических уравнений при решении задачи многофазной фильтрации. В разделе 3 обсуждаются свойства системы и вытекающий из этих свойств формат представления матрицы. Описанию метода выделения уравнения для давления посвящен 4-й раздел. В разделе 5 приведены результаты численных экспериментов и их анализ.

2. Происхождение системы линейных уравнений

Мы применяем метод выделения уравнения для давления с целью построения предобусловливателя матрицы системы линейных уравнений, возникающей при решении задачи многофазной фильтрации.

Рассмотрим среду с пористостью ϕ и тензором проницаемости K. Пусть D и G – глубина и ускорение свободного падения соответственно. В многофазном течении каждой фазе m соответствует давление P_m, плотность ρ_m, насыщенность S_m $(\sum_m S_m = 1)$, относительная проницаемость k_m и вязкость μ_m. Поскольку растворенные компоненты, обозначаемые индексом M, могут находиться в нескольких фазах, определим массовую долю компоненты M в фазе m через n_{mM} $(\sum_M n_{mM} = 1)$, тогда общая концентрация компоненты равна $N_M = \sum_m \rho_m S_m n_{mM}$. Закон сохранения массы

для каждой компоненты задается выражением

$$\frac{\partial(\phi N_M)}{\partial t} + \nabla \cdot U_M = q_M + \sum_m \phi S_m R_{mM}. \tag{1}$$

Источник в правой части q_M представляет скважины, члены R_{mM} обозначают химические реакции. Определим поток U_M как общий массовый поток компоненты M

$$U_M = \sum_m \rho_m \left(n_{mM} V_m - \phi S_m D_{mM} \nabla n_{mM} \right), \tag{2}$$

где V_m — скорость фазы m, а D_{mM} представляет тензор диффузии/дисперсии. Значения V_m подчинены закону Дарси:

$$V_m = -K \frac{k_m}{\mu_m} (\nabla P_m - \rho_m G \nabla D). \tag{3}$$

Система уравнений замыкается добавлением связи для капиллярных давлений $P_{cm_1m_2} = P_{m_1} - P_{m_2}$ на границе фаз m_1 и m_2 и относительных проницаемостей k_m, являющихся функциями от насыщенностей S_m всех фаз, характерными для каждой породы. Кроме этого, добавляются уравнения состояния, формулирующие зависимость плотности ρ_m и вязкости от давления P_m и массовой доли n_{mM}.

Например, для простой модели двух несмешиваемых фаз (нефть-вода, воздух-вода, нефть-газ) члены реакций и дисперсионные члены равны нулю, а фаза может быть приравнена к одной из компонент: $w \equiv W$, $n \equiv N$. Таким образом, $n_{nN} = 1$, $n_{wW} = 1$, $n_{nW} = 0$, $n_{wN} = 0$ и закон сохранения массы принимает вид

$$\frac{\partial(\phi N_W)}{\partial t} + \nabla \cdot (\rho_w V_w) = q_W, \tag{4}$$

$$\frac{\partial(\phi N_N)}{\partial t} + \nabla \cdot (\rho_n V_n) = q_N. \tag{5}$$

Классическая трехфазная модель нелетучей нефти учитывает три различные фазы: воду, нефть (нелетучая фракция углеводорода) и газ (летучая фракция углеводорода). Соответственно, можно задать систему трех уравнений, аналогичную (4)-(5).

Применим теперь процедуру линеаризации для упрощенной модели, где игнорируются капиллярное давление и сила тяжести. В многофазной модели фильтрации каждой ячейке сетки соответствуют $n+m$ уравнений. Первые n уравнений — закон сохранения для

n компонент M_i:

$$\Delta_t M_i = Q_i \Delta t, \quad i = 1, \ldots, n. \tag{6}$$

Здесь Q_i представляет течение между ячейками и скважинами:

$$Q_i = \sum_{\kappa} T_{i\kappa}(p_\kappa - p) - q_i, \tag{7}$$

$\sum\limits_{\kappa}$ обозначает суммирование по всем соседствующим ячейкам сетки κ; p и p_κ — давление в сеточной ячейке и ее соседях; q_i — производительность скважины по компоненте i; $T_{i\kappa}$ — проводимость для компоненты i между ячейкой и ее соседом κ. Уравнения (6)-(7) получаются из консервативной аппроксимации (например, метод смешанных конечных элементов).

При использовании полностью неявной схемы изменения $\Delta_t M_i = M_i^{k+1} - M_i^k$ и $Q_i = Q_i^{k+1}$ неизвестны на $k+1$ шаге по времени и вычисляются методом Ньютона. Пусть $M_i^{k,l+1}$, $Q_i^{k,l+1}$ — очередные итерационные приближения к M_i^{k+1}, Q_i^{k+1}. Уравнение (6) может быть переписано как

$$M_i^{k,l+1} - M_i^{k,l} + M_i^{k,l} - M_i^k - Q_i^{k,l}\Delta t = (Q_i^{k,l+1} - Q_i^{k,l})\Delta t. \tag{8}$$

Поскольку $M_i^{k+1} - M_i^k = Q_i^{k+1}\Delta t$, ньютоновская невязка равна

$$r_i = M_i^{k,l} - M_i^k - Q_i^{k,l}\Delta t,$$

и (8) может быть переписано в форме приращений:

$$\delta M_i + r_i = \delta Q_i \Delta t, \quad i = 1, \ldots, n. \tag{9}$$

Отметим, что для n компонент всегда существует множество $n+m$ переменных $\{X_j\}$, $j = 1, \ldots, n+m$, таких, что каждое M_i — однозначная функция $\{X_j\}$. Первые n переменных из $\{X_j\}$ являются первичными, остальные — вторичными. Для замыкания системы (9) добавим дополнительные m ограничений, выражающих равновесие фаз, единичную сумму насыщенностей, и другие ограничения модели. Общая форма этих ограничений имеет вид

$$\delta L_i + r_i = 0, \quad i = n+1, \ldots, n+m. \tag{10}$$

Линеаризация (9)-(10) порождает линейные уравнения

$$\sum_{j=1}^{n+m} g_{ij}^M \delta X_j + r_i = \sum_{j=1}^{n+m} g_{ij}^Q \delta X_j, \quad i = 1, \ldots, n, \tag{11}$$

$$\sum_{j=1}^{n+m} g_{ij}^{L}\delta X_j + r_i = 0, \quad i = n+1, \ldots, n+m, \tag{12}$$

где g_{ij}^{M}, g_{ij}^{Q}, и g_{ij}^{L} порождены аккумулирующими, потоковыми и источниковыми членами якобиана соответственно.

В силу локальности ограничений модели вторичные переменные могут быть исключены локально по ячейкам, поэтому система (11)-(12) сводится к системе

$$Ax = f, \tag{13}$$

где матрица A — дополнение по Шуру матрицы системы (11)-(12), в которой каждая строка соответствует закону сохранения какой-то компоненты в каждой ячейке сетки.

3. Формат и свойства матрицы системы

В данном параграфе опишем формат матрицы системы, который наилучшим образом отражает ее свойства. Система дифференциальных уравнений модели многофазной фильтрации описывает одновременно несколько физических процессов с разным количеством вовлеченных компонент. Двухуровневая блочная структура матрицы представляется удобным инструментом для отражения этого факта. На первом уровне все неизвестные системы разделены на ассоциированные с ячейками сетки и связанные со скважинами. Обозначим их через x_r и x_w соответственно.

Неизвестные, определяемые элементами сетки, группируются в зависимости от того, какой подобласти принадлежит ячейка и какого она типа. Не ограничивая общности, будем считать, что ячейки могут быть двух типов: $\mathcal{C}$-ячейки, в которых заданы концентрации и давление (вычисляемые на верхнем временном слое), и $\mathcal{I}$-ячейки, в которых задано только давление (на верхнем временном слое).

С учетом блочной структуры первого уровня система имеет вид

$$\begin{pmatrix} A_{rr} & A_{rw} \\ A_{wr} & A_{ww} \end{pmatrix} \begin{pmatrix} x_r \\ x_w \end{pmatrix} = \begin{pmatrix} f_r \\ f_w \end{pmatrix}, \tag{14}$$

где

$$A_{rr} = \begin{pmatrix} A_C & A_{CI} \\ A_{IC} & A_I \end{pmatrix}. \tag{15}$$

Второй уровень блочного разбиения связан с упорядочением неизвестных, связанных с $\mathcal{C}$-ячейками. Будем считать, что сначала идут все неизвестные, ассоциированные с первой ячейкой, потом

со второй и т.д. При этом для каждого такого узла сначала идут $n-1$ неизвестных, определяющих концентрации, а потом давление в данной ячейке. В результате подблок A_C может быть представлен в виде

$$A_C = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{pmatrix}, \tag{16}$$

где m — число $\mathcal{C}$-ячеек, а каждый $n \times n$-блок A_{ij} имеет вид

$$A_{ij} = \begin{pmatrix} A_s^{ij} & A_{sp}^{ij} \\ A_{ps}^{ij} & A_p^{ij} \end{pmatrix} \tag{17}$$

с квадратной матрицей A_s^{ij} размера $n-1$. Если обозначить через n_I число $\mathcal{I}$-ячеек, а через n_w количество ассоциированных со скважинами неизвестных, то из (14)-(17) следует, что A_C — $mn \times mn$-матрица, A_{rr} — $(mn+n_I)\times(mn+n_I)$-матрица. Блочная структура матрицы будет использована при построении предобусловливателя, который мы опишем в следующем параграфе.

Укажем некоторые особенности рассматриваемой матрицы. Она является большой разреженной несимметричной и плохо обусловленной. Наиболее приемлемым способом решения системы является итерационный метод с качественным предобусловливателем.

Физические свойства якобиана указывают на то, что, как правило, *взаимодействие между резервуарными переменными главным образом локально.* В алгебраических терминах это означает, что элементы внедиагональных блоков в A_C, отвечающие за взаимодействия разных переменных в разных ячейках, малы по отношению к аналогичным элементам внутри диагонального блока (взаимодействие внутри ячейки). Однако связи между переменными, ассоциированными с сеткой и скважинами, достаточно сильные, поэтому элементами блоков A_{rw} и A_{wr} нельзя пренебрегать при построении предобусловливателя.

4. Описание предобусловливателя

В этом разделе опишем сам процесс построения предобусловливателя для матрицы A, основанный на физических, а не на геометрических свойствах матрицы. Поскольку матричные блоки соответствуют разным физическим процессам, представляется разумным предобусловливать разные блоки по-разному, используя их

свойства. Такой подход имеет два потенциальных преимущества: во-первых, можно строить хороший предобусловливатель только для самых жестких блоков, предобусловливая другие блоки вычислительно дешевыми технологиями. Во-вторых, этот подход не зависит от структуры сетки, поскольку предобусловливатели для блоков могут опираться только на их алгебраическую структуру. Так как диагональные блоки матрицы A связаны ненулевыми внедиагональными блоками, нашей целью является попытка разъединить или расщепить их.

Мы исследуем две версии предобусловливателя, различия в применении которых обусловлены наличием или отсутствием в системе неизвестных, связанных со скважинами. В каждом случае операция предобусловливания включает в себя стадию инициализации и правило действия на вектор в итерационном процессе.

4.1. Предобусловливатель для системы без неизвестных, соответствующих скважинам. Изложение предобусловливателей начнем с более простого случая, когда в системе есть только неизвестные, ассоциированные с ячейками сетки. Матрица системы имеет вид $A = A_{rr}$, где A_{rr} описывается формулой (15).

Целью стадии инициализации является стремление отщепить блок матрицы, отвечающий давлению, и попытаться хорошо его предобусловить. Несмотря на то что давление в каждой ячейке слабо зависит от концентрации в других узлах, эта связь внутри каждой ячейки достаточно сильная. Поэтому сначала удалим зависимость давления от концентрации внутри каждого диагонального блока матрицы. Это осуществляется благодаря модификации исходной системы (13) с помощью умножения слева на матрицу G

$$GAx = Gf, \tag{18}$$

где матрица G является блочно-диагональной

$$G = \begin{pmatrix} G^{11} & & & & \\ & G^{22} & & & \\ & & \ddots & & \\ & & & G^{mm} & \\ & & & & I_{n_I} \end{pmatrix}, \tag{19}$$

а I_{n_I} обозначает единичную матрицу порядка n_I. При этом каждый блок G^{kk} $(k = 1, ..., m)$ ассоциируется с k-й $\mathcal{C}$-ячейкой и строится с учетом следующих условий на матрицу $\tilde{A} = GA$:

1) $\tilde{A}^{kk}_{ps} = 0$;
2) подматрица $\tilde{A}^{kk}_{ss}$ является верхней треугольной;
3) при действии на вектор матрица G сохраняет его норму.

В качестве k-го блока G^{kk} будем брать матрицу вида

$$G^{kk} = \prod_{j=n-1}^{1} \prod_{i=n}^{j+1} G_k^{(ij)},$$

где $G_k^{(ij)}$ — матрица Гивенса, обнуляющая элементы $(A^{kk})_{ij}$. Получаем систему $\tilde{A}x = \tilde{f}$, которая есть следствие предобусловливания (13) слева матрицей G.

Далее из матрицы $\tilde{A}$ вырезаем $(m + n_I) \times (m + n_I)$-подматрицу $\tilde{A}_p$, отвечающую только давлению в каждом узле сетки:

$$\tilde{A}_p = C\tilde{A}C^t, \tag{20}$$

где

$$C = \begin{pmatrix} e_n & & & & \\ & e_n & & & \\ & & \ddots & & \\ & & & e_n & \\ & & & & I_{n_I} \end{pmatrix}, \quad e_n = (0, ..., 0, 1). \tag{21}$$

Используя алгоритм неполного LU-разложения с порогом τ [4], формируем предобусловливатель C_p для матрицы A_p. На этом стадия инициализации завершается.

Для простоты изложения правила действия предобусловливателя на вектор в итерационном процессе в системе $\tilde{A}x = \tilde{f}$ поменяем порядок неизвестных, поставив последними переменные, отвечающие давлению

$$\tilde{A} = \begin{pmatrix} \tilde{A}_s & \tilde{A}_{sp} \\ \tilde{A}_{ps} & \tilde{A}_p \end{pmatrix}.$$

Это позволяет описать процесс действия на вектор как применение блочного метода Гаусса–Зейделя для решения системы с матрицей

$$C = \begin{pmatrix} [\tilde{A}_s] & \tilde{A}_{sp} \\ O & \tilde{A}_p \end{pmatrix},$$

где $[\tilde{A}_s]$ получается из $\tilde{A}_s$ обнулением поддиагональных элементов. Вместо решения системы с A_p применяем предобусловливатель C_p. Нахождение действия $([\tilde{A}_s])^{-1}$ на вектор представляет собой решение системы линейных уравнений с верхней треугольной матрицей.

4.2. Предобусловливатель для системы с неизвестными, ассоциированными со скважинами. Теперь опишем процесс решения системы, в которой присутствуют неизвестные, отвечающие скважинам:

$$\begin{pmatrix} A_{rr} & A_{rw} \\ A_{wr} & A_{ww} \end{pmatrix} \begin{pmatrix} x_r \\ x_w \end{pmatrix} = \begin{pmatrix} f_r \\ f_w \end{pmatrix}. \tag{22}$$

Основная сложность заключается в том, что элементы внедиагональных блоков не являются малыми по отношению к диагональным, поэтому простое пренебрежение ими может привести к очень медленной сходимости итерационного процесса. Чтобы найти неизвестные в (22), исключим переменные x_w, используя дополнение по Шуру, и перейдем к системе

$$\hat{A}x_r = \hat{f}, \tag{23}$$

где

$$\hat{A} = A_{rr} - A_{rw}A_{ww}^{-1}A_{wr}, \quad \hat{f} = f_r - A_{rw}A_{ww}^{-1}f_w.$$

После нахождения x_r несложно вычислить

$$x_w = A_{ww}^{-1}(f_w - A_{wr}x_r).$$

Матрица A_{ww}, как правило, невырождена, блочно-диагональная, и блоки ее имеют небольшой порядок, поэтому ее легко обратить прямым методом. Отметим, что слагаемое $A_{rw}A_{ww}^{-1}A_{wr}$ вносит дополнительные ненулевые элементы в разреженную структуру матрицы A_{rr}. В результате разреженность $\hat{A}$ ухудшается. По этой причине невыгодно хранить $\hat{A}$, а разумно в итерационном методе выполнять операцию умножения матрицы $\hat{A}$ на вектор как совокупность последовательного умножения матриц $A_{wr}, A_{ww}^{-1}, A_{rw}$ и A_{rr} на вектор.

В силу того что элементы блоков A_{rw} и A_{wr} достаточно велики, физические свойства матрицы $\hat{A}$ очень сильно отличаются от свойств A_{rr}. Поэтому полностью идентичное применение предобусловливателя, описанного в предыдущем разделе, не дает приемлемых результатов. Одним из выходов является построение методом выделения уравнения для давления предобусловливателя для матрицы

$$\hat{B} = A_{rr} - A_{rw}L_{ww}^{-1}A_{wr}, \quad L_{ww} = \operatorname{diag}\{(A_{ww})_{ii}\}_{i=1}^{n_w}.$$

Матрица $\hat{B}$ является менее заполненной в силу диагональности L_{ww}.

На стадии инициализации мы модифицируем систему (23), умножая ее слева на матрицу G^I:

$$G^I \hat{A} x_r = G^I \hat{f},$$

где G^I — блочно-диагональная матрица с блоками

$$G^I = \text{block diag}\{G_1^I, ..., G_m^I, I_{n_I}\},$$

$$G_k^I = \begin{cases} I_n, & \text{если } (A_{rw})_k \neq 0, \\ \prod\limits_{j=n-1}^{1} \prod\limits_{i=n}^{j+1} G_k^{(ij)}, & \text{иначе,} \end{cases}$$

а $G_k^{(ij)}$ — матрицы Гивенса, обнуляющие элемент $(A^{kk})_{ij}$. Данное преобразование приводит к верхнему треугольному виду те диагональные блоки, которые одинаковы у $\hat{A}$ и $\hat{B}$. В результате приходим к системе

$$\tilde{A} x = \tilde{f}.$$

После этого, сформировав матрицу

$$B = \tilde{A}_{rr} - A_{rw} L_{ww}^{-1} A_{wr},$$

приводим к верхнему треугольному виду те из ее диагональных блоков, которые были проигнорированы преобразованием, осуществляемым матрицей G^I. Для этого строим $\tilde{B} = G^{II} B$ с матрицей

$$G^{II} = \text{block diag}\{G_1^{II}, ..., G_m^{II}, I_{n_I}\},$$

$$G_k^{II} = \begin{cases} I_n, & \text{если } (A_{rw})_k = 0, \\ \prod\limits_{j=n-1}^{1} \prod\limits_{i=n}^{j+1} G_k^{(ij)}, & \text{иначе.} \end{cases}$$

Несложно проверить, что

$$B^{-1} = (\tilde{B})^{-1} G^{II}. \tag{24}$$

Поэтому задача построения предобусловливателя для B эквивалентна поиску предобусловливателя для $\tilde{B}$, который можно построить по аналогии с алгоритмом из предыдущего раздела. Для этого меняем порядок неизвестных

$$\tilde{B} = \begin{pmatrix} \tilde{B}_s & \tilde{B}_{sp} \\ \tilde{B}_{ps} & \tilde{B}_p \end{pmatrix}$$

и формируем предобусловливатель C_p для матрицы $\tilde{B}_p$, используя неполное LU-разложение.

Исходя из (24), операцию действия предобусловливателя на вектор определим как умножение матрицы G^{II} на вектор и применение блочного метода Гаусса–Зейделя для решения системы с матрицей

$$C = \begin{pmatrix} [\tilde{B}_s] & \tilde{B}_{sp} \\ O & \tilde{B}_p \end{pmatrix},$$

где $[\tilde{B}_s]$ получается из $\tilde{B}_s$ обнулением поддиагональных элементов. Для обращения $\tilde{B}_p$ используем C_p.

Из описания метода видно, что нет необходимости хранить G^I, так как она используется только на стадии инициализации. G^{II} может быть сохранена в памяти как набор пар чисел, каждая из которых определяет локальную матрицу Гивенса.

5. Численные эксперименты

Описанный в данной работе метод был протестирован на системах, порождаемых двумя многокомпонентными моделями.

Первая модель была 3-компонентной, вторая содержала 21 компоненту. Порядки матриц систем равны 37771 и 15746 соответственно. В табл. 1 и 2 представлены результаты сравнения метода выделения уравнения для давления и варианта ILU-разложения с порогом τ [4] в случае отсутствия неизвестных, ассоциируемых со скважинами. В первых столбцах таблиц указаны номер шага по времени и номер ньютоновской итерации. Так как в методе выделения уравнения для давления размер матрицы для давления меньше размера всей матрицы, мы можем уменьшать параметр τ, увеличивая качество предобусловливателя C_p. Как видно из табл. 1 и 2, метод выделения уравнения для давления дает существенный выигрыш по сравнению с ILU-разложением не только в числе итераций и во времени, но и в используемой памяти.

Результаты численных экспериментов решения систем линейных уравнений с неизвестными, ассоциированными со скважинами, приведены в табл. 3 и 4. Тестирование также проводилось на матрицах 3- и 21-компонентных моделей, порядков 65420 и 15750 (причем в первом случай размер матрицы A_I существенно больше размера A_C). В таблицах приведено сравнение метода выделения уравнения для давления и ILU при двух различных критериях остановки итерационного процесса. Как показывают полученные данные, предла-

Таблица 1

3-компонентная система (сеточная часть)

Временной шаг	Ньютоновский шаг	Метод ⇒	выделение уравнения для давления(ILU для A_p)				ILU
		Параметр ILU, $\tau \Downarrow$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-3}
1	1	итерации	4	2	2	1	11
		время,10^{-2}	2	2	2	2	6
		память, kB	98	97	101	101	560
1	2	итерации	3	2	1	1	12
		время,10^{-2}	2	2	2	2	7
		память, kB	97	90	102	92	580
2	1	итерации	5	3	2	1	12
		время,10^{-2}	2	3	3	2	6
		память, kB	98	98	99	99	596

Таблица 2

21-компонентная система (сеточная часть)

Временной шаг	Ньютоновский шаг	Метод ⇒	выделение уравнения для давления(ILU для A_p)				ILU
		Параметр ILU, $\tau \Downarrow$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-3}
1	1	итерации	14	11	8	7	9
		время,10^{-2}	119	98	79	75	139
		память, kB	383	383	383	385	2402
1	2	итерации	22	13	10	9	11
		время,10^{-2}	174	112	95	90	145
		память, kB	383	383	386	387	2590
1	3	итерации	21	12	9	9	11
		время,10^{-2}	168	105	88	90	145
		память, kB	385	385	383	385	2607

Таблица 3

3-компонентная система

Временной шаг	Ньютоновский шаг	критерий остановки	$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-6}$	
		Метод	выделение уравнения для давления (ILU для $\tilde{A}_p$)	ILU	выделение уравнения для давления (ILU для $\tilde{A}_p$)	ILU
1	2	итерации	3	7	7	16
		время	0.42	0.48	0.5	0.7
		память, kB	9630	10050	9630	10050
1	3	итерации	3	7	8	17
		время	0.41	0.46	0.53	0.72
		память, kB	9708	10140	9708	10140
2	3	итерации	4	10	13	22
		время	0.53	0.64	0.79	0.99
		память, kB	10252	10700	10252	10700

гаемый метод на этих матрицах в каких-то случаях выигрывает по числу итераций и времени, а иногда уступает ILU.

Отсутствие выигрыша метода выделения уравнения для давления при наличии переменных, связанных со скважинами, объясняется ухудшением физических свойств матрицы системы (23) по сравнению с матрицей (15) из-за того, что внедиагональные блоки в матрице $\hat{A}$ системы (22) достаточно велики по отношению к диагональным. Отметим, что в случае многокомпонентных систем наблюдается существенный выигрыш по используемой памяти (табл. 4), в то время как доминирование однокомпонентной модели в системе (13) нивелирует это преимущество рассматриваемого метода (табл. 3).

Таблица 4

21-компонентная система

Временной шаг	Ньютоновский шаг	критерий остановки	$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-6}$	
		Метод	выделение уравнения для давления (ILU для $\tilde{A}_p$)	ILU	выделение уравнения для давления (ILU для $\tilde{A}_p$)	ILU
1	1	итерации	3	4	11	9
		время	0.39	0.50	0.72	0.61
		память, kB	207	2436	207	2436
1	2	итерации	6	4	13	11
		время	0.49	0.55	0.80	0.69
		память, kB	224	2631	224	2631
1	3	итерации	6	5	13	12
		время	0.49	0.58	0.81	0.72
		память, kB	226	2648	226	2648

Список литературы

[1] Азиз Х., Сеттари Э. Математическое моделирование пластовых систем. — М.: Недра, 1982.

[2] Cao H., Tchelepi H.A., Wallis J., Yardumian H. Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation // *SPE 96809, Proc. SPE Conference.* 2005, Dallas.

[3] Bell J., Trangenstein J., Shubin G. Conservation Laws of Mixed Type Describing Three-Phase Flow in Porous Media. // *SIAM J. Appl. Math.* 1986. V.46. P. 1000-1017.

[4] Diyankov O. The ILU-type preconditioner with 2 drop tolerance parameters // *Numerical and Computational Challenges in Science*

and Engineering, Workshop on Numerical Linear Algebra in Scientific and Engineering Applications October 29 – November 2, 2001.

[5] Lacroix S., Vassilevski Yu., Wheeler M., Wheeler J. Iterative solution methods for modeling multiphase flow in porous media fully implicitly // *SIAM J. Sci. Comp.* 2003. V.25. No.3. P.905-926.

[6] Lacroix S., Vassilevski Yu., Wheeler M. Decoupling preconditioners in the Implicit Parallel Accurate Reservoir Simulator (IPARS) // *Numer. Linear Algebra Appl.* 2001. V.8, No. 8. P.537-549.

[7] Trangenstein J., Bell J. Mathematical structure of the black-oil model for petroleum reservoir simulation // *SIAM J. Appl. Math.* 1989. V. 49. P. 749-783.

[8] Wallis J.R., Kendall R.P., and Little T.E. Constrained Residual Acceleration of Conjugate Residual Methods // *SPE 13536, presented at the SPE Resrv. Simul. Symp.* February 10-13, 1985. P. 415-428 — Dallas, Texas,

[9] Watts J. A compositional formulation of the pressure and saturation equations // *SPE Reservoir Engineering.* 1986. V.1. P. 243-252.

Технология построения тетраэдральных сеток для областей, заданных в САПР[a]

Ю. Василевский, А. Вершинин, А. Данилов, А. Пленкин

В работе изложена новая технология построения трехмерных сеток в областях с дискретными границами. Дискретизация границы задается наиболее общим форматом для технологий САПР — поверхностными триангуляциями.

1. Введение

При генерации сетки в трехмерной области необходимо определить ее трехмерную геометрию и топологию. Наиболее общий инструментарий для этого обеспечивается современными технологиями САПР. Данные о границе области можно получить либо во взаимодействии с геометрическим ядром САПР, либо в виде наиболее простых форматов (STL, VRML и т.д.). В первом случае доступна более детальная информация за счет привязки к геометрическому ядру [1]. Во втором случае общность данных и гибкость по отношению к типу САПР обеспечиваются за счет использования дискретной, а не кусочно-гладкой границы, что приводит к потере точности ее представления. С другой стороны, точность может быть повышена за счет бóльшего числа граничных элементов, выдаваемых САПР. Кроме того, точность приближения границы может быть повышена на основе кусочно-полиномиального восполнения дискретной границы [2].

В данной статье рассматривается алгоритм построения конформных тетраэдральных сеток внутри областей, заданных своей поверхностной триангуляцией. Последняя представляет собой простейший

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (код проекта 04-07-90336) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

формат выходных данных САПР. В приложениях геометрия поверхности может быть очень сложной, а исходная поверхностная сетка может содержать очень вытянутые треугольники. Построенная тетраэдризация должна быть:

- конформной (любые два тетраэдра пересекаются либо по общей целой грани, либо по целому ребру, либо в общей вершине, либо не пересекаются);
- по возможности регулярной (любой тетраэдр не сильно отличается от равностороннего, если это не диктуется геометрией границы);
- разгрубляющейся при удалении от границы.

Последние два свойства диктуются соображениями минимального объема данных для задания трехмерной сетки. Отметим, что, отталкиваясь от произвольной триангуляции границы области, нельзя получить сетку, обладающую указанными свойствами. Поэтому построение тетраэдризации производится в два этапа. Во-первых, строится новая триангуляция поверхности области, близкая к исходной, треугольники которой являются почти равносторонними (регулярными). Во-вторых, по новым граничным данным строится тетраэдризация области.

Построенная тетраэдральная сетка является начальной дискретизацией области, которой впоследствии можно придать желаемые свойства (априорно или апостериорно) [3, 4].

Предлагаемая технология построения тетраэдризации базируется на процедуре выделения почти плоских связных подобластей в исходной триангуляции (раздел 2.1), использовании метода продвигаемого фронта для покрытия их регулярной сеткой (раздел 2.2) и дальнейшего построения трехмерной сетки (раздел 3). Пример применения представленной технологии рассмотрен в разделе 4.

2. Построение регулярной триангуляции граничной поверхности

Целью первого этапа представляемой технологии является построение регулярной триангуляции, аппроксимирующей заданную конформную триангуляцию поверхности. Размер треугольников должен зависеть от кривизны поверхности: на участках большей кривизны треугольники должны быть более мелкими, чем на почти

плоских кусках поверхности. Подобная неоднородность поверхностной сетки обеспечит разумное распределение узлов при сохранении качества аппроксимации исходной поверхности.

Входными данными для первого этапа являются конформная триангуляция поверхности и возможная маркировка треугольников, обеспечивающая разделение поверхности на регионы. Каждый регион будет обрабатываться отдельно, причем геометрические границы регионов сохранятся в исходном виде. Работой алгоритма первого этапа управляют три вещественных параметра, один из которых отвечает за скорость разгрубления треугольников, два других влияют на построение полигональных подобластей.

Алгоритм построения регулярной триангуляции заданной дискретной поверхности представляет собой итерационную процедуру:

Пока исходная триангуляция не исчерпана:

1. Выделить связную подобласть (полигон), треугольники которой лежат в одной плоскости (в пределах заданной точности).

2. С помощью поворота плоскости полигона на плоскость OXY и обнуления Z-координат отобразить полигон на плоскость OXY.

3. Разбить границы полигона и применить процедуру построения двумерной регулярной триангуляции области с заданным следом на границе.

4. Спроектировать (по оси Z) каждый узел новой триангуляции на повернутый исходный полигон. Поскольку последний слабо отклоняется от плоскости OXY, качество спроектированных треугольников практически не ухудшается. Поворотом плоскости OXY на плоскость полигона завершить обратную замену координат.

5. Исключить из заданной триангуляции выделенный полигон.

Таким образом, сначала поверхность разбивается на подобласти, для которых задача сводится к двумерной, а потом из полученных триангуляций полигонов собирается триангуляция всей поверхности. Основные операции с полигоном продемонстрированы на рис. 1.

Изложенный выше алгоритм построения двумерной регулярной триангуляции может порождать два вида ошибок: при построении полигона и при генерации регулярной триангуляции. Для обеспечения робастности алгоритма используется метод возврата алгоритма

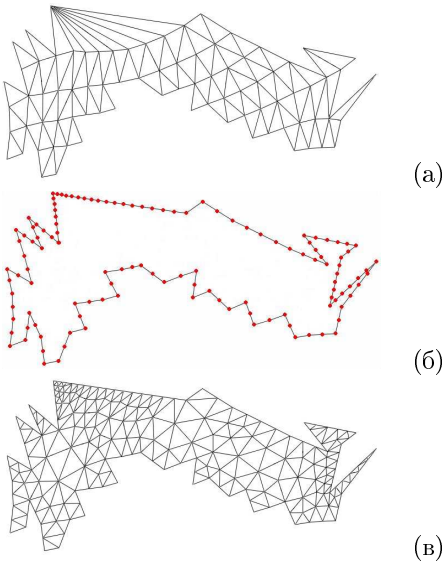


Рис. 1. Исходная сетка в полигоне (а), новое разбиение границы полигона (б), новая сетка в полигоне (в)

в случае возникновения ошибок, при котором полигон формируется заново с учетом типа возникающей ошибки (см. следующие разделы).

2.1. Выделение полигона и дискретизация его границы. Рассмотрим процедуру выделения полигональной подобласти более подробно. Полигональная подобласть — связное множество треугольников, принадлежащих с заданной точностью одной плоскости. Точность определяется максимально допустимым расстоянием от треугольника до плоскости h и максимальным отклонением между нормалью к треугольнику и нормалью к заданной плоскости ε.

Формирование полигона начинается с выбора начального треугольника. Его плоскость считается плоскостью будущего полигона, а граница назначается текущей границей полигона. Далее работает итерационный алгоритм построения полигона. Выбирается одно из граничных ребер полигона и проверяются все треугольники исходной сетки, которые граничат с полигоном через это ребро. Треугольник добавляется в полигон, если его метка (маркер) совпадает с меткой основного треугольника, расстояние от плоскости по-

лигона до любой из вершин треугольника меньше h и синус угла между нормалью к основному треугольнику и нормалью к кандидату меньше ε. Через одно ребро может быть добавлен только один треугольник. Если треугольник удовлетворяет всем условиям, то он добавляется в полигон и строится новая граница; в противном случае граничное ребро помечается как проверенное и рассматривается следующее. После того как все граничные ребра оказываются помеченными, полигон считается построенным.

Дискретизация границы полигона, задаваемая исходной триангуляцией, может оказаться сильно неравномерной, поскольку исходная триангуляция может быть нерегулярной (по форме треугольников). Неоднородность дискретизации границы неизбежно приведет к появлению треугольников с очень острыми углами вблизи границы либо даже к ошибке в построении регулярной сетки (см. следующий раздел). Поэтому при новом разбиении границы мы должны попытаться избавиться от резких перепадов шага разбиения.

Воспользуемся тем свойством (типичным для САПР) исходной сетки, что она аппроксимирует границу области с заданной пользователем точностью. В силу этого локальный размер треугольников (возможно анизотропных) исходной сетки уже согласован с кривизной поверхности, и для построения регулярной триангуляции достаточно использовать минимальные размеры треугольников исходной триангуляции. Определим для каждой вершины исходной триангуляции параметр шага как минимальную высоту из всех сходящихся в вершине треугольников. Параметр шага задает размер регулярной сетки в окрестности узлов граничных ребер выделенного полигона. Для каждого граничного ребра длины L зададим непрерывную функцию параметра шага как линейную сшивку заданных параметров шага w_A, w_B в концевых точках ребра. При этом расположение узлов новой дискретизации вдоль ребра AB задается формулой

$$\vec{r}_i = \vec{r}_A + \frac{\vec{r}_B - \vec{r}_A}{L} * (w'_A + d * i/2) * (i+1), \quad i = 1, \ldots, N,$$

$$d = \frac{w'_B - w'_A}{Q},\ w'_A = w_A * \frac{Q+1}{N+1},\ w'_B = w_B * \frac{Q+1}{N+1},\ Q = \frac{2*L}{w_A + w_B} - 1,$$

где N — целая часть Q.

Отметим, что после выделения полигона, его поворота и обнуления Z-координаты возможно самопересечение границы плоского полигона, который необходимо триангулировать. Подобная ситуация не позволяет использовать алгоритм продвигаемого фронта, см. следующий раздел. Поэтому в случае возникновения самопересечения

производится возврат алгоритма и полигон строится заново, причем при добавлении очередного треугольника в полигон дополнительно проверяется возможность самопересечения границы. Возврат алгоритма осуществляется и в случае сбоя в работе двумерного генератора новой триангуляции полигона. Это возможно в случае возникновения некритической ошибки генератора, которая не может быть самостоятельно обработана (см. следующий раздел). В этом случае необходимо упростить геометрию полигона: он перестраивается с параметрами $h/2$ и $\varepsilon/2$. При многократном повторении перестройки полигон в пределе устремится к плоскому фрагменту поверхности, для которого генератор регулярной сетки обязательно сработает.

2.2. Алгоритм построения регулярной триангуляции двумерной области. Рассмотрим генерацию двумерных триангуляций алгоритмом продвигаемого фронта [5]. Назначение алгоритма — генерация конформных двумерных триангуляций областей, заданных своими границами. Области могут быть как односвязными, так и многосвязными, однокомпонентными и многокомпонентными. Области должны быть ограниченными и иметь кусочно-гладкую границу без самопересечений, представленную своей дискретизацией (набором отрезков). След построенной триангуляции на границе области должен совпадать с исходной дискретизацией границы. Вещественный параметр алгоритма $c_f \geqslant 1$ задает желаемую скорость разгрубления треугольников по мере удаления от границы области.

На предварительном этапе алгоритма применяется сдвигово-масштабное преобразование координат, переводящее все вершины границы области внутрь единичного квадрата. Далее применяется алгоритм продвигаемого фронта, последовательно строящий треугольники сетки. Заданная дискретизация границы области формирует фронт. Фронт — набор отрезков, характеризуемый направлением своего возможного продвижения в сторону еще не триангулированной области. На каждом шаге алгоритма выбирается один отрезок из текущего фронта и на основании его строится треугольник. Этот треугольник вырезается из области, и меняется фронт. Процесс продолжается до тех пор, пока во фронте есть хотя бы одно ребро.

Рассмотрим один шаг алгоритма. Из фронта выбирается ребро с наименьшей длиной, и на его основании строится равнобедренный треугольник, высота которого определяется из желаемого размера сетки. У нового треугольника две вершины зафиксированы, а третью можно двигать, следя за тем, чтобы треугольник не пересекал

фронт. Далее рассматривается окрестность этой вершины, и если в нее попадают вершины из фронта, то среди них выбирается ближайшая и объявляется выделенной. Затем проверяется, не пересекает ли треугольник фронт, и если пересекает, то выбирается новая выделенная вершина — одна из вершин ребра, пересекшегося с треугольником. Этот процесс продолжается до тех пор, пока не найдется подходящий треугольник. При этом если было рассмотрено более 100 различных выделенных вершин, то процесс останавливается и вызывается обработчик ошибок.

После рассмотрения всех ребер во фронте к узлам построенной сетки применяется процедура сглаживания, корректируются топологии связей балансировкой числа сходящихся в узле ребер и применяется обратное сдвигово-масштабное преобразование в исходные координаты.

Алгоритм очень удобен для триангулирования областей, полученных программным путем, так как достаточно только указать желаемый след (возможно, неравномерный) на границе, формат входных данных просто организован и доступен один параметр, влияющий на процесс триангуляции.

Основные проблемы, которые могут возникнуть при реализации данного алгоритма, можно разделить на три группы. Во-первых, это ошибки, возникающие при нехватке памяти. Во-вторых, это ошибки в исходных данных. В-третьих, это некритические ошибки. Первые две группы ошибок для работы алгоритма заканчиваются полной остановкой. Третья группа ошибок передает управление обработчику ошибок, который пытается исправить положение. Как правило, некритические ошибки локальны, то есть они чувствительны к триангуляции в некоторой небольшой окрестности, а также зависят от параметра разгрубления. В случае появления некритической ошибки обработчик находит проблемное ребро, удаляет в уже построенной триангуляции все соседние с этим ребром треугольники, обновляет фронт, меняет параметр разгрубления и возвращает управление алгоритму триангулирования. Наш опыт показывает, что подобная релаксация делает алгоритм робастным при условии, что входные данные не противоречат задаче построения регулярной триангуляции.

Можно доказать, что для любой области алгоритм продвигаемого фронта работает за конечное время. Более того, необходимым и достаточным условием срабатывания алгоритма является требование, что граница области должна быть замкнутой, правильно ориенти-

рованной, ограничивать конечную область и не должна самопересекаться, а шаг заданной дискретизации границы меняется плавно.

3. Построение тетраэдризации трехмерной области

Рассмотрим алгоритм продвигаемого фронта [5] для генерации конформных тетраэдральных сеток трехмерной области, след которых на границе области совпадает с заданной поверхностной триангуляцией. Для минимизации объема сеточных данных размер тетраэдров должен увеличиваться при продвижении от границы области к ее центру.

Входными данными алгоритма является граничная сетка области, которая должна быть треугольной, конформной, правильно ориентированной, без самопересечений. Для правильной ориентации треугольника необходимо перечислить его вершины в порядке обхода по часовой стрелке, если наблюдать этот треугольник изнутри области. Допускается касание частей границы как по вершине, так и по ребру.

На предварительном этапе алгоритма применяется сдвигово-масштабное преобразование координат, переводящее все вершины границы области внутрь единичного куба. Далее применяется алгоритм продвигаемого фронта, последовательно строящий тетраэдры сетки. Заданная дискретизация границы области формирует фронт. Фронт — набор треугольников в пространстве, характеризуемый направлением своего возможного продвижения в сторону еще не тетраэдризованной области. На каждом шаге алгоритма выбирается треугольная грань из текущего фронта и на основании ее строится тетраэдр. Этот тетраэдр вырезается из области, и меняется фронт. Процесс продолжается до тех пор, пока во фронте есть хотя бы одна треугольная грань.

Рассмотрим один шаг алгоритма. Среди всех граней фронта выбирается рабочая грань с минимальной площадью — $\triangle ABC$, для которой из ее центра масс O проводится нормаль внутрь области и ставится четвертая точка D (кандидат на вершину будущего тетраэдра) так, чтобы $|OD| = c_f \cdot \sqrt{2/3}|\partial ABC|/3$, где $|\partial ABC|$ — периметр $\triangle ABC$, c_f — параметр разгрубления тетраэдров. Рассмотрим рабочую область — минимальный шар W с центром D, такой что вершины рабочей грани лежат в W, и сформируем список всех граней, которые попали в W. Среди всех вершин этих граней рассматриваются те, которые лежат в нужном полупространстве от рабочей

грани, а также попадают внутрь сферы, описанной вокруг построенного тетраэдра $ABCD$. Из рассмотренных вершин, если такие нашлись, выбирается ближайшая к точке O и называется кандидатом для точки D. Если же такой точки не нашлось, а в некоторой маленькой окрестности точки D есть вершины из фронта, то кандидатом для D становится ближайшая к ней вершина фронта. Таким образом на каждом шаге алгоритма выполнено следующее условие: расстояние между любыми двумя точками фронта не превосходит $\rho_{\min}$, где $\rho_{\min}$ — минимальное расстояние между вершинами в исходной дискретизации. Далее начинается перебор кандидатов вершины D для рабочей грани $\triangle ABC$. В переборе участвуют: текущий кандидат K, список граней из рабочей области $\triangle_1 \dots \triangle_n$ и список уже рассмотренных кандидатов $P_1 \dots P_m$ (в самом начале перебора $m = 0$). На очередном шаге перебора проверяется, пересекает ли тетраэдр $ABCK$ какую-нибудь из граней $\triangle_1 \dots \triangle_n$. Если пересечений нет, то этот тетраэдр лежит целиком внутри области, и на этом перебор останавливается. Если же нашлась такая грань $\triangle_i = \triangle A'B'C'$, с которой пересекся наш тетраэдр, то среди точек A', B', C' выбираются сначала те точки, которые не попали в список рассмотренных, лежат в нужном полупространстве и попадают в рабочую область, и среди выбранных точек выбирается точка с минимальным суммарным расстоянием до точек A, B, C — точка K'. Текущий кандидат записывается в список проверенных, вместо него кандидатом становится вершина K', и перебор продолжается. Если же ни одна из точек A', B', C' не подошла, то перебор останавливается, при этом тетраэдр не строится, а рабочая грань $\triangle ABC$ и грань $\triangle A'B'C'$ помечаются особым образом, при этом они остаются во фронте, но не могут быть выбраны на следующих шагах алгоритма в качестве рабочих граней.

Процесс построения тетраэдров продолжается, пока во фронте остаются обычные грани. Фронт из оставшихся помеченных граней можно разбить на несколько замкнутых частей, ограничивающих некоторые многогранники. Как правило, каждый многогранник получается невыпуклым, а количество его граней мало. Попытаемся найти такую точку внутри многогранника, из которой видны все его грани. Если это удалось, то разобьем многогранник на тетраэдры. В противном случае запустим процесс регуляризации, описанный ниже, найдем в уже построенной сетке все тетраэдры, имеющие общие вершины с проблемными многогранниками, удалим их из сетки, обновим фронт и применим алгоритм к обновленному фронту. Если

после нескольких таких рестартов сетка так и не построилась, то алгоритм останавливается с ошибкой.

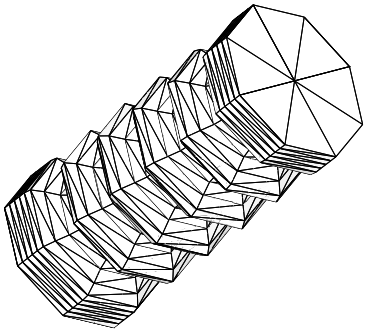
Регуляризация сетки сводится к итерационной сдвижке узлов в центр масс суперэлементов, состоящих из тетраэдров с этим общим узлом, и к изменению топологии тетраэдризации [6]. Для этого в цикле по внутренним граням сетки для каждой грани $\triangle ABC$ находятся два соседних тетраэдра $ABCD$ и $ABCE$ и среди соседних тетраэдров выбираются те, вершины которых лежат среди точек A, B, C, D, E. Рассматриваются два случая: когда количество тетраэдров равно двум и когда их количество равно трем (третьим тетраэдром, например, может оказаться тетраэдр $ACDE$, когда отрезок DE не пересекает $\triangle ABC$). Определим, задают ли найденные тетраэдры выпуклый шестигранник $ABCDE$, и если это так, то выберем лучшее из двух его разбиений: на два тетраэдра с общей гранью и на три тетраэдра с одним общим ребром.

По окончании успешной процедуры регуляризации применяется обратное сдвигово-масштабное преобразование в исходные координаты.

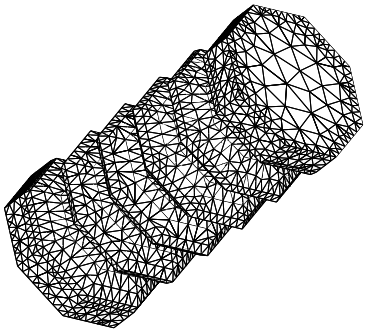
Обсудим некоторые вопросы устойчивости алгоритма. Можно доказать, что часть алгоритма продвигаемого фронта, обрабатывающая обычные грани, работает конечное время. Единственной проблемой для алгоритма являются те многогранники, в которых найти нужную точку для разбиения на тетраэдры не удается. На практике такие случаи встречаются редко, и они почти не встречаются после проведения нескольких попыток регуляризации сетки. Отметим, что если исходная поверхностная триангуляция такова, что для любых двух ребер с общей вершиной их длины отличаются не сильно, рассмотренный алгоритм построения тетраэдральной сетки срабатывает успешно. Именно поэтому исходная поверхностная сетка должна быть заменена на регулярную триангуляцию.

4. Пример применения предлагаемой технологии

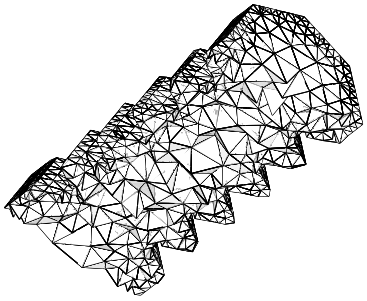
Рассмотрим деталь, заданную в некоторой САПР. Ее поверхностная сетка, состоящая из треугольников, характеризуется резкими скачками формы и размера треугольников (см. рис. 2,а). Трехмерный алгоритм генерации тетраэдральной сетки не может отработать с таким начальным фронтом. Алгоритм перестройки поверхностной сетки порождает регулярную триангуляцию поверхности детали, изображенную на рис. 2,б. Для такого начального фронта трехмерный алгоритм продвигаемого фронта срабатывает успешно. На



(а)



(б)



(в)

Рис. 2. Исходная поверхностная триангуляция (а), новая поверхностная триангуляция (б), часть тетраэдральной сетки (в)

рис. 2,в изображен разрез построенной тетраэдральной сетки. Подчеркнем, что построенная сетка должна рассматриваться лишь как начальная трехмерная дискретизация, которой потом можно придать желаемые свойства, поэтому свойство регулярности этой сетки не является критически важным.

Список литературы

[1] Баранов Л., Боровиков С., Иванов И., Крюков И. Опыт взаимодействия сеточного генератора с САПР // *Прикладная геометрия, построение расчетных сеток и высокопроизводительные вычисления.* 2004. Т.2. С.143-153. – М.: ВЦ РАН.

[2] Dyadechko V., Lipnikov K., Vassilevski Yu. Hessian based anisotropic mesh adaptation in domains with discrete boundaries // *Russ. J. Numer. Anal. Math. Modelling.* 2005. V. 20, No. 4. P. 391-402.

[3] Lipnikov K., Vassilevski Yu. Parallel adaptive solution of 3D boundary value problems by Hessian recovery // *Comp. Methods Appl. Mech. Engnr.* 2003. V.192, No. 11-12. P. 1495-1513.

[4] Василевский Ю. В., Липников К. Н. Оценки ошибки для управляемых адаптивных алгоритмов на основе восстановления гессиана // *ЖВМ и МФ.* 2005. Т.45, №8. С. 1424-1434.

[5] George P. L. Automatic mesh generation and finite element computation // *Handbook of Num. Anal.* 1996. V. 4. P. 127-148.

[6] Barth T. J. Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations: AGARD Report 787, 1992.

Параллельное трехмерное моделирование распространения примесей в пористых средах[a]

Ю. В. Василевский, И. В. Капырин

В работе рассмотрен конечноэлементный метод решения трехмерных задач конвекции-диффузии на неструктурированных тетраэдральных сетках. Описаны аспекты параллельной реализации на суперкомпьютере модели распространения примесей в пористых средах. Представлены экспериментальные результаты расчета тестового задания Couplex1 [1] и проведен их анализ.

1. Введение

Современные модели распространения примесей базируются на уравнении конвекции-диффузии. Для учета геометрических особенностей расчетной области, а также уменьшения численной диффузии и ошибок аппроксимации уравнения требуется большое число узлов сетки. При расчете переноса химических и ядерных загрязнений характерное время их распространения составляет до миллиона лет. Эти факторы предъявляют существенные требования к машинной памяти и скорости работы программы. Снять ограничения на память, накладываемые большим количеством степеней свободы, и достигнуть приемлемого времени выполнения программы позволяют параллелизация алгоритма и расчет на многопроцессорной ЭВМ.

Рассматриваемый здесь метод предложен Ж. Жаффре (INRIA) и основан на разных подходах к аппроксимации эллиптической и гиперболической частей пространственного оператора: в нем используются явная схема для конвективного члена и схема Кранка-Николсон для диффузионного. Метод обладает следующим рядом достоинств:

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (код проекта 04-07-90336) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

- консервативен благодаря непрерывности как конвективных, так и диффузионных потоков;
- формально обеспечивает второй порядок точности по времени и по пространству;
- требует на каждом шаге по времени решения одной глобальной линейной системы с симметричной положительно определенной матрицей;
- обладает приемлемыми ограничениями на устойчивость $\tau \sim h$ (по сравнению с полностью явными схемами, где $\tau \sim h^2$).

Для обеспечения устойчивости по отношению к сеточному числу Пекле используется технология «вверх по потоку» (upwind). Все шаги метода, за исключением решения глобальной системы смешанных конечных элементов, выполняются локально на тетраэдрах и, следовательно, легко распараллеливаются. Глобальная система линейных уравнений, степени свободы которой ассоциированы с гранями в сетке, решается с помощью метода минимальных невязок с параллельным предобусловливателем, основанным на декомпозиционном методе агрегирования [10].

Метод применяется к тестовой задаче Couplex1 о распространении ядерных загрязнений. Ее решение сопряжено с рядом трудностей:

- вытянутостью области в одном направлении;
- скачком коэффициентов диффузии на границе геологических слоев (на 8 порядков), что приводит к чередованию доминирующих процессов переноса;
- требованием отслеживать концентрацию загрязнителя в пределах от 10^{-12} до 10^{-1}.

Задача решалась на суперкомпьютере MVS-15000 с использованием от 4 до 16 процессоров.

Статья построена следующим образом: в первой части дана дифференциальная постановка модельной задачи и описан численный метод ее решения; во второй части рассматриваются аспекты параллелизации алгоритма; в третьей — постановка задачи Couplex1 и результаты, полученные при ее решении на разном числе процессоров.

2. Постановка задачи и численный метод ее решения

2.1. Задача конвекции-диффузии. Пусть Ω — ограниченная полиэдральная область в $\mathbb{R}^3$ с границей $\partial\Omega$. Рассмотрим следующую модельную задачу (для простоты изложения приняты однородные граничные условия типа Дирихле):

$$\begin{aligned} \frac{\partial C}{\partial t} - \nabla \cdot D\nabla C + \vec{b}\cdot\nabla C &= f \quad \text{в } \Omega\times(0,T), && \text{(1a)}\\ C &= 0 \quad \text{на } \partial\Omega\times(0,T), && \text{(1b)}\\ C &= g_0(x) \text{ при } t=0. && \text{(1c)} \end{aligned}$$

Здесь C — концентрация загрязнителя, $\vec{b}$ — бездивергентное векторное поле конвективного потока, $f\in L_2(\Omega)$ — функция источников или стоков, $D(x)$ — симметричный положительно определенный тензор диффузии.

Перейдем к смешанной постановке задачи:

$$\begin{aligned} \frac{\partial C}{\partial t} + \nabla\vec{r} + \vec{b}\cdot\nabla C = f \quad &\text{в } \Omega\times(0,T), && \text{(2a)}\\ \vec{r} = -D\nabla C \quad &\text{в } \Omega\times(0,T), && \text{(2b)}\\ C = 0 \quad &\text{на } \partial\Omega\times(0,T), \quad C=g_0(x) \text{ при } t=0. && \text{(2c)} \end{aligned}$$

В системе (2) введена новая переменная $\vec{r}$ — диффузионный поток. Обозначим через $(\cdot,\cdot)_S$ скалярное произведение в $L_2(S)$, причем $(\cdot,\cdot)\equiv(\cdot,\cdot)_\Omega$. Введем пространства

$$\begin{aligned} V &= L_2(\Omega),\\ W &= H(\mathrm{div},\Omega) = \{\vec{w}\in(L_2(\Omega))^3 : \quad \nabla\vec{w}\in L_2(\Omega)\}. \end{aligned}$$

Тогда можно переформулировать задачу (2) в слабой смешанной постановке: найти такую пару $(\vec{r},C)\in W\times V$, для которой

$$\begin{aligned} \left(\frac{\partial C}{\partial t}, v\right) + (\nabla\vec{r}, v) + \left(\vec{b}\cdot\nabla C, v\right) = (f,v), \quad \forall v\in V, && \text{(3a)}\\ \left(D^{-1}\vec{r},\vec{w}\right) - (C,\nabla\vec{w}) = 0, \quad \forall\vec{w}\in W. && \text{(3b)} \end{aligned}$$

2.2. Дискретизация задачи. Пусть ε_h — конформное разбиение Ω на тетраэдры, для которого пересечение любых двух тетраэдров может быть либо целой гранью, либо целым ребром, либо одной точкой, либо пустым множеством. Введем на сетке ε_h конечноэлементные пространства $V_h\in V$ и $W_h\in W$. Пусть V_h — пространство функций, разрывных на гранях тетраэдров и линейных

внутри них (разрывные конечные элементы). Ограничение $W_h \in W$ означает для любой внутренней грани непрерывность нормальной составляющей вектора $\vec{w}$ из W_h. Простейшим представителем W_h является пространство Равьера-Тома низшего порядка $RT_0(\Omega, \varepsilon_h)$ [2], базисные функции которого линейны внутри тетраэдров, имеют единичный поток через одну из граней сетки и нулевой — через все остальные. Ограничения V_h и W_h на тетраэдр E выглядят следующим образом:

$$\begin{aligned} V_h(E) &= P_1(E), \\ W_h(E) &= (P_0(E))^3 \oplus ((x,y,z) \cdot P_0(E)), \end{aligned}$$

где $P_i(E)$ — пространство многочленов i-й степени на E. Концентрация C и потоки $\vec{r}$ аппроксимируются в пространствах V_h и W_h соответственно:

$$C = \sum_{E \in \varepsilon_h} \sum_{i=1}^{4} C_{E,i} v_{Ei}, \tag{4}$$

$$\vec{r} = \sum_{e \in \partial\varepsilon_h} r_e \vec{w}_e. \tag{5}$$

Здесь v_{Ei} — базисные функции пространства $P_1(E)$; $\vec{w}_e$ — базисная функция пространства RT_0, имеющая единичный поток через грань e; r_e — диффузионный поток через эту грань. Смешанная конечно-элементная постановка задачи (3) имеет вид

$$\sum_{E \in \varepsilon_h} \left(\frac{\partial C}{\partial t}, v \right)_E + \sum_{E \in \varepsilon_h} (\nabla \vec{r}, v)_E + \sum_{E \in \varepsilon_h} \left(\vec{b} \cdot \nabla C, v \right)_E = \sum_{E \in \varepsilon_h} (f, v)_E, \quad \forall v \in V_h, \tag{6a}$$

$$\left(D^{-1} \vec{r}, \vec{w} \right) - (C, \nabla \vec{w}) = 0, \quad \forall \vec{w} \in W_h. \tag{6b}$$

Шаг предлагаемой схемы по времени состоит из трех подшагов (переменные интегрирования, объем dx и площадь ds в дальнейшем опускаются):

$$I.\quad \int\limits_E \frac{C^{n+\frac{1}{2}}-C^n}{\Delta t/2}m+\int\limits_E\left(\frac{3}{2}\operatorname{div}\vec{r}^n-\frac{1}{2}\operatorname{div}\vec{r}^{n-1}\right)m- \\ -\int\limits_E \vec{b}C^n\cdot\nabla m+\int\limits_{\partial E}\vec{b}C^n_{in}\cdot\vec{n}m=\int\limits_E f^n m \\ \forall m\in V_h(E),\quad \forall E\in\varepsilon_h. \tag{7a}$$

$$II.\quad \int\limits_E \frac{C^*-C^n}{\Delta t}m+\int\limits_E\left(\frac{\operatorname{div}\vec{r}^n+\operatorname{div}\vec{r}^{n+1}}{2}\right)m- \\ -\int\limits_E \vec{b}C^{n+\frac{1}{2}}\cdot\nabla m+\int\limits_{\partial E}\vec{b}C^{n+\frac{1}{2}}_{in\ or\ out}\cdot\vec{n}m=\int\limits_E f^{n+\frac{1}{2}}m, \tag{7b}$$
$$\forall m\in V_h(E),\quad \forall E\in\varepsilon_h,$$

$$\int\limits_\Omega D^{-1}\left(\frac{\vec{r}^n+\vec{r}^{n+1}}{2}\right)\vec{w}-\int\limits_\Omega\left(\frac{C^n+C^*}{2}\right)\operatorname{div}\vec{w}=0, \\ \forall\vec{w}\in W_h. \tag{7c}$$

III. Slope limiter $C^\star\longrightarrow C^{n+1}$.

На шаге (7a) явно вычисляется промежуточная концентрация, которая в (7b) используется для нахождения конвективных членов. На шаге (7b)-(7c) применяется явная схема по времени для аппроксимации конвективной части оператора и схема Кранка-Николсон для диффузионной части. При этом для стабилизации схемы в конвективном члене используется технология «вверх по потоку» (upwind), то есть в интеграле по грани концентрация $C^{n+1/2}_{in\ or\ out}$ берется с того тетраэдра, у которого конвективный поток в направлении внешней нормали к данной грани положителен. Для реализации шага (7b)-(7c) воспользуемся разложением пространства $V_h(E)$ на ортогональные подпространства: $V_h(E)=V^0(E)\oplus\tilde{V}^1(E)$, где $V^0(E)$ — константа, а $\tilde{V}^1(E)$ — линейные на E функции с нулевым средним. Тогда второй шаг распадается на два подшага:

$$IIa.\int\limits_E \frac{\bar{C}^*-\bar{C}^n}{\Delta t}v\quad+\int\limits_E\left(\frac{\operatorname{div}\vec{r}^n+\operatorname{div}\vec{r}^{n+1}}{2}\right)v+ \\ +\int\limits_{\partial E}\vec{b}C^{n+1/2}_{in\ or\ out}\cdot\vec{n}v=\int\limits_E f^{n+\frac{1}{2}}v, \\ \forall v\in V^0(E),\quad \forall E\in\varepsilon_h, \tag{8a}$$

$$\int\limits_{\Omega} D^{-1}\left(\frac{\vec{r}^n+\vec{r}^{n+1}}{2}\right)\vec{w} \quad -\int\limits_{\Omega}\left(\frac{\bar{C}^n+\bar{C}^*}{2}\right)\operatorname{div}\vec{w}=0, \qquad \forall\vec{w}\in W_h. \tag{8b}$$

$$IIb. \quad \int\limits_E \frac{\tilde{C}^*-\tilde{C}^n}{\Delta t}\tilde{v}-\int\limits_E \vec{b}C^{n+1/2}\cdot\nabla\tilde{v}+\int\limits_{\partial E}\vec{b}C^{n+1/2}_{in\ or\ out}\cdot\vec{n}\tilde{v}= =\int\limits_E f^{n+\frac{1}{2}}\tilde{v} \quad \forall\tilde{v}\in\tilde{V}^1(E), \qquad \forall E\in\varepsilon_h. \tag{8c}$$

Заметим, что такое разложение подпространств неприменимо на тетраэдрах, имеющих грани на границе типа Дирихле. Для таких тетраэдров мы применяем отдельную технологию, которая имеет только экспериментальное обоснование и поэтому не вошла в данную статью.

В силу определения пространства W_h диффузионные потоки не входят в (8c). Шаг (8c) выполняется локально по тетраэдрам с возможным использованием данных из соседних тетраэдров $(C^{n+1/2}_{in\ or\ out})$. В (8c) $\tilde{C}^*$ и $\tilde{C}^n$ — ортогональные константе линейные составляющие концентраций C^* и C^n.

В уравнениях (8a),(8b) $\bar{C}$ — средняя концентрация на тетраэдре. Система (8a),(8b) представляет собой стандартную постановку метода смешанных конечных элементов (МСКЭ). Аналогично [4] проведем гибридизацию системы МСКЭ. Для этого уберем ограничение непрерывности потоков тестовых функций $\vec{w}_h$ через грани, заменив W_h на $\tilde{W}_h=\{w: w|_E\in RT_0(E)\}=\bigotimes\limits_{E\in\varepsilon_h} RT_0(E)$, а также введем множители Лагранжа $\lambda\in L_h$, где

$$L_h=\left\{\mu\in L_2\left(\bigcup_{e\in\partial\varepsilon_h} e\right): \mu|_e\in W_h\cdot\vec{\nu_e} \quad \forall e\in\partial\varepsilon_h,\ \vec{\nu_e}\perp e\right\}.$$

Получаем гибридную постановку задачи:

$$\int\limits_E \frac{\bar{C}^*-\bar{C}^n}{\Delta t}v+\int\limits_E\frac{\operatorname{div}\vec{r}^n+\operatorname{div}\vec{r}^{n+1}}{2}v+\int\limits_{\partial E}\vec{b}C^{n+1/2}_{in\ or\ out}\cdot\vec{n}v= =\int\limits_E f^{n+\frac{1}{2}}v \quad \forall v\in V^0(E), \quad \forall E\in\varepsilon_h; \tag{9a}$$

$$\int_E D^{-1}\frac{\vec{r}^{n+1}+\vec{r}^{n}}{2}\vec{w} - \int_E \frac{\bar{C}^n+\bar{C}^*}{2}\operatorname{div}\vec{w} + \\ + \int_{\partial E\setminus\partial\Omega} \frac{\lambda^{n+1}+\lambda^n}{2}\vec{w}\cdot\vec{n} = 0 \quad \forall\vec{w}\in\tilde{W}_h(E), \quad \forall E\in\varepsilon_h; \tag{9b}$$

$$\sum_{E\in\varepsilon_h}\int_{\partial E\setminus\partial\Omega} \vec{r}^{n+1}\cdot\vec{n}\mu = 0 \quad \forall\mu\in L_h. \tag{9c}$$

Ассоциированная с (9) алгебраическая система принимает вид

$$\mathbf{A}r^{n+1} + \mathbf{B}\bar{C}^* + \mathbf{C}\lambda^{n+1} = F_1, \tag{10a}$$
$$\mathbf{B}^T r^{n+1} - \mathbf{G}\bar{C}^* = F_2, \tag{10b}$$
$$\mathbf{C}^T r^{n+1} = 0, \tag{10c}$$

где r^{n+1} — разложение $\vec{r}$ по базису $\tilde{W}_h(E)$, в F_1 и F_2 собраны все члены, содержащие источник f и значения переменных с временных слоев n и $n+1/2$. Обозначим через $\mathbf{A}_E, \mathbf{B}_E, \mathbf{G}_E$ сужения операторов $\mathbf{A}, \mathbf{B}$ и $\mathbf{G}$ на элемент E.

$$\mathbf{A}_{E,ij} = (D^{-1}\vec{w}_{E,i}, \vec{w}_{E,j})_E, \quad \mathbf{B}_{E,i} = -(1, \vec{w}_{E,i})_E, \quad \mathbf{G}_E = \left(1, \frac{1}{\Delta t}\right)_E.$$

Матрица $\mathbf{C}^T$ в (10c) отвечает за непрерывность диффузионных потоков r^{n+1} через грани сетки. Она создается путем ассемблирования ее локальных составляющих, элементы которых выражаются формулой

$$\mathbf{C}_{E,ij} = (1, \vec{w}_{E,i}\cdot\vec{\nu}_E^j)_{e_E^j},$$

где e_E^j — j-я грань тетраэдра E, $(j = 1,\ldots,4)$, ν_E^j — нормаль к ней. Уравнения (9a),(9b) формируются отдельно на каждом тетраэдре и не включают переменных с других элементов, поэтому (10a) и (10b) являются результатом ассемблирования локальных МСКЭ систем. В системе (10a)-(10c) производится исключение неизвестных. Во-первых, благодаря блочной структуре и положительной определенности матрицы $\mathbf{A}$ возможно исключение r^{n+1}:

$$\mathbf{B}^T\mathbf{A}^{-1}(F_1 - \mathbf{B}\bar{C}^* - \mathbf{C}\lambda^{n+1}) - \mathbf{G}\bar{C}^* = F_2, \tag{11a}$$
$$C^T\mathbf{A}^{-1}(F_1 - \mathbf{B}\bar{C}^* - \mathbf{C}\lambda^{n+1}) = 0. \tag{11b}$$

Во-вторых, так как матрица $\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ блочно-диагональная, исключим $\bar{C}^*$:

$$\begin{aligned}[\mathbf{C}^T\mathbf{A}^{-1}\mathbf{C} - \mathbf{C}^T\mathbf{A}^{-1}\mathbf{B}(\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}+\mathbf{G})^{-1}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{C}]\lambda^{n+1} = \\ = \mathbf{C}^T\mathbf{A}^{-1}F_1 + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{B}(\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}+\mathbf{G})^{-1}(F_2 - \mathbf{B}^T\mathbf{A}^{-1}F_1).\end{aligned} \tag{12}$$

В результате исключения получена система с положительно определенной симметричной матрицей, размерность которой совпадает с количеством граней в сетке. При этом исключение неизвестных производится на уровне одного тетраэдра. Определив из (12) λ, восстановим концентрации и потоки локально по формулам

$$\bar{C}^* = -(\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}+\mathbf{G})^{-1}(F_2 - \mathbf{B}^T\mathbf{A}^{-1}F_1 + \mathbf{B}^T\mathbf{A}^{-1}\mathbf{C}\lambda^{n+1}), \tag{13a}$$

$$r^{n+1} = \mathbf{A}^{-1}(F_1 - \mathbf{B}\bar{C}^* - \mathbf{C}\lambda^{n+1}). \tag{13b}$$

На третьем подшаге схемы (7a)-(7c) к полученному решению C^* для стабилизации применяется ограничитель наклона — лимитер [9]. Он реализован как минимизация функционала

$$J(C^{n+1}_{E,1},\ldots,C^{n+1}_{E,4}) = \sum_{i=1}^{4} \|C^{n+1}_{E,i} - C^*_{E,i}\|^2 \tag{14}$$

на каждом тетраэдре E с ограничениями

$$\bar{C}^{n+1}_E = \bar{C}^*_E, \qquad \min(i) \le C^{n+1}_{E,i} \le \max(i), \tag{15}$$

$\min(i)/\max(i)$ — минимальное и максимальное значения $\bar{C}^*_E$ на всех тетраэдрах, содержащих i-ю вершину. Подробно использование лимитера описано в [3].

3. Параллелизация алгоритма

3.1. Разбиение области по процессорам. Разбиение области между процессорами производится с помощью алгоритма инерциальной бисекции [11]. Сетка разрезается на непересекающиеся подобласти-слои, соответствующие разным процессорам. Число переменных равномерно распределяется между слоями. У каждого слоя, кроме первого и последнего, есть два соседних, что позволяет в большинстве случаев пересылать данные только двум процессорам-соседям.

3.2. Параллелизация локальных шагов. Первый шаг схемы (7) явный, выполняется локально на каждом тетраэдре и не использует межпроцессорных обменов. Подшаг (8c) требует информации о концентрации в точках соседнего тетраэдра, лежащего «выше по потоку» . Для этого шага и для лимитера (7c) необходим обмен значениями концентрации C в точках, лежащих на общей границе слоев.

При решении уравнений (8a)-(8b) матрица и правая часть глобальной системы (12) создаются путем локального исключения неизвестных и последующего ассемблирования глобальной матрицы и правой части. Исключение неизвестных (11),(12) делается независимо на каждом процессоре. Процедура ассемблирования состоит в сложении двух уравнений с соседних тетраэдров, соответствующих их общей грани. При этом на процессоре для внутренних тетраэдров мы получаем полноценные строки системы (12), а для пограничных — дисассемблированные, у которых недостающие данные находятся на другом процессоре. Этот факт необходимо учесть при умножении глобальной матрицы на вектор и инициализации предобусловливателя. Таким образом, глобальная система формируется параллельно. После ее решения восстановление потоков и концентраций осуществляется локально по тетраэдрам.

3.3. Параллелизация итерационного метода решения глобальной системы. Перепишем систему (12) в виде

$$M\lambda = b.$$

Для последовательного итерационного решения системы можно применить метод сопряженных градиентов с предобусловливателем, основанным на методе неполного разложения либо первого порядка ILUt [8], либо второго порядка ILU2 [6]. Для параллельного решения воспользуемся методом декомпозиции области. Разобьем множество элементов вектора λ на m непересекающихся подмножеств, ассоциированных с распределением тетраэдров по m процессорам (и m подобластям). Матрица M допускает блочное представление M_{ij}, $i,j = 1,...,m$, соответствующее этому разбиению. Пусть $L_{ii} = L_{ii}^T > 0$ — локальные последовательные предобусловливатели для диагональных блоков M_{ii}, $i = 1,\ldots,m$, а L_1 — блочно-диагональная матрица, состоящая из этих блоков. Матрица L_1 — аддитивный предобусловливатель Шварца для M с минимальными перекрытиями между подобластями. Его эффективность зависит от числа подобластей m, ширины налегания подобластей и скачков

диффузионного коэффициента. Для того чтобы уменьшить негативное влияние малого налегания и исключить зависимость от числа подобластей и скачков коэффициента, мы применим поправку на некотором грубом сеточном уровне:

$$L = L_1 + L_2(I - ML_1)$$

со следующим предобусловливателем L_2. Пусть $\tilde{n}_i$ — число ненулевых строк в матрице $M_i = [M_{i1}, \ldots, M_{i,i-1}, M_{i,i+1}, \ldots, M_{i,m}]$ (без диагонального блока). Определим

$$\tilde{n} = \sum_{i=1}^{m} \tilde{n}_i + m$$

и предположим, что строки матрицы M упорядочены таким образом, что в каждой матрице M_i ненулевые строки идут первыми. Тогда локальная матрица агрегирования $T_{ii} \in \mathbb{R}^{n_i \times (\tilde{n}_i + 1)}$ задается как

$$T_{ii} = \begin{pmatrix} I_i & 0 \\ 0 & e_i \end{pmatrix}, \quad e_i = (1, \ldots, 1)^T \in \mathbb{R}^{n_i - \tilde{n}_i},$$

где I_i — единичная матрица. Определим глобальную блочно-диагональную матрицу агрегирования T

$$T = \text{block diag}\{T_{11}, \ldots, T_{mm}\}$$

и матрицу жесткости $\tilde{M}$ на грубом подпространстве

$$\tilde{M} = T^T M\, T, \qquad \tilde{M} \in \mathbb{R}^{\tilde{n} \times \tilde{n}}.$$

Предобусловливатель L_2 определяется неявно через уравнение

$$L_2 = T\, \tilde{L}\, T^T,$$

где $\tilde{L}$ — последовательный ILU2 или ILUt предобусловливатель для $\tilde{M}$. Отметим, что L_2 — сглаживатель в пространстве агрегированных векторов $\tilde{V} = \{v \in \mathbb{R}^n\colon\ v = T\tilde{v},\, \tilde{v} \in \mathbb{R}^{\tilde{n}}\}$, которые могут целиком обрабатываться на одном процессоре, поскольку $\tilde{n} \ll n$. В силу того что предобусловливатель L несимметричен, будем использовать метод минимальных невязок для решения предобусловленной системы (12). Анализ предобусловливателя L представлен в работе [10], детальное рассмотрение одной его параллельной реализации можно найти в [7]. Параллельные обмены при применении рассматриваемого предобусловливателя сводятся к:

- локальной пересылке между процессорами-соседями для распределения результата применения L_1;
- локальной пересылке при вычислении невязки;
- глобальной пересылке на корневой процессор агрегированного вектора для применения $\tilde{L}$.

4. Результаты численного эксперимента

4.1. Постановка тестового задания Couplex1. Тестовое задание Couplex1 было предложено французским агентством по радиоактивным отходам «Андра» [1]. Изначально задача формулировалась как двумерная, мы же расширяем ее до трех измерений. Геометрия расчетной области представлена на рис.1. Она представляет собой параллелепипед $\Omega = (0, 25000) \times (0, 25) \times (0, 695)$ в метрах. Геологические слои расположены следующим образом:

- юрский известняк $0 < z < 200$;
- глина $200 < z < 295 + \frac{350-295}{25000}x$;
- известняк $295 + \frac{350-295}{25000}x < z < 595$;
- мергель $595 < z < 695$.

Источник загрязнений (контейнер с радиоактивными отходами) считается прямоугольным параллелепипедом, расположенным в глине $R = (x, y, z) \in (18440, 21680) \times (0, 25) \times (244, 250)$. Конвективные потоки подчиняются законам Дарси и неразрывности

$$\vec{u} = -K\nabla H, \qquad \operatorname{div} \vec{u} = 0, \tag{16}$$

где H — гидродинамическое давление, K — тензор проницаемости среды. Они посчитаны в рамках отдельной задачи и известны. Распределение гидродинамического давления показано на рис.2. Оно не зависит от координаты y, линии уровня идут по возрастанию слева направо. Конвективная скорость достаточно велика только в известняке и юрском известняке. В других слоях распространение загрязнения определяется преимущественно диффузионным процессом. Подробное описание поля скоростей для задачи Couplex1 можно найти в [5].

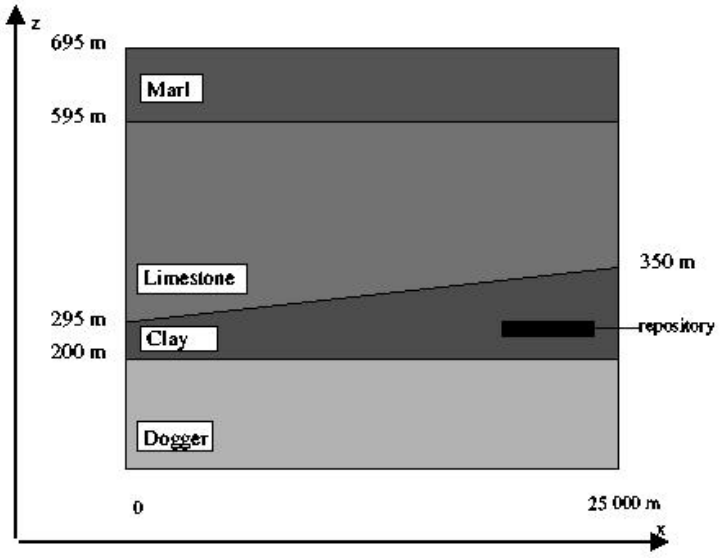


Рис. 1. Геометрия задачи Couplex1 в плоскости $x - z$

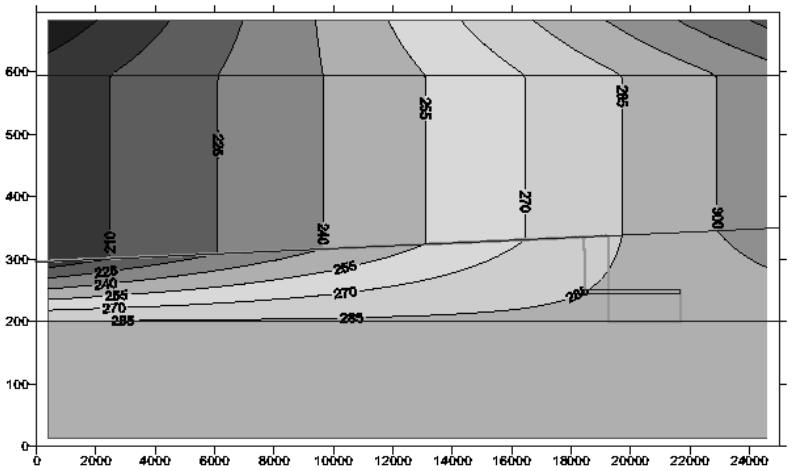


Рис. 2. Распределение гидродинамического давления.

В качестве радиоактивного загрязнителя рассматривается йод-129, характеристики источника заданы во времени (рис.3). Распространение загрязнения описывается следующим уравнением конвекции-диффузии-распада:

$$\omega\left(\frac{\partial C}{\partial t}+\log\frac{2}{T_I}C\right)-\nabla(D\nabla C)+\vec{u}\nabla C=f \quad \text{в } \Omega\times[0,T], \tag{17}$$

где

- ω — эффективная пористость среды; равна 0.001 в глине и 0.1 в остальных слоях;
- $T_I = 1.57\cdot 10^7$ лет — период полураспада ^{129}I ;
- D — эффективный тензор диффузии-дисперсии; определяется следующим образом:

$$D = d_e I + |\vec{u}|\left[\alpha_L E(\vec{u}) + \alpha_T (I - E(\vec{u}))\right], \quad E_{kj}(u) = \frac{u_k u_j}{|\vec{u}|^2}. \tag{18}$$

Значения коэффициентов, входящих в (18), представлены в табл. 1.

За начальное время $t = 0$ принят момент, когда контейнер начинает течь, что задает нулевые начальные условия для концентрации во всей области Ω. Граничные условия не зависят от времени:

$$\begin{aligned}
\frac{\partial C}{\partial \vec{n}} = 0 \quad & \text{на } \{0\}\times(0,25)\times(295,595),\\
& \{0\}\times(0,25)\times(0,200),\\
& (0,25000)\times\{0,25\}\times(0,695).\\
D\nabla C\cdot\vec{n} - C\vec{u}\cdot\vec{n} = 0 \quad & \text{на } (0,25000)\times(0,25)\times\{0\}.\\
C = 0 \quad & \text{на остальных границах.}
\end{aligned}$$

В задаче требуется проиллюстрировать поверхности уровня концентрации $10^{-12}, 10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}$ в разные моменты времени.

4.2. Результаты расчетов. Для расчета задачи Couplex1 была создана тетраэдральная сетка, содержащая 786 тысяч тетраэдров и 1.65 миллиона граней. Сетка имеет сгущения к источнику загрязнений и границе между глиной и известняком. Шаг схемы по времени составляет 25 лет (первые 10 шагов — по 100 лет), что обеспечивает устойчивость схемы на данной пространственной сетке. В каждом эксперименте делается 1000 шагов. Сравниваются времена счета на

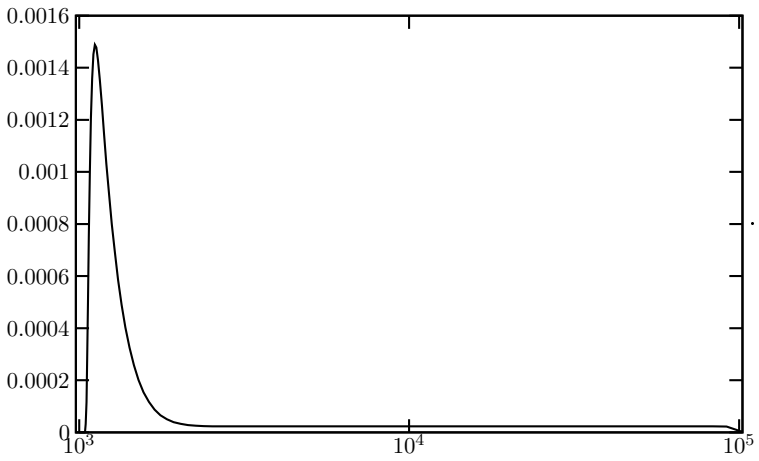


Рис. 3. Мощность источника как функция времени (в годах)

Таблица 1

Коэффициенты диффузии-дисперсии

	d_e(м^2/год)	α_L(м)	α_T(м)
Юрский известняк	5.0e-4	50	1
Глина	9.48e-7	0	0
Известняк	5.0e-4	50	1
Мергель	5.0e-4	0	0

Таблица 2

Результаты параллельных расчетов

число процессоров⇒ предобусловливатель⇓	4	8	16
ILU2 $\tau_1 = 10^{-3}$ $\tau_2 = 10^{-6}$	$T_4 = 25612c$ $\bar{N}_{iter} = 48$	$T_8 = 14283c$ $\bar{N}_{iter} = 46$ $S_8 = 0.9$	$T_{16} = 9466c$ $\bar{N}_{iter} = 47$ $S_{16} = 0.68$
ILU2 $\tau_1 = 10^{-4}$ $\tau_2 = 10^{-6}$	$T_4 = 14584c$ $\bar{N}_{iter} = 17$	$T_8 = 8223c$ $\bar{N}_{iter} = 18$ $S_8 = 0.89$	$T_{16} = 5722c$ $\bar{N}_{iter} = 18$ $S_{16} = 0.64$
ILU2 $\tau_1 = 10^{-4}$ $\tau_2 = 10^{-5}$	$T_4 = 15619c$ $\bar{N}_{iter} = 17$	$T_8 = 8282c$ $\bar{N}_{iter} = 18$ $S_8 = 0.94$	$T_{16} = 4943c$ $\bar{N}_{iter} = 19$ $S_{16} = 0.79$
ILUt $lfil = 100$ $droptol = 10^{-4}$	$T_4 = 17235c$ $\bar{N}_{iter} = 17$	$T_8 = 11191c$ $\bar{N}_{iter} = 18$ $S_8 = 0.77$	$T_{16} = 7057c$ $\bar{N}_{iter} = 19$ $S_{16} = 0.61$

разном числе процессоров, с разными вариантами преобусловливания в подобластях.

Расчеты проводились на суперкомпьютере MVS-15000, принадлежащем Межведомственному суперкомпьютерному центру РАН. MVS-15000 имеет 461 вычислительный узел. Каждый узел содержит два 64-разрядных процессора IBM PowerPC 970 с тактовой частотой 1600МГц и 4Гб оперативной памяти. Для обменов данными при счете задач используется сеть Myrinet с производительностью около 450Мбайт/с в режиме обмена «точка-точка» .

Результаты расчетов приведены в табл. 2. На рис. 4 и 5 представлено решение задачи в моменты времени 13250 лет (500 шагов) и 25750 лет (1000 шагов). На них изображены поверхности уровня, соответствующие концентрации $10^{-12}, 10^{-11}, 10^{-10}, 10^{-6}$, черным цветом закрашен источник загрязнения.

Из рисунков видно, что в глине загрязнение распространяется в вертикальном направлении. Доходя до известняка, оно сносится сильным горизонтальным конвективным потоком. В ячейках табл. 2 представлены время расчета на разном числе процессоров (T_4, T_8, T_{16}), среднее число итераций на шаг по времени $(\bar{N}_{iter})$ и показатели эффективности параллелизации $S_i (i \in \{8, 16\})$, опреде-

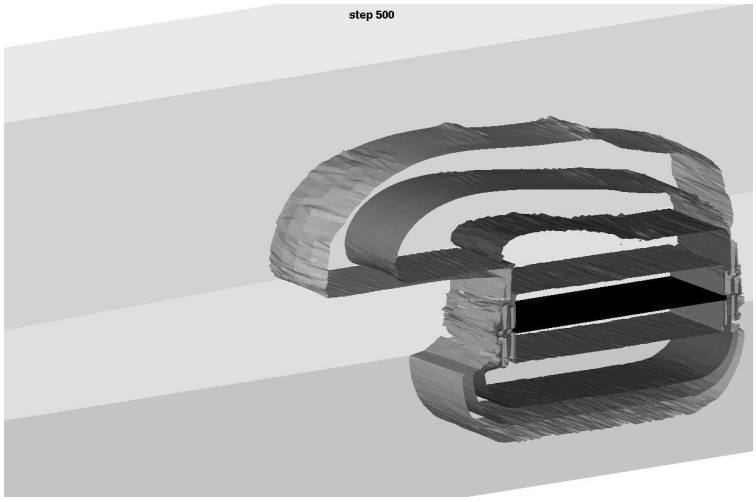


Рис. 4. Решение при t=13250

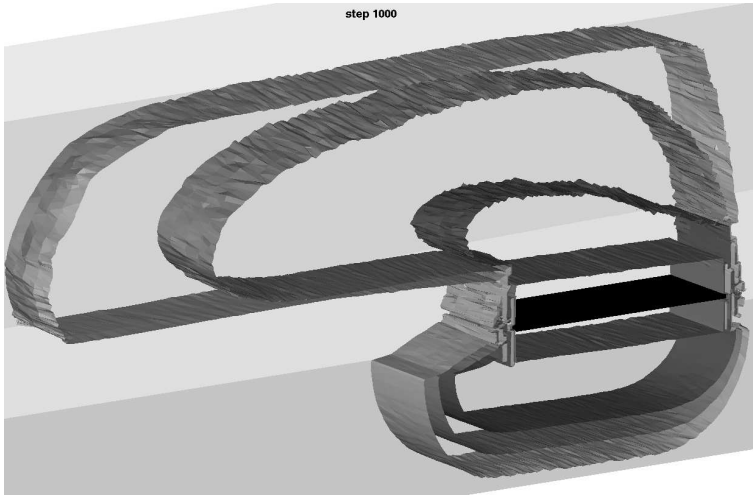


Рис. 5. Решение при t=25750

ляемые по формуле

$$S_i = \frac{T_4}{T_i} \cdot \frac{4}{i}, \qquad i = 8, 16 \ .$$

Критерием остановки итерационного метода является падение евклидовой нормы невязки до 10^{-20}. Малый рост числа итераций при увеличении числа процессоров говорит об эффективности параллельного предобусловливателя. Время расчета при увеличении числа процессоров уменьшается, однако эффективность параллелизации падает, так как повышается доля обменов данными в общем объеме работы. В то же время отметим, что с увеличением числа процессоров растет объем доступной памяти, что позволяет выбрать лучшие параметры предобусловливателя и значительно ускорить выполнение расчета.

Список литературы

[1] Bourgeat A., Kern M., Schumacher S., Talandier J. The COUPLEX test cases: Nuclear waste disposal simulation // *Computational Geosciences.* 2004. V. 8. P. 83-98.

[2] Brezzi F., Fortin M. Mixed and hybrid finite element methods. — New York: Springer Verlag, 1991.

[3] Chavent G., Jaffre J. Mathematial models and finite elements for reservoir simulation. — North Holland, Amsterdam: 1986.

[4] Chen. Z. Equivalence between and multigrid algorithms for nonconforming and mixed vethods for second-order elliptic problems // *East-West Journal of Numerical Mathematics.* 1996. V. 4. P. 1-33.

[5] Hoteit H., Ackerer Ph., Mose R. Nuclear waste disposal simulations: Couplex test cases // *Computational Geosciences.* 2004. V. 8. P. 99-124.

[6] Kaporin I. High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition // *Numer. Linear Algebra Appl.* 1998. V. 5. P. 483-509.

[7] Lipnikov K., Vassilevski Yu. Parallel adaptive solution of 3D boundary value problems by Hessian recovery // *Comp. Methods Appl. Mech. Engnr.* 2003. V.192. P.1495-1513.

[8] Saad Y. Iterative methods for sparse linear systems, Second Edition — Philadelphia, PA: SIAM, 2003.

[9] Siegel P., Mose R., Ackerer Ph., Jaffre J. Solution of the advection-diffusion equation using a combination of discontinuous and mixed finite elements // *International Journal for Numerical Methods in Fluids.* 1997. V. 24. P. 595-613.

[10] Vassilevski Yu. A hybrid domain decomposition method based on aggregation // *Numer. Linear Algebra Appl.* 2004. V. 11. P. 327-341.

[11] Williams R. Performance of dynamic load balancing algorithms for unstructured mesh calculations // *Concurrency: Practice and experience.* 1992. V. 3. P. 457-481.

Методы разложения тензора[a]

И. В. Оселедец, Д. В. Савостьянов

Проведен обзор и классификация существующих методов разложения трехмерного тензора, дано их краткое сравнение.

1. Введение

Разложение многомерных массивов оказалось очень важным и полезным инструментом во многих приложениях и теоретических исследованиях. Оказывается, что задачи разложения многомерных массивов нельзя решить с помощью методов, разработанных для матриц. Большинство трудностей, возникающих при решении этих задач, можно увидеть уже на примере трёхмерных массивов, о которых и пойдёт речь в этой статье.

Трёхмерные массивы впервые появились в факторном анализе в статистике [6] как массивы данных. Сейчас разложения трёхмерных массивов применяются в задачах обработки сигнала (signal processing), химии [1], психологии и социологии. Читателю, интересующемуся этой областью приложений многомерных массивов, мы рекомендуем обзор [2].

Ещё одной областью применения тензорных разложений является эффективное решение интегральных и дифференциальных уравнений на тензорных сетках (сетках, являющихся прямым произведением одномерных сеток). В этом случае даже полное вычисление элементов матрицы уже при не очень больших размерах является невыполнимой задачей, поэтому необходимо строить аппроксимации по малому множеству элементов трёхмерного массива. В настоящем

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

сборнике мы предлагаем алгоритм построения так называемого *разложения Таккера* трёхмерного массива [15].

В этой статье мы даём обзор существующих методов построения тензорных разложений и кратко обсуждаем их достоинства и недостатки.

2. Обозначения и определения

Мы будем рассматривать трёхмерные массивы (мы их также будем называть тензорами), т.е. объекты с тремя индексами:

$$\mathcal{A} = [\mathcal{A}_{ijk}], \qquad i = 1, \ldots, n_1, \quad j = 1, \ldots, n_2, \quad k = 1, \ldots, n_3,$$

где n_1, n_2, n_3 — *размеры тензора*. Тензоры мы будем обозначать буквами $\mathcal{A}, \mathcal{B}, \ldots$. Естественным образом определяется сумма двух тензоров, умножение тензора на число. Норму тензора мы определим как

$$\|\mathcal{A}\|_F = \left(\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \mathcal{A}_{ijk}^2 \right)^{1/2}.$$

Тензоры также можно умножать на матрицы по заданному направлению (индексу). Например, обозначение

$$\mathcal{B} = \mathcal{A} \times_i U,$$

где U — матрица размера $m \times n_1$, означает, что

$$\mathcal{B}_{ijk} = \sum_{i'=1}^{n_1} u_{ii'} \mathcal{A}_{i'jk}$$

и $\mathcal{B}$ — тензор с размерами $m \times n_2 \times n_3$. Операции $\mathcal{A} \times_j V, \mathcal{A} \times_k W$ определяются аналогично.

Важным понятием является *развёртка тензора по направлению*. Для трёхмерного массива таких развёрток три, по одной на каждый индекс. Например, развёртка по индексу k определяется как прямоугольная матрица размерами $n_3 \times n_1 n_2$ с элементами

$$A^{(3)}_{(ij)k} = \mathcal{A}_{ijk},$$

где строки матрицы $A^{(3)}$ пронумерованы двумя индексами. Развёртки по индексам i, j (обозначим их $A^{(1)}$ и $A^{(2)}$) определяются аналогично. Развёртывание тензора позволяет рассматривать его как

набор матриц-срезок. В частности, тензор $\mathcal{A}$ можно рассматривать как набор из n_3 матриц A_k размерами $n_1 \times n_2$, определяемых как

$$(A_k)_{ij} = \mathcal{A}_{ijk}.$$

Представление тензора как набора матриц позволяет записывать задачи тензорной аппроксимации на матричном языке, что часто (но не всегда!) позволяет использовать хорошо известные матричные алгоритмы для построения тензорных аппроксимаций или разложений.

Теперь определим, что же такое «тензорные разложения». Существуют два важнейших тензорных разложения: разложение Таккера и каноническое разложение. Разложением Таккера тензора $\mathcal{A}$ называется разложение вида

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{r_1} \sum_{\beta=1}^{r_2} \sum_{\gamma=1}^{r_3} g_{\alpha\beta\gamma} u_{i\alpha} v_{j\beta} w_{k\gamma}, \tag{1}$$

где матрицы $U = [u_{i\alpha}], V = [v_{j\beta}], W = [w_{k\gamma}]$ имеют ортонормированные столбцы. Массив $g_{\alpha\beta\gamma}$ называется *ядром* разложения Таккера. Разложение Таккера является одним из возможных обобщений сингулярного разложения матриц. Сингулярное разложение можно обобщить на трёхмерные массивы по-другому, с помощью так называемого *канонического разложения тензора*:

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}, \tag{2}$$

где матрицы U, V, W называются *опорными матрицами,* или просто *факторами* разложения (2), число r называется *тензорным рангом* тензора $\mathcal{A}$.

3. Свойства тензорных разложений

Приведём некоторые известные свойства тензорных разложений. Начнём с разложения Таккера. Если возможно *точное* представление тензора $\mathcal{A}$ в виде (1), то таккеровские матрицы U, V, W можно найти как левые («короткие») сингулярные векторы развёрток $A^{(1)}, A^{(2)}, A^{(3)}$:

$$A^{(1)} = U\Lambda_1\Phi_1^T, \qquad A^{(2)} = V\Lambda_2\Phi_2^T, \qquad A^{(3)} = W\Lambda_3\Phi_3^T.$$

Разложение Таккера всегда можно построить так, чтобы его ядро обладало важными дополнительными свойствами:

- *полная ортогональность*, т.е. для любого направления матрицы-срезки ортогональны во фробениусовом (евклидовом) скалярном произведении;

- *упорядоченность*, т.е. тот факт, что фробениусовы нормы матриц-срезок по любому направлению не возрастают с увеличением номера срезки.

В случае же лишь приближённого выполнения равенств (1) такой способ даёт ошибку порядка соответствующих сингулярных чисел матриц $A^{(1)}, A^{(2)}, A^{(3)}$, чего обычно бывает достаточно.

Если для нахождения U, V, W и $\mathcal{G}$ поставить задачу минимизации

$$\|\mathcal{A} - \mathcal{G} \times_i U \times_j V \times_k W\| \to \min,$$

то получающееся решение называется *HOSVD* (High-order Singular Value Decomposition) [8]. Имеющиеся алгоритмы вычисления HOSVD основаны на простом методе переменных направлений, который для данной задачи работает вполне приемлемо.

На данный момент мы ввели четыре различных ранга: r_1, r_2, r_3 и r. Легко видеть, что r_1 является числом линейно-независимых столбцов в тензоре $\mathcal{A}$, r_2 — строчек, а r_3 — векторов по третьему направлению (назовём их условно «распорки»). Вспомним, что ранг матрицы можно определять тремя способами: как столбцовый ранг, как строчной ранг и как минимальное возможное число слагаемых в представлении матрицы A в виде суммы *скелетонов*:

$$A = \sum_{\alpha=1}^{r} u_\alpha v_\alpha^T.$$

Хорошо известно, что все эти ранги равны. Перенести это утверждение на тензоры нельзя, так как числа r_1, r_2, r_3 в общем случае никак не связаны. Единственное, что можно сказать, что $r_1, r_2, r_3 \leqslant r$. Ранги r_1, r_2, r_3 (их ещё можно назвать *рангами мод,* или *модовыми рангами*) не превосходят n_1, n_2, n_3 соответственно. Однако тензорный ранг r может превышать линейные размеры тензора, что подтверждается простым подсчётом степеней свободы в (2). Число параметров в левой части равенства равно $n_1 n_2 n_3$, а в правой $(n_1+n_2+n_3)r$, поэтому оценка на максимально возможный ранг тензора при заданных размерах имеет следующий вид:

$$r \sim (n_1 n_2 n_3)/(n_1 + n_2 + n_3).$$

В частности, для кубических массивов ($n_1 = n_2 = n_3 = n$) получаем $r \sim n^2$. Точные верхние оценки на тензорный ранг до сих пор не получены. На данный момент известны ответы лишь для не очень больших размеров.

Другим удивительным фактом является то, что тензорный ранг r зависит от поля, в котором ищется решение, т.е. тензорный ранг одного и того же массива в $\mathbb{R}$ и $\mathbb{C}$ может быть разным. Далее, в случае матриц размера $n \times n$ множество матриц ранга меньше n имеет меру ноль, однако для тензоров это не так. Например, для размеров $2\times2\times2$ оказывается, что доля тензоров ранга 2 составляет 79%, а тензоров ранга 3 — 21%.

Пусть теперь нам нужно удовлетворить равенства (2) не точно, а всего лишь приближённо. В этом случае естественно поставить задачу о *наилучшем приближении ранга* r. В случае матриц замечательным методом нахождения такого приближения является сингулярное разложение. Однако для тензоров всё опять не так. Может даже случиться, что не существует наилучшего приближения заданного ранга, т.к. множество тензоров фиксированного тензорного ранга не замкнуто!

Приведенные выше свойства показывают, что каноническое разложение хоть и похоже по виду, но разительно отличается по свойствам от сингулярного разложения, причём все «хорошие» свойства разложения в двумерном случае теряются. Однако у канонического всё-таки есть одно «хорошее» свойство — свойство единственности. Его формулирует так называемая теорема Крускала.

Теорема Крускала. *Пусть ранги* U, V, W *из (2) равны* r_1, r_2, r_3, *и при этом:*

- *любые* r_1 *столбцов в* U *линейно независимы;*
- *любые* r_2 *столбцов из* V *линейно независимы;*
- *любые* r_3 *столбцов из* W *линейно независимы;*
- *и* $r_1 + r_2 + r_3 \geqslant 2r + 2$.

Тогда трилинейное разложение (2) единственно с точностью до умножения на числа и согласованных перестановок векторов в факторах.

Доказательство Крускала не являлось конструктивным. Более конструктивный аналог был получен в работе [5].

Закончим это параграф тем, что запишем каноническое разложение в терминах матриц-срезок A_k :

$$A_k = U\Lambda_k V^T, \qquad k = 1, \ldots, n_3, \tag{3}$$

где матрицы U и V это матрицы факторов, а матрицы Λ_k — диагональные матрицы порядка r, причём

$$(\Lambda_k)_{\alpha\alpha} = w_{k\alpha}.$$

Эквивалентность (2) и (3) легко проверяется. Разложение Таккера получается заменой в (3) диагональных матриц на произвольные.

Запись канонического разложения в форме (3) позволяет трактовать его как *задачу одновременного приведения матриц*, и это, в свою очередь, даёт основу для самых эффективных алгоритмов тензорных разложений на данный момент.

4. Методы построения тензорных разложений

4.1. Минимизационные методы. Среди всех методов построения канонического разложения можно выделить в отдельную группу «минимизационные» методы. «Минимизационными» мы будем называть те методы, в которых стандартные методы минимизации будут применяться для данной конкретной задачи аппроксимации тензора:

$$\|\mathcal{A}_{ijk} - \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}\| \to \min. \tag{4}$$

Метод переменных направлений [6, 7] (в английской литературе метод называется ALS) является самым простым способом решения задачи (2). Он состоит в следующем. Пусть даны некоторые приближения U, V, W к решению U^*, V^*, W^* проблемы (4). Тогда по V, W находится новое значение $\widehat{U}$ из решения задачи наименьших квадратов:

$$\widehat{U} = \arg\min_{U} \left(\|\mathcal{A}_{ijk} - \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}\|^2 \right). \tag{5}$$

Задача (5) распадается на n_1 независимых подзадач для каждого $i = 1, \ldots, n_1$:

$$\sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \left(\mathcal{A}_{ijk} - \sum_{\alpha=1}^{r} \widehat{u}_{i\alpha} v_{j\alpha} w_{k\alpha} \right)^2 \to \min. \tag{6}$$

Уравнения (6) представляют собой совокупность линейных переопределённых систем размерами $n_2 n_3 \times r$ с *с одной и той же матрицей* (обозначим её через Φ), но с разными правыми частями. Прямое применение стандартных методов (например, QR–разложения) к матрице Φ потребует $\mathcal{O}(n_2 n_3 r^2)$ арифметических операций. Однако для матрицы Φ можно быстро вычислить матрицу Грама

$$\Gamma = \Phi^T \Phi = (V^T V) \circ (W^T W),$$

где через $\circ$ обозначается адамарово (поэлементное) произведение матриц. Матрица Γ может быть получена за $\mathcal{O}((n_2 + n_3)r)$ арифметических операций. После этого для нахождения $\widehat{u}_{i\alpha}$ нужно решить n_1 систем с матрицей Φ. Стоимость такого вычисления составляет $\mathcal{O}(r^3 + n_1 r^2)$ операций. После того как новая матрица U найдена, аналогичным способом по матрицам U, W оценивается матрица V, и так далее до тех пор пока не будет достигнута сходимость.

Метод ALS несложно запрограммировать, стоимость одной итерации невелика, и на каждой итерации невязка не возрастает. Все эти достоинства делают рассматриваемый метод самым популярным методом вычисления тензорных аппроксимаций. Однако ALS имеет несколько очень существенных недостатков. Во-первых, число итераций, необходимых для получения удовлетворительных результатов, может быть очень большим (тысячи или даже миллионы). Во-вторых, и это самое главное, метод может попасть в локальный минимум. Однако если имеется хорошее начальное приближение к точке минимума, то применение метода может оказаться довольно полезным.

Существует несколько различных «улучшений» метода ALS. В частности, можно запускать его с различных случайных начальных приближений и сравнивать результаты, выбирая наилучший. Однако часто и это не помогает. Можно строить различные «регуляризации» метода, как, например, в [11].

Стандартным методом нелинейной оптимизации является метод Ньютона, поэтому естественно применить его для решения задачи (4). Однако на данный момент в литературе описан только метод для аппроксимации тензоров тензором единичного ранга [10]. Для случая матриц такой алгоритм был бы достаточен, т.к. наилучшую аппроксимацию ранга r можно получить путём последовательных приближений матрицами ранга 1. Однако для тензоров это неверно, поэтому желательно искать сразу приближение нужного ранга r. Такой алгоритм мы предлагаем в [18]. В той же статье мы так-

же строим метод Гаусса-Ньютона для минимизации (4) и проводим сравнение этих методов.

4.2. «Матричные методы». Минимизационные методы не учитывают специальный вид функционала, который требуется минимизировать, поэтому в некоторых условиях для построения канонического разложения гораздо более эффективными оказываются методы, которые мы условно назовём «матричными». Они основаны на представлении тензора $\mathcal{A}$ как набора матриц-срезок A_k. С этой точки зрения построение канонического разложения для тензора $\mathcal{A}$ равносильно одновременному представлению срезок A_k в виде

$$A_k = U\Lambda_k V^T, \tag{7}$$

где U, V — матрицы, совпадающие с факторами U, V канонического разложения, а элементы диагональных матриц Λ_k, будучи записанными в строки, все вместе составляют фактор W канонического разложения.

Для описания первого метода поиска такого разложения будем считать матрицы U, V квадратными и невырожденными. Тогда, взяв некоторую линейную комбинацию

$$B = \sum_k c_k A_k = U \sum_k (c_k \Lambda_k) V^T = U\Lambda V^T$$

и «поделив» все равенства (7) на B, получим

$$A_k B^{-1} = U\Lambda_k \Lambda^{-1} U^{-1}.$$

Обозначив $A_k B^{-1} = B_k$, видим, что задача сведена к одновременному приведению всех матриц B_k с помощью одного и того же преобразования подобия к диагональному виду (задача одновременной редукции!):

$$B_k = A_k B^{-1} = U\tilde{\Lambda}_k U^{-1}.$$

«Качество» такого приведения зависит от двух факторов. Во-первых, матрица B может быть близка к вырожденной, поэтому любая ошибка (возникающая, например, если точного разложения не существует, но есть аппроксимация с некоторой точностью) умножится на $||B^{-1}||$, что может привести к росту погрешности. Поэтому коэффициенты c_k надо выбирать аккуратно. Второй вопрос состоит в том, как находить матрицу подобия. Самый простой подход таков: возьмём любую матрицу, например B_1, и найдём её собственное разложение:

$$B_2 = SDS^{-1}.$$

Если собственные значения матрицы B_2 различны, а все матрицы B_k можно привести к диагональному виду, то можно показать, что такой выбор будет удачным. Если же у B_2 есть кратные собственные значения, то нужно опять брать некоторую линейную комбинацию матриц B_k и уже у неё вычислять собственное разложение.

Требование невырожденности матриц U и V можно ослабить, «разрешив» им быть прямоугольными, но полного столбцового ранга. В этом случае с помощью редукции Таккера нужно перейти к ядру, для которого матрицы U и V становятся квадратными.

На практике обычно делают так: вычисляют разложение Таккера, выбирают первые две срезки A_1 и A_2 (эти срезки имеют две наибольшие нормы) и решают два уравнения

$$A_1 = U\Lambda_1 V^T, \qquad A_2 = U\Lambda_2 V^T.$$

Несложно увидеть в этих двух уравнениях *обобщённую задачу на собственные значения* для матричного пучка $\lambda_1 A_1 + \lambda_2 A_2$. Для нахождения левых и правых собственных векторов можно (и даже нужно!) использовать стандартные процедуры, основанные на QZ-алгоритме. После того как U и V найдены, матрица W оценивается из решения линейной задачи наименьших квадратов. Полученный результат можно использовать как хорошее начальное приближение для минимизационных методов нахождения тензорной аппроксимации, таких как метод ALS или Ньютона.

Естественным развитием идеи использования обобщённого собственного разложения является использование *суперобобщённого разложения Шура*. Кратко поясним, о чём идёт речь. Одним из недостатков описанной выше процедуры является то, что необходимо оперировать с неортогональными матрицами: вычислять обратные матрицы, вычислять собственное разложение. Всё это может привести к тому, что вычисленное таким образом приближение может сильно отличаться от истинного. Поэтому возникает естественная идея переписать уравнения в терминах ортогональных матриц. Представим матрицы U и V как $U = QR$, $V = ZL$, где Q и Z — ортогональные матрицы, R — верхнетреугольная а L — нижнетреугольная. Тогда

$$Q^T A_k Z = T_k = R\Lambda_k L^T,$$

где T_k — верхнетреугольные. Теперь построение разложения состоит из двух этапов. На первом этапе находятся ортогональные матрицы Q и Z такие, что матрицы

$$T_k = Q^T A_k Z \tag{8}$$

максимально возможно верхнетреугольные («суперобобщённое разложение Шура»). Вторым этапом решается задача

$$T_k = R\Lambda_k L^T. \tag{9}$$

Уравнения (9) сводятся к серии линейных задач наименьших квадратов [14], и их решение не представляет никакой сложности.

Для нахождения суперобобщённого разложения Шура существует несколько методов. Первый алгоритм — это EQZ-алгоритм (extended QZ algorithm), предложенный в [13]. Его идея состоит в следующем. Сначала находится матрица Q такая, что QA_k максимально верхнетреугольные. Матрица Q ищется в виде произведения нескольких матриц отражений. Первая матрица отражения H выбирается так, чтобы минимизировать

$$\sum_k \sum_{i=2}^{n} (HA_k)_{i1}^2 \to \min,$$

вторая — чтобы минимизировать сумму квадратов элементов во вторых столбцах, начиная с третьего элемента и т.д. После этого аналогичным образом находится матрица Z и процесс продолжается, пока не будет достигнута сходимость. Описанный EQZ-алгоритм является обобщением QZ-алгоритма, однако не так эффективен. Основная причина этого состоит в том, что для EQZ-алгоритма совершенно непонятно, как реализовывать сдвиги, а именно сдвиги делают стандартный QZ-алгоритм действительно эффективным.

Другой способ состоит в том, чтобы искать матрицы Q и Z в виде произведений матриц вращений. В этом случае, как было показано в [14], на каждом шаге нужно находить корень многочлена 8-й степени.

В [16] мы предлагаем третий, более быстрый вариант, основанный на решении суперобобщённой задачи на собственные значения.

Упомянем о ещё одном интересном подходе. Ограничимся симметричным случаем, когда $V = U$. Опять будем предполагать, что U является невырожденной квадратной матрицей. Тогда перепишем уравнения в виде

$$U^{-1} A_k U^{-T} = \Lambda_k \tag{10}$$

и, вводя новые переменные $X = U^{-1}$, получим следующую минимизационную проблему:

$$||XA_kX^T||_{\text{off}} \to \min,$$

где $||\cdot||^2_{\text{off}}$ — сумма квадратов всех внедиагональных элементов матрицы.

В работе [12] для нахождения матрицы X был предложен следующий шаг обновления. Пусть X — имеющееся приближение к решению, тогда будем искать новое значение X' в виде

$$X' = (I + W)X,$$

где $W_{ii} = 0$. Также разобьём матрицы $C_k = XAX^T$ в сумму диагональной и внедиагональной частей:

$$C_k = D_k + E_k.$$

Тогда

$$C'_k = (I + W)(D_k + E_k)(I + W^T) \approx D_k + E_k + WD_k + D_kW^T,$$

где были отброшены малые слагаемые WW^T и, что самое важное, WE_k и E_kW^T, так как мы предполагаем, что внедиагональная часть мала. После это нужно решить предопределённую линейную систему относительно W. Оказывается, что для элементов W можно написать короткие формулы [12]:

$$W_{ij} = \frac{z_{ij}y_{ji} - z_{ii}y_{ij}}{z_{jj}z_{ii} - z_{ij}^2}, \qquad i < j,$$

$$W_{ji} = \frac{z_{ij}y_{ij} - z_{jj}y_{ji}}{z_{jj}z_{ii} - z_{ij}^2}, \qquad i > j,$$

где

$$z_{ij} = \sum_k (D_k)_{ii}(D_k)_{jj}, \qquad y_{ij} = \sum_k (D_k)_j(E_k)_{ij}.$$

Вычисления производятся очень быстро и скорость сходимости почти квадратичная. Однако здесь, при плохо обусловленной матрице X, матрица $U = X^{-1}$ может очень сильно отличаться от истинного решения минимизационной задачи (4).

Все вышеперечисленные методы ограничены случаем, когда матрицы U, V имеют полные столбцовые ранги. Для таких тензоров ядро Таккера имеет размеры $r \times r \times q$ и тензорный ранг r. Для задачи разложения тензоров, у которых тензорный ранг оказывается больше их размера, найденные нами в литературе «матричные» методы оказываются не применимы. В [17] мы предлагаем такой метод для случая тензорного ранга, превышающего размер.

Заключение

В данной статье мы дали обзор алгоритмов для построения канонического разложения тензоров. Используя эти методы, можно строить разложения тензоров не очень большого ранга и размеров максимум порядка нескольких сотен. Для тензоров же большого размера и/или высокого ранга построение разложения с помощью описанных методов затруднительно.

Если тензор имеет «приемлемые» размеры, то на первый план выходит проблема *устойчивого* нахождения тензорной аппроксимации. При применении стандартных минимизационных методов со случайного начального вектора возможно попадание в локальный минимум — это их основной недостаток. Однако хорошее начальное приближение можно получать с помощью описанных матричных методов.

Работа с очень большими массивами требует принципиально иной техники, основанной на так называемом *крестовом методе*[15], позволяющем получать разложения Таккера этих массивов за почти линейное по их размеру время (даже не вычисляя полностью всех их элементов), после чего получившиеся небольшие ядра Таккера можно «дожимать» с помощью приведённых в данной статье методов.

Список литературы

[1] Ибрагимов И.В. Новый подход к решению задачи обобщённого сингулярного разложения // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 193–201.

[2] Comon P. Tensor decomposition: State if the Art and Applications // *In IMA Conf. mathematics in Signal Processing*, Warwick, UK, Dec. 18-20,2000.
`http://www.i3s.fr/~comon/FichiersPs/ima2000.ps`

[3] Tucker L. R. Some mathematical notes on three-mode factor analysis // *Psychometrika.* 1966. V. 31. P. 279-311.

[4] 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.

[5] Leurgans S., Ross R. T., Abel R. B. A decomposition for three-way arrays // *SIAM J. Matrix Anal. Appl.* 1993. V. 14. P. 1064-1083.

[6] 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.

[7] 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.

[8] De Lathauwer L., de Moor B., Vandewalle J. A multilinear singular value decomposition // *SIAM J. Matrix Anal. Appl.* 2000. V.21. P. 1324-1342.

[9] 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.

[10] Zhang T., Golub G.H. Rank-one approximation to high-order tensors // *SIAM J. Matrix Anal. Appl.* 2001. V. 23. P. 534-550.

[11] Wang J.-H., Hopke P.K., Hancewicz T.M., Zhang S.L. Application of modified least squares regression to spectroscopic image analysis. // *Anal. Chim. Acta.* 2003. V. 476. P. 93-109.

[12] Ziehe A., Kawanabe M., Hamerling S., Müller K.-R. A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation // *Journal of Machine Learning Research.* 2004. V. 5. P. 801-818.

[13] A.-J. Van der Veen and A. Paulraj, An analytical constant modulus algorithm // *IEEE Trans. Signal Process.* 1996. V. 44. P. 1136-1155.

[14] L. De Lathauwer, B. De Moor and J. Vanderwalle. Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition // *SIAM J. Matrix Anal. Appl.* 2004. V. 26. P. 295-227.

[15] Оселедец И. В., Савостьянов Д. В. Трёхмерный аналог алгоритма крестовой аппроксимации и его эффективная реализация // *Настоящий сборник.* С. 65-99.

[16] Оселедец И. В., Савостьянов Д. В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Настоящий сборник.* С. 101-116.

[17] Оселедец И. В., Савостьянов Д. В. Об одном алгоритме построения трилинейного разложения // *Настоящий сборник.* С. 117-130.

[18] Оселедец И. В., Савостьянов Д. В. Минимизационные методы разложения тензора и их сравнение // *Настоящий сборник.* С. 131-146.

Трехмерный аналог крестового метода аппроксимации и его эффективная реализация[a]

И.В. Оселедец, Д. В. Савостьянов

Рассмотрена задача вычисления для заданного трехмерного массива размером $n \times n \times n$ его аппроксимации разложением Таккера. Показано, как обобщить крестовый метод аппроксимации матриц на случай трехмерных массивов. Предложен алгоритм, позволяющий в определенных предположениях построить разложение заданного массива за почти линейное по n число действий, при этом не вычисляя даже всех его элементов. На численных примерах доказана надежность и эффективность этого метода. Применение описанного алгоритма позволяет находить для массива, который при плотном хранении занимал бы 2 петабайта, его структурированную аппроксимацию, на хранение которой требуется менее $100\,Mb$ памяти. Такое вычисление может быть проведено на обычной персональной станции за время порядка часа.

1. Введение

Во многих прикладных задачах (например, связанных с решением интегральных уравнений) возникает необходимость решать линейные системы $Ax = y$ с плотными матрицами больших (порядка сотен тысяч или миллиона) или очень больших (превышающих десять миллионов) размеров. В случаях, когда рассматриваемая матрица имеет специальный вид (скажем, является теплицевой), можно построить вычислительные алгоритмы, сложность которых будет

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

почти линейна по размеру матрицы. Однако в случае, когда матрица не обладает заранее определенной структурой, решение линейной системы может быть при таких размерах слишком сложным даже для современной вычислительной техники. В общем случае для решения таких задач матрицу A аппроксимируют матрицей специального вида $\tilde{A}$ и предлагают быстрые алгоритмы для решения задачи с матрицей $\tilde{A}$. При использовании для решения линейной системы итерационных методов (по сути, это основной способ работы с большой плотной матрицей), вопрос сводится к построению быстрого алгоритма умножения на матрицу $\tilde{A}$. Большинство известных методов используют для этой цели явный вид ядра интегрального оператора [1, 2, 3, 4], матричная трактовка этих методов рассматривается в [5, 6, 7, 8], а методы аппроксимации, использующие только элементы матрицы A, рассматриваются в [9, 10, 11, 12, 13].

Не так давно предложено искать аппроксимации матрицы A в виде суммы тензорных произведений [14, 15, 16]

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

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, [17]). Пусть сомножители A_α^s, $s = 1, \cdots, m$, являются квадратными матрицами размера p. Тогда исходная 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, [18]).

В большинстве случаев число кронекеровских сомножителей m выбирается равным размерности физического пространства. В нашей работе мы будем обсуждать случай $m = 3$. В часто обсуждаемом случае, когда элементы a_{ij} матрицы A 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, индексы i, j матрицы A можно выразить с помощью мультииндексов (многомерных индексов)

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

Тут 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. В некоторых случаях такое представление индексов элемента матрицы с помощью мультииндекса вводят более искусственно. В любом случае, как только мы имеем матрицу, элементы которой заданы с помощью мультииндексов, мы можем расщепить и перегруппировать компоненты мультииндексов следующим образом:

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

вводя новые парные индексы $\mathrm{i} = (i_1, j_1)$, $\mathrm{j} = (i_2, j_2)$, $\mathrm{k} = (i_3, j_3)$. 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

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

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

массив a_{ijk} должен обладать представлением в виде

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

называемым еще *трилинейным разложением.*

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

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

рассматривают как задачу минимизации нормы отклонения

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

Для минимизации этого функционала могут применяться различные стандартные итерационные методы минимизации [19, 28, 31], 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 ее решения. Другая сложность — необходимость априорно знать ранг трилинейной аппроксимации, что в большинстве случаев невозможно, или последовательно решать серию задач минимизации (2) с разными r.

Под «матричными» методами получения разложения (1) 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. [28, 29]), они обладают сложностью порядка n^4, однако условие $r = n$ достаточно сильно ограничивает их применение, поскольку в практических случаях ничем не гарантируется, что даже наилучшее трилинейное разложение ранга $r = n$ для массива $\mathcal{A}$ обладает требуемой 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 [30], его сложность $\mathcal{O}(n^4)$. Суммируя сказанное и принимая в рассмотрение 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 $n = 128$.

В качестве промежуточного (впрочем, иногда и достаточного) шага при поиске трилинейного разложения рассматривают поиск разложения Таккера [20]

$$a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'}, \tag{3}$$

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 отыскание для заданного массива $\mathcal{A} = [a_{ijk}]$ разложения Таккера, размеры ядра которого много меньше, чем размеры исходного массива, с тем, чтобы к этому ядру возможно было применить упомянутые выше методы поиска трилинейного разложения. Если для ядра $\mathcal{G}$ удается найти трилинейное разложение, это сразу означает, что тем самым найдено трилинейное разложение того же ранга для исходного массива $\mathcal{A}$.

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

$$\begin{aligned} A^{(1)} &= [a^1_{i(jk)}] = [a_{ijk}], \\ A^{(2)} &= [a^2_{j(ki)}] = [a_{ijk}], \\ A^{(3)} &= [a^3_{k(ij)}] = [a_{ijk}], \end{aligned} \tag{4}$$

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

$$A^{(1)} = U\Sigma_1\Phi_1^T, \qquad A^{(2)} = V\Sigma_2\Phi_2^T, \qquad A^{(3)} = W\Sigma_3\Phi_3^T \tag{5}$$

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

$$g_{i'j'k'} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_{ijk} u_{ii'} v_{jj'} w_{kk'}. \tag{6}$$

Сложность SVD-разложений составляет $\mathcal{O}(n^4)$, что для наших целей тоже слишком много. 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.

Между тем, для аналогичной по идее задачи аппроксимации обычной матрицы $A = [a_{ij}]$ размера $n \times n$ матрицей малого ранга известен алгоритм *крестовой аппроксимации* [8, 13], не требующий вычисления всех элементов исходной матрицы, а использующий лишь процедуру генерации этих элементов. Пользуясь этой процедурой, алгоритм крестовой аппроксимации вычисляет порядка $n \log^{\beta} n \log^{\gamma} \varepsilon^{-1}$ элементов, где положительные числа β и γ зависят от свойств матрицы A, а ε определяет требуемую точность аппроксимации. В нашей работе мы предлагаем алгоритм, являющийся обобщением крестового метода аппроксимации на случай не матриц, а трехмерных массивов. При построении нового метода мы сохраняем основной принцип крестового метода: воспринимаем трехмерный массив $\mathcal{A}$ не как заданный набор из n^3 чисел, а как процедуру, по заданным индексам i, j, k вычисляющую элемент a_{ijk}. За счет этого предлагаемый алгоритм обладает той же, почти линейной по n оценкой сложности, то есть строит аппроксимацию $\tilde{\mathcal{A}}_r$ 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 [15, 16]. Мы же при описании алгоритма будем предполагать, что искомая аппроксимация с заданной точностью у рассматриваемого массива существует, и остановимся на способе ее отыскания. Здесь и далее возникающая в оценках сложности метода величина r будет означать ранг (ε-ранг) трилинейного разложения для исходного массива. В большинстве практических случаев в задачах, связанных с решением интегральных уравнений, этот ранг

можно оценить как $r \sim \log^{\beta} n \log^{\gamma} \varepsilon^{-1}$, таким образом, об оценках вида $\mathcal{O}(nr^d)$ мы будем говорить, как о *почти линейных* по n.

2. Крестовый метод

2.1. Крестовый метод для двумерных массивов. Метод быстрого умножения, основанный на приближении матрицы матрицей малого ранга, был впервые предложен в [21]. В работах [8, 10, 13] эта задача аппроксимации была связана с отысканием в матрице A столбцов и строк, на пересечении которых располагается подматрица *максимального объема* (максимального 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 [22]. В работах [23, 24] в качестве такого шаблона выбираются строка и столбец. Именно такой выбор мы положим в основу нашего трехмерного метода крестовой аппроксимации, поэтому для полноты изложения повторим основное описание этого метода здесь.

Алгоритм 1 (Cross 2D). 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 какой-то столбец, скажем, первый. Устанавливается $j_1 = 1$.

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 всех предыдущих скелетонов. Затем находится наибольший по модулю элемент в строке, причем элемент из столбца j_p вновь 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

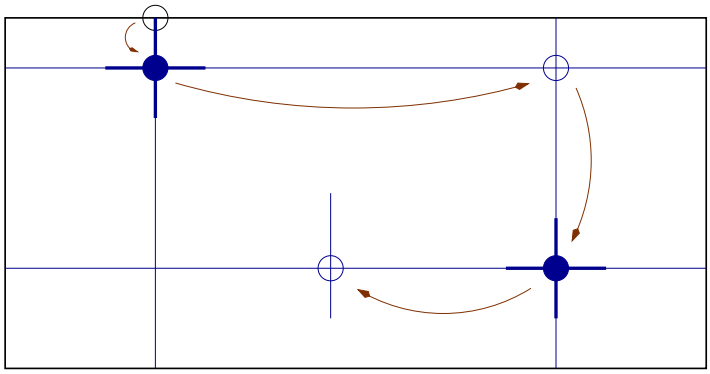


Рис. 1. Схема работы крестового метода

$\tilde{A}_p$ 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

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

считается достаточно точной, если выполняется критерий

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

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 недопустимо. Впрочем, норму $\|\tilde{A}_p\|_F$ мы, конечно, можем найти быстрее. В самом деле, т.к.

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

то для вычисления F-нормы матрицы ранга p нам необходимо подсчитать p^2 скалярных произведений, то есть затратить $\mathcal{O}(np^2)$ опе-

раций. Кроме того, если известна норма $\|\tilde{A}_{p-1}\|_F$, то после вычисления скелетона $u_p v_p^T$ для вычисления $\|\tilde{A}_p\|_F$ необходимо дополнительно найти только p скалярных произведений векторов u_p и v_p на ранее полученные столбцы и строки, то есть совершить $\mathcal{O}(pn)$ операций. Таким образом, мы можем быстро пересчитывать норму аппроксиманта $\tilde{A}_p$ на каждом шаге алгоритма.

Для оценки погрешности $\|A - \tilde{A}_p\|_F$ можно обратить внимание на следующий факт: алгоритмически крестовый метод эквивалентен методу Гаусса с выбором ведущего элемента по строке и столбцу. Как известно (см., например, [25]), если метод Гаусса применен к матрице ранга r, на шаге $r+1$ ведущий элемент будет равен нулю, если матрица отличается от матрицы ранга r на величину ε, на шаге $r+1$ возникнет ведущий элемент порядка ε. Пользуясь этой аналогией, мы можем следить за мерой убывания норм вычисляемых векторов u_p и v_p и считать аппроксимацию достаточно точной, если произведение их норм стало достаточно малой величиной, к примеру согласно критерию

$$(n-p)\|c_p\|_2\|v_p\|_2 < \varepsilon\|\tilde{A}_r\|_F.$$

Предложенная версия крестового алгоритма требует $2rn$ вычислений элементов матрицы A и $\mathcal{O}(r^2 n)$ дополнительных операций, связанных главным образом с вычитанием из получаемых элементов a_{ij} значений уже насчитанных p скелетонов в позициях i, j.

2.2. Как нельзя построить трехмерный крестовый метод. 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, к сожалению, 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 за-

дач. 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.

Несколько запоздало определим *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

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

2.3. Как можно построить трехмерный крестовый метод. Рассмотрим развертки массива $\mathcal{A}$, определенные формулой (4), 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 каждой из разверток $A^{(1)}$, $A^{(2)}$, $A^{(3)}$ существует аппроксимация ранга r, то есть матрицы вида

$$\tilde{A}_r^{(1)} = U\Psi_1^T \qquad \tilde{A}_r^{(2)} = V\Psi_2^T \qquad \tilde{A}_r^{(3)} = W\Psi_3^T,$$

где U, V, W имеют размер $n \times r$, а матрицы Ψ_1, Ψ_2, Ψ_3 имеют размер $n^2 \times r$. Базисы, заданные «короткими» компонентами разложения, то есть матрицами U, V, W, задают факторы разложения Таккера (надо всего лишь найти ортогональные базисы $\tilde{U}$, $\tilde{V}$, $\tilde{W}$ тех же подпространств). Ядро разложения Таккера вычисляется с помощью свертки вида (6), где вместо точных значений a_{ijk} используются их

приближенные значения, представленные с помощью любого из полученных крестовых разложений. Например, пользуясь разложением для разверки по первому направлению, $\tilde{A}_r^{(1)} = U\Psi_1^T$, имеем

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

Подставив это выражение в формулу свертки (6), получаем

$$\begin{aligned} g_{i'j'k'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r} u_{i\alpha}\psi^1_{jk\alpha}\right)\tilde{u}_{ii'}\tilde{v}_{jj'}\tilde{w}_{kk'} = \\ &= \sum_{\alpha=1}^{r}(u_\alpha, \tilde{u}_{i'})\left(\sum_{j=1}^{n}\sum_{k=1}^{n}\tilde{v}_{jj'}\tilde{w}_{kk'}\psi^1_{jk\alpha}\right). \end{aligned} \tag{7}$$

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. Вычисление ортогональных матриц $\tilde{U}$, $\tilde{V}$ и $\tilde{W}$ имеет линейную по n сложность. Вычисление скалярных произведений $(u_\alpha, u_{i'})$ в (7) требует $\mathcal{O}(nr^2)$ операций, вычисление сумм

$$\sum_{j=1}^{n}\sum_{k=1}^{n}\tilde{v}_{jj'}\tilde{w}_{kk'}\psi^1_{jk\alpha} = \left(\tilde{V}^T\Psi_{1\alpha}\tilde{W}\right)_{j'k'}$$

сводится к вычислению r произведений вида $\tilde{V}^T\Psi_{1\alpha}\tilde{W}$, с набором из r матриц $\Psi_{1\alpha} = [(\psi^1_\alpha)_{jk}]$, имеющих размер $n \times n$ и упорядоченных по индексу α. Вычисление таких произведений выполняется за $\mathcal{O}(n^2 r^2)$ операций. Таким образом, общая сложность метода порядка $\mathcal{O}(n^2 r^2)$.

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.

2.4. 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 (4). Вспомнив, что эти строки есть не что иное, как срезки

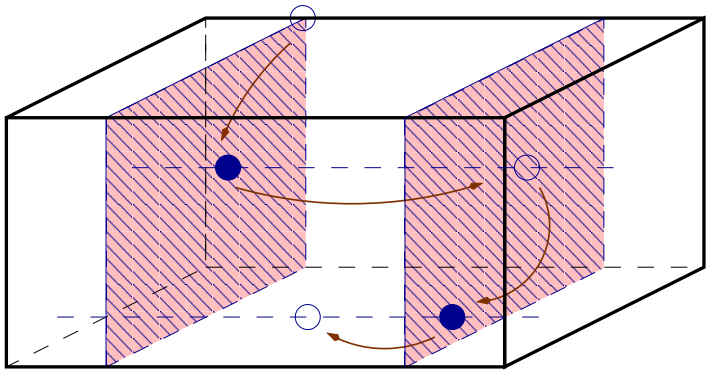


Рис. 2. Схема работы трехмерного крестового метода

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 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 (1), 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 вычислении же строк матрицы, то есть «длинных» векторов, мы вновь применяем крестовый метод, полу-

чая их аппроксимации за почти линейное по n число вычислений.

0 Пусть p — номер текущего шага. На первом шаге ($p = 1$) в массиве $\mathcal{A}$ выбирается какая-то срезка $A_k = [(a_k)_{ij}]$, скажем, первая. Устанавливается $k_1 = 1$. 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

$$\tilde{A}_{k_p} = \sum_{q=1}^{r} u_{pq} v_{pq}^T.$$

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

2 Вычисляется «короткая» 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:

$$w_p := w_p / w_{k_p p}.$$

3 К уже вычисленной аппроксимации добавляется r триплетов вида

$$\tilde{\mathcal{A}} = \tilde{\mathcal{A}} + \tilde{A}_{k_p} \otimes w_p = \tilde{\mathcal{A}} + \left(\sum_{q=1}^{r} u_{pq} v_{pq}^T \right) \otimes w_p = \tilde{\mathcal{A}} + \sum_{q=1}^{r} u_{pq} \otimes v_{pq} \otimes w_p.$$

Теперь значения получившейся аппроксимации $\tilde{A}$ совпадают 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

$$\tilde{a}_{ijk} = \sum_{p=1}^{r} \left(\sum_{q=1}^{r} u_{ipq} v_{jpq} \right) w_{kp} = \sum_{\alpha=1}^{r^2} u_{i\alpha} v_{j\alpha} w_{k\alpha}, \tag{8}$$

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 3.1. Оценочно его сложность составляет $\mathcal{O}(cnr)$ операций, где c — число внутренних итераций метода. Хотя точных теоретических обоснований его достоверности и оценки числа внутренних итераций мы не имеем, на всех рассмотренных нами примерах он демонстрирует быструю сходимость (количество итераций не превосходит величины порядка r^2) 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

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

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 малости 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 (6), где элементы массива $\mathcal{A}$ представлены с помощью одного из имеющихся трилинейных разложений, выражающих a_{ijk} в виде (8). В результате имеем

$$\begin{aligned} g_{i'j'k'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\sum_{\alpha=1}^{r^2} u_{i\alpha}v_{j\alpha}w_{k\alpha}\right)\tilde{u}_{ii'}\tilde{v}_{jj'}\tilde{w}_{kk'} = \\ &= \sum_{\alpha=1}^{r^2}(u_\alpha, \tilde{u}_{i'})(v_\alpha, \tilde{v}_{j'})(w_\alpha, \tilde{w}_{k'}). \end{aligned} \tag{9}$$

Для вычисления использованных в формуле (9) 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.

2.5. Как мы строим трехмерный крестовый метод. Алгоритм 2 дает первичное представление о том, каким образом разложение трехмерного массива может быть вычислено за почти линейное по размеру массива число операций. Однако возникающий в оценках сложности множитель r^4 на практике может существенно замедлять работу метода. Для того чтобы метод стал по-настоящему эффективным, необходимо предложить еще более экономный формат хранения полученных срезок A_{k_p}, который позволит снизить количество используемых векторов с $\mathcal{O}(r^2)$ до величины $\mathcal{O}(r)$ и уменьшить каким-то образом количество действий, необходимых для вычитания из полученных элементов значений уже насчитанного аппроксиманта. Необходимо также, чтобы новый формат давал возможность быстрее оценивать F–норму массива $\mathcal{A}$ и отклонение аппроксиманта от точного значения.

Наше предложение состоит в том, чтобы аппроксимировать насчитанные срезки A_{k_p} в виде

$$A_{k_p} = UB_pV^T, \tag{10}$$

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 (10) 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 (10).

Алгоритм 3 (Cross 3D). Пусть задан трехмерный массив $\mathcal{A}$ размера $n \times n \times n$ и требуется получить его аппроксимацию за число операций, порядка nr^3. Как и прежде, один из индексов i, j, k назовем направляющим; в приводимом ниже алгоритме таковым будет индекс k.

0 Провести один цикл работы алгоритма 2, то есть шаг с $p = 1$. После него $p = 2$ и массив $\mathcal{A}$ аппроксимирован в виде

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

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

$$U_1 = UR_u, \qquad V_1 = VR_v.$$

Тогда $\mathcal{A}$ представляется в виде

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

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

Далее положим $w_1 := w_1 / \|w_1\|_F$, то есть будем нормировать все вычисляемые векторы u_p, v_p, w_p. В векторе w_1 вычислим максимальный элемент, пусть он имеет индекс k_2.

1.1 С помощью крестового метода срезка A_{k_p} вычисляется в виде малоранговой аппроксимации

$$A_{k_p} \approx \tilde{A}_{k_p} = \sum_{q=1}^{r} u_{pq} v_{pq}^T,$$

причем из вычисляемых элементов массива $\mathcal{A}$ **не** вычитаются значения предыдущей аппроксимации.

1.2 Текущие опорные матрицы U, V расширяются добавлением векторов u_{pq}, v_{pq}, $q = 1, \ldots, r$, и полученные новые наборы $[UU_p]$, $[VV_p]$ ортогонализуются

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

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

1.3 Вычисляется новое ядро B_p размером $2r \times 2r$

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

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 $\mathcal{A}_{p-1}$ представлен в виде

$$\mathcal{A}_{p-1} = \left(\sum_{q=1}^{p-1} U' B'_q V'^T \right) \otimes w_\beta,$$

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

Положим также $B'_p = B_p$.

1.4 В полученной срезке A_{k_p} отыскивается максимальный (или почти максимальный) элемент. 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 $W = [w_1, \ldots, w_{p-1}]$

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

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

$$\tilde{\mathcal{A}}_p = \left(\sum_{q=1}^{p} U' B''_q V'^T \right) \otimes w_q. \tag{11}$$

Он совпадает с точными значениями массива $\mathcal{A}$ на позициях p вычисленных срезок и p вычисленных векторов–распорок. Размеры матриц U' и V' равны $n \times 2r$, размер ядер B''_q равен $2r \times 2r$. 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 $2r \times 2r \times p$ и строится его разложение Таккера (3) с помощью SVD-алгоритма, описанного на с. 69 формулами (4), (5), (6). Пусть факторы найденного разложения $U_\clubsuit, V_\clubsuit, W_\clubsuit$, ядро $G_\clubsuit$, то есть имеет место равенство

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

Обозначив

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

имеем

$$B''_k = U_\clubsuit B_{\clubsuit k} V_\clubsuit^T, \qquad k = 1, \ldots, p. \tag{12}$$

где ортогональные матрицы $U_\clubsuit$ и $V_\clubsuit$ имеют размер $2r \times r$, ядра $B_{\clubsuit k}$ имеют размер $r \times r$.

3.2 Подставив (12) в (11), получаем следующее представление для аппроксиманта:

$$\tilde{\mathcal{A}}_p = \left(\sum_{k=1}^{p}(U'U_{\clubsuit})B_{\clubsuit\,k}(V'V_{\clubsuit})^T\right)\otimes w_k = \left(\sum_{k=1}^{p}UB_kV^T\right)\otimes w_k, \tag{13}$$

где матрицы $U = U'U_{\clubsuit}$, $V = V'V_{\clubsuit}$ ортогональны, а ядра $B_k = B_{\clubsuit\,k}$ имеют размер $r \times r$. Таким образом, формат (10) хранения срезок аппроксиманта восстановлен.

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:

$$(n-p)\|B_{\clubsuit\,p}\|_F < \varepsilon\|\mathcal{B}_{\clubsuit}\|_F. \tag{14}$$

Некоторые дополнительные замечания о критерии остановки метода обсуждаются в пункте 3.4.

2.6. Сложность полученного метода. Обсудим сложность метода, описанного алгоритмом 3. Вступительный шаг **0** требует порядка $\mathcal{O}(nr^2)$ вычислений элемента массива $\mathcal{A}$ и $\mathcal{O}(nr^2)$ дополнительных операций при насчете 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 $\|\tilde{\mathcal{A}}\|_F = \|B_1\|_F$), не зависит от n, то есть является пренебрежимо малым.

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

1.1 Вычисление срезки A_{k_p} требует $\mathcal{O}(rn)$ вычислений элемента массива $\mathcal{A}$ и $\mathcal{O}(r^2n)$ дополнительных операций.

1.2 Ортогонализация расширенной матрицы требует $\mathcal{O}(r^2n)$ действий, методы по повышению эффективности ее реализации описаны в пункте 3.2.

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

1.4 Для того чтобы найти в новой срезке максимальный элемент, требуется $\mathcal{O}(crn)$ действий, однако для более быстрой реализации этого шага требуется предварительно «дожать» срезку, снизив число векторов в ее малоранговом разложении с $2r$ до r. Способ вычисления такого быстрого дожимания и замечания по его реализации предложены в пункте 3.3.

2.1 Чтобы вычислить распорку w_p, требуется насчитать $\mathcal{O}(n)$ элементов массива $\mathcal{A}$ и произвести $\mathcal{O}(prn)$ дополнительных действий для вычитания из него значений уже насчитанного 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, полностью же сложность алгоритма составляет $\mathcal{O}(r^2 n)$ вычислений элемента $\mathcal{A}$ и $\mathcal{O}(r^3 n)$ дополнительных действий.

3. Важные детали

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

3.1. 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 (малорангового) разложения

$$A = UV^T,$$

где U, V — матрицы $n \times r$. Эту задачу, конечно, легко можно решить простым сравнением всех n^2 элементов матрицы A, затратив $\mathcal{O}(rn^2)$ действий. Однако мы заинтересованы найти алгоритм, сложность которого является почти линейной по n. Таким образом, вычислять и сравнивать все элементы матрицы A мы не можем, и требуется каким-то образом указать *подмножество элементов* матрицы A, число которых мало по сравнению с n^2 и среди которых обязательно находится максимальный по модулю элемент A или достаточно близкий к нему. Предлагаемый алгоритм базируется на следующей гипотезе.

Гипотеза. Рассмотрим подматрицы размера $r \times r$ в матрице A ранга r. Пусть B — подматрица максимального объема среди всех таких подматриц, тогда

$$v_1(B) \geqslant v_1(A)/r.$$

где $v_1(A)$ — максимальное значение модуля элемента матрицы A.

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

Подматрица $r \times r$ максимального объема в A лежит на пересечении строк $i_1, \ldots, i_r$, совпадающих со строками, на которых расположена подматрица максимального объема в U, и столбцов $j_1, \ldots, j_r$, совпадающих со столбцами, на которых расположена подматрица максимального объема в V^T. Для отыскания в матрице $n \times r$ подматрицы максимального объема размера $r \times r$ применим алгоритм, предложенный в работе [26]. Для полноты изложения опишем его здесь.

Алгоритм 4. Пусть матрица U имеет размер $n \times r$. Требуется найти ее подматрицу размера $r \times r$, объем которой максимален среди всех подматриц такого размера.

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

1 Вычислим

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

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_γ новую подматрицу в первых r строках матрицы A и вернемся на шаг **1.**

иначе завершим алгоритм.

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

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

$$A_\gamma := A_\gamma + uv^T,$$

$$u_i = e_{ij_*}, \quad v_j = a_{i^*j} - e_{j^*j}, \qquad i,j = 1,\ldots,r, \tag{15}$$

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

Обратная матрица в этом случае тоже модифицируется преобразованием ранга 1 (формула Шермана-Вудбери-Моррисона):

$$\mathcal{A}_\gamma^{-1} := A_\gamma^{-1} - A_\gamma^{-1}u\left(1 + v^T A_\gamma^{-1}u\right)^{-1} v^T A_\gamma^{-1},$$

таким образом, и подматрица B модифицируется формулой

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

С учетом вида u, v (15) 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 1, которое выполняется за nr операций.

Отыскав с помощью предложенного алгоритма подматрицы наибольшего объема в факторах малорангового разложения U и V, мы тем самым определили строки и столбцы матрицы A, в которых стоит подматрица максимального объема размером $r \times r$. Вычисление всех элементов этой подматрицы и определение максимального среди них требует $\mathcal{O}(r^3)$ действий. По нашей гипотезе (подтвержденной практическими расчетами), этот элемент совпадает или весьма близок к максимальному элементу всей матрицы, то есть может быть использован в качестве хорошего ведущего элемента для крестового метода.

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

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

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

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

Таким образом, 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 исходный вектор u_p в значительной степени лежит в пространстве U,

к которому требуется его ортогонализовать, 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 [27].

Для наших целей мы используем блочную версию этого алгоритма.

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

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

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

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

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 $[UU_p]$.

3.3. Как уменьшить ранг аппроксимации у срезки. При использовании алгоритма 2 на шаге **1b** и алгоритма 3 на шаге **1.4** требуется отыскать максимальный элемент в срезке A, заданной в виде малоранговой аппроксимации

$$A = UV^T.$$

Процедура отыскания максимального элемента матрицы A, описанная в пункте 3.1, находит подматрицы наибольшего объема (или близкие к таковым) в факторах U и V размера $n \times r$. Сложность одной итерации этой процедуры зависит от размера и ранга этих матриц, причем число итераций этого метода тоже растет с увеличением

r, поэтому при больших значениях r поиск подматрицы максимального объема может существенно замедляться. Однако на практике может оказаться так, что размер матриц U, V^T может быть сильно завышен по сравнению с реальным рангом матрицы A. Причиной тому могут служить:

- чрезмерно жесткий критерий остановки метода крестовой аппроксимации (алгоритм 1),
- слишком жесткий выбор критерия ε для алгоритма 3, особенно в случае, когда сами элементы a_{ijk} вычисляются с ошибкой, вносящей шум, способный значительно увеличить ранг вычисляемых срезок A_{k_p},

и тому подобные факторы.

Однако мы можем значительно снизить ранг скелетонного разложения у рассматриваемой матрицы A, внеся в ее элементы дополнительную небольшую ошибку, которая, однако, не повлияет на результат решения задачи об отыскании максимального (или близкого к нему) элемента — относительное изменение в нем будет минимальным среди всех элементов. Процедуры снижения ранга у матрицы, полученной в ходе работы крестового метода, называемые методами «дожимания» матрицы, рассматривались в [23], здесь приводится одна из них.

Алгоритм 5 («дожимание»). Итак, пусть существует некая аппроксимация

$$A = UV^T,$$

факторы U, V имеют размер $n \times r$, и требуется построить аналогичную аппроксимацию со сниженным значением r, внеся небольшую легко контролируемую ошибку в матрицу A.

1 Ортогонализуем их столбцы и строки, например, с помощью процедуры QR-ортогонализации.

$$U = Q_u R_u, \qquad V = Q_v R_v,$$

Тогда

$$A = Q_u R_u R_v^T Q_v^T.$$

2 Образуем матрицу $B = R_u R_v^T$.

3 Построим ее сингулярное разложение $U_b \Sigma V_b^T =: B$. 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

$$A = U_a \Sigma V_a^T, \qquad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r).$$

Видим, что полученное разложение есть с точностью до нулевых сингулярных чисел SVD–разложение матрицы A, вычисленное без явного ее формирования. Значит, с этим разложением можно работать как с разложением матрицы A.

3 Если погрешность в аппроксимации составляет ε, а мы допускаем погрешность $\tilde{\varepsilon} > \varepsilon$, то для снижения ранга аппроксимации надо отбросить младшие сингулярные числа, руководствуясь условием

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

где $\tilde{r}$ — новое, уменьшенное число скелетонов, аппроксимирующих A с точностью ε.

4 Вычислим $\tilde{r}$ столбцов матриц U_a, V_a, отвечающих старшим $\tilde{r}$ сингулярным числам $\sigma_1, \ldots, \sigma_{\tilde{r}}$. Новое малоранговое разложение задается формулой

$$\tilde{A} = \tilde{U}_a \tilde{\Sigma} \tilde{V}_a,$$

где $\tilde{U}_a, \tilde{V}_a$ — матрицы размера $n \times \tilde{r}$, содержащие первые вычисленные старшие столбцы U_a, V_a, а диагональная матрица $\tilde{\Sigma}$ содержит старшие сингулярные числа $\sigma_1, \ldots, \sigma_{\tilde{r}}$.

Сложность метода $\mathcal{O}(r^2 n)$ — основные затраты вновь идут на ортогонализацию. Отметим, что при отыскании максимального элемента в срезке, уже представленной, как в пункте **1.4** алгоритма 3, в виде $A = UBV^T$, где U, V уже ортогональны, остается только найти сингулярное разложение для B, что делает применение дожимателя еще более дешевым.

Практические расчеты показывают, что предварительное использование «дожимателя» с мягким критерием $\tilde{\varepsilon}$ способно значительно снизить ранг аппроксимации и тем самым ускорить процедуру отыскания максимального элемента в срезке A (так, что затраты на само дожимание оказываются несущественными).

3.4. Как проверять точность аппроксимации. Как бы ни был хорош критерий остановки (14), 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 $\mathrm{p} = (i_p, j_p, k_p),\ p = 1, \ldots, n,$ выполняется

$$\|a_{\mathrm{p}} - \tilde{a}_{\mathrm{p}}\|_2 \leqslant \|a_{\mathrm{p}}\|_2,$$

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.

4. 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

$$\begin{aligned}
\mathcal{A} &\approx \tilde{\mathcal{A}}^{\{1\}} = \sum_{k=1}^{r_1} \left(V^{\{1\}} B_k^{\{1\}} W^{\{1\}T} \right) \otimes u_k^{\{1\}}, \\
\mathcal{A} &\approx \tilde{\mathcal{A}}^{\{2\}} = \sum_{k=1}^{r_2} \left(W^{\{2\}} B_k^{\{2\}} U^{\{2\}T} \right) \otimes v_k^{\{2\}}, \\
\mathcal{A} &\approx \tilde{\mathcal{A}}^{\{3\}} = \sum_{k=1}^{r_3} \left(U^{\{3\}} B_k^{\{3\}} V^{\{3\}T} \right) \otimes w_k^{\{3\}}.
\end{aligned} \tag{16}$$

При реальном 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 и от размера «опорных» матриц.

2. Ортогональные столбцы $u^{\{1\}}$, $v^{\{2\}}$, $w^{\{3\}}$ использовались в качестве факторов U, V, W разложения Таккера (3). Ядро разложения вычислялось с помощью свертки (6), где вместо массива $\mathcal{A}$ использовалась одна из полученных аппроксимаций, что в данном случае приводит к выражению для элемента ядра

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

Таким образом, построено 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 ядра могут оказаться несколько большими, чем это в самом деле необходимо. Чтобы снизить размеры ядра, применим к нему метод редукции Таккера.

3. К ядру $\mathcal{G}$ применяется метод редукции Таккера (4)-(6), снижающий размерность ядра до $r \times r \times r$. Этот размер и приведен в таблицах.

Табл. 1 демонстрирует значения рангов, точности и размера полученной аппроксимации для массива $\mathcal{A}$, табл. 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. 3. 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, конечно же, позволяет надеяться,

Таблица 1

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

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	2.5$_{10}$−4	8	2.3$_{10}$−6	10	1.4$_{10}$−8	12	3.1$_{10}$−10
128	6	6.8$_{10}$−4	8	4.4$_{10}$−6	11	3.1$_{10}$−8	13	6.4$_{10}$−10
256	6	8.8$_{10}$−4	9	3.9$_{10}$−6	12	5.4$_{10}$−8	15	3.0$_{10}$−10
512	7	7.4$_{10}$−4	10	1.4$_{10}$−6	13	6.8$_{10}$−8	16	5.2$_{10}$−10
1024	7	5.7$_{10}$−4	11	4.8$_{10}$−6	14	4.0$_{10}$−8	18	2.7$_{10}$−10
2048	7	6.6$_{10}$−4	12	2.0$_{10}$−6	16	4.0$_{10}$−8	19	4.0$_{10}$−10
4096	8	3.2$_{10}$−4	12	6.3$_{10}$−6	17	3.4$_{10}$−8	21	3.5$_{10}$−10
8192	8	6.3$_{10}$−4	13	3.3$_{10}$−6	18	1.9$_{10}$−8	22	4.5$_{10}$−10
16384	9	7.9$_{10}$−4	14	3.5$_{10}$−6	19	7.2$_{10}$−8	24	5.6$_{10}$−10
32768	9	6.4$_{10}$−4	14	8.8$_{10}$−6	20	5.2$_{10}$−8	25	3.8$_{10}$−10
65536	9	4.0$_{10}$−4	15	6.3$_{10}$−6	21	2.5$_{10}$−8	26	5.2$_{10}$−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

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

Ранг и точность построенного разложения Таккера

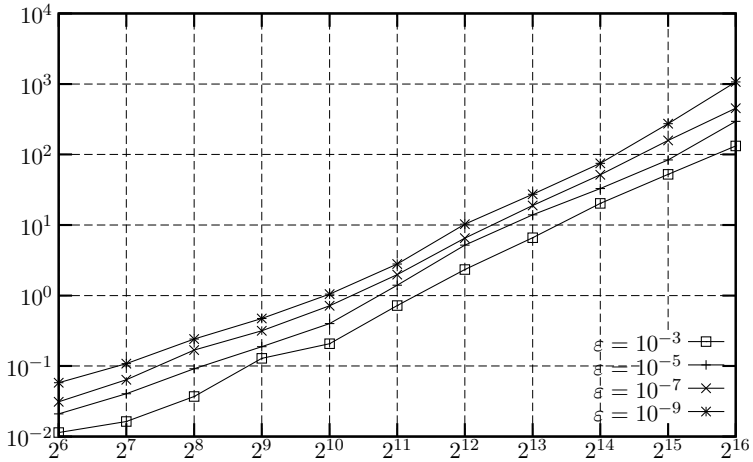
ε	$1._{10}{-}3$		$1._{10}{-}5$		$1._{10}{-}7$		$1._{10}{-}9$	
n	r	err	r	err	r	err	r	err
64	7	$3.7_{10}{-}4$	11	$3.9_{10}{-}6$	14	$5.7_{10}{-}8$	18	$2.2_{10}{-}10$
128	8	$5.1_{10}{-}4$	12	$5.9_{10}{-}6$	17	$2.0_{10}{-}8$	20	$5.6_{10}{-}10$
256	9	$4.1_{10}{-}4$	14	$6.4_{10}{-}6$	19	$3.4_{10}{-}8$	23	$4.5_{10}{-}10$
512	10	$4.9_{10}{-}4$	15	$6.7_{10}{-}6$	21	$2.9_{10}{-}8$	26	$3.2_{10}{-}10$
1024	10	$5.5_{10}{-}4$	17	$3.2_{10}{-}6$	23	$3.9_{10}{-}8$	29	$4.7_{10}{-}10$
2048	11	$5.0_{10}{-}4$	18	$5.2_{10}{-}6$	25	$6.8_{10}{-}8$	31	$5.9_{10}{-}10$
4096	12	$8.4_{10}{-}4$	19	$4.2_{10}{-}6$	27	$3.5_{10}{-}8$	34	$3.3_{10}{-}10$
8192	12	$6.8_{10}{-}4$	20	$6.0_{10}{-}6$	28	$5.8_{10}{-}8$	36	$3.6_{10}{-}10$
16384	13	$2.7_{10}{-}4$	22	$4.8_{10}{-}6$	31	$5.6_{10}{-}8$	39	$2.6_{10}{-}10$
32768	13	$8.5_{10}{-}4$	23	$6.1_{10}{-}6$	32	$7.1_{10}{-}8$	41	$5.5_{10}{-}10$
65536	14	$6.2_{10}{-}4$	24	$6.5_{10}{-}6$	34	$7.8_{10}{-}8$	44	$1.4_{10}{-}9$

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

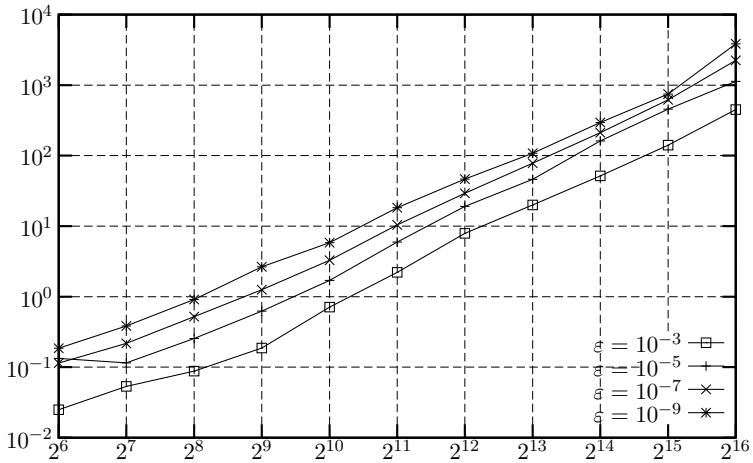
ε		$1._{10}{-}3$		$1._{10}{-}5$		$1._{10}{-}7$		$1._{10}{-}9$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2Mb	7		11		14		18	
128	16Mb	8		12		17		20	
256	128Mb	9		14		19		23	
512	1Gb	10		15		21		26	
1024	8Gb	10		17		23		29	
2048	64Gb	11	<1	18	<1	25	1.18	31	1.46
4096	512Gb	12	1.15	19	1.78	27	2.54	34	3.2
8192	4Tb	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. 3. Время на построение аппроксимации, с.

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



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



что предложенный алгоритм может быть очень полезен для решения трехмерных задач, сводимых дискретизацией на тензорных сетках к линейным системам с плотными матрицами больших и сверхбольших размеров.

Список литературы

[1] 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.

[2] 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.

[3] Rokhlin V. Rapid solution of integral equations of classical potential theory // *J. Comput. Physics.* 1985. V. 60. P. 187-207.

[4] Rokhlin V. Rapid solution of integral equations of scattering theory in two dimensions // *J. Comput. Physics.* 1990. V. 86. P. 414-439

[5] Тыртышников Е. Е. Методы быстрого умножения и решение уравнений // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 4-41.

[6] Горейнов С. А. Мозаично-скелетонные аппроксимации матриц, порожденных асимптотически гладкими и осцилляционными ядрами // *Матричные методы и вычисления* — М.: ИВМ РАН, 1999. С. 42-76.

[7] Sun X., Pitsianis N. P. A matrix version of the fast multipole methos // *SIAM Review.* 2001. V. 43, No. 2. P. 289-300.

[8] Tyrtyshnikov E. E. Mosaic–skeleton approximations // *Calcolo.* 1996. V. 33 (1-2). P. 47-57.

[9] Горейнов С. А., Замарашкин Н. Л., Тыртышников Е. Е. Псевдоскелетные аппроксимации матриц // *Доклады Российской академии наук.* 1995. V. 343 (2). P. 151-152.

[10] Goreinov S. A., Tyrtyshnikov E. E. The maximal-volume concept in approximation by low-rank matrices // *Contemporary Mathematics.* 2001. V. 208. P. 47-51.

[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] Tyrtyshnikov E. E. Incomplete cross approximation in the mosaic–skeleton method // *Computing.* 2000. V. 4. P. 367-380.

[14] Hackbush W., Khoromskij B. N., Tyrtyshnikov E. E. Hierarchical Kronecker tensor-product approximations. // *J. Numer. Math.* 2005. V. 13. P. 119-156.

[15] Tyrtyshnikov E. E. Kronecker-product approximations for some function-related matrices // *Linear Algebra Appl.* 2004. V. 379. P. 423-437.

[16] Тыртышников Е. Е. Тензорные аппроксимации матриц, порожденных асимптотически гладкими функциями // *Матем. сб.* 2003. Т. 194, 6. С. 147-160.

[17] Ibraghimov I. Application of the three-way decomposition for matrix compression // *Numer. Linear Algebra Appl.* 2002. V. 9, No. 6-7. P. 551-565.

[18] 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.

[19] Dennis J. E., Schabel R. B. Numerical methods for unconstrained optimization and nonlinear equations. — Plentice-Hall, Englewood Clis, 1983.

[20] Tucker L. R. Some mathematical notes on three-mode factor analysis. // *Psychometrika.* 1966. V. 31. P. 279-311.

[21] Tyrtyshnikov E. E. Matrix approximations and cost-effective matrix-vector multiplication — М.: ИВМ РАН, 1993.

[22] 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

[23] Савостьянов Д. В. Мозаично–скелетонные аппроксимации. Выпускная квалификационная работа на степень бакалавра — М.: ИВМ РАН, 2001.

[24] Bebendorf M. Approximation of boundary element matrices // *Numer. Math.* 2000. V. 86, No. 4. P. 565-589.

[25] Голуб Дж., Ван Лоун Ч. Матричные вычисления / Пер. с. англ. — М.: Мир, 1999.

[26] 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.

[27] Парлетт Б. Симметричная проблема собственных значений. Численные методы / Пер. с англ. — М.: Мир, 1983.

[28] Оселедец И. В., Савостьянов Д. В. Методы разложения тензора // *Настоящий сборник.* С. 51-64.

[29] Оселедец И. В., Савостьянов Д. В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Настоящий сборник.* С. 101-116.

[30] Оселедец И. В., Савостьянов Д. В. Об одном алгоритме построения трилинейного разложения // *Настоящий сборник.* С. 117-130.

[31] Оселедец И. В., Савостьянов Д. В. Минимизационные методы аппроксимации тензоров и их сравнение // *Настоящий сборник.* С. 131-146.

Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров[a]

И. В. Оселедец, Д. В. Савостьянов

Рассматривается проблема одновременного приведения семейства матриц с помощью ортогональных преобразований к треугольному виду. Показано, что такое приведение может быть осуществлено с помощью дефляции. На каждом шаге дефляции необходимо решать супер-обобщённую задачу на собственные значения, которая является прямым обобщением обобщённой задачи на собственные значения на случай нескольких матриц. Для решения этой задачи построен быстрый вариант метода Гаусса-Ньютона и исследованы его свойства сходимости. Эффективность предложенного метода проиллюстрирована некоторыми численными примерами.

1. Введение

В данной работе мы будем рассматривать следующую задачу *трилинейного разложения* трёхмерного массива (тензора):

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}, \tag{1}$$

где $\mathcal{A}$ — заданный тензор с размерами $n_1 \times n_2 \times n_3$, число r называется *тензорным рангом,* а подлежащие определению матрицы $U = [u_{i\alpha}], V = [v_{j\alpha}], W = [w_{k\alpha}]$ — *факторами* разложения (1).

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

Нас будет интересовать случай, когда один из размеров тензора равен его рангу. Также мы будем предполагать, что массив имеет размеры $n \times n \times q$ и $r = n$. Этот случай хоть и является частным, однако довольно часто встречается в приложениях. Более того, алгоритмы разложения тензора при таких ограничениях на размер и ранг могут служить промежуточным шагом для других методов, в которых это ограничение ослабляется [7].

Можно показать, что задачу (1) можно решать, используя суперобобщённое разложение Шура матриц-срезок A_k массива $\mathcal{A}$ по индексу k:

$$(A_k)_{ij} = \mathcal{A}_{ijk}.$$

Суперобобщённым разложением Шура называется следующая задача:

Для данных q матриц $A_1, \ldots, A_q$ размера $n \times n$ найти ортогональные матрицы Q и Z такие, что

$$B_k = QA_kZ$$

максимально (насколько это возможно) верхнетреугольны.

Если мы умеем находить суперобобщённое разложение Шура для матриц $A_1, \ldots, A_q$, то мы умеем находить трилинейное разложение для тензора $\mathcal{A}$ (например, см. [5, 6]). Поэтому теперь мы целиком сосредоточимся на создании метода для построения суперобобщённого разложения Шура.

Задача одновременного приведения семейства матриц к заданному специальному виду (треугольному, как в нашем случае, или, например, диагональному) с помощью одинаковых преобразований (в нашем случае это умножение каждой матрицы на ортогональные, в качестве примера можно упомянуть подобие или унитарное подобие с одной и той же матрицей перехода, преобразования эквивалентности) является одной из самых интересных проблем матричного анализа [2, 3, 5], и развитие эффективных алгоритмов для таких задач очень важно.

Название «суперобобщённое разложение Шура» объясняется тем, что для случая $q = 2$ (две матрицы) мы получаем просто обобщённое разложение Шура. В вещественном случае для того чтобы две матрицы можно было привести к треугольному виду с помощью умножения на ортогональные матрицы, необходимо, чтобы собственные значения соответствующего матричного пучка были вещественны.

Очевидно, что для *произвольных* матриц A_k приведение именно к треугольному виду с приемлемой точностью осуществить нельзя, поэтому мы всегда будем предполагать существование таких ортогональных матриц Q и Z, что

$$QA_kZ = T_k + E_k, \tag{2}$$

где T_k верхнетреугольные и «остатки» E_k удовлетворяют некоторому «условию малости»:

$$\left(\sum_{k=1}^{r}(||E_k||_F)^2\right)^{1/2} = \varepsilon.$$

Если же матрицы A_k являются срезками массива $\mathcal{A}$, то (2) гарантируется существованием у $\mathcal{A}$ хорошего приближения тензорного ранга n. Осталось лишь его найти!

Алгоритмы для одновременного приведения семейства матриц к какому-либо виду (в нашем случае треугольному) часто получаются как обобщение соответствующих алгоритмов для матрицы или пары матриц. Отметим два подхода к решению проблемы одновременного приведения матриц. Первый, очень общий подход, предложен в [1]. В двух словах, идея состоит в использовании обыкновенного дифференциального уравнения для матриц

$$B_k(t) = Q(t)A_kZ(t),$$

являющегося непрерывным аналогом метода Ньютона, причём в правой части вместо градиента используется проекция градиента, что обеспечивает сохранение матрицами B_k необходимого вида («изоспектральный поток»). При $t \to \infty$ мы получаем требуемые треугольные матрицы. Однако эффективная численная реализация этого метода требует интегрирования большой и часто очень жёсткой системы обыкновенных дифференциальных уравнений на большом промежутке времени.

Другой подход состоит в использовании вращений Якоби. На каждом шаге уже насчитанные матрицы Q и Z умножаются на матрицы вращений, а сами матрицы вращений Q_i, Z_i выбираются так, чтобы минимизировать целевую функцию. В нашем примере

$$\sum_{k=1}^{q}||Q_i(QA_kZ)Z_i||_{LF}^2,$$

где $\|\cdot\|_{LF}^2$ — сумма квадратов всех элементов матрицы, находящихся ниже главной диагонали. Такой алгоритм был предложен в [5], и там было показано, что на каждом шаге необходимо находить корень полинома восьмой степени.

Самым эффективным методом вычисления обобщённого разложения Шура (когда $q = 2$) является QZ-алгоритм, поэтому естественно построить его аналог для случая $q > 2$, что было сделано в работе [4]. Однако следует подчеркнуть, что эффективная реализация QZ-алгоритма основана на использовании сдвигов и совершенно не ясно, как обобщить сдвиги на случай $q > 2$.

В данной работе мы предлагаем новый алгоритм для вычисления суперобобщённого разложения Шура. Мы покажем, что это разложение может быть вычислено с помощью последовательных шагов *дефляции*. На каждом шаге решается оптимизационная задача с $2n+q$ неизвестным, которая является прямым обобщением обобщённой задачи на собственные значения, поэтому мы будем называть её *суперобобщённой задачей на собственные значения*. Для нахождения решения построен быстрый вариант метода Гаусса-Ньютона и показано, что матрицы, получающиеся на каждом итерационном шаге, могут быть вычислены с помощью «дешёвых» рекуррентных формул. Мы также исследуем локальную скорость сходимости нашего метода. Численные эксперименты подтверждают эффективность метода. Например, вычисление суперобобщённого разложения Шура 128 матриц с размерами 128×128 занимает меньше минуты. Также на тестовых примерах метод показал свою хорошую устойчивость: при добавлении случайного шума к матрицам A_q невязка построенного разложения оказывалась порядка уровня этого шума.

2. Суперобобщённая задача на собственные значения

2.1. Переформулировка исходной задачи. Предположим, нам даны матрицы $A_1, \ldots, A_q$ размера $n \times n$ и нужно найти ортогональные матрицы Q и Z такие, что матрицы $QA_1Z, \ldots, QA_rZ$ максимально возможно верхнетреугольные. Естественно производить вычисление Q и Z следующим образом. Найдём сначала ортогональные матрицы Q и Z такие, что матрицы QA_iZ перейдут в блочно-треугольные матрицы следующего вида:

$$QA_kZ \approx \begin{pmatrix} \lambda_k & v_k^T \\ 0 & B_k \end{pmatrix}, \tag{3}$$

где B_k — матрицы размера $(n-1)\times(n-1)$. Если представление (3) найдено, то мы совершили один шаг *дефляции* и можно перейти к блочно-треугольному разложению матриц B_k. Совершив $n-1$ шагов, мы приведём матрицы A_k к треугольному виду. Потому теперь нашей основной целью становится решение задачи (3).

Уравнения (3) эквиваленты следующим:

$$QA_kZe_1 \approx \lambda_k e_1,$$

где e_1 — первый столбец единичной матрицы. Так как Q является ортогональной матрицей, мы можем умножить эти уравнения слева на Q^T:

$$A_kZe_1 \approx \lambda_k Q^T e_1.$$

Введём векторы $x = Ze_1$ и $y = Q^Te_1$. Необходимо решить в смысле наименьших квадратов следующую переопределённую систему нелинейных уравнений:

$$A_kx = \lambda_k y. \tag{4}$$

Видно, что при $q = 2$ система (4) становится обобщённой задачей на собственные значения для пары матриц A_1, A_2, поэтому мы будем называть задачу (4) *суперобобщённой задачей на собственные значения*. Так как векторы x, y и вектор λ определены с точностью до умножения на константу, введём дополнительное нормировочное условие

$$||x||_2 = 1.$$

Как именно решать суперобобщённую задачу на собственные значения, будет рассказано в следующем разделе, а пока мы будем считать, что уже умеем её решать. Тогда алгоритм вычисления суперобобщённого разложения Шура записывается следующим образом.

Алгоритм 1.

Пусть даны q матриц $A_1, \ldots, A_q$ размера $n\times n$. Для нахождения суперобобщённого разложения Шура нужно сделать следующие шаги

1. Установить

$$m = n, \quad B_i = A_i, \qquad i = 1, \ldots, q, \quad Q = Z = I.$$

2. Если $m = 1$, остановиться.

3. Решить суперобобщенную задачу на собственные значения:

$$B_kx = \lambda_k y, \qquad k = 1, \ldots, q.$$

4. Найти матрицы Хаусхолдера Q_m, Z_m размера $m \times m$, такие что
$$x = \alpha_1 Q_m^T e_1, \qquad y = \alpha_2 Z_m e_1.$$

5. Вычислить матрицы C_k, являющиеся угловыми подматрицами размера $(m-1) \times (m-1)$ в матрицах $\widehat{B}_k$, определённых как
$$\widehat{B}_k = Q B_k Z = \begin{pmatrix} \alpha_k & v_k^T \\ \varepsilon_k & C_k \end{pmatrix}.$$

6. Установить
$$Q := \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Q_m \end{pmatrix} Q,$$
$$Z := Z \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Z_m \end{pmatrix}.$$

7. Установить $m := m - 1$, $B_k = C_k$ и перейти к шагу 2.

2.2. Метод Гаусса-Ньютона для суперобобщённой задачи на собственные значения. В этом параграфе мы построим алгоритм для решения (4). Запишем уравнения (4) поэлементно:
$$\sum_{j=1}^{n} A_{ij}^k x_j = \lambda_k y_i. \tag{5}$$

Введём матрицы a_j размера $q \times n$
$$(a_j)_{ki} = A_{ij}^k, \qquad k = 1, \ldots, q, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n,$$
и вектор $\lambda = (\lambda_1, \ldots, \lambda_q)$. Тогда уравнения (5) превращаются в
$$\sum_{j=1}^{n} x_j a_j = \lambda y^T. \tag{6}$$

Уравнения (6) можно рассматривать как переопределённую систему нелинейных уравнений. Мы предложим быстрый вариант метода Гаусса-Ньютона для решения этой системы и выпишем некоторые оценки сходимости.

Идея метода Гаусса-Ньютона состоит в том, чтобы линеаризовать систему уравнений, как в обычном методе Ньютона, и решить получившуюся переопределённую систему линейных уравнений.

Линеаризация уравнений (6) около некоторой точки (x, λ, y) приводит к следующей системе:

$$\sum_{j=1}^{n} \widehat{x}_j a_j = \triangle\lambda y^T + \lambda \triangle y^T + \lambda y^T. \tag{7}$$

Для краткости $x + \triangle x$ было обозначено через $\widehat{x}$ и $||\widehat{x}||_2 = 1$. На каждом шаге необходимо решать уравнения (7) в смысле наименьших квадратов. Как это сделать? Оказывается, что неизвестные $\triangle y$ и $\triangle\lambda_k$ можно исключить следующим образом. Найдём такую матрицу Хаусхолдера H размера $n \times n$, что

$$Hy = he_1,$$

и матрицу Хаусхолдера C размера $q \times q$, что

$$C\lambda = ce_1$$

(здесь через e_1 обозначен первый столбец единичной матрицы порядка q). Умножая (7) на C слева и на H^T справа, получаем

$$\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j = ce_1 \triangle\widehat{y}^T + h\triangle\widehat{\lambda} e_1^T + hce_1 e_1^T, \tag{8}$$

где $\widehat{a}_j = Ca_jH^T$, $\triangle\widehat{y} = H\triangle y$ и $\triangle\widehat{\lambda} = C\triangle\lambda$.

Заметим теперь, что уравнения (8) распадаются на две независимых подсистемы. Для нахождения $\widehat{x}$ нужно минимизировать

$$\left\| \sum_{j=1}^{n} b_j \widehat{x}_j \right\|_F^2, \qquad \|\widehat{x}\| = 1, \tag{9}$$

где матрицы b_j получаются из $\widehat{a}_j$ занулением элементов в первой строчке и первом столбце. Когда $\widehat{x}$ найден, $\triangle\widehat{y}$ и $\triangle\widehat{\lambda}$ могут быть определены из уравнений

$$\left(\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j \right)_{k1} = h\triangle\widehat{\lambda}_k, \qquad k = 2, \ldots, q,$$

$$\left(\sum_{j=1}^{n} \widehat{x}_j \widehat{a}_j \right)_{1i} = c\triangle\widehat{y}_i, \qquad i = 2, \ldots, n.$$

Однако этот способ требует явного формирования матриц Хаусхолдера и матриц $\widehat{a}_j$, а, как будет показано позднее, при вычислении $\widehat{x}$ этого можно избежать. Поэтому мы предлагаем другой подход для оценки новых λ и y по известному $\widehat{x}$. После того как найдено новое значение $\widehat{x}$, мы оцениваем y и λ с помощью одной итерации степенного метода:

$$\widetilde{\lambda} = by, \qquad \widetilde{y} = b^T\lambda, \tag{10}$$

где

$$b = \sum_{j=1}^{n} \widehat{x}_j a_j.$$

Задача (9) является на самом деле задачей нахождения минимального сингулярного значения матрицы

$$B = [\mathrm{vec}(b_1), \ldots, \mathrm{vec}(b_n)],$$

где оператор vec преобразует матрицу в вектор, развёртывая её столбец за столбцом.

Поэтому $\widehat{x}$ является нормированным собственным вектором, отвечающим минимальному собственному значению матрицы Грама $\Gamma = B^T B$:

$$\Gamma\widehat{x} = \gamma_{\min}\widehat{x}.$$

Элементы Γ задаются как скалярные произведения столбцов из B, поэтому

$$\Gamma_{sl} = (b_s, b_l)_F,$$

где $(\cdot,\cdot)_F$ — фробениусово (евклидово) скалярное произведение матриц.

Решение задачи (7) состоит из двух этапов:

1. Вычисление матрицы Γ.
2. Нахождение минимального собственного значения и соответствующего собственного вектора матрицы Γ.

Подсчитаем количество арифметических операций, необходимое для шага 1. Прямое вычисление требует $\mathcal{O}(n^2q + nq^2)$ (вычисление b_j) + $\mathcal{O}(n^2qn)$ (вычисление B^TB) арифметических операций. Полная стоимость шага 1 составляет

$$\mathcal{O}(n^3q + n^2q + nq^2) = \mathcal{O}(n^3q).$$

Однако матрица Γ может быть вычислена без явного формирования матриц Хаусхолдера.

2.3. Вычисление матрицы Грама. Теперь мы опишем, как быстро вычислять элементы матрицы Γ. Так как

$$\Gamma_{sl} = (b_s, b_l)_F, \qquad i, j = 1, \ldots, n,$$

нам нужно вычислить скалярные произведения $(b_s, b_l)_F$. Из определения b_j следует, что

$$b_j = \widehat{a}_j - \widehat{a}_j e_1 e_1^T - e_1 e_1^T \widehat{a}_j + (\widehat{a}_j)_{11} e_1 e_1^T.$$

Требуемые скалярные произведения равны

$$(b_s, b_l)_F = (\widehat{a}_s, \widehat{a}_l)_F - (\widehat{a}_s e_1, \widehat{a}_l e_1) - (\widehat{a}_s^T e_1, \widehat{a}_l^T e_1) + (\widehat{a}_s)_{11}(\widehat{a}_l)_{11}.$$

Так как $\widehat{a}_j = C a_j H^T$, мы получаем окончательную формулу

$$\Gamma_{sl} = (b_s, b_l)_F = (a_s, a_l)_F - \frac{(a_s y, a_l y)}{||y||^2} - \frac{(a_s^T \lambda, a_l^T \lambda)}{||\lambda||^2} + \frac{(a_s y, \lambda)(a_l y, \lambda)}{||y||^2 ||\lambda||^2}. \tag{11}$$

Формула (11) позволяет нам быстро вычислять элементы матрицы Γ. Заметим, что

$$(a_s, a_l)_F = \sum_{ki} (a_s)_{ki} (a_l)_{ki} = \sum_{ki} (A_k)_{is} (A_k)_{il} = (\sum_{k=1}^{n} A_k A_k^T)_{sl},$$

поэтому первое слагаемое $(a_s, a_l)_F$ может быть вычислено один раз для всех y и λ за $\mathcal{O}(n^3 q)$ операций. Стоимость вычисления векторов $a_s y$ и $a_s^T \lambda$ для всех $s = 1, \ldots, n$ равна $\mathcal{O}(n^2 q + q^2 n)$. Стоимость вычисления скалярных произведений $(a_s y, a_l y)$ и $(a_s^T \lambda, a_l^T \lambda)$ того же порядка. Суммарная стоимость вычисления Γ равна

$$\mathcal{O}(n^3 q),$$

один раз для заданных матриц $A_1, \ldots, A_k$ плюс

$$\mathcal{O}(n^2 q + n q^2)$$

операций для каждых конкретных векторов y и λ.

Теперь мы готовы описать алгоритм полностью.

Алгоритм 2.

Пусть даны матрицы $A_1, \ldots, A_q$, начальное приближение x^0, y^0, λ^0 к решению задачи (4). Тогда выполнить:

1. Установить $k = 0$ и вычислить начальную матрицу Грама
$$\Gamma_0 = \sum_{i=1}^{q} A_i^T A_i.$$
2. Если процесс сошёлся, остановиться.
3. Вычислить векторы $a_s y^k$ и $a_s \lambda^k$ для всех $s = 1, \ldots, n$.
4. Вычислить матрицу Γ, используя формулу (11).
5. Вычислить x^{k+1} как собственный вектор, отвечающий наименьшему собственному значению матрицы Γ.
6. Вычислить y^{k+1} и λ^{k+1} по формулам (10).
7. Увеличить k на 1 и перейти к шагу 2.

Важно отметить, что матрица Γ может пересчитываться во время работы алгоритма 1 для вычисления обобщённого разложения Шура. Действительно, самая «дорогая» работа в алгоритме 2 — это вычисление матрицы
$$\Gamma_0 = \sum_{k=1}^{q} B_k^T B_k.$$
После QZ преобразования каждой B_k матрица Γ_0 переходит в
$$\sum_{k=1}^{q} \widehat{B}_k^T \widehat{B}_k = \sum_{k=1}^{q} (Q_m B_k Z_m)^T Q_m B_k Z_m = Z_m^T \Gamma_0 Z_m.$$
Нам нужно вычислить
$$\widehat{\Gamma}_0 = \sum_{k=1}^{q} C_k^T C_k,$$
где C_k — это подматрица порядка $m-1$ матрицы $\widehat{B}_k$, занимающая столбцы $2, \ldots, m$ и строки $2, \ldots, m$. Легко показать, что
$$(C_k^T C_k)_{ij} = (\widehat{B}_k^T \widehat{B}_k)_{(i+1)(j+1)} - (\widehat{B}_k)_{i1} (\widehat{B}_k)_{j1},$$
$$i = 1, \ldots, n-1, \quad j = 1, \ldots, n-1,$$

поэтому

$$(\widehat{\Gamma}_0)_{ij} = (Z_m^T \Gamma_0 Z_m)_{(i+1)(j+1)} - \sum_{k=1}^{q} (\widehat{B}_k)_{i1} (\widehat{B}_k)_{j1},$$

$$i = 1, \ldots, n-1, \quad j = 1, \ldots, n-1.$$

Стоимость такого пересчёта составляет всего $\mathcal{O}(n^2 q)$.

Нам осталось исследовать свойства сходимости нашего алгоритма.

3. Сходимость

Предположим, что векторы x^*, y^* и вектор λ^* являются решением минимизационной задачи

$$\sum_{k=1}^{q} ||A_k x - \lambda_k y||^2 \to \min, \qquad ||x||_2 = 1, \tag{12}$$

и при этом

$$A_k x^* = \lambda_k^* y^* + \varepsilon_k. \tag{13}$$

Пусть у нас также есть некоторое приближение к решению:

$$y = y^* + \delta y,$$

$$\lambda = \lambda^* + \delta\lambda,$$

и новое приближение x к x^* вычисляется с помощью Алгоритма 2. Что можно сказать о $||x - x^*||$?

Запишем (13) в терминах матриц a_j (8):

$$\sum_{j=1}^{n} x_j^* a_j = \lambda^* (y^*)^T + \varepsilon, \tag{14}$$

где мы также ввели невязку ε. Нам потребуются также нормированные векторы

$$\widetilde{y} = \frac{y}{||y||}, \quad \widetilde{\lambda} = \frac{\lambda}{||\lambda||}, \quad \widetilde{y}^* = \frac{y^*}{||y^*||}, \quad \widetilde{\lambda}^* = \frac{\lambda^*}{||\lambda^*||}.$$

Вектор x является собственным вектором матрицы Γ (11). Обозначим через Γ^* матрицу (11) при $y = y^*$ и $\lambda = \lambda^*$. Тогда верна следующая

Лемма.

$$|(\Gamma x^* - \Gamma^* x^*)_s| \leqslant ||a_s|| \left(||y^*||\,||\lambda^*||(4\,||\delta\widetilde{y}||^2 + ||\delta\widetilde{y}||\,||\delta\widetilde{\lambda}|| + 4\,||\widetilde{\lambda}||^2 \right) + \\ + ||\varepsilon|| \left(||\delta\widetilde{y}|| + ||\delta\widetilde{\lambda}|| \right)) + \mathcal{O}(\delta^3 + ||\varepsilon||\delta^2 \Big),$$

где $\delta = \max(||\delta y||, ||\delta\lambda||)$.

Доказательство.

Используя определение матрицы Γ и равенство (14) получаем:

$$\begin{aligned}((\Gamma - \Gamma_0)x^*)_s &= -\left(a_s\widetilde{y}, \sum_{l=1}^{n} x_l^* a_l \widetilde{y}\right) - \left(a_s^T\widetilde{\lambda}, \sum_{l=1}^{n} x_l^* a_l^T \widetilde{\lambda}\right) + \\ &+ \left(a_s\widetilde{y}, \widetilde{\lambda}\right)\left(a_s\widetilde{y}, \widetilde{\lambda}\right)\left(\sum_{l=1}^{n} x_l^* a_l \widetilde{y}, \widetilde{\lambda}\right) = \\ &= -(a_s\widetilde{y}, \lambda^*)(y^*, \widetilde{y}) - (a_s\widetilde{y}, \varepsilon\widetilde{y}) - \left(a_s^T\widetilde{\lambda}, y^*\right)\left(\lambda^*, \widetilde{\lambda}\right) - \\ &- \left(a_s^T\widetilde{\lambda}, \varepsilon^T\widetilde{\lambda}\right) + \left(a_s\widetilde{y}, \widetilde{\lambda}\right)((y^*, \widetilde{y})\left(\lambda^*, \widetilde{\lambda}\right) + \left(\varepsilon\widetilde{y}, \widetilde{\lambda}\right)).\end{aligned}$$

Установим теперь

$$\widetilde{y} = \widetilde{y}^* + \delta\widetilde{y}, \qquad \widetilde{\lambda} = \widetilde{\lambda}^* + \delta\widetilde{\lambda}.$$

Так как $||\widetilde{y}|| = ||\widetilde{y}^*|| = 1$, то

$$(\delta\widetilde{y}, y^*) = -2\,||y||^*\,||\delta\widetilde{y}||^2,$$

и мы получаем следующие члены первого порядка по δ (обозначим их через Φ_1):

$$\begin{aligned}\Phi_1 = -(a_s\delta\widetilde{y}, \varepsilon\widetilde{y}^*) &- (a_s\widetilde{y}^*, \varepsilon\delta\widetilde{y}) - (a_s^T\widetilde{\lambda}^*, \varepsilon^T\delta\widetilde{\lambda}) - (a_s^T\delta\widetilde{\lambda}, \varepsilon^T\widetilde{\lambda}^*) + \\ &+ (a_s\widetilde{y}^*, \delta\widetilde{\lambda})(\varepsilon\widetilde{y}^*, \widetilde{\lambda}^*) + (a_s\delta\widetilde{y}, \widetilde{\lambda}^*)(\varepsilon\widetilde{y}^*, \widetilde{\lambda}^*) + \\ &+ (a_s\widetilde{y}^*, \widetilde{\lambda}^*)\left((\varepsilon\delta\widetilde{y}, \widetilde{\lambda}^*) + (\varepsilon\widetilde{y}^*, \delta\widetilde{\lambda})\right).\end{aligned}$$

Число Φ_1 можно оценить сверху как

$$\Phi_1 \leqslant 4\,||a_s||\,||\varepsilon||\,(||\delta\widetilde{y}|| + ||\delta\widetilde{\lambda}||).$$

Члены второго порядка оцениваются аналогично (мы отбрасываем члены порядка $\mathcal{O}(\delta^2||\varepsilon||)$). Приведём здесь только конечный результат:

$$|\Phi_2| \leqslant ||y^*||\,||\lambda^*||\,||a_s||\,||\delta\widetilde{y}||\,||\delta\widetilde{\lambda}|| + 4\,||a_s||\,||y^*||\,||\lambda^*||\,(||\delta\widetilde{y}||^2 + ||\delta\widetilde{\lambda}||^2).$$

Для завершения доказательства осталось заметить, что

$$|(\Gamma - \Gamma^*)x^*)_s| \leqslant (|\Phi_1| + |\Phi_2|) + \mathcal{O}(\delta^3 + \delta^2\varepsilon).$$

Теперь мы можем оценить $||x - x^*||$:

Теорема 1.

Если x вычисляется по y и λ с помощью Алгоритма 2, а векторы x^*, y^* и λ^* являются решением минимизационной задачи (12), то

$$||x - x^*|| \leqslant \frac{1}{\gamma_{n-1} - \gamma_n} \sqrt{\sum_{s=1}^{n} ||a_s||^2 (6\delta^2 ||y^*||\ ||\lambda^*|| + 2\ ||\varepsilon||\ \delta) + \mathcal{O}(\delta^3 + ||\varepsilon||\delta^2)}, \tag{15}$$

где

$$\delta = \max(||\delta y||, ||\delta\lambda||)$$

и γ_n, γ_{n-1} — два младших собственных числа матрицы Γ^*.

Если мы будем рассматривать матрицу Γ как возмущение матрицы Γ^*, то x является возмущением собственного вектора x^*. Используя теорему о возмущении собственного вектора симметричной вещественной матрицы, получаем, что

$$||x - x^*|| \leqslant \frac{1}{\gamma_{n-1} - \gamma_n} ||(\Gamma - \Gamma^*)x^*||.$$

Применяя теперь неравенство (15) и принимая во внимание, что

$$||\delta\widetilde{y}|| \leqslant ||\delta y||, \qquad ||\delta\widetilde{\lambda}|| \leqslant ||\delta\lambda||,$$

мы получаем (15).

Оценка (15) полностью описывает локальные свойства сходимости метода. Если $||\varepsilon|| = 0$ (т.е. возможно точное приведение матриц A_k к блочно-треугольному виду), сходимость остаётся квадратичной. В случае ненулевого, но достаточно малого $||\varepsilon||$ сходимость становится линейной, но скорость сходимости пропорциональна $||\varepsilon||$.

Замечание.

Численные эксперименты показывают, что в случае достаточно малого $||\varepsilon||$ алгоритм сходится глобально. На данный момент мы не имеем каких-либо строгих формулировок этого утверждения.

4. Численные эксперименты

В этом параграфе мы представим некоторые численные эксперименты, демонстрирующие эффективность нашего метода. Тестовые примеры создавались следующим образом. Мы генерировали случайные матрицы U, V, W размеров $n \times n, n \times n$ и $n \times q$ соответственно и формировали тензор

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{n} u_{i\alpha} v_{j\alpha} w_{k\alpha}.$$

Элементы матриц брались из равномерного распределения на отрезке [-1,1]. После этого матрицы A_k формировались как срезки этого тензора по направлению k. Также мы накладывали на элементы тензора мультипликативный шум

$$\mathcal{A}_{ijk} = \mathcal{A}_{ijk}(1 + \sigma\phi),$$

где случайная величина ϕ также имела равномерное распределение на отрезке $[-1, 1]$, а σ это некоторый «уровень шума».

После этого с помощью алгоритма 1 матрицы A_k одновременно приводились к верхнетреугольному виду. Нас интересовали следующие величины:

- Скорость сходимости, её зависимость от n, q и σ.
- Устойчивость: зависимость невязки вычисленного разложения от σ.

Мы обнаружили, что скорость сходимости алгоритма не зависит сколько-нибудь заметно от σ, поэтому не приводим никаких таблиц по этой зависимости.

Мы выполнили два эксперимента. Сначала мы фиксируем q и σ, установив их равными 10 и 10^{-6} соответственно, и меняем n. Время работы алгоритма 1 (в секундах) приведено в табл. 1. Во втором эксперименте мы полагаем $q = n$. Результаты счёта приведены в табл. 2.

Проверка устойчивости проводилась следующим образом. Мы установили $q = n = 64$ и меняли уровень шума. Для каждого σ создавалось 10 наборов матриц и вычислялись минимальная, максимальная и средняя невязки.

Из приведённых численных экспериментов видно, что предложенный метод построения суперобобщённого разложения Шура является, во-первых, быстрым, а во-вторых (что нам кажется самым важным), алгоритм даёт достаточно достоверное разложение.

Таблица 1

Время счёта (в секундах) для вычисления суперобобщённого разложения Шура для 10 матриц при разных n

n	Время
16	0.01
32	0.11
64	1.6
128	14.77
256	210.61

Таблица 2

Время счёта (в секундах) для вычисления суперобобщённого разложения Шура для n матриц размеров $n \times n$ при разных n

n	Время счёта
16	0.02
32	0.14
64	3.41
128	54.96
256	810.77

Таблица 3

Невязки для различных уровней шума

σ	*Средняя невязка*	*Минимальная невязка*	*Максимальная невязка*
10^{-16}	$2 \cdot 10^{-15}$	$9 \cdot 10^{-16}$	$5 \cdot 10^{-15}$
10^{-15}	$7 \cdot 10^{-15}$	$1 \cdot 10^{-15}$	$2 \cdot 10^{-14}$
10^{-14}	$3 \cdot 10^{-14}$	$4 \cdot 10^{-15}$	$8 \cdot 10^{-14}$
10^{-13}	$5 \cdot 10^{-14}$	$4 \cdot 10^{-14}$	$8 \cdot 10^{-14}$
10^{-12}	$2 \cdot 10^{-12}$	$4 \cdot 10^{-13}$	$4 \cdot 10^{-12}$
10^{-11}	$6 \cdot 10^{-11}$	$4 \cdot 10^{-12}$	$1 \cdot 10^{-11}$
10^{-10}	$4 \cdot 10^{-10}$	$4 \cdot 10^{-11}$	$1 \cdot 10^{-9}$
10^{-9}	$2 \cdot 10^{-8}$	$4 \cdot 10^{-10}$	$5 \cdot 10^{-8}$
10^{-8}	$4 \cdot 10^{-8}$	$4 \cdot 10^{-9}$	$1 \cdot 10^{-7}$
10^{-7}	$2 \cdot 10^{-7}$	$4 \cdot 10^{-8}$	$7 \cdot 10^{-7}$
10^{-6}	$6 \cdot 10^{-7}$	$4 \cdot 10^{-7}$	$1 \cdot 10^{-6}$
10^{-5}	$2 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$5 \cdot 10^{-5}$
10^{-4}	$4 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1 \cdot 10^{-3}$
10^{-3}	$2 \cdot 10^{-3}$	$4 \cdot 10^{-4}$	$6 \cdot 10^{-3}$

Список литературы

[1] Chu M. A continues Jacobi-like approach to the simultaneous reduction of real matrices // *Linear Alg. Appl.* 1991. V. 147. P. 75-96.

[2] Bunse-Gerstner A., Byers R., Mehrmann V. Numerical methods for simultaneous diagonalization // *SIAM J. Matrix Anal. Appl.* 1993. V. 14. P. 927-949

[3] Ziehe A., Kawanabe M., Hamerling S., Müller K.-R. A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation // *Journal of Machine Learning Research* 2004. V. 5. P. 801-818.

[4] Van der Veen A.-J., Paulraj A. An analytical constant modulus algorithm // *IEEE Trans. Signal Process.* 1996. V. 44. P. 1136-1155

[5] De Lathauwer L., De Moor B., Vanderwalle J. Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition // *SIAM J. Matrix Anal. Appl.* 2004. V. 26. P. 295-227

[6] Оселедец И. В., Савостьянов Д. В. Методы разложения тензора. // *Настоящий сборник.* С. 51-64.

[7] Оселедец И. В., Савостьянов Д. В. Об одном алгоритме построения трилинейного разложения // *Настоящий сборник.* С. 117-130.

Об одном алгоритме построения тензорного разложения[a]

И.В. Оселедец, Д. В. Савостьянов

Предложен быстрый алгоритм для поиска переопределенного разложения для заданного трехмерного тензора, то есть аппроксимации его тензором малого тензорного ранга, в случае, когда этот ранг превосходит размеры исходного тензора. Доказана сходимость метода, проведены численные эксперименты.

1. Введение

В данной работе мы будем рассматривать следующую задачу о *приближении заданного тензора (трёхмерного массива) тензором малого тензорного ранга*:

$$\mathcal{A}_{ijk} \approx \mathcal{B}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}, \tag{1}$$

где $\mathcal{A} = \mathcal{A}_{ijk}$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, $k = 1, \ldots, n_3$ — заданный тензор (трёхмерный массив) с размерами $n_1 \times n_2 \times n_3$, число r называется тензорным рангом массива $\mathcal{B}$, а матрицы $U = [u_{i\alpha}]$, $V = [v_{j\alpha}]$, $W = [w_{k\alpha}]$ *опорными матрицами,* или *факторами* разложения (1). Тензоры с элементами вида $u_i v_j w_k$ мы будем называть *триплетами*. Обзор имеющихся методов для решения задачи (1) можно найти в [3].

Нас будет интересовать случай, когда $r > \max(n_1, n_2, n_3)$. При таком условии говорят о *переопределённом разложении*. Для упрощения мы также будем рассматривать только *кубический случай*:

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

$n_1 = n_2 = n_3 = n$. Также мы будем предполагать, что в (1) выполняется точное равенство:

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}. \tag{2}$$

В данной работе будет построен быстрый алгоритм для вычисления тензорного разложения при $n+1 \leqslant r \leqslant 2n-3$.

2. Описание алгоритма

Мы будем рассматривать тензор $\mathcal{A}$ как набор из n матриц-*срезок* размера $n \times n$:

$$(A_k)_{ij} = \mathcal{A}_{ijk}, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n.$$

В терминах матриц A_k разложение (2) можно записать в виде

$$A_k = \sum_{\alpha=1}^{r} w_{k\alpha} R_\alpha, \tag{3}$$

где R_α — матрицы ранга 1, заданные в виде

$$R_\alpha = u_\alpha v_\alpha^T.$$

Сложим теперь равенства (3), домножив их на некоторые коэффициенты λ_k:

$$\sum_{k=1}^{n} \lambda_k A_k = \sum_{\alpha=1}^{r} \widehat{w}_\alpha R_\alpha,$$

где вектор $\widehat{w}$ размера r равен

$$\widehat{w} = W^T \lambda.$$

У нас есть n свободных переменных λ_k. Потребуем теперь, чтобы в векторе $\widehat{w}$ было $n-1$ нулей. Для этого достаточно предположить, что среди r векторов (напомним, $r > n$) w_α существует n линейно независимых. В таком случае получаем

$$\sum_{k=1}^{n} \lambda_k A_k = B,$$

где матрица B есть сумма $r-n+1$ матриц вида $u_\beta v_\beta^T$, т.е. её ранг не превосходит $r-n+1$. Поэтому получаем следующее уравнение:

$$\sum_{k=1}^{n} \lambda_k A_k = XY^T, \tag{4}$$

где X, Y имеют размеры $n \times q$, где $q = r-n+1 \geqslant 2$. Сделаем здесь ещё одно предположение на ранг тензора r :

$$q = r-n+1 \leqslant n-2.$$

Почему мы делаем именно такое предположение? Ответ очень простой: мы хотим использовать (4) как уравнение, определяющее X и Y. Если $q \geqslant n$, то (4) не накладывает никаких ограничений на λ. Если же $q = n-1$ равенство (4) означает лишь, что

$$\det(\sum_{k=1}^{n} \lambda_k A_k) = 0,$$

т.е. у нас всего лишь одно уравнение на n неизвестных, чего явно недостаточно.

Согласно построению, столбцы матрицы X есть линейная комбинация некоторых q столбцов матрицы U, а столбцы матрицы Y есть линейная комбинация некоторых q столбцов матрицы V, причём эти столбцы принадлежат одним и тем же триплетам тензорного разложения.

Вариант алгоритма Гаусса-Ньютона для решения задачи (4) будет описан в следующем параграфе. Пусть теперь у нас есть подпрограмма, которая способна решать задачу (4) и находить значения λ_k, X и Y. Покажем, что используя эту подпрограмму как чёрный ящик, можно построить разложение (2).

Как уже было сказано, столбцы X и Y есть линейная комбинация некоторых q столбцов матрицы U и V *с одинаковыми номерами*:

$$X = [u_{\alpha_1}, u_{\alpha_2}, \ldots, u_{\alpha_q}]\Phi_1, \tag{5}$$

$$Y = [v_{\alpha_1}, v_{\alpha_2}, \ldots, v_{\alpha_q}]\Phi_2, \tag{6}$$

где Φ_1 и Φ_2 суть квадратные $q \times q$ матрицы. Развёртывая наш тензор $\mathcal{A}$ не по индексу k, а, например, по i и выписывая задачу, аналогичную (4), мы можем найти матрицу Z, столбцы которой дают базис в линейной оболочке столбцов $w_{\alpha_1}, \ldots, w_{\alpha_q}$:

$$Z = [w_{\alpha_1}, w_{\alpha_2}, \ldots, w_{\alpha_q}]\Phi_3. \tag{7}$$

Действительно, если мы обозначим через C_i срезки тензора по индексу i :

$$(C_i)_{kj} = \mathcal{A}_{ijk},$$

то задача для определения Z запишется как

$$\sum_{i=1}^{n} \lambda_i C_i = ZY^T.$$

Так как Y нам известно, то для нахождения Z достаточно решить линейную систему наименьших квадратов относительно неизвестных λ_i, Z.

После того как матрицы X, Y, Z найдены, построим их QR-разложения:

$$X = Q_x R_x, \qquad Y = Q_y R_y, \qquad Z = Q_z R_z,$$

где Q_x, Q_y, Q_z имеют размер $n \times q$, матрицы R_x, R_y, R_z имеют размер $q \times q$. Далее построим ортогональные матрицы H_x, H_y, H_z размера $n \times n$, которые переводят столбцы Q_x, Q_y, Q_z в первые столбцы единичной матрицы

$$H_x Q_x = H_y Q_y = H_z Q_z = I_q,$$

и свернём тензор $\mathcal{A}$ с матрицами H_x, H_y и H_z по индексам i, j, k соответственно:

$$\mathcal{B}_{ijk} = \sum_{i'=1}^{n} \sum_{j'=1}^{n} \sum_{k'=1}^{n} \mathcal{A}_{i'j'k'} (H_x)_{ii'} (H_y)_{jj'} (H_z)_{kk'}. \tag{8}$$

Теперь наша задача свелась к задаче нахождения тензорного разложения для тензора $\mathcal{B}$:

$$\mathcal{B}_{ijk} = \sum_{\alpha=1}^{r} \widehat{u}_{i\alpha} \widehat{v}_{j\alpha} \widehat{w}_{k\alpha},$$

где

$$\widehat{U} = H_x U, \qquad \widehat{V} = H_y V, \qquad \widehat{W} = H_z W.$$

Представим $\mathcal{B}$ в виде

$$\mathcal{B} = \mathcal{B}' + \mathcal{B}'',$$

где в $\mathcal{B}'$ включены триплеты, отвечающие номерам $\alpha_1, \ldots, \alpha_q$, а в $\mathcal{B}''$ — все остальные.

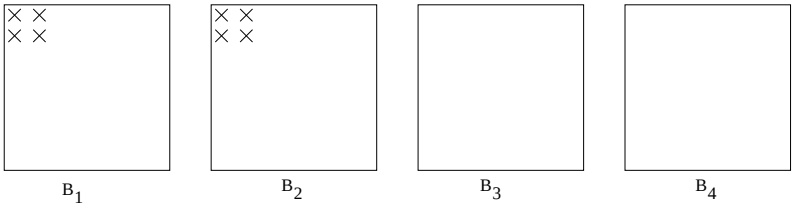


Рис. 1. Положение ненулевых элементов в тензоре $\mathcal{B}'$

В силу построения матриц H_x, H_y, H_z элементы $\mathcal{B}'_{ijk}$ могут быть не равны нулю лишь при $i, j, k \leqslant q$ (рис. 1). Поэтому если мы рассмотрим тензор $\widehat{\mathcal{B}}$ с размерами $n \times n \times (n-q)$, заданный как

$$\widehat{\mathcal{B}} = [B_{q+1}, ..., B_n]$$

(B_k – это срезки массива $\mathcal{B}$ по индексу k), то это будет тензор ранга $r - q = r - (r - n + 1) = n - 1$. Так как ранг равен $n - 1$, то используя метод редукции Таккера, задачу можно свести к тензору с размерами $(n-1) \times (n-1) \times (n-q)$ и рангом $n-1$. Опять будем считать, что у нас есть процедура („чёрный ящик“), вычисляющая разложение тензора $\widehat{\mathcal{B}}$:

$$\widehat{\mathcal{B}}_{ijk} = \sum_{\beta=1}^{n-1} u^0_{i\beta} v^0_{j\beta} w^0_{k\beta},$$

где размеры матриц факторов U_0, V_0, W_0 равны $n \times (n-1)$, $n \times (n-1)$, $(n-q) \times (n-1)$ соответственно.

Легко видеть, что столбцы матриц U_0 и V_0 совпадают с некоторыми $(n-1)$ столбцами матриц $\widehat{U}, \widehat{V}$, т.е. к данному моменту мы нашли:

- $(n-1)$ столбец из $\widehat{U}$ и $(n-1)$ столбец из $\widehat{V}$ с некоторыми номерами $\beta_1, \ldots, \beta_{n-1}$.
- Элементы с индексами $(q+1), \ldots, n$ в столбцах $\widehat{W}$ с теми же самыми номерами $\beta_1, \ldots, \beta_{n-1}$.

Как найти теперь первые q элементов в каждом из столбцов $w_{\beta_1,\ldots,\beta_n}$? Для этого рассмотрим первые q срезок тензора $\mathcal{B}$. В них

элементы тензора $\mathcal{B}'$ влияют только на угловые подматрицы размером $q \times q$, поэтому нужные нам элементы $\widehat{W}$ находятся с помощью решения q линейных системы наименьших квадратов

$$\sum_{i=q+1}^{n} \sum_{j=q+1}^{n} \left(\mathcal{B}_{ijk} - \sum_{\beta=1}^{n-1} u^0_{i\beta} v^0_{j\beta} \widehat{w}_{k\beta} \right)^2 \to \min, \qquad k = 1, \ldots, q.$$

После того как $(n-1)$ триплет найден, мы фактически нашли тензор $\mathcal{B}''$. Поэтому, вычитая его из тензора $\mathcal{B}$, мы получим тензор $\mathcal{B}'$ ранга q. Применяя опять подпрограмму разложения тензора с рангом, равным одному из размеров, находим оставшиеся q триплетов.

Подчеркнём, что для тензора с рангом, равным одному из размеров, существуют несколько эффективных алгоритмов, которые, однако, нельзя применить для разложения тензора с рангом $r > n$. В этом и состоит наш основной результат: используя апробированные алгоритмы для построения разложения тензора с $r = n$, мы получаем свой алгоритм для разложения тензора с $r > n$.

В качестве процедуры разложения тензора при $r = n$ мы будем использовать быстрый метод, основанный на использовании суперобобщённого разложения Шура [2, 1]. Для вычисления этого разложения мы будем использовать быстрый алгоритм, предложенный в [4]. Запишем теперь приведённую процедуру в виде алгоритма.

Алгоритм 1.

Пусть дан тензор $\mathcal{A}$ размера $n \times n \times n$ ранга $r > n$. Тогда алгоритм вычисления разложения (2) для тензора $\mathcal{A}$ состоит из следующих шагов:

1. Установить $q = r - n + 1$.
2. Решить задачу
$$\sum_{k=1}^{n} \lambda_k A_k = XY^T,$$
где X, Y имеют размер $n \times q$, а A_k — срезки тензора $\mathcal{A}$ по индексу k.
3. Используя найденные Y, найти Z из решения линейной системы наименьших квадратов относительно переменных λ, Z
$$\sum_{i=1}^{n} \lambda_i C_i = ZY^T,$$
где C_i — срезки тензора $\mathcal{A}$ по индексу i.

4. Найти QR-разложение матриц X, Y, Z :

$$X = Q_x R_x, \qquad Y = Q_y R_y, \qquad Z = Q_z R_z.$$

5. Найти ортогональные матрицы H_x, H_y, H_z размера $n \times n$ (как произведение q матриц отражений), которые переводят столбцы матриц Q_x, Q_y, Q_z соответственно в первые q столбцов единичной матрицы.

6. Вычислить матрицы B_k, $k = 1, \ldots, n$, по формуле

$$B_k = \sum_{k'=1}^{n} (H_z)_{kk'} H_x A'_k H_y^T, \qquad k = 1, \ldots, n.$$

7. Сформировать тензор $\widehat{\mathcal{B}}$:

$$\widehat{\mathcal{B}} = [B_{q+1}, \ldots, B_n]$$

и построить приближение к нему ранга $(n-1)$ с помощью алгоритма, описанного в [4]:

$$\widehat{\mathcal{B}}_{ijk} = \sum_{\beta=1}^{n-1} \widehat{u}_{i\beta} \widehat{v}_{j\beta} w^0_{k\beta}.$$

8. Записать векторы $w^0_{k\beta}$ в массив $\widehat{W}$:

$$\widehat{w}_{(k+q)\beta} = w^0_{k\beta}, \qquad \beta = 1, \ldots, n-1, \quad k = 1, \ldots, n-q.$$

9. Найти элементы $\widehat{w}_{k\beta}$, $\beta = 1, \ldots, n-1$, $k = 1, \ldots, q$, из решения q линейных систем наименьших квадратов:

$$\sum_{i=q+1}^{n} \sum_{j=q+1}^{n} \left(\mathcal{B}_{ijk} - \sum_{\beta=1}^{n-1} \widehat{u}_{i\beta} \widehat{v}_{j\beta} \widehat{w}_{k\beta} \right)^2 \to \min, \qquad k = 1, \ldots, q.$$

10. Вычислить тензор $\mathcal{B}'$ размером $q \times q \times q$:

$$\mathcal{B}'_{ijk} = \mathcal{B}_{ijk} - \sum_{\beta=1}^{n-1} \widehat{u}_{i\beta} \widehat{v}_{j\beta} \widehat{w}_{k\beta}, \qquad i, j, k = 1, \ldots, q.$$

11. Вычислить разложение тензора $\mathcal{B}'$:

$$\mathcal{B}_{ijk} = \sum_{\beta=1}^{q} u'_{i\beta} v'_{j\beta} w'_{k\beta}$$

и записать элементы векторов $u'_\beta, v'_\beta, w'_\beta$ в первые q элементов векторов $\widehat{u}_{\beta+n-1}, \widehat{v}_{\beta+n-1}, \widehat{w}_{\beta+n-1}$ соответственно, а остальные $(n-q)$ элементов этих векторов положить равными нулю.

12. Вернуться в исходный базис:

$$U = H_x^T \widehat{U}, \qquad V = H_y^T \widehat{V}, \qquad W = H_z^T \widehat{W}.$$

Тогда $U = [u_{i\alpha}],\ V = [v_{j\alpha}],\ W = [w_{k\alpha}]$ — искомое разложение.

Весь алгоритм будет работать и в том случае, когда равенство (2) выполнено не точно, а с некоторой погрешностью ε.

3. Алгоритм MLR

В этом разделе мы приведём метод решения задачи (4). Мы будем называть его алгоритмом MLR (mixed low rank), или смешанной аппроксимации, т.к. в левой части стоит выражение, линейное по неизвестным, а справа — матрица малого ранга. При фиксированном λ матрицы X, Y находятся с помощью сингулярного разложения (задача о наилучшем приближении матрицы матрицей заданного ранга), а при фиксированных X, Y вектор λ находится с помощью решения линейной задачи наименьших квадратов. Можно предложить простой алгоритм переменных направлений, основанный на этих наблюдениях: фиксируем λ и находим X, Y, потом по этим X, Y находим λ и т.д. Недостатком метода переменных направлений (в английской литературе такие методы часто называются ALS (alternating least squares)) является довольно медленная сходимость.

Построим вариант метода Гаусса-Ньютона для решения задачи (4). Идея метода Гаусса-Ньютона состоит в следующем. Будем рассматривать (4) как переопределённую систему нелинейных уравнений, линеаризуем её относительно некоторой точки (λ, X, Y) :

$$\sum_{k=1}^{r} \lambda_k A_k = X \triangle Y^T + \triangle X Y^T + X Y^T \tag{9}$$

(для краткости мы пишем λ вместо $\lambda+\triangle\lambda$). Будем решать её как переопределённую систему линейных уравнений. Отметим, что λ, X, Y определены с точностью до умножения на константу, поэтому добавим нормировочное условие

$$||\lambda||_2 = 1.$$

Заметим, что с помощью «короткого» сингулярного разложения мы всегда можем представить матрицу XY^T в виде $X'Y'^T$, где столбцы X' и Y' ортогональны; поэтому мы всегда будем считать, что X и Y имеют ортогональные столбцы. В этом случае мы можем найти ортогональные матрицы H_1 и H_2 размера $n \times n$, такие что

$$H_1 X = I_q \Lambda_1, \qquad H_2 Y = I_q \Lambda_2,$$

где I_q — первые q столбцов единичной матрицы размера n, а Λ_1, Λ_2 — диагональные матрицы размера $q \times q$. Домножая уравнения (9) на H_1 слева и на H_2^T справа, получаем эквивалентную систему:

$$\sum_{k=1}^{n} \lambda_k \widehat{A}_k = I_q \triangle\widehat{Y}^T + \triangle\widehat{X} I_q^T + I_q \Lambda_1 \Lambda_2 I_q^T, \tag{10}$$

где введены новые неизвестные

$$\triangle\widehat{Y} = \Lambda_1 H_1 \triangle Y, \qquad \triangle\widehat{X} = \Lambda_2 H_2 \triangle X, \qquad \widehat{A}_k = H_1 A_k H_2^T.$$

Правая часть (10) равна нулю при $i \geqslant q+1$, $j \geqslant q+1$, поэтому для определения λ имеем следующую задачу

$$\left\| \sum_{k=1}^{n} \lambda_k \widetilde{A}_k \right\|_F \to \min,$$

где матрицы $\widetilde{A}_k$ получаются из $\widehat{A}_k$ занулением первых q столбцов и первых q строчек. Значит, для нахождения λ нужно найти правый сингулярный вектор $n^2 \times n$ матрицы

$$V = [\mathrm{vec}(\widetilde{A}_1), \ldots, \mathrm{vec}(\widetilde{A}_n)],$$

отвечающий её наименьшему сингулярному числу. Это можно сделать, например, с помощью сингулярного разложения. Укажем более экономичный способ (хотя и менее устойчивый в некоторых случаях), основанный на использовании матрицы Грама $\Gamma = V^T V$. Вектор λ должен быть собственным вектором матрицы Γ, отвечающим

её наименьшему собственному числу. Элементами матрицы Грама являются скалярные произведения столбцов в V :

$$\Gamma_{sl} = (v_s, v_l) = (\widetilde{A}_s, \widetilde{A}_l)_F.$$

Используя определение матриц $\widetilde{A}_s$, имеем

$$(\widetilde{A}_s, \widetilde{A}_l)_F = (\widehat{A}_s, \widehat{A}_l)_F - \sum_{\alpha=1}^{q} (\widehat{A}_s e_\alpha, \widehat{A}_l e_\alpha) - \sum_{\alpha=1}^{q} (\widehat{A}_s^T e_\alpha, \widehat{A}_l^T e_\alpha) + \\ + \sum_{\alpha=1}^{q}\sum_{\beta=1}^{q} (\widehat{A}_s e_\alpha, e_\beta)(\widehat{A}_l e_\alpha, e_\beta),$$

где через e_α обозначается столбец с номером α в единичной матрице. Если мы введём нормированные векторы

$$\widetilde{x}_\alpha = \frac{x_\alpha}{||x_\alpha||}, \qquad \widetilde{y}_\alpha = \frac{y_\alpha}{||y_\alpha||},$$

то, используя определение $\widehat{A}_s$ и матриц H_1, H_2, получаем следующую формулу для вычисления элементов матрицы Грама:

$$\Gamma_{sl} = (A_s, A_l)_F - \sum_{\alpha=1}^{q} ((A_s \widetilde{y}_\alpha, A_l \widetilde{y}_\alpha) + (A_s^T \widetilde{x}_\alpha, A_l^T \widetilde{x}_\alpha)) \\ - \sum_{\alpha=1}^{q}\sum_{\beta=1}^{q} (A_s \widetilde{y}_\alpha, \widetilde{x}_\beta)(A_l \widetilde{y}_\alpha, \widetilde{x}_\beta). \tag{11}$$

Затраты на вычисление различных слагаемых в (11) приведены в табл. 1. Суммарная сложность метода равна $\mathcal{O}(n^4) + N\mathcal{O}(n^3 q + n^2 q^2)$ операций, где N — число шагов алгоритма MLR, необходимых для сходимости.

После того как матрица Γ найдена, с помощью любого стандартного алгоритма определяется ее собственный вектор, отвечающий наименьшему собственному значению. По найденному значению λ новые матрицы X' и Y' несложно оцениваются через одну итерацию степенного метода:

$$\Phi = \sum_{k=1}^{n} \lambda_k A_k, \qquad X_{\text{new}} = \Phi Y, \qquad Y_{\text{new}} = \Phi^T X. \tag{12}$$

Полностью новый алгоритм выглядит следующим образом.

Алгоритм 2 (MLR).
Пусть даны матрицы $A_1, \ldots, A_n$, *число* q *и начальные приближения* $X_0, Y_0 \in \mathbb{R}^{n \times q}$ *к решению задачи (4). Тогда:*

1. Установить $X = X_0, Y = Y_0$.
2. Если процесс сошёлся, закончить.
3. Вычислить короткое сингулярное разложение:
$$XY^T = X'Y'^T,$$
$$X := X', \qquad Y := Y',$$
где столбцы X', Y' ортогональны.
4. Вычислить матрицу Γ по формуле (11) и найти её собственный вектор, отвечающий наименьшему собственному значению:
$$\Gamma\lambda = \gamma_{\min}\lambda.$$
5. Вычислить новые значения X и Y по формуле (12) и перейти к шагу 2.

Для определения количества итераций, необходимых для алгоритма MLR, проведём исследование сходимости метода.

4. Сходимость

Исследование сходимости метода Гаусса-Ньютона проводится следующим образом. Пусть λ^*, X^*, Y^* — решение минимизационной задачи

$$||\sum_{k=1}^{n} \lambda_k A_k - XY^T||_F \to \min, \qquad ||\lambda||_2 = 1, \tag{13}$$

и у нас есть некоторое приближение к решению:

$$X = X^* + \delta X, \qquad Y = Y^* + \delta Y, \qquad \lambda = \lambda^* + \delta\lambda,$$

и при этом

$$\sum_{k=1}^{n} \lambda_k^* A_k - X^*(Y^*)^T = E,$$

Таблица 1

Сложность вычисления различных слагаемых в (11)

Что вычисляем	Число операций
$(A_s, A_l)_F$	$\mathcal{O}(n^4)$
$A_s^T \widetilde{x}_\alpha$	$\mathcal{O}(n^3 q)$
$A_s \widetilde{y}_\alpha$	$\mathcal{O}(n^3 q)$
$(A_l \widetilde{y}_\alpha, \widetilde{x}_\beta)$	$\mathcal{O}(n^2 q^2)$

Таблица 2

Время вычисления разложения (в секундах) при $\sigma = 0$ *и различных* r, n

n	r	Время
16	24	0.1 с.
32	48	0.5 с.
64	96	17 с.
128	192	311 с.

Таблица 3

Зависимость невязки от уровня шума σ *при* $n = 100, r = 150$

σ	Невязка
10^{-15}	$1 \cdot 10^{-13}$
10^{-10}	$1 \cdot 10^{-9}$
10^{-5}	$3 \cdot 10^{-4}$

где $||E|| = \varepsilon$. Введём также невязку $\delta = \max(||\delta Y||, ||\delta X||)$. Тогда аналогично теореме 1 из [4] доказывается следующая

Теорема

Пусть по данным X, Y с помощью одного шага метода Гаусса-Ньютона вычисляется новое значение $\widehat{\lambda}$. Тогда справедлива следующая оценка:

$$||\widehat{\lambda} - \lambda^*|| \leqslant \frac{(\sum_{k=1}^{n} ||A_k||^2)^{1/2}}{\gamma_n - \gamma_{n-1}}(C_1\delta^2||X^*||\ ||Y^*|| + C_2\varepsilon\delta) + \mathcal{O}(\delta^3 + \delta^2\varepsilon),$$

где C_1, C_2 — некоторые небольшие постоянные, а γ_n, γ_{n-1} – два наименьших по модулю собственных числа матрицы Γ^* (Γ^* — это матрица (11) при $X = X^*$, $Y = Y^*$).

То есть если мы знаем X и Y с точностью δ, то мы находим λ уже с точностью $\mathcal{O}(\delta^2 + \varepsilon\delta)$. Так как мы предполагаем, что ε мало (или даже равно нулю), то скорость сходимости почти квадратичная.

Замечание. Методы, основанные на использовании матрицы Грама, могут привести к снижению точности, вплоть до $\sqrt{\eta}$, где η — машинная точность. Есть два способа исправить положение. Во-первых, можно использовать сингулярное разложение, или QR-разложение матрицы V, однако прямое применение этих методов увеличивает вычислительную сложность до $\mathcal{O}(n^4)$ на одну операцию, поэтому нужно разрабатывать специальные варианты этих разложений в этом случае. Второй способ, более простой с алгоритмической точки зрения, состоит в том, чтобы вычислять матрицу Грама в арифметике повышенной точности, вычислять разложение Холецкого $\Gamma = RR^T$ и фактор Холецкого R сохранять в одинарной точности.

5. Численные эксперименты

Мы провели следующие численные эксперименты, подтверждающие эффективность нашего алгоритма. Для различных r и n создавались случайные матрицы U, V, W с элементами, равномерно распределёнными на $[-1, 1]$, и образовывались тензоры

$$\mathcal{A}_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha}v_{j\alpha}w_{k\alpha}.$$

На эти тензоры налагался мультипликативный шум:

$$\mathcal{A}_{ijk} = \mathcal{A}_{ijk}(1 + \sigma\phi),$$

где ϕ брались из равномерного распределения на отрезке $[-1, 1]$, а σ — уровень шума. После этого с помощью алгоритма 1 вычислялось тензорное разложение (1) и рассчитывалась невязка. В табл. 2 приведено время счёта для различных значений r и n, а в табл. 3 демонстрируется устойчивость алгоритма (т.е. зависимость наблюдаемой невязки от уровня шума).

Список литературы

[1] Van der Veen A.-J., Paulraj A. An analytical constant modulus algorithm // *IEEE Trans. Signal Process.* 1996. V. 44. P. 1136-1155

[2] De Lathauwer L., De Moor B., Vanderwalle J. Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition // *SIAM J. Matrix Anal. Appl.* 2004. V. 26. P. 295-227.

[3] Оселедец И. В., Савостьянов Д. В. Методы разложения тензора // *Настоящий сборник.* С. 51-64.

[4] Оселедец И. В., Савостьянов Д. В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Настоящий сборник.* С. 101-116.

Минимизационные методы аппроксимации тензоров и их сравнение

И. В. Оселедец, Д. В. Савостьянов

Рассматривается приложение различных стандартных минимизационных методов к проблеме трилинейной аппроксимации тензоров и дается их сравнение на основе численных экспериментов.

1. Введение

В данной статье мы рассмотрим различные стандартные минимизационные методы в применении к проблеме *трилинейной аппроксимации тензоров*, которая формулируется следующим образом. Пусть дан некий трёхмерный массив (тензор) $\mathcal{A} = [a_{ijk}]$ размером $n_1 \times n_2 \times n_3$. Требуется найти такие матрицы $U = [u_{i\alpha}]$, $V = [v_{j\alpha}]$, $W = [w_{k\alpha}]$ размерами $n_1 \times r$, $n_2 \times r$, $n_3 \times r$, которые минимизируют функционал

$$\left\| a_{ijk} - \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha} \right\|, \tag{1}$$

где

$$\| \varphi_{ijk} \| = \left(\sum_{ijk} \varphi_{ijk}^2 \right)^{1/2}.$$

Мы рассмотрим три метода решения задачи (1):

`als`) метод переменных направлений;

`gn`) метод Гаусса-Ньютона;

`lm`) метод Ньютона с подбором шага

и проведём их сравнение на некоторых примерах.

2. Метод переменных направлений

Метод переменных направлений (в английской литературе метод называется ALS) является самым простым способом решения задачи (1). Он состоит в следующем. Пусть даны некоторые приближения U, V, W к решению U^*, V^*, W^* проблемы (1). Тогда по V, W находится новое значение $\widehat{U}$ из решения задачи наименьших квадратов:

$$\widehat{U} = \arg\min_U \left(\left\| a_{ijk} - \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha} \right\|^2 \right). \tag{2}$$

Задача (2) распадается на n_1 независимых подзадач для каждого $i = 1, ..., n_1$:

$$\sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \left(a_{ijk} - \sum_{\alpha=1}^{r} \widehat{u}_{i\alpha} v_{j\alpha} w_{k\alpha} \right)^2 \to \min. \tag{3}$$

Уравнения (3) представляют собой совокупность линейных переопределённых систем размерами $(n_2 n_3) \times r$ с *с одной и той же матрицей* (обозначим её через Φ), но с разными правыми частями. Прямое применение стандартных методов (например, QR–разложения) к матрице Φ потребует $\mathcal{O}(n_2 n_3 r^2)$ арифметических операций. Однако для матрицы Φ можно быстро вычислить матрицу Грама

$$\Gamma = \Phi^T \Phi = (V^T V) \circ (W^T W),$$

где через $\circ$ обозначается адамарово (поэлементное) произведение матриц. Матрица Γ может быть получена за $\mathcal{O}((n_2 + n_3) r)$ арифметических операций. После этого для нахождения $\widehat{u}_{i\alpha}$ нужно решить n_1 систем с матрицей Φ. Стоимость такого вычисления составляет $\mathcal{O}(r^3 + n_1 r^2)$ операций. После того как новая матрица U найдена, аналогичным способом по матрицам U, W оценивается матрица V, и так далее до тех пор пока не будет достигнута сходимость.

Метод ALS несложно запрограммировать, стоимость одной итерации невелика, и на каждой итерации невязка не возрастает. Все эти достоинства делают рассматриваемый метод самым популярным методом вычисления тензорных аппроксимаций. Однако ALS имеет несколько очень существенных недостатков. Во-первых, число итераций, необходимых для получения удовлетворительных результатов, может быть очень большим (тысячи или даже миллионы). Во-вторых, и это самое главное, метод может попасть в *локальный минимум* функционала (1). Однако если имеется хорошее начальное

приближение к точке минимума, то применение метода может оказаться довольно полезным.

3. Метод Гаусса-Ньютона

3.1. Описание метода Гаусса-Ньютона и общие оценки сходимости. В этом параграфе мы опишем метод Гаусса-Ньютона для минимизации функционала (1). Напомним основную идею метода. Пусть нужно решить переопределённую систему нелинейных уравнений

$$f(x) = 0, \tag{4}$$

где $f : \mathbb{R}^n \to \mathbb{R}^m$, $m > n$. Решением назовём вектор x, который минимизирует функционал

$$||f(x)||^2 \to \min,$$

т.е. решение понимается в смысле наименьших квадратов. Линеаризуя (4) относительно некоторой точки x (как в обычном методе Ньютона), получаем

$$f(x) + Df(x)\triangle x = 0.$$

Решаем получившуюся переопределённую систему линейных уравнений

$$\triangle x = -Df(x)^{\dagger} f, \tag{5}$$

где Df — якобиан отображения, а через $A^{\dagger}$ мы обозначаем матрицу, псевдообратную к A. Выражение (5) и есть формула одного шага метода Гаусса-Ньютона. Значение, получающееся после k итераций метода Ньютона при начальном значении x, будем обозначать $\Phi_k(x)$. Локальные свойства сходимости метода (5) демонстрируют две следующие теоремы, приведенные в работе [1].

Теорема 1. *Пусть ζ такая, что $f(\zeta) = 0$, $Df(\zeta)$ инъективна (т.е. в нашем конечномерном случае имеет полный столбцовый ранг) и x такое, что*

$$v = ||x - \zeta||\gamma_1(f, \zeta) \leqslant \frac{3 - \sqrt{7}}{2}.$$

Тогда итерации метода Ньютона $x_k = \Phi_k(x)$ сходятся к ζ квадратично:

$$||x_k - \zeta|| \leqslant \left(\frac{1}{2}\right)^{2^k - 1} ||x - \zeta||,$$

где γ_1 *определяется как*

$$\gamma_1(f,x) = \alpha_1(f,x)\beta_1(f,x),$$
$$\alpha_1(f,x) = ||(Df(x))^\dagger||\ ||f(x)||,$$
$$\beta_1(f,x) = \sup_{k\geqslant 2}\left(||(Df(x))^\dagger||\ \left\|\frac{D^k f(x)}{k!}\right\|\right)^{\frac{1}{k-1}}.$$

Теорема 2. *Пусть* x *и* ζ *удовлетворяют* $Df(\zeta)^\dagger f(\zeta) = 0$, $Df(\zeta)$ *инъективна и*

$$v = ||x-\zeta||\gamma_1(f,\zeta) \leqslant 1 - \frac{\sqrt{2}}{2},$$

тогда для итераций метода Ньютона $x_k = \Phi_k(x)$ *верна оценка*

$$||x_k - \zeta|| \leqslant \lambda^k ||x-\zeta||,$$

где

$$\lambda = \frac{v + \sqrt{2}(2-v)\alpha_1(f,\zeta)}{1-4v+2v^2}.$$

Для сходимости необходимо выполнение условия $\lambda < 1$, *что достигается при*

$$\alpha_1(f,\zeta) \leqslant \frac{1}{2\sqrt{2}}.$$

Теоремы 1 и 2 полностью характеризуют локальную сходимость метода. Если решение системы (4) существует, то сохраняется квадратичная скорость сходимости обычного метода Ньютона. Если же равенства (4) выполняются лишь с некоторой точностью ε, то скорость сходимости становится линейной, однако показатель сходимости пропорционален ε. Отметим, что важнейшей характеристикой, влияющей на скорость сходимости, является величина $||Df^\dagger||$.

3.2. Метод Гаусса-Ньютона в нашем случае. Рассмотрим теперь метод Гаусса-Ньютона для минимизации функционала (4). После линеаризации получаем

$$||\mathcal{R}_{ijk} - \sum_{\alpha=1}^{r}(\triangle u_{i\alpha}v_{j\alpha}w_{k\alpha} + u_{i\alpha}\triangle v_{j\alpha}w_{k\alpha} + u_{i\alpha}v_{j\alpha}\triangle w_{k\alpha})||^2 \to \min, \quad (6)$$

где введена невязка

$$\mathcal{R}_{ijk} = a_{ijk} - \sum_{\alpha=1}^{r} u_{i\alpha}v_{j\alpha}w_{k\alpha}.$$

В отличие от метода ALS, система (6) уже не распадается на отдельные системы, т.к. переменные $\triangle U, \triangle V, \triangle W$ являются связанными друг с другом. Матрица системы (обозначим её опять через Φ) является структурированной и имеет блочный вид

$$\Phi = [\Phi_1, \Phi_2, \Phi_3], \tag{7}$$

где элементы матриц Φ_1, Φ_2, Φ_3 задаются как

$$(\Phi_1)_{(ijk)(i'\alpha)} = \delta_{ii'} v_{j\alpha} w_{k\alpha},$$

$$(\Phi_2)_{(ijk)(j'\alpha)} = \delta_{jj'} u_{i\alpha} w_{k\alpha},$$

$$(\Phi_2)_{(ijk)(k'\alpha)} = \delta_{kk'} u_{i\alpha} v_{j\alpha}.$$

В этих выражениях Φ_1, Φ_2, Φ_3 были пронумерованы тремя индексами (i, j, k) а столбцы — парами $(i'\alpha)$, $(j'\alpha)$, $(k'\alpha)$ соответственно. Величина $\delta_{ii'}$ равна единице при $i = i'$ и нулю в противном случае.

Вопрос о том, как использовать такую структуру для вычисления нужных нам матричных разложений (например, QR-разложения) остаётся открытым. Возможно использование итерационных методов, так как произведения Φ, Φ^T на векторы можно вычислять эффективно. Однако в этом случае потребуется использовать некоторое предобусловливание. Сложность прямого метода с использованием QR–разложения составляет $\mathcal{O}(n_1 n_2 n_3 (n_1 + n_2 + n_3) r^2)$ операций. Если же мы будем использовать метод нормальных уравнений, для которого необходимо вычислять матрицу Грама $\Gamma = \Phi^T \Phi$, то мы можем уменьшить вычислительную сложность. Система нормальных уравнений имеет вид

$$\begin{aligned}
\sum_\alpha \triangle u_{i\alpha}(v_\alpha, v_\beta)(w_\alpha, w_\beta) + \sum_\alpha u_{i\alpha}(\triangle v_\alpha, v_\beta)(w_\alpha, w_\beta) = \\
= \sum_{jk} \mathcal{R}_{ijk}\, v_{j\beta} w_{k\beta}, \\
\sum_\alpha \triangle v_{j\alpha}(w_\alpha, w_\beta)(u_\alpha, u_\beta) + \sum_\alpha v_{j\alpha}(\triangle w_\alpha, w_\beta)(v_\alpha, v_\beta) = \\
= \sum_{ki} \mathcal{R}_{ijk}\, w_{k\beta} u_{i\beta}, \\
\sum_\alpha \triangle w_{k\alpha}(u_\alpha, u_\beta)(v_\alpha, v_\beta) + \sum_\alpha w_{k\alpha}(\triangle u_\alpha, u_\beta)(v_\alpha, v_\beta) = \\
= \sum_{ij} \mathcal{R}_{ijk}\, u_{i\beta} v_{j\beta}.
\end{aligned} \tag{8}$$

Введём «матрицы моментов» X, Y, Z и матрицы Грама $\Gamma_u, \Gamma_v, \Gamma_w$

$$X = (\triangle u_\alpha, u_\beta), \quad Y = (\triangle v_\alpha, v_\beta), \quad Z = (\triangle w_\alpha, w_\beta),$$

$$\Gamma_u = U^T U = (u_\alpha, u_\beta),$$

$$\Gamma_v = V^T V = (v_\alpha, v_\beta),$$

$$\Gamma_w = W^T W = (w_\alpha, w_\beta).$$

Домножая равенства (8) на $u_{i\alpha}, v_{j\alpha}, w_{k\alpha}$ и суммируя по i, j, k соответственно, получим следующие уравнения для матриц моментов X, Y, Z:

$$(\Gamma_v \circ \Gamma_w)X + (\Gamma_w \circ Y^T + \Gamma_v \circ Z^T)\Gamma_u = Q_1, \tag{9}$$

$$(\Gamma_w \circ \Gamma_u)Y + (\Gamma_u \circ Z^T + \Gamma_w \circ X^T)\Gamma_v = Q_2, \tag{10}$$

$$(\Gamma_u \circ \Gamma_v)Z + (\Gamma_v \circ X^T + \Gamma_u \circ Y^T)\Gamma_w = Q_3. \tag{11}$$

где

$$Q_1 = \sum_{ijk} \mathcal{R}_{ijk}\, u_{i\alpha} v_{j\beta} w_{k\beta},$$

$$Q_2 = \sum_{ijk} \mathcal{R}_{ijk}\, u_{i\beta} v_{j\alpha} w_{k\beta},$$

$$Q_3 = \sum_{ijk} \mathcal{R}_{ijk}\, u_{i\beta} v_{j\beta} w_{k\alpha},$$

или в матрично-векторной форме

$$M(U, V, W)\triangle = q. \tag{12}$$

Число неизвестных теперь стало равным $3r^2$, поэтому для решения системы (9-11) прямым методом требуется $\mathcal{O}(r^6)$ арифметических операций. Важно отметить, что система (9) структурированная, и можно показать, что матрица системы разрежена. Вычисление произведения матрицы системы на вектор требует $\mathcal{O}(r^3)$ операций, поэтому разумно использовать итерационный метод, например метод сопряжённых градиентов. Однако для устойчивой сходимости опять необходимо построить предобусловливатель.

3.3. Сходимость. Анализ сходимости проведём в частном, но довольно часто встречающемся случае *абсолютно симметричного тензора,* т.е.

$$a_{ijk} = a_{P(i,j,k)},$$

где $P(i,j,k)$ — любая перестановка индексов (i,j,k). В этом случае решения (1) обладают свойством $U = V = W$ и $n_1 = n_2 = n_3 = n$. Для анализа сходимости мы должны определить, при выполнении каких условий матрица $\Phi = Df(u)$ (7) имеет полный столбцовый ранг, и как оценить $||\Phi^{\dagger}||$? Ответ на эти вопросы дает следующая теорема.

Теорема 3. *Пусть матрица* $U = [u_{i\alpha}] \in \mathbb{R}^{n\times r}$ *имеет полный столбцовый ранг. Тогда якобиан* $Df(u)$ *имеет полный столбцовый ранг* nr*, матрица* $M(u)$ *из (12) невырождена и имеет место оценка*

$$\sigma_{\min}(Df(u)) \geqslant \sigma_{\min}(U)^2. \tag{13}$$

Доказательство.

Рассмотрим произведение матрицы $Df(u)$ на вектор $\triangle u$

$$h = (Df(u)\triangle u)_{ijk} = \sum_{\alpha=1}^{r} \triangle u_{i\alpha}u_{j\alpha}u_{k\alpha} + u_{i\alpha}\triangle u_{j\alpha}u_{k\alpha} + u_{i\alpha}u_{j\alpha}\triangle u_{k\alpha}.$$

Так как матрица U имеет полный столбцовый ранг, то существует такая невырожденная матрица $\Psi \in \mathbb{R}^{n\times n}$, что

$$\sum_{i'=1}^{n} u_{i'\alpha}\Psi_{ii'} = \delta_{i\alpha}, \quad i = 1,...,n, \quad \alpha = 1,...,r,$$

и при этом

$$||\Psi|| \leqslant \frac{1}{\sigma_{\min}(U)}.$$

Определим вектор $\widehat{h}$ как

$$\widehat{h} = \sum_{i'j'k'} h_{i'j'k'}\Psi_{ii'}\Psi_{jj'}\Psi_{kk'}.$$

Тогда

$$\widehat{h} = Df(I_r)\triangle\widehat{u},$$

где через I_r обозначена матрица размера n на r с элементами $\delta_{i\alpha}$, а

$$\triangle \widehat{u} = \Psi \triangle u.$$

Матрица $Df(I_r)$ имеет полный столбцовый ранг. Её сингулярные числа есть корни квадратные из собственных чисел матрицы $M(I_r) = Df(I_r)^T Df(I_r)$. Легко видеть, что матрица $M(I_r)$ является диагональной с элементами на диагонали, равными единице или тройке. Поэтому

$$||\widehat{h}|| \geqslant ||\triangle \widehat{u}|| \geqslant ||\Psi||\ ||\triangle u||.$$

Так как

$$\widehat{h} = \sum_{i'j'k'} h_{i'j'k'} \Psi_{ii'} \Psi_{jj'} \Psi_{kk'},$$

то

$$||\widehat{h}|| \leqslant ||\Psi||^3\ ||h||,$$

поэтому

$$||h|| = ||Df(u) \triangle u|| \geqslant \frac{||\triangle u||}{||\Psi||^2},$$

откуда следует, что $Df(u)$ имеет полный столбцовый ранг, и справедлива оценка

$$\sigma_{\min}(Df(u)) \geqslant \sigma_{\min}(U)^2.$$

Используя теоремы 1, 2 и 3, получаем верхнюю оценку на показатель скорости сходимости λ:

$$\lambda = \mathcal{O}(\varepsilon\ \mathrm{cond}(U)^2).$$

Надо отметить, что эта оценка является довольно грубой. В частности, матрица U может иметь линейно-зависимые столбцы, а якобиан всё равно будет невырожденным.

4. Метод Ньютона с подбором шага

4.1. Описание метода. Мы также использовали для решения задачи (1) *демпфированный метод Ньютона с подбором шага методом Левенберга-Марквадта.* Демпфированный метод Ньютона является гибридом метода наискорейшего спуска и метода Ньютона. Напомним, что в методе наискорейшего спуска для минимизации функционала $F(x)$ шаги выбираются вдоль направления антиградиента

$$h_{\mathrm{sd}} = -F'(x),$$

а в методе Ньютона шаги определяются из решения уравнения

$$(F''(x))h_{\mathrm{n}} = -F'(x).$$

Метод наискорейшего спуска обладает линейной сходимостью, метод Ньютона — квадратичной, но только в некоторой окрестности стационарной точки. Таким образом, метод наискорейшего спуска имеет смысл использовать на начальной, а метод Ньютона — на конечной стадии вычислений, когда x близок к искомому x^*. Существуют различные методы «умного» переключения между этими двумя методами, одним из которых является демпфированный метод Ньютона. В нем шаг итерации определяется из решения уравнения

$$(F''(x) + \mu I)h = -F'(x),$$

и поведение алгоритма на текущей итерации определяется значением параметра μ. Конкретнее, при очень больших значениях μ направление шага h практически совпадает с направлением h_{sd} :

$$h \approx -\frac{1}{\mu}F'(x),$$

при очень малых — с направлением h_{n}.

Поскольку параметр μ определяет не только направление, но и длину шага, становится возможным предложить метод, не требующий отдельной процедуры выбора длины шага. Существенным, однако, является вопрос о выборе μ в течение итерационного процесса. В методе Левенберга-Марквадта изменение величины μ определяется тем, насколько хорошо поведение функции $F(x)$ на текущем шаге может быть приближено с помощью *квадратичной модели*

$$L(h) = F(x) + F'(x)h + \frac{1}{2}h^T F''(x)h.$$

Конкретнее, на каждом шаге вычисляется *коэффициент выгоды*

$$\rho = \frac{F(x) - F(x+h)}{L(0) - L(h)}.$$

Если ρ достаточно велик (близок к единице), это означает, что квадратичная модель $L(h)$ достаточно близка к значению $F(x+h)$, а стало быть, в этой области хорошо применим метод Ньютона и значение μ можно уменьшить, чтобы шаг следующей итерации был более близок к шагу h_{n}. Если значение ρ невелико или вообще отрицательно, то квадратичная модель плохо описывает поведение функции и

необходимо увеличить значение μ, с тем чтобы, во-первых, сделать шаг более близким к шагу метода наискорейшего спуска h_{sd}, а во-вторых, уменьшить его длину.

Изначально стратегия модернизации μ была предложена Марквадтом в следующем виде

если $\rho < 0.25$

$\mu := 2 \cdot \mu$;

иначе, если $\rho > 0.75$

$\mu := \frac{1}{3} \cdot \mu$.

Метод оказался достаточно гибким к небольшим изменениям порогов 0.25 и 0.75. Однако из-за разрывного поведения критерия Марквадта на практике иногда наблюдались колебания, замедляющие сходимость. Более гладкий критерий был предложен в [2]:

если $\rho > 0$

$\mu := \mu \cdot \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$;

иначе

$\mu := 2 \cdot \mu$.

Сравнить две описанные стратегии обновления помогает рис. 1.

В окончательном виде метод Ньютона с выбором шага по критерию Левенберга-Марквадта выглядит так:

Алгоритм (Newton).

Установить $k = 0,\ x = x_0,\ A = F''(x),\ g = F'(x)$;

$done = .\mathrm{false}.$

пока $(k < k_{\max})$ и $(.\mathrm{not}.done)$

установить $k := k + 1$;

решить $(A + \mu I)h = -g$ (разложением Холецкого);

если шаг h достаточно мал

$done = .\mathrm{true}.$

иначе

$$x_{\mathrm{new}} = x + h,\ \rho := \frac{F(x) - F(x_{\mathrm{new}})}{L(0) - L(h)};$$

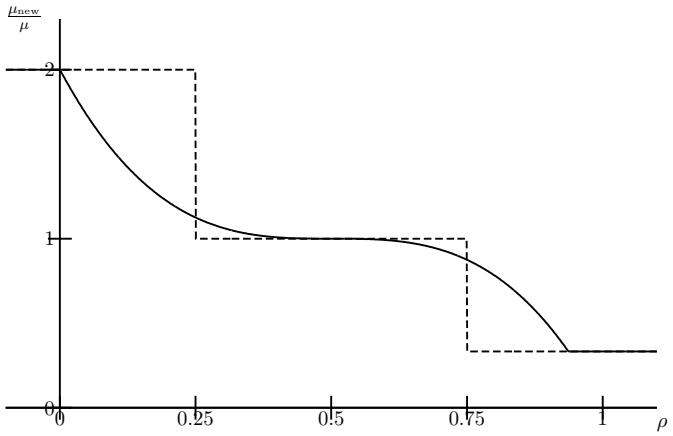


Рис. 1. Стратегии обновления μ : Марквадта (пунктирная линия), сглаженная (сплошная линия)

если $\rho > 0$
  $x := x_{\text{new}}$, $A := F''(x)$, $g := F'(x)$;
  если g достаточно мал
    $done = .\text{true.}$
  $\mu := \mu \cdot \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$;
иначе
  $\mu := 2\mu$

Этот метод, как и метод наискорейшего спуска, является вполне робастным, и в случае, если $F''(x^*)$ невырождена, его сходимость на финальном участке квадратичная. Сложность метода $\mathcal{O}(n^3)$ на одну итерацию (разложение Холецкого).

4.2. Функционал отклонения и его производные. Для решения задачи (1) проведем минимизацию функционала

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

Градиент F' выражается в виде

$$\frac{\partial F}{\partial u_{i\alpha}} = \sum_{jk} \mathcal{R}_{ijk}\, v_{j\alpha} w_{k\alpha},$$

$$\frac{\partial F}{\partial v_{j\alpha}} = \sum_{ki} \mathcal{R}_{ijk}\, w_{k\alpha} u_{i\alpha},$$

$$\frac{\partial F}{\partial w_{k\alpha}} = \sum_{ij} \mathcal{R}_{ijk}\, u_{i\alpha} v_{j\alpha},$$

где $\mathcal{R}_{ijk} = \sum_{\beta} u_{i\beta} v_{j\beta} w_{k\beta} - a_{ijk}$.

Гессиан F'' определяется формулами

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial u_{i\beta}} = \sum_{jk} v_{j\alpha} v_{j\beta} w_{k\alpha} w_{k\beta} = (v_\alpha, v_\beta)(w_\alpha, w_\beta),$$

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial u_{i'\beta}} = 0, \qquad i \neq i';$$

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial v_{j\beta}} = u_{i\beta} v_{j\alpha}(w_\alpha, w_\beta) + \delta_{\alpha\beta} \sum_k \mathcal{R}_{ijk}\, w_{k\alpha},$$

где $\delta_{\alpha\beta} = 1$ при $\alpha = \beta$ и $\delta_{\alpha\beta} = 0$ при $\alpha \neq \beta$.

Таким образом:

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial u_{i'\beta}} = \delta_{ii'}(v_\alpha, v_\beta)(w_\alpha, w_\beta),$$

$$\frac{\partial^2 F}{\partial v_{j\alpha} \partial v_{j'\beta}} = \delta_{jj'}(u_\alpha, u_\beta)(w_\alpha, w_\beta),$$

$$\frac{\partial^2 F}{\partial w_{k\alpha} \partial w_{k'\beta}} = \delta_{kk'}(u_\alpha, u_\beta)(v_\alpha, v_\beta),$$

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial v_{j\beta}} = u_{i\beta} v_{j\alpha}(w_\alpha, w_\beta) + \delta_{\alpha\beta} \sum_k \mathcal{R}_{ijk}\, w_{k\alpha},$$

$$\frac{\partial^2 F}{\partial u_{i\alpha} \partial w_{k\beta}} = u_{i\beta} w_{k\alpha}(v_\alpha, v_\beta) + \delta_{\alpha\beta} \sum_j \mathcal{R}_{ijk}\, v_{j\alpha},$$

$$\frac{\partial^2 F}{\partial v_{j\alpha} \partial w_{k\beta}} = v_{j\beta} w_{k\alpha}(u_\alpha, u_\beta) + \delta_{\alpha\beta} \sum_i \mathcal{R}_{ijk}\, u_{i\alpha}.$$

5. Численные эксперименты

При сравнении различных минимизационных алгоритмов нас в первую очередь интересует скорость убывания целевой функции $F(x)$. Рассматриваемые нами методы обладают весьма разной асимптотикой сложности, поэтому использовать для их сравнения убывание $F(x)$ после совершения одного и того же числа итераций было бы некорректно. Реальное время исполнения алгоритма — это тоже не очень хороший параметр для исследования, поскольку оно может зависеть от особенностей реализации алгоритма на конкретной платформе и меняться в десятки раз при использовании специализированных векторизованных библиотек и задействовании опций оптимизации компилятора. Однако, в конечном итоге все же именно реальное расчетное время определяет применимость того или иного метода, поэтому мы остановили свой выбор на том, чтобы сравнивать уменьшение целевой функции (нормы отклонения), которое различные методы произвели за одинаковое *время счета.*

Рассмотрев графики 1 и 2, демонстрирующие убывание невязки в ходе работы минимизационных методов на типовых для нас примерах массивов, можно дать следующие (не претендующие на глобальность) комментарии.

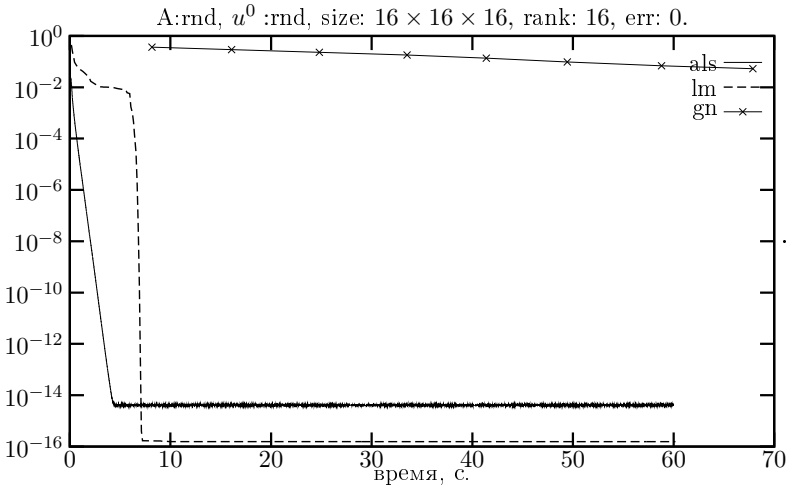
(а) Случайный массив размера $16 \times 16 \times 16$ и точного ранга 16. Случайное начальное приближение.
Метод Ньютона дает более качественное разложение, чем метод переменных направлений, однако более медленное. Метод Гаусса-Ньютона не успевает заметно уменьшить невязку.

(б) Случайный массив размера $16 \times 16 \times 16$ и ранга 16. Наложен мультипликативный шум порядка 10^{-5}. Случайное начальное приближение.
Метод Ньютона сходится к тому же решению, что и метод переменных направлений, но второй метод делает это быстрее. Метод Гаусса-Ньютона сходится слишком медленно.

(в) Ядро разложения Таккера размером $23 \times 23 \times 23$. Исходный массив имел элементы порядка $1/r$, где $r \sim \sqrt{i^2 + j^2 + k^2}$. Ищется трилинейное разложение ранга 23. Случайное начальное приближение.
Метод Ньютона проигрывает методу переменных направлений как по скорости, так и по точности. Метод Гаусса-Ньютона заметно проигрывает им обоим.

График 1

(а) Случайный массив точного ранга



(б) Случайный массив с наложенным шумом

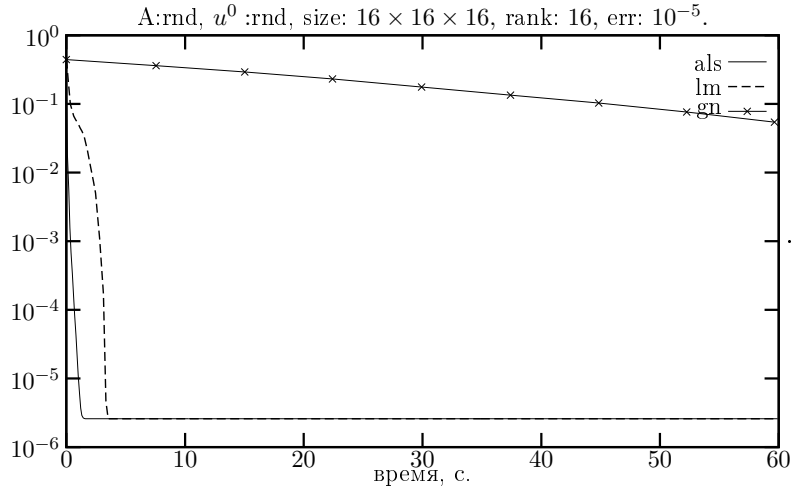
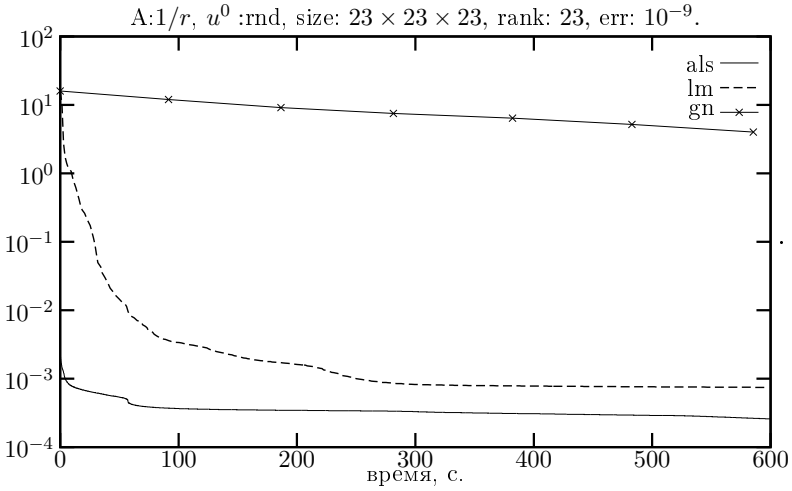
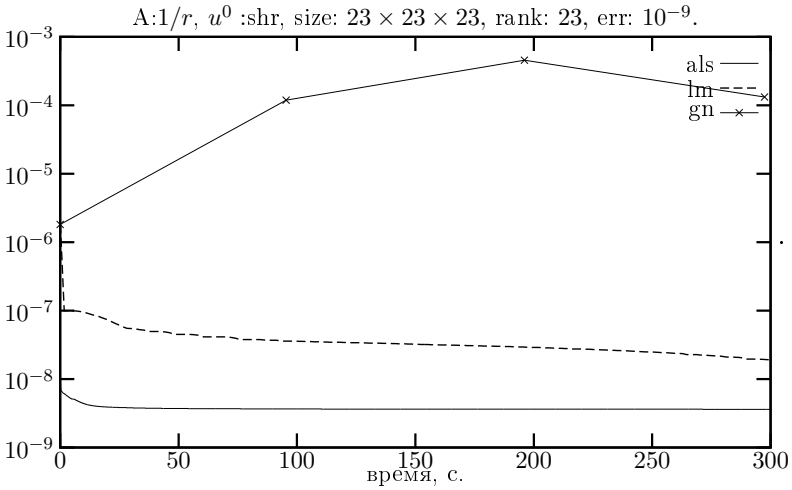


График 2

(в) Ядро разложения Таккера, образ $1/r$



(д) То же, с хорошим начальным приближением



(д) Ядро разложения Таккера размером $23 \times 23 \times 23$. Исходный массив имел элементы порядка $1/r$. Ищется трилинейное разложение ранга 23. Начальное приближение — суперобобщенное разложение Шура [4].
Метод Ньютона уступает методу переменных направлений. Метод Гаусса-Ньютона ухудшает начальную невязку и заметно проигрывает им обоим.

Таким образом, самый простой в реализации метод переменных направлений оказывается вполне хорошо применимым на практике. Усовершенствование методов типа Ньютона возможно с применением методов доверительных областей.

Список литературы

[1] Dedieu, J.P., Schub M. Newton's method for overdetermined systems of equations // *Math. Comp.* 2000. V. 69 (281). P. 1099-1115

[2] Nielsen H. B. Damping parameter in Marquardt's Method — IMM, DTU, 1999.
`http://www.imm.dtu.dk/~hbn/publ/TR9905.ps`

[3] Оселедец И.В., Савостьянов Д.В. Методы разложения тензора. // *Настоящий сборник.* С. 51-64.

[4] Оселедец И.В., Савостьянов Д.В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Настоящий сборник.* С. 101-116.

Тензорные ранги сверхбольших трехмерных матриц[a]

И. В. Оселедец, Д. В. Савостьянов

Рассмотрены основные типы трехмерных массивов, возникающих при решении сингулярных и гиперсингулярных интегральных уравнений в трехмерном пространстве. Исследована асимптотическая зависимость их тензорных рангов от размера, дано сравнение с теоретическими оценками.

1. Введение

При решении интегральных уравнений в трехмерном пространстве возникают линейные системы с матрицами очень больших размеров. Например, дискретизация простейшего уравнения

$$\int\limits_{\mathcal{V}} \frac{u(y)}{|x-y|} dy = f(x), \qquad x \in \mathcal{V},$$

на сетке $n \times n \times n$ в единичном кубе приведет к системе с матрицей A размера $n^3 \times n^3$. В специальных случаях, например если сетка регулярна по всем трем направлениям, матрица обладает специальной структурой (является теплицевой), что позволяет хранить ее в $\mathcal{O}(n^3)$ ячейках памяти и умножать на вектор за $\mathcal{O}(n^3 \log n)$ арифметических действий. Однако в общем случае затраты на хранение и умножение такой матрицы на вектор оцениваются как n^6, что накладывает существенные ограничения на практическое применение таких алгоритмов либо делает их чрезвычайно требовательными к вычислительным ресурсам. Более того, даже требования

[a]Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 04-07-90336 и 05-01-00721) и программы фундаментальных исследований Отделения математических наук РАН «Вычислительные и информационные проблемы решения больших задач» по проекту «Матричные методы в интегральных и дифференциальных уравнениях» .

«эффективных» алгоритмов, имеющих асимптотику $\mathcal{O}(n^3)$, могут оказаться очень существенными уже при достаточно умеренных n, скажем, порядка тысячи.

Одним из активно развивающихся в настоящий момент подходов к решению таких линейных систем является построение для исходной матрицы A аппроксимации $\tilde{A}$ в виде суммы *тензорных произведений* [1, 2, 3]

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

где матрицы U, V, W имеют размер $n \times n$. Если такое приближение для матрицы A с умеренным значением ранга r существует, то можно заменить исходную задачу решением системы с матрицей $\tilde{A}$, что позволяет значительно уменьшить затраты памяти (в [3] показано, что тензорные аппроксимации дают сверхлинейное сжатие данных) и времени ее решения.

Задачу построения тензорного разложения можно свести к задаче построения *трилинейного* (или *канонического*) разложения для трехмерного массива $\mathcal{A} = [a_{ijk}]$, полученного переупорядочением и перенумерацией элементов исходной матрицы A (см., напр., [3, 8]). Трилинейным разложением называется представление массива $\mathcal{A}$ в виде

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

Общие методы отыскания такого разложения описаны в [7]. В работе [8] предложен быстрый метод построения такой аппроксимации, являющийся обобщением на трехмерный случай метода *крестовой аппроксимации* [4, 5] и имеющий почти линейную асимптотику сложности по n (так называется сложность порядка $n \log^\gamma n$ действий). По заданному массиву $\mathcal{A}$, а точнее, лишь по малой части его элементов, трехмерный крестовый метод строит его разложение с достаточно небольшим значением r. Однако принципиальным является вопрос — действительно ли при фиксированной погрешности аппроксимации полученное разложение имеет минимальный среди всех возможных (или близкий к нему) ранг. Особенно важным этот вопрос становится при больших размерах массива. Переформулируем его в следующем виде.

Вопрос.

Как значение ранга трехмерного массива ведет себя асимптотически при увеличении его размера? Согласуется ли предсказан-

ная асимптотика для ранга с асимптотикой рангов разложений, практически вычисляемых трехмерным крестовым методом?

Ответу на этот вопрос для модельных массивов, соответствующих матрицам, порожденным сингулярными и гиперсингулярными интегральными уравнениями, и посвящена эта статья.

2. Теоретические оценки

В работе [3] теоретически обосновано наличие у матриц, порожденных асимптотически гладкими функциями $a_{ij} = F(\mathbf{x}_i - \mathbf{y}_j)$, тензорного разложения и приведены оценки на ранг и точность такого разложения в зависимости от свойств асимптотической гладкости порождающей функции. Оценки представлены в виде

$$\begin{aligned} r &\leqslant \left(c_0 + c_1 \log h^{-1}\right) p^{m-1} + \tau, \\ |\{A - \tilde{A}_r\}_{ij}| &\leqslant c_2 \gamma^p \|\mathbf{x}_i - \mathbf{y}_j\|^g. \end{aligned} \tag{1}$$

Здесь $m = 3$ — размерность пространства, параметры c_0, c_1, c_2 и τ зависят от размерности пространства и сеток $\mathbf{x}_i, \mathbf{y}_j$, h — минимальное расстояние от узлов сетки до сингулярности, величина γ меньше 1, параметр g описывает асимптотическую гладкость функции F :

$$|D^{\mathbf{p}} F(\mathbf{v})| \leqslant c d^p p! \|v\|^{g-p}, \qquad \forall p \geqslant 0. \tag{2}$$

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

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

Рассматривая второе из уравнений (1) как оценку абсолютной погрешности аппроксимации

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

и учитывая что

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

перепишем его в виде оценки на относительную точность аппроксимации

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

поскольку именно в терминах относительной погрешности сформулирован критерий точности в практически используемом алгоритме.

Отсюда заключаем $p \sim c_3 \log \varepsilon^{-1} + c_4$, 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 $h^{-1} \sim n$, таким образом, оценка на асимптотическое поведение ранга может быть выписана в виде

$$r \leqslant c \log n \log^2 \varepsilon^{-1}. \tag{3}$$

Экспериментальной проверке этой асимптотической оценки мы посвятим следующий раздел.

3. Практические значения ранга

Мы рассматривали массивы $\mathcal{A} = [a_{ijk}]$ следующих типов:

a) $a_{ijk} = \dfrac{1}{i + j + k}$,

b) $a_{ijk} = \dfrac{1}{\sqrt{i^2 + j^2 + k^2}} = \dfrac{1}{\rho}$,

c) $a_{ijk} = \dfrac{1}{(i^2 + j^2 + k^2)} = \dfrac{1}{\rho^2}$,

d) $a_{ijk} = \dfrac{1}{(i^2 + j^2 + k^2)^{3/2}} = \dfrac{1}{\rho^3}$.

Значения индексов i, j, k всегда изменялись в пределе $1 \leqslant i, j, k \leqslant n$, величину n мы в дальнейшем будем называть *размером* трехмерного массива.

Вычисления происходили следующим образом.

- Сначала с помощью трехмерного метода крестовой аппроксимации [8] мы получали *разложение Таккера* (см. [6, 7, 8]) для этих массивов.
- Затем к получившемуся ядру разложения Таккера применялись «матричные» [7, 9] методы поиска трилинейного разложения, ранг которого совпадает с размером ядра. Обычно такие методы дают хорошее начальное приближение. В случае, если это не так, мы увеличивали ранг искомого трилинейного разложения на 1, то есть искали *переопределенное разложение* [10]. Этого оказывалось достаточно, чтобы получить начальное приближение с хорошей точностью.

- Затем использовалась комбинация «минимизационных» [7, 11] методов поиска трилинейного разложения, являющихся весьма эффективными при наличии хорошего начального приближения. По трилинейному разложению ядра трилинейное разложение всего массива строится элементарно.

Графики 1-2, иллюстрируют связь ранга получившегося разложения и размера исходного массива. Зависимость ранга r от $\log_2 n$, которую они демонстрируют, *линейна,* причем для некоторых массивов на больших размерах эта зависимость становится еще более слабой (см., например, график для массива d, точность $\varepsilon = 10^{-4}$ или 10^{-5}).

Графики 3-4 иллюстрируют связь ранга получившегося разложения и точности аппроксимации. Из их вида не вполне очевидно, является ли описываемая ими зависимость квадратичной или линейной, то есть выполняется ли на практике теоретическая оценка $r \sim \log^2 \varepsilon^{-1}$ или же она может быть снижена (по крайней мере, для рассматриваемых массивов) до $r \sim \log \varepsilon^{-1}$.

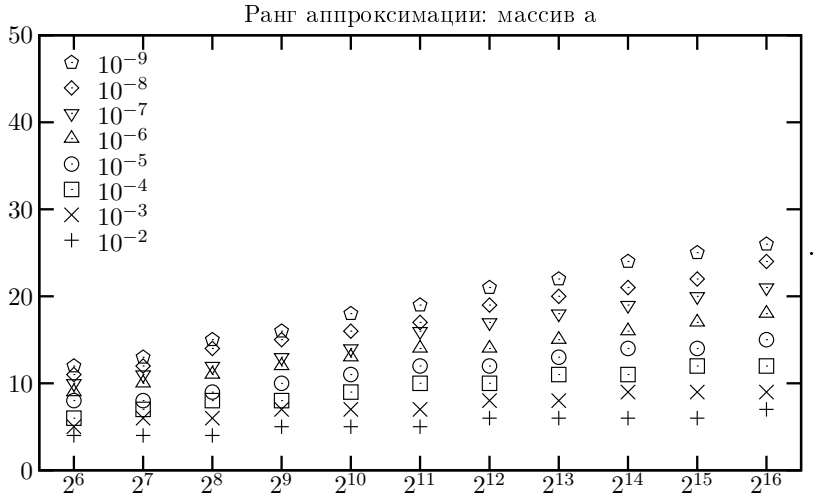
Чтобы прояснить ответ на этот вопрос, изобразим эти же зависимости на отдельных графиках. Графики 5-8 содержат зависимости ранга от размера массива, где для каждой отдельно взятой точности ε показана прямая, наилучшим образом (в смысле задачи наименьших квадратов), приближающая указанное множество точек.

Графики 9-12 содержат зависимости ранга от точности, где для каждого отдельно взятого размера n показаны прямая и парабола, наилучшим образом приближающие указанное множество точек. Их рассмотрение дает понять, что старший коэффициент в экспериментальной зависимости полученного ранга точности аппроксимации весьма невелик (некоторым исключением является, возможно, график для массива d размера $n = 2^{16}$), и с высокой точностью можно описывать эту зависимость r от $\log n$ как линейную. Таким образом, можно считать экспериментально подтвержденным (для рассматриваемых массивов), что

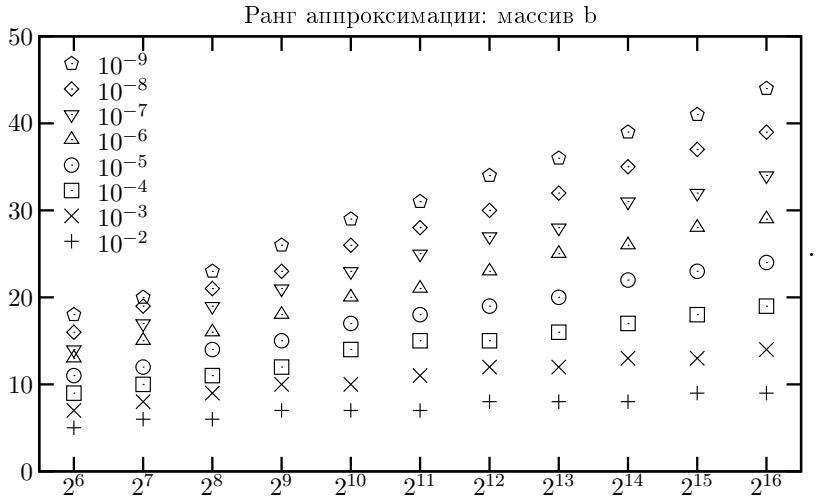
$$r \sim \log n \log \varepsilon^{-1}.$$

График 1

Зависимость ранга массивов от их размера



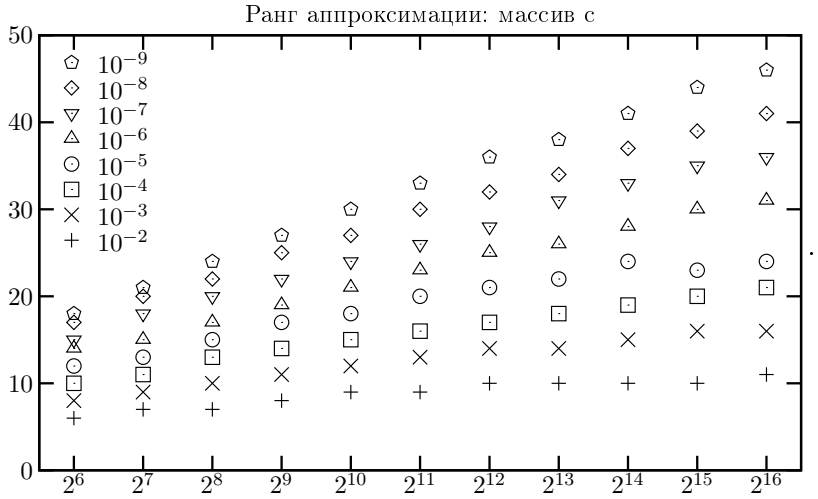
.



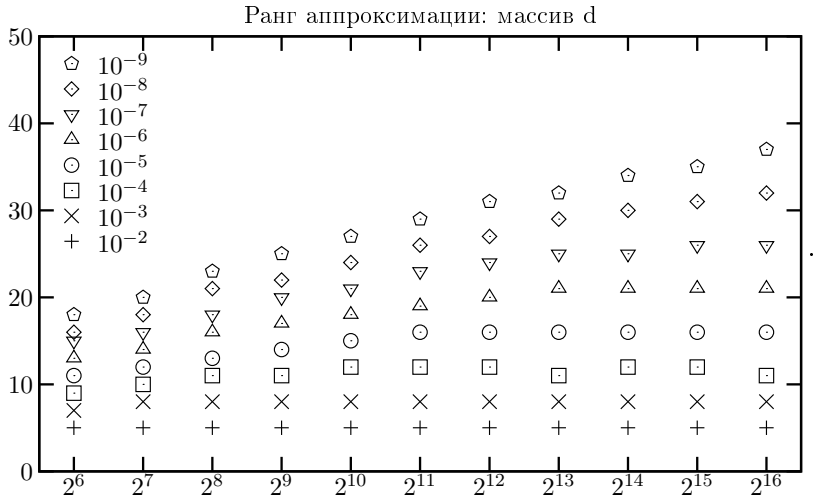
.

График 2

Зависимость ранга массивов от их размера



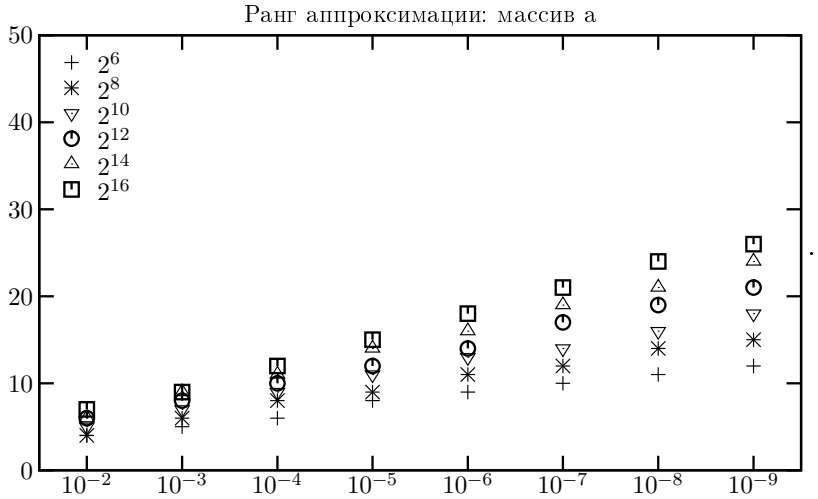
.



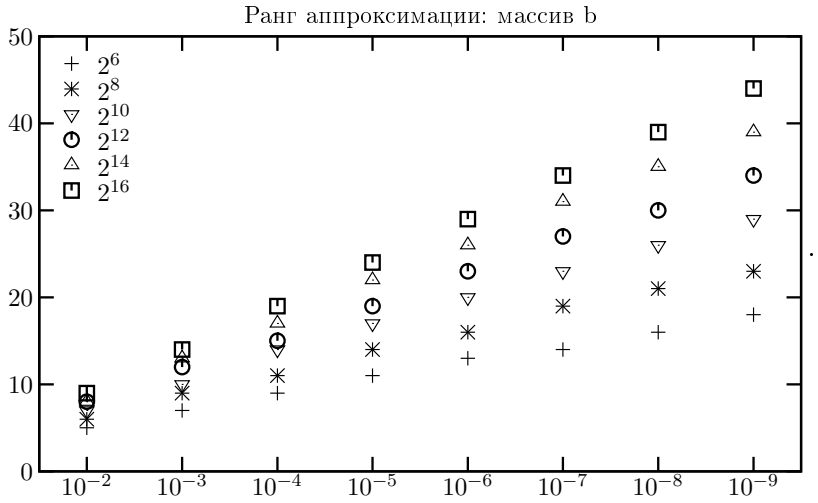
.

График 3

Зависимость ранга массивов от точности



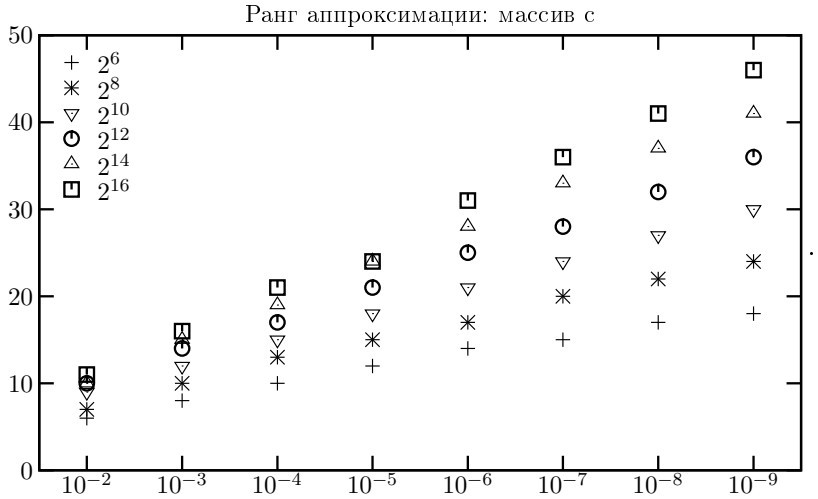
.



.

График 4

Зависимость ранга массивов от точности



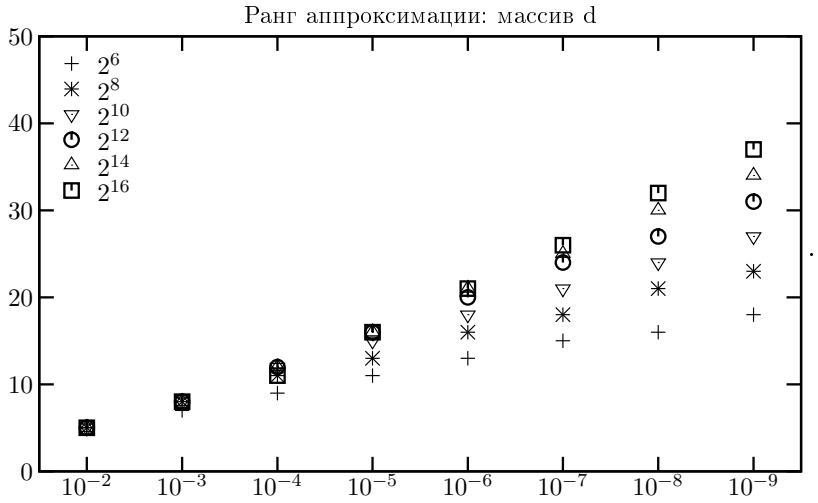


График 5

Зависимость ранга массивов от их размера,
линейная гипотеза, массив a

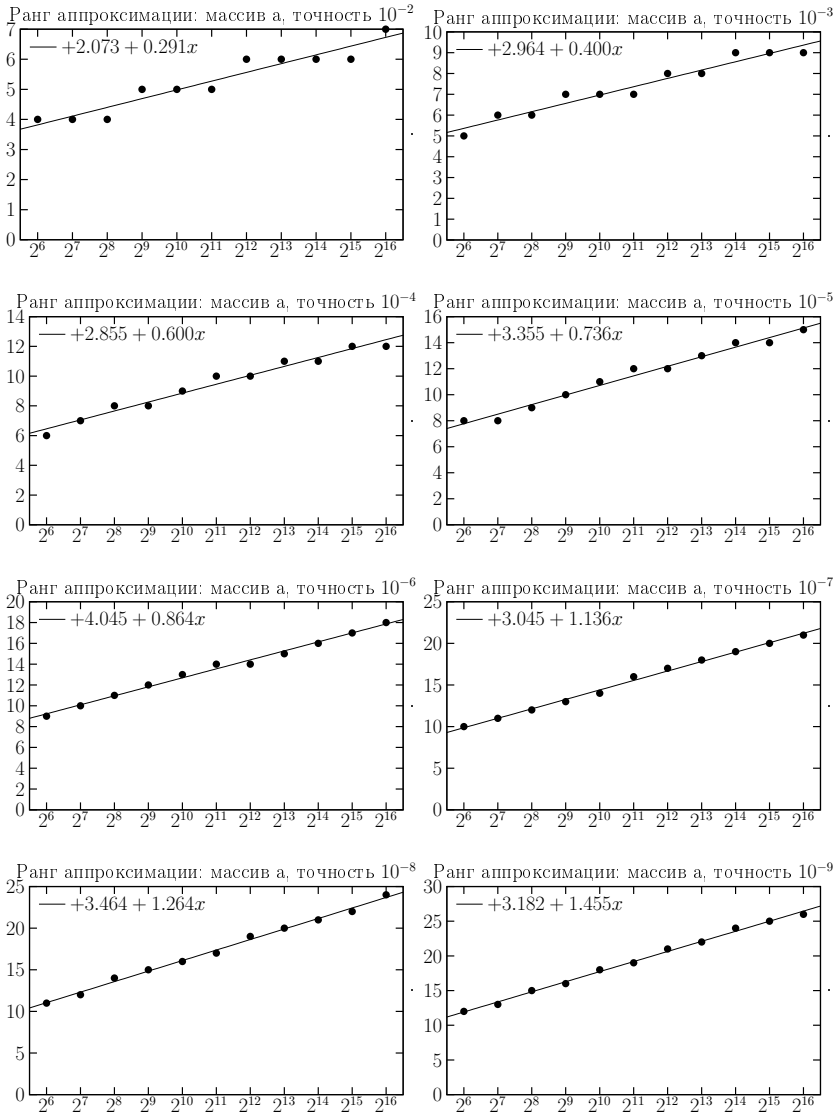


График 6

Зависимость ранга массивов от их размера,
линейная гипотеза, массив b

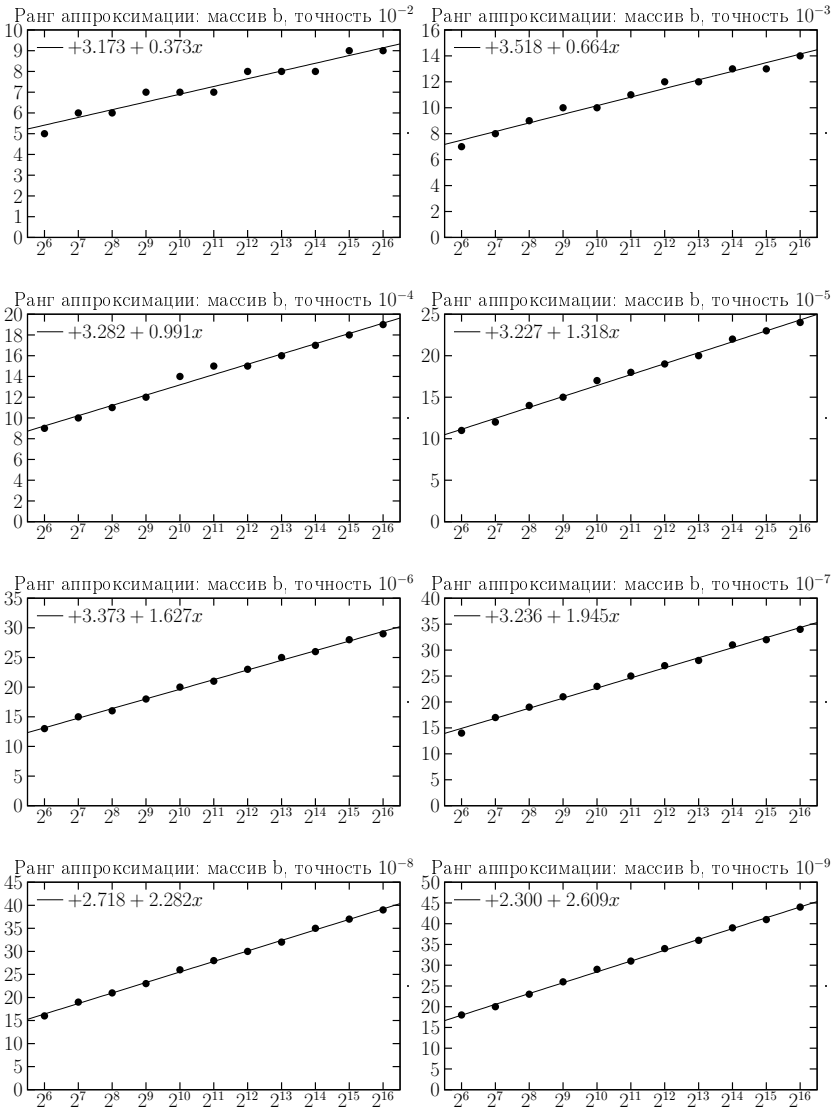


График 7

Зависимость ранга массивов от их размера,
линейная гипотеза, массив c

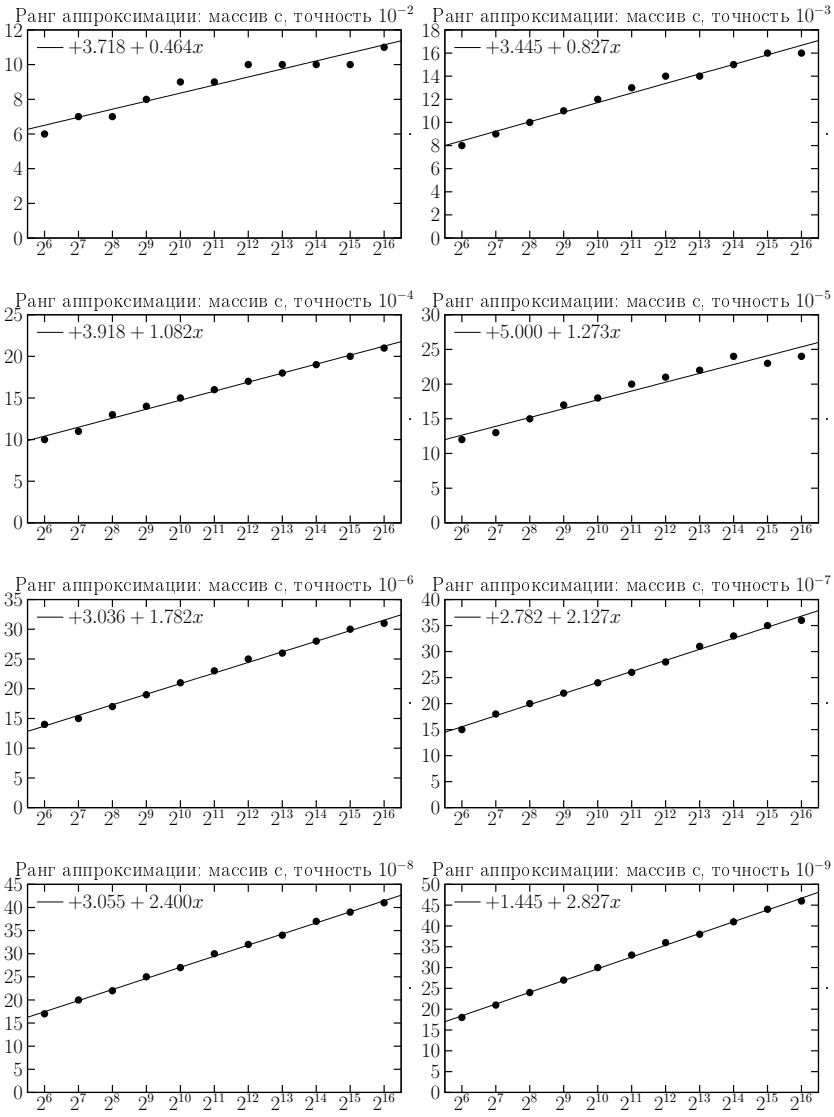


График 8

Зависимость ранга массивов от их размера,
линейная гипотеза, массив d

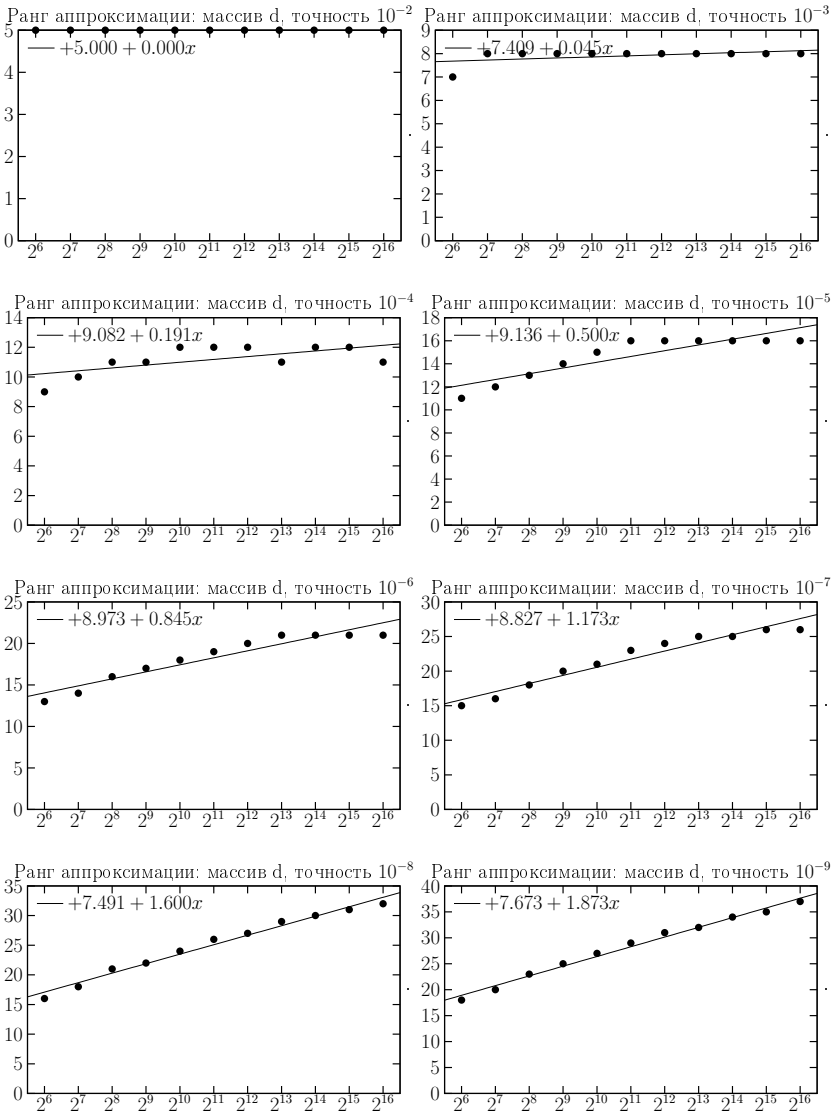


График 9

Зависимость ранга массивов от точности,
линейная и квадратичная гипотезы, массив a

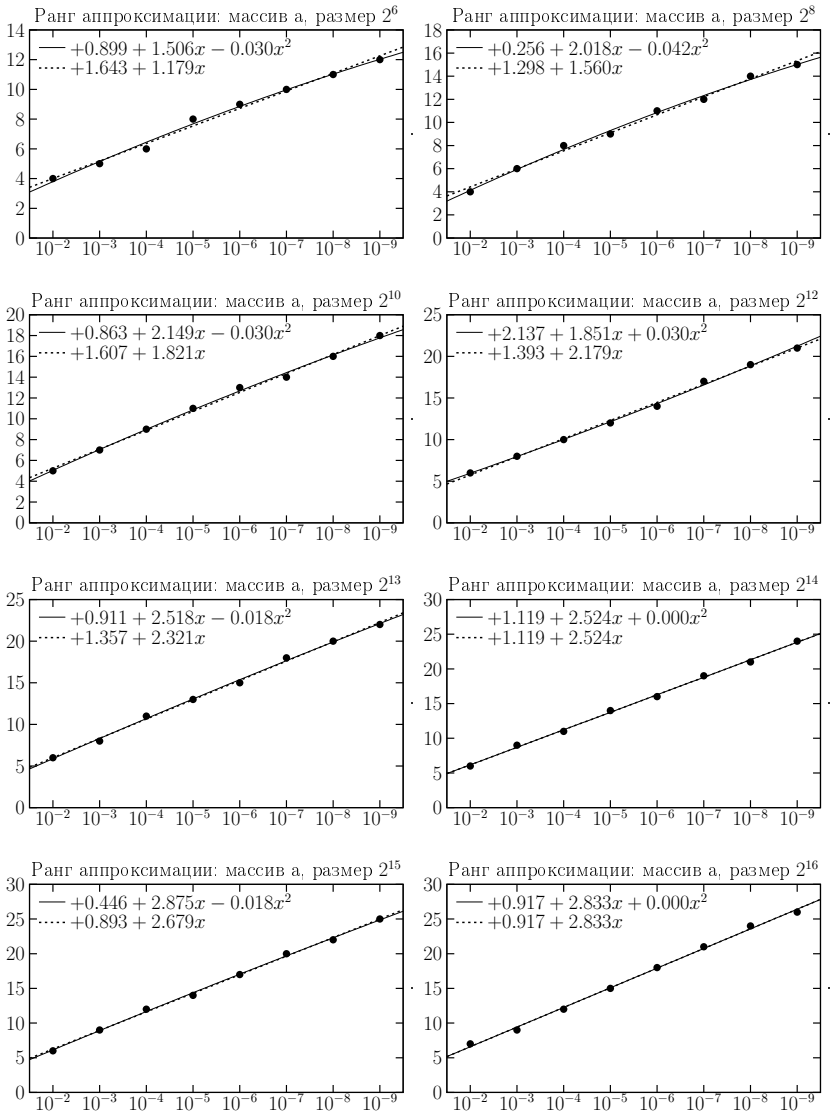


График 10

Зависимость ранга массивов от точности,
линейная и квадратичная гипотезы, массив b

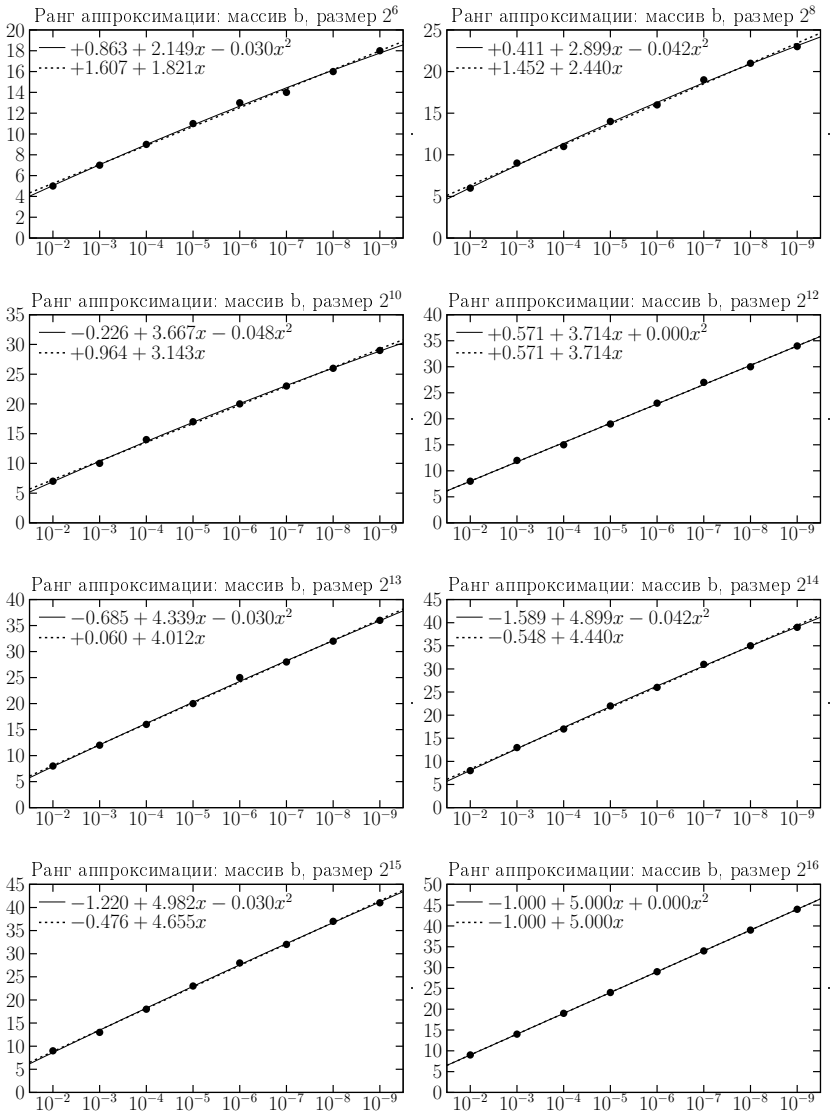


График 11

Зависимость ранга массивов от точности,
линейная и квадратичная гипотезы, массив c

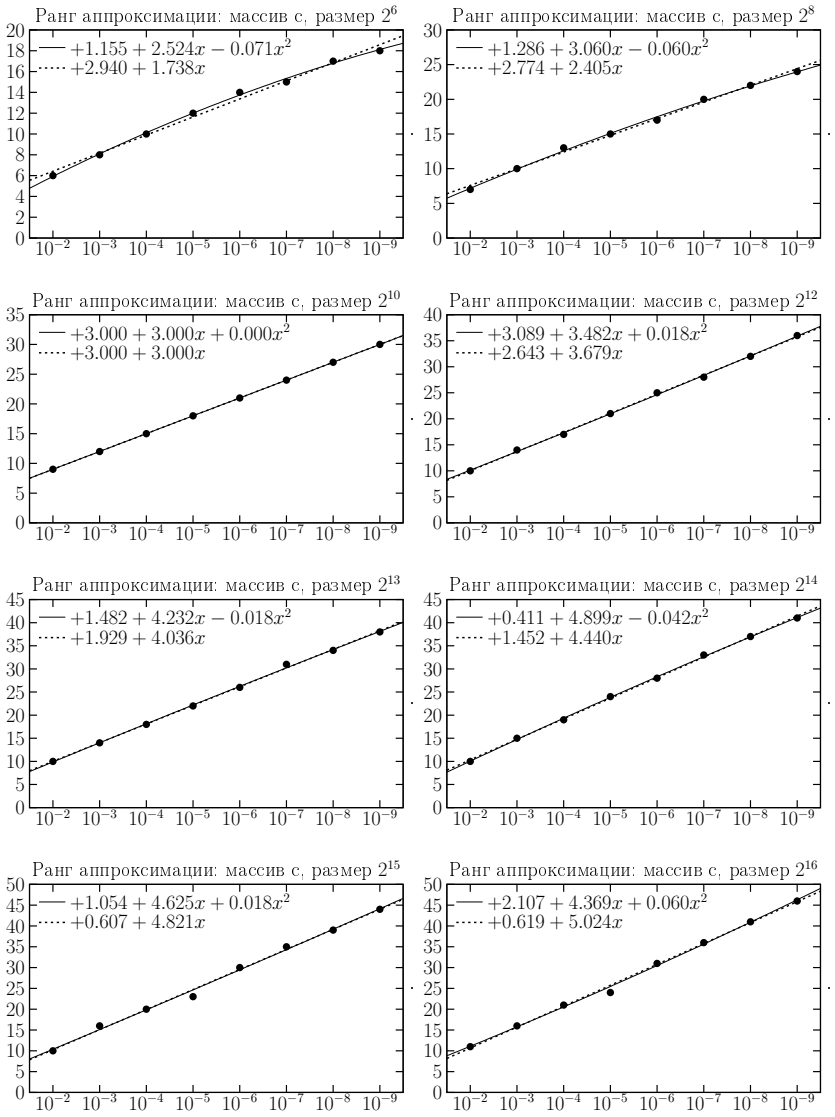
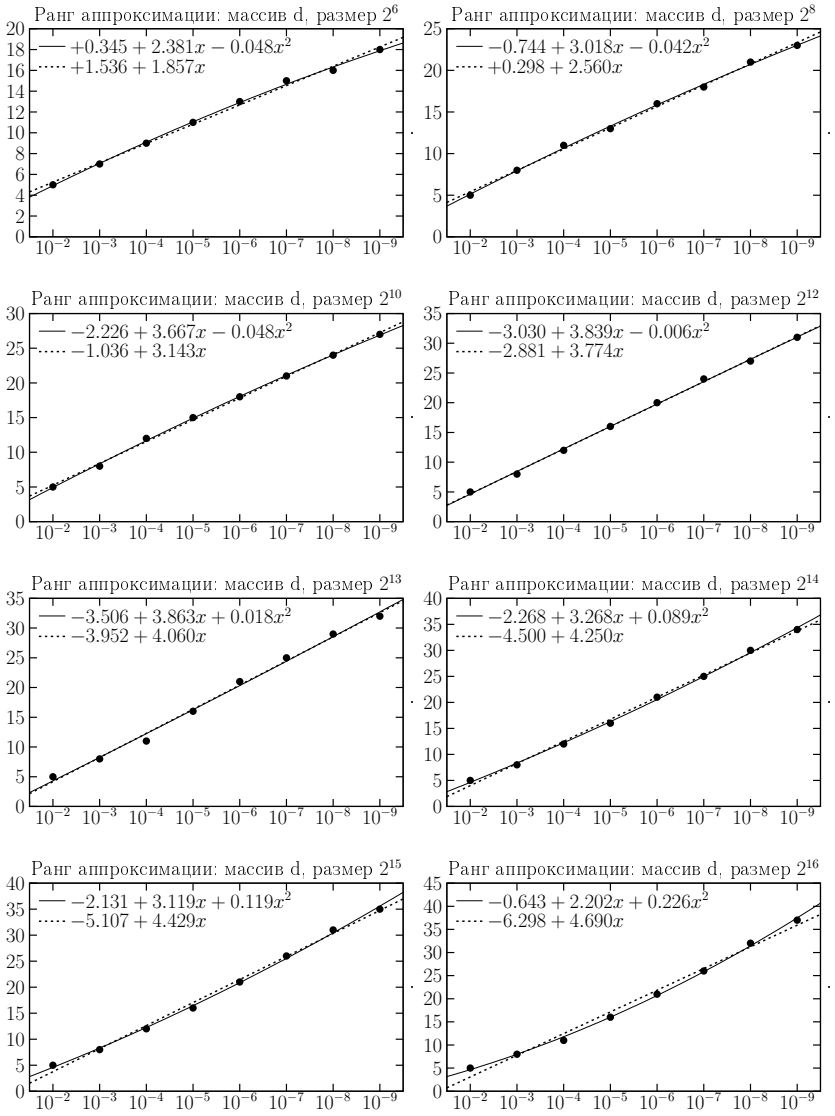


График 12

Зависимость ранга массивов от точности,
линейная и квадратичная гипотезы, массив d



Для массива b снижение асимптотики ранга по ε можно обосновать теоретически, поскольку функция $f(x,y,z) = 1/\sqrt{x^2+y^2+z^2}$ является гармонической. Однако для этого потребуется специальная техника доказательства, отличная от представленной в [3]. Для остальных массивов этот факт является чисто экспериментальным.

4. 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 13-16 показана зависимость времени работы трехмерного крестового метода от размера массива; на графиках 17-20 — 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 $r \sim \log n$.

График 13

Зависимость времени работы алгоритма Cross3D от размера, линейная и логарифмическая ($t \sim n \log^{\gamma} n$) гипотезы, массив a

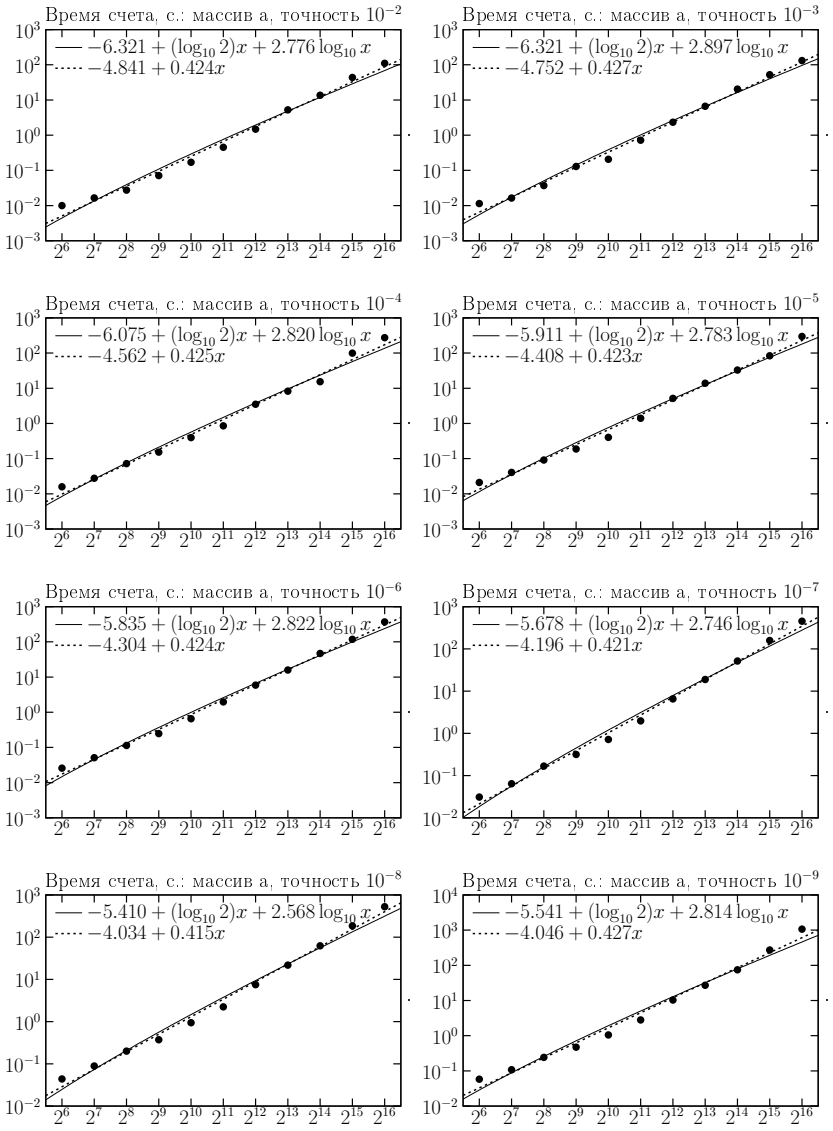


График 14

Зависимость времени работы алгоритма Cross3D от размера, линейная и логарифмическая ($t \sim n \log^{\gamma} n$) гипотезы, массив b

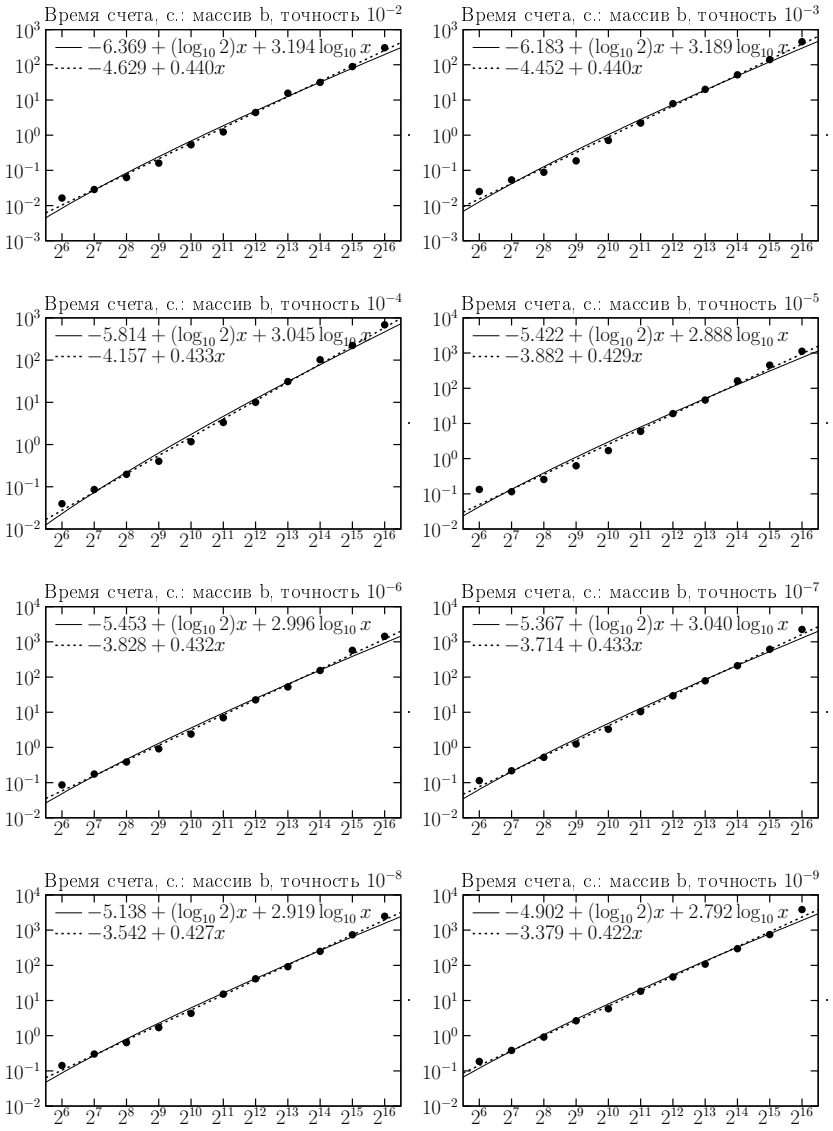


График 15

Зависимость времени работы алгоритма Cross3D от размера, линейная и логарифмическая ($t \sim n \log^{\gamma} n$) гипотезы, массив c

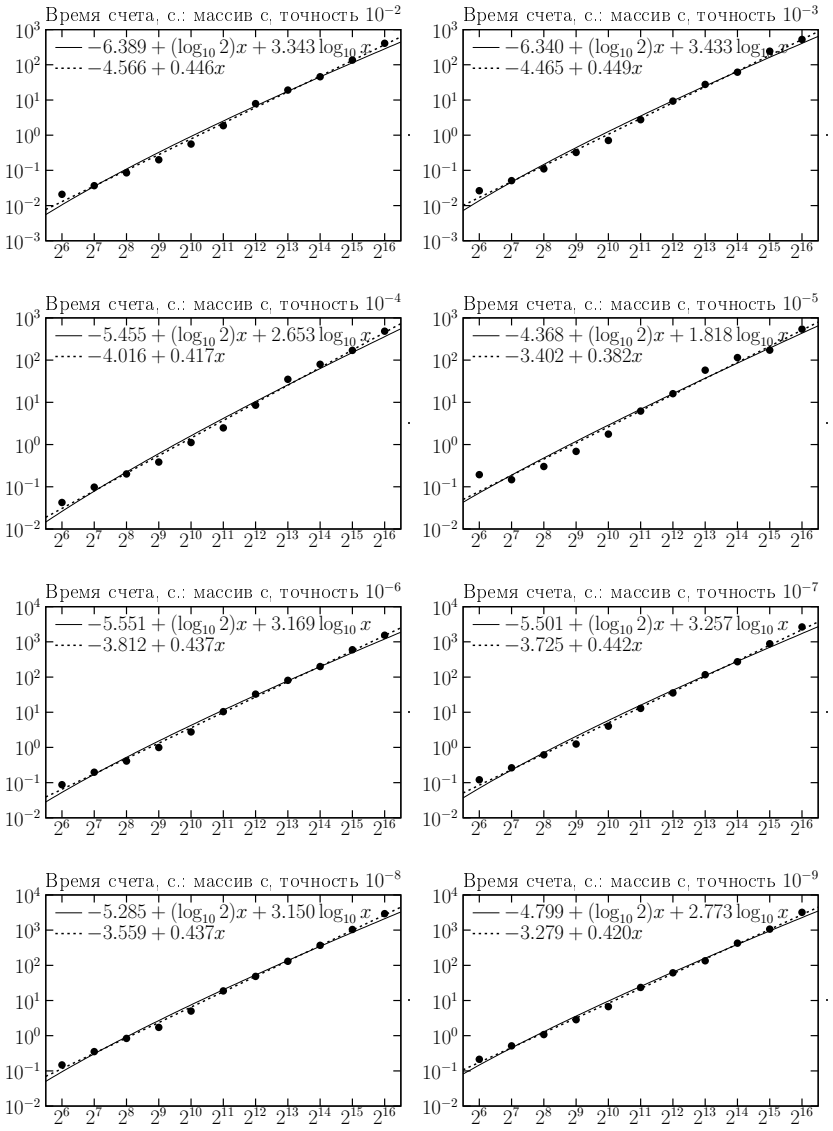


График 16

Зависимость времени работы алгоритма Cross3D от размера, линейная и логарифмическая ($t \sim n \log^\gamma n$) гипотезы, массив d

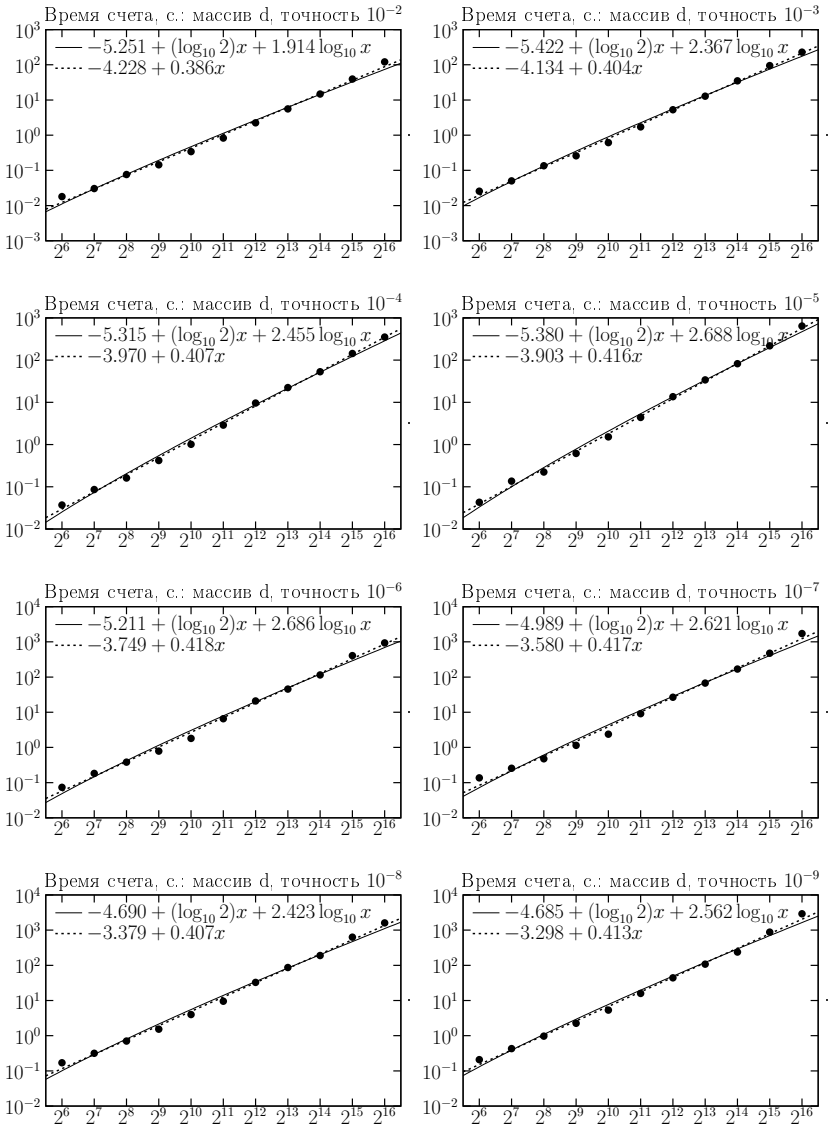


График 17

Зависимость времени работы алгоритма Cross3D от точности, линейная и логарифмическая ($t \sim \log^{\gamma} \varepsilon^{-1}$) гипотезы, массив a

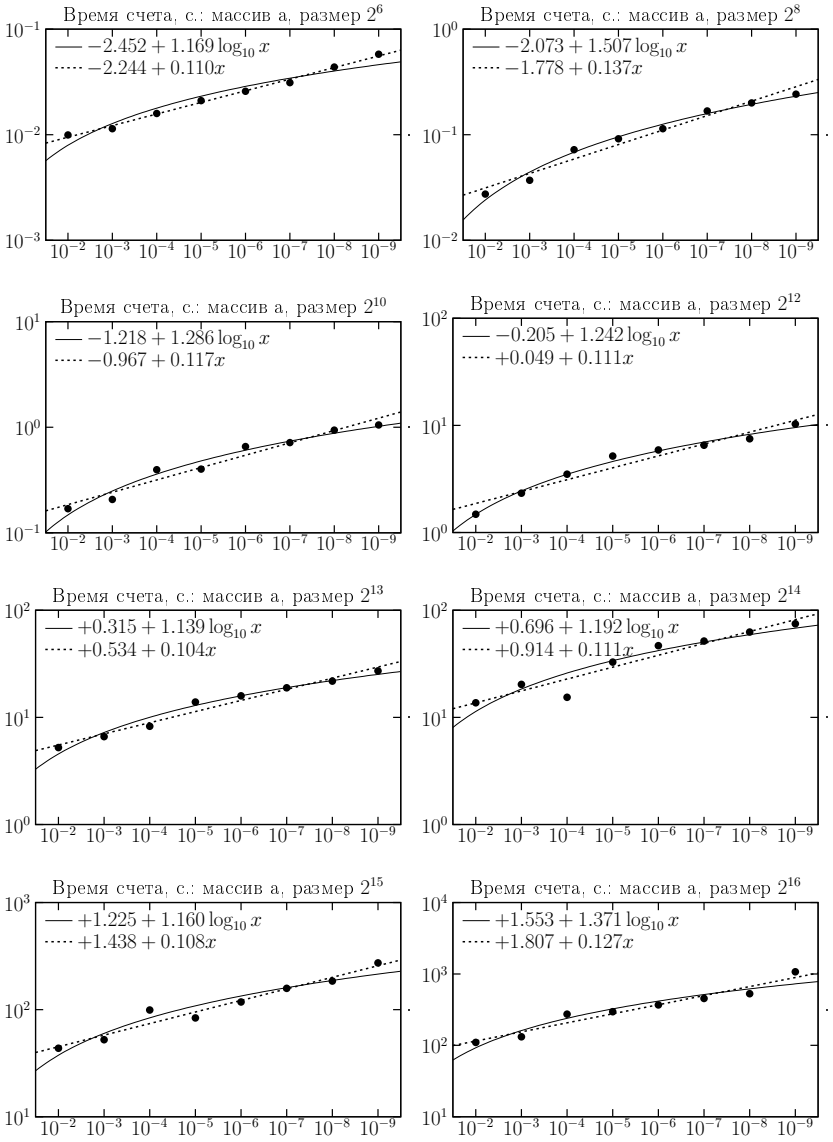


График 18

Зависимость времени работы алгоритма Cross3D от точности, линейная и логарифмическая ($t \sim \log^{\gamma} \varepsilon^{-1}$) гипотезы, массив b

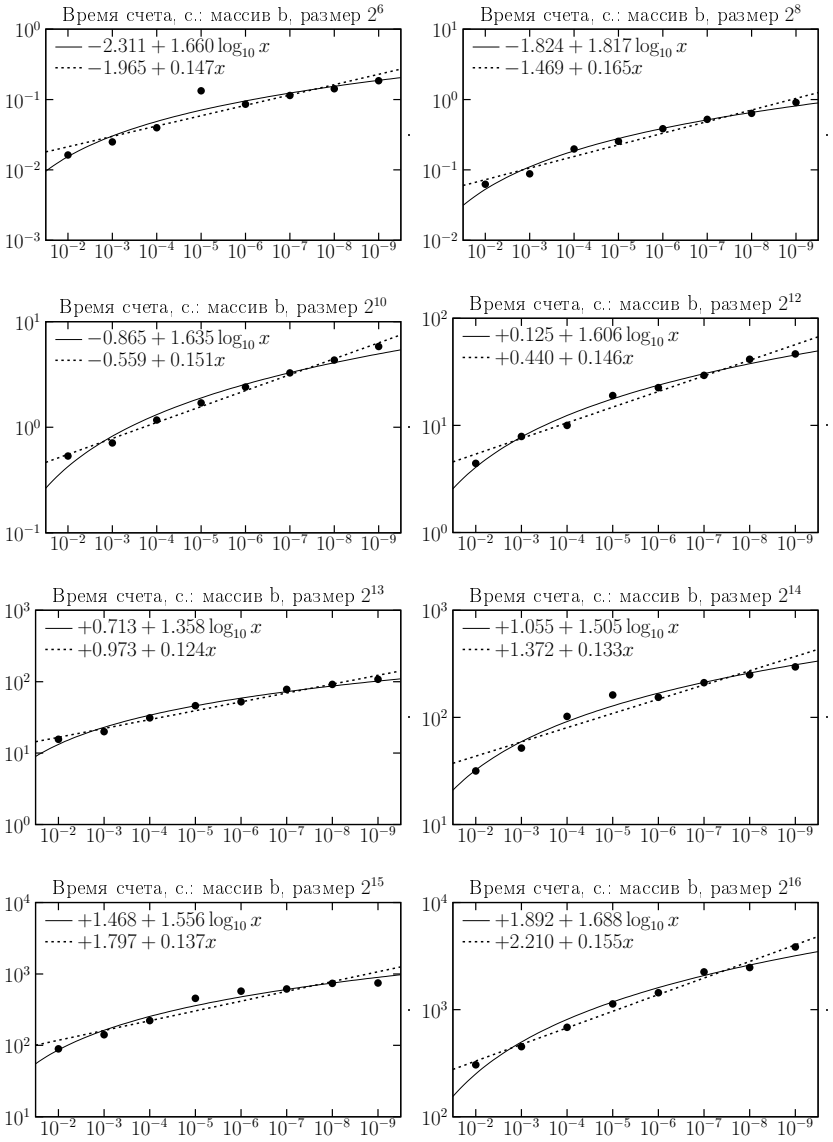


График 19

Зависимость времени работы алгоритма Cross3D от точности, линейная и логарифмическая ($t \sim \log^{\gamma} \varepsilon^{-1}$) гипотезы, массив c

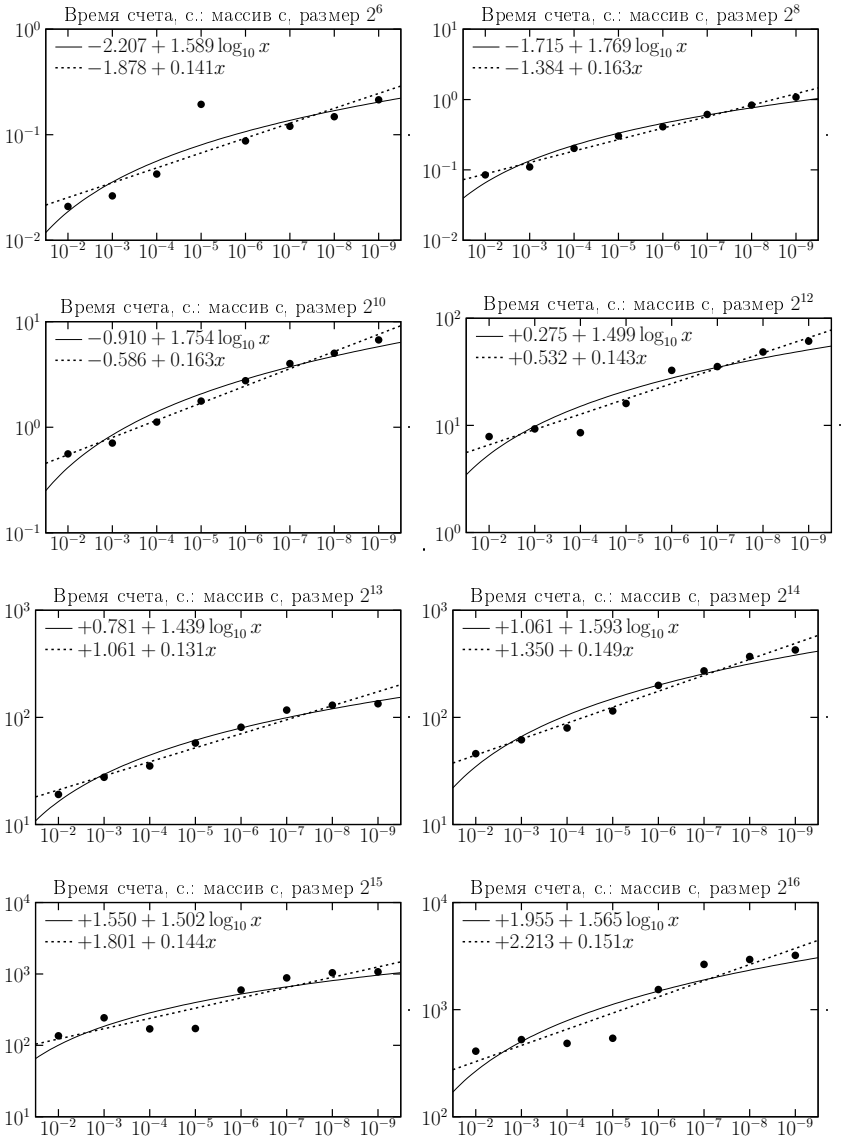
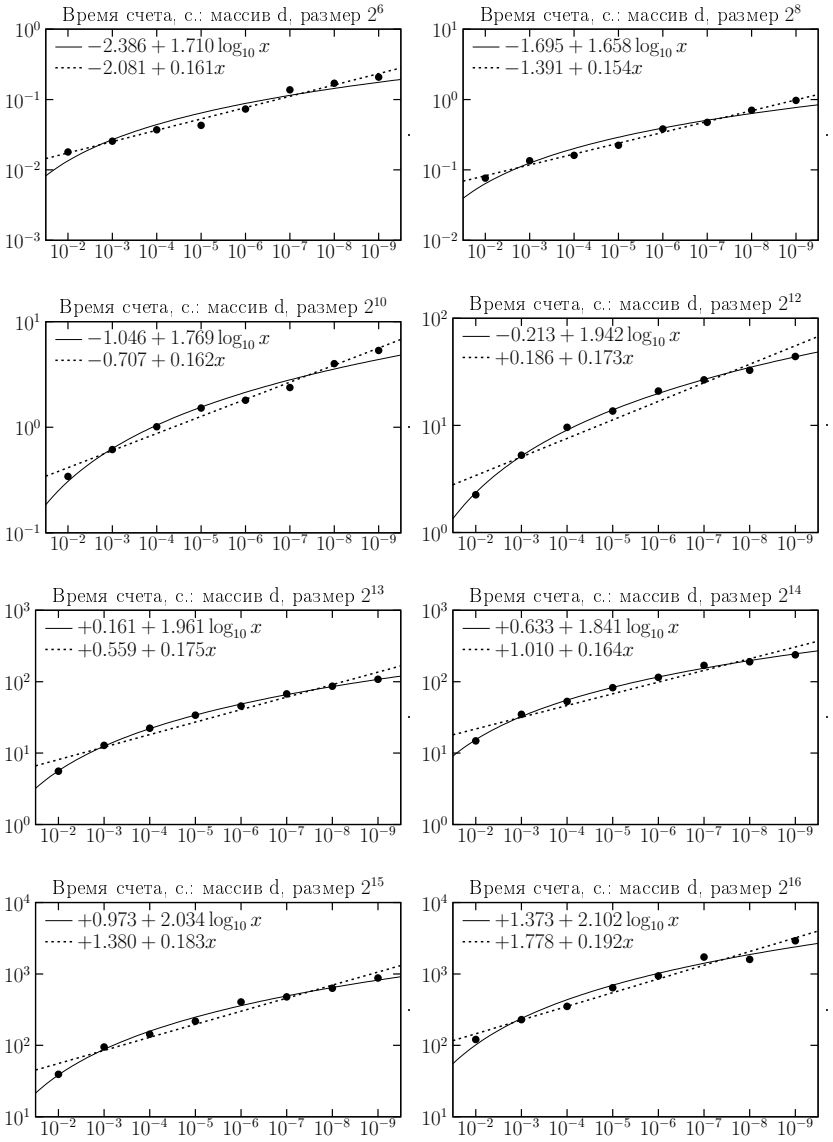


График 20

Зависимость времени работы алгоритма Cross3D от точности, линейная и логарифмическая ($t \sim \log^{\gamma} \varepsilon^{-1}$) гипотезы, массив d



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 $\gamma = 3$.

Заключение

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

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

Это означает, что трехмерный крестовый метод является *очень перспективным* инструментом аппроксимации для больших и сверхбольших трехмерных массивов, а значит и плотных матриц, возникающих в трехмерных задачах на тензорных сетках. При этом время, которое требуется для сжатия массивов размером порядка петабайта (2^{50} байт) до размера порядка сотни мегабайт, составляет около часа на обычной персональной вычислительной машине, и, что еще приятнее, асимптотическое поведение этого времени оценивается почти линейно по размеру массива

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

где коэффициент γ практически тоже не превосходит трех.

Список литературы

[1] Hackbush W., Khoromskij B. N., Tyrtyshnikov E. E. Hierarchical Kronecker tensor-product approximations // *J. Numer. Math.* 2005. V. 13. P. 119-156.

[2] Tyrtyshnikov E. E. Kronecker-product approximations for some function-related matrices // *Linear Algebra Appl.* 2004. V. 379. P. 423-437.

[3] Тыртышников Е. Е. Тензорные аппроксимации матриц, порожденных асимптотически гладкими функциями // *Матем. сб.* 2003. Т. 194, 6. С. 147-160.

[4] Tyrtyshnikov E. E. Mosaic–skeleton approximations // *Calcolo.* 1996. V. 33 (1-2). P. 47-57.

[5] Tyrtyshnikov E. E. Incomplete cross approximation in the mosaic–skeleton method // *Computing.* 2000. V. 4. P. 367-380.

[6] Tucker L. R. Some mathematical notes on three-mode factor analysis. // *Psychometrika.* 1966. V. 31. P. 279-311.

[7] Оселедец И.В., Савостьянов Д.В. Методы разложения тензора // *Настоящий сборник.* С. 51-64.

[8] Оселедец И.В., Савостьянов Д.В. Трехмерный аналог алгоритма крестовой аппроксимации и его эффективная реализация // *Настоящий сборник.* С. 65-99.

[9] Оселедец И.В., Савостьянов Д.В. Быстрый алгоритм для одновременного приведения матриц к треугольному виду и аппроксимации тензоров // *Настоящий сборник.* С. 101-116.

[10] Оселедец И.В., Савостьянов Д.В. Об одном алгоритме построения трилинейного разложения // *Настоящий сборник.* С. 117-130.

[11] Оселедец И.В., Савостьянов Д.В. Минимизационные методы аппроксимации тензоров и их сравнение // *Настоящий сборник.* С. 131-146.

СОДЕРЖАНИЕ

Научное издание

МАТРИЧНЫЕ
МЕТОДЫ И ТЕХНОЛОГИИ
РЕШЕНИЯ БОЛЬШИХ ЗАДАЧ

Сборник научных трудов
под ред. Е. Е. Тыртышникова

Институт вычислительной математики Российской академии наук
119991 Москва, ул. Губкина, д. 8.

Лицензия ИД № 03991 от 12.02.2001.
Оригинал-макет изготовлен в ИВМ РАН
Подписано в печать 14.12.2005 г.
Формат $60 \times 90\,{}^{1}\!/_{16}$. Печать офсетная. Бумага офсетная №1
Печ. л. 11. Тираж 250 экз. Заказ 7784
Отпечатано согласно представленному оригинал-макету
в ФГУП „Производственно-издательский комбинат ВИНИТИ“.
140010 Люберцы Московской обл., Октябрьский пр-т, 403.
Тел. 554-21-86