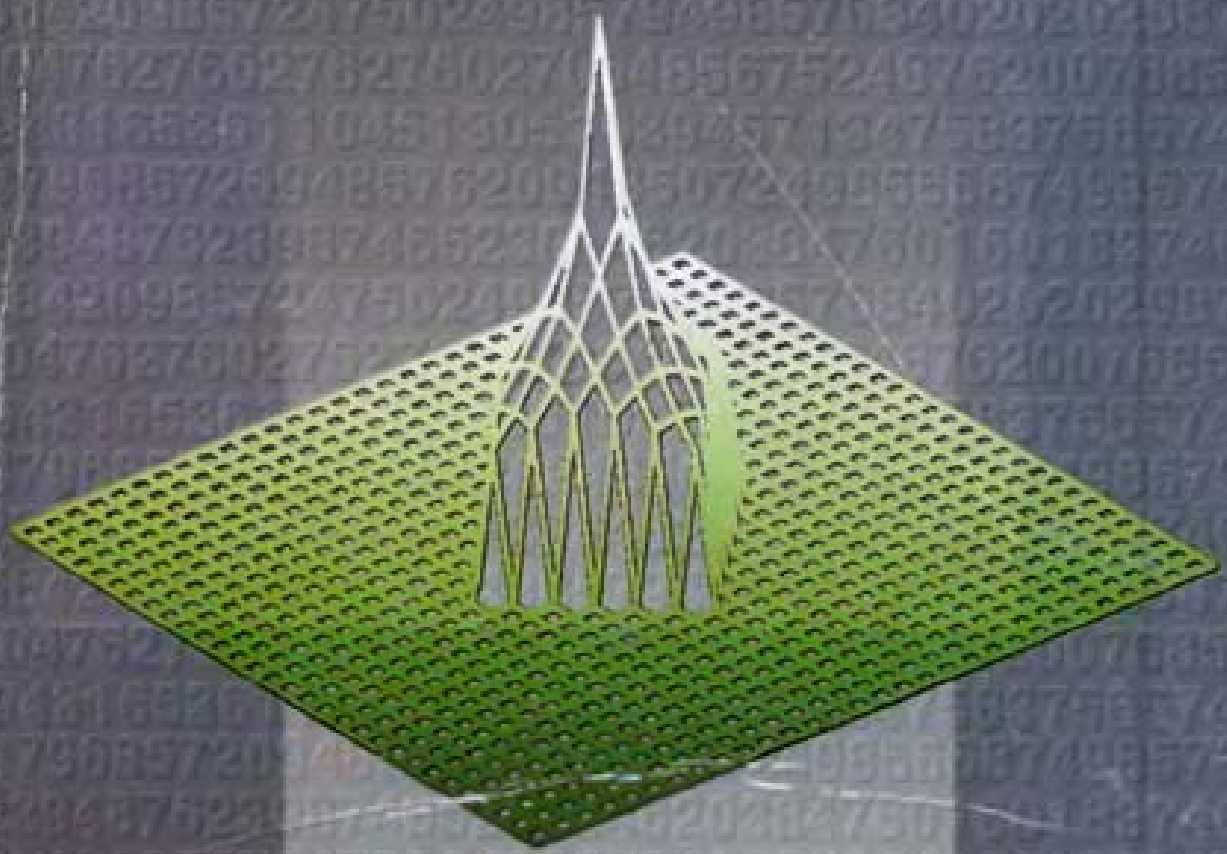
19
Д 509

ДЖ. ДЕММЕЛЬ

ВЫЧИСЛИТЕЛЬНАЯ ЛИНЕЙНАЯ АЛГЕБРА

ТЕОРИЯ И ПРИЛОЖЕНИЯ



Издательство «МИР»

Дж. ДЕММЕЛЬ

ВЫЧИСЛИТЕЛЬНАЯ ЛИНЕЙНАЯ АЛГЕБРА

ТЕОРИЯ И ПРИЛОЖЕНИЯ

Перевод с английского Х. Д. Икрамова



Москва «Мир» 2001

Applied Numerical Linear Algebra

James W. Demmel

University of California
Berkeley, California



Society for Industrial and Applied Mathematics

Philadelphia

УДК 519.852.6
ББК 22.193
Д 30

Деммель Дж.

Д 30 Вычислительная линейная алгебра. Теория и приложения. Пер. с англ. — М.: Мир, 2001. — 430 с., ил.

ISBN 5-03-003402-1

Книга известного американского математика-вычислителя представляет собой учебник повышенного уровня по вычислительным методам линейной алгебры, рядом особенностей выделяющийся среди изданий этого типа:

— знакомит с современными методами решения линейных систем, задач наименьших квадратов, вычисления собственных значений и сингулярных разложений;

— прививает читателям навыки эффективного решения реальных задач путем выбора наилучших алгоритмов;

— содержит упражнения и задачи, облегчающие усвоение материала;

— изложение сопровождается многочисленными ссылками на Интернет-ресурсы по реализации конкретных алгоритмов (Matlab, LAPACK);

— материал книги самодостаточен, от читателя требуется только знакомство с основами линейной алгебры.

Для студентов и аспирантов вузов и университетов, изучающих вычислительную математику и ее приложения.

ББК 22.193

РФФИ

Издание осуществлено при поддержке Российского фонда фундаментальных исследований по проекту № 00-01-14024

Редакция литературы по математическим наукам

Copyright © 1997 by the Society for Industrial and Applied Mathematics
© перевод на русский язык, оформление, «Мир», 2001

ISBN 5-03-003402-1 (русск.)
ISBN 0-89871-389-7 (англ.)

От переводчика

В недоброй памяти ельцинское десятилетие серьезная иностранная литература по математике на русский язык практически не переводилась. Наоборот, в эти годы многие российские математики публиковали свои книги за рубежом. Мотивы были разные: для одних — невозможность издать книгу на родине, для других — шанс приобрести или утвердить свое научное имя на Западе, для третьих — средство занять там хорошую вакансию. Не будем осуждать этих последних; вспомним лучше, какая научная политика проводилась долгое время российскими верхами, и удивимся еще раз их своекорыстию и слепоте.

Кажется, что положение дел понемногу начинает улучшаться. Предлагаемая книга есть одно из свидетельств этому. Читатель среднего и старшего возраста, думаю, будет рад увидеть на обложке знакомую эмблему издательства «Мир», в девяностые годы почти исчезнувшую с полок в книжных магазинах (исчезли, впрочем, и сами полки для научной литературы). Надеюсь, что у книги будут и молодые читатели, возможно, не знающие, что в советское время «Мир» был почти монополистом в области переводной литературы по естественным наукам.

Отрадно, что возрождение издательства опирается на книги такой актуальной тематики и такого высокого качества, как книга Дж. Деммеля. Эта книга — лучшая из многих учебников по вычислительной линейной алгебре, изданных за последние годы на Западе, а ее автор является бесспорным лидером в данной области.

В оригинале книга предназначалась для аспирантов первого года обучения, специализирующихся в компьютерных и инженерных науках. Учитывая различие вузовских программ по математике в России и США, можно сказать, что у нас эта книга будет доступна (и полезна) студентам математических факультетов, начиная со второго курса, а также студентам многих инженерных и естественно-научных специализаций, сталкивающимся с необходимостью проводить сколько-нибудь сложные расчеты. Много нового для себя найдут в книге и специалисты.

Издание этого перевода было поддержано Российским фондом фундаментальных исследований. Хочется поблагодарить отдел издательских проектов фонда за обнадеживающую тенденцию в его работе и хороший выбор книги.

Х. Д. Икрамов

Предисловие к русскому изданию

Известие, что моя книга переведена на русский язык, доставило мне большое удовлетворение. Вычислительная линейная алгебра всегда являлась полем международного сотрудничества; я надеюсь, что этот перевод будет вкладом в продолжение такого сотрудничества.

После выхода в 1997 г. английского издания этой книги в вычислительной алгебре продолжали происходить значительные события. К ним относится появление:

более эффективных и точных математических алгоритмов (таких, как обещанный в разд. 5.3 новый вариант обратной итерации, или ряд усовершенствований в итерационных методах из гл. 6 и 7);

более быстрого и дружественного к пользователю программного обеспечения (методов, более полно использующих современные компьютерные, в особенности параллельные архитектуры по сравнению с теми, что описаны в разд. 2.6, и более совершенных программных библиотек, реализующих эти методы);

улучшенных справочных материалов (см., например, книгу Bai Z., Demmel J., Dongarra J., Ruhe A., van der Vorst H. (editors). Templates for the Solution of Algebraic Eigenvalue Problems — A Practical Guide. SIAM, 2000, помогающую пользователю в выборе наилучшего, среди многих существующих, численного метода при решении спектральных задач).

Я полагаю, что, несмотря на эти изменения, книга по-прежнему достигает целей, намеченных в ее первоначальном предисловии. В частности, при подготовке студентов, специализирующихся в различных инженерных и естественно-научных дисциплинах, для решения практических задач с использованием наиболее эффективных из имеющихся методов. Я надеюсь, что читатели русского перевода получат удовольствие от того взаимодействия математики, компьютерных приемов и приложений, которое делает вычислительную линейную алгебру столь живой наукой.

Джим Деммель

12 января 2001 г.

Предисловие

В этом учебнике рассматриваются прямые и итерационные методы для решения линейных систем, задачи наименьших квадратов, задачи на собственные значения и методы вычисления сингулярного разложения. Предварительные варианты книги использовались автором, начиная с 1990 г., при чтении спецкурсов для аспирантов на кафедре математики Калифорнийского Университета в Беркли, а еще ранее — в Курантовском Институте. При написании этого учебника я старался добиться следующих целей:

1. Учебник должен быть привлекательным для аспирантов первого года различных инженерных и естественнонаучных специализаций.
2. Он должен быть самодостаточным, предполагая у читателя лишь хорошее владение студенческим курсом линейной алгебры.
3. Учащиеся должны освоить математические основы данной области, а также научиться писать свои хорошие программы численных методов либо уметь выбирать такие программы среди существующих.
4. Учащиеся должны приобрести практические навыки эффективного решения реальных задач. В частности, они должны получить представление о наилучших методах внутри каждого раздела. Даже если в учебнике анализируются простейшие варианты методов, они должны знать, в какой ситуации им понадобятся основные версии и где их можно найти.
5. Все сказанное нужно осуществить в течение одного семестра, поскольку большинство учащихся не располагают бóльшим временем для изучения данного предмета.

Побудительным мотивом при написании этой книги мне послужило то обстоятельство, что существующие учебники, даже очень хорошие, не удовлетворяют указанным требованиям. Книга Голуба и Ван Лоана [121] слишком энциклопедична по стилю, а в то же время опускает ряд важных вопросов, например многосеточные методы, декомпозицию области и недавно разработанные алгоритмы для задач на собственные значения. Некоторых из современных алгоритмов нет и в книгах Уоткинса [252] и Трефетена и Бау [243].

Я полагаю, что указанные пять целей мной достигнуты. Трудней всего было удовлетворить пятое требование, особенно потому, что со временем книга разрасталась за счет включения в нее результатов последних исследований и новых разделов, отражающих запросы моих коллег. При одном из возможных разумных построений односеместрового курса, основанного на материале книги, в него должны были бы войти:

- глава 1, исключая раздел 1.5.1;
- глава 2, исключая разделы 2.2.1, 2.4.3, 2.5, 2.6.3 и 2.6.4;
- глава 3, исключая разделы 3.5 и 3.6;
- глава 4, вплоть до (и включая) раздел 4.4.5;
- глава 5, исключая разделы 5.2.1, 5.3.5, 5.4 и 5.5;
- глава 6, кроме разд. 6.3.3, 6.5.5, 6.5.6, 6.6.6, 6.7.2, 6.7.3, 6.7.4, 6.8, 6.9.2 и 6.10;
- глава 7, вплоть до (и включая) разд. 7.3.

Среди примечательных особенностей этой книги хочу упомянуть:

- специальную страницу в Интернете, содержащую написанные на языке Matlab программы для примеров и домашних заданий в тексте книги;
- многочисленные рекомендации и справки, относящиеся к наилучшим современным программам (из пакета LAPACK и других);
- обсуждение вопроса о том, как влияют на конструирование алгоритмов современные компьютерные архитектуры с использованием кэш-памяти;
- сравнение производительности конкурирующих алгоритмов для задач наименьших квадратов и симметричных задач на собственные значения;
- обсуждение ряда итерационных методов от простейшего метода Якоби до многосеточных алгоритмов с детальным сравнением их производительности при решении уравнения Пуассона на квадратной сетке;
- подробное обсуждение и численные примеры для алгоритма Ланцоша, используемого при решении симметричных спектральных задач;
- численные примеры, взятые из различных приложений, простирающихся от колебаний механических систем до вычислительной геометрии;
- разделы о «теории относительных возмущений» и соответствующих высокоточных алгоритмах для симметричных спектральных задач и сингулярного разложения;
- интерпретацию спектральных алгоритмов как динамических систем.

Адрес http://www.siam.org/books/demmel/demmel_class упоминаемой выше Интернет-страницы в тексте книги заменяется на сокращение HOMEPAGE. Используются также следующие два сокращения: PARALLEL_HOMEPAGE заменяет адрес http://www.siam.org/books/demmel/demmel_parallelclass и относится к смежному курсу по параллельным вычислениям, который ведет автор; NETLIB есть сокращение для http://www.netlib.org.

Домашние задания, в зависимости от их сложности, помечаются описаниями «легкое», «средней трудности» или «трудное». Задания со значительным объемом программирования сопровождаются меткой «программирование».

Я обязан многим лицам за участие в подготовке книги. Наиболее значительно было участие Zhojun Bai, который использовал материалы книги в Texas A & M and the University of Kentucky и сделал ряд замечаний и полезных предложений. Отмечу Alan Edelman (MIT), Martin Gutknecht (ETH Zurich), Velvel Kahan (Barkeley), 5которые преподавали по рукописи книги в своих учебных центрах, а также Richard Lehoucq, Beresford Parlett и многих анонимных читателей, которые дали подробные комментарии по различным разделам книги. Кроме того, Alan Edelman и Martin Gutknecht проявили большое гостеприимство, когда я готовил окончательный вариант книги. Таблица 2.2 взята из диссертации моего бывшего студента Xiaoye Li. Я благодарен Mark Adams, Tzu-Yi Chen, Inderjit Dhillon, Jian Xun He, Melody Ivory, Xiaoye Li, Bernd Pfrommer, Huan Ren, Ken Stanley, которые вместе с другими студентами из университетов Courant, Berkeley, Kentucky и из MIT в течение ряда лет помогали мне вычищать рукопись. Большую помощь в наборе книги и подготовке рисунков оказали мне Bob Untiedt и Selene Victor. Фотография для обложки английского издания предоставлена Megan. Наконец, отмечу Kathy Yelick, которая оказывала мне поддержку много лет и как никто другой ожидала завершения этого проекта.

Джеймс Деммель
Беркли, Калифорния Июнь 1997 г.

Глава 1

Введение

1.1. Основные обозначения

В этой книге мы будем постоянно оперировать с *матрицами, векторами и числами (скалярами)*. Всякая матрица будет обозначаться прописной буквой, скажем A, а ее элемент (i,j) будет обозначаться a_{ij}. Если матрица задана выражением типа $A+B$, то будем писать $(A+B)_{ij}$. В более подробных алгоритмических описаниях будет иногда использоваться символ $A\,(i,j)$; для обозначения подматрицы, лежащей на пересечении строк с i-й по j-ю и столбцов с k-го по l-й, используется обозначение $A\,(i:j,\ k:l)$, заимствованное из $Matlab^{TM1}$ [184]. Вектор обозначается строчной буквой, скажем x, а его i-й элемент записывается как x_i. Почти всюду мы рассматриваем только векторы-столбцы, иначе говоря, одностолбцовые матрицы. Для обозначения скаляров используются строчные греческие (а иногда латинские) буквы. Символы $\mathbf{R}$, $\mathbf{R}^n$ и $\mathbf{R}^{m\times n}$ обозначают соответственно множества вещественных чисел, n-мерных вещественных векторов и $m\times n$ вещественных матриц, а символы $\mathbf{C}$, $\mathbf{C}^n$ и $\mathbf{C}^{m\times n}$ — комплексные числа, векторы и матрицы. В некоторых случаях для обозначения $m\times n$-матрицы A используется сокращенная запись $A^{m\times n}$. *Транспонирование* матрицы A обозначается символом A^T: $(A^T)_{ij}=a_{ji}$. Для комплексной матрицы A используется также понятие сопряженной матрицы A^*: $(A^*)_{ij}=\bar{a}_{ji}$. Символы $\Re z$ и $\Im z$ обозначают соответственно вещественную и мнимую части комплексного числа z. Для $m\times n$-матрицы A под $|\,A\,|$ понимается $m\times n$-матрица, составленная из абсолютных величин элементов A: $(|\,A\,|)_{ij}=|\,A_{ij}\,|$. Неравенства типа $|\,A\,|\le|\,B\,|$ следует понимать как системы покомпонентных неравенств: $|\,a_{ij}\,|\le|\,b_{ij}\,|$ для всех i и j. Аналогичное обозначение используется для векторов: $(|\,x\,|)_i=|\,x_i\,|$. Завершение доказательства помечается символом $\square$, а завершение примера — символом $\diamond$. Прочие обозначения будут вводиться по мере необходимости.

[1] Matlab — это зарегистрированный торговый знак фирмы The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760, USA, tel. 508-647-7000, fax 508-647-7001, info@mathworks.com, http://www.mathworks.com.

1.2. Стандартные задачи вычислительной линейной алгебры

Мы будем рассматривать следующие стандартные задачи:

- *Системы линейных уравнений*: решить систему $Ax = b$. Здесь A — заданная невырожденная $n \times n$-матрица, вещественная или комплексная, b — заданный n-мерный вектор-столбец и x — вектор с n компонентами, который мы хотим вычислить.
- *Задачи наименьших квадратов*: вычислить вектор x, минимизирующий $\| Ax - b \|_2$. Здесь A, b и x имеют размерность соответственно $m \times n$, $m \times 1$ и $n \times 1$, а величина $\| y \|_2 = \sqrt{\sum_i |y|^2}$ называется 2-*нормой* вектора y. Если $m > n$ (т.е. число уравнений превышает число неизвестных), то говорят, что система *переопределена*. В этом случае удовлетворить систему $Ax = b$ точно, вообще говоря, не удается. Если $m < n$, то система *недоопределена* и может иметь бесконечно много решений.
- *Задачи на собственные значения*: для заданной $n \times n$-матрицы A найти ненулевой вектор x и число λ, такие, что $Ax = \lambda x$.
- *Задачи на сингулярные значения*: для заданной $m \times n$-матрицы A найти ненулевой вектор x и число λ, такие, что $A^T Ax = \lambda x$. Мы увидим, что этот частный тип задач на собственные значения настолько важен, что заслуживает отдельного рассмотрения и специальных алгоритмов.

Именно эти стандартные задачи выделены по той причине, что они чрезвычайно часто встречаются в инженерной и научной практике. На протяжении всей книги они будут иллюстрироваться простыми примерами, взятыми из инженерных приложений, статистики и других областей. Эти стандартные задачи допускают много вариантов, которые мы также рассмотрим; упомянем, например, обобщенные задачи на собственные значения $Ax = \lambda Bx$ (разд. 4.5) и задачи наименьших квадратов, имеющие «неполный ранг». Такая задача $\min_x \| Ax - b \|_2$ имеет неединственное решение вследствие линейной зависимости столбцов матрицы A (разд. 3.5).

Мы убедимся в том, как важно использовать *специальную структуру*, которую может иметь задача. Например, при решении системы линейных уравнений размера $n \times n$ посредством самой общей версии гауссова исключения приходится выполнять $2/3\ n^3$ операций с плавающей точкой. Если дополнительно известно, что система симметрична и положительно определена, то можно сократить работу вдвое, используя иной алгоритм, называемый методом Холесского. Если вдобавок матрица системы *ленточная* и *полуширина ленты* составляет $\sqrt{n}$ (т.е. $a_{ij} = 0$, если $|i - j| > \sqrt{n}$), то стоимость решения системы можно уменьшить до $O(n^2)$ операций, применяя ленточный вариант метода Холесского. Пусть, наконец, предыдущая информация конкретизируется сообщением, что решается уравнение Пуассона в квадрате, дискретизированное методом конечных разностей на 5-точечном шаблоне. Такое сообщение определяет матрицу почти единственным образом. В этом случае, используя многосеточный алгоритм, можно снизить стоимость решения системы до $O(n)$ операций, чем достигается уже почти предельная скорость (в том смысле, что на вычисление каждой компоненты решения затрачивается не зависящая от n работа см. разд. 6.4).

1.3. Общие аспекты

Мы постоянно будем использовать (и держать в поле зрения) несколько общих понятий и аспектов. Перечислим их:

1. Матричные разложения.
2. Теория возмущений и числа обусловленности.
3. Влияние погрешностей округления на алгоритмы (учитывающее свойства используемой арифметики с плавающей точкой).
4. Анализ скорости алгоритмов.
5. Программирование численных алгоритмов.

1.3.1. Матричные разложения

Разложением матрицы A называется ее представление в виде произведения нескольких «более простых» матриц, облегчающее решение рассматриваемой задачи. Приведем два примера.

Пример 1.1. Пусть нужно решить систему $Ax = b$. Если A — нижнетреугольная матрица

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

то решение легко достигается посредством *прямой подстановки*:

for $i = 1$ to n
  $x_i = (b_i - \sum_{k=1}^{i-1} a_{ik}x_k)/a_{ii}$
end for

Аналогичный прием, называемый *обратной подстановкой*, работает в случае верхнетреугольной матрицы A. Чтобы использовать эти факты для решения системы $Ax = b$ общего вида, нам потребуется описываемое ниже матричное разложение. Оно является всего лишь иной формулировкой гауссова исключения.

Теорема 1.1. *Пусть $n \times n$-матрица A невырождена. Тогда найдутся матрица-перестановка P (т. е. единичная матрица с переставленными строками), невырожденная нижнетреугольная матрица L и невырожденная верхнетреугольная матрица U, такие, что $A = P \cdot L \cdot U$. Чтобы решить систему $Ax = b$, мы решаем эквивалентную систему $PLUx = b$ следующим образом:*

$$\begin{aligned} LUx &= P^{-1}b = P^Tb && \text{(перестановка компонент вектора } b\text{)}, \\ Ux &= L^{-1}(P^Tb) && \text{(прямая подстановка)}, \\ x &= U^{-1}(L^{-1}P^Tb) && \text{(обратная подстановка)}. \end{aligned}$$

Эта теорема будет доказана в разделе 2.3. ◇

Пример 1.2. *Каноническое разложение Жордана* $A = VJV^{-1}$ явным образом указывает собственные значения и собственные векторы матрицы A. Здесь V — невырожденная матрица, в число столбцов которой входят собственные векторы, а J — *жорданова каноническая форма* матрицы A. Она представляет собой специального вида треугольную матрицу, на диагонали которой находятся

собственные значения A. Мы увидим, что, с вычислительной точки зрения, предпочтительней находить *разложение Шура* $A = UTU^*$, где U — унитарная матрица (т.е. столбцы матрицы U ортонормированы), а T — верхнетреугольная матрица, содержащая на диагонали собственные значения матрицы A. Форма Шура T может быть вычислена быстрее и более точно, чем жорданова форма J. Мы обсудим разложения Жордана и Шура в разд. 4.2. ◇

1.3.2. Теория возмущений и числа обусловленности

Результаты, получаемые численными алгоритмами, редко бывают совершенно точными. Имеются два источника ошибок. Во-первых, во входных данных алгоритма могут содержаться ошибки, вызванные предыдущими вычислениями (или измерениями). Во-вторых, ошибки могут порождаться самим алгоритмом вследствие используемых внутри него приближений. И в том, и в другом случае, чтобы оценить погрешности вычисленных результатов, нужно понимать, в какой степени может измениться (или *возмутиться*) решение задачи при слабом возмущении ее входных данных.

Пример 1.3. Пусть $f(x)$ — вещественнозначная дифференцируемая функция вещественной переменной x. Требуется вычислить $f(x)$, при этом значение x точно не известно. Предположим, что вместо x заданы значение $x + \delta x$ и граница для величины δx. Если никакой другой информации нет, то лучшее, что можно сделать, — это вычислить $f(x + \delta x)$ и попытаться оценить абсолютную погрешность $|f(x+\delta x) - f(x)|$. Если использовать для f простое линейное приближение, то получим $f(x + \delta x) \approx f(x) + \delta x f'(x)$; следовательно, погрешность равна $|f(x+\delta x) - f(x)| \approx |\delta x| \cdot |f'(x)|$. Мы назовем $|f'(x)|$ *абсолютным числом обусловленности* функции f в точке x. Если число $|f'(x)|$ велико, то погрешность может быть большой даже для малого δx. В этом случае мы говорим, что f *плохо обусловлена* в точке x. ◇

Мы назвали число обусловленности *абсолютным*, потому что оно позволяет оценить абсолютную погрешность $|f(x + \delta x) - f(x)|$, если задана граница для абсолютного изменения $|\delta x|$ входной величины x. Для оценивания погрешности будет часто использоваться и следующее, по существу эквивалентное выражение:

$$\frac{|f(x + \delta x) - f(x)|}{|f(x)|} \approx \frac{|\delta x|}{|x|} \cdot \frac{|f'(x)| \cdot |x|}{|f(x)|}.$$

Это выражение оценивает *относительную погрешность* $|f(x+\delta x) - f(x)|/|f(x)|$произведением *относительного* изменения входа $|\delta x|/|x|$ и множителя $|f'(x)| \cdot |x|/|f(x)|$, называемым *относительным числом обусловленности*. Часто для краткости мы будем говорить просто о *числе обусловленности*.

Число обусловленности — это именно то, что нужно для понимания, как ошибка во входных данных воздействует на вычисленный результат: чтобы получить оценку погрешности вычисленного решения, мы просто умножаем число обусловленности на границу входной ошибки.

Для каждой из рассматриваемых задач мы выведем выражение для соответствующего числа обусловленности.

1.3.3. Влияние ошибок округления на алгоритмы

Чтобы анализировать ошибки, порождаемые самим алгоритмом, требуется исследовать влияние погрешностей округлений в арифметических операциях; для краткости, будем говорить просто об округлениях. В такого рода исследованиях мы будем пользоваться свойством, присущим большинству хороших алгоритмов. Это свойство, называемое *обратной устойчивостью*, определяется следующим образом.

> Обозначим через $\mathrm{alg}(x)$ наш алгоритм для $f(x)$; в вычисляемый алгоритмом результат включены эффекты округлений. Будем говорить, что $\mathrm{alg}(x)$ является *обратно устойчивым алгоритмом* для $f(x)$, если для всякого x найдется «малое» δx, такое, что $\mathrm{alg}(x) = f(x + \delta x)$. Величина δx называется *обратной ошибкой*. На неформальном уровне можно сказать, что мы получаем точный ответ $(f(x+\delta x))$ для слабо возмущенной задачи $(x + \delta x)$.

Сказанное означает, что для погрешности можно предложить такую оценку:

$$\text{погрешность} = |\mathrm{alg}(x) - f(x)| = |f(x + \delta x) - f(x)| \approx |f'(x)| \cdot |\delta x|,$$

т.е. произведение абсолютного числа обусловленности $|f'(x)|$ и величины обратной ошибки $|\delta x|$. Итак, если алгоритм $\mathrm{alg}(\cdot)$ обратно устойчив, то величина $|\delta x|$ всегда мала, поэтому, если абсолютное число обусловленности не слишком велико, мала будет и погрешность. Отсюда следует, что обратная устойчивость как свойство алгоритма весьма привлекательна, и большинство алгоритмов, которые мы рассмотрим, этим свойством обладают. Поскольку будут известны соответствующие числа обусловленности, мы получим оценки погрешностей для всех вычисленных решений.

Чтобы показать, что алгоритм обратно устойчив, нужно иметь представление о погрешностях округлений основных машинных операций с плавающей точкой и о том, как эти ошибки распространяются внутри алгоритма. Эти темы обсуждаются в разд. 1.5.

1.3.4. Анализ скорости алгоритмов

При выборе алгоритма для решения задачи нужно, конечно, принимать в учет не только обратную устойчивость, но и скорость (называемую также производительностью). Существуют разные способы оценивания скорости. Если в нашем распоряжении имеются конкретная задача, конкретная реализация алгоритма и конкретный компьютер, то можно, разумеется, просто инициализировать алгоритм и выяснить, как долго он работает. Такой подход может потребовать больших усилий и времени, поэтому желательно было бы найти более простой способ оценивания. В типичном случае оценку производительности конкретного алгоритма хотелось бы иметь *до* его машинной реализации.

Традиционный способ оценивания временны́х затрат алгоритма состоит в подсчете числа выполняемых им *операций с плавающей точкой*, или *флопов*. Мы будем приводить такие подсчеты для всех рассматриваемых алгоритмов. Однако для современных машинных архитектур эти подсчеты способны ввести в заблуждение по той причине, что передача данных внутри компьюте-

ра к устройству, где эти данные, например, перемножаются, может потребовать значительно больше времени, чем реальное умножение. Это в особенности справедливо для параллельных компьютеров, но верно также и для таких последовательных машин, как рабочие станции и персональные компьютеры. К примеру, умножение матриц на рабочей станции IBM RS6000/590 можно ускорить с 65 мегафлопов (миллионов операций с плавающей точкой, выполняемых за секунду) до 240 мегафлопов, т.е. почти в четыре раза, посредством разумного переупорядочения операций в стандартном алгоритме (и правильного выбора оптимизирующего транслятора). Мы вернемся к обсуждению этой темы в разд. 2.6.

Если рассматривается *итерационный* алгоритм (т.е. алгоритм, который, вместо того чтобы остановиться после заранее заданного известного числа шагов, генерирует последовательность приближений, сходящуюся к результату), то уместно спросить, сколько шагов потребуется, чтобы уменьшить погрешность до приемлемого уровня. Чтобы ответить на этот вопрос, нужно прежде всего определить, является ли сходимость *линейной* (на каждом шаге погрешность умножается на постоянное число $0 < c < 1$, т.е. $|\text{погрешность}_i| \leq c\,|\text{погрешность}_{i-1}|$) или более быстрой, например, *квадратичной* ($|\text{погрешность}_i| \leq c\,|\text{погрешность}_{i-1}|^2$). Если сравниваются два итерационных алгоритма, и оба линейны, то можно спросить, какой из них имеет меньшую константу c. Итерационные методы решения линейных систем и анализ их сходимости составляют предмет гл. 6.

1.3.5. Программирование численных алгоритмов

Три главных соображения, которыми нужно руководствоваться при машинной реализации численных методов или выборе готовой программы, это *простота использования, надежность* и *скорость*. Большинство алгоритмов, рассматриваемых в книге, уже реализованы тщательно написанными программами, учитывающими эти соображения. Если какая-то из существующих программ способна решить вашу задачу, то простота ее использования может перевешивать все остальные факторы, например скорость работы. В самом деле, если задачу нужно решить лишь один или несколько раз, то, как правило, легче воспользоваться универсальной программой, составленной экспертами, чем писать свою более специализированную программу.

Существуют три формы использования программных продуктов, разработанных другими специалистами. Первая форма — это традиционная библиотека, представляющая собой набор программ для решения фиксированного круга задач, скажем, решения линейных систем, вычисления собственных значений, и так далее. Примером может служить библиотека LAPACK [10], обсуждаемая в данной книге. Это собрание наисовременнейших программ, существующее в двух версиях: на языках Fortran и C. Как и многие другие подобные библиотеки, LAPACK распространяется бесплатно (см. NETLIB на WWW[1]). LAPACK обеспечивает надежность и скорость вычислений (например, в нем реализована отмеченная выше возможность ускорения матричных умножений), однако от пользователя требуется внимательное отношение к выбору структур данных

[1] Напоминаем, что в книге NETLIB используется как сокращение для URL-префикса http://www.netlib.org.

и построению последовательностей вызова подпрограмм. На протяжении всей книги мы будем информировать читателя о программном обеспечении этого типа.

Второй формой являются программные среды, значительно более дружественные к пользователю по сравнению с библиотеками типа LAPACK; это достигается, правда, ценой некоторой утраты производительности. Эту форму представляют коммерческая система Matlab [184] и ряд подобных ей систем. Matlab можно рассматривать как простую интерактивную программную среду, где все переменные интерпретируются как матрицы (при этом скаляры суть 1×1-матрицы) и большинство операций линейной алгебры реализованы как встроенные функции. Например, оператор «$C = A * B$» присваивает переменной C значение произведения матриц A и B, а оператор «$A = \mathrm{inv}(B)$» делает значением A матрицу, обратную к B. В среде Matlab можно легко и быстро реализовать алгоритм-прототип, а затем, запустив его, проверить, как он работает. Так как, однако, Matlab принимает ряд алгоритмических решений автоматически, не консультируясь с пользователем, то получаемые этим путем программы могут работать медленнее, чем удачно выбранные библиотечные программы.

Третья форма — это *шаблоны* или, иначе говоря, рецепты для сборки сложных алгоритмов из более простых «строительных блоков». Шаблоны полезны в ситуации, когда алгоритм можно построить многими разными способами, но при этом отсутствует простое правило выбора наилучшей конструкции для конкретной входной задачи; таким образом, значительная роль в конструировании должна быть оставлена пользователю. Примером подобного подхода может служить книга «Шаблоны для решения линейных систем. Строительные блоки для итерационных методов» [24]. Аналогичный набор шаблонов разрабатывается в настоящее время для задач на собственные значения.

1.4. Пример: вычисление многочлена

Мы хотим проиллюстрировать понятия теории возмущений, числа обусловленности, обратной устойчивости и анализа погрешностей округлений на примере *вычисления многочлена*

$$p(x) = \sum_{i=0}^{d} a_i x^i.$$

Напомним правило Горнера для вычисления многочлена:

$p = a_d$
for $i = d - 1$ down to 0
    $p = x * p + a_i$
end for

Применим эту процедуру к многочлену $p(x) = (x-2)^9 = x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$. Рассматривая нижнюю часть рисунка 1.1, мы видим, что вблизи точки $x = 2$ значение $p(x)$, вычисляемое по правилу Горнера, ведет себя совершенно непредсказуемым образом и по справедливости может быть названо «шумом». Действительный график $p(x)$ показан в верхней части рис. 1.1.

Попробуем разобраться, какие выводы для себя мы должны сделать из рисунка. С этой целью посмотрим, что произойдет, если попытаться найти нуль многочлена $p(x)$ с помощью простой программы вычисления нулей, основанной на методе бисекции (деления пополам); эта программа представлена ниже алгоритмом 1.1.

Бисекция применяется к начальному интервалу $[x_{low}, x_{high}]$, на котором $p(x)$ изменяет знак (т.е. $p(x_{low}) \cdot p(x_{high}) < 0$); тем самым, многочлен $p(x)$ должен иметь нуль на этом интервале. Алгоритм вычисляет значение $p(x_{mid})$ в средней точке интервала $x_{mid} = (x_{low} + x_{high})/2$, а затем выясняет, в какой из половин исходного интервала — левой $[x_{low}, x_{mid}]$ или правой $[x_{mid}, x_{high}]$, — $p(x)$ изменяет знак. В любом случае мы находим интервал половинной длины, содержащий нуль многочлена $p(x)$. Процесс бисекции можно продолжать, пока не будет получен интервал желаемой малости.

Таким образом, решение о выборе левой или правой половины интервала зависит от знака числа $p(x_{mid})$. Исследуя график $p(x)$ в нижней части рис. 1.1, мы видим, что при изменении x этот знак стремительно осциллирует между плюсом и минусом. Тем самым малейшее возмущение x_{low} или x_{high} способно полностью изменить последовательность знаковых решений, а также конечный интервал. И действительно, в зависимости от выбора начальных значений x_{low} и x_{high}, алгоритм может сойтись к *любой точке* «области шума», т.е. отрезка от 1.95 до 2.05 (см. вопрос 1.21).

Чтобы получить полное объяснение этого примера, мы должны обратиться к свойствам арифметики с плавающей точкой.

Алгоритм 1.1. *Вычисление нулей функции $p(x)$ методом бисекции.*

```
proc bisect (p, x_low, x_high, tol)
/* найти корень уравнения p(x) = 0 на отрезке [x_low, x_high]
    в предположении, что p(x_low) · p(x_high) < 0 */
/* останов, если корень найден с точностью до ± tol */
p_low = p(x_low)
p_high = p(x_high)
while x_high − x_low > 2 · tol
    x_mid = (x_low + x_high)/2
    p_mid = p(x_mid)
    if p_low · p_mid < 0 then /* имеется корень на [x_low, x_mid] */
        x_high = x_mid
        p_high = p_mid
    else if p_mid · p_high < 0 then /* имеется корень на [x_mid, x_high] */
        x_low = x_mid
        p_low = p_mid
    else /* x_mid является корнем */
        x_low = x_mid
        x_high = x_mid
    end if
end while
root = (x_low + x_high)/2
```

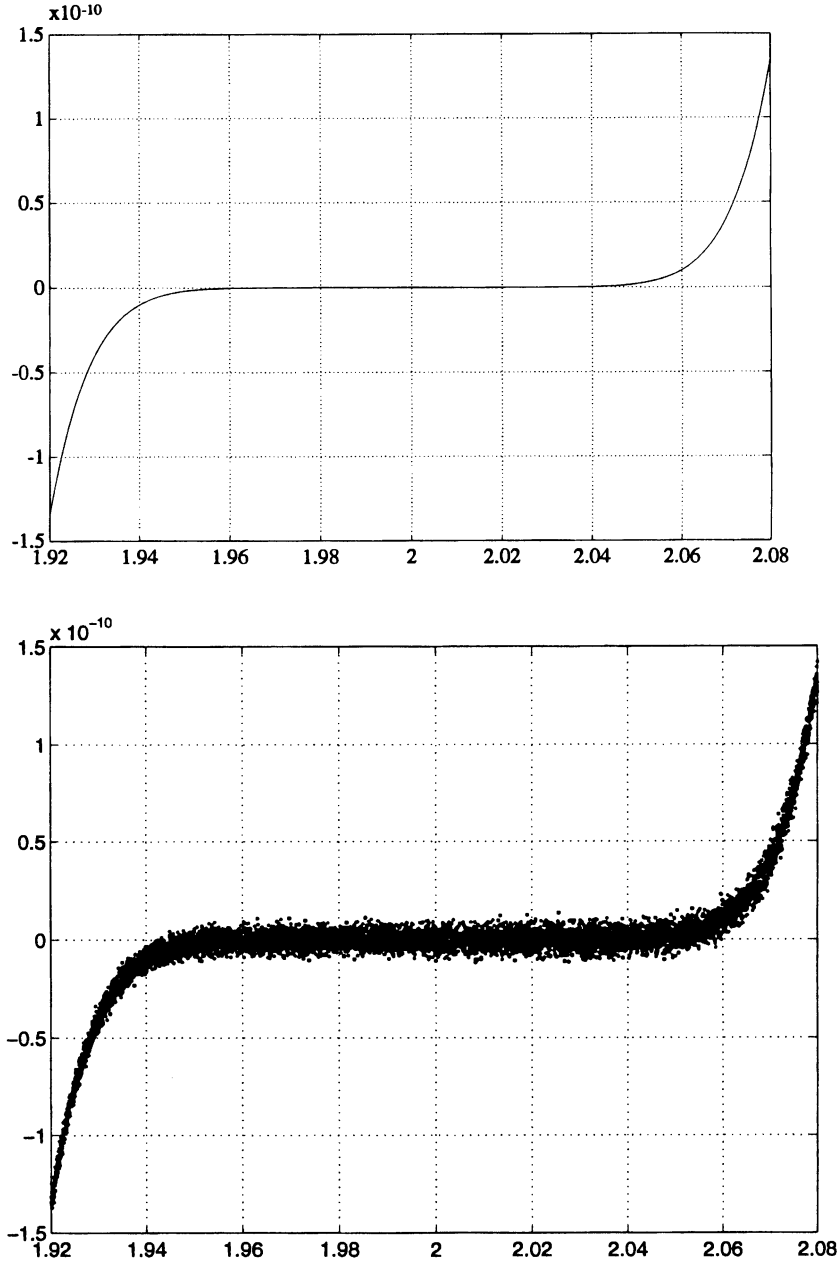
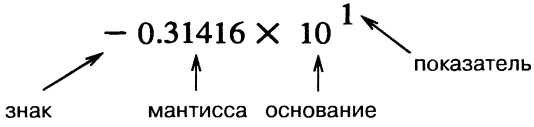


Рис. 1.1. График многочлена $y = (x-2)^9 = x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$. Значения многочлена вычислены в 8000 равноотстоящих точек посредством формулы $y = (x-2)^9$ (верхняя часть рисунка) и правила Горнера (нижняя часть).

1.5. Арифметика с плавающей точкой

Число — 3.1416 можно записать в *математической нотации* как



Схожее представление, называемое представлением с *плавающей точкой*, используется в компьютерах. Правда, основанием обычно является число 2 (с такими исключениями, как 16 для машины IBM 370 и 10 для некоторых решающих таблиц и большинства калькуляторов). Например, $0.10101_2 \times 2^3 = 5.25_{10}$.

Число с плавающей точкой называется *нормализованным*, если старший разряд его мантиссы отличен от нуля. К примеру, число $0.10101_2 \times 2^3$ нормализовано, а число $0.010101_2 \times 2^4$ нет. Обычно числа с плавающей точкой нормализуют, что дает выигрыш в двух отношениях: всякое ненулевое число с плавающей точкой в этом случае имеет единственное представление в виде строки битов, и старший разряд двоичной мантиссы можно не хранить в явном виде (поскольку он всегда равен 1); за счет сэкономленного бита можно удлинить мантиссу.

Самыми важными среди параметров, описывающих числа с плавающей точкой, являются основание, число разрядов (битов) мантиссы, определяющее точность представления, и число разрядов (битов) показателя, определяющее область изменения показателей и, тем самым, наибольшее и наименьшее из представимых чисел. Различные арифметики с плавающей точкой разнятся между собой также способом округления вычисленных результатов, решениями, которые принимаются в отношении чисел, слишком близких к нулю (машинные нули) или слишком больших (переполнения), наличием или отсутствием символов $\pm\infty$ и некоторых полезных нечисел. Нечисло, называемое еще неопределенной величиной или специальным операндом, иногда обозначается NaN. Все эти понятия обсуждаются ниже.

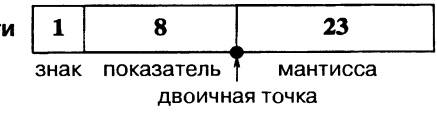
Рассмотрим прежде всего вопрос о точности представления чисел. К примеру, число 0.31416×10^1 имеет пять десятичных разрядов, поэтому в нем могла быть потеряна информация, меньшая, чем 0.5×10^{-4}. Это означает, что если x — вещественное число, для которого наилучшим пятиразрядным представлением является 0.31416×10^1, то *относительная ошибка представления* для 0.31416×10^1 может быть оценена как

$$\frac{|x - 0.31416 \times 10^1|}{0.31416 \times 10^1} \leq \frac{0.5 \times 10^{-4}}{0.31416 \times 10^1} \approx 0.16 \times 10^{-4}.$$

Максимум относительной ошибки представления, возможной для нормализованного числа, реализуется в числе 0.10000×10^1, представляющем собой наиболее точное пятиразрядное представление для всех чисел в интервале от 0.999995 до 1.00005. Относительная ошибка для этого числа оценивается величиной $0.5 \cdot 10^{-4}$. Более общо, для арифметики с плавающей точкой, имеющей p-разрядную мантиссу и основание β, *максимальная относительная ошибка представления* равна $0.5 \times \beta^{1-p}$. Эта величина, к тому же, равна половине

расстояния между 1 и ближайшим бóльшим числом с плавающей точкой, т.е. числом $1 + \beta^{1-p}$.

На протяжении истории компьютеров использовались многие различные комбинации основания, длины мантиссы и области показателей. К счастью, в настоящее время почти общепринятым является *IEEE-стандарт двоичной арифметики*. Он реализован рабочими станциями Sun, DEC, HP и IBM, а также всеми персональными компьютерами. IEEE-арифметика предусматривает два типа чисел с плавающей точкой: *числа обычной точности* (с 32-битовым представлением) и *числа двойной точности* (64 бита).



Пусть в представлении IEEE-числа обычной точности s, e и $f < 1$ суть, соответственно, 1-битовый знак, 8-битовый показатель и 23-битовая мантисса; тогда само число равно $(-1)^s \cdot 2^{e-127} \cdot (1 + f)$. Максимальная относительная ошибка представления равна $2^{-24} \approx 6 \cdot 10^{-8}$, а область положительных нормализованных чисел простирается от 2^{-126} (*порог машинного нуля*) до $2^{127} \cdot (2 - 2^{-23}) \approx 2^{128}$ (*порог переполнения*) или, приблизительно, от 10^{-38} до 10^{38}. Расположение этих чисел с плавающей точкой на вещественной числовой прямой показано на рис. 1.2 (где, в целях иллюстрации, для мантисс принято 3-битовое представление).



Пусть в представлении IEEE-числа двойной точности s, e и $f < 1$ суть, соответственно, 1-битовый знак, 11-битовый показатель и 52-битовая мантисса; тогда само число равно $(-1)^s \cdot 2^{e-1023} \cdot (1 + f)$. Максимальная относительная ошибка представления равна $2^{-53} \approx 10^{-16}$, а границами области показателей являются 2^{-1022} (*порог машинного нуля*) и $2^{1023} \cdot (2 - 2^{-52}) \approx 2^{1024}$ (*порог переполнения*), т.е., приблизительно, 10^{-308} и 10^{308}.

Пусть символ $\odot$ обозначает одну из четырех бинарных операций $+$, $-$, $*$ и $/$. Если точный результат вычисления $a \odot b$ не может быть представлен как число с плавающей точкой, то, прежде чем записать его в память или регистр, нужно аппроксимировать его каким-либо близко расположенным числом с плавающей точкой. Это приближение будем обозначать через $\mathrm{fl}(a \odot b)$. Разность $(a \odot b) - \mathrm{fl}(a \odot b)$ называется *погрешностью округления*. Если $\mathrm{fl}(a \odot b)$ всегда является ближайшим числом с плавающей точкой к числу $a \odot b$, то будем говорить, что арифметика *округляет правильно* (или, просто, *округляет*). IEEE-арифметика обладает этим привлекательным свойством. (Если число $a \odot b$ находится точно посредине между двумя соседними числами с плавающей точкой, то из двух возможных значений для $\mathrm{fl}(a \odot b)$ IEEE-арифметика выбирает число с нулевым последним разрядом мантиссы; такой выбор называется *округлением до ближайшего четного*.) Предположим, что используемая арифметика округляет правильно и число $a \odot b$ не выходит за пределы области допусти-

мых показателей (в противном случае, мы получили бы *машинный нуль* или *переполнение*). Тогда можно записать $\mathrm{fl}(a \odot b)$ как

$$\mathrm{fl}(a \odot b) = (a \odot b)(1 + \delta), \tag{1.1}$$

где $|\delta|$ не превышает числа ε, называемого *машинным эпсилон* или *машинной точностью* и обозначаемого *macheps*. Поскольку округление производится с наивысшей возможной точностью, то ε равно максимальной относительной ошибке представления $0.5 \cdot \beta^{1-p}$. IEEE-арифметика обеспечивает также, что $\mathrm{fl}(\sqrt{a}) = \sqrt{a}(1 + \delta)$, где $|\delta| \leq \varepsilon$. Эти соотношения определяют наиболее распространенную модель анализа погрешностей округлений, и именно такая модель используется в данной книге. Почти аналогичные соотношения справедливы для комплексной арифметики с плавающей точкой (см. вопрос 1.12). Отметим, однако, что формула (1.1) игнорирует некоторые заслуживающие интереса детали.

1.5.1. О некоторых важных деталях

IEEE-арифметика включает в себя также *субнормальные числа*, т. е. ненормализованные числа с плавающей точкой, имеющие наименьший возможный показатель. Субнормальные числа суть очень малые числа, находящиеся между нулем и наименьшим нормализованным числом с плавающей точкой (см. рис. 1.2). Наличие таких чисел в арифметике означает, что разность $\mathrm{fl}(x - y)$ никогда не может быть машинным нулем. Результатом является замечательное свойство, состоящее в том, что равенство $\mathrm{fl}(x - y) = 0$ возможно тогда и только тогда, когда $x = y$. Если бы это свойство отсутствовало, то формулу (1.1) пришлось бы изменить так, чтобы она учитывала эффект машинных нулей. Модифицированная формула имела бы вид

$$\mathrm{fl}(a \odot b) = (a \odot b)(1 + \delta) + \eta,$$

где по-прежнему $|\delta| \leq \varepsilon$, а величина $|\eta|$ ограничена очень малым числом, равным наибольшей ошибке, которую может повлечь за собой появление машинного нуля (для IEEE-арифметики обычной точности это $2^{-150} \approx 10^{-45}$, а для IEEE-арифметики двойной точности — $2^{-1075} \approx 10^{-324}$).

IEEE-арифметика предусматривает также символы $\pm\infty$ и NaN (Not a Number — НеЧисло). Символы $\pm\infty$ генерируются при переполнении и в дальнейшем ведут себя в соответствии со следующими арифметическими правилами: $x/\pm\infty = 0$ для всякого числа с плавающей точкой x; $x/0 = \pm\infty$ для всякого ненулевого числа с плавающей точкой x; $+\infty + \infty = +\infty$, и т.д. Любая операция, результат которой, конечный или бесконечный, не определен корректно, генерирует символ NaN. Примерами могут служить $\infty - \infty$, $\frac{\infty}{\infty}$, $\frac{0}{0}$, $\sqrt{-1}$, $\mathrm{NaN} \odot x$, и т.п.

В каждом из следующих случаев — арифметическая операция некорректна и порождает NaN; произошло переполнение; встретилось деление на нуль, вследствие чего генерируется $\pm\infty$; получен машинный нуль, — выставляется *индикатор особого случая* (exception flag). В дальнейшем состояние индикатора может быть проверено программой пользователя. Эти особенности арифметики позволяют разрабатывать более надежные программы (поскольку программа может обнаружить и исправить особые случаи вместо того, чтобы просто

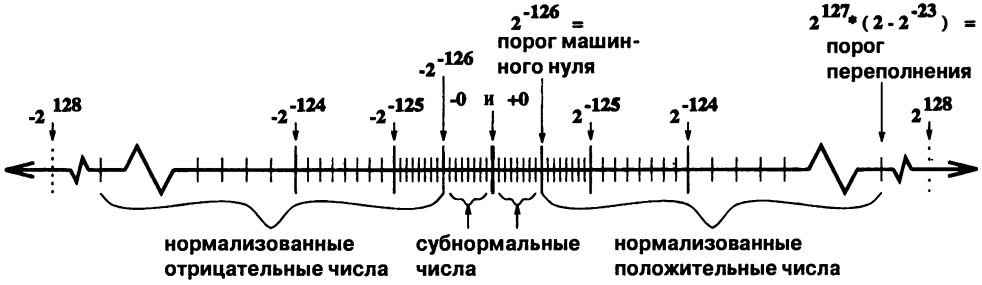


Рис. 1.2. Вещественная числовая прямая, на которой числа с плавающей точкой отмечены сплошными штрихами. Размер показанной области соответствует IEEE-арифметике обычной точности, однако для мантисс, в целях иллюстрации, принято 3-битовое представление. Таким образом, между двумя соседними степенями 2 содержится только $2^3 - 1 = 7$ чисел с плавающей точкой, а не $2^{23} - 1$. Расстояние между соседними штрихами остается постоянным между степенями 2 и увеличивается/уменьшается вдвое при переходе через такую степень. Точечными штрихами указаны числа $+2^{128}$ и -2^{128}; оба они на одну единицу последнего разряда больше (по абсолютной величине) порога переполнения (т.е. наибольшего числа с плавающей точкой) $2^{127} \cdot (2 - 2^{-23})$. Рисунок симметричен относительно нуля; $+0$ и -0 различны как битовые строки IEEE-арифметики, но считаются численно равными. Деление на нуль является единственной операцией, которая дает разные результаты, а именно $+\infty$ и $-\infty$, для нулевых аргументов, снабженных различными знаками.

прекратить свое исполнение) и, вместе с тем, более быстрые (поскольку можно отказаться от «параноидального» стиля программирования с множеством тестов и ветвей, предназначенных для обхода возможных, хоть и маловероятных особых случаев). Подходящие примеры можно найти в вопросе 1.19, комментариях, сопровождающих лемму 5.3, и в статье [81].

Наиболее дорогостоящей из известных ошибок, вызванных неправильной обработкой особого случая арифметики с плавающей точкой, является крушение ракеты Ариан 5 Европейского космического агентства, произошедшее 4 июня 1996 г. Подробности можно узнать по WWW-адресу HOME/ariane5rep.html.

Хотя почти все современные компьютеры используют IEEE-арифметику или правильное округление, все же имеются исключения. Наиболее важными из них являются машины, производимые фирмой Cray Research[1]. Однако будущие поколения компьютеров Cray будут, вполне возможно, работать на основе IEEE-арифметики[2].

Поскольку различия между значениями $\mathrm{fl}(a \odot b)$, вычисленными на компьютере Cray и на IEEE-машине, обычно касаются лишь 14-го десятичного разряда или и того меньше, читатель вправе спросить, так ли уж они важ-

[1] Мы включаем в эту линию машины типа NEC SX-4; эта машина имеет специальный режим, в котором ее арифметика совпадает с арифметикой компьютеров Cray. Мы исключаем из этой линии машины Cray T3D и Cray T3E, представляющие собой параллельные компьютеры, собранные из процессоров DEC Alpha; арифметика последних почти не отличается от IEEE-арифметики (разве что машинные нули, для достижения большей скорости, интерпретируются как точный нуль).

[2] В 1996 г. фирма Cray Research была приобретена компанией Silicon Graphics.

ны. В самом деле, большинство алгоритмов вычислительной линейной алгебры нечувствительны к деталям способа округления. Однако оказывается, что некоторые алгоритмы проще реализуемы или более надежны, если округление производится правильно. Вот два примера.

Если на компьютере Cray C90 вычесть единицу из ближайшего к ней меньшего числа с плавающей точкой, то будет получено значение -2^{-47}, вдвое большее (по абсолютной величине) правильного значения -2^{-48}. Между тем, для корректности алгоритма разделяй-и-властвуй существенно, чтобы даже очень малые разности вычислялись с высокой относительной точностью. Этот алгоритм в настоящее время является наиболее быстрым методом вычисления собственных значений и собственных векторов симметричной матрицы. Требуется весьма нетривиальная модификация алгоритма для того, чтобы обеспечить его корректность на машинах Cray (см. разд. 5.3.3).

Ошибка на машине Cray возможна и при вычислении функции $\arccos(x/\sqrt{x^2+y^2})$, поскольку вследствие чрезмерного округления аргумент арккосинуса может оказаться бóльшим 1. Это исключено в IEEE-арифметике (см. вопрос 1.17).

Чтобы наш анализ погрешностей был пригоден для Cray C90 и других компьютеров Cray, мы можем заменить исходную модель моделью, описываемой уравнениями $\mathrm{fl}(a \pm b) = a(1+\delta_1) \pm b(1+\delta_2)$, $\mathrm{fl}(a * b) = (a * b)(1+\delta_3)$ и $\mathrm{fl}(a/b) = (a/b)(1+\delta_3)$, где $|\delta_i| \le \varepsilon$, а ε лишь небольшим множителем отличается от максимальной относительной ошибки представления.

Вкратце можно сказать, что правильное округление и другие характеристики IEEE-арифметики предназначены для того, чтобы сохранить в силе как можно больше математических соотношений, используемых при выводе формул. Конструировать алгоритмы проще, зная, что (за исключением переполнений и машинных нулей) величина $\mathrm{fl}(a-b)$ вычисляется с малой относительной ошибкой (в противном случае, может неправильно работать процедура разделяй-и-властвуй) и что $-1 \le c \equiv \mathrm{fl}(x/\sqrt{x^2+y^2}) \le 1$ (иначе произойдет отказ при вычислении $\arccos(c)$). Имеется и ряд других математических соотношений, используемых (нередко неосознанно) при разработке алгоритмов. По поводу дополнительных сведений об IEEE-арифметике и ее связях с численным анализом см. [159, 158, 81].

Учитывая существование различных арифметик с плавающей точкой, спрашивается, как же писать переносимые программы, зависящие от арифметики? Например, итерационные алгоритмы, которые мы рассматриваем в последних главах книги, часто включают в себя циклы типа следующего:

```
повторить
    ...
    модифицировать значение e
пока «e не станет пренебрежимо мало по сравнению с f»
```

Здесь $e \ge 0$ есть некоторая мера ошибки, а $f > 0$ — константа сравнения (пример можно найти в разд. 4.4.5). Под «пренебрежимой малостью» мы понимаем выполнение неравенства $e \le c \cdot \varepsilon \cdot f$, где $c \ge 1$ — умеренная константа, выбираемая из компромисса между требованиями точности и скорости сходимости. Поскольку в неравенство входит машинно-зависимая константа ε, то в прошлом его проверка часто заменялась *по видимости* машинно-независимым тестом

«верно ли, что $e + cf = cf$?» Идея такого теста состоит в том, что если $e < c\varepsilon f$ или, возможно, e чуть меньше, чем требуется этим неравенством, то добавление e к cf и последующее округление снова дают cf. Однако на некоторых компьютерах и при использовании некоторых трансляторов (см. по этому поводу следующий абзац) данный тест может потребовать от e *гораздо большей* малости, чем необходимо, или даже оказаться невыполнимым. Поэтому наилучший тест все же использует ε явным образом. Выясняется, что, проявляя достаточную осмотрительность, можно вычислить ε машинно-независимым способом. Это делают, например, подпрограммы пакета LAPACK slamch (для обычной точности) и dlamch (для двойной точности). Эти программы, кроме того, вычисляют или оценивают порог переполнения (без переполнения!), порог машинного нуля и другие параметры. В вопросе 1.19 обсуждается пример переносимой программы, которая использует эти машинные параметры явно.

Иногда бывает нужна более высокая точность, чем та, которую обеспечивают IEEE-стандарты обычной и двойной точности. Например, повышенная точность используется алгоритмом итерационного уточнения, методом, применяемым для улучшения точности вычисленного решения системы $Ax = b$ (см. разд. 2.5.1). Поэтому в IEEE-арифметике определена еще одна, более высокая точность, называемая *расширенной двойной*. К примеру, *все* арифметические операции процессора Intel Pentium (как и его предшественников, вплоть до Intel 8086/8087) выполняются в регистрах расширенной двойной точности; эти регистры имеют длину 80 бит, из которых 64 используются для представления мантиссы и 15 для представления показателя. К сожалению, не все языки и трансляторы позволяют пользователю объявлять переменные расширенной двойной точности и оперировать с ними.

На уровне архитектуры лишь немногие компьютеры имеют какие-либо иные возможности помимо расширенной двойной арифметики. Однако существует несколько способов программного моделирования более точной арифметики. На машинах DEC Vax и DEC Alpha, Sun Sparc и IBM RS6000 некоторые трансляторы позволяют пользователю объявлять переменные *учетверенной точности* (называемые также переменными *дважды двойной точности* или переменными типа *real*16*) и проводить вычисления с ними. Поскольку такая арифметика моделируется посредством меньшей разрядности, работать она может в несколько раз медленнее, чем арифметика двойной точности. Обычная точность компьютеров Cray эквивалентна, в смысле разрядности, IEEE-арифметике двойной точности, которая имеет примерно вдвое меньше разрядов, чем Cray-арифметика двойной точности. Последняя моделируется программно и работает относительно медленно. Отметим еще, что существуют алгоритмы и пакеты для моделирования очень высокой разрядности; они могут использовать целочисленную арифметику [20, 21] или основную арифметику с плавающей точкой [204, 218] (см. по этому поводу вопрос 1.18).

Наконец, упомянем *интервальную арифметику* как стиль вычислений, автоматически генерирующий гарантированные границы для ошибки. Каждая переменная в интервальном вычислении представляется парой чисел с плавающей точкой, из которых одно является нижней границей, а другое — верхней. В процессе вычислений округления производятся таким образом, чтобы гарантировать правильность пересчитываемых нижних и верхних границ. Например, при сложении интервалов $a = [a_l, a_u]$ и $b = [b_l, b_u]$ числа $a_l + b_l$ и

$a_u + b_u$ округляются соответственно *вниз* и *вверх* до ближайших чисел с плавающей точкой c_l и c_u. Это гарантирует, что интервал $c = [c_l, c_u]$ содержит в себе сумму любых двух чисел, взятых из a и b. К сожалению, если, по наивности, кто-то просто возьмет свою программу и заменит в ней все переменные и операции с плавающей точкой интервальными переменными и операциями, то, скорей всего, интервалы, генерируемые модифицированной программой, будут быстро расширяться (возможно даже получение интервала $[-\infty, +\infty]$) и скоро не будут давать никакой полезной информации. (Простым примером может служить многократное повторение оператора $x = x - x$, где x — интервал; вместо $x = 0$ мы получим систему интервалов, в которой при каждом вычитании длина $x_u - x_l$ интервала x удваивается.) Можно все же так модифицировать старые алгоритмы или сконструировать новые, чтобы получать разумные гарантированные оценки для ошибки [4, 140, 162, 190]. Однако алгоритмы такого рода нередко оказываются в несколько раз более медленными, чем алгоритмы, рассматриваемые в этой книге. Оценки ошибок, которые мы даем, в отличие от интервальных границ, не гарантированы в строгом математическом смысле, но они достаточно достоверны почти во всех ситуациях. (Позже мы обсудим этот вопрос более подробно.) Интервальная арифметика далее в этой книге обсуждаться не будет.

1.6. Еще раз о вычислении многочлена

Применим модель округлений (1.1) к анализу вычисления многочлена по правилу Горнера. Напомним основную программу:

$p = a_d$
for $i = d - 1$ down to 0
    $p = x \cdot p + a_i$
end for

Теперь мы пометим индексами промежуточные результаты так, чтобы каждому соответствовал единственный символ (в частности, p_0 — это конечный результат):

$p_d = a_d$
for $i = d - 1$ down to 0
    $p_i = x \cdot p_{i+1} + a_i$
end for

Далее, для каждой операции с плавающей точкой введем множитель $(1 + \delta_i)$, учитывающий округление:

$p_d = a_d$
for $i = d - 1$ down to 0
    $p_i = ((x \cdot p_{i+1})(1 + \delta_i) + a_i)(1 + \delta_i'), \quad$ где $|\delta_i|, |\delta_i'| \le \varepsilon$
end for

В результате, получим следующую формулу для вычисленного значения многочлена:

$$p_0 = \sum_{i=0}^{d-1} \left[(1 + \delta_i') \prod_{j=0}^{i-1} (1 + \delta_j)(1 + \delta_j') \right] a_i x^i + \left[\prod_{j=0}^{d-1} (1 + \delta_j)(1 + \delta_j') \right] a_d x^d .$$

Это громоздкое выражение типично, если в анализе пытаются проследить за распространением внутри алгоритма каждой погрешности округления. Мы упростим его, используя следующие верхние и нижние оценки:

$$(1+\delta_1)\cdots(1+\delta_j) \le (1+\varepsilon)^j \le \frac{1}{1-j\varepsilon} = 1 + j\varepsilon + O(\varepsilon^2),$$
$$(1+\delta_1)\cdots(1+\delta_j) \ge (1-\varepsilon)^j \ge 1 - j\varepsilon.$$

Эти оценки верны, если $j\varepsilon < 1$. В типичном случае мы исходим из (разумного) допущения, что $j\varepsilon \ll 1$ (для IEEE-арифметики обычной точности это все равно, что $j \ll 10^7$), и используем приближения

$$1 - j\varepsilon \le (1+\delta_1)\cdots(1+\delta_j) \le 1 + j\varepsilon.$$

Это позволяет написать

$$\begin{aligned} p_0 &= \sum_{i=0}^{d}(1+\overline{\delta}_i)a_i x^i, \quad \text{где } |\overline{\delta}_i| \le 2d\varepsilon \\ &= \sum_{i=0}^{d}\overline{a}_i x^i \end{aligned}$$

Итак, вычисленное значение p_0 для $p(x)$ есть точное значение слабо возмущенного многочлена с коэффициентами $\overline{a}_i$. Это означает, что вычисление многочлена $p(x)$ есть «обратно устойчивый» процесс. «Обратная ошибка», измеряемая как максимальное относительное изменение коэффициентов $p(x)$, ограничена величиной $2d\varepsilon$.

Используя оценку для обратной ошибки, оценим погрешность в вычисленном значении многочлена:

$$\begin{aligned} |p_0 - p(x)| &= \left|\sum_{i=0}^{d}(1+\overline{\delta}_i)a_i x^i - \sum_{i=0}^{d}a_i x^i\right| \\ &= \left|\sum_{i=0}^{d}\overline{\delta}_i a_i x^i\right| \le \sum_{i=0}^{d}\varepsilon 2d|a_i \cdot x^i| \\ &\le 2d\varepsilon\sum_{i=0}^{d}|a_i \cdot x^i|. \end{aligned}$$

Заметим, что величина $\sum_i |a_i x^i|$ ограничивает сверху наибольшее значение многочлена, которое могло бы получиться, не будь взаимного уничтожения при сложении чисел с различными знаками. Наша оценка есть произведение этой величины и числа $2d\varepsilon$. Аналогичные результаты дает анализ вычисления скалярного произведения и многих других выражений, схожих с многочленами.

Если положить $\overline{\delta}_i = \varepsilon \cdot \operatorname{sign}(a_i x^i)$, то видно, что полученная нами оценка погрешности, с точностью до умеренного множителя $2d$, достижима. Отсюда следует, что дробь

$$\frac{\sum_{i=0}^{d}|a_i x^i|}{|\sum_{i=0}^{d}a_i x^i|}$$

может служить *относительным числом обусловленности* для задачи вычисления многочлена.

Ценой удвоения числа операций найденная оценка для погрешности легко может быть вычислена:

$p = a_d$, $bp = |a_d|$
for $i = d - 1$ down to 0
  $p = x \cdot p + a_i$
  $bp = |x| \cdot bp + |a_i|$
end for
error bound $= bp = 2d \cdot \varepsilon \cdot bp$

Таким образом, подлинное значение многочлена находится в интервале $[p - bp, p + bp]$, а гарантированное число верных десятичных знаков равно $-\log_{10}(|\frac{bp}{p}|)$. Для обсуждавшегося выше многочлена $(x-2)^9$ границы его значений изображены в верхней части рис. 1.3. (Читатель может спросить, не теряет ли граница для погрешности своей справедливости вследствие округлений, произведенных при ее вычислении. Оказывается, что в данном случае округления не создают проблем; в качестве упражнения предлагаем читателю самому убедиться в этом.)

Нижняя часть рис. 1.3 есть график функции $-\log_{10}|\frac{bp}{p}|$, т.е. нижней границы для числа верных десятичных знаков. График свидетельствует, что мы должны встретиться с затруднениями при попытке вычислить $p(x)$ с высокой точностью, если $p(x)$ близко к нулю. Что же особенного в случае $p(x) = 0$? Дело в том, что произвольно малая ошибка ε, сделанная при вычислении $p(x) = 0$, имеет следствием бесконечную относительную ошибку $\frac{\varepsilon}{p(x)} = \frac{\varepsilon}{0}$. Иными словами, наша граница для относительной погрешности $2d\varepsilon \sum_{i=0}^{d} |a_i x^i| / |\sum_{i=0}^{d} a_i x^i|$ бесконечна.

Определение 1.1. *Задача, число обусловленности которой бесконечно, называется* некорректной. *В противном случае задача называется* корректной.[1]

Существует простая геометрическая интерпретация числа обусловленности: оно показывает, насколько далек $p(x)$ от многочлена, для которого задача вычисления значения в точке x некорректна.

Определение 1.2. *Пусть* $p(z) = \sum_{i=0}^{d} a_i z^i$ *и* $q(z) = \sum_{i=0}^{d} b_i z^i$. Относительным расстоянием $d(p, q)$ от p до q *называется наименьшее число* α, *удовлетворяющее неравенствам* $|a_i - b_i| \leq \alpha(p, q) \cdot |a_i|$ *для* $0 \leq i \leq d$. *(Если все* $a_i \neq 0$, *то можно просто положить* $d(p, q) = \max_{0 \leq i \leq d} |\frac{a_i - b_i}{a_i}|$.*)*

Заметим, что если $a_i = 0$, то для того, чтобы число $d(p, q)$ было конечно, b_i также должно быть нулем.

[1] Это определение до некоторой степени нестандартно, поскольку включает в число некорректных задачи, решения которых изменяются непрерывно (но при этом недифференцируемым образом). Примерами таких задач могут служить вычисление кратных корней многочленов и вычисление кратных собственных значений матриц (разд. 4.3). Корректную задачу можно еще описать как задачу, для которой число верных знаков вычисленного решения, с точностью до заранее фиксированного числа последних разрядов, всегда совпадает с числом разрядов арифметики, используемой для вычислений; если это не так, мы имеем некорректную задачу. Например, при вычислении кратных корней многочлена обычно теряется *половина* (или даже более того) точности арифметики.

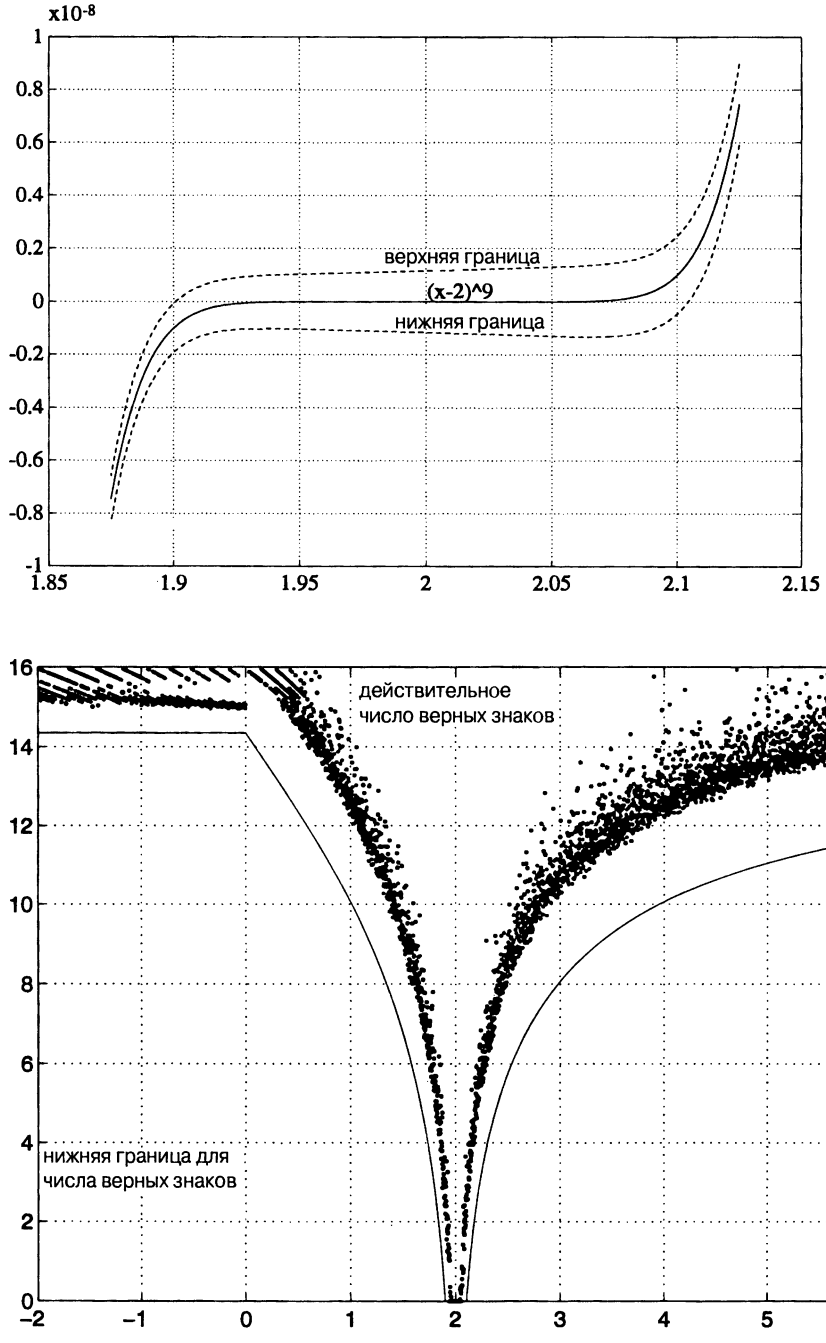


Рис. 1.3. График границ погрешности для значений многочлена $y = (x-2)^9$, вычисленных по правилу Горнера.

Теорема 1.2. *Предположим, что многочлен* $p(z) = \sum_{i=0}^{d} a_i z^i$ *не равен тождественно нулю. Тогда*

$$\min\{d(p,q) \text{ для многочленов } q, \text{ таких, что } q(x) = 0\} = \frac{|\sum_{i=0}^{d} a_i x^i|}{\sum_{i=0}^{d} |a_i x^i|}.$$

Другими словами, относительное расстояние от p *до ближайшего многочлена* q*, число обусловленности которого в точке* x *бесконечно (т.е.* $q(x) = 0$*), обратно числу обусловленности многочлена* $p(x)$*.*

Доказательство. Положим $q(z) = \sum b_i z^i = \sum (1 + \varepsilon_i) a_i z^i$, тогда $d(p,q) = \max_i |\varepsilon_i|$. Из равенства $q(x) = 0$ выводим $|p(x)| = |q(x) - p(x)| = |\sum_{i=0}^{d} \varepsilon_i a_i x^i| \leq \sum_{i=0}^{d} |\varepsilon_i a_i x^i| \leq \max_i |\varepsilon_i| \sum_i |a_i x^i|$, откуда, в свою очередь, следует, что $d(p,q) = \max |\varepsilon_i| \geq |p(x)| / \sum_i |a_i x^i|$. Многочлен q, находящийся на требуемом расстоянии от p, получается при выборе

$$\varepsilon_i = \frac{-p(x)}{\sum |a_i x^i|} \cdot \operatorname{sign}(a_i x^i). \qquad \square$$

Это простое взаимно-обратное соотношение между числом обусловленности и расстоянием до ближайшей некорректной задачи очень типично для численного анализа, и позже мы встретим его снова.

В начале этой главы было сказано, что канонические формы матриц будут использоваться как вспомогательное средство при решении линейно-алгебраических задач. Например, знание точной жордановой канонической формы делает вычисление точных собственных значений тривиальным. Существует аналогичная каноническая форма для многочленов, облегчающая вычисление их значений с высокой точностью; она имеет вид $p(x) = a_d \prod_{i=1}^{d} (x - r_i)$. Иными словами, в представлении многочлена используются его старший коэффициент a_d и корни $r_1, \ldots, r_n$. Вычисление значения $p(x)$ проводится посредством очевидного алгоритма

```
p = a_d
for i = 1 to d
    p = p · (x − r_i)
end for
```

Нетрудно показать, что вычисленное значение p удовлетворяет соотношению $p = p(x) \cdot (1 + \delta)$, где $|\delta| \leq 2d\varepsilon$, т.е. значение $p(x)$ всегда находится с высокой относительной точностью. Однако для этого нужно знать корни многочлена!

1.7. Векторные и матричные нормы

Нормы используются для измерения погрешностей в матричных вычислениях, поэтому нужно уметь обращаться с ними и уметь вычислять их.

Отсутствующие в этом параграфе доказательства предлагаются в качестве задач в конце главы.

Определение 1.3. *Пусть* $\mathcal{B}$ *— вещественное линейное пространство* $\mathbf{R}^n$ *или комплексное линейное пространство* $\mathbf{C}^n$*. Пространство называется* нормированным*, если задана функция* $\|\cdot\| : \mathcal{B} \to \mathbf{R}$*, называемая* нормой *и обладающая следующими свойствами:*

1) $\|x\| \geq 0$, *и* $\|x\| = 0$ *тогда и только тогда, когда* $x = 0$ (положительная определенность),
2) $\|\alpha x\| = |\alpha| \cdot \|x\|$ *для всякого вещественного (или комплексного) числа* α (однородность),
3) $\|x + y\| \leq \|x\| + \|y\|$ (неравенство треугольника).

Пример 1.4. Чаще всего используемыми нормами являются $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$, где $1 \leq p < \infty$, (мы называем их *p-нормами*), а также $\|x\|_\infty = \max_i |x_i|$ (так называемая ∞-*норма*). Кроме того, если $\|x\|$ — произвольная норма, а C — любая невырожденная матрица, то $\|Cx\|$ — также норма. ◇

Мы видим, что для измерения погрешностей можно использовать много различных норм, и важно уметь выбирать наиболее подходящую из них. Пусть, например, $x_1 = [1, 2, 3]^T$ и $x_2 = [1.01, 2.01, 2.99]^T$, где все компоненты измерены в метрах. Тогда x_2 есть хорошее приближение к x_1, поскольку относительная погрешность $\frac{\|x_1 - x_2\|_\infty}{\|x_1\|_\infty} \approx 0.0033$. Напротив, вектор $x_3 = [10, 2.01, 2.99]^T$ является плохим приближением, так как $\frac{\|x_1 - x_3\|_\infty}{\|x_1\|_\infty} = 3$. Предположим, однако, что для измерения первых компонент в качестве единицы теперь выбран километр, а не метр. Если мы по-прежнему работаем с ∞-нормой, то векторы $\hat{x}_1$ и $\hat{x}_3$ становятся близкими:

$$\hat{x}_1 = \begin{bmatrix} 0.001 \\ 2 \\ 3 \end{bmatrix}, \quad \hat{x}_3 = \begin{bmatrix} 0.01 \\ 2.01 \\ 2.99 \end{bmatrix} \quad \text{и} \quad \frac{\|\hat{x}_1 - \hat{x}_3\|_\infty}{\|\hat{x}_1\|_\infty} \approx 0.0033.$$

Чтобы привести все компоненты к одной и той же единице, так чтобы одинаковые по важности погрешности различных компонент вносили одинаковый вклад в норму, мы должны были бы при сравнении $\hat{x}_1$ и $\hat{x}_3$ использовать норму

$$\|\hat{x}\|_s \equiv \left\| \begin{bmatrix} 1000 & & \\ & 1 & \\ & & 1 \end{bmatrix} \hat{x} \right\|_\infty$$

В линейной алгебре часто используются *скалярные произведения*. Мы определим сейчас это понятие как обобщение *стандартного скалярного произведения* $\sum_i x_i y_i$.

Определение 1.4. *Пусть* $\mathcal{B}$ *— вещественное (комплексное) линейное пространство. Функция* $\langle \cdot, \cdot \rangle : \mathcal{B} \times \mathcal{B} \to \mathbf{R}(\mathbf{C})$ *называется* скалярным произведением, *если она обладает следующими свойствами:*

1) $\langle x, y \rangle = \langle y, x \rangle$ *(или* $\overline{\langle y, x \rangle}$*)*,
2) $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$,
3) $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$ *для всякого вещественного (или комплексного) числа* α,
4) $\langle x, x \rangle \geq 0$, *и* $\langle x, x \rangle = 0$ *тогда и только тогда, когда* $x = 0$.

Пример 1.5. Скалярными произведениями являются функции $\langle x, y \rangle = y^T x = \sum_i x_i y_i$ над $\mathbf{R}$ и $\langle x, y \rangle = y^* x = \sum_i x_i \bar{y}_i$ над $\mathbf{C}$. (Напомним, что вектор $y^* = \bar{y}^T$ называется сопряженным к вектору y.) ◇

Определение 1.5. *Векторы* x *и* y *ортогональны, если* $\langle x, y \rangle = 0$.

Наиболее важным из свойств скалярного произведения является то, что оно удовлетворяет неравенству Коши—Шварца. Последнее может быть использовано для доказательства того, что функция $\sqrt{\langle x, x\rangle}$ есть норма. Эту норму мы в дальнейшем часто используем.

Лемма 1.1. *Неравенство Коши—Шварца.* $|\langle x, y\rangle| \leq \sqrt{\langle x, x\rangle \cdot \langle y, y\rangle}$.

Лемма 1.2. *Функция* $\sqrt{\langle x, x\rangle}$ *является нормой.*

Имеется взаимно-однозначное соответствие между скалярнымми произведениями и *симметричными (эрмитовыми) положительно определенными матрицами*, определение которых мы сейчас дадим. Эти матрицы часто встречаются в приложениях.

Определение 1.6. *Вещественная симметричная (комплексная эрмитова) матрица* A положительно определена, *если* $x^T Ax > 0$ ($x^* Ax > 0$) *для всех* $x \neq 0$.

Для краткости, мы будем называть симметричные и эрмитовы положительно определенные матрицы соответственно с.п.о. матрицами и э.п.о. матрицами.

Лемма 1.3. *Пусть* $\mathcal{B} = \mathbf{R}^n$ *(или* $\mathbf{C}^n$*) и пусть* $\langle \cdot, \cdot \rangle$ *— скалярное произведение. Тогда существует с.п.о. (э.п.о.) матрица* A *размера* $n \times n$*, такая, что* $\langle x, y\rangle = y^T Ax$ ($y^* Ax$)*. Обратно, если* A *— с.п.о. (э.п.о.) матрица, то функция* $y^T Ax$ ($y^* Ax$) *является скалярным произведением.*

Следующие две леммы полезны в ситуации, когда мы хотим преобразовать границу погрешности, заданную в одной норме, в границу, записанную посредством другой нормы.

Лемма 1.4. *Пусть* $\|\cdot\|_\alpha$ *и* $\|\cdot\|_\beta$ *— две нормы на* $\mathbf{R}^n$ *(или* $\mathbf{C}^n$*). Тогда существуют константы* $c_1, c_2 > 0$*, такие, что* $c_1\|x\|_\alpha \leq \|x\|_\beta \leq c_2\|x\|_\alpha$ *для всех векторов* x.

Мы будем говорить, что нормы $\|\cdot\|_\alpha$ и $\|\cdot\|_\beta$ *эквивалентны* с *константами эквивалентности* c_1 и c_2.

Лемма 1.5.

$$\begin{array}{ccccc} \|x\|_2 & \leq & \|x\|_1 & \leq & \sqrt{n}\|x\|_2, \\ \|x\|_\infty & \leq & \|x\|_2 & \leq & \sqrt{n}\|x\|_\infty, \\ \|x\|_\infty & \leq & \|x\|_1 & \leq & n\|x\|_\infty \end{array}$$

Помимо векторных норм, мы нуждаемся еще в *матричных нормах*, чтобы измерять погрешность в матрицах.

Определение 1.7. *Функция* $\|\cdot\|$ *называется* матричной нормой *на множестве* $m \times n$*-матриц, если она является векторной нормой на соответствующем* mn*-мерном пространстве, т.е.:*

1) $\|A\| \geq 0$ *и* $\|A\| = 0$ *тогда и только тогда, когда* $A = 0$,
2) $\|\alpha A\| = |\alpha| \cdot \|A\|$,
3) $\|A + B\| \leq \|A\| + \|B\|$.

Пример 1.6. Функции $\max_{ij}|a_{ij}|$ и $(\sum|a_{ij}|^2)^{1/2} = \|A\|_F$ называются соответственно *тах-нормой* и *нормой Фробениуса.* ◇

Следующее определение полезно, если нужно оценить норму произведения матриц. Нам часто придется делать это при выводе границ для погрешностей.

Определение 1.8. *Пусть* $\|\cdot\|_{m\times n}$, $\|\cdot\|_{n\times p}$ *и* $\|\cdot\|_{m\times p}$ *— матричные нормы на множествах соответственно* $m\times n$, $n\times p$ *и* $m\times p$*-матриц. Эти нормы называются* взаимно согласованными, *если* $\|A\cdot B\|_{m\times p}\le\|A\|_{m\times n}\cdot\|B\|_{n\times p}$ *для любых* $m\times n$*-матрицы* A *и* $n\times p$*-матрицы* B.

Определение 1.9. *Пусть* A *— матрица размера* $m\times n$, *а* $\|\cdot\|_{\hat m}$ *и* $\|\cdot\|_{\hat n}$ *— векторные нормы на пространствах соответственно* $\mathbf{R}^m$ *и* $\mathbf{R}^n$. *Тогда функция*

$$\|A\|_{\hat m\hat n}\equiv\max_{\substack{x\neq0\\ x\in\mathbf{R}^n}}\frac{\|Ax\|_{\hat m}}{\|x\|_{\hat n}}$$

называется операторной (*или* индуцированной, *или* подчиненной) нормой.

Следующая лемма снабжает нас обильным источником матричных норм, пригодных для оценивания погрешностей.

Лемма 1.6. *Всякая операторная норма является матричной нормой.*

Определяемые ниже *ортогональные* и *унитарные* матрицы являются существенными ингредиентами почти всех алгоритмов для задач наименьших квадратов и задач на собственные значения.

Определение 1.10. *Вещественная квадратная матрица* Q *называется* ортогональной, *если* $Q^{-1}=Q^T$. *Комплексная квадратная матрица* Q *называется* унитарной, *если* $Q^{-1}=Q^*$.

Все строки (а также столбцы) ортогональной (или унитарной) матрицы имеют единичные 2-нормы и попарно ортогональны, что следует из соотношений $QQ^T=Q^TQ=I(QQ^*=Q^*Q=I)$.

Следующая лемма суммирует существенные свойства определенных выше норм и матриц. Эти свойства будут нужны нам в последующем тексте книги.

Лемма 1.7. 1. *Неравенство* $\|Ax\|\le\|A\|\cdot\|x\|$ *верно для любой векторной нормы и соответствующей операторной нормы, а также для векторной 2-нормы и матричной нормы Фробениуса.*

2. *Неравенство* $\|AB\|\le\|A\|\cdot\|B\|$ *верно для всякой операторной нормы, а также для нормы Фробениуса. (Иными словами, всякая операторная норма, так же как и норма Фробениуса, сама с собой согласована.)*
3. *Норма Фробениуса и тах-норма не являются операторными нормами.*
4. *Для нормы Фробениуса, а таже для операторной нормы, индуцированной векторной 2-нормой, верно равенство* $\|QAZ\|=\|A\|$ *для любых ортогональных или унитарных матриц* Q *и* Z. *В действительности, это не что иное, как теорема Пифагора.*
5. *Верно равенство* $\|A\|_\infty\equiv\max_{x\neq0}\frac{\|Ax\|_\infty}{\|x\|_\infty}=\max_i\sum_j|a_{ij}|$ = *(максимальная строчная сумма модулей).*
6. $\|A\|_1\equiv\max_{x\neq0}\frac{\|Ax\|_1}{\|x\|_1}=\|A^T\|_\infty=\max_j\sum_i|a_{ij}|=$ *(максимальная столбцевая сумма модулей).*

7. $\|A\|_2 \equiv \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sqrt{\lambda_{\max}(A^*A)}$. *Здесь символ* $\lambda_{\max}$ *обозначает наибольшее собственное значение.*
8. $\|A\|_2 = \|A^T\|_2$.
9. $\|A\|_2 = \max_i |\lambda_i(A)|$, *если* A — нормальная *матрица, т.е.* $AA^* = A^*A$.
10. *Если* A — *матрица размера* $n \times n$, *то* $n^{-1/2}\|A\|_2 \leq \|A\|_1 \leq n^{1/2}\|A\|_2$.
11. *Если* A — *матрица размера* $n \times n$, *то* $n^{-1/2}\|A\|_2 \leq \|A\|_\infty \leq n^{1/2}\|A\|_2$.
12. *Если* A — *матрица размера* $n \times n$, *то* $n^{-1}\|A\|_\infty \leq \|A\|_1 \leq n\|A\|_\infty$.
13. *Если* A — *матрица размера* $n \times n$, *то* $\|A\|_2 \leq \|A\|_F \leq n^{1/2}\|A\|_2$.

Доказательство. Мы докажем только утверждение 7 и внесем остальные доказательства в вопрос 1.16.

Поскольку A^*A — эрмитова матрица, то существует спектральное разложение $A^*A = Q\Lambda Q^*$, где Q — унитарная матрица (ее столбцы являются собственными векторами), а $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ — диагональная матрица, содержащая собственные значения; все они должны быть вещественны. Заметим, что все λ_i неотрицательны. Действительно, если бы какое-то из них, скажем λ, было отрицательно, то с помощью соответствующего собственного вектора q мы пришли бы к противоречию: $0 \leq \|Aq\|_2^2 = q^TA^TAq = q^T\lambda q = \lambda\|q\|_2^2 < 0$. Теперь имеем

$$\begin{aligned}
\|A\|_2 &= \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{x\neq 0} \frac{(x^*A^*Ax)^{1/2}}{\|x\|_2} = \max_{x\neq 0} \frac{(x^*Q\Lambda Q^*x)^{1/2}}{\|x\|_2} \\
&= \max_{x\neq 0} \frac{((Q^*x)^*\Lambda Q^*x)^{1/2}}{\|Q^*x)\|_2} = \max_{y\neq 0} \frac{(y^*\Lambda y)^{1/2}}{\|y\|_2} = \max_{y\neq 0} \sqrt{\frac{\sum \lambda_i y_i^2}{\sum y_i^2}} \\
&\leq \max_{y\neq 0} \sqrt{\lambda_{\max}} \sqrt{\frac{\sum y_i^2}{\sum y_i^2}} = \sqrt{\lambda_{\max}},
\end{aligned}$$

Эта оценка достижима: достаточно взять в качестве y подходящий столбец единичной матрицы. □

1.8. Литература и смежные вопросы к главе 1

В конце каждой главы мы будем перечислять наиболее важные публикации по ее теме. Их библиографические описания можно найти в упорядоченном по алфавиту списке литературы в конце книги. Кроме того, мы будем кратко комментировать смежные вопросы, не обсуждаемые в основном тексте.

Наиболее современной и полной книгой по матричным вычислениям является книга Дж. Голуба и Ч. Ван Лоуна [121]; там же содержится обширная библиография. Недавняя книга Д. Уоткинса [252] представляет собой введение в матричные вычисления, рассчитанное на студента или начинающего аспиранта. Книга Л. Трефетена и Д. Бау [243] — еще один хороший учебник для аспирантов. Классическая монография Дж. Уилкинсона [262], хотя и несколько устарела, остается замечательным справочным пособием. Старая книга Дж. Стюарта [235] все еще является отличным учебником того же уровня, что и книга Уоткинса.

Более подробную информацию об анализе ошибок можно найти в недавно изданной книге Н. Хигэма [149]. Старыми, но по-прежнему хорошими источниками являются книги Дж. Уилкинсона [261] и В. Кахана [157].

Недавно опубликован хороший обзор Д. Гольдберга под названием «Что должен знать об арифметике с плавающей точкой любой специалист по информатике» [119]. Формальное описание IEEE-арифметики дается в [11, 12, 159], а также в справочных руководствах, публикуемых производителями компьютеров. Обсуждение анализа погрешностей в рамках IEEE-арифметики можно найти в [54, 70, 159, 158]; см. также библиографические ссылки, даваемые в этих публикациях.

Наиболее общее рассмотрение вопроса о числах обусловленности и расстоянии до ближайшей некорректной задачи проводится в работе автора [71] и серии статей С. Смейла и М. Шуба [219 — 222]. Векторные и матричные нормы подробно обсуждаются в [121, разд. 2.2 и 2.3].

1.9. Вопросы к главе 1

Вопрос 1.1 (*легкий; Z. Bai*). Пусть A — ортогональная матрица. Показать, что $\det(A) = \pm 1$. Показать, что если B — также ортогональная матрица и $\det(A) = -\det(B)$, то матрица $A + B$ вырождена.

Вопрос 1.2 (*легкий; Z. Bai*). *Рангом* матрицы называется размерность линейного подпространства, натянутого на ее столбцы. Показать, что матрица A тогда и только тогда имеет ранг 1, когда она допускает представление $A = ab^T$ для некоторых векторов-столбцов a и b.

Вопрос 1.3 (*легкий; Z. Bai*). Показать, что если матрица одновременно является ортогональной и треугольной, то она диагональная. Что можно сказать о ее диагональных элементах?

Вопрос 1.4 (*легкий; Z. Bai*). Матрица называется *строго верхнетреугольной*, если она верхнетреугольная и имеет нулевые диагональные элементы. Показать, что если A — строго верхнетреугольная размера $n \times n$, то $A^n = 0$.

Вопрос 1.5 (*легкий; Z. Bai*). Пусть $\|\cdot\|$ — векторная норма на $\mathbf{R}^n$, и пусть $C \in \mathbf{R}^{m \times n}$. Показать, что если $\operatorname{rank}(C) = n$, то $\|x\|_C \equiv \|Cx\|$ является векторной нормой.

Вопрос 1.6 (*легкий; Z. Bai*). Показать, что если $0 \neq s \in \mathbf{R}^n$ и $E \in \mathbf{R}^{n \times n}$, то

$$\left\| E\left(I - \frac{ss^{\mathrm{T}}}{s^T s}\right)\right\|_F^2 = \|E\|_F^2 - \frac{\|Es\|_2^2}{s^T s}.$$

Вопрос 1.7 (*легкий; Z. Bai*). Проверить, что $\|xy^H\|_F = \|xy^H\|_2 = \|x\|_2\|y\|_2$ для любых векторов $x, y \in \mathbf{C}^n$.

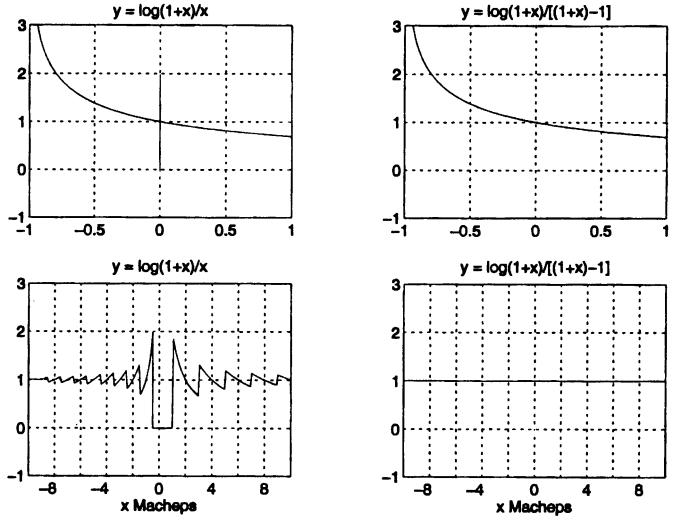
Вопрос 1.8 (*средней трудности*). Сопоставляя каждому многочлену $p(x) = \sum_{i=0}^{d} a_i x^i$ степени d вектор его коэффициентов, можно отождествить множество таких многочленов с пространством $\mathbf{R}^{d+1}$. Фиксируем число x, и пусть S_x — множество многочленов, для которых относительное число обусловленности по отношению к задаче вычисления в точке x бесконечно (т.е. такие

многочлены равны нулю в точке x). Несколькими словами опишите геометрически S_x как подмножество пространства $\mathbf{R}^{d+1}$. Пусть $S_x(\kappa)$ — множество многочленов, относительное число обусловленности которых не меньше, чем κ. В нескольких словах дайте геометрическое описание этого множества. Опишите геометрически, как изменяется $S_x(\kappa)$ при $\kappa \to \infty$.

Вопрос 1.9 (*средней трудности*). На рисунке, сопровождающем этот вопрос, помещены графики функции $y = \log(1+x)/x$, вычисленной двумя разными способами. С математической точки зрения, y есть гладкая функция от x в окрестности точки $x = 0$, принимающая в нуле значение 1. Однако, если вычислять y по исходной формуле, то будут получены графики в левой части рисунка (верхний показывает функцию в области $x \in [-1, 1]$, а нижний — в области $x \in [-10^{-15}, 10^{-15}]$). Очевидно, что вычисление по этой формуле вблизи $x = 0$ неустойчиво. С другой стороны, можно использовать следующий алгоритм:

$d = 1 + x$
if $d = 1$ then
    $y = 1$
else
    $y = \log(d)/(d-1)$
end if

Этому способу соответствуют два графика в правой части рисунка с правильным поведением вблизи точки $x = 0$. Объясните это явление, доказав, что в арифметике с плавающей точкой второй алгоритм должен давать результат хорошей точности. Считайте, что программа для логарифма обеспечивает хорошую точность при любом значении аргумента. (Это справедливо для любой правильной реализации логарифмической функции.) Если это облегчит ваши рассуждения, считайте, что вы работаете в IEEE-арифметике с плавающей точкой. (На компьютерах Cray плохие результаты могут давать оба алгоритма.)



Вопрос 1.10 (*средней трудности*). Показать, что в отсутствии переполнений и машинных нулей справедливо соотношение $\mathrm{fl}\left(\sum_{i=1}^{d} x_i y_i\right) = \sum_{i=1}^{d} x_i y_i(1+\delta_i)$, где $|\delta_i| \leq \delta\varepsilon$. Опираясь на это, доказать следующее: пусть обычным способом вычисляется произведение матриц $A^{m\times n}$ и $B^{n\times p}$. Тогда, в отсутствие переполнений и машинных нулей, выполняется оценка $|\mathrm{fl}(A\cdot B) - A\cdot B| \leq n\cdot\varepsilon\cdot|A|\cdot|B|$. Символ $|A|$ обозначает матрицу с элементами $|a_{ij}|$, а неравенство между матрицами следует понимать как покомпонентное.

Данный результат будет использован в разд. 2.4.2 при анализе ошибок округлений в гауссовом исключении.

Вопрос 1.11 (*средней трудности*). Пусть L — нижнетреугольная матрица, и система $Lx = b$ решается прямой подстановкой. Показать, что в отсутствие переполнений и машинных нулей вычисленное решение $\hat{x}$ удовлетворяет системе $(L+\delta L)\hat{x} = b$, где $|\delta l_{ij}| \leq n\varepsilon|l_{ij}|$ и ε — машинная точность. Это означает, что прямая подстановка является обратно устойчивым процессом. Показать, что сказанное выше сохраняет силу для решения верхнетреугольной системы методом обратной подстановки. Данный результат будет использован в разд. 2.4.2 при анализе ошибок округлений в гауссовом исключении.

Вопрос 1.12 (*средней трудности*). При анализе влияния ошибок округлений была принята следующая модель (см. (1.1)):

$$\mathrm{fl}(a \odot b) = (a \odot b)(1+\delta),$$

где символ $\odot$ обозначает одну из четырех основных операций $+, -, *$ и $/$, а $|\delta| \leq \varepsilon$. Чтобы утверждать, что результаты нашего анализа справедливы и для вычислений с *комплексными* числами, нужно доказать аналогичную формулу для четырех основных комплексных операций. Пусть δ теперь обозначает *комплексное* число, ограниченное по модулю малым кратным числа ε. Доказать, что формула верна для комплексных операций сложения, вычитания, умножения и деления. Ваш алгоритм для комплексного деления должен давать результаты $a/a \approx 1$ как в случае очень большого $|a|$ (большего, чем квадратный корень из порога переполнения), так и в случае очень малого $|a|$ (меньшего, чем квадратный корень из порога машинного нуля). Всегда ли верно, что и вещественная, и мнимая части произведения двух комплексных чисел вычисляются с высокой относительной точностью?

Вопрос 1.13 (*средней трудности*). Доказать лемму 1.3.

Вопрос 1.14 (*средней трудности*). Доказать лемму 1.5.

Вопрос 1.15 (*средней трудности*). Доказать лемму 1.6.

Вопрос 1.16 (*средней трудности*). Доказать все утверждения леммы 1.7, за исключением утверждения 7. Указание к утверждению 8: использовать то обстоятельство, что для $n \times n$-матриц X и Y матрицы XY и YX имеют одни и те же собственные значения. Указание к утверждению 9: использовать тот факт, что нормальность матрицы равносильна существованию для нее полной системы ортонормированных собственных векторов.

Вопрос 1.17 (*трудный; W. Kahan*). Мы отмечали, что на машине Cray наблюдалась ошибка при вычислении выражения $\arccos(x/\sqrt{x^2+y^2})$, вызванная

тем, что вследствие округления аргумент арккосинуса оказывался бóльшим 1. Показать, что в отсутствие переполнений и машинных нулей подобное явление невозможно в IEEE-арифметике. Указание: вам будет недостаточно простой модели $\mathrm{fl}(a \odot b) = (a \odot b)(1+\delta)$, где $|\delta|$ мало. Продумайте вычисление выражения $\sqrt{x^2}$ и покажите, что, в отсутствие переполнений и машинных нулей, равенство $\mathrm{fl}(\sqrt{x^2}) = x$ выполняется *точно*. На машине же Cray YMP в численных экспериментах, проведенных Лю (A. Liu), это равенство нарушалось в 5% случаев. Вы тоже могли бы провести некоторые численные эксперименты и объяснить их. Вы получите дополнительные баллы по данному заданию, если докажете аналогичный результат для *десятичной* арифметики с правильным округлением. (Доказательство в этом случае иное.) Этот вопрос был поставлен У. Каханом (Kahan), обнаружившим ошибку в Cray-программе (автор — J. Sethian).

Вопрос 1.18 (*трудный*). Пусть a и b — нормализованные IEEE-числа двойной точности. Рассмотрим следующий алгоритм, выполняемый в IEEE-арифметике:

если $(|a| < |b|)$, то поменять a и b местами
$s_1 = a + b$
$s_2 = (a - s_1) + b$

Доказать следующие утверждения:

1. В отсутствие переполнений и машинных нулей единственная ошибка округления, совершаемая в этом алгоритме, происходит при вычислении $s_1 = \mathrm{fl}(a+b)$. Другими словами, оба вычитания $s_1 - a$ и $(s_1 - a) - b$ выполняются *точно*.
2. Равенство $s_1 + s_2 = a + b$ выполняется *точно*. Это означает, что фактически s_2 есть ошибка, совершаемая при округлении точной суммы $a+b$ до числа s_1.

Таким образом, данная программа по существу моделирует арифметику *учетверенной* точности, представляя точную сумму $a+b$ с помощью двух чисел: одно (s_1) состоит из старших битов, а другое (s_2) — младших.

Применяя этот и аналогичные трюки систематическим образом, можно эффективно смоделировать все четыре основные операции для арифметики *произвольной* разрядности, используя лишь операции стандартной машинной арифметики и не обращаясь к действиям с отдельными битами [204]. Именно так реализована 128-битовая арифметика на машинах IBM RS6000 и Cray (во втором случае реализация значительно менее эффективна, так как компьютеры Cray не имеют IEEE-арифметики).

Вопрос 1.19 (*трудный; програмирование*). Данное задание иллюстрирует трудности, сопряженные с разработкой высоконадежных программ численного анализа. Ваша задача — написать программу, вычисляющую 2-норму $s \equiv \|x\|_2 = \left(\sum_{i=1}^{\infty} x_i^2\right)^{1/2}$ по заданным $x_1, \ldots, x_n$. Наиболее очевидный (и неудовлетворительный) алгоритм выглядит так:

$s = 0$
for $i = 1$ to n

$s = s + x_i^2$
endfor
$s = \mathrm{sqrt}(s)$

Мы считаем этот алгоритм неудовлетворительным, потому что он не обладает совокупностью следующих желательных свойств:

1. Результат должен вычисляться с высокой точностью (т. е. почти все разряды вычисленного результата должны быть верными), если $||x||_2$ не находится (почти) за пределами области нормализованных чисел с плавающей точкой.
2. Алгоритм должен быть в большинстве случаев почти столь же быстр, как и приведенная выше программа.
3. Он должен работать на любой «разумной» машине, включая, возможно, и те, арифметика которых отлична от IEEE-арифметики. Это означает, что работа алгоритма не может приводить к останову, если $||x||_2$ не (почти) превосходит наибольшего числа с плавающей точкой.

Чтобы проиллюстрировать имеющиеся здесь трудности, отметим, что указанный выше алгоритм терпит неудачу, если $n = 1$, а x_1 больше квадратного корня из наибольшего числа с плавающей точкой (в этом случае x_1^2 вызывает переполнение, что приводит к выдаче символа $+\infty$ в IEEE-арифметике и останову в большинстве других арифметик), а также если $n = 1$ и x_1 меньше квадратного корня из наименьшего нормализованного числа с плавающей точкой (в этом случае x_1^2 является машинным нулем и алгоритм может выдать нуль в качестве результата). Масштабирование чисел x_i путем деления их на $\max|x_i|$ приводит к потере свойства 2, поскольку деление обычно гораздо дороже умножения или сложения. Если же умножать x_i на $c = 1/\max_i |x_i|$, то возможно переполнение при вычислении c, даже если $\max_i |x_i| > 0$.

Программа вычисления 2-нормы настолько важна, что была включена в число *основных подпрограмм линейной алгебры*, или BLAS (от Basic Linear Algebra Subroutines). Эти подпрограммы должны иметься на любом компьютере [169]. Подробное обсуждение пакета BLAS дается в разд. 2.6.1; документацию к нему и реализации-шаблоны можно найти в NETLIB/blas. В частности, шаблон, находящийся в NETLIB/cgi-bin/netlibget.pl/blas/snrm2.f, обладает свойствами 1 и 3, но не свойством 2. Эти шаблоны задуманы как отправная точка при разработке реализаций для конкретных архитектур (что является более простой задачей, чем порученное вам создание полностью переносимой программы). Итак, при составлении своей программы вы должны думать о вычислении $||x||_2$ как об одном из фрагментов библиотеки численного анализа, имеющейся на любом компьютере.

Тщательно продуманная реализация вычисления 2-нормы описана в [35].

Для проверки правильности своей реализации вы можете воспользоваться тестовой программой из NETLIB/blas/sblat1. Ваш отчет по настоящему заданию должен содержать полностью тестированную реализацию, а также сопоставление времени ее работы с временем приведенного выше «очевидного» алгоритма на тех примерах, где обе программы работают успешно. Интересно, насколько вы сможете приблизиться к выполнению всех выдвинутых условий. Частое использование слова «почти» в их формулировках указывает область

для компромисса: можно частично ослабить одно из условий, чтобы более полно удовлетворить другие. Например, вы можете исследовать, насколько упростится задача, если ограничиться машинами с IEEE-арифметикой.

Указание: считать, что алгоритму известны значения порогов переполнения и машинного нуля. Существуют переносимые программы для вычисления этих значений (см. NETIB/cgi-bin/netlibget.pl/lapack/util/slamch.f).

Вопрос 1.20 (*легкий; средней трудности*). Мы проиллюстрируем чувствительность корней многочлена к малым возмущениям его коэффициентов с помощью Matlab-программы Polyplot, находящейся в HOMEPAGE/Matlab/polyplot.m.[1] Многочлен на входе программы задается набором своих корней. К его коэффициентам программа прибавляет случайные возмущения, вычисляет корни возмущенного многочлена и наносит их на диаграмму. Входными параметрами являются:

r=вектор, составленый из корней многочлена;
e=максимальное относительное возмущение в произвольном коэффициенте многочлена;
m=число возмущенных многочленов, чьи корни выводятся на диаграмму.

2 (*легкий*). Первый пункт вашего задания — применить программу к приведенным ниже входным данным. Во всех случаях выбирайте m достаточно большим, чтобы получилась по-настоящему плотная диаграмма, но не настолько большим, чтобы пришлось ждать результата чересчур долго. Значение m, равное нескольким сотням или, может быть, 1000, вполне подходит. Вы можете изменить масштаб на осях, если разрешение графика слишком мало или слишком велико.

- r=(1:10); e=1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8,
- r=(1:20); e=1e-9, 1e-11, 1e-13, 1e-15,
- r=[2,4,8,16,...,1024]; e=1e-1, 1e-2, 1e-3, 1e-4.

Постройте и свои примеры с комплексно-сопряженными корнями. Какие корни наиболее чувствительны к возмущениям?

3 (*средней трудности*). Второй пункт вашего задания — модифицировать программу так, чтобы она вычисляла число обусловленности c(i) для каждого корня. Иначе говоря, относительное возмущение e в любом коэффициенте не должно изменять корень r(i) более чем на e*c(i). Измените программу таким образом, чтобы она строила круги с центрами r(i) и радиусами e*c(i), и убедитесь, что эти круги содержат корни возмущенных многочленов (по крайней мере, при e настолько малом, что линеаризация, используемая при определении числа обусловленности, правильно отражает положение вещей). Вы должны представить несколько диаграмм с кругами и возмущенными собственными значениями, а также свои объяснения наблюдаемых явлений.

4 (*средней трудности*). Для последней части задания обратите внимание на то, что формула для c(i) теряет смысл, когда ее знаменатель $p'(\mathrm{r(i)})$ равен нулю. Это означает, что r(i) есть кратный корень многочлена $p(x)$. Тем не менее мы можем и для кратного корня надеяться получить какие-то верные

[1] Напомним, что в книге HOMEPAGE используется как сокращение для URL-префикса http://www.siam.arg/books/demmel/demmel_class.

знаки в вычисленном его приближении. В этой части задания выясняется, насколько чувствительным к возмущениям может быть кратный корень многочлена. Положим $p(x) = q(x) \cdot (x - \mathrm{r(i)})^m$, где $q(\mathrm{r(i)}) \neq 0$, а m — кратность корня r(i). Нужно вычислить m корней, ближайших к r(i), для слабо возмущенного многочлена $p(x) - q(x)\varepsilon$ и убедиться, что они отстоят от r(i) на расстоянии $|\varepsilon|^{1/m}$. Если, например, $m = 2$, то корень r(i) возмущается на $\varepsilon^{1/2}$, что много больше, чем ε, если $|\varepsilon| \ll 1$. Бóльшие значения m приводят к еще бóльшим возмущениям. Пусть ε — величина порядка машинного эпсилон, представляющая ошибки округлений в процессе вычисления корня. Из сказанного выше следует, что в случае m-кратного корня лишь $\frac{1}{m}$-я часть разрядов вычисленного приближения будут верными.

Вопрос 1.21 (*средней трудности*). Применить метод бисекции (см. алгоритм 1.1) к вычислению корней уравнения $p(x) = (x-2)^9 = 0$. Значения многочлена $p(x)$ вычисляются по правилу Горнера. Использовать Matlab-программу, находящуюся в HOMEPAGE/Matlab/bisect.m, или составить собственную программу. Убедиться в том, что очень малые изменения входного интервала могут совершенно изменить вычисленное значение корня. Изменить алгоритм таким образом, чтобы процесс деления пополам прекращался, когда ошибка в вычисленном значении $p(x)$ столь велика, что правильный знак многочлена определить не удается. Использовать для этого оценку ошибки, обсуждавшуюся в основном тексте главы.

Глава 2

Решение линейных уравнений

2.1. Введение

В этой главе обсуждаются теория возмущений, алгоритмы и анализ ошибок при решении системы линейных уравнений $Ax = b$. Все алгоритмы являются вариантами гауссова исключения. Они называются *прямыми методами* по той причине, что в отсутствие ошибок округления дают точное решение системы за конечное число шагов. В противоположность этому, в *итерационных методах*, обсуждаемых в гл. 6, строится последовательность $x_0, x_1, x_2, \ldots$ всё лучших приближенных решений системы $Ax = b$; итерации прекращаются (с вычислением следующего вектора x_{i+1}, когда получено достаточно точное приближение x_i. Какой метод, прямой или итерационный, окажется более быстрым или более точным, зависит от матрицы A и скорости, с какой последовательность $\{x_i\}$ сходится к $x = A^{-1}b$. Сравнительные достоинства прямых и итерационных методов подробно обсуждаются в гл. 6. Пока что мы ограничимся указанием, что пользователю следует выбрать прямые методы, если отсутствует информация о происхождении матрицы A[1] или решение системы нужно получить с гарантированной точностью и в заранее известное время.

Остальная часть данной главы организована следующим образом. В разд. 2.2 обсуждается теория возмущений для системы $Ax = b$; на этой теории основываются практические оценки ошибок из разд. 2.4. Алгоритм гауссова исключения для случая плотных матриц, описан в разд. 2.3. В разд. 2.4 анализируются ошибки в гауссовом исключении и выводятся практические оценки для них. В разделе 2.5 показано, как улучшить точность решения, вычисленного гауссовым исключением, посредством простого и экономичного итерационного метода. Чтобы обеспечить высокую скорость работы гауссова исключения и других алгоритмов линейной алгебры, вычисления должны учитывать организацию памяти современных компьютеров. Эта тема обсуждается в разд. 2.6. Наконец, в разд. 2.7 рассмотрены более быстрые варианты гауссова исключения, предназначенные для матриц со специальными свойствами, часто встречающимися в практических задачах, такими как симметрия ($A = A^T$) или разреженность (многие элементы в A являются нулями).

[1] Так, в гл. 6 будет рассмотрен случай, когда A возникает при приближенном решении конкретного дифференциального уравнения, а именно уравнения Пуассона.

В разделах 2.2.1 и 2.5.1 обсуждаются недавние нововведения, используемые программами библиотеки LAPACK.

В ходе изложения мы указываем на ряд открытых вопросов, имеющихся в данной области.

2.2. Теория возмущений

Пусть имеем системы $Ax = b$ и $(A+\delta A)\hat{x} = b+\delta b$; нашей целью является оценка нормы вектора $\delta x \equiv \hat{x}-x$. В дальнейшем в роли $\hat{x}$ будет выступать вычисленное решение системы $Ax = b$. Мы попросту вычтем одно из указанных равенств из другого, а затем найдем δx. Вот как это можно сделать: после вычитания

$$\begin{array}{rl} & (A+\delta A)(x+\delta x) = b+\delta b \\ - & \qquad [Ax = b] \\ \hline & \delta Ax + (A+\delta A)\delta x = \delta b \end{array}$$

приведем полученное соотношение к виду

$$\delta x = A^{-1}(-\delta A\hat{x} + \delta b). \tag{2.1}$$

Беря здесь нормы и используя первую часть леммы 1.7, а также неравенство треугольника для векторных норм, находим

$$\|\delta x\| \le \|A^{-1}\|(\|\delta A\| \cdot \|\hat{x}\| + \|\delta b\|). \tag{2.2}$$

(Предполагается, что векторная и матричная нормы согласованы в том смысле, как это определено в разд. 1.7. Например, можно взять произвольную векторную норму и индуцированную ею матричную норму.) Трансформируя это неравенство, получаем

$$\frac{\|\delta x\|}{\|\hat{x}\|} \le \|A^{-1}\| \cdot \|A\| \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\hat{x}\|}\right). \tag{2.3}$$

Произведение $\kappa(A) = \|A^{-1}\| \cdot \|A\|$ называется *числом обусловленности* матрицы A[1], поскольку оно оценивает относительное изменение $\|\delta x\|/\|\hat{x}\|$ как кратное относительного изменения $\|\delta A\|/\|A\|$ во входных данных. (Строго говоря, мы должны были бы показать, что неравенство (2.2) превращается в равенство при некотором выборе возмущений δA и δb; в противном случае κ(A) было бы лишь верхней оценкой для числа обусловленности. См. по этому поводу вопрос 2.3.) Если δA и δb малы, то мала и величина, на которую умножается $\kappa(A)$, что приводит к малой верхней границе для относительной ошибки $\|\delta x\|/\|\hat{x}\|$.

Наша верхняя оценка зависит от δx (входящего в $\hat{x}$), что, по видимости, затрудняет ее интерпретацию. В действительности, однако, эта оценка весьма полезна с практической точки зрения, поскольку вычисленное решение $\hat{x}$ известно, следовательно, и оценка легко может быть вычислена. Мы покажем теперь, как вывести теоретически более привлекательную оценку, не зависящую от δx.

[1] С более педантической точки зрения, это число обусловленности для задачи обращения матрицы. Задаче вычисления собственных значений матрицы A соответствует, например, другое число обусловленности.

Лемма 2.1. *Пусть выбранная матричная норма* $\|\cdot\|$ *обладает свойством* $\|AB\| \le \|A\|\,\|B\|$. *Тогда, если* $\|X\| < 1$, *то матрица* $I - X$ *обратима,* $(I - X)^{-1} = \sum_{i=0}^{\infty} X^i$ *и* $\|(I-X)^{-1}\| \le \frac{1}{1-\|X\|}$.

Доказательство. Матричная сумма $\sum_{i=0}^{\infty} X^i$ сходится тогда и только тогда, когда сходится каждый элемент этой матрицы. Используем тот факт (получаемый применением леммы 1.4 к примеру 1.6), что для произвольной нормы существует константа c, такая, что $|x_{jk}| \le c\|X\|$. Тогда $|(X^i)_{jk}| \le c\|X^i\| \le c\|X\|^i$, т. е. каждый элемент матрицы $\sum X^i$ мажорируется сходящейся геометрической прогрессией $\sum c\|X\|^i = \frac{c}{1-\|X\|}$ и, следовательно, сам сходится. Поэтому последовательность $S_n = \sum_{i=0}^{n} X^i$ сходится при $n \to \infty$ к некоторому S и $(I-X)S_n = (I-X)(I+X^2_X + \cdots + X^n) = I - X^{n+1} \to I$ при $n \to \infty$, поскольку $\|X^i\| \le \|X\|^i \to 0$. Таким образом, $(I-X)S = I$ и $S = (I-X)^{-1}$. Заключительная оценка выводится так: $\|(I-X)^{-1}\| = \|\sum_{i=0}^{\infty} X^i\| \le \sum_{i=0}^{\infty} \|X^i\| \le \sum_{i=0}^{\infty} \|X\|^i = \frac{1}{1-\|X\|}$. □

Разрешая наше первое уравнение $\delta A x + (A + \delta A)\delta x = \delta b$ относительно δx, получаем

$$\begin{aligned}\delta x &= (A+\delta A)^{-1}(-\delta A x + \delta b)\\ &= [A(I + A^{-1}\delta A)]^{-1}(-\delta A x + \delta b)\\ &= (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta A x + \delta b).\end{aligned}$$

Беря здесь нормы, деля обе части на $\|x\|$, используя первую часть леммы 1.7 и неравенство треугольника, предполагая, наконец, что δA настолько мало, что $\|A^{-1}\delta A\| \le \|A^{-1}\|\,\|\delta A\| < 1$, приходим к желаемой оценке:

$$\begin{aligned}\frac{\|\delta x\|}{\|x\|} &\le \|(I + A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \left(\|\delta A\| + \frac{\|\delta b\|}{\|x\|}\right)\\ &\le \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|}\left(\|\delta A\| + \frac{\|\delta b\|}{\|x\|}\right) \quad \text{по лемме 2.1}\\ &= \frac{\|A^{-1}\| \cdot \|A\|}{1 - \|A^{-1}\| \cdot \|A\| \frac{\|\delta A\|}{\|A\|}}\left(\frac{\|dA\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|x\|}\right)\\ &\le \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right),\end{aligned} \tag{2.4}$$

поскольку $\|b\| = \|Ax\| \le \|A\| \cdot \|x\|$.

Эта оценка выражает относительную ошибку решения $\|\delta x\|/\|x\|$ как кратное относительных ошибок $\|\delta A\|/\|A\|$ и $\|\delta b\|/\|b\|$ во входных данных. Если величина $\|\delta A\|$ достаточно мала, то множитель $\kappa(A)/\left(1 - \kappa(A)\frac{\|\delta A\|}{\|A\|}\right)$ близок к числу обусловленности $\kappa(A)$.

Следующая теорема полнее раскрывает смысл предположения $\|A^{-1}\| \cdot \|\delta A\| = \kappa(A)\frac{\|\delta A\|}{\|A\|} < 1$: оно гарантирует, что матрица $A + \delta A$ невырожденна; это необходимо для существования δx. Теорема также дает геометрическую характеризацию числа обусловленности.

Теорема 2.1. *Пусть матрица A невырожденна. Тогда*

$$\min\left\{\frac{\|\delta A\|_2}{\|A\|_2} : A+\delta A \text{ вырожденна}\right\} = \frac{1}{\|A^{-1}\|_2\cdot\|A\|_2} = \frac{1}{\kappa(A)}.$$

Следовательно, расстояние до ближайшей вырожденной матрицы (некорректная задача) = 1/(число обусловленности).

Доказательство. Достаточно показать, что

$$\min\{\|\delta A\|_2 : A+\delta A \text{ вырожденна}\} = \frac{1}{\|A^{-1}\|_2}.$$

Чтобы убедиться, что указанный минимум не меньше, чем $\|A^{-1}\|_2^{-1}$, заметим: если $\|\delta A\|_2 < \|A^{-1}\|_2^{-1}$, то $1 > \|\delta A\|_2\,\|A^{-1}\|_2 \geq \|A^{-1}\delta A\|_2$. Согласно лемме 2.1, матрица $I + A^{-1}\delta A$ обратима, а потому обратима матрица $A+\delta A$.

Чтобы показать, что минимум равен $\|A^{-1}\|_2^{-1}$, построим возмущение δA с нормой $\|A^{-1}\|_2^{-1}$, такое, что матрица $A+\delta A$ вырожденна. Поскольку $\|A^{-1}\|_2 = \max_{x\neq 0}\frac{\|A^{-1}x\|_2}{\|x\|_2}$, найдется вектор x, такой, что $\|x\|_2 = 1$ и $\|A^{-1}\|_2 = \|A^{-1}x\| > 0$. Положим теперь $y = \frac{A^{-1}x}{\|A^{-1}x\|_2} = \frac{A^{-1}x}{\|A^{-1}\|}$; таким образом, $\|y\|_2 = 1$. Положим $\delta A = \frac{-xy^T}{\|A^{-1}\|_2}$.

Тогда

$$\|\delta A\|_2 = \max_{z\neq 0}\frac{\|xy^Tz\|_2}{\|A^{-1}\|_2\|z\|_2} = \max_{z\neq 0}\frac{|y^Tz|}{\|z\|_2}\frac{\|x\|_2}{\|A^{-1}\|_2} = \frac{1}{\|A^{-1}\|_2},$$

где максимум достигается, когда z есть произвольное ненулевое кратное вектора y. Матрица $A+\delta A$ вырожденна, так как

$$(A+\delta A)y = Ay - \frac{xy^Ty}{\|A^{-1}\|_2} = \frac{x}{\|A^{-1}\|_2} - \frac{x}{\|A^{-1}\|_2} = 0. \qquad \square$$

Мы видим теперь, что расстояние до ближайшей некорректной задачи и число обусловленности взаимно обратны для двух задач: вычисления многочлена и решения линейных уравнений. Подобная взаимная обратность весьма характерна для численного анализа [71].

Имеется несколько иной способ построения теории возмущений для задачи $Ax = b$; позже, в разд. 2.4.4 этот способ потребуется нам для вывода практических оценок ошибки. Пусть $\hat{x}$ — произвольный вектор, тогда разность $\delta x \equiv \hat{x} - x = \hat{x} - A^{-1}b$ может быть оценена следующим образом. Назовем вектор $r = A\hat{x} - b$ *невязкой* вектора $\hat{x}$; невязка r равна нулю, если $\hat{x} = x$. Это позволяет нам написать $\delta x = A^{-1}r$ и получить оценку

$$\|\delta x\| = \|A^{-1}r\| \leq \|A^{-1}\|\cdot\|r\|. \tag{2.5}$$

Эта простая оценка привлекательна с практической точки зрения, поскольку r легко вычисляется, если дано приближенное решение $\hat{x}$. Кроме того, нет видимой нужды в оценивании δA и δb. В действительности, оба описанных подхода весьма тесно связаны, что выявляется следующей теоремой.

Теорема 2.2. *Пусть $r = A\hat{x} - b$. Тогда найдется возмущение δA, такое, что $\|\delta A\| = \frac{\|r\|}{\|\hat{x}\|}$ и $(A+\delta A)\hat{x} = b$. Не существует никакого возмущения δA меньшей*

нормы, удовлетворяющего уравнению $(A+\delta A)\hat{x}=b$. *Таким образом,* δA *есть наименьшая возможная обратная ошибка (измеряемая посредством нормы). Это утверждение справедливо для любой векторной нормы и порожденной ею матричной нормы (или выборе* $\|\cdot\|_2$ *для векторов и* $\|\cdot\|_F$ *для матриц).*

Доказательство. Равенство $(A+\delta A)\hat{x}=b$ эквивалентно соотношениям $\delta A\hat{x}=b-A\hat{x}=-r$, поэтому $\|r\|=\|\delta A\cdot\hat{x}\|\le\|\delta A\|\cdot\|\hat{x}\|$, откуда $\|\delta A\|\ge\frac{\|r\|}{\|\hat{x}\|}$. Остальную часть доказательства мы проведем только для 2-нормы и порожденной ею матричной нормы. Положим $\delta A=\frac{-r\hat{x}^T}{\|\hat{x}\|_2^2}$. Легко проверить, что $\delta A\hat{x}=-r$ и $\|\delta A\|_2=\frac{\|r\|_2}{\|\hat{x}\|_2}$. □

Таким образом, наименьшее значение $\|\delta A\|$, позволяющее получить вектор $\hat{x}$, удовлетворяющий условиям $(A+dA)\hat{x}=b$ и $r=A\hat{x}-b$, указывается теоремой 2.2. Привлекая оценку (2.2) (с $\delta b=0$), получаем

$$\|\delta x\|\le\|A^{-1}\|\left(\frac{\|r\|}{\|\hat{x}\|}\cdot\|\hat{x}\|\right)=\|A^{-1}\|\cdot\|r\|,$$

т. е. приходим к оценке (2.5).

Все наши оценки зависят от возможности оценить число обусловленности $\|A\|\cdot\|A^{-1}\|$. Мы вернемся к этой задаче в разд. 2.4.3. Оценки чисел обусловленности вычисляются программами библиотеки LAPACK, например программой `sgesvx`.

2.2.1. Теория относительных возмущений

В предыдущем разделе мы показали, как оценить норму ошибки $\delta x=\hat{x}-x$ приближенного решения $\hat{x}$ системы $Ax=b$. Наша оценка для $\|\delta x\|$ была пропорциональна числу обусловленности $\kappa(A)=\|A\|\cdot\|A^{-1}\|$ и нормам $\|\delta A\|$ и $\|\delta b\|$ в предположении, что $\hat{x}$ удовлетворяет уравнению $(A+\delta A)\hat{x}=b+\delta b$.

Во многих случаях эта оценка вполне удовлетворительна, но все же не всегда. Наша цель в данном разделе — указать ситуации, когда эта оценка слишком пессимистична, и развить альтернативную теорию возмущений, дающую лучшие оценки. Позднее, в разд. 2.5.1 эта теория будет использована для обоснования оценок ошибки, вычисляемых подпрограммами библиотеки LAPACK, такими, как `sgesvx`.

При первом чтении книги этот раздел может быть опущен.

Вот пример ситуации, где оценка ошибки из предыдущего раздела чересчур пессимистична.

Пример 2.1. Пусть $A=\mathrm{diag}(\gamma,1)$ (диагональная матрица с диагональными элементами $a_{11}=\gamma$ и $a_{22}=1$) и $b=[\gamma,1]^T$, где $\gamma>1$. Тогда $x=A^{-1}b=[1,1]^T$. Всякий разумный прямой метод вычислит очень точное приближение $\hat{x}$ к решению системы $Ax=b$ (посредством двух делений b_i/a_{ii}); в то же время число обусловленности $\kappa(A)=\gamma$ может быть как угодно велико. Следовательно, как угодно велика может быть и наша оценка ошибки (2.3).

Причина, почему число обусловленности $\kappa(A)$ заставляет нас переоценивать ошибку, состоит в следующем: оценка (2.2), содержащая это число, основана на предположении, что возмущение δA ограничено по норме, но *в остальном*

произвольно; на последнем свойстве построено доказательство достижимости оценки (2.2) в вопросе 2.3. В противоположность этому, возмущения δA, отвечающие реальным ошибкам округления, не являются произвольными, а имеют специальную структуру, не отражаемую нормой самой по себе. Наименьшее δA, соответствующее в нашей задаче вектору $\hat{x}$, можно определить так: простой анализ ошибок округления показывает, что $\hat{x}_i = (b_i/a_{ii})/(1+\delta_i)$, где $|\delta_i| \le \varepsilon$. Тем самым $(a_{ii} + \delta_i a_{ii})\hat{x}_i = b_i$. Это можно переписать как $(A + \delta A)\hat{x} = b$, где $\delta A = \operatorname{diag}(\delta_1 a_{11}, \delta_2 a_{22})$. Тогда $\|\delta A\|$ может достигать величины $\max_i |\varepsilon a_{ii}| = \varepsilon\gamma$. Применяя оценку ошибки (2.3) с $\delta b = 0$, находим

$$\frac{\|\delta x\|_\infty}{\|\hat{x}\|_\infty} \le \gamma\left(\frac{\varepsilon\gamma}{\gamma}\right) = \varepsilon\gamma.$$

В отличие от этого, реальная ошибка удовлетворяет соотношениям

$$\begin{aligned}
\|\delta x\|_\infty &= \|\hat{x} - x\|_\infty \\
&= \left\| \begin{bmatrix} (b_1/a_{11})/(1+\delta_1) - (b_1/a_{11}) \\ (b_2/a_{22})/(1+\delta_2) - (b_2/a_{22}) \end{bmatrix} \right\|_\infty \\
&= \left\| \begin{bmatrix} -\delta_1/(1+\delta_1) \\ -\delta_2/(1+\delta_2) \end{bmatrix} \right\|_\infty \\
&\le \frac{\varepsilon}{1-\varepsilon}
\end{aligned}$$

или

$$\frac{\|\delta x\|_\infty}{\|\hat{x}\|_\infty} \le \varepsilon/(1-\varepsilon)^2,$$

что примерно в γ раз меньше предыдущей оценки. ◇

Для данного примера структуру реального возмущения δA можно описать следующим образом: $|\delta a_{ij}| \le \varepsilon |a_{ij}|$, где ε — некоторое очень малое число. В более сжатой форме это можно описать как

$$|\delta A| \le \varepsilon |A| \tag{2.6}$$

(по поводу обозначений см. разд. 1.1). Мы говорим в таких случаях, что δA есть *малое относительное покомпонентное возмущение* матрицы A. Поскольку на практике часто можно добиться того, чтобы δA удовлетворяла оценке (2.6), а δb — оценке $|\delta b| \le \varepsilon |b|$ (см. разд. 2.5.1), то мы построим теорию возмущений, основываясь на этих оценках для δA и δb.

Начнем с уравнения (2.1):

$$\delta x = A^{-1}(-\delta A\hat{x} + \delta b).$$

Переходя к абсолютным величинам и применяя неравенство треугольника, получаем

$$\begin{aligned}
|\delta x| &= |A^{-1}(-\delta A\hat{x} + \delta b| \\
&\le |A^{-1}|(|\delta A| \cdot |\hat{x}| + |\delta b|) \\
&\le |A^{-1}|(\varepsilon |A| \cdot |\hat{x}| + \varepsilon |b|) \\
&= \varepsilon(|A^{-1}|)(|A| \cdot |\hat{x}| + |b|)).
\end{aligned}$$

Используя любую векторную норму, для которой $|||z||| = \|z\|$ (таковы max-норма, 1-норма и норма Фробениуса) приходим к оценке

$$\|\delta x\| \le \varepsilon \||A^{-1}|(|A| \cdot |\hat{x}| + |b|)\|. \tag{2.7}$$

Предположим на время, что $\delta b = 0$. Тогда (2.7) можно ослабить до оценки

$$\|\delta x\| \le \varepsilon \||A^{-1}| \cdot |A|\| \cdot \|\hat{x}\|$$

или

$$\frac{\|\delta x\|}{\|\hat{x}\|} \le \varepsilon \||A^{-1}| \cdot |A|\|. \tag{2.8}$$

Это неравенство служит обоснованием для того, чтобы назвать величину $\kappa_{CR}(A) \equiv \||A^{-1}| \cdot |A|\|$ *относительным покомпонентным числом обусловленности* матрицы A или, для краткости, ее *относительным числом обусловленности*. Иногда эту величину называют также числом обусловленности Бауэра [26] или числом обусловленности Скила [225, 226, 227]. По поводу доказательства того, что оценки (2.7) и (2.8) достижимы, см. вопрос 2.4.

Напомним, что теорема 2.1 устанавливает связь между числом обусловленности $\kappa(A)$ и расстоянием от A до ближайшей вырожденной матрицы. Относительно аналогичной интерпретации числа $\kappa_{CR}(A)$ см. [72, 208].

Пример 2.2. Вернемся к нашему предыдущему примеру с $A = \mathrm{diag}(\gamma, 1)$ и $b = [\gamma, 1]^T$. Легко проверить, что $\kappa_{CR}(A) = 1$, поскольку $|A^{-1}||A| = I$. В действительности, $\kappa_{CR}(A) = 1$ для любой диагональной матрицы A, что соответствует нашему интуитивному ощущению, согласно которому диагональные системы уравнений должны решаться с высокой точностью. ◇

Рассмотрим более общий случай, где D — произвольная невырожденная диагональная матрица, а B — произвольная невырожденная матрица. Тогда

$$\begin{aligned}\kappa_{CR}(DB) &= \||(DB)^{-1}| \cdot |(DB)|\| \\ &= \||B^{-1}D^{-1}| \cdot |DB|\| \\ &= \||B^{-1}| \cdot |B|\| \\ &= \kappa_{CR}(B).\end{aligned}$$

Это означает, что если матрица DB *плохо масштабирована*, т. е. B — хорошо обусловленная матрица, а DB обусловлена плохо (из-за того, что диагональные элементы в D сильно различаются по величине), то мы должны надеяться на возможность решения системы $(DB)x = b$ с высокой точностью, несмотря на плохую обусловленность матрицы DB. Эта тема обсуждается в дальнейшем в разд. 2.4.4, 2.5.1 и 2.5.2.

В заключение мы приведем, как и в предыдущем разделе, оценку ошибки, использующую только невязку $r = A\hat{x} - b$:

$$\|\delta x\| = \|A^{-1}r\| \le \||A^{-1}| \cdot |r|\|. \tag{2.9}$$

Здесь было применено неравенство треугольника. В разд. 2.4.4 мы увидим, что эта оценка может быть подчас много меньше аналогичной оценки (2.5); так будет, в частности, если A плохо масштабирована. Справедливо, кроме того, утверждение, аналогичное теореме 2.2 [193].

Теорема 2.3. *Наименьшее число $\varepsilon > 0$, для которого существуют возмущения δA и δb, удовлетворяющие оценкам $|\delta A| \le \varepsilon |A|$ и $|\delta b| \le \varepsilon |b|$ и уравнению $(A + \delta A)\hat{x} = b + \delta b$, называется покомпонентной относительной обратной ошибкой. Оно выражается через невязку $r = A\hat{x} - b$ формулой*

$$\varepsilon = \max_i \frac{|r_i|}{(|A| \cdot |\hat{x}| + |b|)_i}.$$

Относительно доказательства теоремы см. вопрос 2.5.

Программы библиотеки LAPACK, такие, как `sgesvx`, вычисляют покомпонентную относительную обратную ошибку ε (для соответствующей переменной в LAPACK'е принято имя `BERR`).

2.3. Гауссово исключение

Основным алгоритмом решения линейных систем $Ax = b$ является *гауссово исключение*. Чтобы описать его, нам нужно вначале определить понятие *матрицы-перестановки*.

Определение 2.1. Матрица-перестановка P — *это матрица, полученная из единичной произвольной перестановкой строк.*

Наиболее важные свойства матриц-перестановок даются следующей леммой.

Лемма 2.2. *Пусть P, P_1 и P_2 — это матрицы-перестановки порядка n, а X — произвольная $n \times n$-матрица. Тогда*

1. *PX отличается от X только порядком строк. XP отличается от X только порядком столбцов.*
2. $P^{-1} = P^T$.
3. $\det(P) = \pm 1$.
4. *$P_1 \cdot P_2$ также является матрицей-перестановкой.*

Относительно доказательства см. вопрос 2.6.

Теперь можно полностью описать наш алгоритм для решения системы $Ax = b$.

Алгоритм 2.1. *Решение системы $Ax = b$ посредством гауссова исключения.*

1. *Разложить A в произведение $A = PLU$, где*
   P — матрица-перестановка,
   L — нижняя унитреугольная матрица (т. е. матрица с единицами на главной диагонали),
   U — невырожденная верхнетреугольная матрица.
2. *Решить систему $PLUx = b$ относительно вектора LUx, переставляя компоненты вектора b: $LUx = P^{-1}b = P^Tb$.*
3. *Решить систему $LUx = P^{-1}b$ относительно вектора Ux посредством прямой подстановки (прямого хода): $Ux = L^{-1}(P^{-1}b)$.*
4. *Решить систему $U_x = L^{-1}(P^{-1}b)$ относительно вектора x посредством обратной подстановки: $x = U^{-1}(L^{-1}P^{-1}b)$.*

Алгоритм для разложения $A = PLU$ будет выведен несколькими способами. Начнем с пояснения, почему нужна матрица-перестановка P.

Определение 2.2. Ведущей главной подматрицей порядка j *матрицы* A *называется подматрица* $A(1:j, 1:j)$.

Теорема 2.4. *Следующие два утверждения эквивалентны:*

1. *Существуют единственные нижняя унитреугольная матрица* L *и невырожденная верхнетреугольная матрица* U, *такие, что* $A = LU$.
2. *Все ведущие главные подматрицы матрицы* A *невырождены.*

Доказательство. Покажем вначале, что из первого утверждения следует второе. Равенство $A = LU$ может еще быть записано как

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix},$$

где A_{11}, L_{11} и U_{11} — ведущие главные подматрицы порядка j соответствующих матриц. Следовательно, $\det A_{11} = \det(L_{11}U_{11}) = \det L_{11} \det U_{11} = 1 \cdot \prod_{k=1}^{j} (U_{11})_{kk} \neq 0$, поскольку L унитреугольная, а U треугольная матрицы.

Докажем, что из второго утверждения следует первое, индукцией по n. Это тривиально для 1×1-матриц: $a = 1 \cdot a$. Переходя к $n \times n$-матрице $\tilde{A}$, мы должны найти однозначно определенные треугольные матрицы L и U порядка $n-1$, однозначно определенные векторы l и u размерности $n-1$ и однозначно определенное число η, такие, что

$$\tilde{A} = \begin{bmatrix} A & b \\ c^T & \delta \end{bmatrix} = \begin{bmatrix} L & 0 \\ l^T & 1 \end{bmatrix} \begin{bmatrix} U & u \\ 0 & \eta \end{bmatrix} = \begin{bmatrix} LU & Lu \\ l^T U & l^T u + \eta \end{bmatrix}.$$

По предположению индукции, существуют единственные матрицы L и U, для которых $A = LU$. Положим теперь $u = L^{-1}b$, $l^T = c^T U^{-1}$ и $\eta = \delta - l^T u$; все эти величины определены единственным образом. Согласно предположению индукции, диагональные элементы матрицы U отличны от нуля, а $\eta \neq 0$ по той причине, что $0 \neq \det(\tilde{A}) = \det(U) \cdot \eta$. □

Таким образом, LU-разложение без выбора главного элемента может оказаться невозможным для (хорошо обусловленных) невырожденных матриц, например для матрицы-перестановки.

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

В матрице P вырожденны ведущие главные подматрицы порядков 1 и 2. Это означает, что мы должны ввести перестановки в гауссово исключение.

Теорема 2.5. *Для невырожденной матрицы* A *существуют перестановки* P_1 *и* P_2, *нижняя унитреугольная матрица* L *и невырожденная верхнетреугольная матрица* U, *такие, что* $P_1 A P_2 = LU$. *Можно ограничиться одной из двух матриц* P_1 *и* P_2.

Замечание: *P_1A переупорядочивает строки в A, тогда как AP_2 переупорядочивает столбцы; P_1AP_2 переупорядочивает и строки, и столбцы.*

Доказательство. Подобно многим матричным разложениям, достаточно разобраться со случаем блочных 2×2-матриц. Говоря более формально, мы применим индукцию по порядку n. Утверждение очевидно для 1×1-матриц: $P_1 = P_2 = L = 1$ и $U = A$. Предположим, что утверждение верно для порядка $n-1$. Невырожденная матрица A должна иметь ненулевые элементы, поэтому выберем перестановки P_1' и P_2' так, чтобы элемент (1, 1) матрицы $P_1'AP_2'$ был отличен от нуля. (Достаточно использовать только одну из матриц P_1' и P_2', поскольку невырожденность A означает, что в каждой ее строке и каждом столбце имеются ненулевые элементы.)

Теперь мы запишем требуемое разложение, а затем его неизвестные компоненты:

$$P_1'AP_2' = \begin{bmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \cdot \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & U_{12} \\ L_{21}u_{11} & L_{21}U_{12} + \tilde{A}_{22} \end{bmatrix}. \tag{2.10}$$

Здесь A_{22} и $\tilde{A}_{22}$ — матрицы порядка $n-1$, а L_{21} и U_{12}^T имеют размер $(n-1) \times 1$.

Определяя компоненты этого блочного 2×2-разложения, получаем $u_{11} = a_{11} \neq 0$, $U_{12} = A_{12}$ и $L_{21}u_{11} = A_{21}$. Поскольку $u_{11} = a_{11} \neq 0$, можно найти $L_{21} = \frac{A_{21}}{a_{11}}$. Наконец, из $L_{21}U_{12} + \tilde{A}_{22} = A_{22}$ следует, что $\tilde{A}_{22} = A_{22} - L_{21}U_{12}$.

Мы хотели бы применить к $\tilde{A}_{22}$ предположение индукции; однако, чтобы сделать это, нужно проверить, что $\det \tilde{A}_{22} \neq 0$. Поскольку $\det P_1'AP_2' = \pm \det A \neq 0$, а также

$$\det P_1'AP_2' = \det \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \cdot \det \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} = 1 \cdot (u_{11} \cdot \det \tilde{A}_{22}),$$

то число $\det \tilde{A}_{22}$ должно быть ненулевым.

Итак, по предположению индукции, найдутся перестановки $\tilde{P}_1$ и $\tilde{P}_2$, такие, что $\tilde{P}_1\tilde{A}_{22}\tilde{P}_2 = \tilde{L}\tilde{U}$, где L нижняя унитреугольная, а U невырожденная верхнетреугольная матрицы. Подставляя это равенство в написанное выше блочное 2×2-разложение, получаем

$$\begin{aligned} P_1'AP_2' &= \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{P}_1^T\tilde{L}\tilde{U}\tilde{P}_2^T \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1^T\tilde{L} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{U}\tilde{P}_2^T \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ L_{21} & \tilde{P}_1^T\tilde{L} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12}\tilde{P}_2 \\ 0 & \tilde{U} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_2^T \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1^T \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \tilde{P}_1L_{21} & \tilde{L} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12}\tilde{P}_2 \\ 0 & \tilde{U} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_2^T \end{bmatrix}, \end{aligned}$$

что дает требуемое разложение матрицы A:

$$P_1 A P_2 = \left(\begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1 \end{bmatrix} P_1' \right) A \left(P_2' \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_2 \end{bmatrix} \right)$$
$$= \begin{bmatrix} 1 & 0 \\ \tilde{P}_1 L_{21} & \tilde{L} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12}\tilde{P}_2 \\ 0 & \tilde{U} \end{bmatrix}. \square$$

В следующих двух предложениях описаны простые способы выбора перестановок P_1 и P_2, гарантирующих, что гауссово исключение будет успешно проведено для невырожденной матрицы A.

Следствие 2.1. *Можно положить $P_2' = I$ и выбрать P_1' таким образом, чтобы a_{11} был наибольшим по абсолютной величине элементом своего столбца; это означает, что элементы матрицы $L_{21} = \frac{A_{21}}{a_{11}}$ ограничены по абсолютной величине числом 1. Более общо, при вычислении i-го столбца матрицы L на i-м шаге гауссова исключения строки с i-й по n-ю переупорядочиваются так, чтобы наибольший по абсолютной величине элемент i-го столбца находился на главной диагонали. Такая организация процесса называется «гауссовым исключением с частичным выбором главного элемента» или, для краткости, методом* GEPP. *Этот метод гарантирует, что все элементы матрицы L ограничены по абсолютной величине числом 1.*

Метод GEPP — это наиболее распространенный способ практической реализации гауссова исключения. В следующем разделе мы обсудим его численную устойчивость. Другой, более дорогостоящий способ выбора P_1 и P_2 указан в следствии 2.2. Этот способ почти никогда не используется на практике, хотя и существуют немногочисленные примеры, когда он дает решения хорошей точности, между тем как метод GEPP «проваливается» (см. вопрос 2.14). Новый способ также кратко обсуждается в следующем разделе.

Следствие 2.2. *Можно выбрать P_1' и P_2' таким образом, чтобы a_{11} был наибольшим по абсолютной величине элементом всей матрицы. Более общо, при вычислении i-го столбца матрицы L на i-м шаге гауссова исключения строки и столбцы с номерами от i до n переупорядочиваются так, чтобы наибольший по абсолютной величине элемент образованной ими подматрицы находился в позиции (i, i). Такая организация процесса называется «гауссовым исключением с полным выбором главного элемента» или, для краткости, методом* GECP.

Следующий алгоритм реализует теорему 2.5, выполняя перестановки, вычисляя первый столбец матрицы L и первую строку матрицы U и перестраивая A_{22} в матрицу $\tilde{A}_{22} = A_{22} - L_{21}U_{12}$. Мы запишем алгоритм вначале в привычных обозначениях языков программирования, а затем в обозначениях Matlab'а.

Алгоритм 2.2. *LU-разложение с выбором главного элемента:*

for $i = 1$ *to* $n - 1$
использовать перестановки, чтобы обеспечить условие $a_{ii} \neq 0$
(перестановки применяются и к матрицам L и U)
/* *так, в методе GEPP меняются местами строки j и i матриц A и L, где $|a_{ji}|$ — наибольший элемент в $|A(i:n, i)|$; в методе GECP меняются местами строки j и i матриц*

```
            A и L, а также столбцы k и i матриц A и U,
            где |a_jk| — наибольший элемент в |A(i : n, i : n)| */
    /* вычислить столбец i матрицы L (матрицу L_21 в (2.10)) */
    for j = i + 1 to n
        l_ji = a_ji/a_ii
    end for
    /* вычислить строку i матрицы U (матрицу U_12 в (2.10)) */
    for j = i to n
        u_ij = a_ij
    end for
    /* обновить A_22 (чтобы получить матрицу
    Ã_22 = A_22 − L_21 U_12 в (2.10)) */
    for j = i + 1 to n
        for k = i + 1 to n
            a_jk = a_jk − l_ji * u_ik
        end for
    end for
end for
```

Заметим, что после того как i-й столбец в A использован для вычисления i-го столбца матрицы L, он нигде не используется в дальнейшем. Точно так же, i-я строка в A не используется после того, как вычислена i-я строка матрицы U. Это позволяет нам записывать L и U по мере того, как они вычисляются, на место матрицы A, так что для хранения этих матриц не требуется дополнительной памяти. При этом L занимает нижний треугольник в A (без главной диагонали, поскольку диагональные единицы в L не хранятся явно), а U занимает верхний треугольник. В результате приходим к следующему упрощенному алгоритму.

Алгоритм 2.3. *LU-разложение с выбором главного элемента и записью L и U на место A:*

```
for i = 1 to n − 1
    использовать перестановки (см. детали в алгоритме 2.2)
    for j = i + 1 to n
        a_ji = a_ji/a_ii
    end for
    for j = i + 1 to n
        for k = i + 1 to n
        a_jk = a_jk − a_ji * a_ik
        end for
    end for
end for
```

Использование обозначений Matlab'а приводит к такому алгоритму:

Алгоритм 2.4. *LU-разложение с выбором главного элемента и записью L и U на место A:*

```
for i = 1 to n − 1
    использовать перестановки (см. детали в алгоритме 2.2)
```

$A(i+1:n,i) = A(i+1:n,i)/A(i,i)$
$A(i+1:n,i+1:n) =$
$\quad A(i+1:n,i+1:n) - A(i+1:n,i) * A(i,i+1:n)$
end for

В последней строке алгоритма $A(i+1:n,i) * A(i,i+1:n)$ есть произведение матрицы размерности $(n-i)\times 1$ (матрицы L_{21}) и матрицы размерности $1\times(n-i)$ (матрицы U_{12}); оно представляет собой квадратную матрицу порядка $n-i$.

Теперь мы выведем данный алгоритм заново, отправляясь от, видимо, самого известного описания гауссова исключения: «взять по очереди каждую строку и вычитать ее кратные из последующих строк с тем, чтобы аннулировать поддиагональные элементы». Прямой перевод этого описания дает следующий алгоритм:

for $i = 1$ to $n-1$ /* для каждой строки i */
  for $j = i+1$ to n /* вычесть кратное строки i из строки j */
  for $k = i$ to n /* ... для столбцов с номерами от i до n ... */
    $a_{jk} = a_{jk} - \frac{a_{ji}}{a_{ii}} a_{ik}$ /* ... аннулировать поддиагональные элементы столбца i */
    end for
  end for
end for

Произведем теперь некоторые усовершенствования в этом алгоритме, изменяя его так, что в конечном счете он окажется идентичен алгоритму 2.3 (кроме выбора главного элемента, который мы опускаем). Обратим, прежде всего, внимание на то, что нет нужды вычислять нулевые элементы под главной диагональю, заранее зная, что они нулевые. Это сокращает цикл по k и приводит к такому алгоритму:

for $i = 1$ to $n-1$
  for $j = i+1$ to n
    for $k = i+1$ to n
      $a_{jk} = a_{jk} - \frac{a_{ji}}{a_{ii}} a_{ik}$
    end for
  end for
end for

Следующее усовершенствование состоит в том, чтобы вычислять отношение a_{ji}/a_{ii} вне внутреннего цикла, поскольку оно не зависит от индекса k этого цикла:

for $i = 1$ to $n-1$
  for $j = i+1$ to n
    $l_{ji} = \frac{a_{ji}}{a_{ii}}$
  end for
  for $j = i+1$ to n
    for $k = i+1$ to n
      $a_{jk} = a_{jk} - l_{ji} a_{ik}$
    end for
  end for
end for

Наконец, мы помещаем множители l_{ji} на места поддиагональных элементов a_{ji}, которые перед тем были аннулированы; последние нигде уже больше не используются. В результате получаем алгоритм 2.3 (с точностью до выбора главного элемента).

Число операций в LU-разложении можно подсчитать, заменяя циклы суммированиями в тех же пределах, а содержимое внутренних циклов — числом операций в них:

$$\sum_{i=1}^{n-1}\left(\sum_{j=i+1}^{n} 1+\sum_{j=i+1}^{n}\sum_{k=i+1}^{n} 2\right)$$
$$=\sum_{i=1}^{n-1}((n-i)+2(n-i)^2)=\frac{2}{3}n^3+O(n^2).$$

Прямая подстановка и обратный ход, выполняемые для L и U с тем, чтобы завершить решение системы $Ax=b$, требуют $O(n^2)$ операций, поэтому полная стоимость решения системы посредством гауссова исключения составляет $\frac{2}{3}n^3+O(n^2)$ операций. Здесь использована формула $\sum_{i=1}^{m} i^k = m^{k+1}/(k+1)+O(m^k)$. Она достаточна для того, чтобы определить член наивысшего порядка в выражении для числа операций.

Реализация гауссова исключения отнюдь не сводится к записи вложенных циклов в алгоритме 2.2. В самом деле, в зависимости от конкретных компьютера, языка программирования и порядка матрицы, простой акт перестановки двух последних циклов по j и k может изменить время работы алгоритма на несколько порядков. Мы подробно обсудим эти вопросы в разд. 2.6.

2.4. Анализ ошибок

Напомним нашу двухшаговую парадигму для получения оценок ошибки в приближенном решении системы $Ax=b$:

1. Проанализировать ошибки округлений с тем, чтобы показать: результат $\hat{x}$ решения системы $Ax=b$ является точным решением возмущенной системы $(A+\delta A)\hat{x}=b+\delta b$ с малыми возмущениями δA и δb. Это пример *обратного анализа ошибок*, причем δA и δb называются *обратными ошибками*.
2. Применяя теорию возмущений из разд. 2.2, получить оценку ошибки; можно использовать, например, оценку (2.3) или (2.5).

В этом разделе мы преследуем две цели. Первая состоит в том, чтобы показать, как нужно реализовать гауссово исключение, чтобы обеспечить малость обратных ошибок δA и δb. Мы хотели бы, в частности, чтобы отношения $\frac{\|\delta A\|}{\|A\|}$ и $\frac{\|\delta b\|}{\|b\|}$ были величинами порядка $O(\varepsilon)$. Мы не можем надеяться на бóльшую малость, поскольку простое округление наибольших элементов в A (или b), превращающее их в числа с плавающей точкой, уже может повлечь за собой соотношение $\frac{\|\delta A\|}{\|A\|}\geq\varepsilon$ (или $\frac{\|\delta b\|}{\|b\|}\geq\varepsilon$). Выясняется, что δA и δb вовсе не обязаны быть малыми, если не позаботиться о разумном выборе главных элементов. Этот вопрос обсуждается в следующем разделе.

Вторая цель заключается в том, чтобы вывести практичные оценки ошибок, которые были бы одновременно легко вычисляемыми и «реалистичными», т. е. близкими к подлинным ошибкам. Выясняется, что наилучшие оценки для $\|\delta A\|$, которые мы сможем обосновать формально, все же, в общем случае, много больше ошибок, наблюдаемых в реальных задачах. Поэтому наши практичные оценки ошибок (выводимые в разд. 2.4.4) опираются на вычисленную невязку $r = A\hat{x} - b$ и оценку (2.5), а не оценку (2.3). Нам нужен, кроме того, недорогой способ оценивания числа $\kappa(A)$; эта задача обсуждается в разд. 2.4.3.

К сожалению, мы не располагаем оценками ошибок, которые бы *всегда* удовлетворяли обеим нашим установкам на экономичность и реалистичность, т. е. оценками, которые бы одновременно

1. могли быть вычислены с затратой числа операций, пренебрежимо малого в сравнении со стоимостью решения системы $Ax = b$ (например, с затратой $O(n^2)$ флопов, тогда как стоимость гауссова исключения составляет $O(n^3)$ флопов);
2. всегда были не меньше подлинной ошибки и превосходили ее не более чем в заранее установленное число (скажем, 100) раз.

Практичные оценки ошибок из разд. 2.4.4 стоят $O(n^2)$ операций, но в некоторых, очень редких случаях они оказываются чересчур малы или чересчур велики по сравнению с действительной ошибкой. Вероятность таких случаев столь мала, что эти оценки широко используются на практике. Единственный способ получить подлинно гарантированные оценки состоит в использовании интервальной арифметики, или арифметики с очень высокой разрядностью, или и того, и другого; стоимость этого способа в несколько раз выше стоимости решения системы $Ax = b$ (см. разд. 1.5).

В действительности, выдвинута гипотеза о том, что оценки, удовлетворяющей обеим нашим установкам на экономичность и реалистичность, не существует; вопрос этот, однако, пока остается открытым.

2.4.1. Необходимость выбора главного элемента

Давайте применим LU-разложение без выбора главных элементов к матрице $A = \begin{bmatrix} .0001 & 1 \\ 1 & 1 \end{bmatrix}$, используя трехразрядную десятичную арифметику с плавающей точкой, и разберемся в причинах, почему получается неверный результат. Заметим, что $\kappa(A) = \|A\|_\infty \|A^{-1}\|_\infty \approx 4$; таким образом, матрица A хорошо обусловлена и мы могли бы ожидать, что системы $Ax = b$ решаются с хорошей точностью.

$$L = \begin{bmatrix} 1 & 0 \\ \mathrm{fl}(1/10^{-4}) & 1 \end{bmatrix}, \text{ число } \mathrm{fl}(1/10^{-4}) \text{ равно } 10^4,$$

$$U = \begin{bmatrix} 10^{-4} & 1 \\ & \mathrm{fl}(1 - 10^4 \cdot 1) \end{bmatrix}, \text{ число } \mathrm{fl}(1-10^4 \cdot 1) \text{ равно } -10^4,$$

таким образом, $LU = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix} \begin{bmatrix} 10^{-4} & 1 \\ & -10^4 \end{bmatrix} = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 0 \end{bmatrix}$,

но $A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix}$.

Заметим, что исходный элемент a_{22} был полностью «утрачен» для вычислений, когда из него пришлось вычесть число 10^4. Мы получили бы те же самые множители L и U, будь a_{22} равен 1, 0 или -2; вообще, любому числу, такому, что $\mathrm{fl}(a_{22} - 10^4) = -10^4$. Поскольку далее алгоритм работает только с L и U, будет получен один и тот же ответ для всех этих различных значений a_{22}, которые соответствуют совершенно различным матрицам A, а потому совершенно различным векторам $x = A^{-1}b$; мы не можем, следовательно, гарантировать хорошую точность результата. Это явление называется *численной неустойчивостью*, поскольку L и U *не* являются точными треугольными множителями матрицы, близкой к A. (По-другому, можно сказать, что число $\|A - LU\|$ сравнимо с $\|A\|$ вместо того, чтобы быть величиной порядка $\varepsilon\|A\|$.)

Посмотрим, что случится, если решать систему $Ax = [1, 2]^T$, пользуясь вычисленным разложением. Правильный ответ: $x \approx [1, 1]^T$. Вместо него мы получим следующее. Решая систему $Ly = [1, 2]^T$, находим $y_1 = \mathrm{fl}(1/1) = 1$ и $y_2 = \mathrm{fl}(2 - 10^4 \cdot 1) = -10^4$; отметим, что значение 2 было утеряно при вычитании из него числа 10^4. Решение системы $U\hat{x} = y$ дает $\hat{x}_2 = \mathrm{fl}((-10^4)/ * -10^4)) = 1$ и $\hat{x}_1 = \mathrm{fl}((1 - 1)/10^{-4}) = 0$, т. е. совершенно ошибочный результат.

Другое предупреждение о потере точности можно получить, сравнивая число обусловленности матрицы A с числами обусловленности L и U. Напомним, что задача решения системы $Ax = b$ была преобразована в решение двух других систем с матрицами L и U, поэтому мы не хотим, чтобы L и U были обусловлены много хуже, чем A. Между тем, здесь число обусловленности матрицы A близко к 4, тогда как числа обусловленности для L и U равны $\approx 10^8$.

В следующем разделе мы покажем, что использование метода GEPP почти всегда исключает неустойчивость, иллюстрируемую данным примером. Если бы здесь был применен этот метод, то прежде всего был бы изменен порядок двух уравнений. Предлагаем читателю проверить, что тогда были бы получены матрицы

$$L = \begin{bmatrix} 1 & 0 \\ \mathrm{fl}(.0001/1) & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ .0001 & 1 \end{bmatrix}$$

и

$$U = \begin{bmatrix} 1 & 1 \\ 0 & \mathrm{fl}(1 - .0001 \cdot 1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix};$$

произведение которых весьма близко к A. Обе матрицы L и U (как и A) обусловлены очень хорошо. Очень хорошую точность имеет и вычисленное решение системы.

2.4.2. Формальный анализ ошибок гауссова исключения

Изложим интуитивные соображения, стоящие за нашим анализом ошибок LU-разложения. Если промежуточные величины, возникающие в произведении $L \cdot U$, очень велики по сравнению с $\|A\|$, то информация, содержащаяся в элементах матрицы A, будет «утрачена», когда из них будут вычтены эти большие величины. Именно это произошло с элементом a_{22} в примере из разд. 2.4.1. Если бы, напротив, промежуточные величины в произведении $L \cdot U$ были сравнимы с элементами матрицы A, то можно было бы рассчитывать на получение

разложения с малой обратной ошибкой $A - LU$. Следовательно, мы хотим оценить наибольшие промежуточные величины в произведении $L \cdot U$. Мы сделаем это, оценивая элементы матрицы $|L| \cdot |U|$ (по поводу обозначений см. разд. 1.1).

Наш анализ будет схож с тем, что мы проделали в разд. 1.6 для задачи вычисления многочлена. Рассматривая там $p = \sum_i a_i x^i$, мы показали, что если число $|p|$ сравнимо с суммой $\sum_i |a_i x^i|$, то p будет вычислено с хорошей точностью.

После того как будет дан общий анализ гауссова исключения, мы используем его для демонстрации того, что метод GEPP (или, при бóльших затратах, метод GECP) обеспечивает сравнимость элементов матрицы $|L|\,|U|$ с $\|A\|$ почти в любой реалистической ситуации.

К сожалению, наилучшие оценки для $\|\delta A\|$, которые мы сможем доказать, все же, в общем случае, много больше ошибок, наблюдаемых в реальных задачах. Поэтому оценки ошибок, используемые на практике, основываются на вычисленной невязке r и оценке (2.5) (или оценке (2.9)), а не на строгой, но пессимистичной оценке, выводимой в данном разделе.

Чтобы упростить обозначения, предположим, что выбор главных элементов в матрице A проделан заранее. Упростим алгоритм 2.2 до двух соотношений; одно из них относится к элементам a_{jk} с $j \le k$, а другое — к элементам a_{jk} с $j > k$. Проследим вначале, что происходит в алгоритме 2.2 с элементом первого типа: этот элемент перевычисляется для каждого значения i от 1 до $j-1$ путем вычитания $l_{ji}u_{ik}$, после чего его значение присваивается элементу u_{jk}; таким образом,

$$u_{jk} = a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik}.$$

Если $j > k$, то и здесь из a_{jk} вычитается $l_{ji}u_{ik}$ для i от 1 до $k-1$, затем полученная сумма делится на u_{kk} и результат становится значением элемента l_{jk}:

$$l_{jk} = \frac{a_{jk} - \sum_{i=1}^{k-1} l_{ji}u_{ik}}{u_{kk}}.$$

Чтобы провести анализ ошибок округлений для этих двух формул, воспользуемся результатом, установленным в вопросе 1.10: скалярное произведение, вычисленное в арифметике с плавающей точкой, удовлетворяет соотношению

$$\mathrm{fl}\left(\sum_{i=1}^{d} x_i y_i\right) = \sum_{i=1}^{d} x_i y_i (1+\delta_i), \quad \text{где} \quad |\delta_i| \le d\varepsilon.$$

Применим этот результат к формуле для u_{jk}, что даст[1]

$$u_{jk} = \left(a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik}(1+\delta_i)\right)(1+\delta'),$$

[1] Строго говоря, наш анализ формулы предполагает, что сначала вычисляется сумма, а затем накопленный результат вычитается из a_{jk}. Однако окончательная оценка не зависит от порядка суммирования.

где $|\delta_i| \le (j-1)\varepsilon$ и $|\delta'| \le \varepsilon$. Разрешая это равенство относительно a_{jk}, получаем

$$\begin{aligned} a_{jk} &= \frac{1}{1+\delta'} u_{jk} \cdot l_{jj} + \sum_{i=1}^{j-1} l_{ji} u_{ik}(1+\delta_i), \quad \text{так как } l_{jj} = 1, \\ &= \sum_{i=1}^{j} l_{ji} u_{ik} + \sum_{i=1}^{j} l_{ji} u_{ik} \delta_i \\ &\left(\text{где } |\delta_i| \le (j-1)\varepsilon \quad \text{и} \quad 1+\delta_j \equiv \frac{1}{1+\delta'}\right) \\ &\equiv \sum_{i=1}^{j} l_{ji} u_{ik} + E_{jk}. \end{aligned}$$

Величина E_{jk} оценивается так:

$$|E_{jk}| = \left| \sum_{i=1}^{j} l_{ji} \cdot u_{ik} \cdot \delta_i \right| \le \sum_{i=1}^{j} |l_{ji}| \cdot |u_{ik}| \cdot n\varepsilon = n\varepsilon (|L| \cdot |U|)_{jk}.$$

Проводя аналогичный анализ формулы для l_{jk}, находим

$$l_{jk} = (1+\delta'') \left(\frac{(1+\delta')(a_{jk} - \sum_{i=1}^{k-1} l_{ji} u_{ik}(1+\delta_i))}{u_{kk}} \right),$$

где $|\delta_i| \le (k-1)\varepsilon$, $|\delta'| \le \varepsilon$ и $|\delta''| \le \varepsilon$. Разрешая это соотношение относительно a_{jk}, имеем

$$\begin{aligned} a_{jk} &= \frac{1}{(1+\delta')(1+\delta'')} u_{kk} l_{jk} + \sum_{i=1}^{k-1} l_{ji} u_{ik}(1+\delta_i) \\ &= \sum_{i=1}^{k} l_{ji} u_{ik} + \sum_{i=1}^{k} l_{ji} u_{ik} \delta_i \\ &\equiv \sum_{i=1}^{k} l_{ji} u_{ik} + E_{jk}, \qquad 1+\delta_k \equiv \frac{1}{(1+\delta')(1+\delta'')}, \end{aligned}$$

где $|\delta_i| \le n\varepsilon$. Итак, как и выше, $|E_{jk}| \le n\varepsilon(|L| \cdot |U|_{jk}$.

В целом, итоги этого анализа ошибок можно суммировать простой формулой $A = LU + E$, где $|E| \le n\varepsilon |L| \cdot |U|$. Беря нормы, получаем $|||E||| \le n\varepsilon |||L||| \cdot |||U|||$. Если норма зависит только от абсолютных величин элементов матрицы (это верно для max-нормы, 1-нормы, нормы Фробениуса, но не для 2-нормы), то это неравенство можно упростить к виду $\|E\| \le n\varepsilon \|L\| \cdot \|U\|$.

Проанализируем теперь оставшуюся часть задачи: решить систему $LUx = b$, решая системы $Ly = b$ и $Ux = y$. Согласно результату, полученному в вопросе 1.11, решение системы $Ly = b$ посредством прямой подстановки дает приближенное решение $\hat{y}$, удовлетворяющее уравнению $(L + \delta L)\hat{y} = b$, где $|\delta L| \le n\varepsilon |L|$. Аналогично, при решении системы $Ux = \hat{y}$ мы получим вектор $\hat{x}$, удовлетворяющий уравнению $(U + \delta U)\hat{x} = \hat{y}$ с $|\delta U| \le n\varepsilon |U|$.

Объединяя эти соотношения, имеем

$$\begin{aligned} b &= (L+\delta L)\hat{y} \\ &= (L+\delta L)(U+\delta U)\hat{x} \\ &= (LU + L\delta U + \delta LU + \delta L\delta U)\hat{x} \\ &= (A - E + L\delta U + \delta LU + \delta L\delta U)\hat{x} \\ &\equiv (A+\delta A)\hat{x}, \quad \text{где } \delta A = -E + L\delta U + \delta LU + \delta L\,\delta U. \end{aligned}$$

Оценим теперь δA, используя неравенство треугольника и наши оценки для E, δL и δU:

$$\begin{aligned} |\delta A| &= |-E + L\delta U + \delta LU + \delta L\delta U| \\ &\le |E| + |L\delta U| + |\delta LU| + |\delta L\delta U| \\ &\le |E| + |L|\cdot|\delta U| + |\delta L|\cdot|U| + |\delta L|\cdot|\delta U| \\ &\le n\varepsilon|L|\cdot|U| + n\varepsilon|L|\cdot|U| + n\varepsilon|L|\cdot|U| + n^2\varepsilon^2|L|\cdot|U| \\ &\approx 3n\varepsilon|L|\cdot|U|. \end{aligned}$$

Беря здесь нормы и предполагая, что $|||X||| = \|X\|$ (как и прежде, это верно для max-нормы, 1-нормы, нормы Фробениуса, но не для 2-нормы), получаем $\|\delta A\| \le 3n\varepsilon\|L\|\cdot\|U\|$.

Таким образом, чтобы понять, в каких случаях гауссово исключение является обратно устойчивым алгоритмом, нужно выяснить, когда верно соотношение $3n\varepsilon\|L\|\cdot\|U\| = O(\varepsilon)\|A\|$. Именно тогда величина $\frac{\|\delta A\|}{\|A\|}$ в оценках теории возмущений имеет порядок $O(\varepsilon)$, как мы того желали (заметим, что $\delta b = 0$).

Основное эмпирическое наблюдение, поддерживаемое десятилетиями вычислительной практики, заключается в том, что метод GEPP *почти* всегда обеспечивает соотношение $\|L\|\cdot\|U\| \approx \|A\|$. Метод гарантирует, что каждый элемент в L ограничен по абсолютной величине единицей, поэтому достаточно рассмотреть $\|U\|$. Определим *коэффициент роста* в методе GEPP[1] как $g_{\text{PP}} = \|U\|_{\max}/\|A\|_{\max}$, где $\|A\|_{\max} = \max_{i,j}|a_{ij}|$; таким образом, устойчивость метода эквивалентна тому, что число g_{PP} мало или является медленно растущей функцией от n. На практике, g_{PP} почти всегда не превосходит n. Поведение в среднем, по-видимому, соответствует функции $n^{2/3}$ или, возможно, даже $n^{1/2}$ [242]. (См. рис. 2.1.) Это делает метод GEPP наиболее предпочтительным выбором для многих задач. К сожалению, существуют немногочисленные примеры, где g_{PP} может достигать величины 2^{n-1}.

Предложение 2.1. *Метод GEPP гарантирует, что* $g_{\text{PP}} \le 2^{n-1}$. *Эта оценка достижима.*

Доказательство. На первом шаге метода происходит перевычисление $\tilde{a}_{jk} = a_{jk} - l_{ji}u_{ik}$, где $|l_{ji}| \le 1$ и $|u_{ik}| = |a_{ik}| \le \max_{rs}|a_{rs}|$; отсюда $|\tilde{a}_{jk}| \le 2\max_{rs}|a_{rs}|$. Таким образом, каждый из $n-1$ основных шагов метода может удвоить величину оставшихся матричных элементов, что приводит к общей оценке 2^{n-1}. Пример в вопросе 2.14 показывает, что эта оценка достижима. □

[1] Это определение немного отличается от определения, обычно используемого в литературе, но, по существу, эквивалентно ему (см.[121, с.115]).

Собирая все приведенные выше оценки, получаем

$$\|\delta A\|_\infty \le 3g_{\mathrm{PP}}n^3\varepsilon\|A\|_\infty. \tag{2.11}$$

Здесь использованы неравенства $\|L\|_\infty \le n$ и $\|U\|_\infty \le ng_{\mathrm{PP}}\|A\|_\infty$. Множитель $3g_{\mathrm{PP}}n^3$ в оценке (2.11) заставляет ее почти всегда сильно переоценивать подлинное значение $\|\delta A\|$, даже если $g_{\mathrm{PP}} = 1$. Например, при $\varepsilon = 10^{-7}$ и $n = 150$, что соответствует матрице весьма умеренного порядка, имеем $3n^3\varepsilon > 1$, что означает: потенциально все верные знаки в вычисленном решении могут быть потеряны. В примере 2.3 сравниваются графики функции $3g_{\mathrm{PP}}n^3\varepsilon$ и подлинной обратной ошибки, чтобы показать, насколько пессимистичен может быть первый из них. Величина $\|\delta A\|$ обычно имеет порядок $O(\varepsilon)\|A\|$, поэтому можно сказать, что *на практике* метод GEPP *обратно устойчив*, хотя мы и можем построить примеры, на которых он «проваливается». В разделе 2.4.4 представлены практичные оценки ошибки в вычисленном решении системы $Ax = b$, много меньшие, чем оценка, получаемая при использовании неравенства $\|\delta A\|_\infty \le 3g_{\mathrm{PP}}n^3\varepsilon\|A\|_\infty$.

Можно показать, что метод GECP еще более устойчив, чем метод GEPP. Оценка для коэффициента роста g_{CP} этого метода в наихудшем случае имеет вид [262, с. 213]

$$g_{\mathrm{CP}} = \frac{\max_{ij}|u_{ij}|}{\max_{ij}|a_{ij}|} \le \sqrt{n \cdot 2 \cdot 3^{1/2} \cdot 4^{1/3} \ldots n^{1/(n-1)}} \approx n^{1/2+\log_e n/4}.$$

Эта верхняя граница также слишком велика для практических целей. Поведение коэффициента g_{CP} в среднем соответствует функции $n^{1/2}$. Старая гипотеза, согласно которой $g_{\mathrm{CP}} \le n$, недавно была опровергнута [99, 122]. Все еще остается открытой задача построения хорошей верхней оценки для g_{CP} (который, как по-прежнему полагают, является величиной порядка $O(n)$).

Дополнительные $O(n^3)$ сравнений, производимых в методе GECP при выборе главных элементов ($O(n^2)$ сравнений на шаг, в то время как в методе GEPP достаточно $O(n)$ сравнений на шаг), делают этот метод значительно более медленным, чем метод GEPP, особенно на высокопроизводительных компьютерах, где операции с плавающей точкой выполняются почти так же быстро, как сравнение. Поэтому обращение к методу GECP редко бывает оправданным (см., однако, разд. 2.4.4, 2.5.1 и 5.4.3).

Пример 2.3. Рисунки 2.1 и 2.2 иллюстрируют оценки обратной ошибки, о которых шла речь выше. Для обоих рисунков генерировались пять случайных матриц A каждой указанной размерности; элементы матриц суть независимые нормально распределенные величины со средним 0 и средним квадратичным отклонением 1. (Тестирование на подобных случайных матрицах может подчас неверно ориентировать в отношении поведения метода для некоторых реальных задач; все же такое тестирование достаточно информативно.) Для каждой матрицы генерировался случайный вектор b с аналогичными характеристиками. Для решения системы $Ax = b$ использовались оба метода GEPP и GECP. На рис. 2.1 изображены коэффициенты роста g_{PP} и g_{CP}. Как и следовало ожидать, оба медленно растут с ростом размерности. Рис. 2.2 показывает поведение двух наших верхних оценок для обратной ошибки: $3n^3\varepsilon g_{\mathrm{PP}}$ (или $3n^3\varepsilon g_{\mathrm{CP}}$) и $3n\varepsilon\frac{\||L|\cdot|U|\|_\infty}{\|A\|_\infty}$. Показана также подлинная обратная ошибка, вычисляемая,



Рис. 2.1. Коэффициенты роста для случайных матриц, $\circ = g_{\text{PP}}$, $+ = g_{\text{CP}}$.

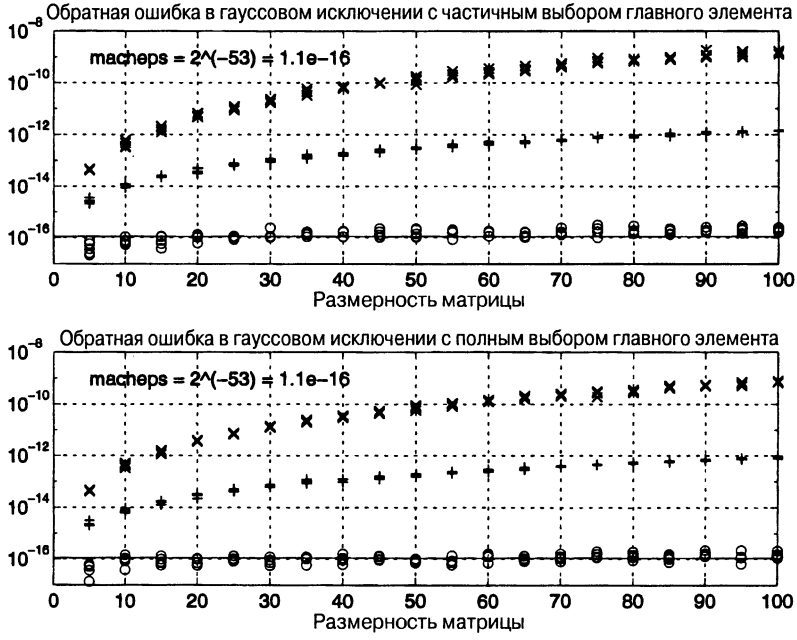


Рис. 2.2. Обратная ошибка в гауссовом исключении для случайных матриц, $\times = 3n^3\varepsilon g$, $+ = 3n\||L|\cdot|U|\|_\infty/\|A\|_\infty$, $\circ = \|Ax-b\|_\infty/(\|A\|_\infty\|x\|_\infty)$.

как это описано в теореме 2.2. Машинное эпсилон указано сплошной горизонтальной прямой на уровне $\varepsilon = 2^{-53} \approx 1.1 \cdot 10^{-16}$. Обе оценки действительно являются верхними границами для подлинной обратной ошибки, но переоценивают ее на несколько порядков. По поводу Matlab-программы, производящей эти графики, см. HOMEPAGE/Matlab/pivot.m. ◇

2.4.3. Оценка чисел обусловленности

Чтобы вычислить практичную оценку ошибки, основанную на неравенстве типа (2.5), нужно уметь оценивать число $\|A^{-1}\|$. Этого достаточно, чтобы суметь оценить и число обусловленности $\kappa(A) = \|A^{-1}\| \cdot \|A\|$, поскольку $\|A\|$ находится легко. Одна из возможностей состоит в том, чтобы вычислить матрицу A^{-1} в явном виде, а затем определить ее норму. Однако это стоило бы $2n^3$ операций, больше, чем $\frac{2}{3}n^3$ операций, уже выполненных для гауссова исключения. (Отметим следствие из сказанного: нет никакого выигрыша в решении системы $Ax = b$ путем вычисления матрицы A^{-1} с последующим умножением ее на b. Это верно даже в том случае, когда при одной и той же A имеется много различных векторов b. См. вопрос 2.2.) Фактом является то, что большинство пользователей не станут вычислять оценки ошибки, если они обходятся так дорого.

Поэтому вместо вычисления A^{-1} мы разработаем гораздо более дешевый алгоритм *оценивания* числа $\|A^{-1}\|$. Такой алгоритм называется *оценщиком обусловленности* и должен обладать следующими свойствами:

1. Он должен использовать $O(n^2)$ операций, если заданы треугольные множители L и U матрицы A. Для достаточно больших n эта цена пренебрежимо мала по сравнению с $\frac{2}{3}n^3$ операций, необходимых для метода GEPP.
2. Он должен давать оценку, почти всегда не превосходящую удесятиренного значения $\|A^{-1}\|$. Это все, что требуется от оценки ошибки, которая должна говорить нам, сколько верных знаков в решении задачи мы имеем. (Ошибка, соответствующая множителю 10, равносильна одному потерянному десятичному разряду.[1])

Имеется много таких оценщиков (их обзор дан в [146]). Мы выбрали для описания метод, применимый не только в решении системы $Ax = b$, но и в ряде других задач; вследствие такой универсальности, он несколько медленней алгоритмов, специализированных для задачи $Ax = b$ (хотя и для этой задачи он достаточно быстр). Подобно большинству других оценщиков, наш алгоритм дает *нижнюю* границу для $\|A^{-1}\|$, а не оценку сверху. Эмпирически эта оценка отличается от $\|A^{-1}\|$ множителем, почти всегда не меньшим, чем 1/10 и обычно находящимся в пределах от $\frac{1}{3}$ до $\frac{1}{2}$. В случае матриц, использованных для рис. 2.1 и 2.2 (их числа обусловленности варьируются от 10 до 10^5), наш оценщик давал несколько верных десятичных разрядов числа $\|A^{-1}\|$ в 83%

[1] Как уже отмечалось, никому не удалось построить оценщик, который бы аппроксимировал число $\|A^{-1}\|$ с некоторой гарантированной точностью и был бы в то же время значительно дешевле явного вычисления матрицы A^{-1}. Была выдвинута гипотеза, что такого оценщика не существует, но она не была доказана.

экспериментов; наихудшая оценка составляла 0.43 от правильного результата. Такой точности более чем достаточно для того, чтобы иметь возможность оценить число верных знаков в полученном решении.

Алгоритм оценивает 1-норму $\|B\|_1$ матрицы B; предполагается, что мы можем вычислять произведения Bx и B^Ty для любых векторов x и y. Мы применим алгоритм к $B = A^{-1}$, поэтому потребуется вычислять векторы $A^{-1}x$ и $A^{-1}y$, т. е. решать линейные системы. При имеющемся LU-разложении матрицы A решение таких систем обходится лишь в $O(n^2)$ операций. Алгоритм был разработан в [138, 146, 148], а его новейшая версия описана в [147]. Напомним, что $\|B\|_1$ определяется как

$$\|B\|_1 = \max_{x\neq 0} \frac{\|Bx\|_1}{\|x\|_1} = \max_j \sum_{i=1}^{n} |b_{ij}|.$$

Легко показать, что максимум по ненулевым векторам x достигается здесь для $x = e_{j0} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$. (Если $\max_j \sum_i |b_{ij}|$ достигается при $j = j_0$, то единственная ненулевая компонента вектора x имеет номер j_0.)

Поиск по всем e_j $(j = 1, \ldots, n)$ означает вычисления всех столбцов матрицы $B = A^{-1}$, что слишком дорого. Поскольку $\|B\|_1 = \max_{\|x\|_1\leq 1} \|Bx\|_1$, мы можем вместо перебора e_j применить метод *наискорейшего подъема* к функции $f(x) \equiv \|Bx\|_1$ на множестве $\|x\|_1 \leq 1$. Последнее, очевидно, является выпуклым множеством, а $f(x)$ — выпуклой функцией. Действительно, для $0 \leq \alpha \leq 1$ имеем $f(\alpha x + (1-\alpha)y) = \|\alpha Bx + (1-\alpha)By\|_1 \leq \alpha\|Bx\|_1 + (1-\alpha)\|By\|_1 = \alpha f(x) + (1-\alpha)f(y)$.

Максимизация $f(x)$ методом скорейшего подъема означает движение x в направлении градиента $\bigtriangledown f(x)$ (если он существует), пока $f(x)$ возрастает. Из выпуклости $f(x)$ следует, что $f(y) \geq f(x) + \bigtriangledown f(x) \cdot (y - x)$ (если $\bigtriangledown f(x)$ существует). Чтобы вычислить $\bigtriangledown f$, *предположим*, что в сумме $f(x) = \sum_i |\sum_j b_{ij}x_j|$ все слагаемые $\sum_j b_{ij}x_j$ отличны от нуля (это верно почти всегда). Положим $\zeta_i = \operatorname{sgn} \sum_j b_{ij}x_j$, тогда $\zeta_i = \pm 1$ и $f(x) = \sum_i \sum_j \zeta_i b_{ij} x_j$. Поэтому $\frac{\partial f}{\partial x_k} = \sum_i \zeta_i b_{ik}$ и $\bigtriangledown f = \zeta^T B = (B^T\zeta)^T$.

Итак, вычисление градиента $\bigtriangledown f(x)$ состоит из трех шагов: $w = Bx$, $\zeta = \operatorname{sgn}(w)$ и $\bigtriangledown f = \zeta^T B$.

Алгоритм 2.5. *Оценщик обусловленности Хэйджера (Hager) выдает нижнюю границу $\|w\|_1$ для $\|B\|_1$:*

выбрать произвольный вектор x с нормой $\|x\|_1 = 1$ /* *например, $x_i = \frac{1}{n}$* */
repeat
  $w = Bx$, $\zeta = \operatorname{sgn}(w)$, $z = B^T\zeta$ /* $z^T = \bigtriangledown f$ */
  if $\|z\|_\infty \leq z^T x$ *then*
    return $\|w\|_1$
  else
    $x = e_j$ *где* $|z_j| = \|z\|_\infty$
  endif
end repeat

Теорема 2.6. 1. *Если на выходе алгоритма получено значение* $\|w\|_1$, *то* $\|w\|_1 = \|Bx\|_1$ *есть локальный максимум функции* $\|Bx\|_1$.

2. *В противном случае,* $\|Be_j\|$ *(значение в конце цикла)* $> \|Bx\|$ *(значение в начале цикла). Тем самым значение функции* $f(x)$ *увеличено.*

Доказательство.

1. В данном случае $\|z\|_\infty \le z^T x$. Вблизи x функция $f(x) = \|Bx\|_1 = \sum_i \sum_j \zeta_i b_{ij} x_j$ линейно зависит от x, поэтому $f(y) = f(x) + \nabla f(x) \cdot (y - x) = f(x) + z^T(y - x)$, где $z^T = \nabla f(x)$. Чтобы показать, что x — локальный максимум, проверим, что $z^T(y - x) \le 0$ при $\|y\|_1 = 1$. Имеем

$$z^T(y - x) = z^T y - z^T x = \sum_i z_i \cdot y_i - z^T x \le \sum_i |z_i| \cdot |y_i| - z^T x$$
$$\le \|z\|_\infty \cdot \|y\|_1 - z^T x = \|z\|_\infty - z^T x \le 0, \quad \text{что и требовалось.}$$

2. В данном случае $\|z\|_\infty > z^T x$. Положим $\tilde{x} = e_j \cdot \operatorname{sign}(Z_j)$, где (j_j) выбрано так, что $\|z_j\| = \|z\|_\infty$. Тогда

$$f(\tilde{x}) \ge f(x) + \nabla f \cdot (\tilde{x} - x) = f(x) + z^T(\tilde{x} - x)$$
$$= f(x) + z^T \tilde{x} - z^T x = f(x) + |z_j| - z^T x > f(x),$$

где последнее неравенство выполняется вследствие выбора j. □

В [147, 148] описаны результаты тестирования несколько улучшенной версии алгоритма на многочисленных случайных матрицах порядков 10, 25 и 50 и с числами обусловленности $\kappa = 10, 10^3, 10^6, 10^9$. В наихудшем случае вычисленное κ составляло 0.44 от подлинного κ. Алгоритм реализован LAPACK-подпрограммой `slacon`. Внутри таких программ LAPACK'a, как `sgesvx`, имеется обращение к `slacon` с получением оценки числа обусловленности. (В действительности, чтобы избежать переполнения в случае вырожденной матрицы, выдается число, обратное к оценке числа обусловленности.) Matlab содержит другой оценщик обусловленности `rcond`. Matlab-программа `cond` вычисляет точное число обусловленности $\|A^{-1}\|_2 \cdot \|A\|_2$ с помощью алгоритмов, обсуждаемых в разд. 5.4; она работает намного медленнее, чем `rcond`.

Оценивание относительного числа обусловленности

Алгоритм из предыдущего раздела можно применить и к оцениванию относительного числа обусловленности $\kappa_{CR}(A = \||A^{-1}| \cdot |A|\|_\infty$ (см. (2.8)) или вычислению границы $\||A^{-1}| \cdot |r|\|_\infty$ (см. (2.9)). Обе задачи можно свести к одной и той же, а именно оцениванию величины $\||A^{-1} \cdot g|\|_\infty$, где g — вектор с неотрицательными компонентами. Чтобы пояснить это, введем вектор e, все компоненты которого равны единице. Из части 5 леммы 1.7 следует, что $\|X\|_\infty = \|Xe\|_\infty$ для матрицы X с неотрицательными элементами. Поэтому

$$\||A^{-1}| \cdot |A|\|_\infty = \||A^{-1}| \cdot |A|e\|_\infty = \||A^{-1}| \cdot g\|_\infty, \quad \text{где} \quad g = |A|e.$$

Мы собираемся оценить $\||A^{-1}| \cdot g\|$ следующим образом. Пусть $G = \mathrm{diag}(g_1, \ldots, g_n)$, тогда $g = Ge$. Отсюда

$$\begin{aligned} \||A^{-1}| \cdot g\|_\infty &= \||A^{-1}| \cdot Ge\|_\infty = \||A^{-1}| \cdot G\|_\infty = \||A^{-1}G|\|_\infty \\ &= \|A^{-1}G\|_\infty. \end{aligned} \tag{2.12}$$

Последнее равенство справедливо, так как $\|Y\|_\infty = \||Y|\|_\infty$ для любой матрицы Y. Таким образом, достаточно оценить max-норму матрицы $A^{-1}G$. Это можно сделать, применяя алгоритм Хэйджера (т. е. алгоритм 2.5) к матрице $(A^{-1}G)^T = GA^{-T}$; в результате будет получена оценка числа $\|(A^{-1}G)^T\|_1 = \|A^{-1}G\|_\infty$ (см. часть 6 леммы 1.7). При этом нужны произведения матрицы GA^{-T} и транспонированной матрицы $A^{-1}G$ с векторами. Умножение на диагональную матрицу G выполняется тривиально, а умножения на A^{-1} и A^{-T} производятся, как и в предыдущем разделе, с использованием LU-разложения матрицы A.

2.4.4. Практичные оценки ошибки

Приведем две практичные оценки ошибки в приближенном решении $\hat{x}$ системы $Ax = b$. Используя неравенство (2.5), получаем первую оценку:

$$\text{ошибка} = \frac{\|\hat{x} - x\|_\infty}{\|\hat{x}\|_\infty} \le \|A^{-1}\|_\infty \cdot \frac{\|r\|_\infty}{\|\hat{x}\|_\infty}, \tag{2.13}$$

где $r = A\hat{x} - b$ — невязка вектора $\hat{x}$. Число $\|A^{-1}\|_\infty$ оцениваем, применяя алгоритм 2.5 к матрице $B = A^{-T}$; это дает оценку для $\|B\|_1 = \|A^{-T}\|_1 = \|A^{-1}\|_\infty$ (см. части 5 и 6 леммы 1.7).

Наша вторая оценка вытекает из более точного неравенства (2.9):

$$\text{ошибка} = \frac{\|\hat{x} - x\|_\infty}{\|\hat{x}\|_\infty} \le \frac{\||A^{-1}| \cdot |r|\|_\infty}{\|\hat{x}\|_\infty}. \tag{2.14}$$

Величину $\||A^{-1}| \cdot |r|\|_\infty$ оцениваем с помощью алгоритма, основанного на равенствах (2.12). Оценка (2.14) (модифицированная, как это описано ниже в разделе «Возможные неприятности») вычисляется такими LAPACK-программами, как, например, `sgesvx`. Имя переменной, соответствующей в LAPACK'е этой оценке, есть `FERR` (от Forward ERRor).

Пример 2.4. Используя тот же набор тестовых примеров, что и для рис. 2.1 и 2.2, мы вычисляли первую оценку ошибки (2.13) и подлинную ошибку. Результаты можно видеть на рис. 2.3. На плоскости переменных (подлинная ошибка, оценка ошибки) каждой системе $Ax = b$, решенной посредством метода GEPP, соответствует символ $\circ$ и каждой системе, решенной методом GECP, соответствует символ $+$. Если бы оценка совпадала с подлинной ошибкой, то символ $\circ$ или $+$ лежал бы на сплошной диагональной линии. Поскольку оценка всегда больше ошибки, то все символы находятся выше диагонали. Если отношение оценки к подлинной ошибке меньше 10, то символы попадают в полосу между сплошной диагональю и первой наддиагональной штрихованной линией. Если указанное отношение заключено между 10 и 100, то символы попадают в полосу между двумя первыми штрихованными наддиагоналями. Большинство оценок

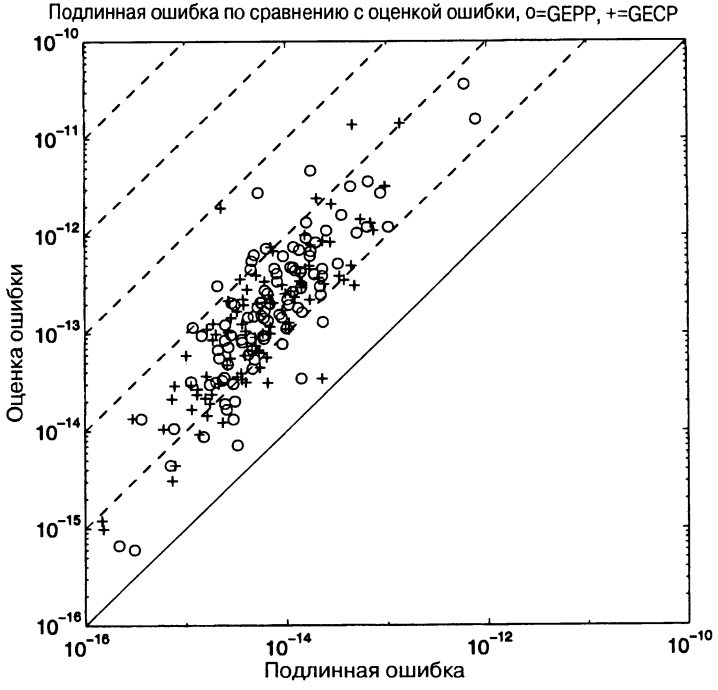


Рис. 2.3. Оценка ошибки (2.13) по сравнению с подлинной ошибкой, $\circ$ = *GEPP*, $+$ = *GECP*.

ошибки соответствует этой полосе и лишь несколько оценок в 1000 раз превосходят подлинную ошибку. Таким образом, вычисляемая нами оценка ошибки указывает число верных десятичных разрядов результата, на 1–2 или, в редких случаях, 3 разряда меньшее правильного числа. По поводу Matlab-программы, производящей эти графики, см., как и выше, HOMEPAGE/Matlab/pivot.m. $\diamond$

Пример 2.5. Приведем пример, иллюстрирующий различие между оценками ошибки (2.13) и (2.14). Этот пример покажет также, что метод GECP иногда может быть точнее метода GEPP. Был взят набор плохо масштабированных задач, построенных следующим образом. Размерности тестовых матриц меняются от 5 до 100 и каждая матрица имеет вид произведения $A = DB$. Матрица B получается присоединением к единичной матрице очень малых (порядка 10^{-7}) внедиагональных элементов, генерируемых случайным образом; таким образом, B обусловлена очень хорошо. Матрица D — диагональная, причем ее диагональные элементы образуют геометрическую прогрессию, начинающуюся с 1 и кончающуюся числом 10^{14}. (Иными словами, отношение $d_{i+1,i+1}/d_{ii}$ не зависит от i). Матрицы A имеют числа обусловленности $\kappa(A) = \|A^{-1}\|_\infty \|A\|_\infty$, почти равные 10^{14}, т. е. очень плохо обусловлены; в то же время, их относительные числа обусловленности $\kappa_{CR}(A) = \||A^{-1}| \cdot |A|\|_\infty = \||B^{-1}| \cdot |B|\|_\infty$ почти равны 1. Как и выше, машинная точность ε равна $2^{-53} \approx 10^{-16}$.

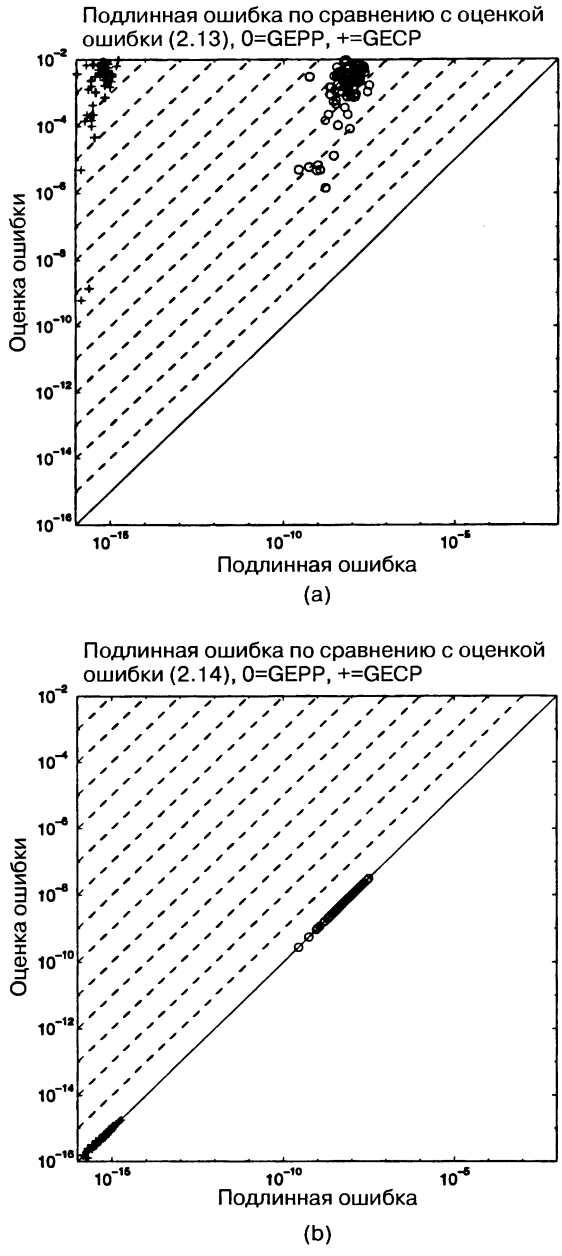


Рис. 2.4. (a) Оценка ошибки (2.13) по сравнению с подлинной ошибкой; (b) оценка ошибки (2.14) по сравнению с подлинной ошибкой.

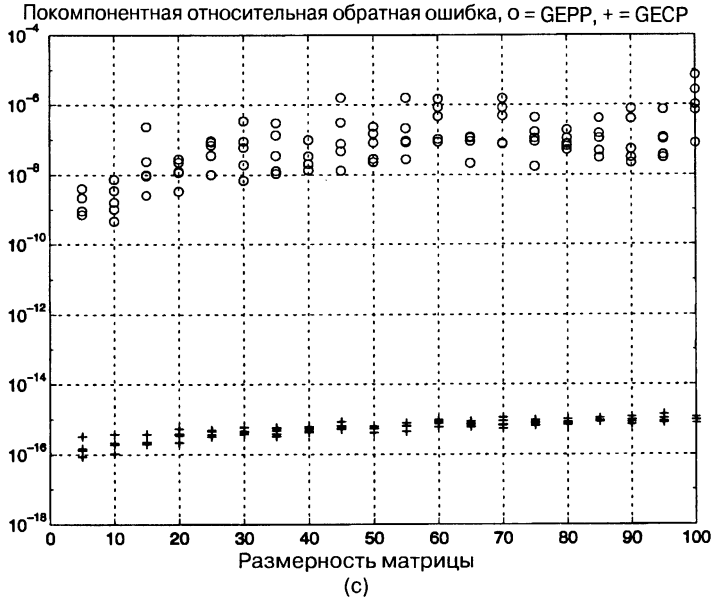


Рис. 2.4. Продолжение. (c) Покомпонентная относительная обратная ошибка из теоремы 2.3.

Примеры вычислялись посредством уже упоминавшейся Matlab-программы HOMEPAGE/Matlab/pivot.m.

Ни в каком из примеров коэффициенты роста g_{PP} и g_{CP} не были больше числа 1.33, а обратная ошибка, указываемая теоремой 2.2, ни разу не превзошла 10^{-15}. Оценщик Хэйджера был очень точен во всех случаях, давая оценку, совпадающую с подлинным числом обусловленности 10^{14} во многих десятичных разрядах.

На рис. 2.4 изображены оценки (2.13) и (2.14) для этих примеров, а также покомпонентная относительная обратная ошибка, указываемая формулой из теоремы 2.3. Кластер из знаков + в верхнем левом углу рис. 2.4(a) показывает, что, хотя метод GECP вычисляет решение с очень малой погрешностью порядка 10^{-15}, оценка (2.13) обычно дает весьма пессимистическое значение, близкое к 10^{-2}. Так происходит потому, что число обусловленности равно 10^{14} и оценка ошибки близка к $10^{-16} \cdot 10^{14} = 10^{-2}$, за исключением маловероятного случая, когда обратная ошибка много меньше числа $\varepsilon \approx 10^{-16}$. Кластер из кругов вверху и посреди того же рисунка свидетельствует, что метод GEPP имеет бóльшую погрешность (порядка 10^{-8}); оценка же (2.13) по-прежнему обычно близка к 10^{-2}.

В противоположность этому, оценка (2.14) почти совершенно точна, что иллюстрируется плюсами и кругами на диагонали рис. 2.4(b). Этот рисунок снова показывает, что метод GECP имеет почти идеальную точность, тогда как метод GEPP теряет почти половину верных разрядов. Это различие в точности объясняется рисунком 2.4(c), показывающим покомпонентную относительную

обратную ошибку из теоремы 2.3 для методов GEPP и GECP. Из рисунка видно, что второй метод имеет почти идеальную обратную ошибку в покомпонентном относительном смысле; поскольку соответствующее относительное число обусловленности равно 1, достигается очень высокая точность. Напротив, метод GEPP не вполне устойчив в этом смысле, теряя от 5 до 10 верных десятичных разрядов.

В разделе 2.5 будет показано, как итерационно улучшить приближенное решение $\hat{x}$. Один шаг этого итерационного процесса повышает точность решения, вычисленного методом GEPP, до уровня точности решения, найденного в методе GECP. Поскольку последний требует значительно большей работы, он очень редко используется на практике. ◇

Возможные неприятности

К сожалению, как уже отмечалось в начале разд. 2.4, оценки (2.13) и (2.14), будучи реализованы на практике, не всегда дают правильные результаты. В данном разделе описаны (немногочисленные!) причины, почему это может происходить, и практические приемы, помогающие частично исправить положение.

Во-первых, как указано в разд. 2.4.3, оценка для $\|A^{-1}\|$, выдаваемая алгоритмом 2.5 (и другими подобными алгоритмами), является лишь нижней границей, хотя вероятность того, что эта граница составляет менее 1/10 от точного значения, очень мала.

Во-вторых, существует хотя и малая, но ненулевая вероятность того, что, вследствие округлений при вычислении невязки $r = A\hat{x} - b$, число $\|r\|$ станет неправдоподобно малым, фактически, нулем; это сделает слишком малой вычисленную нами оценку ошибки. Чтобы предупредить такую возможность, можно добавить к $|r|$ некоторую малую величину. Из вопроса 1.10 известна следующая оценка для ошибок округления при вычислении r:

$$|(A\hat{x} - b) - \mathrm{fl}(A\hat{x} - b)| \leq (n+1)\varepsilon(|A| \cdot |\hat{x}| + |b|). \tag{2.15}$$

Поэтому в оценке (2.14) можно заменить $|r|$ на $|r| + (n+1)\varepsilon(|A| \cdot |\hat{x}| + |b|)$ (так сделано в LAPACK-программе `sgesvx`), а в оценке (2.13) — $\|r\|$ на $\|r\| + (n+1)\varepsilon(\|A\| \cdot \|\hat{x}\| + \|b\|)$. Множитель $n+1$ обычно чересчур велик и при желании может быть опущен.

В-третьих, из-за округлений в гауссовом исключении, примененном к очень плохо обусловленной матрице, могут быть получены настолько неточные матрицы L и U, что оценка (2.14) окажется слишком малой.

Пример 2.6. Приведем пример, принадлежащий Кахану (W. Kahan), который иллюстрирует трудность получения подлинно гарантированных оценок ошибки. В этом примере матрица A будет *точно* вырожденной. Следовательно, любая оценка для $\frac{\|x-\hat{x}\|}{\|x\|}$ не должна быть меньше единицы, указывая, что в вычисленном решении не может быть верных разрядов, поскольку истинное решение не существует.

Вследствие округлений в ходе гауссова исключения будут получены невырожденные, но очень плохо обусловленные матрицы L и U. В этом примере при вычислениях в Matlab'е с IEEE-арифметикой двойной точности вычисленная невязка r из-за округлений оказывается *точным* нулем, поэтому обе оценки

(2.13) и (2.14) обращаются в нуль. Исправление оценки (2.13) путем добавления $4\varepsilon(\|A\| \cdot \|\hat{x}\| + \|b\|)$ дает значение, большее 1, как и требовалось.

К сожалению, вторая, «более точная» оценка (2.14) дает приблизительно 10^{-7}, ошибочно указывая, что семь разрядов вычисленного решения верны.

Вот как строится этот пример. Положим $\chi = 3/2^{29}$, $\zeta = 2^{14}$,

$$A = \begin{bmatrix} \chi\cdot\zeta & -\zeta & \zeta \\ \zeta^{-1} & \zeta^{-1} & 0 \\ \zeta^{-1} & -\chi\cdot\zeta^{-1} & \zeta^{-1} \end{bmatrix} \approx \begin{bmatrix} 9.1553\cdot 10^{-5} & -1.6384\cdot 10^{4} & 1.6384\cdot 10^{4} \\ 6.1035\cdot 10^{-5} & 6.1035\cdot 10^{-5} & 0 \\ 6.1035\cdot 10^{-5} & -3.4106\cdot 10^{-13} & 6.1035\cdot 10^{-5} \end{bmatrix}$$

и $b = A\cdot[1, 1+\varepsilon, 1]^T$. Матрица A может быть вычислена без каких-либо округлений, но при вычислении b округления происходят, что означает: b лишь приближенно принадлежит линейной оболочке столбцов матрицы A, поэтому система $Ax = b$ не имеет решений. Проводя гауссово исключение, получим

$$L \approx \begin{bmatrix} 1 & 0 & 0 \\ .66666 & 1 & 0 \\ .66666 & 1.0000 & 1 \end{bmatrix}$$

и

$$U \approx \begin{bmatrix} 9.1553\cdot 10^{-5} & -1.6384\cdot 10^{4} & 1.6384\cdot 10^{4} \\ 0 & 1.0923\cdot 10^{4} & -1.0923\cdot 10^{4} \\ 0 & 0 & 1.8190\cdot 10^{-12} \end{bmatrix},$$

что дает для обратной матрицы значение

$$A^{-1} \approx \begin{bmatrix} 2.0480\cdot 10^{3} & 5.4976\cdot 10^{11} & -5.4976\cdot 10^{11} \\ -2.0480\cdot 10^{3} & -5.4976\cdot 10^{11} & 5.4976\cdot 10^{11} \\ -2.0480\cdot 10^{3} & -5.4976\cdot 10^{11} & 5.4976\cdot 10^{11} \end{bmatrix}.$$

Отсюда следует, что в вычисленной матрице $|A^{-1}|\cdot|A|$ все элементы приблизительно равны числу $6.7109\cdot 10^7$, поэтому вычисленное $\kappa_{CR}(A)$ есть величина порядка $O(10^7)$. Другими словами, наша оценка для ошибки указывает, что в вычисленном решении имеется $16 - 7 = 9$ верных разрядов, тогда как, в действительности, ни один не верен.

Если исключить сильный рост элементов, то можно показать, что феномен, иллюстрируемый данным примером, не может сделать слишком малой оценку (2.13) (где число $\|r\|$ надлежащим образом увеличено).

Кахан нашел еще семейство вырожденных $n\times n$-матриц, для которых замена одного очень малого элемента (равного $\approx 2^{-n}$) нулем понижает значение $\kappa_{CR}(A)$ до величины порядка $O(n^3)$. Аналогичным образом можно было бы построить примеры, где A не является точно вырожденной матрицей, так что оценки (2.13) и (2.14) верны в точной арифметике, но, вычисленные в машинной арифметике, становятся слишком малы. ◇

2.5. Улучшение точности приближенного решения

Мы только что видели, что ошибка в вычисленном решении системы $Ax = b$ может достигать уровня $\kappa(A)\varepsilon$. Что делать, если эта ошибка слишком велика? Одна из возможностей состоит в том, чтобы провести все вычисление повторно, используя арифметику более высокой разрядности, однако это может потребовать больших затрат и времени, и памяти. К счастью, при не слишком большом $\kappa(A)$ имеются гораздо более экономные методы получения более точного решения.

При решении любого уравнения $f(x) = 0$ можно попытаться использовать метод Ньютона, чтобы улучшить приближенное решение x_i посредством формулы $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$. Применяя эту формулу к функции $f(x) = Ax - b$, получаем один шаг *итерационного уточнения*:

$$
\begin{aligned}
&r = Ax_i - b \\
&\text{решить систему } Ad = r \text{ относительно вектора } d \\
&x_{i+1} = x_i - d.
\end{aligned}
$$

Если бы мы могли точно вычислить вектор $r = Ax_i - b$ и решить систему $Ad = r$ точно, то все закончилось бы в один шаг, чего и следует ожидать от метода Ньютона, применяемого к линейной задаче. Ошибки округлений препятствуют этой моментальной сходимости. Алгоритм представляет теоретический и практический интерес именно в ситуации, когда матрица A настолько плохо обусловлена, что можно получить лишь очень неточное решение системы $Ad = r$ (как и системы $Ax_0 = b$).

Теорема 2.7. *Предположим, что вектор r вычисляется в арифметике двойной точности, и пусть $\kappa(A) \cdot \varepsilon < c \equiv \frac{1}{3n^3 g + 1} < 1$, где n — порядок матрицы A, а g — коэффициент роста. Тогда, повторяя шаги итерационного уточнения, получим вектор x_i, для которого*

$$
\frac{\|x_i - A^{-1}b\|_\infty}{\|A^{-1}b\|_\infty} = O(\varepsilon).
$$

Заметим, что в этой оценке ошибки не присутствует число обусловленности. Это означает, что решение системы можно вычислить с высокой точностью независимо от значения числа обусловленности при условии, что произведение $\kappa(A)\varepsilon$ значительно меньше, чем 1. (На практике, с является слишком консервативной верхней границей и алгоритм часто работает успешно, даже если $\kappa(A)\varepsilon > c$.)

Набросок доказательства. Чтобы доказательство оставалось прозрачным, будем учитывать лишь наиболее важные ошибки округлений. Для краткости, вместо $\|\cdot\|_\infty$ будем писать $\|\cdot\|$. Наша цель — установить неравенство

$$
\|x_{i+1} - x\| \le \frac{\kappa(A)\varepsilon}{c}\|x_i - x\| \equiv \zeta\|x_i - x\|.
$$

По предположению, $\zeta < 1$, поэтому из неравенства следует, что ошибка $\|x_{i+1} - x\|$ монотонно убывает до нуля. (На практике, ее убывание в какой-то момент прекратится вследствие ошибок округления при присваиваниях $x_{i+1} = x_i - d_i$, которые в нашем анализе игнорируются.)

Оценим сначала ошибку в вычисленной невязке r. Имеем $r = \mathrm{fl}(Ax_i - b) = Ax_i - b + f$, где, согласно результату из вопроса 1.10, $|f| \leq n\varepsilon^2(|A| \cdot |x_i| + |b|) + \varepsilon|Ax_i - b| \approx \varepsilon|Ax_i - b|$. Член порядка ε^2 возникает при вычислении r с двойной точностью, а член порядка ε — при округлении вычисленного r до числа с обычной точностью. Поскольку $\varepsilon^2 \ll \varepsilon$, то мы пренебрежем членом порядка ε^2 в оценке для $|f|$.

Далее, имеем $(A + \delta A)d = r$, где, вследствие оценки (2.11), $\|\delta A\| \leq \gamma\varepsilon\|A\|$ и $\gamma = 3n^3g$, хотя эта оценка обычно слишком груба. Как уже упоминалось, мы упрощаем анализ предположением, что присваивание $x_{i+1} = x_i - d$ выполняется точно.

Продолжая игнорировать члены порядка ε^2, получаем

$$\begin{aligned} d &= (A + \delta A)^{-1}r = (I + A^{-1}\delta A)^{-1}A^{-1}r \\ &= (I + A^{-1}\delta A)^{-1}A^{-1}(Ax_i - b + f) \\ &= (I + A^{-1}\delta A)^{-1}(x_i - x + A^{-1}f) \\ &\approx (I - A^{-1}\delta A)(x_i - x + A^{-1}f) \\ &\approx x_i - x - A^{-1}\delta A(x_i - x) + A^{-1}f. \end{aligned}$$

Следовательно, $x_{i+1} - x = x_i - d - x = A^{-1}\delta A(x_i - x) - A^{-1}f$, а потому

$$\begin{aligned} \|x_{i+1} - x\| &\leq \|A^{-1}\delta A(x_i - x)\| + \|A^{-1}f\| \\ &\leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|x_i - x\| + \|A^{-1}\| \cdot \varepsilon \cdot \|Ax_i - b\| \\ &= \|A^{-1}\| \cdot \|\delta A\| \cdot \|x_i - x\| + \|A^{-1}\| \cdot \varepsilon \cdot \|A(x_i - x)\| \\ &\leq \|A^{-1}\| \cdot \gamma\varepsilon \cdot \|A\| \cdot \|x_i - x\| \\ &+ \|A^{-1}\| \cdot \|A\| \cdot \varepsilon \cdot \|x_i - x\| \\ &= \|A^{-1}\| \cdot \|A\| \cdot \varepsilon \cdot (\gamma + 1) \cdot \|x_i - x\|. \end{aligned}$$

Таким образом, из неравенства

$$\zeta = \|A^{-1}\| \cdot \|A\| \cdot \varepsilon(\gamma + 1) = \kappa(A)\varepsilon/c < 1$$

вытекает сходимость итераций. □

Итерационное уточнение или иные варианты метода Ньютона могут использоваться для улучшения приближенных решений и во многих других задачах линейной алгебры.

2.5.1. Итерационное уточнение в арифметике обычной точности

Этот раздел может быть опущен при первом чтении книги.

Иногда для проведения итерационного уточнения отсутствует предусматриваемая им возможность выполнять вычисления с двойной точностью. Если, например, уже входные данные суть числа с двойной точностью, то нам потребовалось бы вычислять невязку с *учетверенной* точностью, что не всегда возможно. На некоторых машинах (скажем, PC с процессором Intel Pentium) доступна расширенная двойная точность, прибавляющая к обычной двойной точности 11 дополнительных двоичных разрядов (см. разд. 1.5). Это меньше, чем нужно для учетверенной точности (которая потребовала бы, по меньшей мере, $2 \cdot 53 = 106$ двоичных разрядов), но все же ощутимо повышает основную точность.

Однако если ни одна из этих возможностей не доступна, итерационное уточнение все-таки можно использовать, вычисляя невязку r с обычной точностью (т. е. той точностью, какую имеют входные данные). В этом случае теорема 2.7 теряет силу. С другой стороны, теорема, приводимая ниже, показывает, что при некоторых технических предположениях имеет смысл выполнить один шаг итерационного уточнения в обычной точности, поскольку это уменьшает покомпонентную относительную обратную ошибку, определенную в теореме 2.3, до уровня $O(\varepsilon)$. Если соответствующее относительное число обусловленности $\kappa_{CR}(A) = \||A^{-1}| \cdot |A|\|$ (см. разд. 2.2.1) значительно меньше обычного числа обусловленности $\kappa(A) = \|A^{-1}\|_\infty \cdot \|A\|_\infty$, то полученное решение будет точней исходного.

Теорема 2.8. *Предположим, что вектор r вычисляется с обычной точностью, и пусть*

$$\|A^{-1}\|_\infty \cdot \|A\|_\infty \cdot \frac{\max_i(|A| \cdot |x|)_i}{\min_i(|A| \cdot |x|)_i} \cdot \varepsilon < 1.$$

Тогда один шаг итерационного уточнения дает вектор x_1, такой, что $(A + \delta A)x_1 = b + \delta b$, где $|\delta a_{ij}| = O(\varepsilon)|a_{ij}|$ и $|\delta b_i| = O(\varepsilon)|b_i|$. Иными словами, покомпонентная относительная обратная ошибка для x_1 имеет минимальный возможный порядок малости. Это означает, например, что при разреженных A и b возмущения δA и δb имеют ту же структуру разреженности, что и, соответственно, A и b.

Доказательство теоремы можно найти в [149]; относительно других деталей см. [14, 225, 226, 227].

Итерационное уточнение с обычной точностью и оценка ошибки (2.14) реализованы в LAPACK'е, например, программой `sgesvx`.

Пример 2.7. Рассмотрим те же матрицы, что в примере 2.5, и проделаем один шаг итерационного уточнения в той же точности, с какой выполнялись остальные вычисления ($\varepsilon \approx 10^{-16}$). Для этих матриц обычное число обусловленности $\kappa(A) \approx 10^{14}$, тогда как $\kappa_{CR}(A) = 1$, поэтому нужно ожидать большого улучшения точности. В самом деле, покомпонентная относительная ошибка метода GEPP и соответствующая ошибка из формулы (2.14) становятся меньше 10^{-15}. Matlab-программу для этого примера можно найти по адресу HOMEPAGE/Matlab/pivot.m. ◇

2.5.2. Уравновешивание

Имеется еще один общий прием для повышения точности при решении линейных систем, называемый *уравновешиванием*. Под ним понимают выбор подходящей диагональной матрицы D с последующим решением системы $DAx = Db$ вместо $Ax = b$. Матрицу выбирают с таким расчетом, чтобы для DA число обусловленности было меньше, чем для A (если это возможно). Так, в примере 2.7, беря в качестве d_{ii} число, обратное 2-норме i-й строки матрицы A, получим матрицу DA, очень близкую к единичной; тем самым число обусловленности снижается с 10^{14} до 1. Можно показать, что подобный выбор матрицы D всегда уменьшает число обусловленности произведения DA до уровня, не более чем в $\sqrt{n}$ раз превышающего минимум по всем возможным выборам диагональной

матрицы D [244]. На практике можно взять и две диагональные матрицы D_{row} и D_{col} с тем, чтобы решать систему $(D_{row}AD_{col})\overline{x} = D_{row}b$, $x = D_{col}\overline{x}$.

Методы итерационного уточнения и уравновешивания реализованы соответственно LAPACK-подпрограммами `sgerfs` и `sgeequ`. В свою очередь, к ним обращаются программы-драйверы типа `sgesvx`.

2.6. Блочные алгоритмы как средство повышения производительности

В конце раздела 2.3 было указано, что, в зависимости от используемого компьютера и решаемой задачи, изменение порядка трех вложенных циклов в алгоритме 2.2, реализующем гауссово исключение, может изменить скорость работы алгоритма на несколько порядков. В этом разделе мы исследуем, почему это возможно, и опишем некоторые линейно-алгебраические программы, учитывающие такую возможность и написанные с особой тщательностью. Речь идет о так называемых *блочных алгоритмах*, самые внутренние циклы которых оперируют с квадратными или прямоугольными блоками матрицы, а не ее строками или столбцами. Соответствующие программы имеются в общедоступных библиотеках, таких, как LAPACK (фортранная версия по адресу NETLIB/lapack)[1] и ScaLAPACK (см. NETLIB/scalapack). LAPACK и его версии на других языках предназначены для персональных компьютеров, рабочих станций, векторных компьютеров и параллельных машин с разделяемой памятью. В их число входят Sun SPARC-center 2000 [238], SGI Power Challenge [223], DEC Alpha Server 8400 [103] и Cray C90/J90 [253, 254]. Для параллельных машин с распределенной памятью, таких, как IBM SP-2 [256], Intel Paragon [257], серия Cray T3 [255] и сети рабочих станций [9], разработана библиотека ScaLAPACK. Упомянутые библиотеки и подробные руководства для пользования ими доступны в NETLIB'е [10, 34].

Более подробное обсуждение алгоритмов для высокопроизводительных (в особенности, параллельных) компьютеров можно найти в Интернете по адресу PARALLEL_HOMEPAGE.

Первоначальным стимулом разработки библиотеки LAPACK была крайне неудовлетворительная производительность ее предшественниц, библиотек LINPACK и EISPACK (также доступных в NETLIB'е) на некоторых мощных компьютерах. Рассмотрим, например, приводимую ниже таблицу, показывающую скорость в мегафлопах LINPACK-программы `spofa` (реализующей метод Холесского) на машине Cray YMP, суперкомпьютере конца 80-х годов. Метод Холесского — это вариант гауссова исключения, предназначенный для симметричных положительно определенных матриц. Он подробно обсуждается в разд. 2.7; пока же нам достаточно знать, что он весьма схож с алгоритмом 2.2. В таблице указана также скорость нескольких других стандартных линейно-алгебраических операций. Cray YMP — это параллельный компьютер, в котором одновременно могут работать до 8 процессоров. Поэтому мы приводим два столбца данных: для однопроцессорного и восьмипроцессорного режимов.

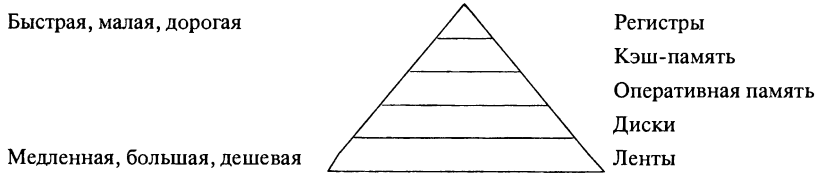
[1] Имеется также C-версия LAPACK'а, называемая CLAPACK (см. NETLIB/clapack). LAPACK++ (см. NETLIB/c++/lapack++) и LAPACK90 (см. NETLIB/lapack90) являются соответственно C++- и Fortran90-интерфейсами для LAPACK'а.

	1 Проц.	8 Проц.
Максимальная скорость	330	2640
Умножение двух матриц ($n = 500$)	312	2425
Умножение матрицы на вектор ($n = 500$)	311	2285
Решение уравнений $TX = B$ ($n = 500$)	309	2398
Решение системы $Tx = b$ ($n = 500$)	272	584
LINPACK (Cholesky, $n = 500$)	72	72
LAPACK (Cholesky, $n = 500$)	290	1414
LAPACK (Cholesky, $n = 1000$)	301	2115

Максимальная скорость компьютера, приведенная в верхней строке, является верхней границей для последующих чисел. Данные по основным линейно-алгебраическим операциям в следующих четырех строках получены с помощью программ, специально разработанных для достижения высокой производительности на машине Cray YMP. Эти данные достаточно близки к максимальной скорости компьютера; исключение составляет программа для задачи $Tx = b$ (решение единственной треугольной системы линейных уравнений), использующая наличие 8 процессоров неэффективно. Задача $TX = B$ — это решение треугольной системы со многими правыми частями (B — квадратная матрица). Данные в таблице относятся к большим векторам и матрицам ($n = 500$).

Программа метода Холесского из LINPACK'а (см. шестую строку таблицы) работает намного медленнее предыдущих программ при том, что применяется к столь же большой матрице и выполняет математически сходные операции. Подобная плохая производительность побудила нас модифицировать эту и другие линейно-алгебраические программы с тем, чтобы заставить их работать столь же быстро, как более простые программы типа матричного умножения. Скорости этих модифицированных программ из LAPACK'а приведены в двух последних строках таблицы. Очевидно, что LAPACK-программы гораздо ближе к максимальной производительности компьютера. Подчеркнем, что программы метода Холесского из обеих библиотек, LINPACK и LAPACK, выполняют одни и те же операции с плавающей точкой; различен лишь порядок операций.

Чтобы понять, как было достигнуто это ускорение, мы должны разобраться, на что тратится машинное время при выполнении программы. Это, в свою очередь, требует понимания того, как функционирует память компьютера. Оказывается, что память любого компьютера, от самой дешевой персональной машины до самого мощного суперкомпьютера, устроена *иерархически* и состоит из ряда различных видов памяти с очень быстрой, дорогостоящей и потому малой памятью на вершине иерархии и медленной, дешевой и очень большой памятью в ее подножии.



К примеру, регистры являются наискорейшим видом памяти; затем следуют кэш-память, основная память, диски и, наконец, ленты как самый медленный, большой и дешевый тип памяти. Полезные арифметические и логические операции могут производиться *только* с данными, находящимися на вершине иерархии, в регистрах. С одного уровня иерархической памяти данные могут пересылаться на соседние уровни, например перемещаться между основной памятью и диском. Скорость движения данных высока на вершине иерархии (между регистрами и кэш-памятью) и низка вблизи ее основания (между диском и основной памятью). В частности, скорость, с какой выполняется арифметика, выше скорости обмена данными на нижних уровнях иерархической памяти в десятки или даже тысяч раз, в зависимости от уровня. Это означает, что при непродуманной реализации алгоритма бóльшая часть времени будет им тратиться не на реальную работу, а на перемещение данных из основания иерархической памяти в регистры.

Вот пример простого алгоритма, в котором, к сожалению, нельзя избежать потери большей части времени на обмен данными, а не на полезную арифметику. Предположим, что нужно сложить две $n \times n$-матрицы настолько большие, что они могут поместиться лишь в большом и медленном уровне иерархической памяти. Для сложения матрицы частями должны быть пересланы в регистры, где и производятся сложения элементов, а суммы должны быть пересланы в обратном направлении. Таким образом, на каждое выполняемое сложение приходится ровно 3 пересылки между быстрой и медленной памятью (считывание двух слагаемых в быструю память и запись одной суммы в медленную память). Пусть t_{arith} (секунд) — время выполнения операции с плавающей точкой и t_{mem} (секунд) — время пересылки одного слова данных из одного уровня памяти в другой, причем $t_{\text{mem}} \gg t_{\text{arith}}$. Тогда время работы всего алгоритма составит $n^2(t_{\text{arith}} + 3t_{\text{mem}})$, что намного больше времени $n^2 t_{\text{arith}}$, требуемого для арифметики самой по себе. Это означает, что сложение матриц обречено выполняться со скоростью самого медленного уровня памяти, хранящего матрицы, а не с гораздо более высокой скоростью сложения чисел. В противоположность этому, как мы увидим позднее, для некоторых других операций, например матричного умножения, можно добиться скорости, характерной для наиболее быстрого уровня памяти, даже если данные первоначально хранятся в наиболее медленном уровне.

Программа метода Холесского из LINPACK'а работает столь медленно потому, что в ней не предусмотрена минимизация движения данных между уровнями памяти на таких машинах, как Cray YMP[1]. Напротив, для матричного умножения и трех других основных алгоритмов линейной алгебры, указанных в таблице, такая минимизация была проделана.

2.6.1. Основные подпрограммы линейной алгебры

Разрабатывать для всякого нового типа компьютеров специальную версию каждой программы, вроде программы метода Холесского — это явно не эффективное решение. Требуется более систематический подход к этой проблеме. Учитывая, что операции типа матричного умножения столь распространены

[1] Хотя в ней была предусмотрена минимизация другого параметра обмена между основной памятью и диском, а именно страничных отказов (*page faults*.)

в вычислениях, производители компьютеров каталогизировали их как «*Основные подпрограммы линейной алгебры*» (*Basic Lenear Algebra Subroutines*), или BLAS [169, 89, 87], и оптимизировали их для своих машин. Другими словами, для высокопроизводительных компьютеров (как и ряда других машин) существуют библиотеки подпрограмм, включающие в себя умножение матриц, матрично-векторное умножение и другие подобные операции, со стандартным интерфейсом на фортране или C, однако тела этих подпрограмм оптимизированы для каждого типа компьютеров. Наша цель — использовать этот оптимизированный пакет BLAS, модифицируя алгоритмы типа алгоритма Холесского так, чтобы они бóльшую часть работы выполняли с помощью BLAS-подпрограмм.

В этом разделе мы дадим общее описание пакета BLAS. В разд. 2.6.2 будет рассмотрена задача оптимизации матричного умножения. Наконец, в разд. 2.6.3 мы покажем, как модифицировать гауссово исключение с тем, чтобы бóльшая часть его работы производилась посредством матричного умножения.

Рассмотрим пакет BLAS более подробно. В таблице 2.1 подсчитаны число обращений к памяти и число операций с плавающей точкой для трех родственных BLAS-подпрограмм. Например, число обращений к памяти для операции `saxpy` в верхней строке таблицы равно $3n + 1$, поскольку требуется переслать из медленной памяти в регистры n значений x_i, n значений y_i и значение α, а затем записать n значений y_i снова в медленную память. В последнем столбце дано значение (точнее, его старший член по n) отношения q числа флопов к числу обращений к памяти.

Смысл коэффициента q в том, что он показывает приблизительное число флопов, приходящихся на одно обращение к памяти, иначе говоря, сколько полезной работы приходится на время, затрачиваемое на перемещение данных. Эта характеристика показывает, насколько быстро алгоритм может работать в *принципе*. Предположим, например, что в алгоритме выполняется f операций с плавающей точкой с затратой t_{arith} секунд на каждую и m обращений к памяти, каждое из которых требует t_{mem} секунд. Тогда общее время работы алгоритма составит

$$f \cdot t_{\text{arith}} + m \cdot t_{\text{mem}} = f \cdot t_{\text{arith}} \cdot \left(1 + \frac{m}{f}\frac{t_{\text{mem}}}{t_{\text{arith}}}\right) = f \cdot t_{\text{arith}} \cdot \left(1 + \frac{1}{q}\frac{t_{\text{mem}}}{t_{\text{arith}}}\right).$$

Мы предполагаем, что арифметика и обращения к памяти не производятся одновременно. Таким образом, чем больше q, тем ближе время работы алгоритма к наилучшему возможному значению $f \cdot t_{\text{arith}}$; так быстро алгоритм работал бы, если бы все данные находились в регистрах. Это означает, что алгоритмы с большими значениями q являются наилучшими строительными блоками для других алгоритмов.

В таблице 2.1 отражена иерархия операций. Операции типа `saxpy` работают с векторами, производят $O(n^1)$ флопов и имеют наихудшие значения q. Они составляют уровень 1 пакета BLAS, или BLAS1 [169]; к этому уровню относятся скалярное произведение векторов, умножение вектора на число и другие простые операции. Такие операции, как умножение матрицы на вектор, работают с матрицами и векторами, производят $O(n^2)$ флопов и имеют несколько лучшие значения q. Они составляют уровень 2 пакета BLAS, или BLAS2 [89, 88]; он включает в себя решение треугольных систем уравнений и матричные

Таблица 2.1. Подсчет числа операций с плавающей точкой (f) и числа обращений к памяти (m) для BLAS-подпрограмм.

Операция	Определение	f	m	$q = f/m$
`saxpy` (BLAS1)	$y = \alpha \cdot x + y$ или $y_i = \alpha x_i + y_i$ $i = 1, \ldots, n$	$2n$	$3n + 1$	$2/3$
Матрично-векторное умножение (BLAS2)	$y = A \cdot x + y$ или $y_i = \sum_{j=1}^{n} a_{ij} x_j + y_i$ $i = 1, \ldots, n$	$2n^2$	$n^2 + 3n$	2
Умножение матриц (BLAS3)	$C = A \cdot B + C$ или $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{jk} + c_{ij}$ $i, j = 1, \ldots, n$	$2n^3$	$4n^2$	$n/2$

модификации ранга 1 (т. е. операцию $A + xy^T$, где x и y — векторы-столбцы). Операции типа матричного умножения работают с парами матриц, производят $O(n^3)$ флопов и имеют наилучшие значения q. Они составляют уровень 3 пакета BLAS, или BLAS3 [87, 86]; к их числу относится решение треугольной системы уравнений со многими правыми частями.

Директория NETLIB/blas содержит документацию и (неоптимизированные) реализации всех BLAS-подпрограмм. Краткое описание всего пакета можно найти на странице NETLIB/blas/blasqr.ps. Это описание дано также в [10, приложение C] (см., кроме того, NETLIB/lapack/lug/lapack_lug.html).

Поскольку наивысшие значения q соответствуют уровню 3, мы попробуем модифицировать наши алгоритмы так, чтобы они опирались на операции типа матричного умножения, а не на матрично-векторное умножение или операцию `saxpy`. (LINPACK-подпрограмма метода Холесского организована как последовательность обращений к процедуре `saxpy`.) Подчеркнем, что модифицированные алгоритмы станут быстрее лишь при использовании оптимизированных BLAS-подпрограмм.

2.6.2. Как оптимизировать матричное умножение

Исследуем подробно задачу реализации матричного умножения $C = A \cdot B + C$ с целью минимизировать движение данных и тем самым повысить производительность. Мы увидим, что получаемая производительность сильно зависит от деталей реализации. Чтобы упростить обсуждение, используем следующую модель компьютера. Предполагается, что матрицы хранятся по столбцам, как это принято в фортране. (Приводимые ниже примеры легко модифицировать для случая, когда матрицы хранятся по строкам, как в C.) Считаем, что в иерархической памяти только два уровня, быстрый и медленный. При этом медленная память достаточно велика для того, чтобы в ней поместить три $n \times n$-матрицы A, B и C. В то же время, быстрая память состоит лишь из M слов, где $2n < M \ll n^2$; это означает, что быстрая память может вместить в себя два столбца или две строки матрицы, но не матрицу целиком. Мы предположим еще, что программист в состоянии управлять движением данных. (На

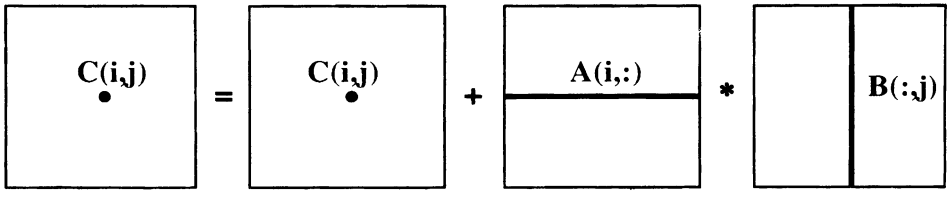
практике движение данных может управляться автоматическими устройствами типа контроллера кэш-памяти. Тем не менее основная схема оптимизации не меняется и в этом случае.)

Простейший возможный алгоритм матричного умножения состоит из трех вложенных циклов, которые мы аннотировали, чтобы указать движения данных.

Алгоритм 2.6. *Неблочная версия матричного умножения (аннотированная с целью показать работу памяти)*

```
for i = 1 to n
    { Считать строку i матрицы A в быструю память }
    for j = 1 to n
        { Считать C_ij в быструю память }
        { Считать столбец j матрицы B в быструю память }
        for k = 1 to n
            C_ij = C_ij + A_ik · B_kj
        end for
        { Записать C_ij обратно в медленную память }
    end for
end for
```

Самый внутренний цикл состоит в модификации элемента C_{ij} скалярным произведением i-й строки матрицы A и j-го столбца матрицы B, что иллюстрируется следующим рисунком:

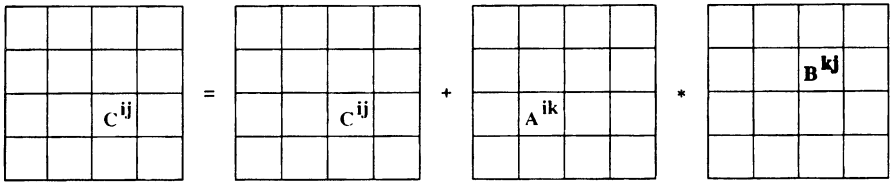


Два внутренних цикла (по j и k) можно еще описать как модификацию i-й строки матрицы C посредством векторно-матричного произведения: i-я строка матрицы A умножается на матрицу B. Это показывает, что мы не сможем получить производительность, более высокую, чем скорость операций из уровней BLAS1 и BLAS2, поскольку именно такие операции используются во внутренних циклах алгоритма.

Приведем подробный подсвет числа обращений к памяти: n^3 при n-кратном считывании матрицы B (по одному разу для каждого значения i), n^2 при считывании матрицы A (считываем по очереди каждую строку и храним ее в быстрой памяти, пока она не становится далее ненужной) и $2n^2$ обращений для матрицы C (каждый элемент C_{ij} поочередно считывается в быструю память, хранится там, пока не закончится его модификация, а затем снова записывается в медленную память). Всего получается $n^3 + 3n^2$ обращений, что дает $q = 2n^3/(n^3 + 3n^2) \approx 2$, т. е. значение, не лучшее, чем в BLAS2, и далекое от наилучшего возможного значения $n/2$ (см. табл. 2.1). Если $M \ll n$, так что в быстрой памяти нельзя разместить даже одну полную строку матрицы A, то

значение q уменьшается до 1, поскольку алгоритм превращается в многократное вычисление скалярных произведений, что соответствует уровню BLAS1. Каждая перестановка трех циклов по i, j и k дает иной алгоритм, но все они имеют приблизительно одно и то же значение коэффициента q.

Мы предпочтем этим алгоритмам метод, использующий *блочное разбиение*: C рассматривается как блочная матрица размера $N \times N$, где C^{ij} — блоки порядка n/N; таким же образом разбиваются на блоки матрицы A и B. Рисунок иллюстрирует это для случая $N = 4$, указывая заодно модификацию алгоритма.



$$\boxed{C^{ij}} = \boxed{C^{ij}} + \sum_{k=1}^{N} \boxed{A^{ik}} * \boxed{B^{kj}}$$

Алгоритм 2.7. *Блочная версия матричного умножения (аннотированная с целью показать работу памяти)*

for $i = 1$ *to* N
  for $j = 1$ *to* N
  { *Считать* C^{ij} *в быструю память* }
    for $k = 1$ *to* N
      { *Считать* A^{ik} *в быструю память* }
      { *Считать* B^{kj} *в быструю память* }
      $C^{ij} = C^{ij} + A^{ik}B^{kj}$
    end for
  { *Записать* C^{ij} *обратно в медленную память* }
  end for
end for

Число обращений к памяти подсчитывается следующим образом. Каждый блок C^{ij} считывается и записывается обратно по одному разу, что дает $2n^2$ обращений; матрица A в целом считывается N раз (поскольку каждая $n/N \times n/N$-подматрица A^{ik} считывается N раз), это дает Nn^2 обращений; столько же обращений требует матрица B (каждая подматрица B^{kj} считывается N раз). Всего получается $(2N + 2)n^2 \approx 2Nn^2$ обращений. Таким образом, чтобы минимизировать число обращений к памяти, нужно взять как можно меньшее N. Однако N подчиняется ограничению $M \geq 3(n/N)^2$, означающему, что быстрая память должна одновременно вмещать в себя по одному блоку матриц A, B и C. Отсюда получаем $N \approx n\sqrt{3/M}$, а потому $q \approx (2n^3)/(2Nn^2) \approx \sqrt{M/3}$, что значительно лучше, чем в предыдущем алгоритме. В частности, значение q растет вместе с M независимо от n; можно ожидать поэтому, что алгоритм эффективен для любого порядка n и его

эффективность растет с увеличением объема M быстрой памяти. Оба этих свойства весьма привлекательны.

Можно показать, что, в действительности, алгоритм 2.7 асимптотически является оптимальным [151]. Иными словами, ни при какой организации матричного умножения (с выполнением прежнего числа $2n^3$ арифметических операций) нельзя получить значение q, большее, чем $O(\sqrt{M})$.

С другой стороны, наш краткий анализ игнорирует ряд практических вопросов:

1. Реальная программа должна уметь работать и с прямоугольными матрицами, где при оптимальном разбиении блоки не обязательно являются квадратными.
2. Наилучшие размеры подматриц сильно зависят от структуры кэш-памяти и регистров компьютера.
3. Могут существовать специальные аппаратные возможности для выполнения умножения и сложения в одном цикле. Может также быть возможным одновременное выполнение нескольких независимых операций умножения-сложения.

Подробное обсуждение этих вопросов для конкретной высокопроизводительной машины, а именно рабочей станции IBM RS6000/590, можно найти в [1] (см. также PARALLEL_HOMEPAGE или http://www.rs6000.ibm.com/resource/technology/essl.html). На рис. 2.5 показаны скорости трех уровней BLAS'а для этой машины. По горизонтальной оси откладывается размерность матрицы, а по вертикальной — скорость в мегафлопах. Пиковая производительность машины составляет 266 мегафлопов. Верхняя кривая (с максимальным значением

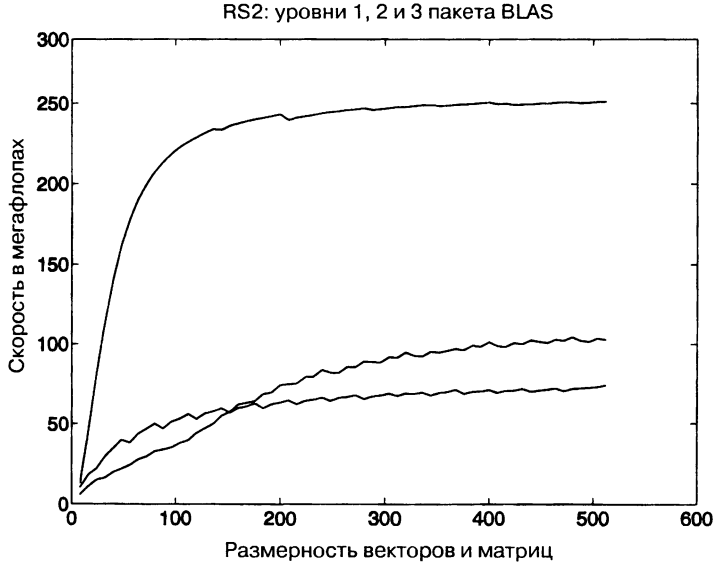


Рис. 2.5. Производительность пакета BLAS на компьютере IBM RS 6000/590.

$\approx$ 250 мегафлопов) соответствует умножению квадратных матриц. Средняя кривая (с максимальным значением $\approx$ 100 мегафлопов) относится к операции умножения квадратной матрицы на вектор. Нижняя кривая (с максимальным значением $\approx$ 75 мегафлопов) отвечает операции `saxpy`. Заметим, что для больших матриц скорость увеличивается. Такой рост наблюдается всегда, поэтому мы постараемся строить алгоритмы с как можно бóльшими матрицами во внутренних матричных умножениях.

Оба приведенных выше алгоритма для умножения матриц выполняют $2n^3$ арифметических операций. Оказывается, что существуют другие алгоритмы матричного умножения, в которых число операций гораздо меньше. Первым из таких алгоритмов был метод Штрассена [3]; он же допускает наиболее простое объяснение. В этом методе сомножители рассматриваются как блочные 2×2-матрицы, умножение которых сводится к семи умножениям и 18 сложениям блоков половинного порядка. Эта идея используется рекурсивно, что дает асимптотическую сложность $n^{\log_2 7} \approx n^{2.81}$ вместо n^3.

Алгоритм 2.8. *Метод Штрассена для матричного умножения*

C = *Strassen(A,B,n)*
/* *Алгоритм вычисляет произведение* $C = A * B$ $n \times n$-*матриц* A *и* B
Предполагается, что n *есть степень двойки* */
if $n = 1$
вычислить $C = A * B$ /* *умножение чисел* */
else
Представить матрицы в блочном виде

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

где блоки A_{ij} *и* B_{ij} *имеют порядок* $n/2$
$P_1 = Strassen(A_{12} - A_{22}, B_{21} + B_{22}, n/2)$
$P_2 = Strassen(A_{11} + A_{22}, B_{11} + B_{22}, n/2)$
$P_3 = Strassen(A_{11} - A_{21}, B_{11} + B_{12}, n/2)$
$P_4 = Strassen(A_{11} + A_{12}, B_{22}, n/2)$
$P_5 = Strassen(A_{11}, B_{12} - B_{22}, n/2)$
$P_6 = Strassen(A_{22}, B_{21} - B_{11}, n/2)$
$P_7 = Strassen(A_{21} + A_{22}, B_{11}, n/2)$
$C_{11} = P_1 + P_2 - P_4 + P_6$
$C_{12} = P_4 + P_5$
$C_{21} = P_6 + P_7$
$C_{22} = P_2 - P_3 + P_5 - P_7$
возвратить матрицу $C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$
end if

Проверка по индукции того, что этот алгоритм правильно перемножает матрицы, является несложным, хотя и довольно скучным упражнением (см. вопрос 2.21). Покажем, что его сложность $O(n^{\log_2 7})$. Обозначим через $T(n)$ число сложений, вычитаний и умножений, выполняемых в алгоритме. Поскольку алгоритм предусматривает 7 обращений к самому себе с матрицами порядка

$n/2$ и 18 сложений $n/2 \times n/2$-матриц, можно составить рекуррентное соотношение $T(n) = 7T(n/2) + 18(n/2)^2$. Замена переменного $m = \log_2 n$ дает новую рекурсию $\overline{T}(m) = 7\overline{T}(m-1) + 18(2^{m-1})^2$, где $\overline{T}(m) = T(2^m)$. Можно проверить, что решение этого линейного разностного уравнения относительно $\overline{T}$ имеет вид $\overline{T}(m) = O(7^m) = O(n^{\log_2 7})$.

Значение алгоритма Штрассена состоит не только в этой меньшей асимптотической сложности, но и в сведении задачи к решению меньших подзадач. Продолжая редукцию, в конечном счете получим подзадачи, помещающиеся в быстрой памяти; как только это достигнуто, можно пользоваться стандартным алгоритмом матричного умножения. Для некоторых компьютеров и матриц относительно высокого порядка этим путем было достигнуто ускорение умножения [22]. Недостатками данного подхода являются большие затраты памяти и несколько худшая численная устойчивость, хотя и достаточная для многих задач [77]. Существует ряд еще более быстрых алгоритмов матричного умножения; в настоящее время рекордный по скорости алгоритм, предложенный Виноградом и Копперсмитом, имеет сложность $O(n^{2.376})$. Однако подобные алгоритмы становятся экономичней алгоритма Штрассена лишь для столь больших значений n, какие не встречаются в практических задачах. Обзор этой тематики см. в [195].

2.6.3. Реорганизация гауссова исключения с целью использования уровня 3 пакета BLAS

Мы модифицируем гауссово исключение с тем, чтобы вначале использовать уровень 2 пакета BLAS, а затем уровень 3. Для простоты, предположим, что выбор главного элемента не нужен.

В действительности, алгоритм 2.4 уже является алгоритмом уровня 2, поскольку бóльшая часть работы производится в его второй строке $A(i+1:n, i+1:n) = A(i+1:n, i+1:n) - A(i+1:n, i) * A(i, i+1:n)$, описывающей *модификацию ранга* 1 подматрицы $A(i+1:n, i+1:n)$. Прочая арифметика в алгоритме, т. е. $A(i+1:n, i) = A(i+1:n, i)/A(i, i)$, выполняется на самом деле путем умножения вектора $A(i+1:n, i)$ на число $1/A(i, i)$ по той причине, что умножение значительно быстрее деления; это также BLAS-операция, но уровня 1. Нам придется немного изменить алгоритм 2.4, чтобы воспользоваться им внутри версии уровня 3.

Алгоритм 2.9. *Реализация LU-разложения без выбора главных элементов с помощью уровня 2 пакета BLAS. Матрица A имеет размер $m \times n$, где $m \geq n$; на ее место записываются $m \times n$-матрица L и $n \times n$-матрица U. Для последующих ссылок наиболее важные строки помечены номерами.*

```
    for i = 1 to min(m − 1, n)
(1)     A(i + 1 : m, i) = A(i + 1 : m, i)/A(i, i)
        if i < n
(2)         A(i + 1 : m, i + 1 : n) = A(i + 1 : m, i + 1 : n)−
                A(i + 1 : m, i) · A(i, i + 1 : n)
    end for
```

Левая часть рис. 2.6 иллюстрирует алгоритм 2.9 в применении к квадратной матрице. К началу i-го шага алгоритма уже вычислены столбцы матрицы

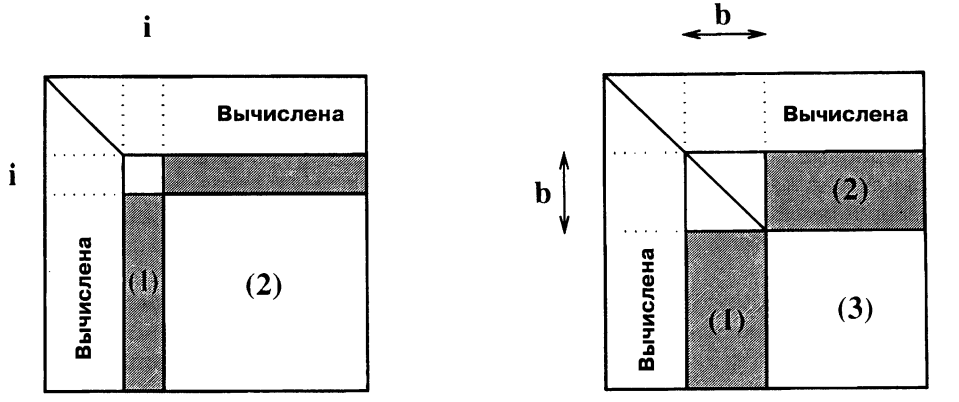


Рис. 2.6. Реализации LU-разложения с помощью уровней 2 и 3 пакета BLAS.

L и строки матрицы U с номерами от 1 до $i-1$; на i-м шаге должны быть вычислены i-й столбец в L и i-я строка в U, а для оставшейся подматрицы в A должна быть проведена модификация ранга 1. Подматрицы на рисунке пронумерованы в соответствии со строками алгоритма ((1) или (2)), в которых они подвергаются пересчету. Модификация ранга 1 в строке (2) заключается в вычитании из подматрицы, помеченной номером (2), произведения закрашенных столбца и строки.

Указанные вычисления будут реорганизованы в алгоритме уровня 3 следующим образом: модификация подматрицы (2) будет *отложена* на b шагов, где b — малое число, называемое *размером блока*, после чего b модификаций ранга 1 будут произведены единовременно как одно матричное умножение. Чтобы понять, как это сделать, предположим, что первые $i-1$ столбцов в L и строк в U уже вычислены. Тогда

$$A = \begin{matrix} \\ i-1 \\ b \\ n-b-i+1 \end{matrix} \begin{matrix} \begin{matrix} i-1 & b & n-b-i+1 \end{matrix} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \end{matrix}$$

$$= \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & I & 0 \\ L_{31} & 0 & I \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & \tilde{A}_{22} & \tilde{A}_{23} \\ 0 & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix},$$

где все матрицы разбиты на блоки одинаковым образом. Ситуацию иллюстрируют правая часть рис. 2.6. Применяя алгоритм 2.9 к подматрице

$$\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix},$$

находим

$$\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix} = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} \cdot U_{22} = \begin{bmatrix} L_{22}U_{22} \\ L_{32}U_{22} \end{bmatrix}.$$

Это позволяет написать

$$\begin{bmatrix} \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} = \begin{bmatrix} L_{22}U_{22} & \tilde{A}_{23} \\ L_{32}U_{22} & \tilde{A}_{33} \end{bmatrix}$$

$$= \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & L_{22}^{-1}\tilde{A}_{23} \\ 0 & \tilde{A}_{33} - L_{32} \cdot (L_{22}^{-1}\tilde{A}_{23}) \end{bmatrix}$$

$$\equiv \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & U_{23} \\ 0 & \tilde{A}_{33} - L_{32} \cdot U_{23} \end{bmatrix}$$

$$\equiv \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & U_{23} \\ 0 & \tilde{\tilde{A}}_{33} \end{bmatrix}.$$

В результате получено модифицированное разложение, где в L вычислены дополнительные b столбцов, а в U — дополнительные b строк:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{23} & I \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & \tilde{\tilde{A}}_{33} \end{bmatrix}.$$

Все это определяет алгоритм, состоящий из трех шагов и иллюстрируемый правой частью рис. 2.6:

(1) Посредством алгоритма 2.9 найти разложение $\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix} = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} \cdot U_{22}$.

(2) Вычислить матрицу $U_{23} = L_{22}^{-1}\tilde{A}_{23}$. Это равносильно решению треугольной системы линейных уравнений со многими правыми частями ($\tilde{A}_{23}$), что является одной операцией уровня 3.

(3) Вычислить матрицу $\tilde{\tilde{A}}_{33} = \tilde{A}_{33} - L_{32}U_{23}$, используя операцию матричного умножения.

Более формальная запись алгоритма имеет следующий вид:

Алгоритм 2.10. *Реализация LU-разложения без выбора главных элементов с помощью уровня 3 пакета BLAS. Матрица A имеет размер $n \times n$; на ее место записываются L и U. Строки алгоритма нумеруются, как и в предыдущем его описании, и соответствуют частям правого рис. 2.6.*

for $i = 1$ *to* $n - 1$ *step* b

(1) *С помощью алгоритма 2.9 найти разложение*

$$A(i:n, i:i+b-1) = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} U_{22}$$

(2) $A(i:i+b-1, i+b:n) = L_{22}^{-1} \cdot A(i:i+b-1, i+b:n)$

/* *вычислить матрицу* U_{23}^{*}/

(3) $A(i+b:n, i+b:n) = A(i+b:n, i+b:n)$

$-A(i+b:n, i:i+b-1) \cdot A(i:i+b-1, i+b:n)$

/* *вычислить матрицу* $\tilde{\tilde{A}}_{33}$ */

end for

Чтобы получить максимальную производительность алгоритма, нужно еще выбрать размер блока b. С одной стороны, хотелось бы взять как можно большее b, поскольку, как мы видели, скорость перемножения матриц увеличивается с ростом их размеров. С другой стороны, можно проверить, что число операций с плавающей точкой, выполняемых в строке (1) алгоритма более медленными операциями уровня 2 и 1, составляет для малых b примерно $n^2b/2$; это число растет с ростом b, поэтому не стоит брать слишком большое значение для b. Оптимальное значение b зависит от машины и может быть найдено для каждого конкретного типа компьютеров. Часто используют значения $b = 32$ и $b = 64$.

Тщательными реализациями алгоритмов 2.9 и 2.10 являются LAPACK-подпрограммы `sgetf2` и `sgetrf` (см. NETLIB/lapack). Дальнейшую информацию о блочных алгоритмах, включая подробные сведения о производительности на ряде различных компьютеров, можно найти в [10] (см. также конспекты лекций на PARALLEL_HOMEPAGE).

2.6.4. Еще раз о параллелизме и других проблемах производительности

В этом разделе мы дадим краткий обзор прочих проблем, связанных с максимально эффективной реализацией гауссова исключения (и других алгоритмов линейной алгебры).

Параллельный компьютер имеет $p(> 1)$ процессоров, которые можно единовременно использовать для решения одной и той же задачи. Можно надеяться решить любую задачу на таком компьютере в p раз быстрее, чем на обычной однопроцессорной машине. Однако подобная «идеальная эффективность» достигается редко даже в том случае, когда в любой момент имеется p независимых подзадач; это происходит из-за накладных расходов, связанных с координацией процессов и рассылкой данных из хранящих их процессоров тем, которые в этих данных нуждаются. Это последнее обстоятельство есть еще один пример *иерархически организованной* памяти: с точки зрения процессора i, его собственная память быстра; однако получение данных из памяти другого процессора j может иногда происходить медленнее в тысячи раз.

Гауссово исключение содержит в себе много возможностей для параллелизации, так как на каждом шаге элементы оставшейся подматрицы могут перевычисляться параллельно и независимо друг от друга. Однако нужна некоторая осмотрительность, чтобы достигнуть высокой производительности. В этой области имеется два стандартных программных продукта. Описанная в предыдущем разделе LAPACK-программа `sgetrf` (см. [10]) предназначена для *параллельных машин с разделяемой памятью* в предположении, что они оснащены параллельными реализациями пакета BLAS. Родственная библиотека, называемая ScaLAPACK (от *Scalable LAPACK* [34, 53]), была разработана для *параллельных машин с распределенной памятью*, т. е. машин, требующих специальных операций для обмена данными между различными процессорами. Все программы доступны в NETLIB'е в поддиректориях LAPACK и ScaLAPACK. Более подробно ScaLAPACK описан в конспектах лекций, помещенных на PARALLEL_HOMEPAGE. Обширные данные о производительности программ решения линейных систем содержатся в техническом отчете

LINPACK Benchmark [85], а более современную версию обзора производительности можно найти по адресу NETLIB/benchmark/performance.ps; см. также Performance Database Server[1]. На май 1997 г. наибольшая скорость решения линейной системы посредством гауссова исключения была достигнута для задачи размерности $n = 215\,000$ на машине Intel ASCI Option Red с $p = 7264$ процессорами. Задача решалась со скоростью, превышающей 1068 гигафлопов при пиковой производительности компьютера 1453 гигафлопа.

Встречаются матрицы настолько большие, что они не могут поместиться в основной памяти никакого из имеющихся компьютеров. Такие матрицы хранят на диске и в ходе гауссова исключения считывают в основную память частями. Конструкция соответствующих программ весьма схожа с описанными выше алгоритмами; программы этого типа тоже включены в ScaLAPACK.

Можно надеяться, что трансляторы станут в конце концов настолько умными, что, исходя из простейшей реализации гауссова исключения посредством трех вложенных циклов, будут способны автоматически «оптимизировать» программу до уровня блочного алгоритма, обсуждавшегося в предыдущем разделе. Хотя в этом направлении в настоящее время ведется большая работа (см. библиографию в недавно изданном учебнике по трансляторам [264]), все еще не существует гарантированно быстрой альтернативы использованию оптимизированных библиотек типа LAPACK и ScaLAPACK.

2.7. Специальные линейные системы

Как уже говорилось в разделе 1.2, важно уметь использовать любую специальную структуру матрицы, чтобы ускорить решение задачи и уменьшить расход памяти. Разумеется, на практике нужно принимать в учет стоимость дополнительных программистских усилий, затрачиваемых на эксплуатацию этой структуры. Пусть, например, нашей единственной целью является минимизация времени вычисления решения, и пусть нужна неделя дополнительной программистской работы, чтобы уменьшить время решения задачи с 10 секунд до одной. Это имеет смысл делать лишь в том случае, если мы собираемся пользоваться полученной программой более чем (1 неделя * 7 дней/в неделе * 24 час/в день * 3600 сек/в часе) /(10 сек " — 1 сек) = 67200 раз. К счастью, существуют достаточно часто встречающиеся специальные структуры, для которых имеются стандартные методы решения, и мы, конечно, должны использовать эти методы. Мы рассмотрим здесь следующие типы специальных матриц:

1. симметричные положительно определенные матрицы (с.п.о. матрицы);
2. симметричные незнакоопределенные матрицы;
3. ленточные матрицы;
4. разреженные матрицы общего вида;
5. плотные матрицы, зависящие от менее чем n^2 независимых параметров.

Будут рассматриваться лишь вещественные матрицы; обобщения методов на случай комплексных матриц очевидны.

[1] http://performance.netlib.org/performance/html/PDStop.html

2.7.1. Вещественные симметричные положительные определенные матрицы

Напомним, что вещественная матрица A называется с.п.о матрицей, если $A = A^T$ и $x^T Ax > 0$ для всех $x \neq 0$. В этом разделе мы покажем, как решить систему $Ax = b$ с с.п.о. матрицей A за половину времени и с использованием вдвое меньшей памяти по сравнению с гауссовым исключением.

Предложение 2.2. 1. *Если X — невырожденная матрица, то A тогда и только тогда является с.п.о. матрицей, когда $X^T AX$ есть с.п.о. матрица.*

2 *Если A — с.п.о. матрица, а H — произвольная главная подматрица в A (т. е. $H = A(j : k, j : k)$ для некоторого $j \leq k$), то H также с.п.о. матрица.*

3 *A тогда и только тогда является с.п.о. матрицей, когда $A = A^T$ и все собственные значения матрицы A положительны.*

4 *Если A — с.п.о. матрица, то все $a_{ii} > 0$ и $\max_{ij} |a_{ij}| = \max_i a_{ii} > 0$.*

5 *A тогда и только тогда является с.п.о. матрицей, когда существует единственная невырожденная нижетреугольная матрица L с положительными диагональными элементами, такая, что $A = LL^T$. Представление $A = LL^T$ называется разложением Холесского, а L — множителем Холесского матрицы A.*

Доказательство.

1. Невырожденность матрицы X означает, что $Xx \neq 0$ для всех $x \neq 0$, поэтому $x^T X^T AXx > 0$, если $x \neq 0$. Таким образом, если A — с.п.о. матрица, то и $X^T Ax$ — с.п.о. матрица. Обратная импликация доказывается с помощью матрицы X^{-1}.
2. Предположим вначале, что $H = A(1 : m, 1 : m)$. Тогда для любого m-вектора y вектор $x = [y^T, 0]$ размерности n удовлетворяет соотношению $y^T Hy = x^T Ax$. Поэтому, если $x^T Ax > 0$ для любого ненулевого вектора x, то $y^T Hy > 0$ для любого ненулевого вектора y; следовательно, H есть с.п.о. матрица. Если H не является ведущей главной подматрицей, то пусть P — перестановка, переводящая H в левый верхний угол матрицы $P^T AP$; к последней применяем утверждение 1.
3. Пусть X — вещественная ортогональная матрица, составленная из собственных векторов матрицы A, тогда $X^T AX = \Lambda$ есть диагональная матрица, содержащая на диагонали (вещественные) собственные значения λ_i. Поскольку $x^T \Lambda x = \sum_i \lambda_i x_i^2$, то Λ тогда и только тогда является с.п.о. матрицей, когда все $\lambda_i > 0$. Остается применить утверждение 1.
4. Пусть e_i — столбец с номером i единичной матрицы. Тогда $e_i^T Ae_i = a_{ii} > 0$ для всех i. Если $|a_{kl}| = \max_{i,j} |a_{ij}|$, но $k \neq l$, положим $x = e_k - \operatorname{sgn}(a_{kl}) e_l$. Тогда $x^T Ax = a_{kk} + a_{ll} - 2|a_{kl}| \leq 0$, что противоречит положительной определенности матрицы A.
5. Предположим, что $A = LL^T$, где L — невырожденная матрица. Тогда $x^T Ax = (x^T L)(L^T x) = \|L^T x\|_2^2 > 0$ для всех $x \neq 0$, поэтому A — с.п.о. матрица. Если A — с.п.о. матрица, то мы докажем существование (треугольной) матрицы L индукцией по порядку n. Если в нашем построении выбирать

положительное значение для каждого l_{ii}, то оно определит L единственным образом. При $n = 1$ полагаем $l_{11} = \sqrt{a_{11}}$, что возможно, поскольку $a_{11} > 0$. Как и в случае гауссова исключения, достаточно разобраться с блочными 2×2-матрицами. Имеем

$$\begin{aligned} A &= \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{A_{12}^T}{\sqrt{a_{11}}} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & \frac{A_{12}}{\sqrt{a_{11}}} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^T & \tilde{A}_{22} + \frac{A_{12}^T A_{12}}{a_{11}} \end{bmatrix}, \end{aligned}$$

таким образом, $(n-1) \times (n-1)$ матрица $\tilde{A}_{22} = A_{22} - \frac{A_{12}^T A_{12}}{a_{11}}$ симметрична. Согласно утверждению 1, $\begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix}$ есть с.п.о. матрица, но тогда, согласно утверждению 2, и $\tilde{A}_{22}$ есть с.п.о. матрица. По предположению индукции, существует (треугольная) матрица $\tilde{L}$, такая, что $\tilde{A}_{22} = \tilde{L}\tilde{L}^T$ и

$$\begin{aligned} A &= \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{A_{12}^T}{\sqrt{a_{11}}} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L}\tilde{L}^T \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & \frac{A_{12}}{\sqrt{a_{11}}} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{A_{12}^T}{\sqrt{a_{11}}} & \tilde{L} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & \frac{A_{12}}{\sqrt{a_{11}}} \\ 0 & \tilde{L}^T \end{bmatrix} \equiv LL^T. \end{aligned}$$

□

Мы можем оформить это индуктивное рассуждение в виде следующего алгоритма.

Алгоритм 2.11. *Алгоритм Холесского:*

for $j = 1$ *to* n
  $l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{1/2}$
  for $i = j+1$ *to* n
    $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}$
  end for
end for

Если A не является положительно определенной матрицей, то (в точной арифметике) алгоритм прекратит работу досрочно при попытке извлечь квадратный корень из отрицательного числа либо разделить на нуль; это наблюдение указывает самый экономичный способ проверки свойства положительной определенности симметричной матрицы.

Как и в случае гауссова исключения, L может быть записана на место нижней половины матрицы A. В алгоритме происходят обращения только к элементам из нижней половины A, поэтому, в действительности, достаточно памяти $n(n+1)/2$ вместо n^2. Число операций с плавающей точкой равно

$$\sum_{j=1}^{n} \left(2j + \sum_{i=j+1}^{n} 2j \right) = \frac{1}{3} n^3 + O(n^2),$$

или примерно половина того, что выполняется в гауссовом исключении. Подобно последнему, алгоритм Холесского может быть модифицирован так, что бóльшая часть операций с плавающей точкой будет в нем производиться с помощью BLAS-подпрограмм уровня 3 (см. LAPACK-программу `spotrf`).

Для численной устойчивости алгоритма Холесского выбор главных элементов не является необходимым (по-другому, мы могли бы сказать, что алгоритм остается численно устойчивым при любом способе выбора главных элементов). Это можно показать следующим образом. Анализ, аналогичный анализу гауссова исключения в разд. 2.4.2, свидетельствует, что вычисленное решение $\hat{x}$ удовлетворяет системе $(A+\delta A)\hat{x}=b$, где $|\delta A| \le 3n\varepsilon|L|\cdot|L^T|$. Согласно неравенству Коши–Шварца и утверждению 4 из предложения 2.2,

$$\begin{aligned}(|L|\cdot|L^T|)_{ij} &= \sum_k |l_{ik}|\cdot|l_{jk}| \le \sqrt{\sum l_{ik}^2}\sqrt{\sum l_{jk}^2} \\ &= \sqrt{a_{ii}}\cdot\sqrt{a_{jj}} \le \max_{ij}|a_{ij}|, \end{aligned} \tag{2.16}$$

поэтому $\||L|\cdot|L^T|\|_\infty \le n\|A\|_\infty$ и $\|\delta A\|_\infty \le 3n^2\varepsilon\|A\|_\infty$.

2.7.2. Симметричные незнакоопределенные матрицы

Естественным образом возникает вопрос, можно ли сэкономить половину времени и памяти при решении системы линейных уравнений с симметричной, но незнакоопределенной матрицей (т. е. матрицей, не являющейся ни положительно определенной, ни отрицательно определенной). Оказывается, что это возможно, однако требуются более сложные разложение и схема выбора главных элементов. Можно показать, что для невырожденной матрицы A найдутся перестановка P, нижняя унитреугольная матрица L и блочно-диагональная матрица D с 1×1 и 2×2 диагональными блоками, такие, что $PAP^T = LDL^T$. Чтобы понять, почему в D нужно допустить 2×2-блоки, достаточно рассмотреть матрицу $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Указанное разложение можно вычислить устойчиво, экономя при этом примерно половину работы и памяти по сравнению со стандартным гауссовым исключением. LAPACK-подпрограмма, вычисляющая это разложение, называется `ssysv`. Описание алгоритма дано в [44].

2.7.3. Ленточные матрицы

Говорят, что A — *ленточная матрица с нижней шириной* b_L и *верхней шириной* b_U, если $a_{ij}=0$ при $i>j+b_L$ или $i<j-b_U$:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1,b_U+1} & & 0 \\ \vdots & & & a_{2,b_U+2} & \\ a_{b_L+1,1} & & & & \ddots \\ & a_{b_L+2,2} & & & a_{n-b_U,n} \\ & & \ddots & & \vdots \\ 0 & & & a_{n,n-b_L} & \cdots \quad a_{n,n} \end{bmatrix}.$$

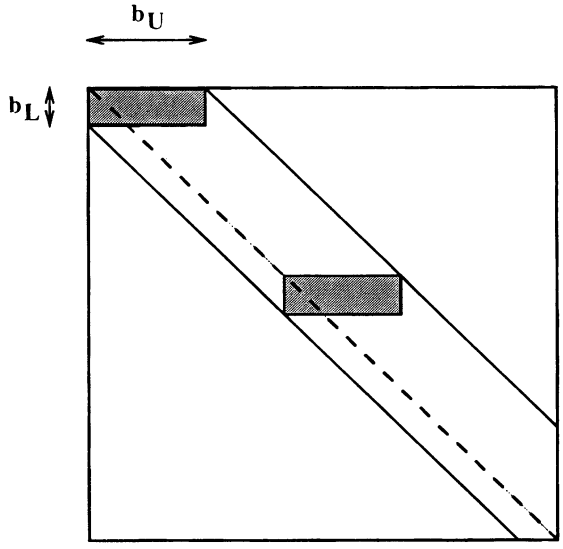


Рис. 2.7. Ленточное LU-разложение без выбора главных элементов.

Ленточные матрицы часто возникают в практических задачах (ниже будет приведен пример); эти матрицы имеет смысл выделить в отдельный класс, поскольку для них треугольные множители L и U также являются «по существу ленточными», вследствие чего их вычисление и хранение удешевляются. Ниже мы объясним, что понимается под термином «по существу ленточные» матрицы. Однако вначале рассмотрим LU-разложение без выбора главных элементов и покажем, что L и U суть ленточные матрицы в обычном смысле, причем они имеют те же ширины, что и A.

Предложение 2.3. *Пусть A — ленточная матрица с нижней шириной b_L и верхней шириной b_U. Пусть разложение $A = LU$ вычисляется без выбора главных элементов. Тогда L имеет нижнюю ширину ленты b_L, а U — верхнюю ширину ленты b_U. Если b_U и b_L малы по сравнению с n, то L и U можно вычислить приблизительно за $2n \cdot b_U \cdot b_L$ арифметических операций, при этом требуется память $n(b_L + b_U + 1)$. Полная стоимость решения системы $Ax = b$ составляет $2nb_U \cdot b_L + 2nb_U + 2nb_L$ операций.*

Набросок доказательства. Достаточно проследить за одним шагом разложения (см. рис. 2.7). На шаге j гауссова исключения закрашенная область перевычисляется путем вычитания произведения первого столбца и первой строки этой области; ширина ленты при этом не увеличивается. □

Предложение 2.4. *Пусть A — ленточная матрица с нижней шириной b_L и верхней шириной b_U. Тогда после применения к A гауссова исключения будут получены ленточная матрица U с верхней шириной ленты, не превосходящей $b_L + b_U$, и «по существу ленточная» матрица L с нижней шириной ленты b_L. Последнее означает, что в каждом столбце матрицы L содержится не более чем $b_L + 1$ ненулевых элементов; поэтому для хранения L достаточно*

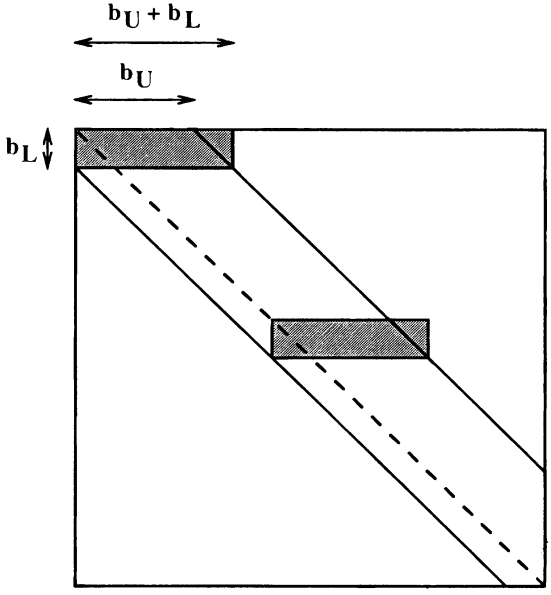


Рис. 2.8. Ленточное LU-разложение с частичным выбором главных элементов.

памяти такого же размера, как и для ленточной матрицы с нижней шириной ленты b_L.

Набросок доказательства. Снова проиллюстрируем доказательство изображением области, изменяемой одним шагом алгоритма. Рис. 2.8 показывает, что (частичный) выбор главных элементов может увеличить верхнюю ширину ленты не более чем на b_L. Последующие перестановки могут переупорядочить элементы ранее вычисленных столбцов, так что в L могут появиться элементы, лежащие ниже поддиагонали с номером b_L. Однако никаких новых ненулевых элементов не появится, поэтому для хранения любого столбца в L по-прежнему будет требоваться b_L слов. □

Гауссово исключение и алгоритм Холесского для ленточных матриц реализованы в LAPACK'е (см., например, программы `ssbsv` и `sspsv`).

Ленточные матрицы часто возникают при дискретизации физических задач с взаимодействиями между ближайшими узлами сетки (предполагается, что неизвестные упорядочены по строкам или по столбцам; см. еще пример 2.9 и разд. 6.3).

Пример 2.8. Рассмотрим на отрезке $[a, b]$ обыкновенное дифференциальное уравнение (ОДУ) $y''(x) - p(x)y'(x) - q(x)y(x) = r(x)$ с граничными условиями $y(a) = \alpha$, $y(b) = \beta$. Предположим еще, что $q(x) \geq \underline{q} > 0$. С помощью данного уравнения можно моделировать, например, распространение тепла в длинном тонком стержне. Чтобы решить дифференциальное уравнение численно, мы *дискретизуем* его, разыскивая решение лишь на сетке с равноотстоящими узлами $x_i = a + ih$, $i = 0, \ldots, N+1$, где $h = (b-a)/(N+1)$ есть шаг сетки. Положим $p_i = p(x_i)$, $r_i = r(x_i)$ и $q_i = q(x_i)$. Нужно вывести уравнения, опре-

деляющие желаемые приближения $y_i \approx y(x_i)$, где $y_0 = \alpha$ и $y_{N+1} = \beta$. Для вывода уравнений заменим значение производной $y'(x_i)$ следующим *конечно-разностным приближением*:

$$y'(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2h}.$$

(Заметьте, что по мере уменьшения h правая часть приближает $y'(x_i)$ все более и более точно.) Аналогичным образом можно аппроксимировать вторую производную:

$$y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}.$$

(Более подробный вывод этого приближения дан в разд. 6.3.1 гл. 6.)

Подставляя эти приближения в дифференциальное уравнение, получаем

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - p_i \frac{y_{i+1} - y_{i-1}}{2h} - q_i y_i = r_i, \quad 1 \le i \le N.$$

Запишем эти соотношения как систему линейных уравнений $Ay = b$, где

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \; b = \frac{-h^2}{2} \begin{bmatrix} r_1 \\ \vdots \\ r_N \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{2} + \frac{h}{4}p_1\right)\alpha \\ 0 \\ \vdots \\ 0 \\ \left(\frac{1}{2} - \frac{h}{4}p_N\right)\beta \end{bmatrix},$$

$$A = \begin{bmatrix} a_1 & -c_1 & & \\ -b_2 & \ddots & \ddots & \\ & \ddots & \ddots & c_{N-1} \\ & & -b_N & a_N \end{bmatrix}, \quad \begin{aligned} a_i &= 1 + \tfrac{h^2}{2} q_i, \\ b_i &= \tfrac{1}{2}\left[1 + \tfrac{h}{2}p_i\right], \\ c_i &= \tfrac{1}{2}\left[1 - \tfrac{h}{2}p_i\right]. \end{aligned}$$

Заметим, что $a_i > 0$, а при достаточно малых h и $b_i > 0$, $c_i > 0$.

Для вектора y имеем несимметричную *трехдиагональную* систему уравнений. Покажем, как преобразовать ее в трехдиагональную же, но симметричную положительно определенную систему, для решения которой можно применить *ленточный алгоритм Холесского*.

Положим $D = \operatorname{diag}\left(1, \sqrt{\frac{c_1}{b_2}}, \sqrt{\frac{c_1 c_2}{b_2 b_3}}, \ldots, \sqrt{\frac{c_1 c_2 \cdots c_{N-1}}{b_2 b_3 \cdots b_N}}\right)$. Тогда от системы $Ay = b$ можно перейти к $(DAD^{-1})(Dy) = Db$, или $\tilde{A}\tilde{y} = \tilde{b}$, где

$$\tilde{A} = \begin{bmatrix} a_1 & -\sqrt{c_1 b_2} & & \\ -\sqrt{c_1 b_2} & a_2 & -\sqrt{c_2 b_3} & \\ & -\sqrt{c_2 b_2} & \ddots & \\ & \ddots & \ddots & -\sqrt{c_{N-1} b_N} \\ & & -\sqrt{c_{N-1} b_N} & a_N \end{bmatrix}.$$

Легко видеть, что $\tilde{A}$ — симметричная матрица, имеющая те же собственные значения, что и A, поскольку A и $\tilde{A} = DAD^{-1}$ *подобны*. (Подробности см.

в разд. 4.2 гл. 4.) Чтобы показать, что $\tilde{A}$ еще и положительно определена, воспользуемся следующей теоремой.

Теорема 2.9 *(Гершгорин). Пусть B — произвольная матрица. Тогда все ее собственные значения принадлежат объединению n кругов*

$$|\lambda - b_{kk}| \leq \sum_{j \neq k} |b_{kj}|.$$

Доказательство. Пусть имеем λ и $x \neq 0$, такие, что $Bx = \lambda x$. Масштабируя x, если это необходимо, будем считать, что $1 = \|x\|_\infty = x_k$. Тогда $\sum_{j=1}^{N} b_{kj}x_j = \lambda x_k = \lambda$, поэтому $\lambda - b_{kk} = \sum_{j=1, j\neq k}^{N} b_{kj}x_j$, откуда выводим

$$|\lambda - b_{kk}| \leq \sum_{j \neq k} |b_{kj}x_j| \leq \sum_{j \neq k} |b_{kj}|. \qquad \square$$

Если h настолько мало, что $\left|\frac{h}{2}p_i\right| < 1$ для всех i, то

$$|b_i| + |c_i| = \frac{1}{2}\left(1 + \frac{h}{2}p_i\right) + \frac{1}{2}\left(1 - \frac{h}{2}p_i\right) = 1 < 1 + \frac{h^2}{2}\underline{q} \leq 1 + \frac{h^2}{2}q_i = a_i.$$

Следовательно, все собственные значения матрицы A находятся в кругах радиуса 1 с центрами в точках $1 + h^2 q_i/2 \geq 1 + h^2\underline{q}/2$; в частности, все собственные значения имеют положительные вещественные части. Поскольку $\tilde{A}$ — симметричная матрица, ее собственные значения вещественны, а потому положительны; таким образом, $\tilde{A}$ положительно определена. Ее наименьшее собственное значение ограничено снизу числом $\underline{q}h^2/2$. Итак, система с матрицей $\tilde{A}$ может быть решена алгоритмом Холесского. Для решения симметричных положительно определенных трехдиагональных систем в LAPACK'е имеется подпрограмма `sptsv`.

В разд. 4.3 мы снова воспользуемся теоремой Гершгорина, чтобы вычислить границы возмущений для собственных значений матрицы. $\diamond$

2.7.4. Разреженные матрицы общего вида

Под разреженной матрицей понимают матрицу, содержащую большое число нулевых элементов. С практической точки зрения, это число должно быть настолько большим, чтобы имело смысл применение алгоритма, избегающего хранения нулевых элементов и оперирования с ними. В главе 6 обсуждаются методы решения разреженных линейных систем, отличающиеся от гауссова исключения и его вариантов. Существует огромное количество методов для разреженных систем и выбор наилучшего из них часто требует значительной информации о матрице системы [24]. В этом разделе мы лишь кратко обрисуем проблематику разреженного гауссова исключения и дадим ссылки на соответствующую литературу и имеющееся программное обеспечение.

В качестве очень простого примера рассмотрим следующую матрицу:

$$A = \begin{bmatrix} 1 & & & & .1 \\ & 1 & & & .1 \\ & & 1 & & .1 \\ & & & 1 & .1 \\ .1 & .1 & .1 & .1 & 1 \end{bmatrix} = LU$$

$$= \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ .1 & .1 & .1 & .1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & .1 \\ & 1 & & & .1 \\ & & 1 & & .1 \\ & & & 1 & .1 \\ & & & & .96 \end{bmatrix}.$$

Она упорядочена таким образом, что при применении к ней гауссова исключения с частичным выбором главного элемента никаких перестановок строк не происходит, и называется (учитывая характер расположения ее ненулевых элементов) *стреловидной матрицей*. Отметим, что ни один из нулевых элементов в A не был *заполнен* при проведении гауссова исключения, так что L и U совместно могут быть сохранены на месте, прежде занимавшемся ненулевыми элементами исходной матрицы. Кроме того, если подсчитывать только существенные арифметические операции (исключая умножение на нуль и сложение с нулем), то их будет лишь 12 (4 деления при вычислении последней строки в L и 8 умножений и сложений при пересчете элемента (5, 5)) вместо $\frac{2}{3}n^3 \approx 83$ операций. Более общо, для стреловидной матрицы A порядка n хранение требует лишь $3n-2$ машинных слова вместо n^2, а применение гауссова исключения обходится в $3n-3$ операции с плавающей точкой вместо $\frac{2}{3}n^3$. Когда n велико, эти значения для памяти и числа операций становятся очень малыми по сравнению со случаем плотной матрицы.

Предположим теперь, что вместо A задана матрица A', получающаяся из A записью строк и столбцов в обратном порядке. Это равносильно обращению порядка уравнений и неизвестных в линейной системе $Ax = b$. При применении к A' метода GEPP снова не происходит перестановок строк и с точностью до двух десятичных разрядов будет найдено разложение

$$A' = \begin{bmatrix} 1 & .1 & .1 & .1 & .1 \\ .1 & 1 & & & \\ .1 & & 1 & & \\ .1 & & & 1 & \\ .1 & & & & 1 \end{bmatrix} = L'U'$$

$$= \begin{bmatrix} 1 & & & & \\ .1 & 1 & & & \\ .1 & -.01 & 1 & & \\ .1 & -.01 & -.01 & 1 & \\ .1 & -.01 & -.01 & -.01 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & .1 & .1 & .1 & .1 \\ & .99 & -.01 & -.01 & -.01 \\ & & .99 & -.01 & -.01 \\ & & & .99 & -.01 \\ & & & & .99 \end{bmatrix}.$$

Как видим, теперь L' и U' совершенно заполнены и их хранение требует n^2 слов. В действительности, все нулевые элементы в A' были заполнены уже после первого шага алгоритма, поэтому приходится выполнять такую же работу, как и для плотного гауссова исключения, т. е. $\frac{2}{3}n^3$ операций.

Данный пример свидетельствует, что порядок строк и столбцов крайне важен для экономии памяти и работы. Выбор оптимальных перестановок для строк и столбцов, минимизирующих память или работу, является чрезвычайно трудной задачей даже в том случае, если не нужно заботиться о выборе главных элементов для обеспечения численной устойчивости (как обстоит дело в алгоритме Холесского). На самом деле, эта задача NP-полна [111], что означает: все известные алгоритмы для отыскания оптимальной перестановки требуют времени, *экспоненциально* растущего вместе с n; таким образом, для больших n эти алгоритмы намного менее эффективны, чем даже плотное гауссово исключение. Поэтому нам придется обратиться к эвристическим подходам, среди которых есть несколько удачных. Некоторые из них будут проиллюстрированы ниже.

Помимо трудности выбора хороших перестановок для строк и столбцов, имеются и другие причины, почему разреженные реализации гауссова исключения или алгоритма Холесского намного более сложны, чем соответствующие методы для плотных матриц. Во-первых, нужно организовать структуру данных для хранения только ненулевых элементов матрицы A; существует несколько таких общеупотребительных структур [93]. Далее, нужна структура данных для хранения новых ненулевых элементов, возникающих в L и U в ходе исключения. Это означает, что либо структура данных должна динамически изменяться в алгоритме, либо мы должны найти дешевый способ ее предварительного вычисления без реального проведения исключения. Наконец, структура данных должна быть использована для минимизации числа операций с плавающей точкой, а также выполнения целочисленных и логических операций в количестве, самое большее пропорциональном числу флопов. Иными словами, мы не можем позволить себе тратить $O(n^3)$ целочисленных и логических операций для отыскания тех немногих операций с плавающей точкой, которые мы согласны выполнять. Более полное описание этих вопросов выходит за рамки данной книги (см. [114, 93]), однако мы дадим справку об имеющемся программном обеспечении.

Пример 2.9. Проиллюстрируем разреженный алгоритм Холесского на реалистическом примере, возникающем при моделировании задачи о смещении механической структуры под воздействием внешних сил. На рис. 2.9 показана простая сетка, нанесенная на двумерное сечение механической структуры с двумя внутренними полостями. Математически задача состоит в том, чтобы определить смещения всех узлов сетки (внутренних по отношению к структуре) под влиянием некоторых сил, приложенных к границе структуры. Узлы сетки пронумерованы числами от 1 до $n = 483$; в практических приложениях значения n были бы гораздо бóльшими. Уравнения, связывающие смещения и силы, приводят к системе линейных уравнений $Ax = b$, где каждому из 483 узлов соответствуют одна строка и один столбец, причем $a_{ij} \neq 0$ в том и только том случае, если узел i связан отрезком прямой с узлом j. Отсюда можно вывести, что A — симметричная матрица; она оказывается также положительно определенной, поэтому для решения системы $Ax = b$ можно использовать алгоритм Холесского. Заметим, что A имеет лишь $nz = 3971$ ненулевых элементов из возможного их числа $483^2 = 233289$; таким образом, A заполнена лишь на $3971/233289 = 1.7\%$. (Сходные задачи моделирования механических структур рассматриваются в примерах 4.1 и 5.1, где приведен подробный вывод матриц A.)

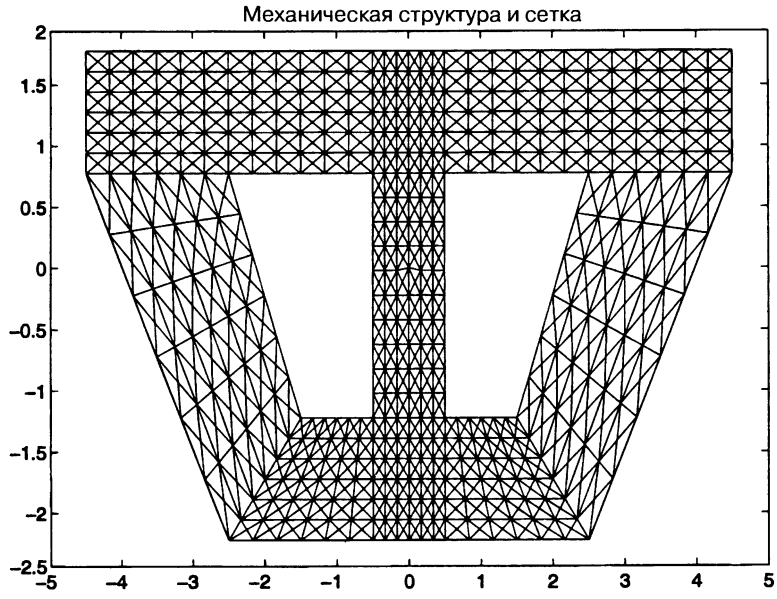


Рис. 2.9. Сетка для механической структуры.

В верхней части рис. 2.10 показана та же сетка, что и на рис. 2.9, а в нижней части этого рисунка — расположение ненулевых элементов в матрице A. При этом 483 узла сетки пронумерованы в некотором «естественном» порядке: узлы логически прямоугольных подструктур нумеруются построчно и одна подструктура нумеруется вслед за другой. Ребра каждой подструктуры окрашены одним и тем же цветом[1]; этим цветам соответствует окраска ненулевых элементов матрицы. Каждая подструктура имеет метку «$(i : j)$», указывающую, что ей соответствуют в A строки и столбцы с номерами от i до j. При этом подматрица $A(i : j, i : j)$ является ленточной матрицей с узкой лентой. (В примере 2.8 и разд. 6.3 описаны другие ситуации, где сетка порождает ленточную матрицу.) Ребра, связывающие различные подструктуры, окрашены красным цветом и соответствуют красным элементам матрицы A, наиболее удаленным от ее главной диагонали.

На верхней паре картинок рис. 2.11 снова показана структура разреженности матрицы A при естественном порядке узлов, а также структура разреженности ее множителя Холесского L. Ненулевые элементы в L, отвечающие ненулевым элементам в A, окрашены черным цветом; новые ненулевые элементы в L, составляющие *заполнение*, окрашены красным цветом. Всего в L 11533 ненулевых элемента, что более чем в пять раз превышает число ненулевых элементов в нижнем треугольнике матрицы A. На вычисление L алгоритмом Холесского затрачивается лишь 296923 флопа, что составляет всего 0.8% от $\frac{1}{3}n^3 = 3.76 \cdot 10^7$ флопов, которых потребовал бы алгоритм для плотных матриц.

[1] Цветной вариант рис. 2.10. см. на обложке книги. — *Прим. перев.*

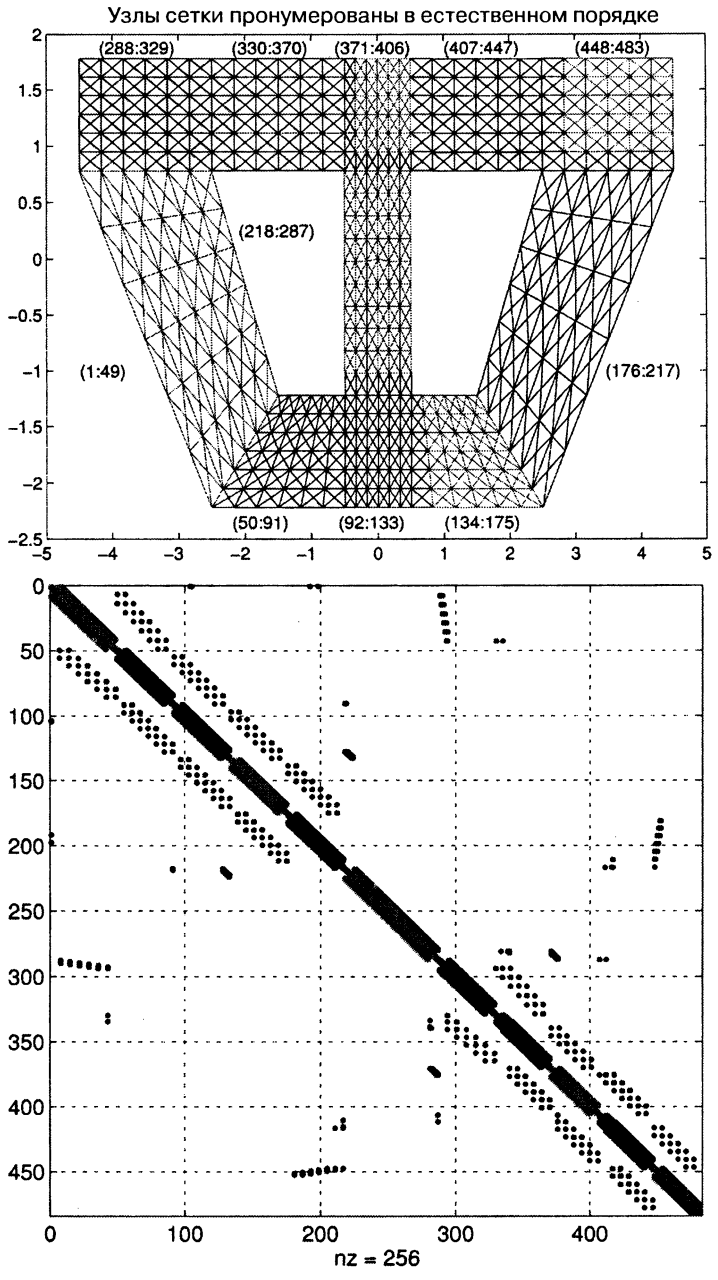


Рис. 2.10. Ребра сетки в верхней части рисунка окрашены и пронумерованы; они находятся в соответствии с ненулевыми элементами разреженной матрицы A в нижней части рисунка. Например, первые 49 узлов сетки (самые левые узлы, окрашенные зеленым цветом) соответствуют строкам и столбцам матрицы A с номерами от 1 до 49.

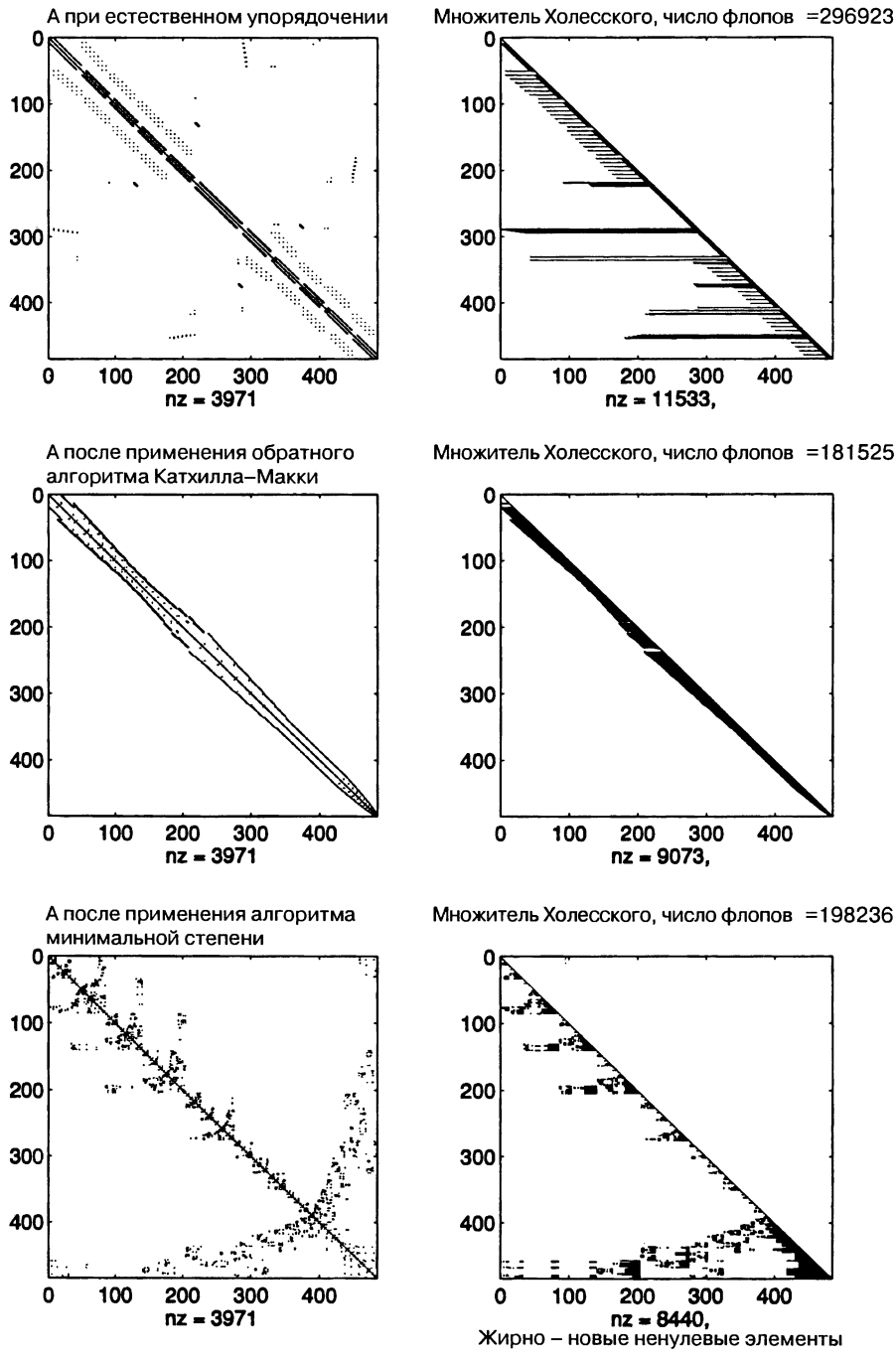


Рис. 2.11. Разреженность и число флопов при различных упорядочениях матрицы A.

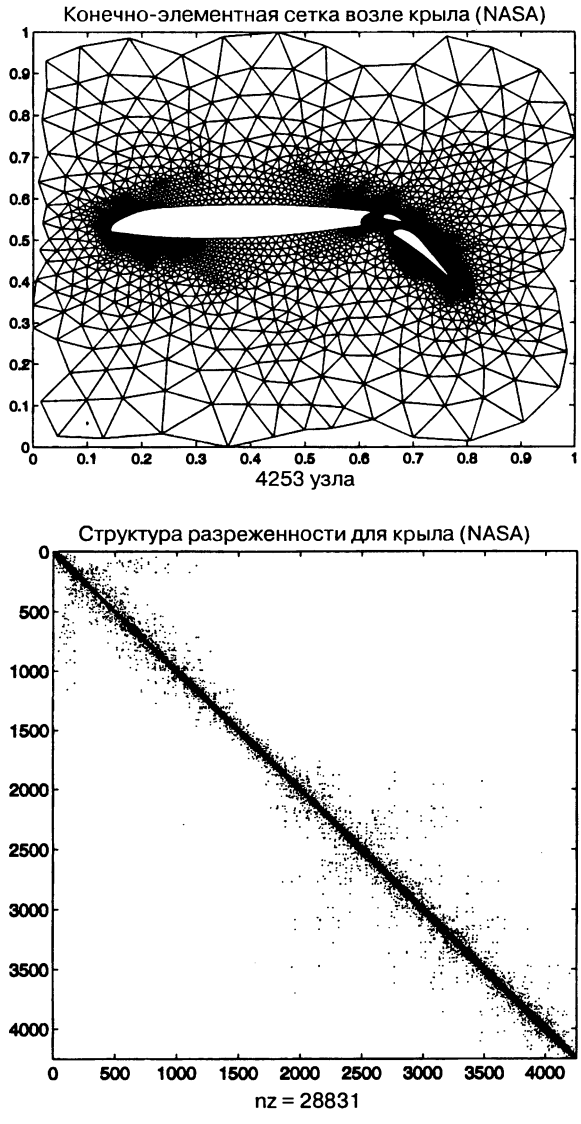


Рис. 2.12. Сетка возле крыла (NASA).

Число ненулевых элементов матрицы L и количество флопов при ее вычислении можно существенно изменить, переупорядочивая строки и столбцы в A. На средней паре картинок рис. 2.11 показаны результаты, полученные с помощью одной популярной схемы переупорядочения, называемой *обратным алгоритмом Катхилл—Макки* [114, 93]; ее назначение — попытаться превратить A в ленточную матрицу с узкой лентой. Как видим, в данном случае это

вполне удалось: заполнение в L уменьшено на 21% (с 11533 элементов до 9073), а число флопов — почти на 39% (с 296923 до 181525).

Другая популярная схема упорядочения называется *алгоритмом минимальной степени* [114, 93]; ее принцип — минимизировать заполнение на каждом шаге алгоритма Холесского. Результаты, полученные посредством этой схемы, показаны на нижней паре картинок рис. 2.11: заполнение в L снижено еще на 7% (с 9073 элементов до 8440), однако число флопов увеличилось на 9% (с 181525 до 198236). ◇

В системе Matlab имеется ряд процедур для обработки разреженных матриц (их список можно получить в Matlab'е с помощью команды «help sparfun») и ряд примеров конкретных разреженных матриц в виде встроенных демосов. Вызвать такой пример на экран можно, вводя команду demo, затем щелкая мышью на пункте «continue», потом на «Matlab/Visit» и, наконец, на «Matrices/Select a demo/Sparse» либо «Matrices/Select a demo/Cmd line demos». Так, на рис. 2.12 воспроизведен пример из Matlab'а — сетка возле крыла. Задача состоит в том, чтобы рассчитать в узлах сетки поток воздуха возле крыла. Дифференциальные уравнения с частными производными, описывающие поток, приводят (при дискретизации) к несимметричной системе линейных уравнений, структура которой также показана на рис. 2.12.

Программное обеспечение для разреженных матриц

Помимо Matlab'а имеется большое количество распространяемых бесплатно либо коммерческих программ для разреженных матриц, написанных на фортране или C. В этой области все еще ведется активная исследовательская работа (особенно в том, что касается высокопроизводительных машин), поэтому невозможно рекомендовать какой-либо алгоритм как наилучший. Таблица 2.2 [177] дает список имеющихся программ, систематизированный в нескольких отношениях. Мы ограничиваемся программами (общедоступными или коммерческими), пользующимися поддержкой, либо экспериментальными программами в тех случаях, когда никаких других для данного типа задач или компьютеров не существует. Более полные списки и объяснения алгоритмов, упоминаемых ниже, можно найти в [177, 94].

Таблица 2.2 устроена следующим образом. Верхняя группа программ, называемая «*последовательные алгоритмы*», предназначена для однопроцессорных рабочих станций и персональных компьютеров. «*Алгоритмы с разделяемой памятью*» разработаны для таких симметричных мультипроцессорных машин, как Sun SPARCcenter 2000 [238], SGI Power Challenge [223], DEC AlphaServer 8400 [103] и Cray C90/J90 [253, 254]. «*Алгоритмы с распределенной памятью*» создавались для таких машин, как IBM SP-2 [256], Intel Paragon [257], компьютеры серии Cray T3 [255] и сети рабочих станций. Как видно из таблицы, бóльшая часть программ была написана для последовательных машин, некоторое количество для машин с разделяемой памятью и лишь совсем немногое (если не говорить об исследовательских программах) для машин с распределенной памятью.

В первом столбце таблицы указан *тип матрицы*. Возможные варианты: несимметричные матрицы, матрицы симметричной структуры (т. е. такие, что $a_{ij} = a_{ji} = 0$ либо, будучи неравны, оба элемента отличны от нуля), симме-

Тип матрицы	Название	Алгоритм	Статус/ источник
Последовательные алгоритмы			
несим.	SuperLU	LL, частичный, BLAS-2.5	Pub/NETLIB
несим.	UMFPACK [62, 63]	MF, Марковиц, BLAS-3	Pub/NETLIB
	MA38 (то же самое, что и UMFPACK)		
несим.	MA48 [96]	Анализ: RL, Марковиц Факторизация: LL, частичный, BLAS-1, SD	Com/HSL
несим.	SPARSE [167]	RL, Марковиц, скалярный	Pub/NETLIB
сим. стр.	MUPS [5]	MF, барьерный, BLAS-3	Com/HSL
	MA42 [98]	Фронтальный, BLAS-3	Com/HSL
сим.	MA27 [97]/MA47 [95]	MF, LDL^T, BLAS-1/BLAS-3	Com/HSL
с.п.о.	Ng & Peyton [191]	LL, BLAS-3	Pub/Автор
Алгоритмы с разделяемой памятью			
несим.	SuperLU	LL, частичный, BLAS-2.5	Pub/UCB
несим.	PARASPAR [270, 271]	RL, Марковиц, BLAS-1, SD	Res/Автор
сим. стр.	MUPS [6]	MF, барьерный, BLAS-3	Res/Автор
несим.	George & Ng [115]	RL, частичный, BLAS-1	Res/Автор
с.п.о.	Gupta et al. [133]	LL, BLAS-3	Com/SGI
с.п.о.	SPLASH [155]	RL, 2-D блочный, BLAS-3	Pub/Stanford
Алгоритмы с распределенной памятью			
сим.	van der Stappen [245]	RL, Марковиц, скалярный	Res/Автор
сим. стр.	Lucas et al. [180]	MF, нет выбора главных элементов, BLAS-1	Res/Автор
с.п.о.	Rothberg & Schreiber [207]	RL, 2-D блочный, BLAS-3	Res/Автор
с.п.о.	Gupta & Kumar [132]	MF, 2-D блочный, BLAS-3	Res/Автор
с.п.о.	CAPSS [143]	MF, полностью параллельный, BLAS-1 (требует задания координат)	Pub/NETLIB

Таблица 2.2. Программы для решения разреженных линейных систем прямыми методами.

Сокращения, используемые в таблице:
несим. = несимметричная
сим. стр. = симметричная структура ненулевых элементов, значения несимметричны
сим. = симметричная, но, возможно, незнакоопределенная
с.п.о. = симметричная и положительно определенная
MF, LL и RL = многофронтальный, смотрящий влево и смотрящий вправо
SD = переключается на плотный алгоритм, когда оставшаяся матрица в достаточной степени заполнена
Pub = распространяется бесплатно; авторы могут помочь в использовании программы
Res = опубликована, но возможность ее получения у авторов не гарантируется
Com = коммерческая
HSL = Harwell Subroutine Library:
http://www.rl.ac.uk/departments/ccd/numerical/hsl/hsl.html.
UCB = http://www.cs.berkeley.edu/~xiaoye/superlu.html.
Stanford = http://www-flash.stanford.edu/apps/SPLASH/.

тричные (но возможно незнакоопределенные) матрицы и симметричные положительно определенные (с.п.о.) матрицы. Во втором столбце указано название программы (либо имена ее авторов).

В третьем столбце сообщаются характеристики алгоритмов; некоторые из них мы не имели возможности обсудить в данном учебнике. Сокращения LL (смотрящий влево), RL (смотрящий вправо), фронтальный, MF (многофронтальный) и LDL^T относятся к различным способам организации трех вложенных циклов, определяющих гауссово исключение. Термины «частичный», «Марковиц» и «барьерный» описывают различные стратегии выбора главных элементов. Обозначение «2-D блочный» характеризует распределение частей матрицы между параллельными процессорами. Алгоритм CAPSS предполагает, что линейная система соответствует некоторой сетке, и требует задания координат x, y и z узлов сетки для того, чтобы распределить матрицу между процессорами.

Кроме того, третий столбец описывает организацию самого внутреннего цикла; возможные варианты — BLAS1, BLAS2, BLAS3 либо скалярный. Символом SD обозначается алгоритм, переключающийся после k-го шага на плотное гауссово исключение, если остающаяся подматрица порядка $n-k$ уже достаточно заполнена.

В четвертом столбце дана информация о статусе и наличии программ. В частности, указано, является ли программа коммерческой или распространяется бесплатно, а также как получить доступ к ней.

2.7.5. Плотные матрицы, зависящие от менее чем $O(n^2)$ параметров

Это слишком общий заголовок, охватывающий огромное разнообразие типов матриц, встречающихся на практике. Мы сможем упомянуть лишь несколько из них.

Матрицы Вандермонда имеют вид

$$V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & & x_n \\ x_0^2 & x_1^2 & & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_0^{n-1} & x_1^{n-1} & & x_n^{n-1} \end{bmatrix}.$$

Заметим, что вычисление следующего матрично-векторного произведения

$$V^T \cdot [a_0, \ldots, a_n]^T = \left[\sum a_i x_0^i, \ldots, \sum a_i x_n^i\right]^T$$

эквивалентно задаче вычисление значений многочлена, поэтому решение системы $V^T a = y$ равносильно полиномиальной интерполяции. С помощью интерполяционного метода Ньютона мы можем решить такую систему за $\frac{5}{2}n^2$ флопов вместо $\frac{2}{3}n^3$. Для решения системы $Va = y$ имеется аналогичный прием и требуется также $\frac{5}{2}n^2$ флопов. См. [121, с. 178].

Матрица Коши C имеет элементы

$$c_{ij} = \frac{\alpha_i \beta_j}{\xi_i - \eta_j},$$

где $\alpha = [\alpha_1, \ldots, \alpha_n]$, $\beta = [\beta_1, \ldots \beta_n]$, $\xi = [\xi_1, \ldots, \xi_n]$ и $\eta = [\eta_1, \ldots, \eta_n]$ — заданные векторы. Самым известным примером является *матрица Гильберта* H, где $h_{ij} = 1/(i+j-1)$; она пользуется дурной славой вследствие чрезвычайно плохой обусловленности. Матрицы этого типа возникают при интерполировании посредством рациональных функций. Предположим, что мы хотим определить коэффициенты x_j рациональной функции

$$f(z) = \sum_{j=1}^{n} \frac{x_j}{z - \eta_j}$$

с фиксированными полюсами η_j из условий $f(\xi_i) = y_i$, $i = 1, \ldots, n$. Совокупность n уравнений $f(\xi_i) = y_i$ представляет собой линейную систему порядка n, матрица коэффициентов которой есть матрица Коши. Оказывается, что обращение матрицы Коши снова дает матрицу этого типа. Имеется замкнутое выражение для C^{-1}, использующее ее связь с задачей интерполирования:

$$(C^{-1})_{ij} = \beta_i^{-1} \alpha_j^{-1} (\xi_j - \eta_i) P_j(\eta_i) Q_i(-\xi_j),$$

где $P_j(\cdot)$ и $Q_i(\cdot)$ — интерполяционные многочлены Лагранжа

$$P_j(z) = \prod_{k \neq j} \frac{\xi_k - z}{\xi_k - \xi_j} \quad \text{и} \quad Q_i(z) = \prod_{k \neq i} \frac{-\eta_k - z}{-\eta_k + \eta_i}.$$

Теплицевы матрицы имеют вид

$$\begin{bmatrix} a_0 & a_1 & a_2 & \ldots & a_n \\ a_{-1} & \ddots & \ddots & \ddots & \vdots \\ a_{-2} & \ddots & \ddots & \ddots & a_2 \\ \vdots & \ddots & \ddots & \ddots & a_1 \\ a_{-n} & \ldots & a_{-2} & a_{-1} & a_0 \end{bmatrix},$$

т. е. они постоянны вдоль каждой диагонали. Эти матрицы возникают в задачах обработки сигналов. Имеются алгоритмы для решения линейных систем с такими матрицами, требующие только $O(n^2)$ операций.

Все указанные методы обобщаются на многие родственные типы матриц, зависящих только от $O(n)$ параметров (см. [121, с. 183] или обзор современного состояния этой области в [160]).

2.8. Литература и смежные вопросы к главе 2

Дальнейшую информацию о задаче решения линейных уравнений в общем случае можно получить в главах 3 и 4 книги [121]. Отношение взаимной обратности между числом обусловленности и расстоянием до ближайшей некорректной задачи более подробно исследуется в [71]. Вероятностный анализ роста главных элементов проделан в [242], а в [122] приведен пример роста в гауссовом исключении с полным выбором главных элементов, нарушающего гипотезу Уилкинсона. Оценщики обусловленности описаны в [138, 146, 148]. Итерационное уточнение в арифметике обычной точности анализируется в [14, 225,

226]. Очень полное обсуждение анализа ошибок для методов решения линейных систем, покрывающее большинство вопросов в этой области, можно найти в [149].

По поводу факторизации симметричных незнакоопределенных матриц см. [44]. Алгоритмы для разреженных матриц описаны в [114, 93]; см. также многочисленные ссылки в табл. 2.2. Программные реализации многих алгоритмов для плотных и ленточных матриц, описанных в этой главе, содержатся в пакетах LAPACK и CLAPACK [10]; там же можно найти обсуждение блочных алгоритмов, пригодных для высокопроизводительных компьютеров. Параллельные реализации алгоритмов даны в пакете ScaLAPACK [34]. Описание пакета BLAS имеется в [87, 89, 169]. Названные программы, как и ряд других, могут быть импортированы из NETLIB'а. Анализ блочных стратегий матричного умножения проведен в [151]. Алгоритм Штрассена для умножения матриц изложен в [3], его поведение в реальных вычислениях описано в [22], а численная устойчивость обсуждается в [77, 149]. Обзор параллельных, в том числе блочных алгоритмов дан в [76], а обзор алгоритмов для структурированных плотных систем, зависящих лишь от $O(n)$ параметров, — в недавней публикации [160]. Дополнительная информация о прямых методах для разреженных систем содержится в [93, 94, 114, 177].

2.9. Вопросы к главе 2

Вопрос 2.1 (*легкий*). С помощью своего излюбленного броузера откройте страницу NETLIB (http://www.netlib.org) и ответьте на следующие вопросы.

1. Предположим, что вам нужна фортран-подпрограмма для вычисления собственных значений и собственных векторов вещественных симметричных матриц в арифметике двойной точности. Найдите такую подпрограмму, пользуясь возможностями для поиска, предоставляемыми NETLIB'ом. Укажите имя и URL-адрес подпрограммы; опишите, как вы нашли ее.
2. Используя Performance Database Server, установите, каков в данное время мировой рекорд скорости при решении плотной системы порядка 100 посредством гауссова исключения. Какова скорость в мегафлопах и на какой машине она достигнута? Проделайте то же самое для плотных линейных систем порядка 1000 и плотных систем «как угодно большого порядка». Пользуясь этой базой данных, выясните, насколько быстро может решить плотную линейную систему порядка 100 ваша рабочая станция. Указание: загляните в отчет LINPACK Benchmark.

Вопрос 2.2 (*легкий*). Пусть нужно решить относительно X уравнение $AX = B$, где A — матрица размера $n \times n$, а X и B имеют размер $n \times m$. Имеются два очевидных алгоритма для этой задачи. В первом алгоритме вычисляется разложение $A = PLU$ с помощью гауссова исключения, а затем каждый столбец в X определяется посредством прямой подстановки и обратного хода. Во втором алгоритме гауссово исключение используется для вычисления обратной матрицы A^{-1}, а затем производится умножение $X = A^{-1}B$. Подсчитайте число флопов, выполняемых каждым алгоритмом, и покажите, что для первого алгоритма оно меньше, чем для второго.

Вопрос 2.3 (*средней трудности*). Пусть $\|\cdot\|$ есть 2-норма, и пусть заданы невырожденная матрица A и вектор b. Показать, что для достаточно малых $\|\delta A\|$ найдутся ненулевые δA и δb, такие, что неравенство (2.2) превращается в равенство. Это оправдывает название «число обусловленности матрицы A», принятое для величины $\kappa(A) = \|A^{-1}\|\,\|A\|$. Указание: используйте идеи из доказательства теоремы 2.1.

Вопрос 2.4 (*трудный*). Показать, что оценки (2.7) и (2.8) достижимы.

Вопрос 2.5 (*средней трудности*). Доказать теорему 2.3. Пусть $r = A\hat{x} - b$. Используя теорему 2.3, показать, что граница (2.9) не превосходит границы (2.7). Это объясняет, почему в LAPACK'е вычисляется оценка, основанная на (2.9) (см. разд. 2.4.4).

Вопрос 2.6 (*легкий*). Доказать лемму 2.2.

Вопрос 2.7 (*легкий; Z. Bai*). Пусть A — невырожденная симметричная матрица, разложенная в произведение $A = LDM^T$, где L и M — нижние унитреугольные матрицы, а D — диагональная матрица. Доказать, что $L = M$.

Вопрос 2.8 (*трудный*). Рассмотрим два способа для решения системы из двух линейных уравнений с двумя неизвестными

$$Ax = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b.$$

Алгоритм 1. Гауссово исключение с частичным выбором главных элементов (метод GEPP).
Алгоритм 2. Формулы Крамера:

$$\begin{aligned} \det &= a_{11} * a_{22} - a_{12} * a_{21}, \\ x_1 &= (a_{22} * b_1 - a_{12} * b_2)/\det, \\ x_2 &= (-a_{21} * b_1 + a_{11} * b_2)/\det. \end{aligned}$$

Показать с помощью численного примера, что формулы Крамера не являются обратно устойчивым алгоритмом. Указание: выбрать почти вырожденную матрицу A и правую часть $[b_1, b_2]^T \approx [a_{12}\, a_{22}]^T$. Какое влияние имеет свойство обратной устойчивости на величину невязки? Свой численный пример вы можете построить вручную на бумаге (используя, например, четырехразрядную десятичную арифметику с плавающей точкой) либо с помощью карманного калькулятора или компьютера.

Вопрос 2.9 (*средней трудности*). Пусть B — верхняя двухдиагональная матрица порядка n, т. е. ненулевые элементы присутствуют лишь на ее главной диагонали и первой наддиагонали. Построить алгоритм, *точно* вычисляющий величину $\kappa_\infty(B) \equiv \|B\|_\infty \|B^{-1}\|_\infty$ (ошибки округлений игнорируются). Другими словами, нельзя пользоваться итерационным алгоритмом типа оценщика Хэйджера. Ваш алгоритм должен быть возможно более экономичным; оказывается, что можно ограничиться $2n - 2$ сложениями, n умножениями, n делениями, $4n - 2$ операциями взятия модуля и $2n - 2$ сравнениями. (Если ваш алгоритм не укладывается в эти границы, но близок к ним, то он вполне приемлем.)

Вопрос 2.10 (*легкий; Z. Bai*). Пусть A — матрица размера $n \times n$. Показать, что $\|AT^TA\|_2 = \|A\|_2^2$ и $\kappa_2(A^TA) = \kappa_2(A)^2$.

Пусть M — положительно определенная $n \times n$-матрица, а L — ее множитель Холесского, так что $M = LL^T$. Показать, что $\|M\|_2 = \|L\|_2^2$ и $\kappa_2(M) = \kappa_2(L)^2$.

Вопрос 2.11 (*легкий; Z. Bai*). Пусть A — симметричная положительно определенная матрица. Показать, что $|a_{ij}| < (a_{ii}\, a_{jj})^{1/2}$.

Вопрос 2.12 (*легкий; Z. Bai*). Показать, что если

$$Y = \begin{pmatrix} I & Z \\ 0 & I \end{pmatrix},$$

где I — единичная $n \times n$-матрица, то $\kappa_F(Y) = \|Y\|_F \, \|Y^{-1}\|_F = 2n + \|Z\|_F^2$.

Вопрос 2.13 (*средней трудности*). В этом вопросе мы обсудим, как решать систему $By = c$, если известен быстрый способ решения систем вида $Ax = b$ и матрица $A - B$ в том или ином смысле «мала».

1. Доказать *формулу Шермана–Моррисона*: пусть A — невырожденная матрица, u и v — векторы-столбцы и матрица $A + uv^T$ также невырожденна. Тогда $(A + uv^T)^{-1} = A^{-1} - (A^{-1}uv^TA^{-1})/(1 + v^TA^{-1}u)$.
   Более общо, доказать *формулу Шермана–Моррисона–Вудбери*: пусть U и V — прямоугольные матрицы размера $n \times k$, где $k \leq n$, и пусть A — квадратная $n \times n$-матрица. Матрица $T = I + V^TA^{-1}U$ тогда и только тогда невырожденна, когда невырожденна матрица $A + UV^T$; при этом $(A + UV^T)^{-1} = A^{-1} - A^{-1}UT^{-1}V^TA^{-1}$.
2. Предположим, что имеется быстрый алгоритм для решения систем вида $Ax = b$. Показать, как построить быстрый алгоритм для решения системы $By = c$, где $B = A + uv^T$.
3. Предположим, что имеется быстрый алгоритм для решения систем вида $Ax = b$, и пусть величина $\|A - B\|$ мала. Описать итерационную схему для решения системы $By = c$. Насколько быстрой сходимости вы ожидаете от своего алгоритма? Указание: использовать итерационное уточнение.

Вопрос 2.14 (*средней трудности; программирование*). Импортируйте из Netlib подпрограмму, решающую систему $Ax = b$ гауссовым исключением с частичным выбором главных элементов. Вы должны найти ее в LAPACK'е (на фортране, NETLIB/lapack) или CLAPACK'е (на языке C, NETLIB/clapack); в обоих случаях `sgesvx` есть главная программа. (Возможно, вы захотите поработать и с более простой программой `sgesv`.) Измените `sgesvx` (и, возможно, некоторые из подпрограмм, к которым она обращается) так, чтобы вместо частичного выполнялся полный выбор; полученную программу назовите `gecp`. Вероятно, простейшим вариантом является модификация программы `sgetf2` и использование ее вместо `sgetrf`. Matlab-реализацию программы `gecp` см. в HOMEPAGE/Matlab/gecp.m. Сравните `sgesvx` и `gecp` на многих случайно порожденных матрицах различных порядков (доходящих, скажем, до 30). Задавая x и вычисляя $b = Ax$, вы можете получить системы, для которых правильный ответ известен. Точность вычисленного решения $\hat{x}$ проверяйте следующим образом. Сначала исследуйте границы ошибок `FERR` («Forward ERRor»)

и BERR («Backward ERRor»), вычисляемые программами; своими словами объясните смысл этих границ. Пользуясь знанием точного ответа, убедитесь, что граница FERR верна. Далее найдите точное число обусловленности путем явного обращения матрицы и сравните это число с оценкой RCOND, определяемой программой. (На самом деле, RCOND есть оценка величины, обратной к числу обусловленности.) Затем проверьте, что отношение $\frac{\|\hat{x}-x\|}{\|\hat{x}\|}$ ограничено умеренным кратным числа *macheps*/Rcond. Наконец, нужно проверить, что (масштабированная) обратная ошибка $R \equiv \|A\hat{x} - b\|/((\|A\| \cdot \|x\| + \|b\|) \cdot macheps)$ в каждом случае является величиной порядка единицы.

Более точно, ваш ответ на данный вопрос должен включать в себя хорошо документированную распечатку программы `gecp`, объяснение, как генерировались случайные матрицы (см. об этом ниже), и таблицу, содержащую следующие столбцы (или, что предпочтительно, графики каждого из столбцов данных, отнесенного к первому столбцу):

- номер тестовой матрицы (соответствующий ее номеру в объяснении способа генерирования);
- ее порядок;
- информация, получаемая от `sgesvx`:
  — коэффициент роста, определяемый программой (в идеальном случае, он не должен быть много больше единицы),
  — оценка числа обусловленности (1/RCOND),
  — отношение числа 1/RCOND к явно вычисленному вами числу обусловленности (в идеальном случае, это отношение должно быть близко к единице),
  — граница ошибки FERR,
  — отношение числа FERR к подлинной ошибке (в идеальном случае, это отношение не меньше единицы и не намного больше ее; исключением является «удачный» случай, когда подлинная ошибка равна нулю),
  — отношение подлинной ошибки к числу ε/RCOND (в идеальном случае, это отношение должно быть не больше единицы или, чуть меньше ее; исключением является «удачный» случай, когда подлинная ошибка равна нулю),
  — масштабированная обратная ошибка R/ε (в идеальном случае, должна быть величиной порядка $O(1)$ или, возможно, $O(n)$),
  — обратная ошибка BERR/ε (в идеальном случае, должна быть величиной порядка $O(1)$ или, возможно, $O(n)$),
  — время работы программы в секундах;
- такая же информация, получаемая от `gecp`.

Данные достаточно распечатывать с одним десятичным разрядом, поскольку нас интересуют лишь приблизительные значения. Являются ли вычисляемые границы ошибок подлинными оценками? Какова сравнительная скорость программ `sgesvx` и `gecp`?

На многих системах точное время работы программы зарегистрировать трудно из-за низкой разрешающей способности таймера. Поэтому время работы следует вычислять так:

t_1 = текущее время
for $i = 1$ to m
    сформировать задачу

```
    решить задачу
endfor
t2 = текущее время
for i = 1 to m
    сформировать задачу
endfor
t3 = текущее время
t = ((t2 − t1) − (t3 − t2))/m
```

Значение m нужно брать настолько большим, чтобы разность $t_2 - t_1$ составляла хотя бы несколько секунд. В этом случае t должно быть надежной оценкой времени решения задачи.

Вы должны провести тестирование как на хорошо обусловленных задачах, так и на плохо обусловленных. Чтобы получить хорошо обусловленную матрицу, выберите матрицу-перестановку P и добавьте к каждому ее элементу малое случайное число. Чтобы получить плохо обусловленную матрицу, возьмите случайную нижнетреугольную матрицу L с малыми диагональными элементами и поддиагональными элементами умеренной величины. Пусть U — аналогичным образом построенная верхнетреугольная матрица, и пусть $A = LU$. (Вы можете также, если захотите, воспользоваться LAPACK-подпрограммой `slatms`, предназначенной для генерирования случайных матриц с требуемым числом обусловленности.)

Испытайте обе программы еще на одном классе $n \times n$-матриц для значений n от 1 до 30. (Если вы работаете в арифметике двойной точности, то лучше довести вычисления до $n = 60$.) Устройство этих матриц покажем для $n = 5$; для других n оно аналогично:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Объяснить степень точности полученых результатов в терминах анализа ошибок, проведенного в разд. 2.4.

Ваш ответ *не* должен содержать распечаток матриц и решений линейных систем.

Помимо того, чтобы научить вас пользоваться оценками ошибок, данное задание преследует еще цель показать вам, как выглядят хорошо реализованные программы численного анализа. В своей практической работе вам, возможно, не раз придется использовать или модифицировать существующие программы вместо того, чтобы писать свою полностью новую программу.

Вопрос 2.15 (*средней трудности; программирование*). Этот вопрос связан с вопросом 2.14. Составьте еще один вариант программы `sgesvx` с именем `sgesvxdouble`; в нем невязка должна вычисляться с двойной точностью при итерационном уточнении решения. Измените границу ошибки `FERR` в `sgesvx`, чтобы отразить увеличившуюся точность. Объясните произведенное изменение. (Это может потребовать от вас разобраться, в первую очередь, с тем, как вычисляется граница ошибки в `sgesvx`). Для того же набора тестовых приме-

ров, что и в предыдущем вопросе, постройте аналогичную таблицу данных. В каких случаях программа `sgesvxdouble` более точна, чем `sgesvx`?

Вопрос 2.16 (*трудный*). Объясните, как следует модифицировать алгоритм Холесского (алгоритм 2.11), чтобы бóльшая часть операций производилась в нем посредством процедур уровня 3 из BLAS. Действуйте, как в алгоритме 2.10.

Вопрос 2.17 (*легкий*). Предположим, что вы имеете в Matlab'е $n \times n$-матрицу A и $n \times 1$-матрицу b. Что означают в Matlab'е записи $A\backslash b$, b'/A и A/b? Как $A\backslash b$ отличается от $\mathrm{inv}(A) * b$?

Вопрос 2.18 (*средней трудности*). Пусть

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

где $k \times k$-подматрица A_{11} невырожденна. Тогда матрица $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ называется *дополнением Шура подматрицы* A_{11} в A или, для краткости, просто дополнением Шура.

1. Показать, что после k шагов гауссова исключения без выбора главных элементов на месте A_{22} находится матрица S.
2. Предположим, что $A = A^T$, подматрица A_{11} положительно определена, а A_{22} отрицательно определена (т. е. $-A_{22}$ положительно определена). Показать, что для невырожденной матрицы A будет (в точной арифметике) работать гауссово исключение без выбора главных элементов, но в машинной арифметике оно может быть численно неустойчивым (это можно показать с помощью примера 2×2).

Вопрос 2.19 (*средней трудности*). Говорят, что матрица A имеет *строгое диагональное преобладание по столбцам*, или, для краткости, диагональное преобладание, если

$$|a_{ii}| > \sum_{j=1,\, j\neq i}^{n} |a_{ij}|.$$

- Показать, что A невырожденна. Указание: воспользоваться теоремой Гершгорина.
- Показать, что в гауссовом исключении с частичным выбором главных элементов строки такой матрицы в действительности не переставляются, т. е. оно идентично исключению без выбора главных элементов. Указание: покажите, что после одного шага гауссова исключения оставшаяся подматрица размера $(n-1) \times (n-1)$ (т. е. *дополнение Шура элемента* a_{11} в A) попрежнему имеет диагональное преобладание. (Более подробно дополнения Шура обсуждаются в вопросе 2.18.)

Вопрос 2.20 (*легкий; Z. Bai*). Пусть дана невырожденная $n \times n$-матрица A. Как, используя гауссово исключение с частичным выбором главных элементов, эффективно решать следующие задачи:

а) найти решение линейной системы $A^k x = b$, где k — натуральное число;

b) найти число $\alpha = c^T A^{-1} b$;

c) решить матричное уравнение $AX = B$, где B — матрица размера $n \times m$.

Вы должны: 1) описать свои алгоритмы; 2) представить их псевдокодом (используя язык типа Matlab'a; алгоритм GEPP записывать не нужно); 3) указать требуемое число операций.

Вопрос 2.21 (*средней трудности*). Для n, являющегося степенью двойки, доказать, что алгоритм Штрассена (алгоритм 2.8) правильно перемножает $n \times n$-матрицы.

Глава 3

Линейные задачи наименьших квадратов

3.1. Введение

Пусть даны $m \times n$-матрица A и m-вектор b. *Линейная задача наименьших квадратов* заключается в разыскании n-вектора x, минимизирующего величину $||Ax - b||_2$. Если $m = n$ и матрица A невырожденна, то решением задачи является вектор $x = A^{-1}b$. Если $m > n$, т. е. число уравнений больше числа неизвестных, то задача называется *переопределенной*; в этом случае, вообще говоря, не существует вектора x, точно удовлетворяющего системе $Ax = b$. Иногда встречаются и *недоопределенные* задачи, где $m < n$, однако мы сосредоточимся на более общем переопределенном случае.

Данная глава организована следующим образом. В остальной части настоящего введения описаны три приложения задачи наименьших квадратов, а именно *аппроксимация данных*, *статистическое моделирование* при наличии шума и *геодезическое моделирование*. В разд. 3.2 обсуждаются три стандартных способа решения задачи наименьших квадратов: *нормальные уравнения*, *QR-разложение* и *сингулярное разложение* (SVD). Последнее часто используется как инструмент в дальнейших главах, поэтому мы выведем некоторые его свойства (хотя описание алгоритмов вычисления SVD отложено до гл. 5). Раздел 3.3 посвящен теории возмущений для задач наименьших квадратов, а разд. 3.4 — деталям реализации и анализу ошибок округлений в нашем главном методе, а именно QR-разложении. Этот анализ приложим ко многим алгоритмам, использующим ортогональные матрицы, включая алгоритмы вычисления собственных значений и SVD из гл. 4 и 5. В разд. 3.5 рассматривается ситуация особенно плохой обусловленности при неполном ранге задачи наименьших квадратов; мы обсуждаем, как в этом случае получить решение приемлемой точности. В разделе 3.7 и вопросах в конце главы даны сведения о других типах задач наименьших квадратов и программах для решения разреженных задач.

Пример 3.1. *Аппроксимация данных* является традиционным приложением метода наименьших квадратов. Пусть даны m пар чисел $(y_1, b_1), \ldots, (y_m, b_m)$. Предположим, что мы хотим найти кубический полином, который бы «наилучшим образом» представлял числа b_i как функцию от y_i. Это означает, что коэффициенты $x_1, \ldots, x_4$ нужно определить так, чтобы многочлен $p(y) = \sum_{j=1}^{4} x_j y^{j-1}$ минимизировал невязку $r_i \equiv p(y_i) - b_i$ для i от 1 до m.

То же самое означает задача минимизации вектора

$$
\begin{aligned}
r \equiv \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} &= \begin{bmatrix} p(y_1) \\ p(y_2) \\ \vdots \\ p(y_m) \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \\
&= \begin{bmatrix} 1 & y_1 & y_1^2 & y_1^3 \\ 1 & y_2 & y_2^2 & y_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_m & y_m^2 & y_m^3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \\
&\equiv A \cdot x - b,
\end{aligned}
$$

где r и b суть m-векторы, A — матрица размера $m \times 4$, а x — вектор размерности 4. Для минимизации r можно выбрать любую норму, например $\|r\|_\infty$, $\|r\|_1$ или $\|r\|_2$. В последнем случае получаем *линейную задачу наименьших квадратов*; она соответствует минимизации сумм квадратов невязок $\sum_{i=1}^m r_i^2$.

В примере на рис. 3.1 гладкая функция $b = \sin(\pi y/5) + y/5$ аппроксимируется многочленами возрастающих степеней по 23 точкам $y = -5, -4.5, -4, \ldots, 5.5, 6$. В левой части рисунка представлены исходные данные (изображены кружками) и четыре аппроксимирующих многочлена степеней 1, 3, 6 и 19. В правой части рисунка норма невязки $\|r\|_2$ представлена как функция от степени многочлена для значений степени от 1 до 20. Обратим внимание на то, что норма невязки убывает при возрастании степени от 1 до 17. Этого и следовало ожидать, ведь увеличение степени многочлена должно позволить нам лучше аппроксимировать данные.

Однако, когда значение степени достигает 18, норма невязки неожиданно и очень резко возрастает. На левом рисунке можно видеть, как хаотически ведет себя график многочлена 19-й степени (сплошная линия). Мы увидим позднее, что это связано с плохой обусловленностью задачи. Обычно, чтобы избежать плохо обусловленных задач, для аппроксимации используют многочлены сравнительно невысоких степеней [61]. В Matlab'е полиномиальная аппроксимация данных осуществляется функцией `polyfit`.

Полиномиальная аппроксимация допускает альтернативу. В более общем случае, имеется система независимых функций $f_1(y), \ldots, f_n(y)$, действующих из $\mathbb{R}^k$ в $\mathbb{R}$, и набор точек $(y_1, b_1), \ldots, (y_m, b_m)$, где $y_i \in \mathbb{R}^k$ и $b \in \mathbb{R}$. Нужно найти наилучшее приближение этих точек вида $b = \sum_{j=1}^n x_j f_j(y)$. Другими словами, нужно выбрать $x = [x_1, \ldots, x_n]^T$ так, чтобы минимизировать невязки $r_i \equiv \sum_{j=1}^n x_j f_j(y_i) - b_i$ для $1 \le i \le m$. Полагая $a_{ij} = f_j(y_i)$, можем записать это в виде $r = Ax - b$, где A — матрица размера $m \times n$, x — вектор размерности n, а b и r — векторы размерности m. Удачный выбор базисных функций $f_i(y)$ может дать лучшее приближение и лучше обусловленную задачу, чем при использовании многочленов [33, 84, 168]. ◇

Пример 3.2. В статистическом моделировании часто приходится оценивать некоторые параметры x_j, основываясь на наблюдениях, «загрязненных» шумом. Предположим, например, что в рамках кампании по приему в колледж желательно предсказать средний балл (b) будущего обучения в нем, исходя из

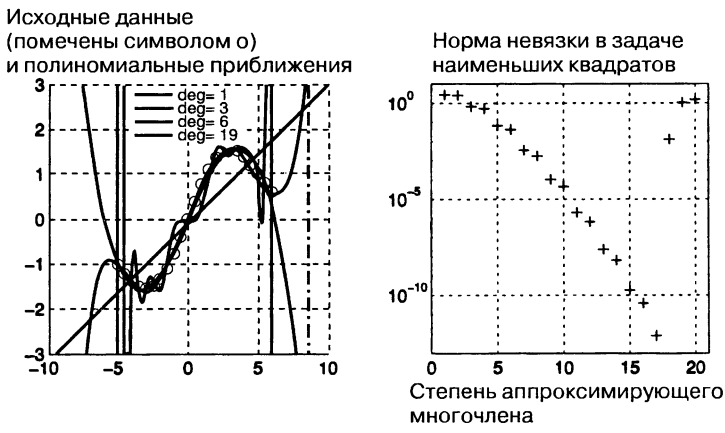


Рис. 3.1. Полиномиальная аппроксимация кривой $b = \sin(\pi y/5) + y/5$ и нормы невязок (см. цветной вариант рисунка на обложке книги. — *Перев.*).

среднего балла (a_1), который абитуриент имел в старших классах школы, и результатов двух тестов готовности: устного (a_2) и письменного (a_3). Основываясь на прошлых данных для ранее принятых студентов, можно построить *линейную модель* вида $b = \sum_{j=1}^{3} a_j x_j$. Наблюдениями являются четверки чисел a_{i1}, a_{i2}, a_{i3} и b_i, по одной для каждого из m студентов в базе данных. Итак, желательно минимизировать величину

$$r \equiv \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \equiv A \cdot x - b,$$

что можно рассматривать как задачу наименьших квадратов.

Существует статистическое обоснование метода наименьших квадратов, который статистики называют *линейной регрессией*. Предположим, что числа a_i известны точно, так что шум присутствует лишь в числах b_i; при этом шумы в различных b_i независимы и нормально распределены с нулевым средним и одним и тем же средним квадратичным отклонением σ. Пусть x — решение задачи наименьших квадратов, а x_T — вектор истинных значений параметров. Тогда x называется *оценкой наибольшего правдоподобия* для x_T, а ошибка $x - x_T$ нормально распределена и имеет нулевое среднее и *ковариационную матрицу* $\sigma^2(A^TA)^{-1}$. Мы встретимся с матрицей $(A^TA)^{-1}$ снова при решении задачи наименьших квадратов методом нормальных уравнений. Более подробные сведения о статистических аспектах задачи[1] можно найти в [33, 259]. ◇

Пример 3.3. Задача наименьших квадратов впервые была поставлена и сформулирована Гауссом при решении практической задачи по заказу германского

[1] Стандартные обозначения в статистике отличаются от принятых в линейной алгебре: статистики пишут $X\beta = y$ вместо $Ax = b$.

правительства. Существуют важные экономические и юридические причины для точного определения границ между участками земли, принадлежащими разным хозяевам. В таких случаях на сцену выступают землемеры, которые определяют эти границы, исходя из известных межевых вех, измеряя некоторые углы и расстояния и производя затем триангуляцию. С течением времени стало необходимым повысить точность задания координат межевых вех. К делу снова приступили землемеры, заново измерившие множество углов и расстояний между вехами. Задача Гаусса состояла в том, чтобы предложить метод обновления координат вех из государственной базы данных, основываясь на этих более точных измерениях. Для этой цели он и изобрел метод наименьших квадратов, который мы вскоре изложим [33].

Задача, решенная Гауссом, не потеряла своей актуальности и к ней периодически приходится возвращаться. В 1974 г. Национальная Геодезическая Служба США начала работу по обновлению геодезической базы данных США, состоящей из приблизительно 700 000 точек. Возросшие стимулы для такой работы включали в себя предоставление достаточно точных данных инженерам и местным планировщикам для строительных проектов и геофизикам для изучения движения тектонических плит земной коры (они могут перемещаться на расстояние до 5 см за год). Соответствующая задача наименьших квадратов была на тот момент самой большой из когда-либо решавшихся: примерно 2.5 миллиона уравнений относительно 400 000 переменных. Кроме того, задача была очень разреженной, что сделало возможным ее решение в 1978 г. на имевшихся в то время компьютерах [164].

Теперь мы кратко обсудим постановку задачи. Она, в действительности, нелинейная и решается посредством аппроксимации ее последовательностью линейных задач; каждая из них есть линейная задача наименьших квадратов. База данных представляет собой список точек (геофизических вех), снабженных координатами: широтой, долготой и, возможно, высотой. Для простоты изложения будем считать землю плоской и сопоставим каждой точке i линейные координаты $z_i = (x_i, y_i)^T$. Для всех точек нужно вычислить поправки $\delta z_i = (\delta x_i, \delta y_i)^T$ так, чтобы измененные координаты $z_i' = (x_i', y_i')^T = z_i + \delta z_i$ лучше соответствовали новым, более точным измерениям. Эти измерения включают в себя как расстояния между избранными парами точек, так и углы между отрезками, соединяющими точку i с точками j и k (см. рис. 3.2). Чтобы понять, как превратить эти новые измерения в уравнения, рассмотрим треугольник на рис. 3.2. Его углы помечены соответствующими (поправленными) координатами; показаны также углы θ и длины сторон L. С помощью этих данных и простых тригонометрических тождеств легко выписать необходимые уравнения. Например, улучшенное измерение для θ_i можно использовать в соотношении

$$\cos^2\theta_i = \frac{[(z_j' - z_i')^T(z_k' - z_i')]^2}{(z_j' - z_i')^T(z_j' - z_i') \cdot (z_k' - z_i')^T(z_k' - z_i')},$$

где $\cos\theta_i$ выражен посредством скалярных произведений соответствующих сторон треугольника. Предполагая, что величины δz_i малы в сравнении с z_i, это соотношение можно линеаризовать таким образом: умножим обе части на знаменатель дроби, выполним все необходимые умножения, что приводит к многочлену четвертой степени относительно «δ-переменных» (вроде δx_i), и отбросим все члены, содержащие в качестве множителя более чем одно δ-переменное. Это

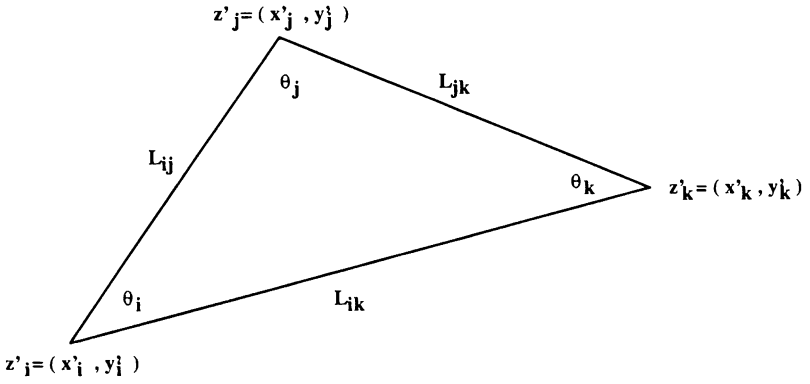


Рис. 3.2. Вывод уравнений при обновлении геодезической базы данных.

дает уравнение, в которое δ-переменные входят линейно. Если собрать подобные линейные уравнения для всех новых измерений углов и расстояний, то получится переопределенная линейная система относительно всех δ-переменных. Мы хотели бы найти наименьшие поправки, т. е. наименьшие значения величин δx_i и т. д., которые бы наилучшим образом соответствовали этим поправкам. Это и есть задача наименьших квадратов. ◇

Позднее, когда будет развит соответствующий аппарат, мы покажем, что и *сжатие изображений* может интерпретироваться как задача наименьших квадратов (см. пример 3.4).

3.2. Матричные разложения для решения линейной задачи наименьших квадратов

Линейная задача наименьших квадратов может быть решена несколькими явными способами, к обсуждению которых мы теперь переходим. Эти способы:

1. нормальные уравнения,
2. QR-разложение,
3. SVD,
4. преобразование в линейную систему (см. вопрос 3.3).

Первый метод наиболее быстрый, но и наименее точен; он пригоден для задач с малым числом обусловленности. Второй метод является наиболее стандартным; его стоимость может превышать стоимость первого метода в два раза. Третий метод наиболее полезен для плохо обусловленных задач, т. е. задач, в которых матрица A не имеет полного ранга; он в несколько раз дороже предыдущих методов. Последний метод позволяет проводить итерационное уточнение приближенного решения в случае плохо обусловленной задачи. Все методы, кроме третьего, могут быть адаптированы для эффективной обработки разреженных матриц [33]. Мы последовательно обсудим все эти методы. Для методов 1 и 2 мы поначалу предположим, что A имеет полный столбцевой ранг n.

3.2.1. Нормальные уравнения

Для вывода *нормальных уравнений* будем искать точку x, в которой градиент функции $\|Ax-b\|_2^2 = (Ax-b)^T(Ax-b)$ обращается в нуль. Итак, мы хотим, чтобы

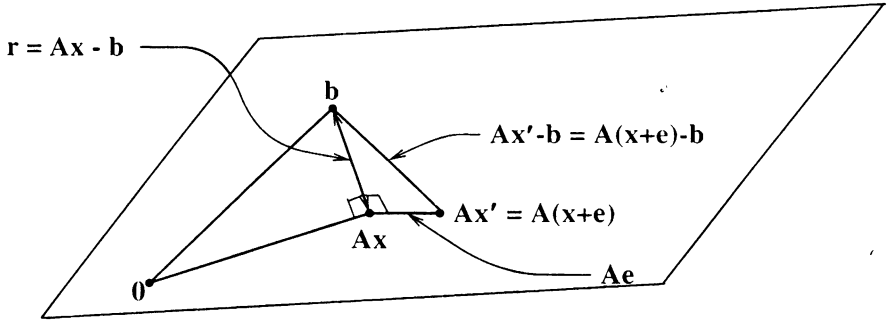
$$\begin{aligned} 0 &= \lim_{e\to 0} \frac{(A(x+e)-b)^T(A(x+e)-b) - (Ax-b)^T(Ax-b)}{\|e\|_2} \\ &= \lim_{e\to 0} \frac{2e^T(A^TAx - A^Tb) + e^TA^TAe}{\|e\|_2}. \end{aligned}$$

Второе слагаемое $\frac{|e^TA^TAe|}{\|e\|_2} \le \frac{\|A\|_2^2\|e\|_2^2}{\|e\|_2}$ стремится к нулю при $e\to 0$, поэтому множитель $A^TAx - A^Tb$ в первом слагаемом также должен быть нулем, т. е. $A^TAx = A^Tb$. Это система из n линейных уравнений от n неизвестных, называемая системой нормальных уравнений.

Почему вектор $x = (A^TA)^{-1}A^Tb$ дает минимум функции $\|Ax-b\|_2^2$? Это можно показать, замечая, что гессиан A^TA положительно определен; следовательно, функция строго выпукла и всякая ее критическая точка есть точка глобального минимума. Или же, полагая $x' = x+e$, можно упростить функцию к виду суммы квадратов:

$$\begin{aligned} (Ax'-b)^T(Ax'-b) &= (Ae+Ax-b)^T(Ae+Ax-b) \\ &= (Ae)^T(Ae) + (Ax-b)^T(Ax-b) \\ &+ 2(Ae)^T(Ax-b) \\ &= \|Ae\|_2^2 + \|Ax-b\|_2^2 + 2e^T(A^TAx - A^Tb) \\ &= \|Ae\|_2^2 + \|Ax-b\|_2^2. \end{aligned}$$

Ясно, что сумма минимизируется выбором $e=0$. Это не что иное, как теорема Пифагора, поскольку невязка $r = Ax-b$ ортогональна к подпространству, натянутому на столбцы матрицы A, т. е. $0 = A^Tr = A^TAx - A^Tb$. Это иллюстрируется рисунком; показанная на нем плоскость есть линейная оболочка столбцов матрицы A, поэтому векторы Ax, Ae и $Ax' = A(x+e)$ принадлежат ей.



Поскольку матрица A^TA — симметричная и положительно определенная, для решения нормальных уравнений можно применить разложение Холесского. Общая стоимость вычисления A^TA и A^Tb и последующего разложения мат-

рицы A^TA составляет $n^2m + \frac{1}{3}n^3 + O(n^2)$ флопов. Так как $m \geq n$, то в общей стоимости преобладает цена n^2m формирования матрицы A^TA.

3.2.2. QR-разложение

Теорема 3.1. (QR-разложение). *Пусть A — матрица размера $m \times n$, причем $m \geq n$. Предположим, что A имеет полный столбцевой ранг. Тогда существуют и единственны $m \times n$-матрица Q с ортонормированными столбцами (т. е. $Q^TQ = I_n$) и верхнетреугольная $n \times n$-матрица R с положительными диагональными элементами r_{ii}, такие, что $A = QR$.*

Доказательство . Будут даны два доказательства этой теоремы. Прежде всего, теорема представляет собой переформулировку процесса ортогонализации Грама–Шмидта [139]. Если этот процесс применить к столбцам a_i матрицы $A = [a_1, a_2, \ldots, a_n]$ в порядке возрастания их номеров, то будет получена система ортонормированных векторов $q_1, \ldots, q_n$, дающих другой базис того же подпространства. Эти ортонормированные векторы являются столбцами матрицы Q. В процессе Грама–Шмидта вычисляются также коэффициенты $r_{ji} = q_j^T a_i$ в выражении каждого столбца как линейной комбинации векторов $q_1, \ldots, q_i : a_i = \sum_{j=1}^{i} r_{ji} q_j$. Эти числа r_{ij} суть элементы матрицы R.

Алгоритм 3.1. *Классический алгоритм Грама–Шмидта (CGS) и модифицированый алгоритм Грама–Шмидта (MGS) для вычисления разложения $A = QR$:*

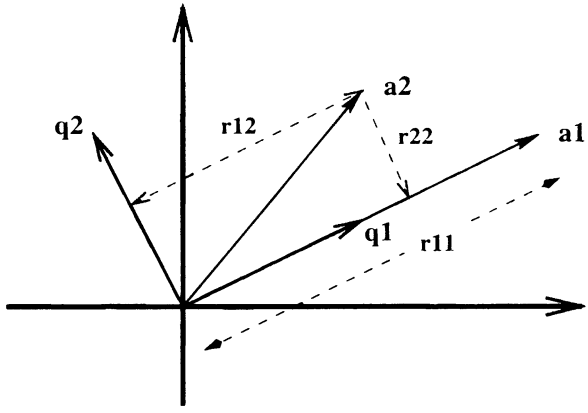
for $i = 1$ *to* n /* *вычислить i-е столбцы матриц Q и R* */
  $q_i = a_i$
  for $j = 1$ *to* $i - 1$ /* *вычесть из a_i компоненту в направлении q_j* */
    $\begin{cases} r_{ji} = q_j^T a_i & CGS \\ r_{ji} = q_j^T q_i & MGS \end{cases}$
  $q_i = q_i - r_{ij} q_j$
  end for
  $r_{ii} = \|q_i\|_2$
    if $r_{ii} = 0$ /* *a_i линейно зависит от $a_1, \ldots, a_{i-1}$* */
  выход
  end if
$q_i = q_i / r_{ii}$

Предоставляем читателю в качестве упражнения проверку, что две формулы для r_{ji} в этом алгоритме математически эквивалентны (см. вопрос 3.1). Если A имеет полный столбцевой ранг, то ни один из элементов r_{ii} не будет нулем. На рисунке процесс Грама—Шмидта проиллюстрирован для 2×2-матрицы A.

Второе доказательство теоремы использует алгоритм 3.2, который будет представлен в разд. 3.4.1. □

К сожалению, в арифметике с плавающей точкой алгоритм CGS численно неустойчив, если столбцы матрицы A почти линейно зависимы. Алгоритм MGS более устойчив и используется в алгоритмах, приводимых в этой книге далее. Однако, если A плохо обусловлена, то матрица Q может сильно отличаться от ортогональной, т. е. величина $\|Q^TQ - I\|$ значительно больше числа ε [31, 32,

33, 149]. Алгоритм 3.2 в разд. 3.4.1 является устойчивой альтернативой при вычислении разложения $A = QR$ (см. вопрос 3.2).



С помощью разложения $A = QR$ мы тремя несколько различающимися способами выведем формулу для вектора x, минимизирующего функцию $\|Ax - b\|_2$. Во-первых, всегда можно найти еще $m - n$ ортонормированных векторов, составляющих матрицу $\tilde{Q}$, так, чтобы матрица $[Q, \tilde{Q}]$ была квадратной и ортогональной (например, можно взять дополнительные $m - n$ линейно независимых векторов, составляющих матрицу $\tilde{X}$, и применять алгоритм 3.1 к невырожденной $n \times n$-матрице $[Q, \tilde{X}]$). Тогда

$$
\begin{aligned}
\|Ax - b\|_2^2 &= \|[Q, \tilde{Q}]^T (Ax - b)\|_2^2 \quad \text{согласно утверждению 4 леммы 1.7} \\
&= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (QRx - b) \right\|_2^2 \\
&= \left\| \begin{bmatrix} I^{n\times n} \\ O^{(m-n)\times n} \end{bmatrix} Rx - \begin{bmatrix} Q^T b \\ \tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} Rx - Q^T b \\ -\tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\
&= \|Rx - Q^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2 \\
&\geq \|\tilde{Q}^T b\|_2^2.
\end{aligned}
$$

Систему $Rx - Q^T b = 0$ можно решить относительно вектора x, потому что A и R имеют один и тот же ранг n; следовательно, матрица R невырождена. Поэтому $x = R^{-1} Q^T b$, а минимальное значение функции $\|Ax - b\|_2$ равно $\|\tilde{Q}^T b\|_2$.

Приведем чуть иное рассуждение, не использующее матрицу $\tilde{Q}$. Запишем вектор $Ax - b$ в виде

$$
\begin{aligned}
Ax - b &= QRx - b = QRx - (QQ^T + I - QQ^T) b \\
&= Q(Rx - Q^T b) - (I - QQ^T) b.
\end{aligned}
$$

Заметим, что векторы $Q(Rx-A^Tb)$ и $(I-QQ^T)b$ ортогональны, так как $(Q(Rx-Q^Tb))^T((I-QQ^T)b) = (Rx-Q^Tb)^T[Q^T(I-QQ^T)]b = (Rx-Q^Tb)^T[0]b = 0$. Поэтому по теореме Пифагора,

$$\begin{aligned}\|Ax-b\|_2^2 &= \|Q(Rx-Q^Tb)\|_2^2 + \|(I-QQ^T)b\|_2^2 \\ &= \|Rx-Q^Tb\|_2^2 + \|(I-QQ^T)b\|_2^2,\end{aligned}$$

где использовано равенство $\|Qy\|_2^2 = \|y\|_2^2$ (см. утверждение 4 леммы 1.7). Эта сумма квадратов будет минимальна, если ее первое слагаемое равно нулю, т. е. если $x = R^{-1}Q^Tb$.

Наконец, есть и третий вывод, который исходит из формулы для решения системы нормальных уравнений:

$$\begin{aligned}x &= (A^TA)^{-1}A^Tb \\ &= (R^TQ^TQR)^{-1}R^TQ^Tb = (R^TR)^{-1}R^TQ^Tb \\ &= R^{-1}R^{-T}R^TQ^Tb = R^{-1}Q^Tb.\end{aligned}$$

Позднее мы покажем, что стоимость данного разложения и последующего вычисления решения x составляет $2n^2m-\frac{2}{3}n^3$ флопов, что примерно вдвое превышает стоимость решения нормальных уравнений, если $m \gg n$, и примерно равно этой стоимости в случае $m = n$.

3.2.3. Сингулярное разложение

SVD — это очень важное разложение, используемое, помимо решения задач наименьших квадратов, для многих других целей.

Теорема 3.2. (SVD). *Пусть A — произвольная $m\times n$-матрица, причем $m \geq n$. Тогда справедливо представление $A = U\Sigma V^T$, где матрица U имеет размер $m \times n$ и удовлетворяет соотношению $U^TU = I$, матрица V — квадратная порядка n и удовлетворяет соотношению $V^TV = I$, а $\Sigma = \mathrm{diag}(\sigma_1,\dots,\sigma_n)$, где $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. Столбцы $u_1,\dots,u_n$ матрицы U называются* левыми сингулярными векторами *(матрицы A). Столбцы $v_1,\dots,v_n$ матрицы V называются* правыми сингулярными векторами. *Величины σ_i называются* сингулярными числами. *(При $m < n$ нужно рассматривать SVD матрицы A^T.)*

Эта теорема допускает следующую геометрическую интерпретацию. Будем рассматривать $m \times n$-матрицу A как линейный оператор, отображающий вектор $x \in \mathbb{R}^n$ в вектор $y = Ax \in \mathbb{R}^m$. Тогда можно выбрать ортогональную координатную систему для $\mathbb{R}^n$ (где ортами осей являются столбцы матрицы V) и другую ортогональную координатную систему для $\mathbb{R}^m$ (со столбцами матрицы U в качестве ортов осей) таким образом, чтобы A приобрела диагональный вид (Σ), т. е. вектор $x = \sum_{i=1}^n \beta_i v_i$ отображался бы в $y = Ax = \sum_{i=1}^n \sigma_i\beta_i u_i$. Иначе говоря, всякая матрица станет диагональной, если выбрать подходящие ортогональные координатные системы в ее области определения и области значений.

Доказательство теоремы 3.2. Применим индукцию по m и n: предположим, что SVD существует для матриц размера $(m-n)\times(n-1)$, и докажем его существование для $m \times n$-матриц. Будем считать, что $A \neq 0$; в противном случае,

можно положить $\Sigma = 0$ и взять в качестве U и V произвольные ортогональные матрицы.

В качестве базиса индукции рассмотрим случай $n = 1$ (поскольку $m \geq n$). В этом случае положим $A = U\Sigma U^T$, где $U = A/\|A\|_2$, $\Sigma = \|A\|_2$ и $V = 1$.

Для индуктивного перехода найдем вектор v, такой, что $\|v\|_2 = 1$ и $\|A\|_2 = \|Av\|_2 > 0$. Такой вектор существует в силу определения $\|A\|_2 = \max_{\|v\|_2=1} \|Av\|_2$. Положим $u = \frac{Av}{\|Av\|_2}$; этот вектор имеет единичную длину. Выберем $\tilde{U}$ и $\tilde{V}$ таким образом, чтобы $U = [u, \tilde{U}]$ была ортогональной $m \times m$-матрицей, а $V = [v, \tilde{V}]$ — ортогональной $n \times n$-матрицей. Теперь имеем

$$U^T A V = \begin{bmatrix} u^T \\ \tilde{U}^T \end{bmatrix} \cdot A \cdot [v \quad \tilde{V}] = \begin{bmatrix} u^T Av & u^T A\tilde{V} \\ \tilde{U}^T Av & \tilde{U}^T A\tilde{V} \end{bmatrix}.$$

Тогда

$$u^T Av = \frac{(Av)^T(Av)}{\|Av\|_2} = \frac{\|Av\|_2^2}{\|Av\|_2} = \|Av\|_2 = \|A\|_2 \equiv \sigma$$

и $\tilde{U}^T v = \tilde{U}^T u$ и $\|Av\|_2 = 0$. Мы утверждаем, что и $u^T A\tilde{V} = 0$. Иначе мы имели бы $\sigma = \|A\|_2 = \|U^T AV\|_2 \geq \|[1, 0, \ldots, 0]U^T AV\|_2 = \|[\sigma | u^T A\tilde{V}]\|_2 > \sigma$, что невозможно. (Здесь было использовано утверждение 7 леммы 1.7.)

Итак, $U^T AV = \begin{bmatrix} \sigma & 0 \\ 0 & \tilde{U}^T A\tilde{V} \end{bmatrix} = \begin{bmatrix} \sigma & 0 \\ 0 & \tilde{A} \end{bmatrix}$. Применяя индуктивное предположение к матрице $\tilde{A}$, имеем $\tilde{A} = U_1 \Sigma_1 V_1^T$, где U_1, Σ_1 и V_1 — соответственно матрицы размера $(m-1) \times (n-1)$, $(n-1) \times (n-1)$ и $(n-1) \times (n-1)$. Отсюда

$$U^T AV = \begin{bmatrix} \sigma & 0 \\ 0 & U_1 \Sigma_1 V_1^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix} \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix}^T$$

или

$$A = \left(U \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix} \right) \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma_1 \end{bmatrix} \left(V \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix} \right)^T.$$

Это и есть требуемое разложение. □

SVD имеет множество важных алгебраических и геометрических свойств. Наиболее важные из них мы сейчас сформулируем.

Теорема 3.3. *Пусть $A = U\Sigma V^T$ есть SVD $m \times n$-матрицы A, причем $m \geq n$. (Аналогичные факты справедливы для $m < n$.)*

1. *Пусть A — симметричная матрица с собственными значениями λ_i и ортонормированными собственными векторами u_i. Другими словами, $A = U\Lambda U^T$ есть спектральное разложение матрицы A; здесь $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, $U = [u_1, \ldots, u_n]$ и $UU^T = I$. Тогда SVD матрицы A имеет вид $A = U\Sigma V^T$, где $\sigma_i = |\lambda_i|$ и $v_i = \mathrm{sgn}(\lambda_i) u_i$, причем $\mathrm{sgn}(0) = 1$.*
2. *Собственными значениями симметричной матрицы $A^T A$ являются числа σ_i^2. Правые сингулярные векторы v_i суть соответствующие ортонормированные собственные векторы.*

3. *Собственными значениями симметричной матрицы* AA^T *являются числа* σ_i^2 *и* $m-n$ *нулей. Левые сингулярные векторы* u_i *суть ортонормированные собственные векторы, отвечающие собственным значениям* σ_i^2. *Дополнительные* $m-n$ *ортогональных векторов можно взять в качестве собственных векторов для собственного значения 0.*
4. *Пусть* $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, *где* A *— квадратная матрица, имеющая SVD* $A = U\Sigma V^T$. *Положим* $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_n)$, $U = [u_1, \dots, u_n]$ *и* $V = [v_1, \dots, v_n]$. *Тогда* $2n$ *собственных значений матрицы* H *— это числа* $\pm\sigma_i$, *а соответствующие нормированные собственные векторы имеют вид* $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$.
5. *Если* A *— матрица полного ранга, то решением задачи* $\min_x \|Ax-b\|_2$ *является вектор* $x = V\Sigma^{-1}U^T b$.
6. $\|A\|_2 = \sigma_1$. *Если* A *— квадратная и невырожденная матрица, то* $\|A^{-1}\|_2^{-1} = \sigma_n$ *и* $\|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}$.
7. *Предположим, что* $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0$. *Тогда ранг матрицы* A *равен* r. *Ее ядро, т. е. подпространство векторов* v, *таких, что* $Av = 0$, *натянуто на столбцы матрицы* V *с номерами от* $r+1$ *до* n, *т. е. является подпространством* $\operatorname{span}(v_{r+1}, \dots, v_n)$. *Образ матрицы* A, *т. е. подпространство векторов вида* Aw *для всех* w, *натянут на столбцы матрицы* U *с номерами* $1, \dots, r$, *т. е. является подпространством* $\operatorname{span}(u_1, \dots, u_r)$.
8. *Пусть* S^{n-1} *— единичная сфера пространства* $\mathbb{R}^n$: $S^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. *Пусть* $A \cdot S^{n-1}$ *есть образ сферы* S^{n-1} *под действием матрицы* A: $A \cdot S^{n-1} = \{Ax : x \in \mathbb{R}^n \text{ и } \|x\|_2 = 1\}$. *Тогда* $A \cdot S^{n-1}$ *является эллипсоидом с центром в нуле пространства* $\mathbb{R}^m$ *и с главными осями* $\sigma_i u_i$.
9. *Положим* $V = [v_1, v_2, \dots, v_n]$ *и* $U = [u_1, u_2, \dots u_n]$, *так что* $A = U\Sigma V^T = \sum_{i=1}^n \sigma_i u_i v_i^T$ *(сумма матриц ранга 1). Тогда ближайшей к* A *(в смысле нормы* $\|\cdot\|_2$*) матрицей ранга* $k < n$ *является матрица* $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$, *причем* $\|A - A_k\|_2 = \sigma_{k+1}$. *Для матрицы* A_k *справедливо также представление* $A_k = U\Sigma_k V^T$, *где* $\Sigma_k = \operatorname{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0)$.

Доказательство.

1. Это утверждение вытекает из определения SVD.
2. $A^TA = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$. Это равенство представляет собой спектральное разложение матрицы A^TA, причем собственными векторами являются столбцы матрицы V, а собственными значениями — диагональные элементы матрицы Σ^2.
3. Выберем матрицу $\tilde{U}$ размера $m \times (m-n)$ так, чтобы квадратная матрица $[U, \tilde{U}]$ была ортогональной. Тогда можем написать

$$AA^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T = [U, \tilde{U}] \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} [U, \tilde{U}]^T.$$

Это спектральное разложение матрицы AA^T.
4. См. вопрос 3.14.

5. $\|Ax-b\|_2^2 = \|U\Sigma V^T x - b\|_2^2$. Поскольку A — матрица полного ранга, то же самое верно для Σ, т. е. Σ обратима. Пусть, как и выше, $[U, \tilde{U}]$ — квадратная ортогональная матрица. Тогда

$$\begin{aligned}\|U\Sigma V^T x - b\|_2^2 &= \left\| \begin{bmatrix} U^T \\ \tilde{U}^T \end{bmatrix} (U\Sigma V^T x - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma V^T x - U^T b \\ -\tilde{U}^T b \end{bmatrix} \right\|_2^2 \\ &= \|\Sigma V^T x - U^T b\|_2^2 + \|\tilde{U}^T b\|_2^2.\end{aligned}$$

Эта сумма минимальна, когда первое ее слагаемое равно нулю, т. е. при $x = V\Sigma^{-1}U^T b$.

6. Из определения очевидно, что 2-норма диагональной матрицы равна наибольшему из модулей диагональных элементов. Используя утверждение 3 леммы 1.7, имеем $\|A\|_2 = \|U^T A V\|_2 = \|\Sigma\|_2 = \sigma_1$ и $\|A^{-1}\|_2 = \|V^T A^{-1} U\|_2 = \|\Sigma^{-1}\|_2 = \sigma_n^{-1}$.

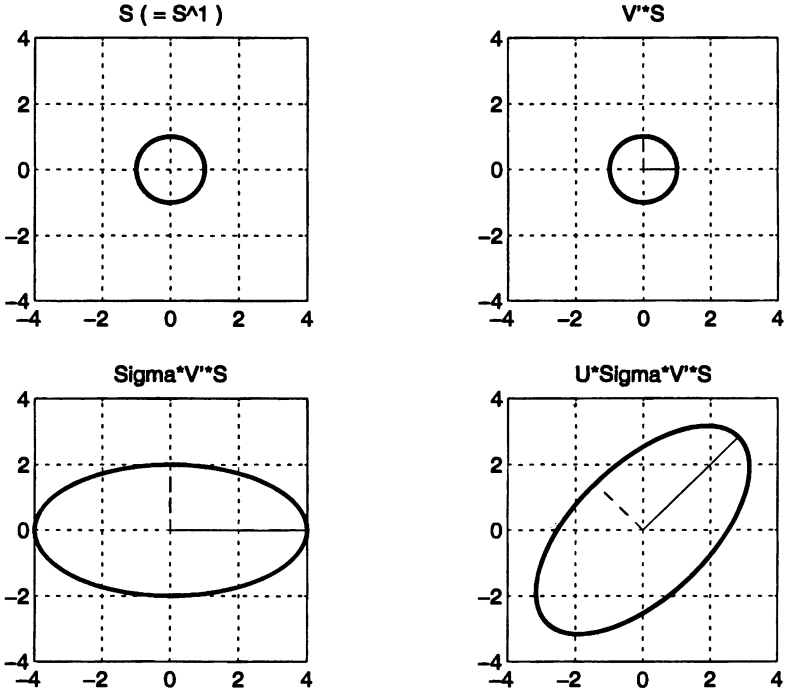
7. Снова выберем $m \times (m-n)$-матрицу $\tilde{U}$ таким образом, чтобы $m \times m$-матрица $\hat{U} = [U, \tilde{U}]$ была ортогональной. Поскольку $\hat{U}$ и V — невырожденные матрицы, A имеет тот же ранг, что и матрица $\hat{U}^T A V = \begin{bmatrix} \Sigma^{n\times n} \\ 0^{(m-n)\times n} \end{bmatrix} \equiv \hat{\Sigma}$, т. е. в силу нашего предположения о Σ, ранг r. Так как равенство $Av = 0$ эквивалентно равенству $\hat{U}^T A V (V^T v) = 0$, то вектор v тогда и только тогда принадлежит ядру матрицы A, когда $V^T v$ принадлежит ядру матрицы $\hat{U}^T A V = \hat{\Sigma}$. Очевидно, что ядро матрицы $\hat{\Sigma}$ натянуто на столбцы единичной матрицы I_n с номерами от $r+1$ до n, поэтому произведения этих столбцов с матрицей V, т. е. векторы $v_{r+1}, \ldots, v_n$, составляют базис в ядре матрицы A. Аналогичное рассуждение показывает, что образ матрицы A получается умножением матрицы $\hat{U}$ на образ матрицы $\hat{U}^T A V = \hat{\Sigma}$, т. е. умножением $\hat{U}$ на первые r столбцов матрицы I_m. Эти произведения суть векторы $u_1, \ldots u_r$.

8. «Построим» множество $AS \cdot S^{n-1}$, применяя к S^{n-1} поочередно сомножители разложения $A = U\Sigma V^T$. На рисунке этот процесс проиллюстрирован для матрицы

$$\begin{aligned} A &= \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 2^{-1/2} & -2^{-1/2} \\ 2^{-1/2} & 2^{-1/2} \end{bmatrix} \cdot \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2^{-1/2} & -2^{-1/2} \\ 2^{-1/2} & 2^{-1/2} \end{bmatrix}^T \\ &\equiv U\Sigma V^T. \end{aligned}$$

Для простоты, предположим, что A — квадратная невырожденная матрица. Так как матрица V ортогональна, т. е. отображает одни нормированные векторы в другие, то $V^T \cdot S^{n-1} = S^{n-1}$. Далее, включение $v \in S^{n-1}$ равносильно условию $\|v\|_2 = 1$, поэтому $w \in \Sigma S^{n-1}$ в том и только в том случае, если $\|\Sigma^{-1} w\|_2 = 1$, или $\sum_{i=1}^n (w_i/\sigma_i)^2 = 1$. Это соотношение определяет эллипсоид с главными осями $\sigma_i e_i$, где e_i есть i-й столбец единичной

матрицы. Наконец, умножение всех векторов $w = \Sigma v$ на матрицу U попросту поворачивает эллипс. При этом векторы e_i переходят в столбцы u_i матрицы U.



9. По построению, матрица A_k имеет ранг k и

$$\|A - A_k\|_2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\| = \left\| U \begin{bmatrix} 0 & & \\ & \sigma_{k+1} & \\ & & \ddots \\ & & & \sigma_n \end{bmatrix} V^T \right\|_2 = \sigma_{k+1}.$$

Остается показать, что не существует матрицы ранга k, более близкой к A. Пусть B — произвольная матрица ранга k, тогда ее ядро имеет размерность $n - k$. Подпространство, натянутое на векторы $v_1, \ldots, v_{k+1}$, имеет размерность $k + 1$. Поскольку для суммы размерностей выполняется неравенство $(n - k) + (k + 1) > n$, указанные подпространства должны иметь нетривиальное пересечение. Пусть h — нормированный вектор из этого пересечения. Тогда

$$\begin{aligned} \|A - B\|_2^2 &\geq \|(A - B)h\|_2^2 = \|Ah\|_2^2 = \|U\Sigma V^T h\|_2^2 \\ &= \|\Sigma(V^T h)\|_2^2 \\ &\geq \sigma_{k+1}^2 \|V^T h\|_2^2 \\ &= \sigma_{k+1}^2. \end{aligned}$$

□

Пример 3.4. Проиллюстрируем последнюю часть теоремы 3.3, использовав ее для *сжатия изображения*. Такой иллюстрацией послужат малоранговые

аппроксимации изображения клоуна. Изображение размера $m \times n$ — это попросту $m \times n$-матрица, элемент (i, j) которой интерпретируется как яркость точки (пиксела) (i, j). Другими словами, элементы матрицы, изменяющиеся, скажем, от 0 до 1, интерпретируются как точки с окраской от черной (что соответствует 0) до белой (соответствует 1) с различными промежуточными степенями серого цвета. (Возможна и цветная окраска.) Вместо того чтобы хранить или передавать все $m \cdot n$ элементов матрицы, представляющей изображение, часто бывает предпочтительным *сжатие* этого изображения, т. е. хранение гораздо меньшего массива чисел, с помощью которых исходный образ все же может быть приближенно восстановлен. Мы покажем сейчас,что для этой цели можно применить утверждение 9 теоремы 3.3.

Рассмотрим изображение на рис. 3.3(а). Это изображение размера 320×200 соответствует матрице A того же размера. Пусть $A = U\Sigma V^T$ есть SVD этой матрицы. Согласно утверждению 9 теоремы 3.3, матрица $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ есть наилучшее приближение ранга k матрицы A в том смысле, что она реализует минимум σ_{k+1} величины $\|A - A_k\|_2$. Заметим, что для восстановления матрицы A_k требуются лишь $m \cdot k + n \cdot k = (m+n) \cdot k$ слов памяти, в которых хранятся векторы $u_1, \ldots, u_k$ и $\sigma_1 v_1, \ldots, \sigma_k v_k$. В то же время для хранения A (или той же матрицы A_k, но в явном виде) нужны $m \cdot n$ слов, т. е. гораздо большая память при малом k. Итак, будем рассматривать A_k как сжатое изображение, хранимое с помощью $(m + n) \cdot k$ слов памяти. Эти приближенные изображения для различных значений k показаны на рис. 3.3; указаны также относительные ошибки σ_{k+1}/σ_1 и коэффициенты сжатия $(m + n) \cdot k/(m \cdot n) = 520 \cdot k/64000 \approx k/123$.

k	Относительная ошибка $= \sigma_{k+1}/\sigma_1$	Коэффициент сжатия $= 520k/64000$
3	.155	.024
10	.077	.081
20	.040	.163

Эти картинки были получены посредством следующих команд (изображение клоуна и некоторые другие изображения можно создать в Matlab'e, пользуясь демонстрационными файлами визуализации; выясните, где находятся эти файлы в вашей локальной инстоляции Matlab'a):

```
load clown.mat; [U,S,V]=svd(X); colormap ('gray');
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)')
```

Имеется множество других, более дешевых, чем SVD, методов сжатия изображений [189, 152]. ◇

Позднее мы увидим, что при $m \gg n$ стоимость решения задачи наименьших квадратов с использованием SVD примерно такая же, как в методе QR; для меньших m она составляет приблизительно $4n^2m - \frac{4}{3}n^3 + O(n^2)$ операций. Точное сравнение стоимостей методов QR и SVD зависит, кроме того, от используемого компьютера. По поводу деталей см. раздел 3.6.

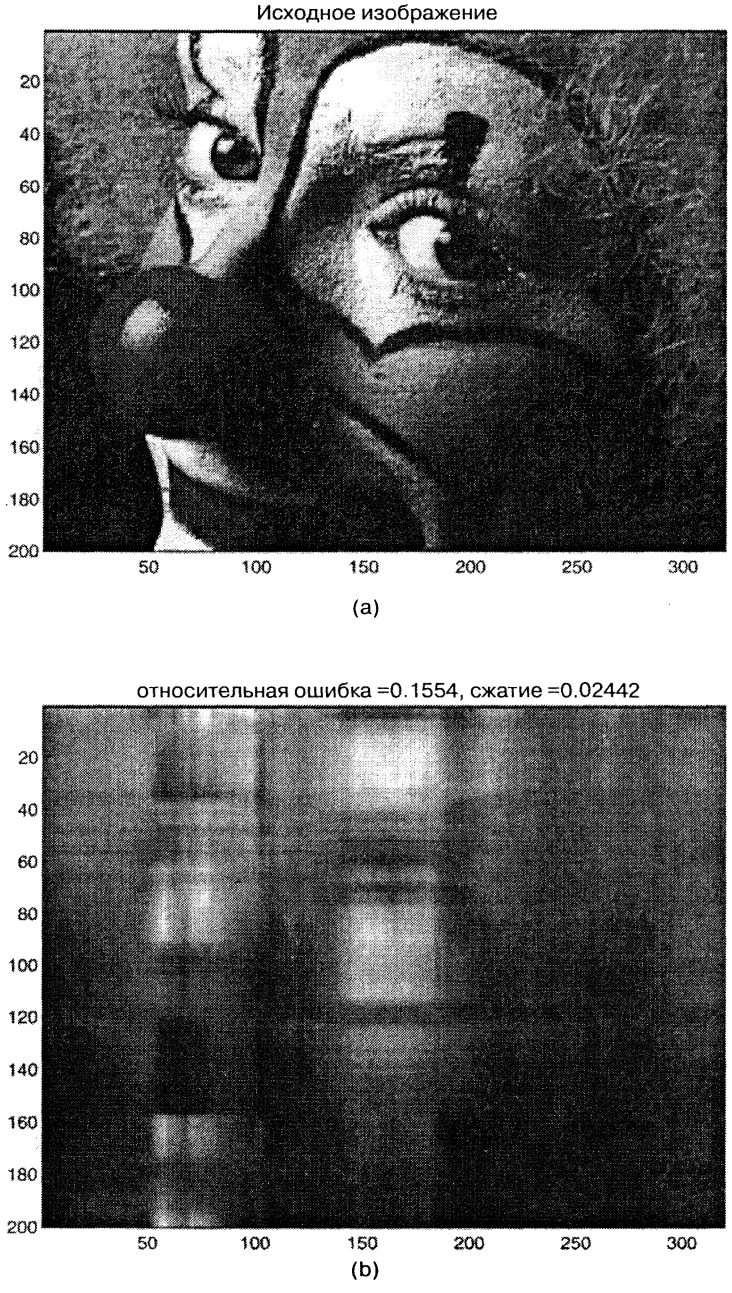
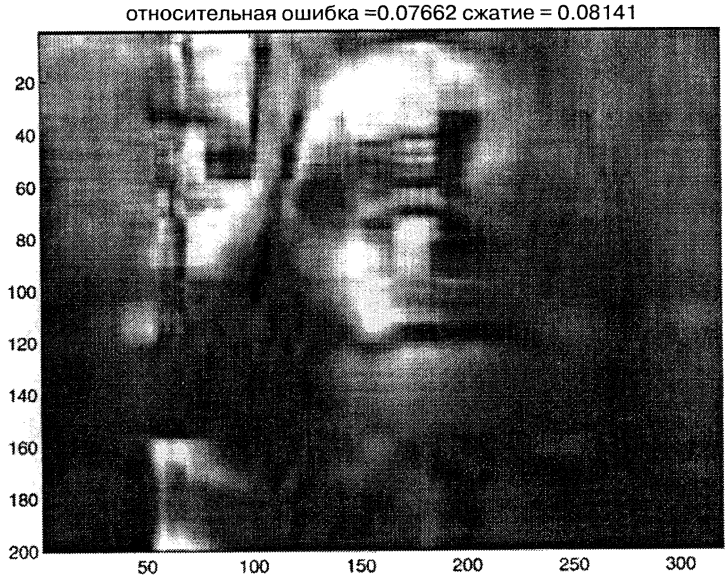
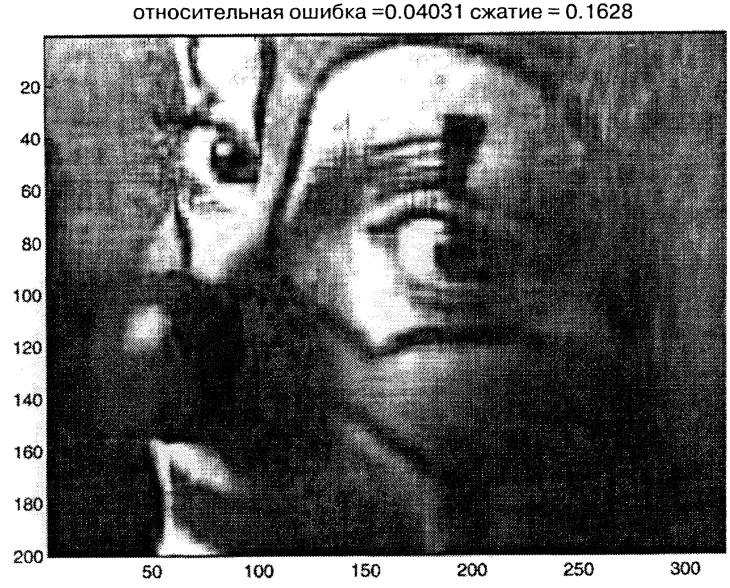


Рис. 3.3. Сжатие изображения посредством SVD. (a) Исходное изображение. (b) Приближение ранга $k = 3$.



(c)



(d)

Рис. 3.3. *Продолжение.* (c) Приближение ранга $k = 10$. (d) Приближение ранга $k = 20$.

Определение 3.1. *Пусть A — матрица размера $m \times n$, причем $m \geq n$. Пусть A — матрица полного ранга и $A = QR = U\Sigma V^T$ суть ее QR-разложение и SVD. Матрица*

$$A^+ \equiv (A^TA)^{-1}A^T = R^{-1}Q^T = V\Sigma^{-1}U^T$$

называется псевдообратной (Мура—Пенроуза) для A. *При $m < n$ полагаем $A^+ \equiv A^T(AA^T)^{-1}$.*

Псевдообратная матрица позволяет записать решение полноранговой переопределенной задачи наименьших квадратов простой формулой $x = A^+b$. Если A — квадратная невырожденная матрица, то эта формула, как и следовало ожидать, принимает вид $x = A^{-1}b$. В Matlab'е псевдообратная для матрицы A вычисляется посредством функции `pinv(A)`. Для A, не имеющей полного ранга, псевдообратная матрица вводится определением 3.2 в разд. 3.5.

3.3. Теория возмущений для задачи наименьших квадратов

Для неквадратной матрицы A число обусловленности в смысле 2-нормы определяется как $\kappa_2(A) \equiv \sigma_{\max}(A)/\sigma_{\min}(A)$. Это определение в случае квадратной матрицы A сводится к обычному. Следующая теорема обосновывает это новое определение.

Теорема 3.4. *Пусть A — полноранговая $m \times n$-матрица, где $m \geq n$. Предположим, что вектор x минимизирует $\|Ax - b\|_2$ и $r = Ax - b$ есть соответствующая невязка. Пусть вектор $\tilde{x}$ минимизирует $\|(A + \delta A)\tilde{x} - (b + \delta b)\|_2$. Предположим, что $\varepsilon \equiv \max\left(\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}\right) < \frac{1}{\kappa_2(A)} = \frac{\sigma_{\min}(A)}{\sigma_{\max}(A)}$. Тогда*

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \varepsilon \cdot \left\{ \frac{2 \cdot \kappa_2(A)}{\cos\theta} + \operatorname{tg}\theta \cdot \kappa_2^2(A) \right\} + O(\varepsilon^2) \equiv \varepsilon \cdot \kappa_{LS} + O(\varepsilon^2),$$

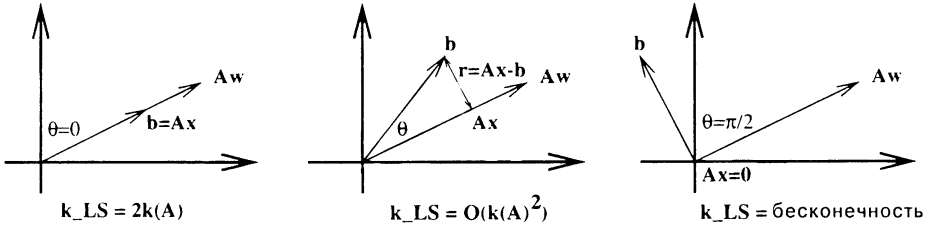
где $\sin\theta = \frac{\|r\|_2}{\|b\|_2}$. Другими словами, θ есть угол между векторами b и Ax; он измеряет, велика или мала норма невязки $\|r\|_2$ (т. е., соответственно, $\|r\|_2$ близка к $\|b\|_2$ или 0). Символ κ_{LS} обозначает число обусловленности для задачи наименьших квадратов.

Набросок доказательства. Разложим вектор $\tilde{x} = ((A + \delta A)^T(A + \delta A))^{-1}(A + \delta A)^T(b + \delta b)$ по степеням величин δA и δb, а затем отбросим все, кроме членов, линейных по δA и δb. □

Условие $\varepsilon\kappa_2(A) < 1$ играет ту же роль, что и при выводе оценки (2.4) для возмущенного решения квадратной системы линейных уравнений $Ax = b$: оно гарантирует, что матрица $A + \delta A$ имеет полный ранг, поэтому вектор $\tilde{x}$ определен однозначно.

Полученную оценку можно интерпретировать следующим образом: если угол θ очень мал или равен нулю, то мала и невязка. В этом случае эффективное число обусловленности равно примерно $2\kappa_2(A)$, т. е. примерно тому же, что и при решении обычной системы линейных уравнений. Если θ не мал, но и не близок к $\pi/2$, то невязка умеренно велика, и эффективное число обусловленности может быть много больше, чем в первом случае, а именно равно

$\kappa_2^2(A)$. Если θ близок к $\pi/2$, так что точное решение почти вырождается, то эффективное число обусловленности становится неограниченным даже при малом $\kappa_2(A)$. Эти три случая иллюстрируются рисунком. С помощью его правой части легко понять, почему число обусловленности бесконечно при $\theta = \pi/2$: в этом случае имеем $x = 0$ и почти любое, как угодно малое изменение в A или b приведет к ненулевому решению $\tilde{x}$, что является «бесконечно» большим относительным изменением.



Оценке из теоремы 3.4 можно придать другую форму, в которой отсутствует член с $O(\varepsilon^2)$ [258, 149]:

$$\frac{\|\tilde{x}-x\|_2}{\|x\|_2} \le \frac{\varepsilon\kappa_2(A)}{1-\varepsilon\kappa_2(A)}\left(2+(\kappa_2(A)+1)\frac{\|r\|_2}{\|A\|_2\|x\|_2}\right), \quad \frac{\|\tilde{r}-r\|_2}{\|r\|_2} \le 1+2\varepsilon\kappa_2(A).$$

Здесь $\tilde{r}$ — невязка возмущенной задачи: $\tilde{r} = (A+\delta A)\tilde{x} - (b+\delta b)$.

Мы увидим, что при правильной реализации оба метода — и QR, и SVD, — численно устойчивы, т. е. дают приближенное решение $\tilde{x}$, которое (точно) минимизирует некоторую функцию вида $\|(A+\delta A)\tilde{x} - (b+\delta b)\|_2$, причем

$$\max\left(\frac{\|\delta A\|}{\|A\|}, \frac{\|\delta b\|}{\|b\|}\right) = O(\varepsilon).$$

Комбинируя этот результат с приведенными выше оценками возмущений, можем получить оценки для ошибки в решении задачи наименьших квадратов подобно тому, как были получены оценки для ошибки в решении системы линейных уравнений.

Метод нормальных уравнений не столь точен. Поскольку решается система $(A^TA)x = A^Tb$, точность определяется числом обусловленности $\kappa_2(A^TA) = \kappa_2^2(A)$. Таким образом, ошибка всегда ограничена величиной типа $\kappa_2^2(A)\varepsilon$, а не просто $\kappa_2(A)\varepsilon$. Следует ожидать поэтому, что при решении нормальных уравнений может быть потеряно вдвое больше верных разрядов, чем в методах, основанных на QR-разложении и SVD.

Кроме того, метод нормальных уравнений *не* всегда устойчив, т. е. вычисленное решение $\tilde{x}$, вообще говоря, не является точкой минимума функции вида $\|(A+\delta A)\tilde{x} - (b+\delta b)\|_2$ с малыми δA и δb. Все же, если число обусловленности мало, метод, скорей всего, даст приближенное решение примерно того же качества, что и QR-разложение и SVD. Поскольку задача наименьших квадратов быстрей всего решается именно посредством нормальных уравнений, этот метод является предпочтительным в случае хорошо обусловленной матрицы A.

В разделе 3.5 мы вернемся к вопросу о том, как следует решать очень плохо обусловленные задачи наименьших квадратов.

3.4. Ортогональные матрицы

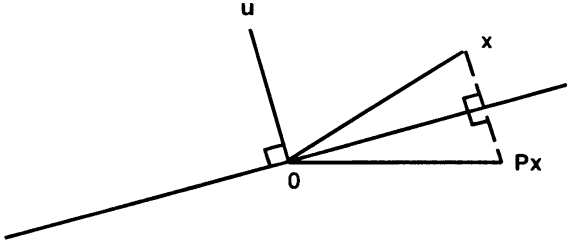
Как было отмечено в разд. 3.2.2, процесс ортогонализации Грама–Шмидта (алгоритм 3.1), примененный к почти линейно зависимым векторам, может дать матрицу Q, столбцы которой не ортогональны; таким образом, с его помощью нельзя вычислить QR-разложение устойчиво.

Поэтому наши алгоритмы будут основаны на использовании некоторых легко вычисляемых ортогональных матриц, называемых *отражениями Хаусхолдера* и *вращениями Гивенса*. Подходящим выбором таких матриц можно получать нулевые компоненты в их векторных сомножителях. Позднее мы покажем, что всякий метод, использующий эти матрицы для исключения элементов в матрице задачи, автоматически является устойчивым. Этот анализ ошибок будет приложи́м к нашим алгоритмам QR-разложения, а также ко многим алгоритмам вычисления собственных значений и SVD в гл. 4 и 5.

Несмотря на отмеченную возможность вычисления матрицы Q с неортогональными столбцами, алгоритм MGS имеет важные применения в численной линейной алгебре. (Напротив, его менее устойчивая версия CGS малополезна.) Среди них вычисление собственных векторов симметричных трехдиагональных матриц с использованием бисекции и обратной итерации (разд. 5.3.4), а также алгоритмы Арнольди и Ланцоша для приведения матрицы к некоторым «конденсированным» формам (разд. 6.6.1, 6.6.6 и 7.4). Алгоритмы Арнольди и Ланцоша используются в качестве основы алгоритмов для решения разреженных линейных систем и вычисления собственных значений разреженных матриц. Алгоритм MGS можно модифицировать таким образом, чтобы задача наименьших квадратов решалась устойчиво, однако столбцы матрицы Q по-прежнему могут сильно отклоняться от ортогональности [33].

3.4.1. Преобразования Хаусхолдера

Преобразованием Хаусхолдера (или отражением) называется матрица вида $P = I - 2uu^T$, где $\|u\|_2 = 1$. Легко видеть, что $P = P^T$ и $PP^T = (I - 2uu^T)(I - 2uu^T) = I - 4uu^T + 4uu^Tuu^T = I$, т. е. матрица P симметрична и ортогональна. Она называется отражением, потому что вектор Px является отражением вектора x относительно плоскости, проходящей через 0 перпендикулярно к u.



Пусть дан вектор x. Тогда легко найти отражение $P = I - 2uu^T$, аннулирующее в векторе x все компоненты, кроме первой: $Px = [c, 0, \ldots, 0]^T = c \cdot e_1$. Это можно сделать следующим образом. Имеем $Px = x - 2u(u^Tx) = c \cdot e_1$, поэтому $u = \frac{1}{2(u^Tx)}(x - ce_1)$, т. е. u есть линейная комбинация векторов x и e_1. Так как $\|x\|_2 = \|Px\|_2 = |c|$, то u должен быть параллелен вектору $\tilde{u} = x \pm \|x\|_2 e_1$,

откуда $u = \tilde{u}/\|\tilde{u}\|_2$. Можно проверить, что при любом выборе знака получится вектор u, удовлетворяющий соотношению $Px = e_1$, если $\tilde{u} \neq 0$. Мы будем пользоваться формулой $\tilde{=}x + \operatorname{sgn}(x_1)e_1$, так как в этом случае не будет взаимного сокращения при вычислении первой компоненты в $\tilde{u}$. Итак, имеем

$$\tilde{u} = \begin{bmatrix} x_1 + \operatorname{sgn}(x_1) \cdot \|x\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{и} \quad u = \frac{\tilde{u}}{\|\tilde{u}\|_2}.$$

Мы будем записывать это как $u = \text{House}\,(x)$. (На практике можно хранить $\tilde{u}$ вместо u, чтобы сэкономить на работе вычисления вектора u; в этом случае вместо $P = I - 2uu^T$ используется формула $P = I - (2/\|\tilde{u}\|_2^2)\tilde{u}\tilde{u}^T$.)

Пример 3.5. Покажем, как с помощью отражений вычисляется QR-разложение матрицы A размера 5×4. Из этого примера будет ясно, как следует действовать для произвольных $m \times n$-матриц. В приводимых ниже выкладках P_i суть ортогональные 5×5-матрицы, x означает произвольный ненулевой элемент, а символ o используется для нулевых элементов.

1. Выбрать матрицу P_1 так, чтобы $\quad A_1 \equiv P_1 A = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix}$.

2. Выбрать матрицу $P_2 = \begin{bmatrix} 1 & 0 \\ 0 & P_2' \end{bmatrix}$ так, чтобы $\quad A_2 \equiv P_2 A_1 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix}$.

3. Выбрать матрицу $P_3 = \begin{bmatrix} 1 & & 0 \\ & 1 & \\ 0 & & P_3' \end{bmatrix}$ так, чтобы $\quad A_3 \equiv P_3 A_2 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix}$.

4. Выбрать матрицу $P_4 = \begin{bmatrix} 1 & & & \\ & 1 & & 0 \\ & & 1 & \\ & 0 & & P_4' \end{bmatrix}$ так, чтобы $\quad A_4 \equiv P_4 A_3 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & o \end{bmatrix}$.

Здесь отражение P_i' выбиралось таким образом, чтобы аннулировать поддиагональные элементы столбца i. Это не изменяет нулей, уже введенных в предыдущие столбцы.

Обозначим полученную верхнетреугольную 5×4-матрицу через $\tilde{R} \equiv A_4$. Тогда $A = P_1^T P_2^T P_3^T P_4^T \tilde{R} = QR$, где Q образована первыми четырьмя столбцами матрицы $P_1^T P_2^T P_3^T P_4^T = P_1 P_2 P_3 P_4$ (так как все матрицы P_i симметричны), а R состоит из первых четырех строк матрицы $\tilde{R}$. ◇

Приведем общий алгоритм QR-разложения, основанный на использовании отражений.

Алгоритм 3.2. *QR-разложение посредством отражений:*

for $i = 1$ *to* $\min(m-1, n)$
  $u_i = \text{House}(A(i:m, i))$
  $P_i' = I - 2u_i u_i^T$
  $A(i:m, i:n) = P_i' A(i:m, i:n)$
end for

Обсудим некоторые детали реализации метода. Матрицы P_i нигде не требуется формировать в явном виде, так как умножение на них производится в соответствии с более экономичной формулой

$$(I - 2u_i u_i^T)A(i:m, i:n) = A(i:m, i:n) - 2u_i(u_i^T A(i:m, i:n)).$$

Для хранения матрицы P_i достаточно запомнить лишь вектор u_i или же $\tilde{u}_i$ и число $\|\tilde{u}_i\|$. Эта информация может храниться в столбце i матрицы A; в действительности, в этом столбце ничего не нужно менять! Таким образом, QR-разложение может быть записано на место матрицы A, причем Q хранится в факторизованной форме $P_1 \dots P_{n-1}$, а P_i хранится в виде вектора $\tilde{u}_i$ в поддиагональных позициях столбца i матрицы A. (Поскольку диагональные позиции заняты элементами r_{ii}, нужен дополнительный массив длины n для хранения первых элементов векторов $\tilde{u}_i$.)

Вспомним, что для решения задачи наименьших квадратов $\min\|Ax - b\|_2$ с помощью QR-разложения $A = QR$ нужно вычислить вектор $Q^T b$. Это делается следующим образом: $Q^T b = P_n P_{n-1} \dots P_1 b$, поэтому b нужно последовательно умножать на $P_1, P_2, \dots P_n$:

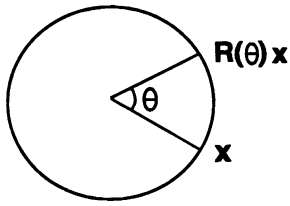
for $i = 1$ to n
  $\gamma = -2 \cdot u_i^T b(i:m)$
  $b(i:m) = b(i:m) + \gamma u_i$
end for

Стоимость этого алгоритма — n скалярных произведений $\gamma = -2 \cdot u_i^T b$ и n операций типа saxpy $b + \gamma u_i$. Стоимость вычисления разложения $A = QR$ этим способом составляет $2n^2 m - \frac{2}{3}n^3$ флопов; цена последующего решения задачи наименьших квадратов при известном QR-разложении равна уже только $O(mn)$ флопов.

В LAPACK'е решение задачи наименьших квадратов методом QR-разложения осуществляется программой `sgels`. QR-разложение можно реорганизовать так, чтобы в нем использовались матричные умножения и другие операции уровня 3 BLAS, подобно тому, как это было сделано для гауссова исключения в разд. 2.6; см. по этому поводу вопрос 3.17. Если в Matlab'е $m \times n$-матрица A имеет больше строк, чем столбцов, а b имеет размер $m \times 1$, то оператор `A\b` решает задачу наименьших квадратов. Если требуется QR-разложение само по себе, то оно вычисляется оператором `[Q,R]=qr(A)`.

3.4.2. Вращения Гивенса

Вращение Гивенса (или просто вращение) $R(\theta) \equiv \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ поворачивает всякий вектор $x \in \mathbb{R}^2$ на угол θ против часовой стрелки (см. рисунок).



Нам потребуется также определить вращение на угол θ в координатной плоскости (i,j) пространства $\mathbb{R}^n$:

$$R(i,j,\theta) \equiv \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \end{array} \\ \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & \cos\theta & & -\sin\theta & \\ & & & & \ddots & & \\ & & & \sin\theta & & \cos\theta & \\ & & & & & & \ddots \\ & & & & & & & 1 \\ & & & & & & & & 1 \end{bmatrix} \end{array}.$$

При заданных i и j и векторе x можно аннулировать компоненту x_j, выбирая $\cos\theta$ и $\sin\theta$ таким образом, чтобы

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix},$$

т. е. $\cos\theta = \frac{x_i}{\sqrt{x_i^2+x_j^2}}$ и $\sin\theta = -\frac{x_j}{\sqrt{x_i^2+x_j^2}}$.

Алгоритм QR-разложения, основанный на вращениях, аналогичен алгоритму, использующему отражения, однако при обработке столбца i элементы в нем аннулируются по одному за шаг (в направлении, скажем, снизу вверх).

Пример 3.6. Проиллюстрируем два промежуточных шага в вычислении QR-разложения матрицы 5×4 посредством вращений. Чтобы перейти от

$$\begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} \quad \text{к} \quad \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix},$$

производим умножения

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c & -s \\ & & & s & c \end{bmatrix} \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & o & x \end{bmatrix}$$

и

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c' & -s' & \\ & & s' & c' & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & o & x \end{bmatrix} = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix}. \qquad \diamond$$

Стоимость QR-разложения с использованием вращений вдвое превышает стоимость метода отражений. В дальнейшем вращения понадобятся нам и для других целей.

Остановимся на некоторых деталях реализации. Как и в методе отражений, мы можем записать сомножители QR-разложения на место матрицы A. Используется тот же прием, т. е. информация, описывающая преобразования, помещается на места аннулируемых элементов. Поскольку вращение аннулирует лишь один элемент, для хранения информации о нем имеется только одно машинное слово. Оно используется следующим образом: пусть $s = \sin\theta$ и $c = \cos\theta$. Если $|s| < |c|$, то в память записывается число $s \cdot \operatorname{sgn}(c)$, в противном случае, число $\frac{\operatorname{sgn}(s)}{c}$. Обозначим хранимое значение через p. Числа s и c восстанавливаются из него так: если $|p| < 1$, то полагаем $s = p$ и $c = \sqrt{1-s^2}$; в противном случае, $c = \frac{1}{p}$ и $s = \sqrt{1-c^2}$. Если всегда запоминать значение s, а затем вычислять $c = \sqrt{1-s^2}$, то при s, близком к 1, значение c будет получено с большой погрешностью. Отметим еще, что вместо исходной пары c, s мы можем восстановить пару $-c$, $-s$; однако для наших целей обе пары эквивалентны.

Имеется возможность выполнения последовательности вращений с затратой меньшего числа флопов, чем в описанном выше процессе. Она состоит в использовании так называемых *быстрых вращений* [7, 8, 33]. Поскольку при вычислении QR-разложения быстрые вращения все же медленней отражений, мы не станем их рассматривать здесь.

3.4.3. Анализ ошибок для ортогональных матриц

Этот анализ устанавливает обратную устойчивость QR-разложения и многих алгоритмов вычисления собственных значений и сингулярных чисел, которые обсуждаются в дальнейшем.

Лемма 3.1. *Пусть P — точное отражение (или вращение), а $\tilde{P}$ — его приближение в арифметике с плавающей точкой. Тогда*

$$\mathrm{fl}(\tilde{P}A) = P(A+E), \quad \|E\|_2 = O(\varepsilon)\cdot\|A\|_2$$

и

$$\mathrm{fl}(A\tilde{P}) = (A+F)P, \quad \|F\|_2 = O(\varepsilon)\cdot\|A\|_2.$$

Набросок доказательства . Применим обычное правило $\mathrm{fl}(a\odot b) = (a\odot b)(1+\varepsilon)$ к формулам для вычисления $\tilde{P}$ и ее умножения на матрицу A. См. по этому поводу вопрос 3.16. □

В словесном описании, лемма утверждает, что умножение на отдельную ортогональную матрицу является обратно устойчивым процессом.

Теорема 3.5. *Пусть к матрице A_0 применяется последовательность ортогональных преобразований. Тогда вычисленное произведение есть точное ортогональное преобразование матрицы $A_0 + \delta A$, где $\|\delta A\|_2 = O(\varepsilon)\|A\|_2$. Иначе говоря, процесс в целом обратно устойчив:*

$$\mathrm{fl}(\tilde{P}_j \tilde{P}_{j-1} \dots \tilde{P}_1 A_0 \tilde{Q}_1 \tilde{Q}_2 \dots \tilde{Q}_j) = P_j \dots P_1 (A_0 + E) Q_1 \dots Q_j,$$

причем $\|E\|_2 = j \cdot O(\varepsilon) \cdot \|A\|_2$. Здесь, как и в лемме 3.1, $\tilde{P}_i$ и $\tilde{Q}_i$ суть ортогональные матрицы, вычисленные в арифметике с плавающей точкой, а P_i и Q_i суть точные ортогональные матрицы.

Доказательство . Положим $\overline{P}_j \equiv P_j \dots P_1$ и $\overline{Q}_j \equiv Q_1 \dots Q_j$. Мы хотим показать, что $A_j \equiv \mathrm{fl}(\tilde{P}_j A_{j-1} \tilde{Q}_j) = \overline{P}_j (A + E_j) \overline{Q}_j$ для некоторой матрицы E_j, такой, что $\|E_j\|_2 = jO(\varepsilon)\|A\|_2$. Будем рекурсивно пользоваться леммой 3.1. При $j = 0$ доказывать нечего. Предположим теперь, что утверждение теоремы верно для индекса $j - 1$. Тогда имеем

$$\begin{aligned} B &= \mathrm{fl}(\tilde{P}_j A_{j-1}) \\ &= P_j (A_{j-1} + E') \quad \text{по лемме 3.1} \\ &= P_j (\overline{P}_{j-1} (A + E_{j-1}) \overline{Q}_{j-1} + E') \quad \text{по предположению индукции} \\ &= \overline{P}_j (A + E_{j-1} + \overline{P}_{j-1}^T E' \overline{Q}_{j-1}^T) \overline{Q}_{j-1} \\ &\equiv \overline{P}_j (A + E'') \overline{Q}_{j-1}, \end{aligned}$$

где

$$\begin{aligned} \|E''\|_2 &= \|E_{j-1} + \overline{P}_{j-1}^T E' \overline{Q}_{j-1}^T\|_2 \le \|E_{j-1}\|_2 + \|\overline{P}_{j-1}^T E' \overline{Q}_{j-1}^T\|_2 \\ &= \|E_{j-1}\|_2 + \|E'\|_2 \\ &= jO(\varepsilon)\|A\|_2, \end{aligned}$$

поскольку $\|E_{j-1}\|_2 = (j-1)O(\varepsilon)\|A\|_2$ и $\|E'\|_2 = O(\varepsilon)\|A\|_2$. Таким же образом анализируется правое умножение на матрицу $\tilde{Q}_j$. □

3.4.4. Почему именно ортогональные матрицы?

Посмотрим, как вела бы себя ошибка, если бы матрицу A_0 в теореме 3.5 вместо ортогональных матриц мы должны были бы умножать на последовательность *неортогональных* матриц. Пусть X — точная неортогональная матрица, а $\tilde{X}$ — ее приближение в арифметике с плавающей точкой. Тогда, применяя обычный анализ округлений матричного умножения, имеем

$$\mathrm{fl}(\tilde{X}A) = XA + E = X(A + X^{-1}E) \equiv X(A + F),$$

где $\|E\|_2 \le O(\varepsilon)\|X\|_2 \cdot \|A\|_2$. Поэтому $\|F\|_2 \le \|X^{-1}\|_2 \cdot \|E\|_2 \le O(\varepsilon) \cdot \kappa_2(X) \cdot \|A\|_2$.

Итак, ошибка $\|E\|_2$ приобретает множитель, равный числу обусловленности $\kappa_2(X) \ge 1$. В более длинном произведении $\tilde{X}_k \dots \tilde{X}_1 A \tilde{Y}_1 \dots \tilde{Y}_k$ ошибка умножилась бы на $\Pi_i \kappa_2(X_i) \cdot \kappa_2(Y_i)$. Этот множитель минимален (и принимает значение 1) тогда и только тогда, когда все матрицы X_i и Y_i ортогональны (или являются скалярными кратными ортогональных матриц).

3.5. Задачи наименьших квадратов неполного ранга

До сих пор при минимизации функции $||Ax - b||_2$ предполагалось, что A — матрица полного ранга. Что произойдет, если ранг матрицы не полон или же A «близка» к матрице неполного ранга? Такие задачи возникают во многих реальных ситуациях, например при выделении сигнала из зашумленных данных, решении некоторых типов интегральных уравнений, цифровом восстановлении изображений вычислении, обратного преобразования Лапласа, и т. д. [141, 142]. Эти задачи очень плохо обусловлены и чтобы превратить их в хорошо обусловленные, приходится накладывать дополнительные ограничения на их решения. Превращение плохо обусловленной задачи в хорошо обусловленную путем наложения дополнительных условий на решение называется *регуляризацией*. Эта техника используется и в других областях численного анализа при возникновении плохо обусловленных задач.

Например, следующее предложение показывает, что для матрицы A, имеющей (в точной арифметике) неполный ранг, решение задачи наименьших квадратов не единственно.

Предложение 3.1. *Пусть A — матрица размера $m \times n$, причем $m \geq n$, и пусть ранг r матрицы A меньше n. Тогда множество векторов x, минимизирующих $||Ax - b||_2$, образует $(n - r)$-мерное линейное подпространство.*

Доказательство. Пусть $Az = 0$. Если x минимизирует $||Ax - b||_2$, то это же верно для вектора $x + z$. □

Вследствие ошибок в элементах матрицы A или округлений при вычислениях, у A, как правило, не будет сингулярных чисел, в точности равных нулю, но будут одно или несколько очень малых сингулярных чисел. Следующее предложение показывает, что единственное решение такой задачи, скорей всего, очень велико и обязательно является очень чувствительным к погрешностям в правой части b (см. также теорему 3.4).

Предложение 3.2. *Пусть $\sigma_{\min} = \sigma_{\min}(A)$ есть наименьшее сингулярное число матрицы A. Предположим, что $\sigma_{\min} > 0$. Тогда:*

1. *если x минимизирует $||Ax - b||_2$, то $||x||_2 \geq |u_n^T b|/\sigma_{\min}$, где u_n — последний столбец матрицы U в разложении $A = U\Sigma V^T$;*
2. *замена b на $b + \delta b$ приводит к замене x на $x + \delta x$, где $||\delta x||_2$ может достигать величины $||\delta b||_2/\sigma_{\min}$.*

Иными словами, если A близка к матрице неполного ранга (т. е. число $\sigma_{\min}$ мало), то решение x плохо обусловлено и, возможно, очень велико.

Доказательство. Для доказательства первого утверждения запишем $x = A^+ b = V\Sigma^{-1}U^T b$, откуда $||x||_2 = ||\Sigma^{-1}U^T b||_2 \geq |(\Sigma^{-1}U^T b)_n| = |u_n^T b|/\sigma_{\min}$. Чтобы доказать утверждение 2, выберем δb, параллельное вектору u_n. □

Обсуждение регуляризации начнем указанием способа регуляризовать задачу наименьших квадратов, которая имеет неполный ранг в *точной арифметике*. Пусть A — матрица размера $m \times n$ и ранга $r < n$. В $(n - r)$-мерном подпространстве решений задачи наименьших квадратов будем искать единственное решение с минимальной нормой (нормальное решение). Это решение охарактеризовано в следующем предложении.

Предложение 3.3. *Для (точно) вырожденной матрицы A векторы x, минимизирующие $||Ax-b||_2$, могут быть охарактеризованы так: пусть A имеет ранг $r < n$ и SVD $A = U\Sigma V^T$. Запишем SVD в виде*

$$A = [U_1, U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} [V_1, V_2]^T = U_1 \Sigma_1 V_1^T, \tag{3.1}$$

где Σ_1 — невырожденная $r \times r$-матрица, а каждая из матриц U_1 и V_1 состоит из r столбцов. Обозначим наименьшее ненулевое сингулярное число матрицы A через $\sigma = \sigma_{\min}(\Sigma_1)$. Тогда:

1. *все решения x описываются формулой $x = V_1\Sigma_1^{-1}U_1^T b + V_2 z$, где z — произвольный вектор;*
2. *решение x тогда и только тогда имеет минимальную 2-норму, когда $z = 0$. В этом случае $x = V_1\Sigma_1^{-1}U_1^T b$ и $||x||_2 \le ||b||_2/\sigma$;*
3. *замена b на $b + \delta b$ может изменить нормальное решение x на величину, норма которой не превосходит $||\delta b||_2/\sigma$.*

Иначе говоря, норма и число обусловленности единственного решения x с минимальной нормой (нормального решения) определяются наименьшим ненулевым сингулярным числом матрицы A.

Доказательство . Выберем $\tilde{U}$ таким образом, чтобы $m \times m$-матрица $[U, \tilde{U}] = [U_1, U_2, \tilde{U}]$ была ортогональной. Тогда

$$\begin{aligned} ||Ax-b||_2^2 &= ||[U,\tilde{U}]^T(Ax-b)||_2^2 \\ &= \left\| \begin{bmatrix} U_1^T \\ U_2^T \\ \tilde{U}^T \end{bmatrix} (U_1\Sigma_1 V_1^T x - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma_1 V_1^T x - U_1^T b \\ -U_2^T b \\ -\tilde{U}^T b \end{bmatrix} \right\|_2^2 \\ &= ||\Sigma_1 V_1^T x - U_1^T b||_2^2 + ||U_2^T b||_2^2 + ||\tilde{U}^T b||_2^2. \end{aligned}$$

1. Величина $||Ax-b||_2$ минимальна, когда $\Sigma_1 V_1^T x = U_1^T b$, или $x = V_1\Sigma_1^{-1}U_1^T b + V_2 z$, так как $V_1^T V_2 z = 0$ для любого вектора z.
2. Столбцы матриц V_1 и V_2 взаимно ортогональны. Поэтому, по теореме Пифагора, $||x||_2^2 = ||V_1\Sigma_1^{-1}U_1^T b||_2^2 + ||V_2 z||_2^2$. Эта сумма минимальна при $z = 0$.
3. Замена b на δb изменяет x на величину, оцениваемую как $||V_1\Sigma_1^{-1}U_1^T\delta b||_2 \le ||\Sigma_1^{-1}||_2||\delta b||_2 = ||\delta b||_2/\sigma$. □

Смысл предложения 3.3 в том, что нормальное решение x единственно и может быть хорошо обусловленным, если наименьшее *ненулевое* сингулярное число не слишком мало. Это дает ключ к практическому алгоритму, обсуждаемому в следующем разделе.

Пример 3.7. Предположим, что проводится медицинское исследование воздействия некоторого препарата на уровень сахара в крови. Для каждого пациента (им присвоены номера от $i = 1$ до m) регистрируются начальный $(a_{i,1})$ и

конечный (b_i) уровень сахара в крови, количество введенного препарата ($a_{i,2}$) и прочая медицинская информация, в том числе вес в каждый из дней недельного курса лечения ($a_{i,3}, \ldots, a_{i,9}$). В целом для каждого пациента измеряются $n < m$ медицинских величин. Целью при этом является прогноз значения b_i по величинам $a_{i,1}, \ldots, a_{i,n}$, что рассматривается как задача наименьших квадратов $\min_x \|Ax - b\|_2$. Планируется использовать решение x для предсказания конечного уровня сахара b_j в крови будущего пациента j посредством вычисления скалярного произведения $\sum_{k=1}^{n} a_{jk}x_k$.

Поскольку обычно вес человека незначительно меняется изо дня в день, столбцы матрицы A с 3-го по 9-й, т. е. те, что содержат веса, скорей всего, будут очень похожи. Для целей нашего рассуждения примем, что столбцы 3 и 4 *одинаковы* (так действительно может быть, если значения весов округлены до ближайшего целого числа фунтов). Это означает, что ранг матрицы A не полон и вектор $x_0 = [0, 0, 1, -1, 0, \ldots, 0]^T$ принадлежит ее ядру. Поэтому, если x есть (нормальное) решение задачи наименьших квадратов $\min_x \|Ax - b\|_2$, то $x + \beta x_0$ также является решением (не обязательно нормальным) для *любого* числа β, например, $\beta = 0$ и $\beta = 10^6$. Есть ли какие-нибудь основания для того, чтобы предпочесть одно значение β другому? Ясно, что выбор $\beta = 10^6$ неудачен: если будущий пациент j приобретет фунт веса с первого дня на второй, то в предсказание конечного уровня сахара в крови, т. е. в величину $\sum_{k=1}^{n} a_{jk}x_k$, эта разница в один фунт войдет умноженной на 10^6. Выбор $\beta = 0$, что соответствует нормальному решению x, гораздо более разумен. ◇

Дополнительные аргументы, обосновывающие использование нормальных решений в задачах неполного ранга, можно найти в [141, 142].

Если A — квадратная невырожденная матрица, то единственным решением задачи $Ax = b$ является, конечно, вектор $x = A^{-1}b$. Если в A строк больше, чем столбцов, а ранг, возможно, не полон, то единственное нормальное решение задачи наименьших квадратов может быть записано в сходном виде $x = A^+b$. Участвующая здесь *псевдообратная матрица Мура–Пенроуза* A^+ определяется следующим образом:

Определение 3.2. (Псевдообратная Мура–Пенроуза A^+ для матрицы A, возможно, неполного ранга.)

Положим $A = U\Sigma V^T = U_1\Sigma_1 V_1^T$, *как в равенстве (3.1). Тогда* $A^+ \equiv V_1\Sigma_1^{-1}U_1^T$. *Это можно записать и в виде* $A^+ = V\Sigma^+U^T$, *где* $\Sigma^+ = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}^+ = \begin{bmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{bmatrix}$.

Таким образом, решение задачи наименьших квадратов всегда дается формулой $x = A^+b$. Если A — матрица неполного ранга, то такой вектор x имеет минимальную норму среди всех решений задачи.

3.5.1. Решение задач наименьших квадратов неполного ранга посредством SVD

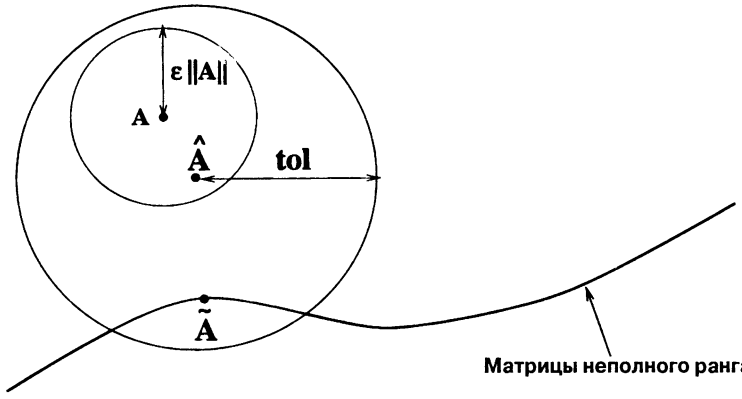
Наша цель состоит в том, чтобы вычислить нормальное решение x в условиях приближенных вычислений. В предыдущем разделе мы видели, что нормальное решение единственно и его число обусловленности зависит от наименьше-

го ненулевого сингулярного числа. Поэтому вычисление нормального решения предполагает знание наименьшего ненулевого сингулярного числа, а потому и ранга матрицы A. Основная трудность при этом состоит в том, что ранг является разрывной функцией от матрицы.

Рассмотрим, например, вырожденную 2×2-матрицу $A = \operatorname{diag}(1, 0)$. Ее наименьшее ненулевое сингулярное число есть $\sigma = 1$. Согласно предложению 3.3, нормальным решением задачи $\min_x \|Ax - b\|_2$ с $b = [1, 1]^T$ является вектор $x = [1, 0]^T$, а соответствующее число обусловленности равно $1/\sigma = 1$. Если посредством произвольно малого возмущения перейти к матрице $\hat{A} = \operatorname{diag}(1, \varepsilon)$, то σ уменьшается до ε, а вектор $x = [1, 1/\varepsilon]^T$ становится огромным, как и число обусловленности $1/\varepsilon$. Как правило, такие малые возмущения величины $O(\varepsilon)\|A\|_2$ и происходят при округлениях. Как мы только что видели, они способны увеличить число обусловленности с $1/\sigma$ до $1/\varepsilon$.

Алгоритмически эта разрывность в поведении учитывается следующим образом: в общем случае, всякое вычисленное сингулярное число $\hat{\sigma}_i$ удовлетворяет оценке $|\hat{\sigma}_i - \sigma_i| \leq O(\varepsilon)\|A\|_2$. Это является следствием обратной устойчивости, а именно вычисленное сингулярное разложение есть точное SVD для слабо возмущенной матрицы $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T = A + \delta A$, причем $\|\delta A\| = O(\varepsilon) \cdot \|A\|$. (Более детальное обсуждение вопроса дано в гл. 5.) Стало быть, всякое $\hat{\sigma}_i$, удовлетворяющее неравенству $\hat{\sigma}_i \leq O(\varepsilon)\|A\|_2$, может рассматриваться как нуль, потому что $\hat{\sigma}_i$ не отличимо от нуля в пределах точности вычислений. Применительно к нашему 2×2-примеру, это означает, что мы должны были бы заменить ε в матрице $\hat{A}$ нулем до решения задачи наименьших квадратов. Это увеличило бы наименьшее ненулевое сингулярное число с ε до 1; соответственно, число обусловленности уменьшилось бы с $1/\varepsilon$ до $1/\sigma = 1$.

Более общо, будем считать, что пользователем задана мера tol неопределенности в элементах матрицы A. Из наличия округлений следует, что $\mathrm{tol} \geq \varepsilon \cdot \|A\|$, но значение tol, в зависимости от происхождения A, может быть и много большим, чем эта граница. Положим теперь $\tilde{\sigma}_i = \hat{\sigma}_i$, если $\hat{\sigma}_i > \mathrm{tol}$, и $\tilde{\sigma}_i = 0$, в противном случае. Пусть $\tilde{\Sigma} = \operatorname{diag}(\tilde{\sigma}_i)$. Произведение $\hat{U}\tilde{\Sigma}\hat{V}^T$ назовем *усеченным* сингулярным разложением матрицы A, поскольку сингулярные числа меньшие, чем tol, заменены нулями. Решим задачу наименьших квадратов, пользуясь усеченным SVD вместо исходного. Обоснованием этого служит оценка $\|\hat{U}\tilde{\Sigma}\hat{V}^T - \hat{U}\hat{\Sigma}\hat{V}^T\|_2 = \|\hat{U}(\tilde{\Sigma} - \hat{\Sigma})\hat{V}^T\|_2 < \mathrm{tol}$, т. е. изменения в A, вызванные заменой $\hat{\sigma}_i$ на $\tilde{\sigma}_i$, меньше, чем изначально присутствующая неопределенность в данных. Основным доводом в пользу $\tilde{\Sigma}$ является то, что среди всех матриц, находящихся от $\hat{\Sigma}$ на расстоянии, не превышающем tol, $\tilde{\Sigma}$ максимизирует наименьшее ненулевое сингулярное число σ. Иначе говоря, такой выбор минимизирует величину нормального решения x и его число обусловленности. На рисунке проиллюстрированы геометрические взаимосвязи между исходной матрицей A, матрицей $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$ и матрицей $\tilde{A} = \hat{U}\tilde{\Sigma}\hat{V}^T$. Каждая из матриц изображается точкой евклидова пространства $\mathbb{R}^{m \cdot n}$. Матрицы неполного ранга образуют в этом пространстве поверхность, что также иллюстрируется рисунком.



Пример 3.8. Проиллюстрируем описанную выше процедуру с помощью двух матриц неполного ранга, имеющих размер 20×10: A_1 ранга $r_1 = 5$ и A_2 ранга $r_2 = 7$. Пусть $A_i = U_i \Sigma_i V_i^T$ $(i = 1, 2)$ — сингулярные разложения этих матриц, где общим размером матриц U_i, Σ_i и V_i является ранг r_i матрицы A_i; это тот же способ записи, что и в предложении 3.3. Ненулевые сингулярные числа матрицы A_i (или матрицы Σ_i) указаны крестиками на рис. 3.4 (для A_1) и рис. 3.5 (для A_2). Заметим, что A_1 имеет пять больших ненулевых сингулярных чисел (все они чуть больше единицы, поэтому соответствующие им крестики сливаются в один, находящийся на правой границе рисунка). В то же время, семь ненулевых сингулярных чисел матрицы A_2 на рис. 3.5 распределены на отрезке, левым концом которого является число $1.2 \cdot 10^{-9} \approx \text{tol}$.

Выберем r_i-мерный вектор x_i' и положим $x_i = V_i x_i'$, $b_i = A_i x_i = U_i \Sigma_i x_i'$; таким образом, x_i есть точное нормальное решение задачи $\min_x \|A_i x - b_i\|_2$. Рассмотрим теперь последовательность возмущенных задач с матрицами $A_i + \delta A$, где возмущение δA выбирается случайно, но так, чтобы значение $\|\delta A\|$ варьировалось в широких пределах. Полагая $\text{tol} = 10^{-9}$, приме́ним к задачам $\min \|(A_i + \delta A) y_i - b_i\|_2$ процедуру усеченного SVD. Сплошные линии на рис. 3.4 и 3.5 указывают значение вычисленного ранга матрицы $A_i + \delta A$ (т. е. количества найденных сингулярных чисел, б/'ольших, чем $\text{tol} = 10^{-9}$) как функцию от $\|\delta A\|_2$ (см. верхние графики) и ошибку $\|y_i - x_i\|_2 / \|x_i\|_2$ (нижние графики). Matlab-программа, создающая эти графики, находится в HOMEPAGE/Matlab/Rank Deficient.m.

Ситуация на рис. 3.4 проще, поэтому мы начнем обсуждение с нее. Матрица $A_1 + \delta A$ имеет пять сингулярных чисел вблизи или чуть правее 1 и еще пять сингулярных чисел, не превышающих $\|\delta A\|_2$. Если $\|\delta A\|_2 < \text{tol}$, то вычисленный ранг будет для $A_1 + \delta A$ тем же, что и для A_1, т. е. будет равен 5. Ошибка тоже медленно увеличивается от уровня порядка машинного эпсилон ($\approx 10^{-16}$) до приблизительно 10^{-10} вблизи $\|\delta A\|_2 = \text{tol}$. Для бо́льших значений $\|\delta A\|_2$ обе величины делают скачок соответственно к 10 и 1. Это согласуется с нашим анализом в предложении 3.3, который показывает, что число обусловленности обратно наименьшему ненулевому сингулярному числу, т. е. наименьшему сингулярному числу, превосходящему tol. При $\|\delta A\|_2 < \text{tol}$ это наименьшее ненулевое сингулярное число близко к (или даже чуть превышает) 1. Поэтому предложение 3.3 предсказывает ошибку порядка $\|\delta A\|_2 / O(1) = \|\delta A\|_2$. Эта

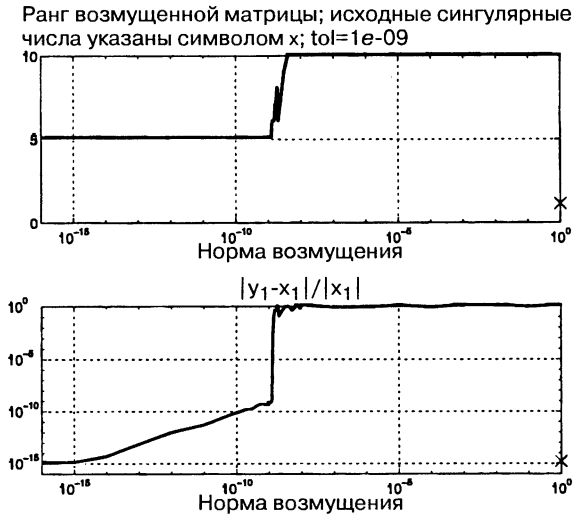


Рис. 3.4. Решение задачи $\min_{y_1} \|(A_1 + \delta A)y_1 - b_1\|_2$ методом усеченного SVD при $\mathrm{tol} = 10^{-9}$. Сингулярные числа матрицы A_1 указаны крестиками. Значения $\|\delta A\|_2$ откладываются по горизонтальной оси. На верхнем графике показан ранг матрицы $A_1 + \delta A$, т. е. количество ее сингулярных чисел, превосходящих tol. На нижнем графике представлено отношение $\|y_1 - x_1\|_2 / \|x_1\|_2$, где x_1 — решение задачи при $\delta A = 0$.

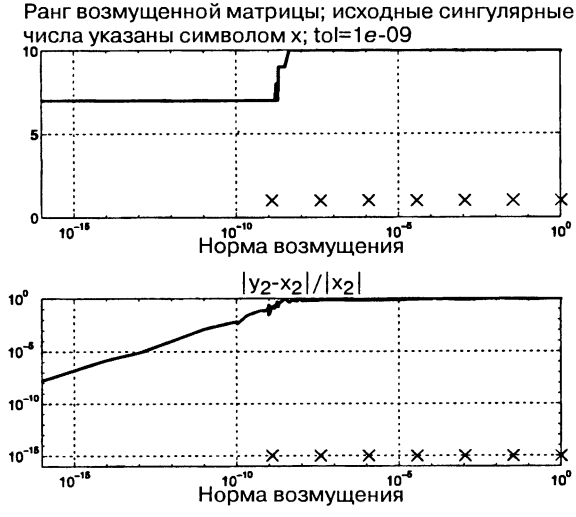


Рис. 3.5. Решение задачи $\min_{y_2} \|(A_2 + \delta A)y_2 - b_2\|_2$ методом усеченного SVD при $\mathrm{tol} = 10^{-9}$. Сингулярные числа матрицы A_2 указаны крестиками. Значения $\|\delta A\|_2$ откладываются по горизонтальной оси. На верхнем графике показан ранг матрицы $A_2 + \delta A$, т. е. количество ее сингулярных чисел, превосходящих tol. На нижнем графиае представлено отношение $\|y_2 - x_2\|_2 / \|x\|_2$, где x_2 — решение задачи при $\delta A = 0$.

хорошо обусловленная ситуация подтверждается малыми значениями ошибки слева от точки $\|\delta A\|_2 = \text{tol}$ на нижнем графике рис. 3.4. С другой стороны, при $\|\delta A\|_2 > \text{tol}$ наименьшее ненулевое сингулярное число имеет порядок $O(\|\delta A\|_2)$. Вследствие малости этого числа, ошибка делает скачок к уровню $\|\delta A\|_2/O(\|\delta A\|_2) = O(1)$, что видно из части нижнего графика рис. 3.4, находящейся правее точки $\|\delta A\|_2 = \text{tol}$.

Ненулевые сингулярные числа матрицы A_2 показаны на рис. 3.5 крестиками. Наименьшее из них равно $1.2 \cdot 10^{-9}$, что чуть больше, чем tol. Поэтому при $\|\delta A\|_2 < \text{tol}$ прогноз для ошибки есть $\|\delta A\|_2/10^{-9}$, что возрастает до $O(1)$ при $\|\delta A\|_2 = \text{tol}$. Это подтверждается нижним графиком рис. 3.5. ◇

3.5.2. Решение задач наименьших квадратов неполного ранга методом QR-разложения с выбором главного столбца

Более экономичной, но подчас менее точной альтернативой методу SVD является QR-разложение *с выбором главного столбца*. Пусть в точной арифметике ранг матрицы A равен $r < n$, и пусть ее первые r столбцов линейно независимы. Тогда QR-разложение A имеет вид

$$A = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

где $r \times r$-подматрица A_{11} невырожденна, а R_{12} — подматрица размера $r \times (n-r)$. При машинных вычислениях мы могли бы надеяться получить матрицу вида

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix}$$

с блоком R_{22} очень малой нормы (порядка $\varepsilon\|A\|_2$). В этом случае можно было бы просто положить $R_{22} = 0$ и минимизировать $\|Ax - b\|_2$ следующим образом: пусть $[Q, \tilde{Q}]$ — квадратная ортогональная матрица, тогда

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (Ax - b) \right\|_2^2 = \left\| \begin{bmatrix} Rx - Q^T b \\ -\tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\ &= \|Rx - Q^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2. \end{aligned}$$

Запишем $Q = [Q_1, Q_2]$ и $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ в согласии с $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$. Функция

$$\|Ax - b\|_2^2 = \|R_{11}x_1 + R_{12}x_2 - Q_1^T b\|_2^2 + \|Q_2^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2$$

минимизируется вектором $x = \begin{bmatrix} R_{11}^{-1}(Q_1^T b - R_{12}x_2) \\ x_2 \end{bmatrix}$ при произвольном x_2. Отметим, что выбор $x_2 = 0$ необязательно минимизирует $\|x\|_2$, но это — разумный выбор, особенно если матрица R_{11} хорошо обусловлена, а произведение $R_{11}^{-1}R_{12}$ мало.

К сожалению, этот метод не надежен, так как R может быть близка к матрице неполного ранга, даже если ни один блок R_{22} в ней не мал. Например, для двухдиагональной $n \times n$-матрицы

$$A = \begin{bmatrix} \frac{1}{2} & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \frac{1}{2} \end{bmatrix}$$

имеем $\sigma_{\min}(A) \approx 2^{-n}$, но $A = Q \cdot R$ с $Q = I$ и $R = A$, и ни один блок R_{22} в R не мал.

Чтобы избежать подобных ситуаций, когда не удается распознать неполноту ранга, можно применить QR-разложение с *выбором главного* столбца. Это означает, что вычисляется разложение $AP = QR$, где P — некоторая матрица-перестановка. Идея состоит в том, чтобы на шаге i (где i меняется от 1 до числа n столбцов матрицы A) выбрать столбец с наибольшей нормой в еще не приведенной части A (т. е. в подматрице, стоящей на пересечении столбцов $i, i+1, \ldots, n$ и строк $i, i+1, \ldots, m$) и переставить его с i-м столбцом. Далее выполняется обычное отражение, аннулирующее элементы $i+1, \ldots, m$ столбца i. Эта стратегия продиктована желанием получить как можно лучше обусловленный блок R_{11} и как можно меньшие элементы в блоке R_{22}.

Пример 3.9. Применим QR-разложение с выбором главного столбца к матрице предыдущего примера (число .5 на главной диагонали и 1 на первой наддиагонали) при $n = 11$. Получим $R_{11,11} = 4.23 \cdot 10^{-4}$, что является хорошим приближением к $\sigma_{\min}(A) = 3.66 \cdot 10^{-4}$. Заметим, что всегда $R_{nn} \geq \sigma_{\min}(A)$. Действительно, замена элемента R_{nn} на 0 приводит к уменьшению ранга; между тем, $\sigma_{\min}(A)$ — это норма наименьшего возмущения, способного понизить значение ранга. ◇

В общем случае можно лишь утверждать, что $\frac{R_{nn}}{\sigma_{\min}(A)} \lesssim 2^n$, но обычно R_{nn} является приемлемым приближением к $\sigma_{\min}(A)$. Наихудший случай, однако, столь же плох, как и наибольший рост главного элемента в методе GEPP.

В самое последнее время предметом интенсивных исследований стали более сложные схемы, так называемые QR-алгоритмы *с выявлением ранга* (*rank-revealing* QR *algorithms*). Эти алгоритмы распознают значение ранга более надежно и подчас быстрее, чем QR-разложение с выбором главного столбца. Алгоритмы с выявлением ранга построены в [28, 30, 48, 50, 109, 126, 128, 150, 196, 236]. Мы вернемся к их обсуждению в следующем разделе.

QR-разложение с выбором главного столбца реализовано LAPACK-подпрограммой `sgeqpf`. LAPACK включает в себя еще несколько аналогичных разложений: QR (`sgerqf`), LQ (`sgelqf`) и QL (`sgeqlf`). Будущие версии LAPACK'а будут содержать улучшенные варианты QR-разложения.

3.6. Сравнение производительности методов для решения задач наименьших квадратов

Какой метод быстрее всего решает плотные задачи наименьших квадратов? Из обсуждения в разд. 3.2 следует, что наиболее быстрым является метод нормальных уравнений, за ним следует метод QR и только потом SVD. Если A — хорошо обусловленная матрица, то метод нормальных уравнений по точности сравним с прочими методами. Поэтому, несмотря на отсутствие обратной устойчивости, метод может использоваться наряду с другими. Если матрица A не является хорошо обусловленной, но и не близка к матрице неполного ранга, то следует обратиться к QR-разложению.

Построение быстрых алгоритмов для задач неполного ранга — это область текущих исследований, поэтому затруднительно рекомендовать единый алгоритм для всех случаев. Мы дадим краткое изложение недавней работы [206], где было проведено сравнение производительности нескольких алгоритмов друг с другом, а также с наиболее быстрым устойчивым алгоритмом для задач полного ранга, а именно QR-разложением без выбора главного столбца, реализованным на основе отражений, как описано в разд. 3.4.1, и с иерархией памяти, оптимизированной в соответствии с вопросом 3.17. Сравнение проводилось на компьютере IBM RS6000/590 в режиме вычислений с двойной точностью. Среди сравниваемых методов были QR-алгоритмы с выявлением ранга, упоминавшиеся в разд. 3.5.2, а также различные реализации метода SVD (см. разд. 5.4). В тестировании использовались матрицы различных порядков и с разнообразным распределением сингулярных чисел. Мы обсудим результаты, полученные для двух типов матриц:

тип 1: случайные матрицы с элементами, равномерно распределенными на отрезке $[-1, 1]$;

тип 2: матрицы, сингулярные числа которых геометрически распределены на отрезке $[1, \varepsilon]$ (иначе говоря, i-е сингулярное число равно γ^i, где $\gamma^n = \varepsilon$).

Матрицы типа 1, в общем случае, хорошо обусловлены, а матрицы типа 2 близки к матрицам неполного ранга. В тестировании участвовали квадратные матрицы как малого, так и большого порядка (соответственно, $n = m = 20$ и $n = m = 1600$). Использовались именно квадратные матрицы, потому что для $m \times n$ матрицы A с m, значительно бóльшим, чем n, имеется более дешевая альтернатива: выполнить QR-разложение в качестве подготовительного шага, а затем применить к полученной матрице R алгоритм с выявлением ранга либо метод SVD. (Так делается в LAPACK'е.) Если $m \gg n$, то стоимость начального QR-разложения доминирует над ценой последующих операций с $n \times n$-матрицей R, а потому все указанные алгоритмы имеют приблизительно одну и ту же стоимость.

Наиболее быстрым вариантом QR-разложения с выявлением ранга был метод из [30, 196]. Для матриц типа 1 отношение времени работы этого метода к времени QR-алгоритма без выбора главного столбца изменялось от 3.2 при $n = m = 20$ до всего лишь 1.1 при $n = m = 1600$. Для матриц типа 2 это отношение изменялось от 2.3 ($n = m = 20$) до 1.2 ($n = m = 1600$). Подобное же отношение для программы `dgeqpf` из текущей версии LAPACK'а колебалось между 2 и 2.5 для матриц обоих типов.

Среди вариантов SVD наиболее быстрым был метод из [58], хотя при $n = m = 1600$ примерно столь же быстро работал алгоритм, основанный на стратегии «разделяй и властвуй» (см. разд. 5.3.3). (Этот алгоритм, кроме того, использовал гораздо меньшую память.) Отношение времени работы SVD к времени QR-алгоритма без выбора главного столбца изменялось для матриц типа 1 от 7.8 (при $n = m = 20$) до 3.3 (при $n = m = 1600$). Для матриц типа 2 это отношение изменялось от 3.5 ($n = m = 20$) до 3.0 ($n = m = 1600$). Для алгоритма `dgelss` из текущей версии LAPACK'а подобное же отношение колебалось от 4 (матрицы типа 2 при $n = m = 20$) до 97 (матрицы типа 1 при $n = m = 1600$). Это огромное замедление следует, по-видимому, отнести к эффектам иерархической организации памяти.

Мы видим, таким образом, что при решении задач неполного ранга имеется баланс между надежностью и скоростью: QR-разложение без выбора главного столбца — это самый быстрый, но и наименее надежный метод, алгоритм SVD — самый медленный, но зато самый надежный, а методы QR с выявлением ранга находятся посредине между тем и другим. Если $m \gg n$, то стоимость всех алгоритмов примерно одинакова. Выбор конкретного метода зависит от сравнительной важности скорости и надежности для пользователя.

Новые версии LAPACK'а будут содержать улучшенные варианты QR-алгоритмов с выявлением ранга и метода SVD для задачи наименьших квадратов.

3.7. Литература и смежные вопросы к главе 3

Наилучшим в настоящее время справочником по задачам наименьших квадратов является книги [33], где излагаются варианты обсуждавшейся здесь основной задачи (такие, как задачи с ограничениями и весами, или перестройка решения при малоранговой модификации задачи), различные способы регуляризации в случае неполного ранга и программы для разреженных задач. См. также гл. 5 книги [121] и [168]. Теория возмущений и границы ошибок для решения задачи наименьших квадратов подробно изложены в [149]. QR-разложения с выявлением ранга обсуждаются в [28, 30, 48, 50, 126, 150, 196, 206, 236]. В этих статьях исследуется, в частности, зависимость между трудоемкостью метода и надежностью определения ранга. В [206] проведено, кроме того, обширное сравнение производительности существующих методов для задач неполного ранга.

3.8. Вопросы к главе 3

Вопрос 3.1 (*легкий*). Показать, что два варианта алгоритма 3.1 - методы CGS и MGS, — математически эквивалентны. Для этого показать, что две формулы для r_{ij} в алгоритме 3.1 дают в точной арифметике один и тот же результат.

Вопрос 3.2 (*легкий*). Этот вопрос иллюстрирует различие в численной устойчивости трех алгоритмов вычисления QR-разложения матрицы: метода отражений (алгоритм 3.2), метода CGS (алгоритм 3.1) и метода MGS (алгоритм 3.1). Импортируйте Matlab-программу QRStability.m из HOMEPAGE/Matlab/QRStability.m. Эта программа генерирует случайные матрицы с задаваемы-

ми пользователем размерами `m` и `n` и числом обусловленности `cnd`; для них указанными выше тремя алгоритмами вычисляется QR-разложение и оценивается точность полученного результата. Это делается с помощью *невязки* $||A - Q \cdot R||/||A||$, которая для устойчивого алгоритма должна быть величиной порядка машинного эпсилон ε, и *меры ортогональности* $||Q^T \cdot Q - I||$, которая тоже должна быть величиной порядка ε. Поработайте с этой программой, используя умеренное число (`samples` = 20) случайных матриц с малыми размерами (например, `m` = 6 и `n` = 4) и числами обусловленности, изменяющимися от `cnd` =1 до `cnd` = 10^{15}. Опишите ваши выводы. Какой из алгоритмов устойчивей других? Попробуйте охарактеризовать возможную величину меры ортогональности $||Q^T \cdot Q - I||$ как функцию от выбора алгоритма, а также чисел `cnd` и ε.

Вопрос 3.3 (*средней трудности; трудный*). Пусть A — матрица полного ранга и размера $m \times n$, где $m \geq n$.

1. (*средней трудности*). Показать, что система линейных уравнений $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \cdot \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$ совместна и компонента x ее решения минимизирует функцию $||Ax - b||_2$. Одно из достоинств этой формулировки состоит в том, что при необходимости получить более точное решение мы можем применить к данной системе алгоритм итерационного уточнения (см. разд. 2.5).
2. (*средней трудности*). Выразить число обусловленности матрицы A через ее сингулярные числа. Указание: использовать сингулярное разложение матрицы.
3. (*средней трудности*). Для обратной матрицы A^{-1}, представленной в блочном 2 × 2-виде, дать явные выражения блоков через блоки матрицы A. Указание: использовать блочное 2 × 2 гауссово исключение. Где нам уже встречался блочный элемент (2, 1)?
4. (*трудный*). Указать способ использования QR-разложения матрицы A в рамках алгоритма итерационного уточнения приближенного решения x.

Вопрос 3.4 (*средней трудности*). *Взвешенные наименьшие квадраты.* Предположим, что некоторые компоненты вектора $Ax - b$ важнее других. Тогда можно снабдить их весами, т. е. числовыми множителями d_i, и решать вместо обычной задачи наименьших квадратов взвешенную задачу $\min||D(Ax - b)||_2$, где диагональная матрица D составлена из чисел d_i. Более общо, вспомним, что для симметричной положительно определенной матрицы C формула $||x||_C \equiv (x^T C x)^{1/2}$ задает норму, поэтому можно рассмотреть задачу минимизации функции $||Ax - b||_C$. Вывести нормальные уравнения для этой задачи, а также постановку, согласованную с предыдущим вопросом.

Вопрос 3.5 (*средней трудности; Z. Bai*). Пусть A — положительно определенная $n \times n$-матрица. Говорят, что векторы u_1 и u_2 A-ортогональны, если $u_1^T A u_2 = 0$. Если $U \in \mathbb{R}^{n \times r}$ и $U^T A U = I$, то говорят, что столбцы матрицы U A-ортогональны. Показать, что всякое линейное подпространство имеет A-ортонормальный базис.

Вопрос 3.6 (*легкий; Z. Bai*). Пусть матрица A имеет форму

$$A = \begin{bmatrix} R \\ S \end{bmatrix},$$

где R — верхнетреугольная $n \times n$-матрица, а S — плотная матрица размера $n \times n$. Описать алгоритм приведения A к верхней треугольной форме, основанный на использовании отражений. Ваш алгоритм не должен «заполнять» нули в R. Он будет, следовательно, более экономичным, чем алгоритм 3.2 в применении к данной матрице A.

Вопрос 3.7 (*средней трудности; Z. Bai*). Пусть $A = R + uv^T$, где R — верхняя треугольная матрица, а u и v суть векторы-столбцы. Указать эффективный алгоритм для вычисления QR-разложения матрицы A. Указание: используя вращения, можно построить алгоритм со сложностью $O(n^2)$ операций, в то время как алгоритм 3.2 требует $O(n^3)$ операций.

Вопрос 3.8 (*средней трудности; Z. Bai*). Пусть $x \in \mathbb{R}^n$, а P — такое отражение, что $Px = \pm\|x\|_2 e_1$. Пусть $G_{1,2}, \ldots, G_{n-1,n}$ — вращения Гивенса, а $Q = G_{12} \ldots G_{n-1,n}$. Предположим, что $Qx = \pm\|x\|_2 e_1$. Должны ли совпадать матрицы P и Q? (Вы должны либо доказать равенство, либо привести контрпример.)

Вопрос 3.9 (*легкий; Z. Bai*). Пусть A — матрица размера $m \times n$, имеющая SVD $A = U\Sigma V^T$. Выразить через U, Σ и V сингулярные разложения следующих матриц:

1. $(A^TA)^{-1}$,
2. $(A^TA)^{-1}A^T$,
3. $A(A^TA)^{-1}$,
4. $A(A^TA)^{-1}A^T$.

Вопрос 3.10 (*средней трудности; R. Schreiber*). Пусть A_k — наилучшее приближение ранга k матрицы A, определенное в утверждении 9 теоремы 3.3. Пусть σ_i обозначает i-е сингулярное число матрицы A. Показать, что матрица A_k определена единственным образом, если $\sigma_k > \sigma_{k+1}$.

Вопрос 3.11 (*легкий; Z. Bai*). Пусть A — матрица размера $m \times n$. Показать, что выбор $X = A^+$ (псевдообратная Мура–Пенроуза) минимизирует функцию $\|AX - I\|_F$ на множестве всех $n \times m$-матриц X. Каково значение этого минимума?

Вопрос 3.12 (*средней трудности; Z. Bai*). Пусть A, B и C — матрицы таких размеров, что произведение A^TCB^T определено корректно. Пусть χ — множество матриц X, минимизирующих величину $\|AXB - C\|_F$, и пусть X_0 — единственный элемент этого множества, минимизирующий $\|X\|_F$. Показать, что $X_0 = A^+CB^+$. Указание: использовать сингулярные разложения матриц A и B.

Вопрос 3.13 (*средней трудности; Z. Bai*). Показать, что матрица A^+ — псевдообратная Мура–Пенроуза для матрицы A — удовлетворяет следующим соотношениям:

$$\begin{aligned} AA^+A &= A, \\ A^+AA^+ &= A^+, \\ A^+A &= (A^+A)^T, \\ AA^+ &= (AA^+)^T. \end{aligned}$$

Вопрос 3.14 (*средней трудности*). Доказать утверждение 4 теоремы 3.3. Пусть $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, где A — квадратная матрица, имеющая SVD $A = U\Sigma V^T$. Положим $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, $U = [u_1, \ldots, u_n]$ и $V = [v_1, \ldots, v_n]$. Доказать, что $2n$ собственных значений матрицы H — это числа $\pm\sigma_i$, а соответствующие нормированные собственные векторы имеют вид $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$. Распространить эти результаты на случай прямоугольной матрицы A.

Вопрос 3.15 (*средней трудности*). Пусть A — матрица полного ранга и размера $m \times n$, где $m < n$. Задача $\min\|Ax - b\|_2$ называется *недоопределенной задачей наименьших квадратов*. Показать, что ее решения образуют линейное многообразие размерности $n - m$. Указать способы вычисления единственного решения с минимальной нормой посредством подходящих модификаций методов нормальных уравнений, QR-разложения и SVD.

Вопрос 3.16 (*средней трудности*). Доказать лемму 3.1.

Вопрос 3.17 (*трудный*). В разделе 2.6.3 было показано, как реорганизовать гауссово исключение так, чтобы на каждом его шаге выполнялись BLAS-операции уровней 2 и 3 с их более высокой скоростью. В этой задаче мы покажем, как умножить матрицу на последовательность отражений, используя BLAS-операции уровней 2 и 3.

1. Пусть $u_1, \ldots, u_b$ — последовательность n-мерных векторов, таких, что $\|u_i\|_2 = 1$ и первые $i-1$ компонент в u_i равны нулю. Пусть $P = P_b \cdot P_{b-1} \ldots P_1$, где $P_i = I - 2u_iu_i^T$ есть отражение. Положим $U = [u_1, \ldots, u_b]$. Показать, что найдется нижнетреугольная $b \times b$-матрица T, такая, что $P = I - UTU^T$. Предложить алгоритм вычисления элементов матрицы T. Таким образом, умножение на b отражений $P_1, \ldots, P_b$ можно заменить тремя умножениями на матрицы U_T, T и U (с предварительным вычислением T).
2. Пусть $\mathrm{House}(x)$ — процедура, которая по заданному вектору x вычисляет нормированный вектор u, такой, что $(I - 2uu^T)x = \|x\|_2 e_1$. В разделе 3.4 было показано, как реализовать такую процедуру. Алгоритм 3.2, вычисляющий QR-разложение $m \times n$-матрицы A, может быть записан следующим образом:

```
for i = 1 : m
    u_i = House(A(i : m, i))
    P_i = I - 2u_i u_i^T
```

$A(i:m, i:n) = P_i A(i:m, i:n)$
endfor

Указать способ эффективной реализации этого алгоритма посредством BLAS-операций уровня 2 (в частности, матрично-векторных умножений и модификаций ранга 1). Каково число флопов для этой реализации? (Можно ограничиться указанием членов с наивысшими степенями n и m.) Достаточно написать короткую программу в тех же обозначениях, что и выше (хотя хорошим способом самопроверки является составление программы в Matlab'е и ее сравнение с функцией QR-разложения самого Matlab'а!).

3. Используя результаты пункта 1, указать способы реализации QR-разложения посредством BLAS-операций уровня 3. Каково будет число операций? Этот прием используется для того, чтобы ускорить QR-разложение подобно тому, как в разд. 2.6 было ускорено гауссово исключение. Он применен в LAPACK-программе `sgeqrf`.

Вопрос 3.18 (*средней трудности*). Нередко приходится решать *задачи наименьших квадратов с ограничениями*, где вектор x помимо минимизации функции $||Ax - b||_2$ должен еще удовлетворять линейному или нелинейному ограничению. Рассмотрим одну из таких задач. Пусть нужно найти вектор x, минимизирующий $||Ax - b||_2$ при наличии линейного ограничения $Cx = d$. Предположим, что A — матрица размера $m \times n$, а C — матрица размера $p \times n$, причем C имеет полный ранг. Предположим также, что $p \leq n$ (тогда система $Cx = d$ заведомо совместна) и $n \leq m + p$ (так что задача в целом не является недоопределенной). Показать, что задача имеет единственное решение, если матрица $\begin{bmatrix} A \\ C \end{bmatrix}$ имеет полный ранг. Указать способ вычисления вектора x, использующий два QR-разложения, а также матрично-векторные умножения и решение треугольных систем линейных уравнений. Указание: вам могут пригодиться LAPACK-программа `sgglse` и ее описание в руководстве для пользователей LAPACK'а [10] (NETLIB/lapack/lug/lapack_lug.html).

Вопрос 3.19 (*трудный; программирование*). Написать программу (на Matlab'е или каком-либо ином языке) для обновления геодезической базы данных методом наименьших квадратов, как это описано в разд. 3.3. В качестве входа принять набор «вех», их приближенные координаты (x_i, y_i) и набор новых измерений углов θ_j и расстояний L_{ij}. Выход должен состоять из поправок $(\delta x_i, \delta y_i)$ для каждой вехи, границы ошибок для этих поправок и картины расположения (триангуляции) старых и новых вех.

Вопрос 3.20 (*трудный*). Доказать теорему 3.4.

Вопрос 3.21 (*средней трудности*). Решить заново пример 3.1, используя метод решения задачи наименьших квадратов неполного ранга из разд. 3.5.1. Улучшается ли при этом точность аппроксимирующих многочленов высокой степени?

Глава 4

Несимметричная проблема собственных значений

4.1. Введение

В разделе 4.2 обсуждаются канонические формы, в разделе 4.3 — теория возмущений и в разделе 4.4 — алгоритмы, решающие проблему собственных значений для (одной) несимметричной матрицы A. Специальному случаю вещественных симметричных матриц $A = A^T$ (а также вычислению SVD) посвящена гл. 5. В разделе 4.5 рассмотрены обобщенные задачи на собственные значения, в которых участвуют две или более матриц. Здесь же указаны приложения из теории колебаний, решения линейных дифференциальных уравнений и вычислительной геометрии, мотивирующие исследование таких задач. Наконец, раздел 4.6 составлен в виде списка, суммирующего все сведения о канонических формах, алгоритмах, стоимостях, приложениях и имеющихся программах.

Условно алгоритмы для задач на собственные значения можно разбить на две группы: *прямые методы* и *итерационные методы*. В этой главе рассматриваются лишь прямые методы, предназначенные для вычисления всех собственных значений и (если требуется) собственных векторов. Обычно прямые методы применяются к плотным матрицам и требуют $O(n^3)$ операций для вычисления всех собственных значений и собственных векторов; эта стоимость сравнительно не чувствительна к тому, каковы в действительности элементы матрицы.

Прямой метод, чаще всего используемый на практике, — это QR-*алгоритм с неявными сдвигами* (см. разд. 4.4.8). Интересно, что после более чем 30 лет безупречной службы этого метода совсем недавно были обнаружены, проанализированы и исправлены некоторые ситуации его расходимости [25, 65]. Доказательства его глобальной сходимости все еще не существует, хотя современная версия метода считается очень надежной. Поэтому задача построения алгоритма, который был бы численно устойчив и при этом глобально (и быстро!) сходился, остается открытой. (Заметим, что «прямые» методы тоже, в действительности, являются итерационными: задача вычисления собственных значений математически эквивалентна отысканию корней многочленов, что не может быть сделано посредством неитерационных методов. Мы называем метод *прямым*, если на практике он (почти) всегда сходится за фиксированное число итераций.)

Итерационные методы, обсуждаемые в гл. 7, обычно применяются к разреженным матрицам либо к матрицам, которые доступны лишь через свои произведения с векторами. Как правило, итерационные методы дают приближе-

ния только для некоторого подмножества собственных значений и собственных векторов; обычно итерации продолжают лишь, пока не получены достаточно точные приближения нескольких (немногих) собственных значений. Характер сходимости методов этой группы сильно зависит от конкретных значений элементов матрицы.

4.2. Канонические формы

Определение 4.1. *Многочлен* $p(\lambda) = \det(A - \lambda I)$ *называется* характеристическим многочленом *матрицы* A. *Корни уравнения* $p(\lambda) = 0$ *суть* собственные значения *этой матрицы.*

Так как степень характеристического многочлена $p(\lambda)$ равна порядку n матрицы A, то он имеет n корней, поэтому A имеет n собственных значений.

Определение 4.2. *Ненулевой вектор* x, *удовлетворяющий условию* $Ax = \lambda x$, *называется* (правым) собственным вектором, *соответствующим собственному значению* λ. *Ненулевой вектор* y, *такой, что* $y^*A = \lambda y^*$, *называется* левым собственным вектором. *(Напомним, что вектор* $y^* = (\overline{y})^T$ *получается из* y операцией сопряжения.)

Большинство наших алгоритмов предусматривают преобразование матрицы к более простым, *каноническим* формам, из которых легко вычисляются собственные значения и собственные векторы. Такого рода преобразования называются *подобиями* (см. определение ниже). Две наиболее известные канонические формы — это *форма Жордана* и *форма Шура.* Форма Жордана полезна в теории, однако найти ее численно устойчивым образом очень трудно. Вот почему наши алгоритмы ориентированы на вычисление формы Шура, а не формы Жордана.

Чтобы оценить значение форм Жордана и Шура, выясним, для каких матриц собственные значения вычисляются легко. Наиболее очевидный случай — это *диагональная матрица*, собственными значениями которой являются ее диагональные элементы. Столь же прост случай *треугольной матрицы*; для нее также собственные значения — это диагональные элементы. Вскоре мы увидим, что матрицы, имеющие форму Жордана или Шура, треугольны. Вспомним, однако, что у вещественной матрицы могут быть комплексные собственные значения, поскольку корни ее характеристического многочлена могут быть как вещественными, так и комплексными. Поэтому не для всякой *вещественной* матрицы найдется *вещественная* треугольная матрица с теми же собственными значениями, так как все собственные значения вещественной треугольной матрицы суть вещественные числа. Следовательно, мы должны либо использовать комплексные числа, либо в поисках канонической формы для вещественных матриц обратиться к чему-то более общему, чем вещественная треугольная матрица. Оказывается, что достаточно рассматривать *блочно-треугольные матрицы*, т. е. матрицы вида

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1b} \\ & A_{22} & \cdots & A_{2b} \\ & & \ddots & \vdots \\ & & & A_{bb} \end{bmatrix}, \tag{4.1}$$

где все блоки A_{ii} — квадратные и все стоящие под ними элементы равны нулю. Легко показать, что характеристический многочлен $\det(A - \lambda I)$ матрицы A есть произведение $\prod_{i=1}^{b} \det(A_{ii} - \lambda I)$ характеристических многочленов блоков A_{ii}. Следовательно, множество $\lambda(A)$ собственных значений матрицы A является объединением $\bigcup_{i=1}^{b} \lambda(A_{ii})$ множеств собственных значений диагональных блоков A_{ii} (см. вопрос 4.1). Мы собираемся вычислять блочно-треугольные канонические формы путем последовательного расщепления больших диагональных блоков на все более и более мелкие. Если исходная матрица A — комплексная, то, в конечном счете, все диагональные блоки станут размера 1×1, поэтому будет получена треугольная каноническая форма. Если же A — вещественная матрица, то окончательная каноническая форма может содержать диагональные блоки как размера 1×1 (они соответствуют вещественным собственным значениям), так и размера 2×2 (соответствуют сопряженным парам комплексных собственных значений). Такая блочно-треугольная матрица называется *квазитреугольной*.

Собственные векторы (блочно) треугольной матрицы определяются легко (см. по этому поводу разд. 4.2.1).

Определение 4.3. *Пусть S — произвольная невырожденная матрица. Говорят, что матрицы A и $B = S^{-1}AS$* подобны; *при этом S осуществляет подобие.*

Предложение 4.1. *Пусть $B = S^{-1}AS$, т. е. матрицы A и B подобны. Тогда A и B имеют одни и те же собственные значения. Вектор x (y) тогда и только тогда является правым (левым) собственным вектором матрицы A, когда $S^{-1}x$ (S^*y) является правым (левым) собственным вектором матрицы B.*

Доказательство. Воспользуемся равенством $\det(X \cdot Y) = \det(X) \cdot \det(Y)$, справедливым для произвольных квадратных матриц X и Y. Имеем $\det(A - \lambda I) = \det(S^{-1}(A - \lambda I)S) = \det(B - \lambda I)$, поэтому A и B имеют один и тот же характеристический многочлен. Соотношение $Ax = \lambda x$ равносильно соотношению $S^{-1}ASS^{-1}x = \lambda S^{-1}x$, или $B(S^{-1}x) = \lambda(S^{-1}x)$. Точно так же, $y^*A = \lambda y^*$ справедливо тогда и только тогда, когда $y^*SS^{-1}AS = \lambda y^*S$, или $(S^*y)^*B = \lambda(S^*y)^*$. ◇

Теорема 4.1 (жорданова каноническая форма). *Для всякой матрицы A найдется невырожденная матрица S, такая, что матрица $S^{-1}AS = J$ имеет каноническую форму Жордана. Это означает, что J является блочно-диагональной матрицей, т. е. $J = \operatorname{diag}(J_{n_1}(\lambda_1), J_{n_2}(\lambda_2), \ldots, J_{n_k}(\lambda_k))$, причем*

$$J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_i \end{bmatrix}^{n_i \times n_i}.$$

С точностью до перестановки диагональных блоков, матрица J определяется единственным образом.

Доказательство этой теоремы можно найти в книгах по линейной алгебре; см., например, [110] или [139].

Матрица $J_m(\lambda)$ называется *жордановым блоком* с собственным значением λ алгебраической кратности m. Если λ_i является собственным значением только одного жорданова блока, причем его порядок равен 1, то говорят, что λ — *простое собственное значение*. Если все $n_i = 1$, то J — диагональная матрица. В этом случае A называют *диагонализуемой* матрицей; в противном случае, A — *дефектная* матрица. Дефектная $n \times n$-матрица имеет *менее чем* n (линейно независимых) собственных векторов, что более подробно разъясняется в приводимом ниже предложении. Хотя в некотором точно определенном смысле дефектные матрицы являются «редкими», наличие менее чем n (линейно независимых) собственных векторов у некоторых матриц есть фундаментальный факт, с которым приходится считаться всякому разработчику алгоритмов для вычисления собственных значений и собственных векторов. В разделе 4.3 мы познакомимся с некоторыми из трудностей, возникающих при работе с дефектными матрицами. Обсуждаемые в гл. 5 симметричные матрицы никогда не бывают дефектными.

Предложение 4.2. *У жорданова блока имеется только один (с точностью до умножения на ненулевое число) правый собственный вектор* $e_1 = [1, 0, \ldots, 0]^T$ *и только один левый собственный вектор* $e_n = [0, \ldots, 0, 1]^T$. *Поэтому* n *собственным значениям матрицы тогда и только тогда соответствуют* n *линейно независимых собственных векторов, когда эта матрица диагонализуема. В этом случае* $S^{-1}AS = \mathrm{diag}(\lambda_i)$, *что эквивалентно равенству* $AS = S\mathrm{diag}(\lambda_i)$; *таким образом,* i*-й столбец матрицы* S *есть правый собственный вектор для* λ_i. *Соотношение* $S^{-1}AS = \mathrm{diag}(\lambda_i)$ *эквивалентно также равенству* $S^{-1}A = \mathrm{diag}(\lambda_i)S^{-1}$, *поэтому вектор, сопряженный с* i*-й строкой матрицы* S^{-1}, *является левым собственным вектором для* λ_i. *Если все* n *собственных значений матрицы* A *различны, то* A — *диагонализуемая матрица.*

Доказательство. Чтобы упростить обозначения, положим $J = J_m(\lambda)$. Легко видеть, что $Je_1 = \lambda e_1$ и $e_n^T J = \lambda e_n^T$, поэтому e_1 и e_n суть соответственно правый и левый собственные векторы для J. Чтобы убедиться в том, что J имеет единственный (с точностью до умножения на число) правый собственный вектор, заметим, что всякий собственный вектор x должен быть решением системы $(J - \lambda I)x = 0$, т. е. принадлежать ядру матрицы

$$J - \lambda I = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}.$$

Очевидно, что ядром матрицы $J - \lambda I$ является $\mathrm{span}(e_1)$, так что действительно имеется только один собственный вектор. Если все собственные значения матрицы A различны, то каждый ее жорданов блок имеет размер 1×1, поэтому $J = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ — диагональная матрица. ◇

Пример 4.1. Проиллюстрируем понятия собственного значения и собственного вектора с помощью задачи о *механических колебаниях*. Как мы увидим, дефектные матрицы могут возникать в естественном физическом контексте.

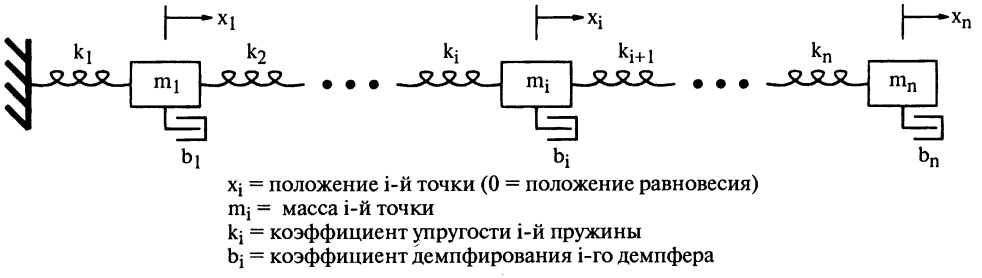


Рис. 4.1. Связанная система материальных точек с демпфированием.

Рассмотрим рис. 4.1, где изображена связанная система материальных точек с демпфированием. Эту систему мы используем для иллюстрации разнообразных задач на собственные значения.

Закон Ньютона $F = ma$ в применении к этой системе дает уравнения

$$\begin{aligned} m_i\ddot{x}_i(t) &= k_i(x_{i-1}(t) - x_i(t)) \\ &\qquad \text{воздействие пружины } i \text{ на точку } i \\ &+ k_{i+1}(x_{i+1}(t) - x_i(t)) \\ &\qquad \text{воздействие пружины } i+1 \text{ на точку } i \\ &- b_i\dot{x}_i(t) \\ &\qquad \text{воздействие демпфера } i \text{ на точку } i \end{aligned} \tag{4.2}$$

или

$$M\ddot{x}(t) = -B\dot{x}(t) - Kx(t), \tag{4.3}$$

где $M = \operatorname{diag}(m_1, \ldots, m_n)$, $B = \operatorname{diag}(b_1, \ldots b_n)$ и

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & & & \\ -k_2 & k_2 + k_3 & -k_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -k_{n-1} & k_{n-1} + k_n & -k_n \\ & & & -k_n & k_n \end{bmatrix}.$$

Предполагается, что все массы m_i положительны. Матрицу M называют *матрицей масс*, B — *матрицей демпфирования* и K — *матрицей жесткости*.

Инженеры-электротехники приходят к аналогичному уравнению при анализе линейных цепей, пользуясь вместо закона Ньютона законами Кирхгофа и некоторыми родственными правилами. В этом случае вектор x представляет токи через ветви цепи, матрица M составлена из индуктивностей, B из сопротивлений и K из полных проводимостей (величин, обратных емкостям).

Воспользуемся стандартным приемом для сведения этого дифференциального уравнения второго порядка к системе уравнений первого порядка, состоящим в замене переменной

$$y(t) = \begin{bmatrix} \dot{x}(t) \\ x(t) \end{bmatrix}.$$

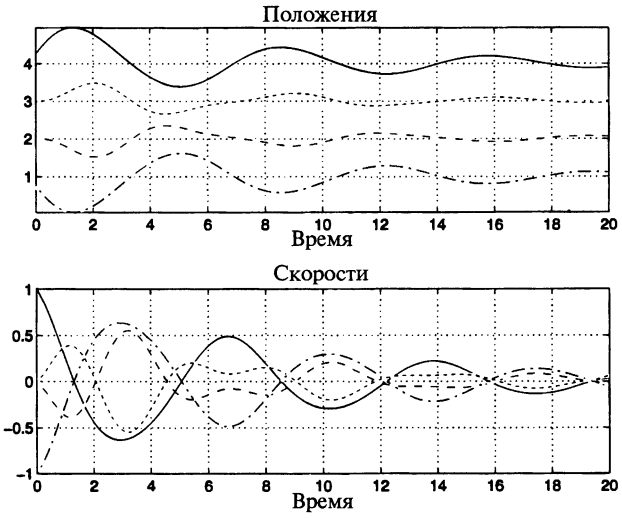


Рис. 4.2. Положения и скорости материальных точек в связанной системе с массами $m_1 = m_4 = 2$ и $m_2 = m_3 = 1$. Все коэффициенты упругости k_i равны 1. Все коэффициенты демпфирования b_i равны 4. Начальные смещения равны $x_1(0) = -0.25$, $x_2(0) = x_3(0) = 0$ и $x_4(0) = 0.25$. Начальные скорости равны $v_1(0) = -1$, $v_2(0) = v_3(0) = 0$ и $v_4(0) = 1$. Положения равновесия суть 1, 2, 3 и 4. Программа, рассчитывающая произвольные связанные системы материальных точек с выдачей соответствующих графиков, содержится в HOMEPAGE/Matlab/massprin g.m.

Это дает

$$\dot{y}(t) = \begin{bmatrix} \ddot{x}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} -M^{-1}B\dot{x}(t) - M^{-1}Kx(t) \\ \dot{x}(t) \end{bmatrix}$$
$$= \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}(t) \\ x(t) \end{bmatrix}$$
$$= \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} \cdot y(t) \equiv Ay(t). \tag{4.4}$$

При решении уравнения $\dot{y}(t) = Ay(t)$ предположим, что значение $y(0)$ задано (т. е. заданы начальные положения $x(0)$ и скорости $\dot{x}(0)$).

Данное дифференциальное уравнение можно решать с помощью представления $y(t) = e^{At}y(0)$, где e^{At} — матричная экспонента. Мы применим другой, более элементарный метод для специального случая диагонализуемой матрицы A. Впрочем, предположение о диагонализуемости выполнено для почти всех значений величин m_i, k_i и b_i. Мы вернемся к анализу остальных ситуаций позже. (Общая задача о вычислении матричных функций типа e^{At} обсуждается в разд. 4.5.1 и вопросе 4.4.)

Для диагонализуемой матрицы A можно написать $A = S\Lambda S^{-1}$, где $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Тогда соотношение $\dot{y}(t) = Ay(t)$ эквивалентно равенству $\dot{y}(t) = S\Lambda S^{-1}y(t)$, или $S^{-1}\dot{y}(t) = \Lambda S^{-1}(t)$, или $\dot{z}(t) = \Lambda z(t)$, где $z(t) \equiv S^{-1}y(t)$. Эта диагональная система дифференциальных уравнений $\dot{z}_i(t) = \lambda_i z_i(t)$ име-

ет решение $z_i(t) = e^{\lambda_i t} z_i(0)$, поэтому $y(t) = S\mathrm{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_n t}) S^{-1} y(0) = Se^{\Lambda t} S^{-1} y(0)$. Рис. 4.2 иллюстрирует решение конкретной задачи с четырьмя массами и пружинами.

Чтобы понять физический смысл недиагонализуемости матрицы A в связанной системе, рассмотрим случай системы, состоящей из единственной точки, одной пружины и одного демпфера. Дифференциальное уравнение такой системы упрощается к виду $m\ddot{x}(t) = -b\dot{x}(t) - kx(t)$, поэтому $A = \begin{bmatrix} -b/m & -k/m \\ 1 & 0 \end{bmatrix}$. Двумя собственными значениями матрицы A являются числа $\lambda_\pm = \frac{b}{2m}\left(-1 \pm \left(1 - \frac{4km}{b^2}\right)^{1/2}\right)$. Случай $\frac{4km}{b^2} < 1$ соответствует *сильному демпфированию*. В этом случае система имеет два вещественных отрицательных собственных значения с полусуммой $-\frac{b}{2m}$, а ее решение с течением времени монотонно стремится к нулю. При $\frac{4km}{b^2} > 1$ имеем *слабо затухающее демпфирование*. У системы два сопряженных комплексных собственных значения с вещественной частью $-\frac{b}{2m}$, а решение, убывая к нулю, осциллирует. В обоих этих случаях матрица системы диагонализуема, так как собственные значения различны. Случай $\frac{4km}{b^2} = 1$ соответствует *критическому демпфированию*. Оба собственных значения вещественны и равны $-\frac{b}{2m}$, а A имеет единственный жорданов 2×2-блок с этим собственным значением. Другими словами, недиагонализуемые матрицы образуют «границу» между двумя физическими режимами поведения: осцилляциями и монотонным убыванием.

Пусть A диагонализуема, но матрица S плохо обусловлена, так что хорошее приближение к S^{-1} вычислить трудно. В этом случае явная формула для решения $y(t) = Se^{\Lambda t} S^{-1} y(0)$ даст очень неточный результат, т. е., с вычислительной точки зрения, она бесполезна. В дальнейшем мы будем постоянно возвращаться к этой механической системе, поскольку она хорошо иллюстрирует многие задачи на собственные значения. ◇

Чтобы продолжить наше обсуждение канонических форм, удобно ввести следующее обобщение понятия собственного вектора:

Определение 4.4. *Подпространство* $\mathbf{X}$ *из* $\mathbb{R}^n$, *обладающее тем свойством, что из* $x \in \mathbf{X}$ *следует включение* $Ax \in \mathbf{X}$, *называется* инвариантным подпространством матрицы A. *Это определение можно записать и включением* $A\mathbf{X} \subseteq \mathbf{X}$.

Простейшим (одномерным) инвариантным подпространством является множество $\mathrm{span}(x)$ всех скалярных кратных собственного вектора x. Укажем аналогичный способ построения инвариантных подпространств большей размерности. Положим $X = [x_1, \ldots, x_m]$, где $x_1, \ldots, x_m$ — произвольная линейно независимая система собственных векторов для собственных значений $\lambda_1, \ldots, \lambda_m$. Тогда $\mathbf{X} = \mathrm{span}(X)$ есть инвариантное подпространство. Действительно, из $x \in \mathbf{X}$ следует, что $x = \sum_{i=1}^m \alpha_i x_i$ для некоторых скаляров α_i, поэтому $Ax = \sum_{i=1}^m \alpha_i A x_i = \sum_{i=1}^m \alpha_i \lambda_i x_i \in \mathbf{X}$. Если ни одно из собственных значений λ_i не равно нулю, то $A\mathbf{X}$ совпадает с $\mathbf{X}$. Описанная конструкция обобщается в следующем предложении. □

Предложение 4.3. *Пусть* A — *матрица порядка* n, *а* $X = [x_1, \ldots, x_m]$ — *произвольная* $n \times m$-*матрица с линейно независимыми столбцами. Обозначим*

через $\mathbf{X} = \mathrm{span}(X)$ *подпространство размерности* m*, натянутое на столбцы матрицы* X*. В таком случае,* $\mathbf{X}$ *тогда и только тогда является инвариантным подпространством матрицы* A*, когда найдется* $m \times m$*-матрица* B*, такая, что* $AX = XB$*. При этом* m *собственных значений матрицы* B *будут в то же время собственными значениями для* A*. (Если* $m = 1$*, то* $X = [x_1]$ *есть собственный вектор, а* B *— собственное значение.)*

Доказательство. Предположим вначале, что $\mathbf{X}$ — инвариантное подпространство. Тогда всякий вектор Ax_i принадлежит $\mathbf{X}$, а потому Ax_i должен быть линейной комбинацией базисных векторов этого подпространства: $Ax_i = \sum_{j=1}^{m} x_j b_{ji}$. Эти соотношения эквивалентны матричному равенству $AX = XB$. Обратно, равенство $AX = XB$ означает, что всякий вектор Ax_i есть линейная комбинация столбцов матрицы X, поэтому $\mathbf{X}$ — инвариантное подпространство.

Пусть теперь $AX = XB$. Возьмем какую-нибудь матрицу $\tilde{X}$ размера $n \times (n-m)$ так, чтобы матрица $\hat{X} = [X, \tilde{X}]$ была невырожденной. Тогда матрицы A и $\hat{X}^{-1}A\hat{X}$ подобны, следовательно, имеют одни и те же собственные значения. Положим $\hat{X}^{-1} = \begin{bmatrix} Y^{m\times n} \\ \tilde{Y}^{(n-m)\times n} \end{bmatrix}$, тогда из $\hat{X}^{-1}\hat{X} = I$ следует, что $YX = I$ и $\tilde{Y}X = 0$. Имеем $\hat{X}^{-1}A\hat{X} = \begin{bmatrix} Y \\ \tilde{Y} \end{bmatrix} \cdot [AX, A\tilde{X}] = \begin{bmatrix} YAX & YA\tilde{X} \\ \tilde{Y}AX & \tilde{Y}A\tilde{X} \end{bmatrix} = \begin{bmatrix} YXB & YA\tilde{X} \\ \tilde{Y}XB & \tilde{Y}A\tilde{X} \end{bmatrix} = \begin{bmatrix} B & YA\tilde{X} \\ 0 & \tilde{Y}A\tilde{X} \end{bmatrix}$. Отсюда заключаем (см. вопрос 4.1), что множество собственных значений матрицы A есть объединение множеств собственных значений матриц B и $\tilde{Y}A\tilde{X}$. ◇

Запишем, например, соотношения $S^{-1}AS = J = \mathrm{diag}(J_{n_i}(\lambda_i))$, определяющие каноническую форму Жордана, в виде $AS = SJ$, где $S = [S_1, S_2, \ldots, S_k]$ и матрица S_i имеет n_i столбцов (столько же, сколько и $J_{n_i}(\lambda_i)$; по поводу обозначений см. теорему 4.1). Тогда из $AS = SJ$ выводим, что $AS_i = S_i J_{n_i}(\lambda_i)$, т. е. подпространства $\mathrm{span}(S_i)$ являются инвариантными подпространствами матрицы A.

Жорданова форма дает нам полную информацию о собственных значениях, собственных векторах и инвариантных подпространствах матрицы. Кроме того, имеются основанные на жордановой форме явные формулы для вычисления e^A или любой другой функции от матрицы (см. разд. 4.5.1). Однако реальное вычисление жордановой формы затруднено по двум причинам:

Первая причина: жорданова форма не является непрерывной функцией от A, поэтому ошибки округлений могут совершенно изменить ее.

Пример 4.2. Пусть матрица уже имеет жорданову форму

$$J_n(0) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}.$$

Взяв произвольно малое ε, добавим число $i \cdot \varepsilon$ к элементу (i,i) для $i = 1, \ldots, n$. Тогда собственными значениями станут m различных чисел $i \cdot \varepsilon$, поэтому жорданова форма изменится от $J_n(0)$ к $\operatorname{diag}(\varepsilon, 2\varepsilon, \ldots, n\varepsilon)$. ◇

Вторая причина: в общем случае, жорданова форма не может быть вычислена устойчиво. Иначе говоря, для вычисленных S и J мы не можем гарантировать, что при некоторой малой матрице δA имеет место равенство $S^{-1}(A + \delta A)S = J$.

Пример 4.3. Предположим, что равенство $S^{-1}AS = J$ выполнено точно и матрица S обусловлена очень плохо (т. е. число $\kappa(S) = \|S\| \cdot \|S^{-1}\|$ очень велико). Пусть нам очень повезло и матрицу S удалось вычислить *точно*, а J — лишь с очень малой ошибкой δJ, так что $\|\delta J\| = O(\varepsilon)\|A\|$. Насколько велика может быть обратная ошибка? Другими словами, насколько велика может быть матрица δA, такая, что $S^{-1}(A + \delta A)S = J + \delta J$? Имеем $\delta A = S\delta J S^{-1}$, поэтому можно лишь сказать, что $\|\delta A\| \leq \|S\| \cdot \|\delta J\| \cdot \|S^{-1}\| = O(\varepsilon)\kappa(S)\|A\|$. Итак, $\|\delta A\|$ может быть много больше, чем $\varepsilon\|A\|$, что исключает свойство обратной устойчивости. ◇

Вместо того чтобы работать с соотношением $S^{-1}AS = J$, где S может быть как угодно плохо обусловленной матрицей, мы для обеспечения устойчивости будем брать S из множества ортогональных матриц (для которых $\kappa_2(S) = 1$). При таком ограничении мы не сможем получить столь простую каноническую форму, как форма Жордана, однако форма, которую мы найдем, почти столь же хороша.

Теорема 4.2 *(каноническая форма Шура). Для произвольной матрицы A найдутся унитарная матрица Q и верхнетреугольная матрица T, такие, что $Q^*AQ = T$. Собственными значениями матрицы A являются диагональные элементы матрицы T.*

Доказательство. Применим индукцию по n. Очевидно, что утверждение теоремы верно для 1×1-матрицы A. Пусть теперь λ — произвольное собственное значение, а u — соответствующий собственный вектор, нормированный так, чтобы $\|u\|_2 = 1$. Возьмем $\tilde{U}$ таким образом, чтобы $U = [u, \tilde{U}]$ была квадратной унитарной матрицей. (Отметим, что λ и u могут быть комплексными даже для вещественной матрицы A.) Имеем

$$U^* \cdot A \cdot U = \begin{bmatrix} u^* \\ \tilde{U}^* \end{bmatrix} \cdot A \cdot [u, \tilde{U}] = \begin{bmatrix} u^*Au & u^*A\tilde{U} \\ \tilde{U}^*Au & \tilde{U}^*A\tilde{U} \end{bmatrix}.$$

Теперь, как и в доказательстве предложения 4.3, можем написать $u^*Au = \lambda u^*u = \lambda$ и $\tilde{U}^*Au = \lambda\tilde{U}^*u = 0$, поэтому $U^*AU \equiv \begin{bmatrix} \lambda & \tilde{a}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix}$. По индуктивному предположению, существует такая унитарная матрица P, что матрица $P^*A_{22}P = \tilde{T}$ верхняя треугольная. Тогда

$$U^*AU = \begin{bmatrix} \lambda & \tilde{a}_{12} \\ 0 & P\tilde{T}P^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \lambda & \tilde{a}_{12}P \\ 0 & \tilde{T} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P^* \end{bmatrix},$$

поэтому матрица

$$\begin{bmatrix} 1 & 0 \\ 0 & P^* \end{bmatrix} U^*AU \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} \lambda & \tilde{a}_{12}P \\ 0 & \tilde{T} \end{bmatrix} = T$$

верхняя треугольная, а $Q = U \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix}$ — унитарная матрица, что и требовалось. ◇

Заметим, что форма Шура не единственна, поскольку собственные значения могут располагаться на диагонали матрицы T в произвольном порядке.

Описанная нами процедура вводит комплексные числа даже тогда, когда матрица A вещественна. Однако для вещественной A мы предпочитаем каноническую форму, которая сама вещественна, потому что такую форму дешевле вычислять. Как отмечено в начале данного параграфа, это означает, что мы должны пожертвовать *треугольной* канонической формой и согласиться на использование *блочно-треугольной* формы.

Теорема 4.3 *(вещественная каноническая форма Шура). Для произвольной вещественной матрицы A найдется вещественная ортогональная матрица V такая, что матрица $V^TAV = T$ верхняя квазитреугольная. Это означает, что T — верхняя блочно-треугольная матрица с диагональными блоками размеров 1×1 и 2×2. Собственными значениями матрицы являются собственные значения ее диагональных блоков. Блоки размера 1×1 соответствуют вещественным собственным значениям, а блоки размера 2×2 — сопряженным парам комплексных собственных значений.*

Доказательство. Как и выше, применим рассуждение по индукции. Если собственное значение λ вещественно, то ему соответствует вещественный собственный вектор. В этом случае доказательство проводится, как в предыдущей теореме. Для комплексного λ обозначим через u соответствующий (необходимо) комплексный собственный вектор: $Au = \lambda u$. Так как $\overline{Au} = A\overline{u} = \overline{\lambda}\overline{u}$, то $\overline{\lambda}$ и $\overline{u}$ также составляют собственную пару. Обозначим через $u_R = \frac{1}{2}u + \frac{1}{2}\overline{u}$ и $u_I = \frac{1}{2i}u - \frac{1}{2i}\overline{u}$ вещественную и мнимую части вектора u. Тогда $\mathrm{span}\{u_R, u_I\} = \mathrm{span}\{u, \overline{u}\}$ есть двумерное инвариантное подпространство матрицы A. Положим $\tilde{U} = [u_R, u_I]$ и пусть $\tilde{U} = QR$ есть QR-разложение матрицы $\tilde{U}$. Итак, подпространство $\mathrm{span}\{Q\} = \mathrm{span}\{u_R, u_I\}$ инвариантно относительно A. Выберем $\tilde{Q}$ так, чтобы матрица $U = [Q, \tilde{Q}]$ была квадратной и ортогональной. Имеем

$$U^T \cdot A \cdot U = \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} \cdot A \cdot [Q, \tilde{Q}] = \begin{bmatrix} Q^TAQ & Q^TA\tilde{Q} \\ \tilde{Q}^TAQ & \tilde{Q}^TA\tilde{Q} \end{bmatrix}.$$

Поскольку Q определяет инвариантное подпространство, найдется 2×2-матрица B, такая, что $AQ = QB$. Теперь, как и в доказательстве предложения 4.3, можем написать $Q^TAQ = Q^TQB = B$ и $\tilde{Q}^TAQ = Q^TQB = 0$, так что $U^TAU = \begin{bmatrix} B & Q^TA\tilde{Q} \\ 0 & \tilde{Q}^TA\tilde{Q} \end{bmatrix}$. Остается применить индуктивное предположение к матрице $\tilde{Q}^TA\tilde{Q}$. ◇

4.2.1. Вычисление собственных векторов с помощью формы Шура

Пусть $Q^*AQ = T$ есть форма Шура матрицы A. Если $Tx = \lambda x$, то $AQx = QTx = \lambda Qx$, т. е. Qx — собственный вектор матрицы A. Таким образом, чтобы найти собственные векторы для A, достаточно найти их для T.

Предположим, что кратность собственного значения $\lambda = t_{ii}$ равна 1 (т. е. λ — простое собственное значение). Запишем равенство $(T - \lambda I)x = 0$ в виде

$$0 = \begin{bmatrix} T_{11} - \lambda I & T_{12} & T_{13} \\ 0 & 0 & T_{23} \\ 0 & 0 & T_{33} - \lambda I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$
$$= \begin{bmatrix} (T_{11} - \lambda I)x_1 + T_{12}x_2 + T_{13}x_3 \\ T_{23}x_3 \\ (T_{33} - \lambda I)x_3 \end{bmatrix}.$$

Здесь блоки T_{11}, T_{22} и T_{33} имеют соответственно размеры $(i-1) \times (i-1)$, 1×1 и $(n-i) \times (n-i)$; разбиение вектора x согласовано с разбиением матрицы T. Поскольку λ — простое собственное значение, матрицы $T_{11} - \lambda I$ и $T_{33} - \lambda I$ невырожденны. Поэтому из соотношения $(T_{33} - \lambda I)x_3 = 0$ следует, что $x_3 = 0$. Отсюда $(T_{11} - \lambda I)x_1 = -T_{12}x_2$. Если положить, например, $x_2 = 1$, то $x_1 = -(T_{11} - \lambda I)^{-1}T_{12}$. Следовательно,

$$x = \begin{bmatrix} (\lambda I - T_{11})^{-1}T_{12} \\ 1 \\ 0 \end{bmatrix}.$$

Иначе говоря, нужно лишь решить треугольную систему уравнений относительно x_1. Чтобы определить из вещественной формы Шура *вещественный* вектор, понадобится решить квазитреугольную систему. Вычисление комплексных собственных векторов из вещественной формы Шура с использованием только вещественной арифметики также связано с решением линейных систем, но проводится несколько более хитрым образом. Подробности можно найти в LAPACK-программе `strevc`.

4.3. Теория возмущений

В этом разделе мы постараемся выяснить, когда собственные значения являются плохо обусловленными и, следовательно, трудно вычислимыми. Мы не только выведем оценки ошибок в вычисленных собственных значениях, но и свяжем числа обусловленности собственных значений с родственными характеристиками, такими, как число обусловленности матрицы из собственных векторов или расстояние до ближайшей матрицы с *бесконечно* плохо обусловленным собственным значением.

Начнем наш анализ с вопроса о том, когда числа обусловленности собственных значений могут быть бесконечны. Как показывает приводимый ниже пример, такова ситуация с кратными собственными значениями.

Пример 4.4. Пусть матрица

$$A = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & & 1 \\ \varepsilon & & & 0 \end{bmatrix}$$

имеет порядок n. Тогда $\lambda^n - \varepsilon = 0$ есть характеристическое уравнение, так что $\lambda = \sqrt[n]{\varepsilon}$ (все n значений радикала). Для малых ε корень n-й степени из

ε значительно больше, чем величина порядка ε. Более формально, число обусловленности бесконечно, так как $\frac{d\lambda}{d\varepsilon} = \frac{\varepsilon^{\frac{1}{n}-1}}{n} = \infty$ в точке $\varepsilon = 0$, если $n \geq 2$. Возьмем, например, $n = 16$ и $\varepsilon = 10^{-16}$. Тогда для каждого собственного значения матрицы A имеем $|\lambda| = .1$. ◇

Итак, следует ожидать, что число обусловленности собственного значения велико, если оно «почти кратно», т. е. найдется малое возмущение δA, такое, что матрица $A + \delta A$ имеет в точности кратное собственное значение. Однако, даже если число обусловленности собственного значения бесконечно, это не означает, что ни одного верного знака в этом значении определить не удастся.

Предложение 4.4. *Собственные значения являются непрерывными функциями матрицы, хотя и не всегда дифференцируемы.*

Доказательство. Достаточно доказать непрерывность корней многочлена, поскольку коэффициенты характеристического многочлена суть непрерывные (даже полиномиальные) функции от элементов матрицы. Используем принцип аргумента из комплексного анализа [2]: число корней многочлена p, находящихся внутри контура γ, равно $\frac{1}{2\pi i}\oint_\gamma \frac{p'(z)}{p(z)}dz$. При малом изменении p мало изменится и $\frac{p'(z)}{p(z)}$, а потому мало изменится и $\frac{1}{2\pi i}\oint_\gamma \frac{p'(z)}{p(z)}dz$. Так как эта величина — целочисленная, она не изменит, в действительности, своего значения, а потому не изменится и число корней внутри γ. Это означает, что корни не могут оказаться снаружи γ, как бы мал ни был этот контур, если возмущение многочлена p достаточно мало. Поэтому корни должны быть непрерывными функциями коэффициентов многочлена. ◇

Теперь мы сосредоточимся на вычислении числа обусловленности простого собственного значения. Пусть λ — простое собственное значение матрицы A, а возмущение δA мало; тогда мы можем опознать собственное значение $\lambda + \delta\lambda$ матрицы $A + \delta A$, «соответствующее» числу λ: это попросту собственное значение, ближайшее к λ. Число обусловленности простого собственного значения легко может быть найдено.

Теорема 4.4. *Пусть λ — простое собственное значение матрицы A, а x и y — соответствующие правый и левый собственные векторы, нормированные так, что $\|x\|_2 = \|y\|_2 = 1$. Пусть $\lambda + \delta\lambda$ — соответствующее собственное значение матрицы $A + \delta A$. Тогда*

$$\delta\lambda = \frac{y^*\delta A x}{y^* x} + O(\|\delta A\|^2),$$

$$|\delta\lambda| \leq \frac{\|\delta A\|}{|y^* x|} + O(\|\delta A\|^2) = \sec\Theta(y,x)\|\delta A\| + O(\|\delta A\|^2),$$

где $\Theta(y,x)$ — острый угол между векторами y и x. Другими словами, $\sec\Theta(y,x) = 1/|y^*x|$ есть число обусловленности собственного значения λ.

Доказательство. Вычитая равенство $Ax = \lambda x$ из $(A + \delta A)(x + \delta x) = (\lambda + \delta\lambda)(x + \delta bx)$, получаем

$$A\delta x + \delta A x + \delta A\delta x = \lambda\delta x + \delta\lambda x + \delta\lambda\delta x.$$

Отбрасывая члены второго порядка (т. е. те,что содержат два «δ-члена» как множители: $\delta A \delta x$ и $\delta\lambda\delta x$) и умножая обе части на y^*, имеем $y^*A\delta x + y^*\delta A x = y^*\lambda\delta x + y^*\delta\lambda x$.

Сокращая $y^*A\delta x$ и $y^*\lambda\delta x$, получаем требуемую формулу $\delta\lambda = (y^*\delta A x)/(y^*x)$. ◇

Заметим, что для жорданова блока e_1 и e_n являются соответственно правым и левым собственными векторами, поэтому число обусловленности собственного значения равно $1/|e_n^* e_1| = 1/0 = \infty$, что согласуется с нашим предыдущим анализом.

В качестве противоположного примера рассмотрим важный специальный случай симметричных матриц. Здесь число обусловленности равно 1, так что собственные значения с большой точностью определяются элементами матрицы.

Следствие 4.1. Пусть A — симметричная (или, более общо, нормальная) матрица (нормальность означает, что $AA^* = A^*A$). Тогда $|\delta\lambda| \le \|\delta A\| + O(\|\delta A\|^2)$.

Доказательство. Для симметричной или нормальной матрицы A собственные векторы попарно ортогональны, т. е. $Q^*AQ = \Lambda$ и $QQ^* = I$. Поэтому правые собственные векторы x (т. е. столбцы матрицы Q) совпадают с левыми собственными векторами y (которые сопряжены со строками матрицы Q^*); следовательно, $1/|y^*x| = 1$. □

Для экспериментальной проверки обусловленности собственных значений можно использовать Matlab-программу, обсуждаемую в вопросе 4.14.

Позднее мы покажем (см. теорему 5.1), что если $\delta A = \delta A^T$, то, в действительности, справедлива оценка $|\delta\lambda| \le \|\delta A\|_2$ независимо от величины числа $\|\delta A\|_2$.

Теорема 4.4 полезна лишь при достаточно малых $\|\delta A\|$. Мы можем устранить в оценке член $O(\|\delta A\|^2)$ и получить простую теорему, верную для возмущений $\|\delta A\|$ произвольной величины, ценой введения дополнительного множителя n в правую часть оценки.

Теорема 4.5 (Бауэр—Файк). *Пусть все собственные значения матрицы A простые (т. е. A — диагонализуемая матрица). Обозначим их через λ_i, и пусть x_i и y_i — соответствующие правые и левые собственные векторы, нормированные так, что $\|x_i\|_2 = \|y_i\| = 1$. Тогда собственные значения матрицы $A + \delta A$ находятся в кругах B_i с центрами λ_i и радиусами $n\frac{\|\delta A\|_2}{|y_i^* x_i|}$.*

Наше доказательство опирается на теорему Гершгорина (теорема 2.9), формулировку которой мы здесь повторим.

Теорема Гершгорина. *Пусть B — произвольная матрица. Тогда ее собственные значения принадлежат объединению n кругов, определяемых неравенствами $|\lambda - b_{ii}| \le \sum_{j\ne i} |b_{ij}|$ $(i = 1, \ldots, n)$.*

Нам понадобятся еще две простые леммы.

Лемма 4.1. *Пусть* $S = [x_1, \ldots, x_n]$ *есть невырожденная матрица правых собственных векторов матрицы* A. *Тогда*

$$S^{-1} = \begin{bmatrix} y_1^*/y_1^* x_1 \\ y_2^*/y_2^* x_2 \\ \vdots \\ y_n^*/y_n^* x_n \end{bmatrix}.$$

Доказательство. Поскольку столбцы x_i матрицы S суть собственные векторы, то $AS = S\Lambda$, где $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Это эквивалентно равенству $S^{-1}A = \Lambda S^{-1}$, поэтому строки матрицы S^{-1} сопряжены с левыми собственными векторами y_i. Итак,

$$S^{-1} = \begin{bmatrix} y_1^* \cdot c_1 \\ \vdots \\ y_n^* \cdot c_n \end{bmatrix}$$

для некоторых констант c_i. Но $I = S^{-1}S$, поэтому $1 = (S^{-1}S)_{ii} = y_i^* x_i \cdot c_i$, откуда $c_{ii} = \frac{1}{y_i^* x_i}$, что и требовалось. ◇

Лемма 4.2. *Если 2-норма каждого столбца (произвольной) матрицы* S *равна 1, то* $\|S\|_2 \le \sqrt{n}$. *Аналогично, если 2-норма каждой строки равна 1, то 2-норма всей матрицы не превосходит* $\sqrt{n}$.

Доказательство леммы. $\|S\|_2 = \|S^T\|_2 = \max_{\|x\|_2=1} \|S^T x_2\|$. Согласно неравенству Коши–Шварца, каждая компонента вектора $S^T x$ ограничена по абсолютной величине единицей. Поэтому $\|S^T x\|_2 \le \|[1, \ldots, 1]^T\|_2 = \sqrt{n}$. ◇

Доказательство теоремы Бауэра–Файка. Приме́ним теорему Гершгорина к соотношению $S^{-1}(A + \delta A)S = \Lambda + F$, где $\Lambda = S^{-1}AS = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, а $F = S^{-1}\delta A S$. Идея состоит в том, чтобы показать, что собственные значения матрицы $A + \delta A$ находятся в кругах с центрами λ_i и радиусами, указанными в формулировке теоремы. Чтобы сделать это, возьмем определяемые теоремой Гершгорина круги, содержащие собственные значения матрицы $\Lambda + F$:

$$|\lambda - (\lambda_i + f_{ii})| \le \sum_{j \ne i} |f_{ij}|.$$

Немного расширяя их, получим круги

$$\begin{aligned} |\lambda - \lambda_i| &\le \sum_j |f_{ij}| \\ &\le n^{1/2} \cdot \left(\sum_j |f_{ij}|^2 \right)^{1/2} \qquad \text{по неравенству Коши–Шварца} \\ &= n^{1/2} \cdot \|F(i,:)\|_2. \end{aligned} \tag{4.5}$$

Теперь дадим оценку для 2-нормы i-й строки $F(i,:)$ матрицы $F = S^{-1}\delta A S$:

$$\begin{aligned}\|F(i,:)\|_2 &= \|(S^{-1}\delta A S)(i,:)\|_2 \\ &\le \|(S^{-1})(i,:)\|_2 \cdot \|\delta A\|_2 \cdot \|S\|_2 \quad \text{по лемме 1.7} \\ &\le \frac{n^{1/2}}{|y_i^* x_i|} \cdot \|\delta A\|_2 \quad \text{согласно леммам 4.1 и 4.2.}\end{aligned}$$

Эта оценка вместе с (4.5) доказывает теорему. ◇

Мы не хотим оставить у читателя впечатление, что кратные собственные значения вовсе не поддаются вычислению, поскольку имеют бесконечные числа обусловленности. В действительности, мы ожидаем, что в вычисленных приближениях будет верна некоторая *доля* разрядов (в отличие от потери некоторого фиксированного числа разрядов при вычислении простых собственных значений). В качестве иллюстрации, рассмотрим 2×2-матрицу с двойным собственным значением 1: $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Заменяя значение 0 ее (наиболее чувствительного) элемента (2, 1) на машинное эпсилон ε, мы изменяем собственные значения с 1 на $1 \pm \sqrt{\varepsilon}$. Иными словами, возмущенные собственные значения совпадают с точными лишь в половине своих разрядов. Для тройного собственного значения следует ожидать, что верными будут около трети разрядов. Аналогичные эмпирические правила верны при бóльших кратностях (см. вопрос 1.20).

Обратимся теперь к геометрическому свойству числа обусловленности, которое наблюдается и в других задачах. Вспомним, что число обусловленности по отношению к задаче обращения, т. е. число $\|A\| \cdot \|A^{-1}\|$, обладает следующим свойством: число, обратное к нему, измеряет расстояние до ближайшей вырожденной матрицы, т. е. матрицы с бесконечным числом обусловленности (см. теорему 2.1). Аналогичный факт справедлив для собственных значений. Поскольку кратные собственные значения имеют бесконечные числа обусловленности, множество матриц с кратными собственными значениями играет ту же роль при вычислении собственных значений, какую вырожденные матрицы играли в задаче обращения, где «быть почти-вырожденной» означало плохую обусловленность.

Теорема 4.6. *Пусть λ — простое собственное значение матрицы A. Пусть x и y — соответствующие нормированные собственные векторы, правый и левый, а $c = 1/|y^*x|$ есть число обусловленности λ. Тогда найдется возмущение δA такое, что λ является кратным собственным значением матрицы $A + \delta A$, причем*

$$\frac{\|\delta A\|_2}{\|A\|_2} \le \frac{1}{\sqrt{c^2 - 1}}.$$

Если $c \gg 1$, т. е. собственное значение плохо обусловлено, то эта верхняя граница для расстояния ведет себя как $1/\sqrt{c^2-1} \approx 1/c$, т. е. как величина, обратная числу обусловленности.

Доказательство. Покажем прежде всего, что без ограничения общности матрицу A можно считать верхней треугольной (имеющей форму Шура) с эле-

ментом $a_{11} = \lambda$. Действительно, преобразование A в форму Шура равносильно замене A матрицей $T = Q^*AQ$, где Q — унитарная матрица. Если x и y — собственные векторы для A, то Q^*x и Q^*y — собственные векторы для T. Поскольку $(Q^*y)^*(Q^*x) = y^*QQ^*x = y^*x$, то переход к форме Шура не меняет числа обусловленности собственного значения λ. (По-другому, это же можно сказать так: число обусловленности есть секанс угла $\Theta(x, y)$ между x и y; при замене x на Q^*x и y на Q^*y векторы x и y поворачиваются одинаковым образом, поэтому угол между ними не меняется.)

Итак, без потери общности, можно считать, что $A = \begin{bmatrix} \lambda & A_{12} \\ 0 & A_{22} \end{bmatrix}$. Тогда $x = e_1$, а y параллелен вектору $\tilde{y} = [1, A_{12}(\lambda I - A_{22})^{-1}]^*$; иначе говоря, $y = \tilde{y}/\|\tilde{y}\|_2$. Таким образом,

$$c = \frac{1}{|y^*x|} = \frac{\|\tilde{y}\|_2}{|\tilde{y}^*x|} = \|\tilde{y}\|_2 = (1 + \|A_{12}(\lambda I - A_{22})^{-1}\|_2^2)^{1/2}$$

или

$$\sqrt{c^2 - 1} = \|A_{12}(\lambda I - A_{22})^{-1}\|_2 \le \|A_{12}\|_2 \cdot \|(\lambda I - A_{22})^{-1}\|_2 \le \frac{\|A\|_2}{\sigma_{\min}(\lambda I - A_{22})}.$$

По определению наименьшего сингулярного числа, найдется возмущение δA_{22} с нормой $\|\delta A_{22}\|_2 = \sigma_{\min}(\lambda I - A_{22})$, такое,что матрица $A_{22} + \delta A_{22} - \lambda I$ вырожденна, т. е. λ есть собственное значение матрицы $A_{22} + \delta A_{22}$. Поэтому λ является двойным собственным значением для матрицы $\begin{bmatrix} \lambda & A_{12} \\ 0 & A_{22} + \delta A_{22} \end{bmatrix}$, причем

$$\|\delta A_{22}\|_2 = \sigma_{\min}(\lambda I - A_{22}) \le \frac{\|A\|_2}{\sqrt{c^2 - 1}},$$

что и требовалось. ◇

В заключение, установим соотношения между числами обусловленности собственных значений и минимумом числа обусловленности $\|S\| \cdot \|S^{-1}\|$ по всем матрицам S, диагонализующим A, т. е. $S^{-1}AS = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Приводимая ниже теорема утверждает, что если какое-либо собственное значение имеет большое число обусловленности, то число обусловленности матрицы S должно быть примерно столь же большим. Иначе говоря, числа обусловленности при вычислении (хуже всего обусловленного) собственного значения и приведении матрицы к диагональной форме приблизительно одинаковы.

Теорема 4.7. *Пусть диагонализуемая матрица A имеет собственные значения λ_i и соответствующие собственные векторы (правые и левые) x_i и y_i; последние нормированы так, что $\|x_i\|_2 = \|y_i\|_2 = 1$. Пусть матрица S такова, что $S^{-1}AS = \Lambda = \mathrm{diag}(\lambda_1, \ldots \lambda_n)$. Тогда $\|S\|_2 \cdot \|S^{-1}\|_2 \ge \max_i 1/|y_i^*x_i|$. Если положить $S = [x_1, \ldots x_n]$, то $\|S\|_2 \cdot \|S^{-1}\|_2 \le n \cdot \max_i 1/|y_i^*x_i|$, т. е. число обусловленности такой матрицы S не более чем в n раз превышает минимальное возможное значение.*

Доказательство этого утверждения можно найти в [69].

В главе 4 руководства для пользователей LAPACK'а дан обзор чисел обусловленности для спектральных задач; в него включены, в частности, собственные векторы, инвариантные подпространства и собственные значения, соответствующие указанному инвариантному подпространству; см. также [161, 237]. Алгоритмы, вычисляющие эти числа обусловленности, реализованы LAPACK-подпрограммами `strsna` и `strsen`; они могут быть активированы и вызовом программ-драйверов `sgeevx` и `sgeesx`.

4.4. Алгоритмы для несимметричной проблемы собственных значений

Мы придем к нашему окончательному алгоритму, а именно хессенбергову QR-алгоритму со сдвигами, отправляясь от более простых методов. Для простоты изложения, будем считать, что матрица A вещественна.

Нашим первым и простейшим алгоритмом будет *степенной метод* (разд. 4.4.1). Он способен находить лишь собственное значение матрицы A, имеющее наибольший модуль, и соответствующий собственный вектор. Чтобы определить другие собственные значения и собственные векторы, степенной метод применяется к матрице $(A - \sigma I)^{-1}$ для некоторого *сдвига* σ. Этот алгоритм называется *обратной итерацией* (разд. 4.4.2); заметим, что собственным значением с наибольшим модулем матрицы $(A - \sigma I)^{-1}$ является число $1/(\lambda_i - \sigma)$, где λ_i — собственное значение матрицы A, ближайшее к σ. Поэтому, выбирая σ, мы можем управлять тем, какие собственные значения будут вычисляться. Другое улучшение степенного метода позволит нам одновременно вычислять целое инвариантное подпространство, а не только отдельный собственный вектор; соответствующий алгоритм называется *ортогональной итерацией* (разд. 4.4.3). Наконец, мы модифицируем ортогональную итерацию так, чтобы вместо A ее было удобно применять к матрице $(A - \sigma I)^{-1}$. Эту модификацию называют QR-итерацией (разд. 4.4.4).

С математической точки зрения, QR-итерация (со сдвигом σ) и есть наш окончательный алгоритм. Однако, чтобы превратить ее в быстрый и надежный на практике метод, остается решить еще несколько проблем (см. разд. 4.4.5). В разделе 4.4.6 обсуждается первое преобразование, предназначенное для того, чтобы ускорить QR-итерацию, а именно приведение исходной плотной матрицы A к *верхней форме Хессенберга* (где ненулевые элементы располагаются только на первой поддиагонали и выше ее). В последующих разделах описана эффективная реализация QR-итерации для верхних хессенберговых матриц. (В разделе 4.4.7 показано, как упрощается верхняя форма Хессенберга для симметричной проблемы собственных значений и при вычислении SVD.)

4.4.1. Степенной метод

Алгоритм 4.1. *Степенной метод: для заданного вектора x_0 выполнять итерации*

$i = 0$
repeat
  $y_{i+1} = Ax_i$

$x_{i+1} = y_{i+1}/\|y_{i+1}\|_2$ *(приближенный собственный вектор)*
$\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$ *(приближенное собственное значение)*
$i = i + 1$
пока метод не сойдется

Применим этот алгоритм вначале к очень простому случаю, когда $A = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, причем $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. В этом случае собственными векторами будут столбцы e_i единичной матрицы. Заметим, что вектор x_i может быть записан и как $x_i = A^i x_0/\|A^i x_0\|_2$, так как множители $1/\|y_{i+1}\|_2$ лишь масштабируют x_{i+1} к единичной длине, но не меняют его направления. Имеем

$$A^i x_0 \equiv A^i \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} \xi_1 \lambda_1^i \\ \xi_2 \lambda_2^i \\ \vdots \\ \xi_n \lambda_n^i \end{bmatrix} = \xi_1 \lambda_1^i \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^i \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^i \end{bmatrix},$$

где сделано предположение $\xi_1 \neq 0$. Поскольку все дроби λ_j/λ_1 меньше 1 по абсолютной величине, вектор $A^i x_0$ становится все более параллелен вектору e_1, а потому вектор $x_i = A^i x_0/\|A^i x_0\|_2$ все более и более приближается к $\pm e_1$, т. е. к собственному вектору, соответствующему старшему собственному значению λ_1. Скорость сходимости зависит от того, насколько малы отношения $|\lambda_2/\lambda_1| \geq \cdots \geq |\lambda_n/\lambda_1|$: чем они меньше, тем сходимость быстрее. Поскольку векторы x_i сходятся к $\pm e_1$, числа $\tilde{\lambda}_i = x_i^T A x_i$ сходятся к старшему собственному значению λ_1.

В исследовании сходимости степенного метода было сделано несколько предположений, и прежде всего, о том, что матрица A диагональная. Переходя к более общему случаю, будем считать, что $A = S\Lambda S^{-1}$ — диагонализуемая матрица, причем $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ и собственные значения занумерованы так, что $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Обозначив через s_i соответствующие нормированные ($\|s_i\|_2 = 1$) собственные векторы, положим $S = [s_1, \ldots, s_n]$ (в предыдущем анализе мы имели $S = I$). Это позволяет написать $x_0 = S(S^{-1}x_0) \equiv S([\xi_1, \ldots, \xi_n]^T)$. Из равенства $A = S\Lambda S^{-1}$ следует, что

$$A_i = \underbrace{(S\Lambda S^{-1}) \cdots (S\Lambda S^{-1})}_{i \text{ раз}} = S\Lambda^i S^{-1},$$

так как все пары $S^{-1} \cdot S$ можно удалить из произведения. Окончательно получаем

$$A^i x_0 = (S\Lambda^i S^{-1}) S \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = S \begin{bmatrix} \xi_1 \lambda_1^i \\ \xi_2 \lambda_2^i \\ \vdots \\ \xi_n \lambda_n^i \end{bmatrix} = \xi_1 \lambda_1^i S \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^i \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^i \end{bmatrix}.$$

Как и прежде, вектор в квадратных скобках сходится к e_1, поэтому вектор $A^i x_0$ становится все более параллелен вектору $Se_1 = s_1$, т. е. собственному вектору для λ_1. Следовательно, $\tilde{\lambda}_i = x_i^T A x_i$ сходится к $s_1^T A s_1 = s_1^T \lambda_1 s_1 = \lambda_1$.

Небольшим изъяном метода является предположение $\xi_1 \neq 0$, т. е. вектор x_0 не должен принадлежать инвариантному подпространству $\mathrm{span}\{s_2, \ldots, s_n\}$. Это предположение с очень большой вероятностью выполняется при случайном выборе x_0. Крупными же недостатками следует считать: сходимость только к собственному значению с наибольшим модулем и соответствующему собственному вектору; зависимость скорости этой сходимости от величины $|\lambda_2/\lambda_1|$, которая может быть близка к 1, что влечет за собой очень медленную сходимость. В действительности, если для вещественной матрицы A собственное значение с наибольшим модулем является комплексным, то имеется пара сопряженных комплексных собственных значений λ_1 и λ_2 с таким модулем: $|\lambda_1| = |\lambda_2|$. В этом случае проведенный выше анализ не работает вовсе. В предельной ситуации ортогональной матрицы *все* собственные значения имеют один и тот же модуль 1.

С помощью программы, помещенной на HOMEPAGE/Matlab/powerplot.m, можно получить графическое представление сходимости степенного метода.

4.4.2. Обратная итерация

Мы преодолеем только что указанные недостатки, применяя степенной метод не к A, а к матрице $(A - \sigma I)^{-1}$, где число σ называется *сдвигом*. Это позволит нам получить сходимость не к λ_1, а к собственному значению, ближайшему к σ. Такой метод называется обратной итерацией, или обращенным степенным методом.

Алгоритм 4.2. *Обратная итерация: для заданного вектора x_0 выполнять итерации*

$i = 0$
repeat
  $y_{i+1} = (A - \sigma I)^{-1} x_i$
  $x_{i+1} = y_{i+1} / \|y_{i+1}\|_2$ *(приближенный собственный вектор)*
  $\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$ *(приближенное собственное значение)*
  $i = i + 1$
пока метод не сойдется

Для анализа сходимости отметим, что из $A = S\Lambda S^{-1}$ следует $A - \sigma I = S(\Lambda - \sigma I)S^{-1}$, откуда $(A - \sigma I)^{-1} = S(\Lambda - \sigma I)^{-1} S^{-1}$. Таким образом, матрица $(A - \sigma I)^{-1}$ имеет те же собственные векторы s_i, что и A, а соответствующие собственные значения равны $((\Lambda - \sigma I)^{-1})_{jj} = (\lambda_j - \sigma)^{-1}$. Те же рассуждения, что и выше, показывают, что x_i должен сходиться к собственному вектору для собственного значения с наибольшим модулем. Более подробно, предположим, что число $|\lambda_k - \sigma|$ меньше всех прочих $|\lambda_i - \sigma|$, тогда собственным значением с наибольшим модулем для матрицы $(A - \sigma I)^{-1}$ будет число $(\lambda_k - \sigma)^{-1}$. Как

и выше, введем запись $x_0 = S[\xi_1, \ldots, \xi_n]^T$ и предположим, что $\xi_k \neq 0$. Тогда

$$(A-\sigma I)^{-i}x_0 = (S(\Lambda-\sigma I)^{-i}S^{-1})S\begin{bmatrix}\xi_1\\ \xi_2\\ \vdots\\ \xi_n\end{bmatrix} = S\begin{bmatrix}\xi_1(\lambda_1-\sigma)^{-i}\\ \vdots\\ \xi_n(\lambda_n-\sigma)^{-i}\end{bmatrix}$$

$$= \xi_k(\lambda_k-\sigma)^{-i}S\begin{bmatrix}\frac{\xi_1}{\xi_k}\left(\frac{\lambda_k-\sigma}{\lambda_1-\sigma}\right)^i\\ \vdots\\ 1\\ \vdots\\ \frac{\xi_n}{\xi_k}\left(\frac{\lambda_k-\sigma}{\lambda_n-\sigma}\right)^i\end{bmatrix}.$$

У вектора в квадратных скобках 1 является k-й компонентой. Так как все дроби $(\lambda_k-\sigma)/(\lambda_i-\sigma)$ меньше единицы по абсолютной величине, то этот вектор сходится к e_k, а потому вектор $(A-\sigma I)^{-i}x_0$ становится все более параллелен вектору $Se_k = s_k$, т. е. собственному вектору, ассоциированному с λ_k. Как и прежде, числа $\tilde{\lambda}_i = x_i^T A x_i$ сходятся к λ_k.

Преимущество обратной итерации по сравнению со степенным методом состоит в ее способности сходиться к любому собственному значению (тому, что ближе всех к сдвигу σ). Выбирая σ вблизи нужного собственного значения, можно добиться очень быстрой сходимости. Таким образом, в отличие от исходного степенного метода, здесь мы не ограничены возможной близостью других собственных значений. Метод особенно эффективен, когда уже имеется хорошее приближение к собственному значению и требуется лишь найти соответствующий собственный вектор (такова ситуация, например, в разд. 5.3.4). Позднее мы укажем, как выбирать σ, не зная собственных значений: ведь именно эти значения мы хотим вычислить прежде всего!

4.4.3. Ортогональная итерация

Еще одно улучшение метода позволит нам получить одновременную сходимость к p-мерному инвариантному подпространству (где $p > 1$), а не только к отдельному собственному вектору. Оно называется *ортогональной итерацией* (а иногда также *итерированием подпространства* или *одновременной итерацией*).

Алгоритм 4.3. *Ортогональная итерация: пусть Z_0 — матрица размера $n \times p$ с ортонормированными столбцами. Выполняются итерации*

$i = 0$
repeat
  $Y_{i+1} = AZ_i$
  Вычислить разложение $Y_{i+1} = Z_{i+1}R_{i+1}$, *используя алгоритм 3.2 (QR-разложение) (линейная оболочка столбцов матрицы Z_{i+1} дает приближенное*

инвариантное
подпространство для A)

$i = i + 1$
пока метод не сойдется

Приведем неформальный анализ этого метода. Предположим, что $|\lambda_p| > |\lambda_{p+1}|$. Если $p = 1$, то и метод, и его анализ таковы же, как в случае степенного метода. При $p > 1$ имеем $\operatorname{span}\{Z_{i+1}\} = \operatorname{span}\{Y_{i+1}\} = \operatorname{span}\{AZ_i\}$, поэтому $\operatorname{span}\{Z_i\} = \operatorname{span}\{A^i Z_0\} = \operatorname{span}\{S\Lambda^i S^{-1} Z_0\}$. Заметим, что

$$S\Lambda^i S^{-1} Z_0 = S \operatorname{diag}(\lambda_1^i, \ldots, \lambda_n^i) S^{-1} Z_0$$
$$= \lambda_p^i S \begin{bmatrix} (\lambda_1/\lambda_p)^i & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0.$$

Поскольку $\left|\frac{\lambda_j}{\lambda_p}\right| \geq 1$, если $j \leq p$, и $\left|\frac{\lambda_j}{\lambda_p}\right| < 1$ при $j > p$, получаем

$$\begin{bmatrix} (\lambda_1/\lambda_p)^i & & \\ & \ddots & \\ & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0 = \begin{bmatrix} V_i^{p \times p} \\ W_i^{(n-p) \times p} \end{bmatrix},$$

где блок W_i сходится к нулю со скоростью $(\lambda_{p+1}/\lambda_p)^i$, а блок V_i к нулю не стремится. Действительно, если блок V_0 имеет полный ранг (это обобщение предположения из разд. 4.4.1 о том, что $\xi_1 \neq 0$), то и все блоки V_i имеют полный ранг. Запишем матрицу из собственных векторов так: $S = [s_1, \ldots, s_n] \equiv [S_p^{n \times p}, \hat{S}_p^{n \times (n-p)}]$, т. е. $S_p = [s_1, \ldots, s_p]$. Тогда имеем $S\Lambda^i S^{-1} Z_0 = \lambda_p^i S \begin{bmatrix} V_i \\ W_i \end{bmatrix} = \lambda_p^i (S_p V_i + \hat{S}_p W_i)$. Отсюда

$$\operatorname{span}(Z_i) = \operatorname{span}(S\Lambda^i S^{-1} Z_0) = \operatorname{span}(S_p V_i + \hat{S}_p W_i) \Rightarrow \operatorname{span}(S_p V_i),$$

т. е. подпространство $\operatorname{span}(Z_i)$ сходится к $\operatorname{span}(S_p V_i) = \operatorname{span}(S_p)$, а это есть инвариантное подпространство, натянутое на первые p собственных векторов. Мы получили требуемый результат.

QR-разложение используется в методе для того, чтобы векторы, образующие базис подпространства $\operatorname{span}\{A^i Z_0\}$, сохраняли линейную независимость, несмотря на наличие округлений.

Заметим, что если в итерациях этого алгоритма следить лишь за первыми $\hat{p}$ столбцами матриц Z_i, где $\hat{p} < p$, то они будут *теми же самыми*, что и при итерировании только первых $\hat{p}$ столбцов матрицы Z_0, а не p столбцов, как в первом случае. Иными словами, ортогональная итерация по существу происходит одновременно для всех значений $\hat{p} = 1, 2, \ldots, p$. Поэтому, если модули *всех* собственных значений попарно различны, то из проведенного выше анализа сходимости вытекает, что линейная оболочка первых $\hat{p}$ столбцов матрицы Z_i при любом $\hat{p} \leq p$ сходится к $\operatorname{span}\{s_1, \ldots, s_{\hat{p}}\}$.

Итак, исследуя ортогональную итерацию, можно положить $p = n$ и $Z_0 = I$. Приводимая ниже теорема показывает, что при определенных предположениях с помощью ортогональной итерации можно вычислять форму Шура матрицы A.

Теорема 4.8. *Пусть ортогональная итерация с $p = n$ и $Z_0 = I$ применяется к матрице A. Если все собственные значения матрицы имеют попарно различные модули и все главные подматрицы $S(1:j, 1:j)$ невырожденны, то матрицы $A_i \equiv Z_i^T A Z_i$ сходятся к верхней форме Шура матрицы A, т. е. к верхней треугольной матрице, на диагонали которой расположены собственные значения. Они упорядочены по убыванию абсолютных величин.*

Набросок доказательства. Предположение о невырожденности матриц $S(1:j, 1:j)$ для всех j обеспечивает невырожденность блока V_0, требуемую предыдущим анализом. Геометрически это условие означает, что ни один вектор из инвариантного подпространства $\mathrm{span}\{s_1, \ldots, s_j\}$ не ортогонален к подпространству $\mathrm{span}\{e_1, \ldots, e_j\}$, натянутому на первые j столбцов матрицы $Z_0 = I$. Заметим теперь, что Z_i — квадратная ортогональная матрица, поэтому матрицы A и $A_i = Z_i^T A Z_i$ подобны. Положим $Z_i = [Z_{1i} Z_{2i}]$, где Z_{1i} состоит из p столбцов. Тогда

$$A_i = Z_i^T A Z_i = \begin{bmatrix} Z_{1i}^T A Z_{1i} & Z_{1i}^T A Z_{2i} \\ Z_{2i}^T A Z_{1i} & Z_{2i}^T A Z_{2i} \end{bmatrix}.$$

Поскольку подпространство $\mathrm{span}\{Z_{1i}\}$ сходится к инвариантному подпространству матрицы A, $\mathrm{span}\{AZ_{1i}\}$ сходится к тому же самому подпространству, а $Z_{2i}^T A Z_{1i}$ сходится к $Z_{2i}^T Z_{1i} = 0$. Учитывая, что сказанное справедливо для каждого $p < n$, заключаем, что все поддиагональные элементы матрицы A_i стремятся к нулю. Таким образом, A_i сходится к верхней треугольной матрице, т. е. к форме Шура. ◇

В действительности, из этого анализа видно, что подматрица $Z_{2i}^T A Z_{1i} = A_i(p+1:n, 1:p)$ должна сходиться к нулю со скоростью $|\lambda_{p+1}/\lambda_p|^i$. Поэтому λ_p будет предельным значением элемента (p,p) в A_i, а сходимость будет происходить со скоростью $\max(|\lambda_{p+1}/\lambda_p|^i, |\lambda_p/\lambda_{p-1}|^i)$.

Пример 4.5. Характер сходимости ортогональной итерации иллюстрируется численным экспериментом, в котором мы взяли $\Lambda = \mathrm{diag}(1, 2, 6, 30)$ и случайную матрицу S (с числом обусловленности ≈ 20), сформировали матрицу $A = S \cdot \Lambda \cdot S^{-1}$ и провели с ней 19 итераций метода при $p = 4$. Рис. 4.3 и 4.4 показывают, как сходится алгоритм. На рис. 4.3 даны графики ошибок $|A_i(p,p) - \lambda_p|$ в вычисляемых собственных значениях (сплошные линии) и приближений $\max(|\lambda_{p+1}/\lambda_p|^i, |\lambda_p/\lambda_{p-1}|^i)$ (пунктирные линии). В полулогарифмической шкале графики обеих величин по существу представляют собой прямые линии с одинаковым угловым коэффициентом. Это означает, что сами величины суть функции вида $y = c \cdot r^i$, где c и r — некоторые константы, причем r (значение углового коэффициента на графиках) для них одно и то же. Именно таким было предсказание нашего анализа.

Аналогично, на рис. 4.4 приведены графики для $\|A_i(p+1:n, 1:p)\|_2$ (сплошные линии) и приближений $|\lambda_{p+1}/\lambda_p|^i$ (пунктирные линии). Эти гра-

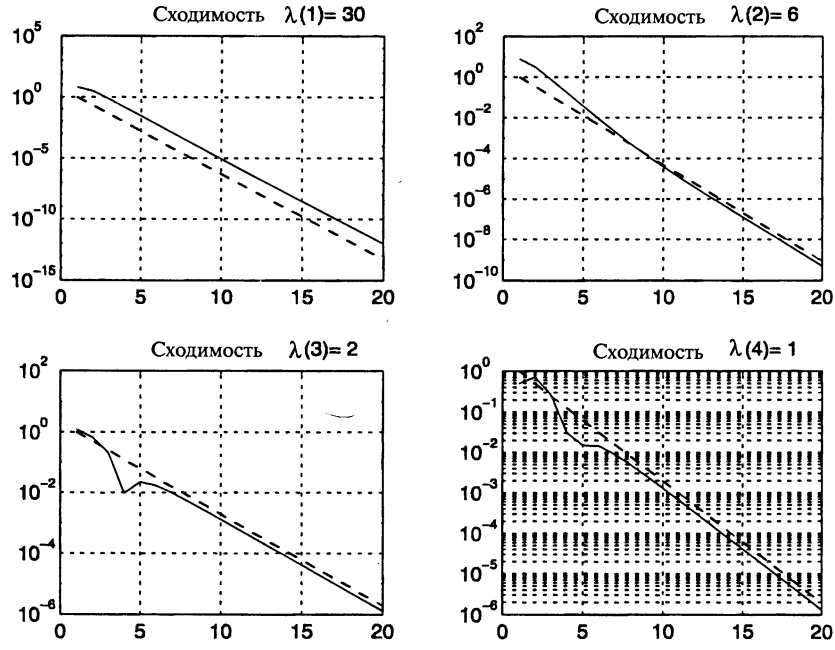


Рис. 4.3. Сходимость диагональных элементов в ортогональной итерации.

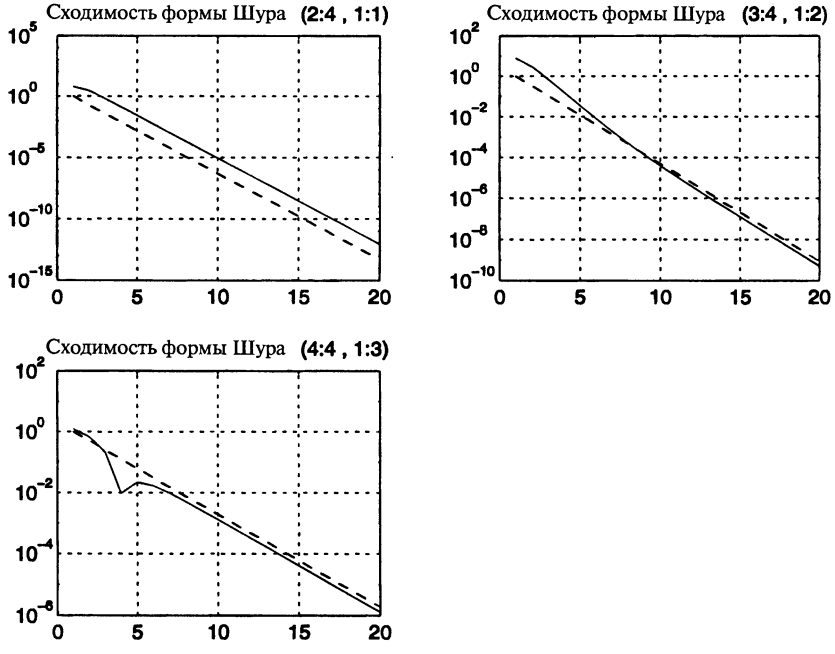


Рис. 4.4. Сходимость к форме Шура в ортогональной итерации.

фики тоже хорошо согласуются. Приведем для сопоставления матрицы A_0 и A_{19}:

$$A = A_0 = \begin{bmatrix} 3.5488 & 15.593 & 8.5775 & -4.0123 \\ 2.3595 & 24.526 & 14.596 & -5.8157 \\ 8.9953 \cdot 10^{-2} & 27.599 & 21.483 & -5.8415 \\ 1.9227 & 55.667 & 39.717 & -10.558 \end{bmatrix},$$

$$A_{19} = \begin{bmatrix} 30.000 & -32.557 & -70.844 & 14.984 \\ 6.7607 \cdot 10^{-13} & 6.0000 & 1.8143 & -.55754 \\ 1.5452 \cdot 10^{-23} & 1.1086 \cdot 10^{-9} & 2.0000 & -.25894 \\ 7.3360 \cdot 10^{-29} & 3.3769 \cdot 10^{-15} & 4.9533 \cdot 10^{-6} & 1.0000 \end{bmatrix}.$$

Этот пример и аналогичные ему могут быть решены с помощью Matlab-программы, помещенной в HOMEPAGE/Matlab/qriter.m. ◇

Пример 4.6. Чтобы понять роль предположения о невырожденности матриц $S(1:j, 1:j)$ в теореме 4.8, рассмотрим диагональную матрицу A, собственные значения которой *не* упорядочены на диагонали по убыванию модулей. Тогда ортогональная итерация дает $Z_i = \mathrm{diag}(\pm 1)$ (диагональная матрица с диагональными элементами ± 1) и $A_i = A$ для всех i, так что собственные значения не переупорядочиваются по убыванию модулей. Покажем теперь, почему необходимо предположение о том, что собственные значения имеют различные модули. Предположим, что A — ортогональная матрица, тогда все собственные значения имеют один и тот же модуль 1. Снова матрицы A_i по существу не меняются в алгоритме. (Строки и столбцы могут умножаться на -1.)

4.4.4. QR-итерация

Наша следующая цель — так модифицировать ортогональную итерацию, чтобы она допускала сдвиги и обращение матрицы, как в разд. 4.4.2. Это повысит эффективность метода и позволит устранить предположение о том, что модули собственных значений различны. Оно было необходимо в теореме 4.8 для обоснования сходимости.

Алгоритм 4.4. *QR-итерация: для заданной матрицы A_0 выполнять итерации*

$i = 0$
repeat
  Вычислить разложение $A_i = Q_i R_i$ *(QR-разложение)*
  $A_{i+1} = R_i Q_i$
  $i = i + 1$
пока метод не сойдется

Поскольку $A_{i+1} = R_i Q_i = Q_i^T (Q_i R_i) Q_i = Q_i^T A_i Q_i$, матрицы A_{i+1} и A_i ортогонально подобны.

Мы утверждаем, что матрица A_i, вычисляемая в QR-итерации, совпадает с матрицей $Z_i^T A Z_i$, неявно вычисляемой в ортогональной итерации.

Лемма 4.3. *Пусть A_i — матрица, вычисляемая в алгоритме 4.4. Тогда $A_i = Z_i^T A Z_i$, где Z_i — матрица, вычисляемая в ортогональной итерации (алгоритм 4.3), исходя из $Z_0 = I$. Таким образом, если все собственные значения имеют различные модули, то матрицы A_i сходятся к форме Шура.*

Доказательство. Применим рассуждение по индукции. Пусть $A_i = Z_i^T A Z_i$. Согласно алгоритму 4.3, $AZ_i = Z_{i+1}R_{i+1}$, где Z_{i+1} и R_{i+1} — соответственно ортогональная и верхнетреугольная матрицы. Но тогда $Z_i^T A Z_i = Z_i^T(Z_{i+1}R_{i+1})$ есть произведение ортогональной матрицы $Q = Z_i^T Z_{i+1}$ и верхнетреугольной матрицы $R = R_{i+1} = Z_{i+1}^T A Z_i$. Это произведение должно быть QR-разложением матрицы $A_i = QR$, поскольку QR-разложение определено единственным образом (с точностью до умножения столбцов в Q и строк в R на -1). Теперь имеем

$$Z_{i+1}^T A Z_{i+1} = (Z_{i+1}^T A Z_i)(Z_i^T Z_{i+1}) = R_{i+1}(Z_i^T Z_{i+1}) = RQ.$$

Именно таким образом QR-итерация отображает A_i в A_{i+1}. Следовательно, $Z_{i+1}^T A Z_{i+1} = A_{i+1}$, что и требовалось. ◇

Многочисленные примеры, иллюстрирующие сходимость QR-итерации, можно активировать с помощью Matlab-программы, о которой говорится в вопросе 4.15.

Из предыдущего анализа мы знаем, что скорость сходимости определяется отношениями собственных значений. Для ускорения сходимости приме́ним сдвиги и обращение.

Алгоритм 4.5. *QR-итерация со сдвигом: для заданной матрицы A_0 выполнять итерации*

$i = 0$
repeat
  Выбрать сдвиг σ_i вблизи некоторого собственного значения матрицы A
  Вычислить разложение $A_i - \sigma_i I = Q_i R_i$ *(QR-разложение)*
  $A_{i+1} = R_i Q_i + \sigma_i I$
  $i = i + 1$
пока метод не сойдется

Лемма 4.4. *Матрицы A_i и A_{i+1} ортогонально подобны.*

Доказательство. $A_{i+1} = R_i Q_i + \sigma_i I = Q_i^T Q_i R_i Q_i + \sigma_i Q_i^T Q_i = Q_i^T(Q_i R_i + \sigma_i I)Q_i = Q_i^T A_i Q_i$. ◇

Если матрица R_i невырожденна, то верна и запись

$$A_{i+1} = R_i Q_i + \sigma_i I = R_i Q_i R_i R_i^{-1} + \sigma_i R_i R_i^{-1} = R_i(Q_i R_i + \sigma_i I)R_i^{-1}$$
$$= R_i A_i R_i^{-1}.$$

Мы утверждаем, что если σ_i — *точное* собственное значение матрицы A_i, то QR-итерация сойдется в один шаг. Действительно, матрица $A_i - \sigma_i I$ вырожденна, раз σ_i является собственным значением, поэтому вырожденна матрица R_i и, значит, какой-то из ее диагональных элементов равен нулю. Предположим, что $R_i(n,n) = 0$. Тогда последняя строка матрицы $R_i Q_i$ будет нулевой, а последняя строка матрицы $A_{i+1} = R_i Q_i + \sigma_i I$ совпадет с $\sigma_i e_n^T$, где e_n — последний столбец единичной матрицы порядка n. Иными словами, последняя строка в A_{i+1} — нулевая с тем исключением, что в позиции (n,n) находится собственное значение σ_i. Это означает, что алгоритм *сошелся*: матрица A_{i+1} имеет верхний

блочно-треугольный вид, нижним диагональным 1×1-блоком которого является число σ_i. Ведущая главная подматрица A' порядка $n-1$ определяет новую, меньшую спектральную задачу, которая может быть решена QR-итерацией без какого-либо изменения собственного значения σ_i: $A_{i+1} = \begin{bmatrix} A' & a \\ 0 & \sigma_i \end{bmatrix}$.

Если σ_i не является точным собственным значением, то элемент $A_{i+1}(n,n)$ считается сошедшимся, если левый нижний блок $A_{i+1}(n, 1:n-1)$ достаточно мал. Из нашего предыдущего анализа следует, что этот блок должен сходиться к нулю со скоростью, определяемой отношением $|\lambda_k - \sigma_i|/\min_{j \neq k}|\lambda_j - \sigma_i|$, где $|\lambda_k - \sigma_i| = \min_j |\lambda_j - \sigma_i|$. Поэтому можно ожидать быстрой сходимости, если σ_i — очень хорошее приближение к собственному значению λ_k.

Имеется и другой способ обоснования ожидаемой быстрой сходимости: нужно распознать в QR-итерации неявно выполняемую обратную итерацию. Если σ_i — точное собственное значение, то последний столбец $\underline{q}_i$ матрицы Q_i является левым собственным вектором матрицы A_i для собственного значения σ_i. В самом деле, $\underline{q}_i^* A_i = \underline{q}_i^*(Q_i R_i + \sigma_i I) = e_n^T R_i + \sigma_i \underline{q}_i^* = \sigma_i \underline{q}_i^*$. Если число σ_i близко́ к некоторому собственному значению, то мы ожидаем, что $\underline{q}_i$ будет близок к соответствующему собственному вектору, поскольку $\underline{q}_i$ параллелен вектору $((A_i - \sigma_i I)^*)^{-1} e_n$ (ниже мы объясним, почему). Иными словами, $\underline{q}_i$ совпадает с вектором, который был бы получен в обратной итерации, примененной к матрице $(A_i - \sigma_i I)^*$ (и вектору e_n). Поэтому мы рассчитываем, что он будет близок к некоторому левому собственному вектору.

Приведем доказательство параллельности $\underline{q}_i$ и вектора $((A_i - \sigma_i I)^*)^{-1} e_n$. Из соотношения $A_i - \sigma_i I = Q_i R_i$ следует, что $(A_i - \sigma_i I) R_i^{-1} = Q_i$. Обратим матрицы в обеих частях этого равенства, а затем возьмем сопряженные к ним. Правая часть Q_i при этом не изменится. В левой же части будет теперь стоять матрица $((A_i - \sigma_i I)^*)^{-1} R_i^*$. Ее последний столбец равен вектору $((A_i - \sigma_i I)^*)^{-1} \cdot [0, \ldots, 0, R_i(n,n)]^T$, лишь множителем отличающемуся от последнего столбца матрицы $((A_i - \sigma_i I)^*)^{-1}$.

Откуда же взять хорошее приближение σ_i к собственному значению, если именно собственные значения мы и желаем вычислить? Мы вернемся к этому вопросу позднее. Пока же отметим, что если процесс уже почти сошелся к *вещественному* собственному значению, то элемент $A_i(n,n)$ должен быть близок к этому собственному значению и число $\sigma_i = A_i(n,n)$ будет хорошим сдвигом. В действительности, такой выбор обеспечивает локальную *квадратичную сходимость*, означающую, что на каждом шаге число верных разрядов приближения *удваивается*. Вот наше объяснение факта квадратичной сходимости. Пусть на шаге i имеем $\|A_i(n, 1:n-1)\|/\|A\| \equiv \eta \ll 1$. Если заменить $A_i(n, 1:n-1)$ нулевым блоком, то A_i станет верхней блочно-треугольной матрицей, а точное собственное значение λ_k возмутится и станет равным элементу $A_i(n,n)$. Если λ_k далеко от других собственных значений, то оно не будет плохо обусловленным, а потому возмущение в нем будет величиной порядка $O(\eta\|A\|)$. Иначе говоря, $|\lambda_k - A_i(n,n)| = O(\eta\|A\|)$. Если положить $\sigma_i = A_i(n,n)$, то можно ожидать, что на следующей итерации $A_{i+1}(n, 1:n-1)$ уменьшится в $|\lambda_k - \sigma_i|/\min_{j \neq k}|\lambda_j - \sigma_i| = O(\eta)$ раз. Как следствие, $\|A_{i+1}(n, 1:n-1)\| = O(\eta^2\|A\|)$, или $\|A_{i+1}(n, 1:n-1)\|/\|A\| = O(\eta^2)$.

Такой характер убывания — от η к $O(\eta^2)$, — и означает квадратичную сходимость.

Пример 4.7. В таблице показаны QR-итерации со сдвигом для той же 4×4-матрицы A_0, что и в примере 4.5. Сдвиги выбираются по правилу $\sigma_i = A_i(4,4)$. Поначалу поведение метода несколько хаотичное, но ближе к концу сходимость становится квадратичной, так что на каждом из последних трех шагов величина $\|A_i(4,1:3)\| \approx |A_i(4,3)|$ как бы возводится в квадрат. Кроме того, на каждом из шагов с четвертого по девятый число верных разрядов в $A_i(4,4)$ удваивается.

$A_0(4,:) =$	$+1.9$	$+56.$	$+40.$	-10.558
$A_1(4,:) =$	$-.85$	-4.9	$+2.2 \cdot 10^{-2}$	-6.6068
$A_2(4,:) =$	$+.35$	$+.86$	$+.30$	0.74894
$A_3(4,:) =$	$-1.2 \cdot 10^{-2}$	$-.17$	$-.70$	1.4672
$A_4(4,:) =$	$-1.5 \cdot 10^{-4}$	$-1.8 \cdot 10^{-2}$	$-.38$	1.4045
$A_5(4,:) =$	$-3.0 \cdot 10^{-6}$	$-2.2 \cdot 10^{-3}$	$-.50$	1.1403
$A_6(4,:) =$	$-1.4 \cdot 10^{-8}$	$-6.3 \cdot 10^{-5}$	$-7.8 \cdot 10^{-2}$	1.0272
$A_7(4,:) =$	$-1.4 \cdot 10^{-11}$	$-3.6 \cdot 10^{-7}$	$-2.3 \cdot 10^{-3}$	0.99941
$A_8(4,:) =$	$+2.8 \cdot 10^{-16}$	$+4.2 \cdot 10^{-11}$	$+1.4 \cdot 10^{-6}$	0.9999996468853453
$A_9(4,:) =$	$-3.4 \cdot 10^{-24}$	$-3.0 \cdot 10^{-18}$	$-4.8 \cdot 10^{-13}$	0.9999999999998767
$A_{10}(4,:) =$	$+1.5 \cdot 10^{-38}$	$+7.4 \cdot 10^{-32}$	$+6.0 \cdot 10^{-26}$	1.000000000000001

К моменту, когда получена матрица A_{10}, не только последняя строка, но и остальная часть матрицы проделала значительный путь к своей предельной форме. Поэтому последующие собственные значения будут вычисляться очень быстро: для каждого будет достаточно одного или двух шагов. Приведем эту матрицу:

$$A_{10} = \begin{bmatrix} 30.000 & -32.557 & -70.844 & 14.985 \\ 6.1548 \cdot 10^{-6} & 6.0000 & 1.8143 & -.55754 \\ 2.5531 \cdot 10^{-13} & 2.0120 \cdot 10^{-6} & 2.0000 & -.25894 \\ 1.4692 \cdot 10^{-38} & 7.4289 \cdot 10^{-32} & 6.0040 \cdot 10^{-26} & 1.0000 \end{bmatrix}. \quad \diamond$$

4.4.5. Как сделать QR-итерацию практичной

Вот вопросы, которые остается решить, чтобы сделать алгоритм более практичным:

1. Метод слишком трудоемок. Стоимость QR-разложения составляет $O(n^3)$ флопов. Поэтому даже в благоприятном случае, когда на вычисление каждого собственного значения требуется в среднем только одна итерация, общая стоимость метода составит $O(n^4)$ флопов. Мы же хотели бы найти алгоритм, стоящий лишь $O(n^3)$ операций.
2. Как следует выбирать σ_i для ускорения сходимости к комплексному собственному значению? Если взять комплексное σ_i, то и вся последующая арифметика должна быть комплексной. Для вещественной матрицы A это означает увеличение общей стоимости процесса приблизительно в 4 раза. Мы же хотели бы найти алгоритм, который в случае вещественной матрицы

A использует только вещественную арифметику и сходится к вещественной форме Шура.

3. Как распознать факт сходимости метода?

Ответы на эти вопросы будут подробно обсуждаться позже. Вкратце же они состоят в следующем:

1. Матрица вначале будет приведена к *верхней форме Хессенберга*, т. е. элементы в A, находящиеся ниже первой поддиагонали, станут нулями ($a_{ij} = 0$, если $i > j+1$) (см. разд. 4.4.6). Далее QR-итерация будет применяться *неявно*, что означает: на каждом шаге матрица Q не будет вычисляться в явном виде, так же как явно не будет производиться умножение на Q (см. разд. 4.4.8). Это снизит стоимость одного шага QR-итерации с $O(n^3)$ до $O(n^2)$, а общую стоимость — с $O(n^4)$ до $O(n^3)$, что и было нашей целью.
   Если A — симметричная матрица, то она будет приведена к трехдиагональной форме, в результате чего стоимость одного шага QR-итерации снизится еще больше, а именно до $O(n)$ операций. Этот случай обсуждается в разд. 4.4.7 и гл. 5.
2. Поскольку комплексные собственные значения вещественных матриц встречаются сопряженными парами, сдвиги на σ_i и $\overline{\sigma}_i$ можно производить совместно. Оказывается, что это позволяет весь процесс вести в вещественной арифметике (см. разд. 4.4.8). Для симметричной матрицы A все собственные значения вещественны и подобной проблемы не возникает.
3. О сходимости судят по «достаточной малости» поддиагональных элементов матрицы A_i. Чтобы найти практичный критерий «малости», воспользуемся понятием обратной устойчивости. Поскольку A_i получается из A ортогональным подобием, в любом случае следует ожидать, что A_i включает в себя ошибки округлений порядка $O(\varepsilon||A||)$. Таким образом, всякий поддиагональный элемент в A_i, по абсолютной величине меньший, чем $O(\varepsilon||A||)$, вполне мог бы быть нулем, поэтому мы и заменяем его на нуль[1]. Если A — верхняя хессенбергова матрица, то замена элемента $a_{p+1,p}$ нулем превращает A в верхнюю блочно-треугольную матрицу $A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$, где A_{11} — размера $p \times p$ и обе подматрицы A_{11} и A_{22} — верхние хессенберговы. После этого собственные значения матрицы A могут быть получены путем независимого вычисления собственных значений для A_{11} и A_{22}. Когда все диагональные блоки станут размера 1×1 или 2×2, алгоритм закончит работу.

4.4.6. Приведение к форме Хессенберга

Для заданной вещественной матрицы A нужно найти такую ортогональную матрицу Q, чтобы QAQ^T была верхней хессенберговой матрицей. Соответ-

[1] На практике используется несколько более жесткий критерий, в котором вместо $||A||$ берется норма соответствующей подматрицы в A. Он учитывает наличие так называемых «градуированных» матриц с большим разбросом в величинах элементов. Поддиагональный элемент может быть заменен нулем и в том случае, когда достаточно мало произведение $a_{p+1,p}a_{p+2,p+1}$ двух последовательных поддиагональных элементов. Детали можно найти в LAPACK-программе slahqr.

ствующий алгоритм является простой вариацией идеи, использованной для QR-разложения.

Пример 4.8. Общую последовательность действий в приведении к форме Хессенберга проиллюстрируем с помощью примера 5-го порядка. В нем Q_i означает отражение порядка 5, выбранное так, чтобы аннулировать в i-м столбце матрицы элементы $i+2,\ldots,n$, не меняя элементов $1,\ldots,i$.

1. Выбрать Q_1 так, чтобы

$$Q_1A = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \end{bmatrix} \quad \text{и} \quad A_1 \equiv Q_1AQ_1^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \end{bmatrix}.$$

   Умножение на Q_1 не изменяет первую строку в A, а умножение на Q_1^T не изменяет первый столбец в Q_1A, сохраняя полученные в нем нули.
2. Выбрать Q_2 так, чтобы

$$Q_2A_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & x & x & x \end{bmatrix} \quad \text{и} \quad A_2 \equiv Q_2A_1Q_2^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & x & x & x \end{bmatrix}.$$

   Умножение на Q_2 изменяет лишь три нижние строки в A_1, а умножение на Q_2^T не изменяет первые два столбца в Q_2A_1, сохраняя имеющиеся в них нули.
3. Выбрать Q_3 так, чтобы

$$Q_3A_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix} \quad \text{и} \quad A_3 = Q_3A_2Q_3^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

   Матрица A_3 — верхняя хессенбергова. В целом, имеем $A_3 = (Q_3Q_2Q_1) \cdot A(Q_3Q_2Q_1)^T \equiv QAQ^T$. ◇

Общий алгоритм для приведения к форме Хессенберга выглядит так:

Алгоритм 4.6. *Приведение к верхней форме Хессенберга:*

если нужна матрица Q, *то положить* $Q = I$
for $i = 1 : n-2$
  $u_i = \text{House}(A(i+1:n, i))$
  $P_i = I - 2u_iu_i^T$ /* $Q_i = \text{diag}(I^{i\times i}, P_i)$ */
  $A(i+1:n, i:n) = P_i \cdot A(i+1:n, i:n)$
  $A(1:n, i+1:n) = A(1:n, i+1:n) \cdot P_i$
  если нужна матрица Q
    $Q(i+1:n, i:n) = P_i \cdot Q(i+1:n, i:n)$ /* $Q = Q_i \cdot Q$ */
  end if
end for

Как и в случае QR-разложения, матрицу P_i не формируют в явном виде: умножение на $I - 2u_i u_i^T$ производится посредством матрично-векторных операций. Векторы u_i могут храниться в поддиагональной части матрицы, снова по аналогии с QR-разложением. Отражения могут применяться с использованием BLAS-операций уровня 3, как это описано в вопросе 3.17. Алгоритм приведения к форме Хессенберга реализован в Matlab'е командой `hess`, а в LAPACK'е — программой `sgehrd`.

Легко подсчитывается, что число флопов в этом алгоритме равно $\frac{10}{3}n^3 + O(n^2)$, а если вычисляется и произведение $Q = Q_{n-1} \dots Q_1$, то $\frac{14}{3}n^3 + O(n^2)$.

Преимущество формы Хессенберга для QR-итерации заключается в том, что один шаг для нее стоит только $6n^2 + O(n)$ флопов вместо $O(n^3)$ в общем случае. Кроме того, эта форма сохраняется алгоритмом, так что матрица остается верхней хессенберговой.

Предложение 4.5. *Форма Хессенберга сохраняется QR-итерацией.*

Доказательство. Легко убедиться в том, что при QR-разложении верхней хессенберговой матрицы (например, матрицы $A_i - \sigma I$) получается верхняя хессенбергова матрица Q (потому что j-й столбец в Q есть линейная комбинация первых j столбцов матрицы $A_i - \sigma I$). После этого легко проверить, что матрица RQ остается верхней хессенберговой. Добавление σI не изменяет формы матрицы. □

Определение 4.5. *Верхняя хессенбергова матрица H называется* неразложимой, *если все элементы на ее первой поддиагонали отличны от нуля.*

Легко видеть, что если $h_{i+1,i} = 0$, т. е. матрица H разложима, то ее собственными значениями являются собственные значения ведущей главной подматрицы порядка i и дополнительной подматрицы порядка $n-i$; обе они хессенберговы. Итак, во всех случаях можно ограничиться рассмотрением неразложимых матриц.

4.4.7. Приведение к трехдиагональной и двухдиагональной формам

Если матрица A симметрична, то каждый полный шаг приведения к форме Хессенберга дает симметричную матрицу, поэтому нули создаются в симметричных позициях. Это означает, что нужно обрабатывать лишь половину матрицы; в результате число операций снижается до $\frac{4}{3}n^3 + O(n^2)$, а если формируется матрица $Q_{n-1} \dots Q_1$, то до $\frac{8}{3}n^3 + O(n^2)$. Такой алгоритм называется *приведением к трехдиагональной форме*. Мы используем его в гл. 5. Алгоритм реализован LAPACK-программой `ssytrd`.

Несколько предвосхищая обсуждение вопроса о вычислении SVD (см. разд. 5.4), напомним (разд. 3.2.3), что собственные значения симметричной матрицы A^TA суть квадраты сингулярных чисел матрицы A. Наш окончательный SVD-алгоритм будет опираться на этот факт, поэтому хотелось бы найти для A форму, которая обеспечивала бы трехдиагональность матрицы A^TA. В качестве такой формы возьмем *верхнюю двухдиагональную* матрицу, т. е. матрицу, ненулевые элементы которой могут располагаться только на главной диагонали и первой наддиагонали. Итак, требуется вычислить ортогональные матрицы Q

и V таким образом, чтобы матрица QAV была верхней двухдиагональной. Соответствующий алгоритм называется *приведением к двухдиагональной форме* и очень похож на приведение к форме Хессенберга и трехдиагональной форме.

Пример 4.9. Проиллюстрируем общую последовательность действий в приведении к двухдиагональной форме с помощью примера 4-го порядка:

1. Выбрать Q_1 так, чтобы

$$Q_1 A = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix} \text{ и } V_1 \text{ так, чтобы } A_1 \equiv Q_1 A V_1 = \begin{bmatrix} x & x & o & o \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix}.$$

   Матрицы Q_1 и V_1 суть отражения, причем V_1 не меняет первый столбец матрицы $Q_1 A$.
2. Выбрать Q_2 так, чтобы

$$Q_2 A_1 = \begin{bmatrix} x & x & o & o \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} \text{ и } V_2 \text{ так, чтобы } A_2 \equiv Q_2 A_1 V_2 = \begin{bmatrix} x & x & o & o \\ o & x & x & o \\ o & o & x & x \\ o & o & x & x \end{bmatrix}.$$

   Здесь Q_2 — отражение, не изменяющее первую строку в A_1, а V_2 — отражение, не меняющее первые два столбца матрицы $Q_2 A_1$.
3. Выбрать Q_3 так, чтобы

$$Q_3 A_2 = \begin{bmatrix} x & x & o & o \\ o & x & x & o \\ o & o & x & x \\ o & o & o & x \end{bmatrix} \quad \text{и положить } V_3 = I, \text{ так что } A_3 = Q_3 A_2.$$

   Отражение Q_3 не меняет первые две строки матрицы A_2. ◇

В общем случае $n \times n$-матрицы A будут получены ортогональные матрицы $Q = Q_{n-1} \dots Q_1$ и $V = V_1 \dots V_{n-2}$ такие, что матрица $QAV = A'$ — верхняя двухдиагональная.

Заметим, что $A'^T A' = V^T A^T Q^T Q A V = V^T A^T A V$, т. е. $A'^T A'$ имеет те же собственные значения, что и $A^T A$, а потому A' имеет те же сингулярные числа, что и A.

Стоимость приведения к двухдиагональной форме составляет $\frac{8}{3}n^3 + O(n^2)$ флопов, а если вычисляются матрицы Q и V, то $4n^3 + O(n^2)$ флопов. Этот алгоритм реализован LAPACK-программой `sgebrd`.

4.4.8. QR-итерация с неявными сдвигами

В этом разделе будет показано, как эффективно реализовать QR-итерацию для верхней хессенберговой матрицы H. Эта реализация будет *неявной* в том смысле, что QR-разложение матрицы H не вычисляется в явном виде. Матрица Q строится неявно как произведение вращений и других простых ортогональных матриц. Излагаемая ниже *неявная Q-теорема* показывает, что эта неявно построенная матрица Q и есть та ортогональная матрица, которая нам нужна. Далее мы указываем, как внедрить в нашу процедуру сдвиг σ,

необходимый для ускорения сходимости. Затем мы обсуждаем прием *двойного* сдвига, используемый для того, чтобы проводить вычисления в вещественной арифметике даже в присутствии комплексных собственных значений. Он состоит в комбинировании двух последовательных QR-итераций с комплексно-сопряженными сдвигами σ и $\overline{\sigma}$; матрица после этого двойного сдвига снова становится вещественной. Наконец, мы обсудим стратегии выбора сдвигов σ и $\overline{\sigma}$, надежно обеспечивающие квадратичную сходимость. Недавно были обнаружены некоторые (редкие) ситуации, когда алгоритм не сходится [25, 65]. Поэтому задача построения полностью надежного и быстрого варианта QR-итерации остается открытой.

Неявная Q-теорема

Наша окончательная реализация QR-итерации будет опираться на следующую теорему.

Теорема 4.9 *(неявная Q-теорема). Пусть $Q^TAQ = H$ — неразложимая верхняя матрица Хессенберга. Тогда первый столбец матрицы Q однозначно (с точностью до знаков) определяет ее 2-й, ... , n-й столбцы.*

Из этой теоремы следует: чтобы в QR-алгоритме вычислить по A_i матрицу $A_{i+1} = Q_i^TA_iQ_i$, достаточно

1. вычислить первый столбец матрицы Q_i (который параллелен первому столбцу матрицы $A_i - \sigma_iI$ и может быть получен из него всего лишь нормировкой) и
2. выбрать остальные столбцы матрицы Q_i так, чтобы Q_i была ортогональна, а A_{i+1} была неразложимой хессенберговой матрицей.

В этом случае неявная Q-теорема гарантирует, что мы правильно вычислим матрицу A_{i+1}, поскольку Q_i определена однозначно с точностью до знаков столбцов, а эти знаки не существенны. (Изменение знаков столбцов в Q_i равносильно замене разложения $A_i - \sigma_iI = Q_iR_i$ на $(Q_iS_i)(S_iR_i)$, где $S_i = \mathrm{diag}(\pm1, \dots, \pm1)$. Но тогда $A_{i+1} = (S_iR_i)(Q_iS_i) + \sigma_iI = S_i(R_iQ_i + \sigma_iI)S_i$. Это ортогональное подобие всего лишь изменяет знаки строк и столбцов в A_{i+1}.)

Доказательство неявной Q-теоремы. Предположим, что $Q^TAQ = H$ и $V^TAV = G$ — неразложимые верхние матрицы Хессенберга, причем матрицы Q и V ортогональны, а их первые столбцы равны. Примем обозначение $(X)_i$ для i-го столбца матрицы X. Мы хотим показать, что $(Q)_i = \pm(V)_i$ для всех $i > 1$ или, иначе говоря, что $W \equiv V^TQ = \mathrm{diag}(\pm1, \dots, \pm1)$.

Поскольку $W = V^TQ$, имеем $GW = GV^TQ = V^TAQ = V^TQH = WH$. Из равенства $GW = WH$ выводим $G(W)_i = (GW)_i = (WH)_i = \sum_{j=1}^{i+1} h_{ji}(W)_j$, откуда $h_{i+1,i}(W)_{i+1} = G(W)_i - \sum_{j=1}^{i} h_{ji}(W)_j$. Учитывая, что $(W)_1 = [1, 0, \dots, 0]^T$, а G имеет верхнюю форму Хессенберга, можем индукцией по i показать, что в столбце $(W)_i$ поддиагональные элементы равны нулю, т. е. матрица W — верхняя треугольная. Так как W в то же время ортогональна, то W — диагональная матрица: $W = \mathrm{diag}(\pm1, \dots, \pm1)$. □

Неявный QR-алгоритм с одинарным сдвигом

Покажем с помощью примера 5-го порядка, как использовать Q-теорему для вычисления матрицы A_1 по $A_0 = A$.

Пример 4.10.

1. Выбрать

$$Q_1^T = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \text{ так, чтобы } A_1 \equiv Q_1^T A Q_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ + & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}.$$

Выбор c_1 и s_1 мы обсудим позже. Пока что эти параметры могут соответствовать произвольному вращению. Символ + в позиции $(3,1)$ называется *выступом* (или горбом). Мы должны избавиться от него, чтобы восстановить форму Хессенберга.

2. Выбрать

$$Q_2^T = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \text{ так, чтобы } Q_2^T A_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

$$A_2 \equiv Q_2^T A_1 Q_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & + & x & x & x \\ o & o & o & x & x \end{bmatrix}.$$

Таким образом, выступ «вытеснен» из позиции $(3,1)$ в позицию $(4,2)$.

3. Выбрать

$$Q_3^T = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix} \text{ так, чтобы } Q_3^T A_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

$$A_3 \equiv Q_3^T A_2 Q_3 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & + & x & x \end{bmatrix}.$$

Выступ вытеснен из позиции $(4,2)$ в позицию $(5,3)$.

4. Выбрать

$$Q_4^T = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix} \quad \text{так, чтобы} \quad Q_4^T A_3 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

$$A_4 \equiv Q_4^T A_3 Q_4 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}.$$

Таким образом, форма Хессенберга восстановлена.

В целом, имеем: для матрицы

$$Q = Q_1 Q_2 Q_3 Q_4 = \begin{bmatrix} c_1 & x & x & x & x \\ s_1 & x & x & x & x \\ & s_2 & x & x & x \\ & & s_3 & x & x \\ & & & s_4 & x \end{bmatrix}$$

матрица $Q^T A Q$ — верхняя хессенбергова. По неявной Q-теореме, первый столбец $[c_1, s_1, 0, \ldots, 0]^T$ матрицы Q однозначно (с точностью до ± 1) определяет остальные столбцы. Выберем первый столбец для Q так, чтобы он был пропорционален первому столбцу матрицы $A - \sigma I$, т. е. вектору $[a_{11} - \sigma, a_{21}, 0, \ldots, 0]^T$. Тогда Q совпадет с ортогональной матрицей в QR-разложении $A - \sigma I$, что и требовалось. ◇

Для $n \times n$-матрицы стоимость одного шага неявной QR-итерации составляет $6n^2 + O(n)$ флопов.

Неявный QR-алгоритм с двойным сдвигом

В этом разделе будет показано, как, выполняя сдвиги σ и $\overline{\sigma}$ совместно, сохранить возможность вычислений в вещественной арифметике. Это существенно для эффективности практической реализации алгоритма, но не для понимания его математики. Поэтому при первом чтении книги данный раздел можно опустить.

Результаты последовательного использования сдвигов σ и $\overline{\sigma}$ можно описать формулами

$$\begin{aligned} A_0 - \sigma I &= Q_1 R_1, \\ A_1 &= R_1 Q_1 + \sigma I, \quad \text{так что} \quad A_1 = Q_1^T A_0 Q_1, \\ A_1 - \overline{\sigma} I &= Q_2 R_2, \\ A_2 &= R_2 Q_2 + \overline{\sigma} I, \quad \text{так что} \quad A_2 = Q_2^T A_1 Q_2 = Q_2^T A_1^T A_0 Q_1 Q_2. \end{aligned}$$

Лемма 4.5. *Матрицы Q_1 и Q_2 можно выбрать так, чтобы:*
(1) *произведение $Q_1 Q_2$ было вещественным и, следовательно,*
(2) *матрица A_2 была вещественной, а*
(3) *первый столбец матрицы $Q_1 Q_2$ легко вычислялся.*

Доказательство. Поскольку $Q_2R_2 = A_1 - \overline{\sigma}I = R_1Q_1 + (\sigma - \overline{\sigma})I$, имеем

$$\begin{aligned} Q_1Q_2R_2R_1 &= Q_1(R_1Q_1 + (\sigma - \overline{\sigma})I)R_1 \\ &= Q_1R_1Q_1R_1 + (\sigma - \overline{\sigma})Q_1R_1 \\ &= (A_0 - \sigma I)(A_0 - \sigma I) + (\sigma - \overline{\sigma})(A_0 - \sigma I) \\ &= A_0^2 - 2(\Re\sigma)A_0 + |\sigma|^2 I \equiv M. \end{aligned}$$

Таким образом, произведение $(Q_1Q_2)(R_2R_1)$ есть QR-разложение вещественной матрицы M. Поэтому матрицы Q_1Q_2 и R_2R_1 могут быть выбраны вещественными. Но тогда и матрица $A_2 = (Q_1Q_2)^T A(Q_1Q_2)$ вещественна.

Первый столбец в Q_1Q_2 пропорционален первому столбцу матрицы $A_0^2 - 2\Re\sigma A_0 + |\sigma|^2 I$, который равен

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2(\Re\sigma)a_{11} + |\sigma|^2 \\ a_{21}(a_{11} + a_{22} - 2(\Re\sigma)) \\ a_{21}a_{32} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

□

Остальные столбцы матрицы Q_1Q_2 вычисляются неявно в соответствии с неявной Q-теоремой. Процесс по-прежнему называется «вытеснением выступа», только выступ теперь размера 2×2, а не 1×1.

Пример 4.11. Проиллюстрируем вытеснение выступа с помощью примера 6-го порядка.

1. Положим $Q_1^T = \begin{bmatrix} \tilde{Q}_1^T & 0 \\ 0 & I \end{bmatrix}$, где первый столбец матрицы $\tilde{Q}_1^T$ определяется приведенной выше формулой. Тогда

$$Q_1^T A = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ + & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \text{ и } A_1 \equiv Q_1^T A Q_1 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ + & x & x & x & x & x \\ + & + & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix}.$$

Заметим, что появился выступ размера 2×2, указываемый плюсами.

2. Выберем отражение Q_2^T, которое, воздействуя только на 2-ю, 3-ю и 4-ю строки матрицы A_1, аннулирует ее элементы $(3,1)$ и $(4,1)$ (это означает, что вне подматрицы, образованной строками и столбцами 2, 3 и 4, Q_2^T совпадает с единичной матрицей):

$$Q_2^T A_1 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & + & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \text{ и } A_2 \equiv Q_2^T A_1 Q_2 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & + & x & x & x & x \\ o & + & + & x & x & x \\ o & o & o & o & x & x \end{bmatrix}.$$

В результате 2×2-выступ «вытеснен» на один столбец вправо.

3. Выберем отражение Q_3^T, которое, воздействуя только на 3-ю, 4-ю и 5-ю строки матрицы A_2, аннулирует ее элементы $(4,2)$ и $(5,2)$ (это означает, что вне подматрицы, образованной строками и столбцами 3, 4 и 5, Q_3^T совпадает с единичной матрицей):

$$Q_3^T A_2 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & + & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \text{ и } A_3 \equiv Q_3^T A_2 Q_3 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & + & x & x & x \\ o & o & + & + & x & x \end{bmatrix}.$$

4. Выберем отражение Q_4^T, которое, воздействуя только на 4-ю, 5-ю и 6-ю строки матрицы A_3, аннулирует ее элементы $(5,3)$ и $(6,3)$ (это означает, что вне подматрицы, образованной строками и столбцами 4, 5 и 6, Q_4^T совпадает с единичной матрицей):

$$A_4 \equiv Q_4^T A_3 A_4 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & + & x & x \end{bmatrix}.$$

5. Выберем матрицу

$$Q_5^T = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & c & s \\ & & & & -s & c \end{bmatrix} \text{ так, чтобы } A_5 = Q_5^T A_4 Q_5 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix}. \quad \diamond$$

Выбор сдвига в QR-алгоритме

Чтобы полностью охарактеризовать шаг хессенберговой QR-итерации с одинарным или двойным сдвигом, мы должны указать способ выбора сдвига σ (и $\overline{\sigma}$). Вспомним (см. конец раздела 4.4.4), что разумным выбором значения одинарного сдвига, асимптотически обеспечивающим квадратичную сходимость к вещественному собственному значению, является правило $\sigma = a_{n,n}$, т. е. значение нижнего углового элемента матрицы A_i. Обобщением этого правила на случай двойного сдвига является *правило Фрэнсиса.* Оно означает, что σ и $\overline{\sigma}$ суть собственные значения нижней угловой 2×2-подматрицы $\begin{bmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{bmatrix}$ матрицы A_i. Это правило позволяет нам получить сходимость как к двум вещественным собственным значениям в нижнем 2×2-углу матрицы, так и к 2×2-блоку с комплексно-сопряженными собственными значениями. Приближение к моменту, когда сходимость наступила, проявляется в том, что элемент $a_{n-1,n-2}$ (а возможно, и $a_{n,n-1}$) становится мал; в этом случае собственные значения

угловой 2×2-подматрицы являются хорошими приближениями к собственным значениям самой матрицы A. Действительно, можно показать, что правило Фрэнсиса асимптотически обеспечивает квадратичную сходимость. Это значит, что если элемент $a_{n-1,n-2}$ (а возможно, и $a_{n,n-1}$) уже достаточно мал, то с каждым следующим шагом его абсолютная величина возводится в квадрат и быстро стремится к нулю. На практике этот прием работает столь хорошо, что для почти всех матриц требуется в среднем два QR-шага для сходимости одного собственного значения. Это оправдывает трактовку QR-итерации как «прямого» метода.

Существуют примеры, для которых QR-итерация со сдвигом Фрэнсиса не сходится в определенном выше смысле (например, матрица

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

остается в этом алгоритме неизменной). Поэтому практические реализации алгоритма, используемые в течение десятилетий, включают в себя так называемый «особый сдвиг»; он применяется в том случае, если предыдущие 10 обычных сдвигов не привели к сходимости. Тем не менее недавно были обнаружены малые множества матриц, для которых не сходятся и эти реализации [25, 65]. Одним из таких множеств является малая окрестность матрицы

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & h & 0 \\ 0 & -h & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Число h получается умножением машинного эпсилона на константу, равную нескольким тысячам. Чтобы устранить подобные случаи несходимости, в современные реализации алгоритма был добавлен еще один «особый сдвиг». Однако построение стратегии сдвигов, которая гарантировала бы быструю сходимость для всех матриц, остается все еще открытой проблемой.

4.5. Другие типы несимметричных спектральных задач

4.5.1. Регулярные матричные пучки и каноническая форма Вейерштрасса

В стандартной проблеме собственных значений разыскиваются числа z, для которых матрица $A - zI$ вырожденна; эти числа называются собственными значениями. Существует несколько важных обобщений этого понятия.

Определение 4.6. *Пусть A и B — матрицы размера $m \times n$. Матрица $A - \lambda B$ называется* матричным пучком, или просто пучком. *Здесь λ — параметр, а не конкретное число.*

Определение 4.7. *Если A и B — квадратные матрицы и $\det(A - \lambda B)$ не равен нулю тождественно, то пучок $A - \lambda B$ называют* регулярным. *В противном случае, пучок называется* сингулярным. *Для регулярного пучка $A - \lambda B$ многочлен $p(\lambda) \equiv \det(A - \lambda B)$ называется* характеристическим многочленом, *а собственные значения пучка $A - \lambda B$ определяются как:*

(1) *корни многочлена* $p(\lambda) = 0$ *и, если* $\deg(p) < n$,
(2) ∞ *(кратности, равной* $n - \deg(p)$*).*

Пример 4.12. Пусть

$$A - \lambda B = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 & & \\ & 0 & \\ & & 1 \end{bmatrix}.$$

Тогда $p(\lambda) = \det(A - \lambda B) = (1 - 2\lambda) \cdot (1 - 0\lambda) \cdot (0 - \lambda) = (2\lambda - 1)\lambda$, поэтому собственными значениями являются $\frac{1}{2}$, 0 и ∞. ◇

Матричные пучки возникают естественным образом во многих математических моделях физических систем; ниже мы приведем соответствующие примеры. В следующем предложении указывается связь между собственными значениями регулярного пучка $A - \lambda B$ и собственными значениями некоторой матрицы.

Предложение 4.6. *Пусть пучок* $A - \lambda B$ *регулярен. Если матрица* B *невырожденна, то все собственные значения пучка конечны и совпадают с собственными значениями матрицы* AB^{-1} *или матрицы* $B^{-1}A$. *Если* B *— вырожденная матрица, то* $A - \lambda B$ *имеет собственное значение* ∞ *кратности* $n - \operatorname{rank}(B)$. *Если матрица* A *невырожденна, то собственные значения пучка* $A - \lambda B$ *суть числа, обратные к собственным значениям матрицы* $A^{-1}B$ *или матрицы* BA^{-1}. *При этом нулевому собственному значению матрицы* $A^{-1}B$ *соответствует бесконечное собственное значение пучка* $A - \lambda B$.

Доказательство. Если λ' — собственное значение пучка и матрица B невырожденна, то $0 = \det(A - \lambda' B) = \det(AB^{-1} - \lambda' I) = \det(B^{-1}A - \lambda' I)$, поэтому λ' является собственным значением и для матриц AB^{-1} и $B^{-1}A$. Для вырожденной B положим $p(\lambda) = \det(A - \lambda B)$ и подставим сюда сингулярное разложение $B = U\Sigma V^T$ матрицы B. Имеем

$$p(\lambda) = \det(A - \lambda U\Sigma V^T) = \det(U(U^T AV - \lambda\Sigma)V^T) = \pm\det(U^T AV - \lambda\Sigma).$$

Поскольку $\operatorname{rank}(B) = \operatorname{rank}(\Sigma)$, то на диагонали матрицы $U^T AV - \lambda\Sigma$ параметр λ появляется только $\operatorname{rank}(B)$ раз. Поэтому степень многочлена $\det(U^T AV - \lambda\Sigma)$ равна $\operatorname{rank}(B)$.

Если A — невырожденная матрица, то равенство $\det(A - \lambda B) = 0$ эквивалентно равенству $\det(I - \lambda A^{-1}B) = 0$ или равенству $\det(I - \lambda BA^{-1}) = 0$. Эти последние равенства могут выполняться лишь, если $\lambda \neq 0$ и $\frac{1}{\lambda}$ есть собственное значение матриц $A^{-1}B$ и BA^{-1}. □

Определение 4.8. *Пусть* λ' *— конечное собственное значение регулярного пучка* $A - \lambda B$. *Ненулевой вектор* x, *такой, что* $(A - \lambda' B)x = 0$ *или, что эквивалентно,* $Ax = \lambda' Bx$, *называется* правым собственным вектором. *Для собственного значения* $\lambda' = \infty$ *правым собственным вектором называется вектор* $x \neq 0$, *такой, что* $Bx = 0$. *Левый собственный вектор пучка* $A - \lambda B$ *определяется как правый собственный вектор пучка* $A^* - \lambda B^*$.

Пример 4.13. Рассмотрим пучок $A - \lambda B$ из примера 4.12. Так как A и B — диагональные матрицы, то правые и левые собственные векторы — это просто столбцы единичной матрицы. ◇

Пример 4.14. Рассмотрим связанную систему материальных точек из примера 4.1. С этой системой естественным образом связаны два матричных пучка. Во-первых, задачу на собственные значения

$$Ax = \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} x = \lambda x$$

можно записать в виде

$$\begin{bmatrix} -B & -K \\ I & 0 \end{bmatrix} x = \lambda \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} x.$$

Эта постановка задачи может быть более удачной, если матрица M очень плохо обусловлена, поскольку в этом случае трудно вычислить хорошие приближения к матрицам $M^{-1}B$ и $M^{-1}K$.

Во-вторых, часто рассматривают случай, когда $B = 0$ (нет затухания). Исходное дифференциальное уравнение в этом случае превращается в $M\ddot{x}(t) + Kx(t) = 0$. Если искать его решения вида $x_i(t) = e^{\lambda_i t}x_i(0)$, то получим $\lambda_i^2 e^{\lambda_i t}Mx_i(0) + e^{\lambda_i t}Kx_i(0) = 0$, или $\lambda_i^2 Mx_i(0) + Kx_i(0) = 0$. Другими словами, $-\lambda_i^2$ есть собственное значение, а $x_i(0)$ — правый собственный вектор пучка $K - \lambda M$. Поскольку, по предположению, матрица M невырожденна, $-\lambda_i^2$ и $x_i(0)$ являются собственным значением и собственным вектором и для матрицы $M^{-1}K$. ◇

Бесконечные собственные значения тоже возникают в практических задачах вполне естественным образом. Например, ниже в этом разделе мы покажем, как бесконечные собственные значения связаны с *импульсным откликом* системы, описываемой обыкновенными дифференциальными уравнениями с линейными ограничениями, иначе говоря, *дифференциально-алгебраическими уравнениями* [41]. См. также вопрос 4.16, где обсуждаются приложения матричных пучков к вычислительной геометрии и машинной графике.

Вспомним, что и теория, и алгоритмы для матричной проблемы собственных значений предусматривали приведение исходной матрицы A с помощью подобия $S^{-1}AS$ к некоторой форме, более «простой», чем A. Приводимое ниже определение указывает, как обобщить понятие подобия на случай матричных пучков. После этого будут обобщены для пучков и формы Жордана и Шура.

Определение 4.9. *Пусть P_L и P_R — произвольные невырожденные матрицы. Говорят, что пучки $A - \lambda B$ и $P_LAP_R - \lambda P_LBP_R$* эквивалентны.

Предложение 4.7. *Эквивалентные регулярные пучки $A - \lambda B$ и $P_LAP_R - \lambda P_LBP_R$ имеют одни и те же собственные значения. Вектор x тогда и только тогда является правым собственным вектором пучка $A - \lambda B$, когда $P_R^{-1}x$ является правым собственным вектором пучка $P_LAP_R - \lambda P_LBP_R$. Вектор y тогда и только тогда является левым собственным вектором пучка $A - \lambda B$, когда $(P_L^*)^{-1}y$ является левым собственным вектором пучка $P_LAP_R - \lambda P_LBP_R$.*

Доказательство.

Равенство $\det(A - \lambda B) = 0$ эквивалентно равенству $\det(P_L(A - \lambda B)P_R) = 0$.

Равенство $(A - \lambda B)x = 0$ эквивалентно равенству $P_L(A - \lambda B)P_RP_R^{-1}x = 0$.

Равенство $(A-\lambda B)^*y=0$ эквивалентно равенству $P_R^*(A-\lambda B)^*P_L^*(P_L^*)^{-1}y=0$.□

В следующей теореме каноническая форма Жордана обобщается на случай регулярных матричных пучков.

Теорема 4.10 *(каноническая форма Вейерштрасса). Для регулярного пучка $A-\lambda B$ найдутся невырожденные матрицы P_L и P_R, такие, что*

$$P_L(A-\lambda B)P_R = \operatorname{diag}(J_{n_1}(\lambda_1)-\lambda I_{n_1},\dots,J_{n_k}(\lambda_{n_k})-\lambda I_{n_k},N_{m_1},\dots,N_{m_\tau}),$$

где $J_{n_i}(\lambda_i)$ — жорданов блок порядка n_i с собственным значением λ_i:

$$J_{n_i}(\lambda_i)=\begin{bmatrix}\lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1\\ & & & \lambda_i\end{bmatrix},$$

а N_{m_i} — «жорданов блок для собственного значения ∞ кратности m_i»:

$$N_{m_i}=\begin{bmatrix}1 & \lambda & & \\ & 1 & \ddots & \\ & & \ddots & \lambda\\ & & & 1\end{bmatrix}=I_{m_i}-\lambda J_{m_i}(0).$$

Доказательство теоремы можно найти в [110].

Приложения форм Жордана и Вейерштрасса к дифференциальным уравнениям

Рассмотрим задачу Коши для линейного дифференциального уравнения $\dot{x}(t)=Ax(t)+f(t)$, $x(0)=x_0$. Ее решение можно выписать в явном виде: $x(t)=e^{At}x_0+\int_0^t e^{A(t-\tau)}f(\tau)d\tau$. Если мы умеем приводить A к форме Жордана: $A=SJS^{-1}$, то, производя в дифференциальном уравнении замену переменного $y(t)=S^{-1}x(t)$, получим $\dot{y}(t)=Jy(t)+S^{-1}f(t)$, $y(0)=y_0=S^{-1}x_0$. Решение этой задачи имеет вид $y(t)=e^{Jt}y_0+\int_0^t e^{J(t-\tau)}S^{-1}f(\tau)d\tau$. Имеется явная формула, позволяющая найти e^{Jt} или любую другую функцию от жордановой матрицы J. (Эту формулу не следует использовать в машинных вычислениях! В вопросе 4.4 обсуждается более надежный подход к вычислению функций от матрицы.) Предположим, что функция f задана своим рядом Тейлора $f(z)=\sum_{i=0}^{\infty}\frac{f^{(i)}(0)z^i}{i!}$, а J — жорданов блок: $J=\lambda I+N$; в матрице N первая наддиагональ состоит из единиц, а прочие элементы равны нулю. Имеем

$$\begin{aligned}f(J)&=\sum_{i=0}^{\infty}\frac{f^{(i)}(0)(\lambda I+N)^i}{i!}\\ &=\sum_{i=0}^{\infty}\frac{f^{(i)}(0)}{i!}\sum_{j=0}^{i}\binom{i}{j}\lambda^{i-j}N^j \quad \text{по формуле бинома Ньютона}\end{aligned}$$

$$= \sum_{j=0}^{\infty}\sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!}\binom{i}{j}\lambda^{i-j}N^j \quad \text{изменяя порядок суммирования}$$

$$= \sum_{j=0}^{n-1} N^j \sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!}\binom{i}{j}\lambda^{i-j}.$$

В последнем равенстве использовано то обстоятельство, что $N^j{=}0$ для $j{>}n-1$. Заметим, что при $j < n$ матрица N^j содержит единицы на j-й наддиагонали и нули во всех прочих позициях. Наконец, заметим, что $\sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!}\binom{i}{j}\lambda^{i-j}$ есть разложение Тейлора для $f^{(j)}(\lambda)/j!$. Таким образом,

$$f(J) = f\left(\begin{bmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix}^{n\times n}\right) = \sum_{j=0}^{n-1} \frac{N^j f^{(j)}(\lambda)}{j!}$$

$$= \begin{bmatrix} f(\lambda) & f'(\lambda) & \frac{f''(\lambda)}{2!} & \vdots & \frac{f^{(n-1)}(\lambda)}{(n-1)!} \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \frac{f''(\lambda)}{2!} \\ & & & \ddots & f'(\lambda) \\ & & & & f(\lambda) \end{bmatrix}. \tag{4.6}$$

Итак, $f(J)$ — верхняя треугольная матрица, на j-й наддиагонали которой стоит число $f^{(j)}(\lambda)/j!$.

Для решения более общей задачи $B\dot{x} = Ax + f(t)$, где пучок $A - \lambda B$ регулярен, используем форму Вейерштрасса. Предположим, что пучок $P_L(A - \lambda B)P_R$ находится в форме Вейерштрасса, и перепишем уравнение в виде $P_L B P_R P_R^{-1}\dot{x} = P_L A P_R P_R^{-1} x + P_L f(t)$. Положим $P_R^{-1}x = y$ и $P_L f(t) = g(t)$. Теперь задача разлагается в сумму подзадач:

$$\begin{bmatrix} I_{n_1} & & & & & \\ & \ddots & & & & \\ & & I_{n_k} & & & \\ & & & J_{m_1}(0) & & \\ & & & & \ddots & \\ & & & & & J_{m_r}(0) \end{bmatrix}\dot{y}$$

$$= \begin{bmatrix} J_{n_1}(\lambda_1) & & & & & \\ & \ddots & & & & \\ & & J_{n_k}(\lambda_k) & & & \\ & & & I_{m_1} & & \\ & & & & \ddots & \\ & & & & & I_{m_r} \end{bmatrix} y + g.$$

Каждая подзадача $\dot{\tilde{y}} = J_{n_i}(\lambda_i)\tilde{y} + \tilde{g}(t) \equiv J\tilde{y} + \tilde{y}(t)$ есть стандартное линейное дифференциальное уравнение, решение которого имеет вид

$$\tilde{y}(t) = \tilde{y}(0)e^{Jt} + \int_0^t e^{J(t-\tau)}\tilde{g}(\tau)d\tau.$$

Решение системы $J_m(0)\dot{\tilde{y}} = \tilde{y} + \tilde{g}(t)$ достигается обратной подстановкой, исходя из последнего уравнения. Запишем систему более подробно:

$$\begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix} \begin{bmatrix} \tilde{\dot{y}}_1 \\ \vdots \\ \vdots \\ \tilde{\dot{y}}_m \end{bmatrix} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \vdots \\ \tilde{y}_m \end{bmatrix} + \begin{bmatrix} \tilde{g}_1 \\ \vdots \\ \vdots \\ \tilde{g}_m \end{bmatrix}.$$

Из m-го (т. е. последнего) уравнения имеем $0 = \tilde{y}_m + \tilde{g}_m$, или $\tilde{y}_m = -\tilde{g}_m$. Уравнение с номером i есть $\tilde{\dot{y}}_{i+1} = \tilde{y}_i + \tilde{g}_i$, откуда $\tilde{y}_i = \tilde{\dot{y}}_{i+1} - \tilde{g}_i$ и, как следствие,

$$\tilde{y}_i = \sum_{k=i}^{m} -\frac{d^{k-i}}{dt^{k-i}}\tilde{g}_k(t).$$

Следовательно, решение зависит от производных функции $\tilde{g}$, а не выражается интегралом от $\tilde{g}$, как для обычного дифференциального уравнения. Поэтому для непрерывной, но не дифференцируемой функции $\tilde{g}$ мы можем получить разрывное решение. Такую ситуацию иногда характеризуют термином *импульсный отклик*. Она может иметь место лишь, если у задачи есть бесконечные собственные значения. Чтобы задача допускала непрерывное решение $\tilde{y}$, при $t = 0$ должны выполняться некоторые условия согласования:

$$y_i(0) = \sum_{k=m}^{i} -\frac{d^{k-i}}{dt^{k-i}}g_k(0).$$

Численные методы решения таких *дифференциально-алгебраических уравнений* (или обыкновенных дифференциальных уравнений с алгебраическими ограничениями), основанные на пошаговом интегрировании, описаны в [41].

Обобщенная форма Шура для регулярных пучков

Точно так же, как мы не можем устойчиво вычислять жорданову форму, мы не можем устойчиво вычислить и ее обобщение — форму Вейерштрасса. Поэтому вместо нее будем вычислять обобщенную форму Шура.

Теорема 4.11 (обобщенная форма Шура). *Для регулярного пучка $A - \lambda B$ найдутся унитарные матрицы Q_L и Q_R, такие, что обе матрицы $Q_L A Q_R = T_A$ и $Q_L B Q_R = T_B$ — верхние треугольные. Собственными значениями пучка $A - \lambda B$ являются числа $T_{A_{ii}}/T_{B_{ii}}$, т. е. отношения диагональных элементов матриц T_A и T_B.*

Доказательство очень похоже на вывод обычной формы Шура. Пусть λ' — собственное значение пучка, а x — соответствующий собственный вектор единичной длины: $\|x\|_2 = 1$. Поскольку $Ax - \lambda' Bx = 0$, векторы Ax и Bx являются кратными одного и того же нормированного вектора y (это верно и в том

случае, если какой-то из векторов Ax и Bx равен нулю. Пусть $X = [x, \tilde{X}]$ и $Y = [y, \tilde{Y}]$ — унитарные матрицы, имеющие первыми столбцами соответственно x и y. Тогда $Y^*AX = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix}$ и $Y^*BX = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} \\ 0 & \tilde{B}_{22} \end{bmatrix}$ по построению. Далее процесс применяется по индукции к пучку $\tilde{A}_{22} - \lambda\tilde{B}_{22}$. □

Для вещественных матриц A и B существует и обобщенная вещественная форма Шура, т. е. найдутся вещественные ортогональные матрицы Q_L и Q_R, такие, что матрица $Q_L A Q_R$ — верхняя квазитреугольная, а матрица $Q_L B Q_R$ — верхняя треугольная.

QR-алгоритм со всеми его усовершенствованиями может быть обобщен на задачу вычисления обобщенной (вещественной) формы Шура. Это обобщение называется QZ-алгоритмом и реализовано в LAPACK'е подпрограммой `sgges`. В Matlab'е QZ-алгоритм активируется командой `eig(A,B)`.

Определенные пучки

Более простой специальный случай, часто возникающий на практике, — это пучок $A - \lambda B$, где $A = A^T$, $B = B^T$ и матрица B положительно определена. Такие пучки называются *определенными.*

Теорема 4.12. *Пусть $A = A^T$, $B = B^T$ и матрица B положительно определена. Тогда найдется вещественная невырожденная матрица X, такая, что $X^TAX = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ и $X^TBX = \mathrm{diag}(\beta_1, \ldots, \beta_n)$. В частности, все собственные значения α_i/β_i пучка $A - \lambda B$ вещественны и конечны.*

Доказательство. Наше доказательство представляет собой алгоритм, фактически используемый для решения задачи:

(1) Пусть $LL^T = B$ есть разложение Холесского матрицы B.

(2) Положим $H = L^{-1}AL^{-T}$; заметим, что H — симметричная матрица.

(3) Положим $H = Q\Lambda Q^T$, где Q ортогональная, а Λ — вещественная диагональная матрица.

Тогда для матрицы $X = L^{-T}Q$ имеем $X^TAX = Q^TL^{-1}AL^{-T}Q = \Lambda$ и $X^TBX = Q^TL^{-1}BL^{-T}Q = I$. □

Отметим, что утверждение теоремы сохраняет силу и в том случае, когда положительно определена не B, а матрица $\alpha A + \beta B$ для некоторых скаляров α и β.

Описанный алгоритм реализован LAPACK-программой `ssygv`.

Пример 4.15. Рассмотрим пучок $K - \lambda M$ из примера 4.14. Он определен, поскольку матрица жесткости K симметрична, а матрица масс M симметрична и положительно определена. В действительности, для этого очень простого примера матрица K трехдиагональная, а матрица M диагональная; диагональным будет и ее множитель Холесского L, а матрица $H = L^{-1}KL^{-T}$ снова симметричная и трехдиагональная. В гл. 5 мы рассмотрим ряд алгоритмов для симметричных трехдиагональных задач на собственные значения. ◇

4.5.2. Сингулярные матричные пучки и каноническая форма Кронекера

Обратимся теперь к сингулярным пучкам. Напомним, что пучок $A-\lambda B$ сингулярен, если матрицы A и B не являются квадратными, либо они квадратные, но $\det(A-\lambda B)=0$ для всех значений λ. Как показывает следующий пример, нужна осмотрительность при распространении понятия собственного значения на этот случай.

Пример 4.16. Пусть $A=\begin{bmatrix}1&0\\0&0\end{bmatrix}$ и $B=\begin{bmatrix}1&0\\0&0\end{bmatrix}$. Производя как угодно малые возмущения, можем получить матрицы $A'=\begin{bmatrix}1&\varepsilon_1\\\varepsilon_2&0\end{bmatrix}$ и $B'=\begin{bmatrix}1&\varepsilon_3\\\varepsilon_4&0\end{bmatrix}$. Собственные значения возмущенного пучка — это отношения $\varepsilon_1/\varepsilon_3$ и $\varepsilon_2/\varepsilon_4$, которые могут быть произвольными комплексными числами. Таким образом, чувствительность собственных значений исходного пучка *бесконечна.* ◇

Как мы увидим ниже, несмотря на эту крайнюю чувствительность, сингулярные пучки используются при моделировании некоторых физических систем.

Покажем теперь, как обобщить формы Жордана и Вейерштрасса на случай сингулярного пучка. Помимо жордановых и «бесконечных жордановых» блоков, мы введем в каноническую форму два новых «сингулярных блока».

Теорема 4.13 (каноническая форма Кронекера). *Пусть A и B — произвольные прямоугольные матрицы размера $m\times n$. Тогда найдутся квадратные невырожденные матрицы P_L и P_R, такие, что матрица $P_LAP_R-\lambda P_LBP_R$ блочно-диагональная и на ее диагонали присутствуют четыре типа блоков, а именно:*

$$J_m(\lambda')-\lambda I=\begin{bmatrix}\lambda'-\lambda&1&&\\&\ddots&\ddots&\\&&\ddots&1\\&&&\lambda'-\lambda\end{bmatrix},\quad \textit{жорданов блок порядка } m;$$

$$N_m=\begin{bmatrix}1&\lambda&&\\&\ddots&\ddots&\\&&\ddots&\lambda\\&&&1\end{bmatrix},\quad \textit{жорданов блок порядка } m \textit{ для } \lambda=\infty;$$

$$L_m=\begin{bmatrix}1&\lambda&&\\&\ddots&\ddots&\\&&1&\lambda\end{bmatrix},\quad \textit{правый сингулярный блок размера } m\times(m+1);$$

$$L_m^T=\begin{bmatrix}1&&\\\lambda&\ddots&\\&\ddots&1\\&&\lambda\end{bmatrix},\quad \textit{левый сингулярный блок размера } (m+1)\times m.$$

Матрица L_m называется правым сингулярным блоком, потому что ее (правое) ядро содержит при всех λ вектор $[\lambda^m, -\lambda^{m-1}, \dots, \pm 1]$. Аналогичный вектор из (левого) ядра имеет матрица L_m^T.

Доказательство теоремы можно найти в [110].

Подобно тому как форма Шура была обобщена в предыдущем разделе на случай регулярных матричных пучков, она может быть распространена и на сингулярные пучки. Обсуждение канонических форм, теории возмущений и существующих программ дано в [27, 79, 246].

Сингулярные пучки используются как модели в теории систем и теории управления. Мы приведем два примера таких приложений.

Приложения формы Кронекера к дифференциальным уравнениям

Пусть нужно решить уравнение $B\dot{x} = Ax + f(t)$, которому соответствует сингулярный пучок $A - \lambda B$. Записывая задачу в виде $P_L B P_R P_R^{-1}\dot{x} = P_L A P_R P_R^{-1} x + P_L f(t)$, мы разложим ее в сумму независимых подзадач, соответствующих диагональным блокам. Имеется четыре типа подзадач, по одному для каждого типа блоков в форме Кронекера. При исследовании регулярных пучков и формы Вейерштрасса уже было показано, как обрабатываются блоки $J_m(\lambda') - \lambda I$ и N_m, поэтому достаточно рассмотреть блоки L_m и L_m^T. Для блока L_m имеем

$$\begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_{m+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{m+1} \end{bmatrix} + \begin{bmatrix} g_1 \\ \vdots \\ g_m \end{bmatrix},$$

т. е.

$$\begin{aligned} \dot{y}_2 &= y_1 + g_1, && \text{или} && y_2(t) = y_2(0) + \textstyle\int_0^t (y_1(\tau) + g_1(\tau))\,d\tau, \\ \dot{y}_3 &= y_2 + g_2, && \text{или} && y_3(t) = y_3(0) + \textstyle\int_0^t (y_2(\tau) + g_2(\tau))\,d\tau, \\ &\vdots \\ \dot{y}_{m+1} &= y_m + g_m, && \text{или} && y_{m+1}(t) = y_{m+1}(0) + \textstyle\int_0^t (y_m(\tau) + g_m(\tau))\,d\tau. \end{aligned}$$

Это означает, что можно взять в качестве y_1 произвольную интегрируемую функцию и получить из нее решение задачи посредством указанных рекурсий. Дело обстоит таким образом потому, что в системе число неизвестных на единицу больше, чем число уравнений, т. е. система *недоопределена*. Для блока L_m^T имеем

$$\begin{bmatrix} 0 & & \\ 1 & \ddots & \\ & \ddots & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & \ddots & \\ & \ddots & 1 \\ & & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} + \begin{bmatrix} g_1 \\ \vdots \\ g_{m+1} \end{bmatrix}$$

или

$$\begin{aligned} 0 &= y_1 + g_1, \\ \dot{y}_1 &= y_2 + g_2, \\ &\vdots \\ \dot{y}_{m-1} &= y_m + g_m, \\ \dot{y}_m &= g_{m+1}, \end{aligned}$$

Начиная с первого уравнения, последовательно получаем

$$\begin{aligned} y_1 &= -g_1, \\ y_2 &= -g_2 - \dot{g}_1, \\ &\vdots \\ y_m &= -g_m - \dot{g}_{m-1} - \ldots - \tfrac{d^{m-1}}{dt^{m-1}} g_1, \end{aligned}$$

а также *условие согласования* $g_{m+1} = -\dot{g}_m - \cdots - \frac{d^m}{dt^m} g_1$. Если функции g_i не удовлетворяют этому условию, то система не имеет решений. Здесь число уравнений на единицу больше, чем число неизвестных, т. е. данная подзадача *переопределена.*

Приложения формы Кронекера к теории систем и теории управления

Управляемым подпространством уравнения $\dot{x}(t) = Ax(t) + Bu(t)$ называется множество *состояний*, которые могут быть достигнуты решением $x(t)$, исходящим из положения $x(0) = 0$, под воздействием *входного управления* $u(t)$. Это уравнение используется для моделирования систем управления (с обратной связью), где $u(t)$ выбирается инженером так, чтобы сообщить решению $x(t)$ требуемые свойства, например ограниченность. С помощью формулы

$$\begin{aligned} x(t) &= \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau = \int_0^t \sum_{i=0}^{\infty} \frac{(t-\tau)^i}{i!} A^i Bu(\tau) d\tau \\ &= \sum_{i=0}^{\infty} A^i B \int_0^t \frac{(t-\tau)^i}{i!} u(\tau) d\tau \end{aligned}$$

можно показать, что управляемое подпространство имеет вид $\operatorname{span}\{[B, AB, A^2B, \ldots, A^{n-1}B]\}$. Компоненты решения $x(t)$, не принадлежащие этому подпространству, не могут управляться изменением $u(t)$. Чтобы определить, может ли моделируемая физическая система управляться входом $u(t)$, приходится вычислять управляемое подпространство на практике. Для этого к сингулярному пучку $[B, A - \lambda I]$ применяют метод типа QR-алгоритма. Относительно деталей см. [78, 246, 247].

4.5.3. Нелинейные задачи на собственные значения

В заключение, рассмотрим *нелинейную проблему собственных значений*, или *матричный многочлен*

$$\sum_{i=0}^{d} \lambda^i A_i = \lambda^d A_d + \lambda^{d-1} A_{d-1} + \cdots + \lambda A_1 + A_0. \tag{4.7}$$

Для большей простоты предположим, что все A_i суть $n \times n$-матрицы, причем A_d невырожденна.

Определение 4.10. *Многочлен* $p(\lambda) = \det\left(\sum_{i=0}^{d} \lambda^i A_i\right)$ *называется* характеристическим многочленом *матричного многочлена (4.7). Собственные значения определяются как корни уравнения* $p(\lambda) = 0$. *Можно проверить, что степень* $p(\lambda)$ *равна* $d \cdot n$, *поэтому имеется* $d \cdot n$ *собственных значений. Пусть* γ *— собственное значение. Ненулевой вектор* x, *такой, что* $\sum_{i=0}^{d} \gamma^i A_i x = 0$,

называется правым собственным вектором *для собственного значения* γ. Левый собственный вектор y *определяется аналогично посредством условия* $\sum_{i=0}^{d} \gamma^i y^* A_i = 0$.

Пример 4.17. Снова обратимся к примеру 4.1. Там мы имели дифференциальное уравнение $M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = 0$ (см. (4.3)). Если искать решения вида $x(t) = e^{\lambda_i t} x_i(0)$, то получим $e^{\lambda_i t}(\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i \lambda(0)) = 0$, или $\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i(0) = 0$. Таким образом, λ_i и $x_i(0)$ суть собственное значение и собственный вектор матричного многочлена $\lambda^2 M + \lambda B + K$. ◇

Поскольку матрица A_d, по предположению, невырожденна, можно умножить исходную задачу на A_d^{-1} и получить эквивалентную задачу с многочленом $\lambda^d I + A_d^{-1} A_{d-1} \lambda^{d-1} + \cdots + A_d^{-1} A_0$. Поэтому, для упрощения обозначений, можем в дальнейшем считать, что $A_d = I$ (по поводу общего случая см. разд. 4.6). В наиболее простом случае, когда все A_i суть матрицы размера 1×1, т. е. скаляры, исходный матричный многочлен совпадает с характеристическим многочленом.

Мы можем свести задачу вычисления собственных значений матричного многочлена к стандартной проблеме собственных значений, пользуясь приемом, аналогичным известному приему преобразования обыкновенного дифференциального уравнения высокого порядка в систему уравнений первого порядка. Начнем с простейшего случая $n = 1$, где все матрицы A_i являются скалярами. Пусть γ — корень многочлена. Тогда вектор $x' = [\gamma^{d-1}, \gamma^{d-2}, \ldots, \gamma, 1]^T$ удовлетворяет соотношению

$$Cx' \equiv \begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & -A_0 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{bmatrix} x' = \begin{bmatrix} -\sum_{i=0}^{d-1} \gamma^i A_i \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} = \begin{bmatrix} \gamma^d \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} = \gamma x'.$$

Таким образом, x' и γ являются собственным вектором и собственным значением матрицы C, которая называется *сопровождающей матрицей* многочлена (4.7).

(В Matlab-программе `roots`, вычисляющей корни многочлена, хессенбергова QR-итерация из разд. 4.4.8 применяется к сопровождающей матрице C. Этот метод вычисления корней, хоть он и дорог, считается в настоящее время одним из самых надежных [100, 117, 241]. Сейчас разрабатываются более дешевые альтернативные методы.)

Та же идея работает и в случае, когда A_i являются матрицами. Матрица C имеет порядок $n \cdot d$ и называется *блочной сопровождающей матрицей* многочлена (4.7). В (блочных) строках $2, \ldots, d$ вместо единиц и нулей стоят теперь соответственно единичная и нулевая матрицы порядка n. Вектор x' принимает вид

$$x' = \begin{bmatrix} \gamma^{d-1} x \\ \gamma^{d-2} x \\ \vdots \\ \gamma x \\ x \end{bmatrix},$$

где x — правый собственный вектор матричного многочлена. По-прежнему выполняется соотношение $Cx' = \gamma x'$.

Пример 4.18. Снова возвращаясь к многочлену $\lambda^2 M + \lambda B + K$, вначале заменим его на многочлен $\lambda^2 I + \lambda M^{-1}B + M^{-1}K$, а затем перейдем к сопровождающей матрице

$$C = \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix}.$$

Она совпадает с матрицей A в уравнении (4.4) примера 4.1. ◇

Отметим, в заключение, что в вопросе 4.16 обсуждается использование матричных многочленов для решения одной задачи из *вычислительной геометрии.*

4.6. Резюме

В приводимом ниже списке суммированы содержащиеся в данной главе сведения о канонических формах, алгоритмах, их стоимостях и приложениях к обыкновенным дифференциальным уравнениям (ОДУ). Даны также некоторые сведения об алгоритмах, использующих симметрию матрицы, хотя более подробно эти алгоритмы рассматриваются в следующей главе. Алгоритмы для разреженных матриц обсуждаются в гл. 7.

- $A - \lambda I$
  - Жорданова форма: для некоторой невырожденной матрицы S имеем

$$A - \lambda I = S \cdot \operatorname{diag}\left(\dots, \begin{bmatrix} \lambda_i - \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i - \lambda \end{bmatrix}^{n_i \times n_i}, \dots \right) \cdot S^{-1}.$$

  - Форма Шура: для некоторой унитарной матрицы Q имеем $A - \lambda I = Q(T - \lambda I)Q^*$, где T — треугольная матрица.
  - Вещественная форма Шура вещественной матрицы A: для некоторой вещественной ортогональной матрицы Q имеем $A - \lambda I = Q(T - \lambda I)Q^T$, где T — вещественная квазитреугольная матрица.
  - Приложение к ОДУ: позволяет решать задачу $\dot{x}(t) = Ax(t) + f(t)$.
  - Алгоритм: выполнить приведение к форме Хессенберга (алгоритм 4.6), а затем с помощью QR-итерации вычислить форму Шура (алгоритм 4.5, реализация которого описана в разд. 4.4.8). Собственные векторы могут быть найдены из формы Шура (как показано в разд. 4.2.1).
  - Стоимость: составляет $10n^3$ флопов, если нужны только собственные значения, $25n^3$ флопов, если требуются также матрицы T и Q, и несколько больше $27n^3$ флопов, если нужно вычислить и собственные векторы. Не все составляющие алгоритма могут использовать BLAS-операции уровня 3, поэтому сравнение с матричным умножением, в действительности, дает менее благоприятные результаты, чем простое сопоставление указанных выше стоимостей с $2n^3$ флопов, требуемых для перемно-

жения $n \times n$-матриц: на IBM RS6000/590 вычисление собственных значений требует времени, большего не в $(10n^3)/(2n^3) = 5$ раз, а в 23 раза при $n = 100$ и 19 раз при $n = 1000$ [10, стр. 62]. При вычислении на той же машине и собственных значений, и собственных векторов происходит замедление не в $(27n^3)/(2n^3) = 13.5$ раз, а в 41 раз при $n = 100$ и 60 раз при $n = 1000$. Таким образом, вычисление собственных значений несимметричных матриц является дорогостоящей процедурой. (Аналогичная процедура для симметричного случая *много* дешевле; см. гл. 5.)
- LAPACK: программы `sgees` для вычисления формы Шура и `sgeev` для вычисления собственных значений и собственных векторов; аналогичные программы `sgeesx` и `sgeevx`, дополнительно вычисляющие границы для погрешностей.
- Matlab: команды `schur` для вычисления формы Шура и `eig` для вычисления собственных значений и собственных векторов.
- Использование симметрии: (более эффективные алгоритмы для случая $A = A^*$ обсуждаются в гл. 5, более точно, в разд. 5.3).

- Регулярный пучок $A - \lambda B (\det(A - \lambda B) \not\equiv 0)$
  - Форма Вейерштрасса: для некоторых невырожденных матриц P_L и P_R имеем

$$A - \lambda B = P_L \cdot \mathrm{diag}\left(\text{Жордан}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix}^{n_i \times n_i} \right) P_R^{-1}.$$

  - Обобщенная форма Шура: для некоторых унитарных матриц Q_L и Q_R имеем $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, где обе матрицы T_A и T_B — треугольные.
  - Обобщенная вещественная форма Шура для вещественных матриц A и B: для некоторых вещественных ортогональных матриц Q_L и Q_R имеем $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^T$, где T_A и T_B — вещественные матрицы, причем T_A — квазитреугольная, а T_B — треугольная.
  - Приложение к ОДУ: позволяет решать задачу $B\dot{x}(t) = Ax(t) + f(t)$, решение которой может негладко зависеть от входных данных (*импульсный отклик*).
  - Алгоритм: приведение матриц A и B соответственно к хессенберговой и треугольной формам; последующее применение QZ-итерации (представляющей собой QR-итерацию, которая неявно проводится для матрицы AB^{-1}).
  - Стоимость: вычисление матриц T_A и T_B требует $30n^3$ флопов. Дополнительное вычисление матриц Q_L и Q_R увеличивает стоимость до $66n^3$ флопов. Если нужны и собственные векторы, то общая стоимость возрастает почти до $69n^3$ флопов. Как и выше, BLAS-операции уровня 3 могут использоваться не во всех частях алгоритма.
  - LAPACK: программы `sgges` для вычисления формы Шура и `sggev` для вычисления собственных значений и собственных векторов; аналогич-

ные программы `sggesx` и `sggevx`, дополнительно вычисляющие границы для погрешностей.
- Matlab: команда `eig` для вычисления собственных значений и собственных векторов.
- Использование симметрии: если $A = A^*$, $B = B^*$ и B положительно определена, то задача может быть сведена к вычислению собственных значений одной симметричной матрицы (см. доказательство теоремы 4.12). Так делается в LAPACK-программах `ssygv`, `sspgv` (для симметричных матриц в «упакованной форме») и `ssbgv` (для симметричных ленточных матриц).

- Сингулярный пучок $A - \lambda B$
  - Форма Кронекера: для некоторых невырожденных матриц P_L и P_R имеем

$$A - \lambda B = P_L \cdot \operatorname{diag}\left(\text{Вейерштрасс}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & 1 & \lambda \end{bmatrix}^{n_i \times (n_i+1)}, \begin{bmatrix} 1 & & \\ \lambda & \ddots & \\ & \ddots & 1 \\ & & \lambda \end{bmatrix}^{(m_i+1)\times m_i}\right) R_R^{-1}.$$

  - Обобщенная верхняя треугольная форма: для некоторых унитарных матриц Q_L и Q_R имеем $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, где T_A и T_B имеют обобщенную верхнюю треугольную форму, диагональные блоки которой соответствуют различным частям формы Кронекера. Подробности относительно формы и алгоритмов см. в [79, 246].
  - Стоимость: для самой общей и надежной версии алгоритма может доходить до $O(n^4)$ флопов в зависимости от особенностей кронекеровой структуры; это гораздо больше, чем для регулярного пучка $A-\lambda B$. Имеется также несколько менее надежный алгоритм со стоимостью $O(n^3)$ флопов [27].
  - Приложение к ОДУ: позволяет решать задачу $B\dot{x}(t) = Ax(t) + f(t)$, которая может быть переопределена или недоопределена.
  - Программы: см. NETLIB/linalg/guptri.

- Матричный пучок $\sum_{i=0}^{d} \lambda^i A_i$ [118]
  - Если $A_d = I$ (или квадратная матрица A_d достаточно хорошо обусловлена для того, чтобы каждую матрицу A_i можно было заменить матрицей $A_d^{-1}A_i$), то линеаризация приводит к стандартной задаче

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda I.$$

– Если матрица A_d плохо обусловлена или вырожденна, то задача линеаризуется к пучку

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda \begin{bmatrix} A_d & & & & & \\ & I & & & & \\ & & I & & & \\ & & & \ddots & & \\ & & & & & I \end{bmatrix}.$$

4.7. Литература и смежные вопросы к главе 4

Обсуждение свойств собственных значений и собственных векторов см. в [139]. Более подробное обсуждение теории возмущений для спектральных задач можно найти в [161, 237, 52] и гл. 4 книги [10]. Теорема 4.7 доказана в [69]. Вопрос о канонических формах Вейерштрасса и Кронекера освещается в [110, 118]. По поводу приложений этих форм к системам и теории управления см. [246, 247, 78]. Их приложения к вычислительной геометрии, графике и машинному проектированию рассматриваются в [181, 182, 165]. Параллельные алгоритмы для несимметричной проблемы собственных значений обсуждаются в [76].

4.8. Вопросы к главе 4

Вопрос 4.1 (*легкий*). Пусть матрица A определена уравнением (4.1). Показать, что $\det(A) = \prod_{i=1}^{b} \det(A_{ii})$ и, как следствие, $\det(A - \lambda I) = \prod_{i=1}^{b} \det(A_{ii} - \lambda I)$. Сделать отсюда вывод, что множество собственных значений матрицы A есть объединение множеств собственных значений блоков $A_{11}, \ldots, A_{bb}$.

Вопрос 4.2 (*средней трудности; Z. Bai*). Пусть A — *нормальная* матрица, т. е. $AA^* = A^*A$. Показать, что если A к тому же треугольная, то она диагональная. Опираясь на этот факт, доказать, что $n \times n$-матрица тогда и только тогда является нормальной, когда она имеет n ортонормированных собственных векторов. Указание: показать, что A — нормальная матрица в том и только в том случае, если ее форма Шура — нормальная матрица.

Вопрос 4.3 (*легкий; Z. Bai*). Пусть λ и μ — различные собственные значения матрицы A. Пусть числу λ соответствует правый собственный вектор x, а числу μ — левый собственный вектор y. Доказать, что x и y ортогональны.

Вопрос 4.4 (*средней трудности*). Предположим, что все собственные значения матрицы A различны. Пусть $f(z) = \sum_{i=-\infty}^{+\infty} a_i z^i$ — функция, определенная на множестве этих собственных значений. Пусть $Q^*AQ = T$ есть форма Шура матрицы A (так что Q унитарная, а T верхнетреугольная матрицы).

1. Показать, что $f(A) = Qf(T)Q^*$. Таким образом, чтобы вычислить $f(A)$, достаточно научиться вычислять $f(T)$. Остальная часть данной задачи имеет целью вывод простых рекуррентных формул для вычисления $f(T)$.
2. Показать, что $(f(T))_{ii} = f(T_{ii})$. Таким образом, диагональ матрицы $f(T)$ может быть вычислена по диагонали T.

3. Показать, что $Tf(T) = f(T)T$.
4. Используя этот результат, показать, что i-я наддиагональ матрицы $f(T)$ может быть вычислена по ее $(i-1)$-й и предшествующим наддиагоналям. Тем самым, отправляясь от главной диагонали матрицы $f(T)$, мы можем вычислить первую наддиагональ, затем вторую, и т. д.

Вопрос 4.5 (*легкий*). Пусть A — квадратная матрица с собственными значениями λ_i. Применяя к форме Шура матрицы A вопрос 4.4 либо к жордановой форме равенство (4.6), убедиться, что собственными значениями матрицы $f(A)$ являются числа $f(\lambda_i)$. Это утверждение называется *теоремой о спектральном отображении.*

Данный результат используется в доказательстве теоремы 6.5 и в разд. 6.5.6.

Вопрос 4.6 (*средней трудности*). В этой задаче будет показано, как решается уравнение *Сильвестра* $AX - XB = C$, где X и C — матрицы размера $m \times n$, A — матрица размера $m \times m$ и B — размера $n \times n$ (при $m = n$, $B = A$ это уравнение называется уравнением *Ляпунова*). Уравнение Сильвестра представляет собой систему из mn линейных уравнений относительно элементов матрицы X.

1. Пусть известны разложения Шура матриц A и B. Указать способ преобразования уравнения $AX - XB = C$ в уравнение $A'Y - YB' = C'$, где A' и B' — верхнетреугольные матрицы.
2. Указать способ последовательного вычисления элементов матрицы Y, аналогичный обратной подстановке. Какое условие для собственных значений матриц A и B гарантирует невырожденность системы уравнений относительно элементов искомой матрицы?
3. Указать способ преобразования матрицы Y в матрицу X.

Вопрос 4.7 (*средней трудности*). Предположим, что матрица $T = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$ имеет форму Шура. Требуется найти матрицу S, такую, что $S^{-1}TS = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$. Оказывается, что можно взять матрицу S вида $\begin{bmatrix} I & R \\ 0 & I \end{bmatrix}$. Указать способ вычисления матрицы R.

Вопрос 4.8 (*средней трудности; Z.Bai*). Пусть A — матрица размера $m \times n$, а B — матрица размера $n \times m$. Показать, что матрицы

$$\begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}$$

подобны. Вывести отсюда, что ненулевые собственные значения матриц AB и BA совпадают.

Вопрос 4.9 (*средней трудности; Z. Bai*). Пусть A — матрица порядка n с собственными значениями $\lambda_1, \ldots, \lambda_n$. Доказать, что

$$\sum_{i=1}^{n} |\lambda_i|^2 = \inf_{\det(S) \neq 0} \|S^{-1}AS\|_F^2.$$

Вопрос 4.10 (*средней трудности; Z. Bai*). Пусть A — матрица порядка n с собственными значениями $\lambda_1, \ldots, \lambda_n$.

1. Показать, что A может быть представлена в виде $A = H + S$, где $H = H^*$ — эрмитова матрица, а $S = -S^*$ — косоэрмитова матрица. Указать явные выражения для H и S через A.
2. Доказать, что $\sum_{i=1}^{n} |\Re\lambda_i|^2 \le \|H\|_F^2$.
3. Доказать, что $\sum_{i=1}^{n} |\Im\lambda_i|^2 \le \|S\|_F^2$.
4. Показать, что A тогда и только тогда является нормальной матрицей (т. е. $AA^* = A^*A$), когда $\sum_{i=1}^{n} |\lambda_i|^2 = \|A\|_F^2$.

Вопрос 4.11 (*легкий*). Пусть λ — простое собственное значение, а x и y — соответствующие собственные векторы, правый и левый. Определим *спектральный проектор*, ассоциированный с λ, формулой $P = xy^*/(y^*x)$. Доказать следующие свойства проектора P:

1. P определен единственным образом несмотря на то, что в его определении вместо векторов x и y можно использовать любые ненулевые их кратные.
2. $P^2 = P$. (Всякая матрица, удовлетворяющая соотношению $P^2 = P$, называется *проектором*).
3. $AP = PA = \lambda P$. (Эти равенства объясняют название «*спектральный* проектор», поскольку P «содержит в себе» информацию о левом и правом собственных подпространствах для собственного значения λ.)
4. Число обусловленности собственного значения λ равно $\|P\|_2$.

Вопрос 4.12 (*легкий; Z. Bai*). Пусть $A = \begin{bmatrix} a & c \\ 0 & b \end{bmatrix}$. Показать, что для обоих собственных значений матрицы A число обусловленности равно $\left(1 + \left(\frac{c}{a-b}\right)^2\right)^{1/2}$.

Таким образом, число обусловленности велико, если разность собственных значений $a - b$ мала в сравнении с числом c, характеризующим величину внедиагональной части матрицы.

Вопрос 4.13 (*средней трудности; Z. Bai*). Пусть A — матрица, x — нормированный вектор ($\|x\|_2 = 1$), μ — число и $r = Ax - \mu x$. Доказать, что найдется матрица E с нормой $\|E\|_F = \|r\|_2$, такая, что μ является собственным значением для $A + E$, а x — соответствующим собственным вектором.

Вопрос 4.14 (*средней трудности; программирование*). В этой задаче используется Matlab-программа, вычисляющая числа обусловленности собственных значений и выдающая на терминал диаграмму расположения собственных значений возмущенной матрицы. (Программа находится в HOMEPAGE/Matlab/eigscat.m.) Входными параметрами являются:

a = исходная матрица,
err = величина возмущения,
m = число обрабатываемых возмущенных матриц.

На выходе будут получены три диаграммы, на которых собственные значения возмущенных матриц указаны специальными символами:

символом «o» обозначается собственное значение исходной матрицы;
символ «x» обозначает собственное значение возмущенной матрицы, полученной из a добавлением вещественного возмущения с нормой err;

символ «.» обозначает собственное значение возмущенной матрицы, полученной из a добавлением комплексного возмущения с нормой err.

Кроме того, печатается таблица, содержащая собственные значения матрицы a и их числа обусловленности.

Вот примеры некоторых матриц, представляющих интерес для нашего эксперимента. (Постарайтесь взять как можно большее m; хороший выбор значения m — это число порядка нескольких сотен.)

```
(1) a = randn(5) (если у a нет комплексных собственных значений,
                  примените процедуру randn повторно)
       err=1e-5, 1e-4, 1e-3, 1e-2, .1, .2

(2) a = diag(ones(4,1),1); err=1e-12, 1e-10, 1e-8

(3) a=[[1 1e6 0     0]; ...
       [0 2   1e-3  0]; ...
       [0 0   3    10]; ...
       [0 0  -1     4]]
       err=1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3

(4) [q,r]=qr(randn(4,4));a=q*diag(ones(3,1),1)*q'
       err=1e-16, 1e-14, 1e-12, 1e-10, 1e-8

(5) a = [[1 1e3 1e6];[0 1 1e3];[0 0 1]],
       err=1e-7, 1e-6, 5e-6, 8e-6, 1e-5, 1.5e-5, 2e-5

(6) a = [[1   0   0   0   0   0]; ...
         [0   2   1   0   0   0]; ...
         [0   0   2   0   0   0]; ...
         [0   0   0   3 1e2 1e4]; ...
         [0   0   0   0   3 1e2]; ...
         [0   0   0   0   0   3]]
       err=1e-10, 1e-8, 1e-6, 1e-4, 1e-3
```

Ваша задача состоит в том, чтобы провести вычисления для этих примеров и сравнить области, локализующие вычисленные собственные значения (так называемый *псевдоспектр* матрицы), с границами ошибок, указанными в разд. 4.3. Каково различие между вещественными и комплексными возмущениями? Что происходит с областями локализации собственных значений, когда возмущение err стремится к нулю? Каков предельный размер этих областей при err, стремящемся к нулю (т. е. сколько верных разрядов имеют вычисленные собственные значения)?

Вопрос 4.15 (*средней трудности; программирование*). В этой задаче используется Matlab-программа, которая выводит на терминал графики диагональных элементов матрицы, обрабатываемой QR-итерацией без сдвигов. Каждому диагональному элементу на графике соответствует своя кривая, полученная из значений этого элемента после каждой QR-итерации. (Программа находится в

HOMEPAGE/Matlab/qrplt.m; ниже мы приводим ее текст.) Входными параметрами являются:

a = исходная матрица,
m = число QR-итераций.

На выходе будут получены графики диагональных элементов.

Вот примеры, для которых нужно использовать эту программу (возьмите достаточно большое значение m с тем, чтобы кривые либо успели сойтись к своему предельному значению, либо обнаружили циклическое поведение):

```
a = randn(6);
b = randn(6); a = b*diag([1,2,3,4,5,6])*inv(b);
a = [[1 10];[-1 1]]; m = 300
a = diag((1.5*ones(1,5)) +
    .01*(diag(ones(4,1),1)+diag(ones(4,1),-1)); m=30
```

Что происходит в том случае, если у матрицы есть комплексные собственные значения?

В каком порядке выстраиваются на диагонали собственные значения матрицы после многих итераций?

Проделайте следующий эксперимент: пусть a — симметричная $n \times n$-матрица. Выполним команду Matlab'a perm=(n:-1:1). Это породит список целых чисел от n до 1 в порядке убывания. Проведем m шагов QR-итерации. Положим a=a(perm,perm); мы называем эту операцию «зеркальным отражением» матрицы a, поскольку она изменяет порядок строк и столбцов на обратный. Снова проделаем m шагов QR-итерации и опять положим a=a(perm,perm). Как эта матрица соотносится с первоначальной? Значение m не должно быть слишком большим (попробуйте взять $m = 5$), иначе ошибки округлений могут помешать вам выявить искомую взаимосвязь. (См. также следствие 5.4 и вопрос 5.25.)

Измените программу так, чтобы она вычисляла разности между диагональными элементами на каждом шаге и их предельными значениями (сделайте это в предположении, что все собственные значения матрицы вещественны). Выведите на терминал графики логарифмов указанных разностей как функций от номера шага m. Как выглядят эти графики асимптотически?

```
hold off
e=diag(a);
for i=1:m,
   [q,r]=qr(a);dd=diag(sign(diag(r)));r=dd*r;q=q*dd;a=r*q; ...
   e=[e,diag(a)];
end
clg
plot(e','w'),grid
```

Вопрос 4.16 (*трудный; программирование*). В этой задаче описывается приложение нелинейной проблемы собственных значений к компьютерной графике, вычислительной геометрии и автоматизированному проектированию; см. также [181, 182, 165].

Пусть $F = [f_{ij}(x_1, x_2, x_3)]$ — матрица, элементами которой являются многочлены от трех переменных x_i. В общем случае, уравнение $\det(F) = 0$ опреде-

ляет некоторую двумерную поверхность S в трехмерном пространстве. Пусть уравнения $x_1 = g_1(t)$, $x_2 = g_2(t)$, $x_3 = g_3(t)$ задают (одномерную) кривую C с параметром t; функции g_i также являются многочленами. Требуется найти пересечение $S \cap C$. Сформулировать эту задачу как проблему собственных значений (что дает способ ее численного решения). Более общо, указать, как следует находить пересечение поверхности $\det(F(x_1, \ldots, x_n)) = 0$ и кривой $\{x_i = g_i(t),\ 1 \le i \le n\}$. Найти максимальное возможное число дискретных решений как функцию от n, порядка d матрицы F и максимальной из степеней многочленов f_{ij} и g_k.

Написать Matlab-программу, решающую эту задачу с $n = 3$ переменными путем сведения к проблеме собственных значений. Входом программы должно быть компактное описание коэффициентов всех многочленов $f_{ij}(x_k)$ и $g_i(t)$, а выходом — список точек пересечения. Например, входная информация может состоять из следующих массивов:

- массива NumTerms$(1 : d, 1 : d)$, где NumTerms (i, j) есть число одночленов в многочлене $f_{ij}(x_1, x_2, x_3)$;
- массива Stems$(1 : 4, 1$:TotalTerms), где TotalTerms есть сумма всех элементов массива NumTerms. Каждый столбец массива Sterms дает описание одного одночлена. При этом первые NumTerms(1,1) столбцов массива Sterms представляют одночлены, входящие в многочлен f_{11}, следующие NumTerms(2,1) столбцов массива Sterms представляют одночлены из f_{21}, и т. д. Одночлен, описываемый столбцом Sterms(1:4,k), имеет вид $\mathrm{Sterm}(4,k) \cdot x_1^{\mathrm{Sterm}(1,k)} \cdot x_2^{\mathrm{Sterm}(2,k)} \cdot x_3^{\mathrm{Sterm}(3,k)}$;
- массива tC(1:3), содержащего степени многочленов g_1, g_2 и g_3 в указанном порядке;
- массива Curve(1: tC(1)+tC(2)+tC(3)+3), содержащего коэффициенты многочленов g_1, g_2 и g_3. Многочлены следуют один за другим и коэффициенты каждого перечисляются в порядке от свободного члена к коэффициенту при наивысшей степени.

Ваша программа должна еще вычислять оценки ошибок для полученных результатов. Это будет возможно лишь, если к решаемой задаче на собственные значения применимы оценки ошибок из теорем 4.4 или 4.5. Если вы встретитесь с более общим случаем, то вычисление границ для ошибок опускается. (Оценки ошибок для более общих спектральных задач можно найти в [10, 237].)

Написать еще одну Matlab-программу, которая для случая $m = 3$ выводила бы графики поверхности S и кривой C, а также помечала бы точки пересечения.

Предусмотрены ли в ваших программах какие-либо ограничения на входные данные? Что произойдет, если S и C не пересекаются? Что произойдет, если C целиком принадлежит S?

Посчитайте с помощью ваших программ, по крайней мере, приводимые ниже примеры. Для первых пяти из них вы сможете найти ответ посредством ручных вычислений. Это поможет проверить правильность работы программ.

1. $g_1 = t$, $g_2 = 1 + t$, $g_3 = 2 + t$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

2. $g_1 = t^3$, $g_2 = 1 + t^3$, $g_3 = 2 + t^3$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.
3. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.
4. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} 1 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 9 \end{bmatrix}$.
5. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 8 \end{bmatrix}$.
6. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & x_1 \\ x_3 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.
7. $g_1 = 7 - 3t + t^5$, $g_2 = 1 + t^2 + t^5$, $g_3 = 2 + t^2 - t^5$,

$$F = \begin{bmatrix} x_1x_2 + x_3^5 & 3 - x_2^2 & 5 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 \\ x_2 - 7x_3^5 & 1 - x_1^2 + x_1x_2x_3^3 & 3 + x_1 + 3x_3 - 9x_2x_3 \\ 2 & 3x_1 + 5x_2 - 7x_3 + 8 & x_1^3 - x_2^4 + 4x_3^5 \end{bmatrix}.$$

Ваш отчет по этому заданию должен состоять из следующих пунктов:

• математическая постановка задачи в терминах проблемы собственных значений;

• описание алгоритма, самое большое, на двух страницах. Оно должно включать в себя «дорожную карту» вашей программы (т. е. имена подпрограмм для всех операций высокого уровня). Из этого описания должно быть видно, как математическая постановка ведет к алгоритму и как соответствуют друг другу алгоритм и программа;

• ответы на следующие вопросы:

— каково максимальное возможное число дискретных решений?

— все ли вычисленные собственные значения представляют реальные пересечения? Для каких собственных значений это верно?

— накладывает ли ваша программа какие-либо ограничения на входные данные?

— что происходит, если S и C не пересекаются?

— что происходит, если S содержит C?

• математическое описание оценок ошибок;

• описание алгоритма, вычисляющего оценки ошибок, самое большое, на двух страницах. Оно должно включать в себя «дорожную карту» вашей программы (т. е. имена подпрограмм для всех операций высокого уровня). Из этого описания должно быть видно, как математическая формулировка ведет к алгоритму и как соответствуют друг другу алгоритм и программа;

• листинг программы.

Для каждого из семи примеров в отчет должны входить:

• исходная постановка задачи;

• полученная из нее задача на собственные значения;

• вычисленные решения;

• графики S и C. Соответствуют ли этим графикам вычисленные решения?

• результат подстановки вычисленных решений в уравнения, определяющие S и C. Удовлетворяются ли эти уравнения (в пределах точности вычислений)?

Глава 5

Симметричная проблема собственных значений и сингулярное разложение

5.1. Введение

В разделе 5.2 обсуждается теория возмущений для симметричной проблемы собственных значений, в разд. 5.3 и 5.4 — алгоритмы ее решения, а в разд. 5.5 (а также в других разделах) — ее приложения. Рассматривается также родственная задача о сингулярном разложении (SVD) матрицы. Поскольку спектральное разложение симметричной матрицы $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$ очень просто связано с SVD матрицы A (см. теорему 3.3), большинство теорем о возмущениях и алгоритмов для симметричной проблемы собственных значений переносятся на SVD.

В соответствии с обсуждением в начале гл. 4, алгоритмы для симметричной проблемы собственных значений (и SVD) в первом приближении можно разделить на две группы: *прямые* и *итерационные методы*. В этой главе рассматриваются только прямые методы. Они предназначены для вычисления всех (или избранного подмножества) собственных значений и (если нужно) собственных векторов. Их стоимость для плотных матриц составляет $O(n^3)$ операций. Итерационные методы обсуждаются в гл. 7.

В последние годы был достигнут значительный прогресс в алгоритмике и приложениях симметричных задач на собственные значения. Проиллюстрируем его тремя примерами.

- В разделе 5.3.3 обсуждается алгоритм для решения симметричной проблемы, основанный на стратегии «разделяй-и-властвуй». Это самый быстрый из имеющихся методов вычисления всех собственных значений и собственных векторов полной или ленточной симметричной матрицы высокого порядка (а также SVD произвольной матрицы). Он значительно быстрее QR-итерации, которая прежде была «рабочей лошадкой» для этого класса задач[1].
- В разделах 5.2.1, 5.4.2 и 5.4.3 обсуждаются высокоточные алгоритмы, основанные на методе Якоби и дифференциальном алгоритме частных и раз-

[1] Из недавних публикаций [201, 203] может возникнуть еще более быстрый и точный алгоритм, основанный на идеях обратной итерации (см. алгоритм 4.2). Однако по состоянию на июнь 1997 г. и теория, и программы алгоритма не приняли еще завершенного вида.

ностей (dqds). Эти алгоритмы способны находить малые собственные значения (или сингулярные числа) с большей точностью, чем альтернативные методы типа «разделяй-и-властвуй», хотя, как и метод Якоби, они работают более медленно.

- В разделе 5.5 исследуется «нелинейная» колебательная система, которая описывается дифференциальным уравнением, называемым *потоком Тода*. Ее непрерывное решение тесно связано с промежуточными шагами QR-алгоритма для симметричной проблемы собственных значений.

Как и в гл. 4, мы будем постоянно обращаться к примеру связанной системы материальных точек для иллюстрации особенностей симметричных задач на собственные значения.

Пример 5.1. Симметричные задачи на собственные значения часто возникают при анализе *механических колебаний*. Подробно рассмотренный образец такого анализа был представлен в примере 4.1. Мы используем обозначения из этого примера, поэтому рекомендуем читателю вначале заглянуть туда. Чтобы задача в примере 4.1 стала симметричной, нужно предположить отсутствие демпфирования. Тогда дифференциальное уравнение движения связанной системы принимает вид $M\ddot{x}(t) = -Kx(t)$, где $M = \mathrm{diag}(m_1, \ldots, m_n)$ и

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & & & \\ -k_2 & k_2 + k_3 & -k_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -k_{n-1} & k_{n-1} + k_n & -k_n \\ & & & -k_n & k_n \end{bmatrix}.$$

Поскольку M — невырожденная матрица, уравнение можно переписать как $\ddot{x}(t) = -M^{-1}Kx(t)$. Разыскивая решения вида $x(t) = e^{\gamma t}x(0)$, получаем $e^{\gamma t}\gamma^2 x(0) = -M^{-1}Ke^{\gamma t}x(0)$, или $M^{-1}Kx(0) = -\gamma^2 x(0)$. Иначе говоря, $-\gamma^2$ есть собственное значение матрицы $M^{-1}K$, а $x(0)$ — соответствующий собственный вектор. В общем случае, матрица $M^{-1}K$ не симметрична, однако мы можем симметризовать ее. Положим $M^{1/2} = \mathrm{diag}(m_1^{1/2}, \ldots, m_n^{1/2})$ и умножим обе части соотношения $M^{-1}Kx(0) = -\gamma^2 x(0)$ на $M^{1/2}$. Получим

$$M^{-1/2}Kx(0) = M^{-1/2}K(M^{-1/2}M^{1/2})x(0) = -\gamma^2 M^{1/2}x(0),$$

или $\widehat{K}\hat{x} = -\gamma^2\hat{x}$, где $\hat{x} = M^{1/2}x(0)$ и $\widehat{K} = M^{-1/2}KM^{-1/2}$. Легко видеть, что матрица

$$\widehat{K} = \begin{bmatrix} \frac{k_1+k_2}{m_1} & \frac{-k_2}{\sqrt{m_1 m_2}} & & & \\ \frac{-k_2}{\sqrt{m_1 m_2}} & \frac{k_2+k_3}{m_2} & \frac{-k_3}{\sqrt{m_2 m_3}} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{-k_{n-1}}{\sqrt{m_{n-2} m_{n-1}}} & \frac{k_{n-1}+k_n}{m_{n-1}} & \frac{-k_n}{\sqrt{m_{n-1} m_n}} \\ & & & \frac{-k_n}{\sqrt{m_{n-1} m_n}} & \frac{k_n}{m_n} \end{bmatrix}$$

симметрична. Поэтому каждое ее собственное значение $-\gamma^2$ вещественно и каждый собственный вектор $\hat{x} = M^{1/2}x(0)$ ортогонален другим собственным векторам.

Матрица $\widehat{K}$ — *трехдиагональная*. К этой специальной форме можно привести любую симметричную матрицу, используя алгоритм 4.6, модифицированный для симметричного случая в разд. 4.4.7. Большинство алгоритмов вычисления собственных значений и собственных векторов симметричной матрицы, описываемых в разд. 5.3, предполагают, что матрица была вначале приведена к трехдиагональной форме.

Пользуясь сингулярным разложением, можно предложить и другую интерпретацию задачи о механических колебаниях. Положим $K_D = \mathrm{diag}(k_1, \ldots, k_n)$ и $K_D^{1/2} = \mathrm{diag}(k_1^{1/2}, \ldots, k_n^{1/2})$. Тогда K может быть разложена в произведение $K = BK_DB^T$, где

$$B = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix}.$$

Это можно проверить несложными вычислениями. Таким образом,

$$\begin{aligned} \widehat{K} &= M^{-1/2}KM^{-1/2} \\ &= M^{-1/2}BK_DB^TM^{-1/2} \\ &= (M^{-1/2}BK_D^{1/2}) \cdot (K_D^{1/2}B^TM^{-1/2}) \\ &= (M^{-1/2}BK_D^{1/2}) \cdot (M^{-1/2}BK_D^{1/2})^T \\ &\equiv GG^T. \end{aligned} \tag{5.1}$$

Следовательно, сингулярные числа матрицы $G = M^{-1/2}BK_D^{1/2}$ суть квадратные корни из собственных значений матрицы $\widehat{K}$, а левые сингулярные векторы первой матрицы являются собственными векторами для второй (см. теорему 3.3). Отметим, что в G отличны от нуля только элементы главной диагонали и первой наддиагонали. Такие матрицы называются *двухдиагональными*; большинство алгоритмов для вычисления SVD начинают свою работу с приведения матрицы к двухдиагональному виду, используя для этого алгоритм из разд. 4.4.7.

Из разложения $\widehat{K} = GG^T$ следует, что матрица $\widehat{K}$ положительно определена (поскольку G невырожденна). Поэтому все собственные значения $-\gamma^2$ матрицы $\widehat{K}$ суть положительные числа. Следовательно, γ — чисто мнимые числа, а решения $x(t) = e^{\gamma t}x(0)$ исходного дифференциального уравнения являются осциллирующими функциями с частотами $|\gamma|$.

Matlab-программа, относящаяся к колебательным связанным системам, находится в HOMEPAGE/Matlab/massspring.m. В Matlab'е имеется и анимация колебаний сходной физической системы. Чтобы запустить ее, введите команду demo, а затем щелкните мышью на continue/fun-extras/miscellaneous/bending. ◇

5.2. Теория возмущений

Пусть A — симметричная матрица с собственными значениями $\alpha_1 \geq \cdots \geq \alpha_n$ и соответствующими нормированными собственными векторами $q_1, \ldots, q_n$. Пусть E — также симметричная матрица, а $\widehat{A} = A + E$ имеет (возмущенные) собственные значения $\hat{\alpha}_1 \geq \cdots \geq \hat{\alpha}_n$ и соответствующие (возмущенные) собственные векторы $\hat{q}_1, \ldots, \hat{q}_n$. Нашей главной целью в данном разделе будет оценивание разностей между собственными значениями α_i и $\hat{\alpha}_i$ и собственными векторами q_i и $\hat{q}_i$ через «величину» матрицы E. Большинство наших оценок принимают в качестве меры величины норму $\|E\|_2$. Исключением является разд. 5.2.1, где обсуждается теория «относительных» возмущений.

Первая оценка для возмущений собственных значений была уже получена в гл. 4. Там было доказано следствие 4.1: *пусть A — симметричная матрица с собственными значениями $\alpha_1 \geq \cdots \geq \alpha_n$. Пусть матрица $A+E$ также симметрична и имеет собственные значения $\hat{\alpha}_1 \geq \cdots \geq \hat{\alpha}_n$. Если α_i — простое собственное значение, то $|\alpha_i - \hat{\alpha}_i| \leq \|E\|_2 + O(\|E\|_2^2)$.*

Слабость этого результата в том, что он предполагает простоту собственного значения и полезен лишь для достаточно малых значений $\|E\|_2$. Следующая теорема устраняет оба этих недостатка.

Теорема 5.1 (Вейль). *Пусть A и E — симметричные $n \times n$-матрицы. Пусть $\alpha_1 \geq \cdots \geq \alpha_n$ — собственные значения матрицы A, а $\hat{\alpha}_1 \geq \cdots \geq \hat{\alpha}_n$ — собственные значения матрицы $\widehat{A} = A + E$. Тогда $|\alpha_i - \hat{\alpha}_i| \leq \|E\|_2$.*

Следствие 5.1. *Пусть G и F — произвольные матрицы одинакового размера, причем G имеет сингулярные числа $\sigma_1 \geq \cdots \geq \sigma_n$, а $G+F$ — сингулярные числа $\sigma_1' \geq \cdots \geq \sigma_n'$. Тогда $|\sigma_i - \hat{\sigma}_i| \leq \|F\|_2$.*

С помощью теоремы Вейля можно получить оценки ошибок для приближенных собственных значений, вычисленных любым обратно устойчивым алгоритмом, например QR-итерацией. Действительно, такой алгоритм вычисляет приближения $\hat{\alpha}_i$, которые являются точными собственными значениями для матрицы $\widehat{A} = A + E$, где $\|E\|_2 = O(\varepsilon)\|A\|_2$. Поэтому ошибки в этих приближениях можно оценить так: $|\alpha_i - \hat{\alpha}_i| \leq \|E\|_2 = O(\varepsilon)\|A\|_2 = O(\varepsilon)\max_j |\alpha_j|$. Это вполне удовлетворительная оценка, особенно для больших собственных значений (тех α_i, что сравнимы по величине с $\|A\|_2$), которые могут быть вычислены с большинством верных разрядов. В малых собственных значениях ($|\alpha_i| \ll \|A\|_2$) число верных разрядов будет меньше (однако см. разд. 5.2.1).

Мы докажем теорему Вейля, опираясь на другой полезный классический результат, а именно минимаксную теорему Куранта—Фишера. Чтобы сформулировать эту теорему, потребуется ввести *отношение Рэлея* (Rayleigh), которое будет играть важную роль и в нескольких алгоритмах, например в алгоритме 5.1.

Определение 5.1. Отношение Рэлея *для симметричной матрицы A и ненулевого вектора u — это число* $\rho(u, A) \equiv (u^T A u)/(u^T u)$.

Приведем некоторые простые, но важные свойства числа $\rho(u, A)$. Во-первых, $\rho(\gamma u, A) = \rho(u, A)$ для всякого ненулевого скаляра γ. Во-вторых, если $Aq_i = \alpha_i q_i$, то $\rho(q_i, A) = \alpha_i$. Более общо, предположим, что A имеет спек-

тральное разложение $Q^TAQ = \Lambda = \mathrm{diag}(\alpha_i)$, где $Q = [q_1, \ldots, q_n]$. Разложим u по собственным векторам q_i: $u = Q(Q^Tu) \equiv Q\xi = \sum_i q_i\xi_i$. Тогда

$$\rho(u, A) = \frac{\xi^TQ^TAQ\xi}{\xi^TQ^TQ\xi} = \frac{\xi^T\Lambda\xi}{\xi^T\xi} = \frac{\sum_i \alpha_i\xi_i^2}{\sum_i \xi_i^2}.$$

Таким образом, $\rho(u, A)$ есть взвешенное среднее собственных значений матрицы A. Наибольшее значение $\max_{u\neq 0} \rho(u, A)$ достигается для $u = q_1$ ($\xi = e_1$) и равно $\rho(q_1, A) = \alpha_1$. Наименьшее значение $\min_{u\neq 0} \rho(u, A)$ реализуется вектором $u = q_n$ ($\xi = e_n$) и равно $\rho(q_n, A) = \alpha_n$. Вместе эти результаты означают, что

$$\max_{u\neq 0} |\rho(u, A)| = \max(|\alpha_1|, |\alpha_n|) = \|A\|_2. \tag{5.2}$$

Теорема 5.2 (минимаксная теорема Куранта—Фишера). *Пусть A — симметричная матрица с собственными значениями $\alpha_1 \geq \cdots \geq \alpha_n$ и соответствующими нормированными собственными векторами $q_1, \ldots, q_n$. Тогда*

$$\max_{\mathbf{R}^j} \min_{0\neq r\in\mathbf{R}^j} \rho(r, A) = \alpha_j = \min_{\mathbf{S}^{n-j+1}} \max_{0\neq s\in\mathbf{S}^{n-j+1}} \rho(s, A).$$

В левой формуле для α_j максимум берется по всем j-мерным подпространствам $\mathbf{R}^j$ пространства $\mathbf{R}^n$, а внутренний минимум — по всем ненулевым векторам r из $\mathbf{R}^j$. Максимум достигается для подпространства $\mathbf{R}^j = \mathrm{Span}(q_1, q_2, \ldots, q_j)$, а минимум реализуется вектором $r = q_j$.

В правой формуле для α_j минимум берется по всем подпространствам $\mathbf{S}^{n-j+1}$ размерности $n - j + 1$, а внутренний максимум — по всем ненулевым векторам s из $\mathbf{S}^{n-j+1}$. Минимум достигается для подпространства $\mathbf{S}^{n-j+1} = \mathrm{Span}(q_j, q_{j+1}, \ldots, q_n)$, а максимум реализуется вектором $s = q_j$.

Пример 5.2. Пусть $j = 1$, тогда α_j есть наибольшее собственное значение. Если $\mathbf{R}^1$ задано, то $\rho(r, A)$ имеет одно и то же значение для всех ненулевых векторов $r \in \mathbf{R}^1$, поскольку эти векторы пропорциональны друг другу. Поэтому левое выражение для α_1 упрощается к виду $\alpha_1 = \max_{r\neq 0} \rho(r, A)$. Точно так же, поскольку $n - j + 1 = n$, единственным подпространством $\mathbf{S}^{n-j+1}$ является все пространство $\mathbf{R}^n$. В результате, правое выражение для α_1 также упрощается и принимает вид $\alpha_1 = \max_{s\neq 0} \rho(s, A)$.

Аналогичным образом можно показать, что для наименьшего собственного значения теорема упрощается до формулы $\alpha_n = \min_{r\neq 0} \rho(r, A)$. ◇

Доказательство минимаксной теоремы Куранта—Фишера. Возьмем произвольные подпространства $\mathbf{R}^j$ и $\mathbf{S}^{n-j+1}$ указанных в формулировке теоремы размерностей. Поскольку сумма их размерностей $j + (n - j + 1) = n + 1$ больше, чем n, найдется ненулевой вектор $x_{\mathbf{RS}} \in \mathbf{R}^j \cap \mathbf{S}^{n-j+1}$. Таким образом,

$$\min_{0\neq r\in\mathbf{R}^j} \rho(r, A) \leq \rho(x_{\mathbf{RS}}, A) \leq \max_{0\neq s\in\mathbf{S}^{n-j+1}} \rho(s, A).$$

Выберем теперь $\widehat{\mathbf{R}}^j$ так, чтобы выражение слева принимало наибольшее значение, и $\widehat{\mathbf{S}}^{n-j+1}$ так, чтобы выражение справа было минимальным. Тогда

$$\begin{aligned}\max_{\mathbf{R}^j}\min_{0\neq r\in\mathbf{R}^j}\rho(r,A) &= \min_{0\neq r\in\widehat{\mathbf{R}}^j}\rho(r,A) \\ &\le \rho(x_{\widehat{\mathbf{R}}\widehat{\mathbf{S}}},A) \\ &\le \max_{0\neq s\in\widehat{\mathbf{S}}^{n-j+1}}\rho(s,A) \\ &= \min_{\mathbf{S}^{n-j+1}}\max_{0\neq s\in\mathbf{S}^{n-j+1}}\rho(s,A).\end{aligned} \tag{5.3}$$

Убедимся теперь, что неравенства в этих выкладках в действительности являются равенствами. С этой целью укажем конкретные подпространства $\mathbf{R}^j$ и $\mathbf{S}^{n-j+1}$, для которых нижняя граница совпадает с верхней. Вначале положим $\underline{\mathbf{R}}^j = \mathrm{Span}(q_1,\dots,q_j)$. Тогда имеем

$$\begin{aligned}\max_{\mathbf{R}^j}\min_{0\neq r\in\mathbf{R}^j}\rho(r,A) &\ge \min_{0\neq r\in\underline{\mathbf{R}}^j}\rho(r,A) \\ &= \min_{0\neq r=\sum_{i\le j}\xi_i q_i}\rho(r,A) \\ &= \min_{\text{не все }\xi_i\text{ равны }0}\frac{\sum_{i\le j}\xi_i^2\alpha_i}{\sum_{i\le j}\xi_i^2}=\alpha_j.\end{aligned}$$

Теперь возьмем $\overline{\mathbf{S}}^{n-j+1} = \mathrm{Span}(q_j,\dots,q_n)$. Имеем

$$\begin{aligned}\min_{\mathbf{S}^{n-j+1}}\max_{0\neq s\in\mathbf{S}^{n-j+1}}\rho(s,A) &\le \max_{0\neq s\in\overline{\mathbf{S}}^{n-j+1}}\rho(s,A) \\ &= \max_{0\neq s=\sum_{i\ge j}\xi_i q_i}\rho(s,A) \\ &= \max_{\text{не все }\xi_i\text{ равны }0}\frac{\sum_{i\ge j}\xi_i^2\alpha_i}{\sum_{i\ge j}\xi_i^2}=\alpha_j.\end{aligned}$$

Итак, α_j одновременно и меньше, и больше обеих границ, которые поэтому совпадают друг с другом и с α_j. □

Пример 5.3. Рис. 5.1 иллюстрирует теорему для случая 3×3-матриц. Поскольку $\rho(u/\|u\|_2,A)=\rho(u,A)$, то можно интерпретировать $\rho(u,A)$ как функцию, заданную на единичной сфере $\|u\|_2=1$. На рис. 5.1 показаны линии уровня этой функции для матрицы $A=\mathrm{diag}(1,.25,0)$. Для этой простой матрицы имеем $q_i=e_i$, где e_i есть i-й столбец единичной матрицы. Картина линий уровня симметрична относительно начала координат в силу свойства $\rho(u,A)=\rho(-u,A)$. Малые пунктирные окружности вблизи $\pm q_1$ опоясывают точки глобального максимума ($\rho(\pm q_1,A)=1$), а малые сплошные окружности вблизи $\pm q_3$ — точки глобального минимума ($\rho(\pm q_3,A)=0$). Две большие окружности составляют линию уровня $\rho(u,A)=.25$, соответствующую среднему собственному значению. Внутри двух узких «ломтей», определяемых большими кругами, имеем $\rho(u,A)<.25$, а внутри широких ломтей выполняется неравенство $\rho(u,A)>.25$.

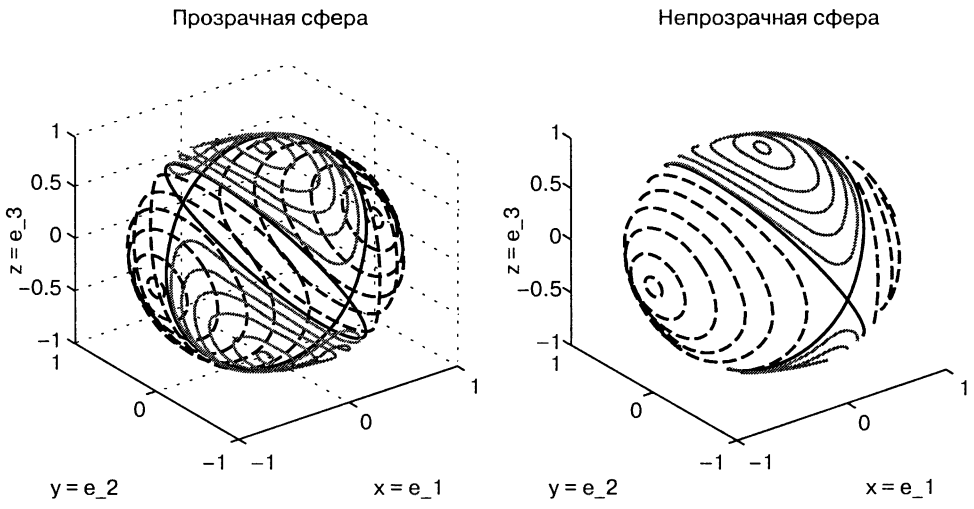


Рис. 5.1. Линии уровня отношения Рэлея на единичной сфере.

Дадим интерпретацию минимаксной теоремы с помощью этого рисунка. Выбор подпространства $\mathbf{R}^2$ равносилен выбору большой окружности C: каждая точка C принадлежит соответствующему подпространству $\mathbf{R}^2$ и всё $\mathbf{R}^2$ состоит из векторов, пропорциональных векторам из C. Таким образом, $\min_{0 \neq r \in \mathbf{R}^2} \rho(r, A) = \min_{r \in C} \rho(r, A)$. При вычислении $\min_{r \in C} \rho(r, A)$ нужно рассматривать четыре случая:

1. C не проходит через точки пересечения $\pm q_2$ двух больших окружностей на рис. 5.1. Ясно, что в этом случае C должна пересекать и узкий ломоть, и широкий; следовательно, $\min_{r \in C} \rho(r, A) < .25$.
2. C проходит через точки пересечения $\pm q_2$, а в остальном находится внутри узких ломтей. Тогда $\min_{r \in C} \rho(r, A) < .25$.
3. C проходит через точки пересечения $\pm q_2$, а в остальном находится внутри широких ломтей. В таком случае $\min_{r \in C} \rho(r, A) = .25$, и этот минимум достигается для $r = \pm q_2$.
4. C совпадает с одной из двух больших окружностей. Тогда $\rho(r, A) = .25$ для всех $r \in C$.

Согласно минимаксной теореме, $\alpha_2 = .25$ есть максимум функции $\min_{r \in C} \rho(r, A)$ относительно всех выборов большой окружности C. Этот максимум достигается в случаях 3 и 4. Если, в частности, C делит широкий ломоть пополам (случай 3), то получаем $\mathbf{R}^2 = \mathrm{Span}(q_1, q_2)$.

Программы для вычерчивания линий уровня типа линий на рис. 5.1, но для случая произвольной симметричной 3×3 матрицы, можно найти на HOMEPAGE/Matlab/RayleighContour.m. ◇

Мы можем теперь приступить к *доказательству теоремы Вейля*:

$$\begin{aligned}
\hat{\alpha}_j &= \min_{\mathbf{S}^{n-j+1}} \max_{0\neq u\in\mathbf{S}^{n-j+1}} \frac{u^T(A+E)u}{u^Tu} \quad \text{по минимаксной теореме} \\
&= \min_{\mathbf{S}^{n-j+1}} \max_{0\neq u\in\mathbf{S}^{n-j+1}} \left(\frac{u^TAu}{u^Tu} + \frac{u^TEu}{u^Tu}\right) \\
&\leq \min_{\mathbf{S}^{n-j+1}} \max_{0\neq u\in\mathbf{S}^{n-j+1}} \left(\frac{u^TAu}{u^Tu} + \|E\|_2\right) \quad \text{согласно (5.2)} \\
&= \alpha_i + \|E\|_2 \quad \text{снова используя минимаксную теорему.}
\end{aligned}$$

Меняя A и $A+E$ ролями, получим неравенство $\alpha_j \leq \hat{\alpha}_j + \|E\|_2$. Вместе эти два неравенства доказывают теорему Вейля.

С минимаксной теоремой Куранта—Фишера тесно связана теорема Сильвестра об инерции. Она понадобится нам в разд. 5.3.4 для обоснования алгоритма бисекции.

Определение 5.2. Инерцией *симметричной матрицы* A *называется тройка целых чисел* $\mathrm{Inertia}A \equiv (\nu, \zeta, \pi)$, *где* ν, ζ *и* π *суть соответственно число отрицательных, нулевых и положительных собственных значений этой матрицы.*

Если X — ортогональная матрица, то матрицы X^TAX и A подобны, а потому имеют одни и те же собственные значения. Если относительно X предполагается только невырожденность, то матрицы X^TAX и A называют *конгруэнтными*. В этом случае собственные значения матриц X^TAX и A, вообще говоря, различны. Однако, как показывает следующая теорема, оба множества собственных значений имеют, по крайней мере, одни и те же знаки.

Теорема 5.3 (теорема Сильвестра об инерции). *Пусть* A *симметричная, а* X *невырожденная матрицы. Тогда матрицы* A *и* X^TAX *имеют одну и ту же инерцию.*

Доказательство. Пусть n — порядок матрицы A. Предположим, что A имеет ν отрицательных собственных значений, а аналогичное число ν' для X^TAX меньше, чем ν. Мы покажем, что это невозможно, приведя данное предположение к противоречию. Пусть $\mathbf{N}$ обозначает ν-мерное отрицательное собственное подпространство матрицы A, т. е. подпространство, натянутое на собственные векторы для ν отрицательных собственных значений. Это значит, что $x^TAx < 0$ для всякого ненулевого вектора $x \in \mathbf{N}$. Пусть $\mathbf{P}$ — неотрицательное собственное подпространство размерности $n-\nu'$ матрицы X^TAX; имеем $x^TX^TAXx \geq 0$ для всякого ненулевого вектора $x \in \mathbf{P}$. Поскольку X — невырожденная матрица, подпространство $X\mathbf{P}$ также имеет размерность $n-\nu'$. Из соотношений $\dim(\mathbf{N}) + \dim(X\mathbf{P}) = \nu + n - \nu' > n$ следует, что в пересечении подпространств $\mathbf{N}$ и $X\mathbf{P}$ должен содержаться ненулевой вектор x. Но тогда $0 > x^TAx$, поскольку $x \in \mathbf{N}$, и $0 \leq x^TAx$, поскольку $x \in X\mathbf{P}$. Это противоречие доказывает, что $\nu \leq \nu'$. Меняя матрицы A и X^TAX ролями, получим неравенство $\nu' \leq \nu$. Таким образом, A и X^TAX имеют одинаковое число отрицательных собственных значений. Аналогичное рассуждение показывает, что обе матрицы имеют одно и то же число положительных собственных значений.

Поэтому и число нулевых собственных значений у них должно быть одинаковым. □

Теперь мы исследуем, насколько могут измениться собственные векторы при переходе от матрицы A к возмущенной матрице $A+E$. Чтобы сформулировать оценку, нам потребуется понятие *отделенности* собственного значения.

Определение 5.3. *Пусть* $\alpha_1 \geq \cdots \geq \alpha_n$ — *собственные значения матрицы* A. Отделенностью *собственного значения* α_i *называется число* $\mathrm{gap}(i,A) = \min_{j\neq i}|\alpha_j - \alpha_i|$. *Если матрица* A *известна из контекста, будем писать просто* $\mathrm{gap}(i)$.

Наш основной результат состоит в том, что чувствительность собственного вектора зависит от отделенности соответствующего собственного значения: если отделенность мала, то собственный вектор чувствителен.

Пример 5.4. Пусть $A = \begin{bmatrix} 1+g & \\ & 1 \end{bmatrix}$ и $A+E = \begin{bmatrix} 1+g & \epsilon \\ \epsilon & 1 \end{bmatrix}$, где $0 < \epsilon < g$. Тогда $\mathrm{gap}(i,A) = g \approx \mathrm{gap}(i, A+E)$ для $i = 1,2$. Собственные векторы матрицы A — это $q_1 = e_1$ и $q_2 = e_2$. Несложное вычисление показывает, что собственные векторы матрицы $A+E$ имеют вид

$$\hat{q}_1 = \beta \cdot \begin{bmatrix} 1+\sqrt{1+\left(\frac{2\epsilon}{g}\right)^2} \\ \frac{2\epsilon}{g} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \frac{\epsilon}{g} \end{bmatrix},$$

$$\hat{q}_2 = \beta \cdot \begin{bmatrix} -\frac{2\epsilon}{g} \\ 1+\sqrt{1+\left(\frac{2\epsilon}{g}\right)^2} \end{bmatrix} \approx \begin{bmatrix} -\frac{\epsilon}{g} \\ 1 \end{bmatrix},$$

где $\beta \approx 1/2$ — нормирующий множитель. Мы видим, что, в первом приближении по ϵ, угол между исходными векторами q_i и возмущенными векторами $\tilde{q}_i$ равен ϵ/g. Итак, угол обратно пропорционален отделенности g. ◇

Общая ситуация по существу та же, что и в только что рассмотренном двумерном случае.

Теорема 5.4. *Пусть* $A = Q\Lambda Q^T = Q\,diag(\alpha_i)Q^T$ *есть спектральное разложение матрицы* A, *а* $A+E = \widehat{A} = \widehat{Q}\widehat{\Lambda}\widehat{Q}^T$ — *спектральное разложение возмущенной матрицы. Положим* $Q = [q_1, \ldots, q_n]$ *и* $\widehat{Q} = [\hat{q}_1, \ldots, \hat{q}_n]$, *где* q_i *и* $\hat{q}_i$ *суть нормированные исходные и возмущенные собственные векторы. Пусть* θ — *острый угол между векторами* q_i *и* $\hat{q}_i$. *Тогда*

$$\frac{1}{2}\sin 2\theta \leq \frac{\|E\|_2}{\mathrm{gap}(i,A)} \quad \textit{при условии, что } \mathrm{gap}(i,A) > 0.$$

Аналогично,

$$\frac{1}{2}\sin 2\theta \leq \frac{\|E\|_2}{\mathrm{gap}(i,A+E)}, \quad \textit{если } \mathrm{gap}(i,A+E) > 0.$$

Заметим, что при $\theta \ll 1$ *имеем* $(1/2)\sin 2\theta \approx \sin\theta \approx \theta$.

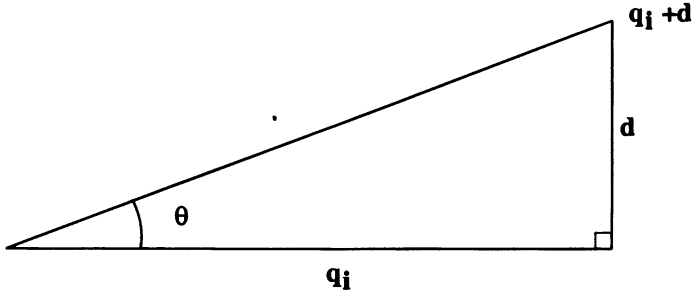
Мы формулируем оценку не только с помощью $\mathrm{gap}(i,A)$, но и $\mathrm{gap}(i,A+E)$ по следующей причине: часто бывают известны лишь собственные значения

матрицы $A + E$. В типичной ситуации, это результаты, найденные использовавшимся спектральным алгоритмом. В подобных случаях, число $\mathrm{gap}(i, A+E)$ определяется тривиально, тогда как $\mathrm{gap}(i, A)$ можно лишь оценить.

Если первая верхняя граница превзойдет $1/2$, т. е. $\|E\|_2 \geq \mathrm{gap}(i, A)/2$, то оценка приобретет вид $\sin 2\theta \leq 1$, т. е. перестанет давать какую-либо информацию об угле θ. Поясним, почему в этой ситуации для θ нельзя найти сколько-нибудь удовлетворительной оценки. Если возмущение E так велико, то собственное значение $\hat{\alpha}_i$ матрицы $A + E$ может, достаточно отдалившись от α_i, превратиться в кратное. В качестве примера, рассмотрим $A = \mathrm{diag}(2, 0)$ и $A + E = I$. Неопределенность в задании собственных векторов такой матрицы $A+E$ слишком велика; действительно, всякий вектор является собственным для матрицы $A + E = I$. Поэтому нет никакого смысла в попытке оценивания угла θ. Те же соображения применимы в том случае, если вторая верхняя граница больше $1/2$.

Доказательство. Достаточно доказать первую верхнюю оценку. Вторая будет следовать из нее, если рассматривать $A + E$ как исходную матрицу, а $A = (A + E) - E$ как возмущенную.

Пусть $q_i + d$ — собственный вектор матрицы $A+E$. Чтобы устранить неопределенность в векторе d, потребуем, чтобы он был ортогонален к q_i (т. е. $d \perp q_i$); ниже мы используем это условие ортогональности. Заметим, что, как следствие, вектор $q_i + d$ не является нормированным, поэтому $\hat{q}_i = (q_i + d)/\|q_i + d\|_2$. Далее, $\mathrm{tg}\,\theta = \|d\|_2$ и $\sec\theta = \|q_i + d\|_2$.



Приравняем i-е столбцы в матричном соотношении $(A + E)\widehat{Q} = \widehat{Q}\widehat{\Lambda}$:

$$(A + E)(q_i + d) = \hat{\alpha}_i(q_i + d). \tag{5.4}$$

Обе части были умножены на число $\|q_i + d\|_2$. Положим $\eta = \hat{\alpha}_i - \alpha_i$. Вычитая из (5.4) равенство $Aq_i = \alpha_i q_i$ и перегруппировывая члены, имеем

$$(A - \alpha_i I)d = (\eta I - E)(q_i + d). \tag{5.5}$$

Поскольку $q_i^T(A - \alpha_i I) = 0$, обе части равенства (5.5) ортогональны к вектору q_i. Поэтому можно написать $z \equiv (\eta I - E)(q_i + d) = \sum_{j\neq i} \zeta_j q_j$ и $d \equiv \sum_{j\neq i} \delta_j q_j$. Используя соотношение $(A - \alpha_i I)q_j = (\alpha_j - \alpha_i)q_j$, получаем

$$(A - \alpha_i I)d = \sum_{j\neq i}(\alpha_j - \alpha_i)\delta_j q_j = \sum_{j\neq i} \zeta_j q_j = (\eta I - E)(q_i + d),$$

или

$$d = \sum_{j \neq i} \delta_j q_j = \sum_{j \neq i} \frac{\zeta_j}{\alpha_j - \alpha_i} q_j.$$

Отсюда

$$\begin{aligned}
\operatorname{tg}\theta &= \|d\|_2 \\
&= \left\| \sum_{j \neq i} \frac{\zeta_j}{\alpha_j - \alpha_i} q_j \right\|_2 \\
&= \left(\sum_{j \neq i} \left(\frac{\zeta_j}{\alpha_j - \alpha_i} \right)^2 \right)^{1/2} && \text{вследствие ортонормальности векторов } q_j \\
&\leq \frac{1}{\operatorname{gap}(i, A)} \left(\sum_{j \neq i} \zeta_j^2 \right)^{1/2} && \text{поскольку наименьший знаменатель равен числу gap(i,A)} \\
&= \frac{\|z\|_2}{\operatorname{gap}(i, A)}.
\end{aligned}$$

Применив теорему Вейля и неравенство треугольника, мы могли бы выписать оценку $\|z\|_2 \leq (\|E\|_2 + |\eta|) \cdot \|q_i + d\|_2 \leq 2\|E\|_2 \sec\theta$, откуда следовало бы, что $\sin\theta \leq 2\|E\|_2/\operatorname{gap}(i, A)$.

Однако можно получить несколько лучший результат, более аккуратно оценивая величину $\|z\|_2 = \|(\eta I - E)(q_i + d)\|_2$. Умножая (5.4) слева на q_i^T, производя сокращения и перегруппировывая, находим $\eta = q_i^T E(q_i + d)$. Отсюда

$$\begin{aligned}
z &= (q_i + d)\eta - E(q_i + d) = (q_i + d)q_i^T E(q_i + d) - E(q_i + d) \\
&= ((q_i + d)q_i^T - I)E(q_i + d),
\end{aligned}$$

а потому $\|z\|_2 \leq \|(q_i + d)q_i^T - I\|_2 \cdot \|E\|_2 \cdot \|q_i + d\|_2$. Мы утверждаем, что $\|(q_i + d)q_i^T - I\|_2 = \|q_i + d\|_2$ (см. вопрос 5.7). Таким образом, $\|z\|_2 \leq \|q_i + d\|_2^2 \cdot \|E\|_2$, поэтому

$$\operatorname{tg}\theta \leq \frac{\|z\|_2}{\operatorname{gap}(i, A)} \leq \frac{\|q_i + d\|_2^2 \|E\|_2}{\operatorname{gap}(i, A)} = \frac{\sec^2\theta \cdot \|E\|_2}{\operatorname{gap}(i, A)},$$

или

$$\frac{\|E\|_2}{\operatorname{gap}(i, A)} \geq \frac{\operatorname{tg}\theta}{\sec^2\theta} = \sin\theta\cos\theta = \frac{1}{2}\sin 2\theta,$$

что и требовалось доказать. □

Аналогичный результат можно доказать для сингулярных векторов (см. вопрос 5.8).

У отношения Рэлея есть и другие хорошие свойства. В нашей следующей теореме показано, что в некотором естественном смысле отношение Рэлея является «наилучшим приближением» собственного значения. На этом свойстве основана RQ-итерация из разд. 5.3.2 и итерационные алгоритмы гл. 7. Кроме

того, с его помощью можно оценить качество приближенной собственной пары, найденной каким угодно способом (а не только посредством обсуждаемых здесь алгоритмов).

Теорема 5.5. *Пусть A — симметричная матрица, x — нормированный вектор, а β — число. Тогда найдется собственная пара (α_i, q_i) матрицы A (т. е. $Aq_i = \alpha_i q_i$), удовлетворяющая оценке $|\alpha_i - \beta| \le \|Ax - \beta x\|_2$. При заданном векторе x величина $\|Ax - \beta x\|_2$ минимизируется выбором $\beta = \rho(x, A)$.*

При наличии несколько большей информации о спектре матрицы A можно получить более точные оценки. Положим $r = Ax - \rho(x, A)x$. Пусть α_i — собственное значение матрицы A, ближайшее к $\rho(x, A)$. Введем величину $\mathrm{gap}' \equiv \min_{j \ne i} |\alpha_j - \rho(x, A)|$, являющуюся вариантом введенного выше понятия отделенности. Пусть θ — острый угол между векторами x и q_i. Тогда

$$\sin\theta \le \frac{\|r\|_2}{\mathrm{gap}'}, \tag{5.6}$$

$$|\alpha_j - \rho(x, A)| \le \frac{\|r\|_2^2}{\mathrm{gap}'}. \tag{5.7}$$

В теореме 7.1 дано обобщение этого результата на случай группы собственных значений.

Обратим внимание на то, что в оценке (5.7) разность между частным Рэлея $\rho(x, A)$ и собственным значением α_i пропорциональна *квадрату* нормы невязки $\|r\|_2$. Именно на этой высокой точности основана кубическая сходимость RQ-итерации из разд. 5.3.2.

Доказательство. Мы докажем только первое утверждение и вынесем остальные в вопросы 5.9 и 5.10 в конце главы.

Если β является собственным значением матрицы A, то утверждение очевидно. Поэтому предположим, что матрица $A - \beta I$ невырожденна. Тогда $x = (A - \beta I)^{-1}(A - \beta I)x$ и

$$1 = \|x\|_2 \le \|(A - \beta I)^{-1}\|_2 \cdot \|(A - \beta I)x\|_2.$$

Пусть $A = Q\Lambda Q^T = Q\mathrm{diag}(\alpha_1, \ldots, \alpha_n)Q^T$ — спектральное разложение матрицы A. Имеем

$$\|(A - \beta I)^{-1}\|_2 = \|Q(\Lambda - \beta I)^{-1}Q^T\|_2 = \|(\Lambda - \beta I)^{-1}\|_2 = 1/\min_i |\alpha_i - \beta|.$$

Таким образом, $\min_i |\alpha_i - \beta| \le \|(A - \beta I)x\|_2$, что и требовалось.

Чтобы показать, что $\beta = \rho(x, A)$ минимизирует $\|Ax - \beta x\|_2$, докажем, что x ортогонален вектору $Ax - \rho(x, A)x$. После этого применение теоремы Пифагора к сумме ортогональных векторов

$$Ax - \beta x = [Ax - \rho(x, A)x] + [(\rho(x, A) - \beta)x]$$

даст

$$\begin{aligned} \|Ax - \beta x\|_2^2 &= \|Ax - \rho(x, A)x\|_2^2 + \|(\rho(x, A) - \beta)x\|_2^2 \\ &\ge \|Ax - \rho(x, A)x\|_2^2, \end{aligned}$$

причем равенство возможно лишь в случае $\beta = \rho(x, A)$.

Проверка ортогональности векторов x и $Ax - \rho(x,A)x$ проводится следующей выкладкой:

$$x^T(Ax - \rho(x,A)x) = x^T\left(Ax - \frac{(x^TAx)}{x^Tx}x\right) = x^TAx - x^TAx\frac{x^Tx}{x^Tx} = 0.$$

Этим теорема доказана. □

Пример 5.5. Проиллюстрируем теорему 5.5 с помощью матрицы из примера 5.4. Пусть $A = \begin{bmatrix} 1+g & \epsilon \\ \epsilon & 1 \end{bmatrix}$, где $0 < \epsilon < g$. Пусть $x = [1,0]^T$ и $\beta = \rho(x,A) = 1+g$. Тогда $r = Ax - \beta x = [0,\epsilon]^T$ и $\|r\|_2 = \epsilon$. Собственными значениями матрицы A являются числа $\alpha_\pm = 1 + \frac{g}{2} \pm \frac{g}{2}(1 + (\frac{2\epsilon}{g})^2)^{1/2}$, а собственные векторы указаны в примере 5.4 (где A называется матрицей $A+E$).

Согласно теореме 5.5, число $\|Ax - \beta x\|_2 = \|r\|_2 = \epsilon$ есть верхняя граница для расстояния от $\beta = 1+g$ до ближайшего собственного значения α_+ матрицы A. Это же предсказывается теоремой Вейля (теорема 5.1). Ниже мы увидим, что эта граница гораздо слабее границы (5.7).

Если ϵ много меньше, чем g, то одно из собственных значений матрицы A близко́ к $1+g$, а соответствующий собственный вектор близок к x. Другое собственное значение находится вблизи 1, а отвечающий ему собственный вектор близок к $[0,1]^T$. Это означает, что $\mathrm{gap}' = |\alpha_- - \rho(x,A)| = \frac{g}{2}(1 + (1 + (\frac{2\epsilon}{g})^2)^{1/2})$, поэтому из (5.6) вытекает такая оценка для угла θ между x и точным собственным вектором:

$$\sin\theta \le \frac{\|r\|_2}{\mathrm{gap}'} = \frac{2\epsilon/g}{1 + (1 + (\frac{2\epsilon}{g})^2)^{1/2}}.$$

Из явных формул для собственных векторов, приведенных в примере 5.4, видно, что полученная верхняя граница в действительности равна $\mathrm{tg}\,\theta$, что для малых θ почти то же самое, что $\sin\theta$. Таким образом, оценка (5.6) весьма точна.

Перейдем теперь к оценке (5.7) для разности $|\beta - \alpha_+|$. Оказывается, что для рассматриваемого 2×2-примера и величина $|\beta - \alpha_+|$, и граница для нее *точно* равны числу

$$\frac{\|r\|_2^2}{\mathrm{gap}'} = \epsilon \cdot \frac{2\epsilon/g}{1 + (1 + (\frac{2\epsilon}{g})^2)^{1/2}}.$$

Вычислим значения указанных границ в специальном случае $g = 10^{-2}$ и $\epsilon = 10^{-5}$. Тогда собственные значения матрицы A будут приближенно равны числам $\alpha_+ = 1.01000001 = 1.01 + 10^{-8}$ и $\alpha_- = .99999999 = 1 - 10^{-8}$. Первая оценка принимает вид $|\beta - \alpha_+| \le \|r\|_2 = 10^{-5}$; эта граница в 10^3 раз превышает действительную ошибку 10^{-8}. Напротив, оценка (5.7) дает $|\beta - \alpha_+| \le \|r\|_2^2/\mathrm{gap}' = (10^{-5})^2/(1.01 - \alpha_-) \approx 10^{-8}$, т. е. это очень точная оценка. Угол θ между x и точным собственным вектором для собственного значения α_+ составляет примерно 10^{-3} и почти то же значение дает граница $\|r\|_2/\mathrm{gap}' = 10^{-5}/(1.01 - \alpha_-) \approx 10^{-3}$. ◇

В заключение обсудим ситуацию, когда имеется кластер из k собственных значений и нужно вычислить соответствующие собственные векторы. Под

«кластером» понимается группа близких собственных значений, находящихся на значительном удалении от собственных значений, не входящих в эту группу. Примером могли бы быть $k = 20$ собственных значений на отрезке $[0.9999, 1.0001]$ при том, что все остальные собственные значения больше, чем 2. Теоремы 5.4 и 5.5 заставляют думать, что отдельные собственные векторы не удастся определить с хорошей точностью. Однако это возможно для k-мерного инвариантного подпространства, натянутого на эти векторы. Детали можно найти в [197].

5.2.1. Теория относительных возмущений

Теперь мы выведем для собственных значений и собственных векторов более точные оценки, чем в предыдущем разделе. Они понадобятся нам при обосновании высокоточных алгоритмов для вычисления собственных значений и сингулярных чисел, описанных в разд. 5.4.2 и 5.4.3.

Сопоставим эти новые оценки с оценками предыдущего раздела для одномерного случая. Пусть даны число α, возмущенное число $\hat{\alpha} = \alpha + e$ и граница $|e| \leq \epsilon$. Тогда можно указать очевидную оценку $|\hat{\alpha} - \alpha| \leq \epsilon$ для *абсолютной погрешности* приближения $\hat{\alpha}$. Именно такой подход был принят в предыдущем разделе. Рассмотрим теперь возмущенное число вида $\hat{\alpha} = x^2\alpha$ и границу $|x^2 - 1| \leq \epsilon$. Здесь можно оценить *относительную погрешность* числа $\hat{\alpha}$:

$$\frac{|\hat{\alpha} - \alpha|}{|\alpha|} = |x^2 - 1| \leq \epsilon.$$

Перенесем это простое сопоставление на случай матриц. В предыдущем разделе мы оценивали *абсолютные* расстояния между собственными значениями α_i матрицы A и собственными значениями $\hat{\alpha}_i$ матрицы $\widehat{A} = A + E$ посредством неравенств вида $|\hat{\alpha}_i - \alpha_i| \leq \|E\|_2$. Теперь же мы станем оценивать *относительные* разности чисел α_i и собственных значений $\hat{\alpha}_i$ матрицы $\widehat{A} = X^T A X$ посредством величины $\epsilon \equiv \|X^T X - I\|_2$.

Теорема 5.6 («относительная» теорема Вейля). *Пусть матрица A имеет собственные значения α_i, а матрица $\widehat{A} = X^T A X$ — собственные значения $\hat{\alpha}_i$. Положим $\epsilon \equiv \|X^T X - I\|_2$. Тогда $|\hat{\alpha}_i - \alpha_i| \leq |\alpha_i|\epsilon$. Если $\alpha_i \neq 0$, то эти оценки можно записать в виде*

$$\frac{|\hat{\alpha}_i - \alpha_i|}{|\alpha_i|} \leq \epsilon. \tag{5.8}$$

Доказательство. Поскольку i-е собственное значение матрицы $A - \alpha_i I$ равно нулю, то же самое, в силу теоремы Сильвестра об инерции, верно для матрицы

$$X^T(A - \alpha_i I)X = (X^T A X - \alpha_i I) + \alpha_i(I - X^T X) \equiv H + F.$$

Согласно теореме Вейля, $|\lambda_i(H) - 0| \leq \|F\|_2$, или $|\hat{\alpha}_i - \alpha_i| \leq |\alpha_i| \cdot \|X^T X - I\|_2 = |\alpha_i|\epsilon$. □

Заметим, что если X — ортогональная матрица, то $\epsilon = \|X^T X - I\|_2 = 0$, и теорема подтверждает, что матрицы $X^T A X$ и A имеют одни и те же собственные значения. Если матрица X «почти» ортогональна, т. е. если число ϵ

мало, то, согласно теореме, собственные значения обеих матриц мало разнятся в смысле относительной погрешности.

Следствие 5.2. *Пусть матрица G имеет сингулярные числа σ_i, а матрица $\widehat{G} = Y^TGX$ — сингулярные числа $\hat{\sigma}_i$. Положим $\epsilon \equiv \max(\|X^TX - I\|_2, \|Y^TY - I\|_2)$. Тогда $|\hat{\sigma}_i - \sigma_i| \le \epsilon\sigma_i$. Если $\sigma_i \neq 0$, то эти оценки можно записать в виде*

$$\frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i} \le \epsilon. \tag{5.9}$$

Чтобы оценить различия между собственными векторами q_i матрицы A и собственными векторами $\hat{q}_i$ матрицы $\widehat{A} = X^TAX$, можно аналогичным образом обобщить теорему 5.4. Для этого нам понадобится определение *относительной отделенности* собственного значения.

Определение 5.4. Относительной отделенностью *собственного значения α_i матрицы A называется число* $\mathrm{rel_gap}(i, A) = \min_{j\neq i} \frac{|\alpha_j - \alpha_i|}{|\alpha_i|}$.

Теорема 5.7. *Пусть матрица A имеет собственные значения α_i и соответствующие нормированные собственные векторы q_i, а матрица $\widehat{A} = X^TAX$ — собственные значения $\hat{\alpha}_i$ и нормированные собственные векторы $\hat{q}_i$. Обозначим через θ острый угол между векторами q_i и $\hat{q}_i$. Положим $\epsilon_1 = \|I - X^{-T}X^{-1}\|_2$ и $\epsilon_2 = \|X - I\|_2$. Предположим, что $\epsilon_1 < 1$ и $\mathrm{rel_gap}(i, X^TAX) > 0$. Тогда*

$$\frac{1}{2}\sin 2\theta \le \frac{\epsilon_1}{1-\epsilon_1} \cdot \frac{1}{\mathrm{rel_gap}(i, X^TAX)} + \epsilon_2.$$

Доказательство. Пусть $\eta = \hat{\alpha}_i - \alpha_i$, $H = A - \hat{\alpha}_i I$ и $F = \hat{\alpha}_i(I - X^{-T}X^{-1})$. Заметим, что

$$H + F = A - \hat{\alpha}_i X^{-T}X^{-1} = X^{-T}(X^TAX - \hat{\alpha}_i I)X^{-1}.$$

Таким образом, $Hq_i = -\eta q_i$ и $(H + F)(X\hat{q}_i) = 0$, т. е. $X\hat{q}_i$ есть собственный вектор матрицы $H + F$, относящийся к ее i-му собственному значению 0. Пусть θ_1 — острый угол между векторами q_i и $X\hat{q}_i$. Согласно теореме 5.4, справедлива оценка

$$\frac{1}{2}\sin 2\theta_1 \le \frac{\|F\|_2}{\mathrm{gap}(i, H + F)}. \tag{5.10}$$

При этом $\|F\|_2 = |\hat{\alpha}_i|\epsilon_1$. Отметим, что величина $\mathrm{gap}(i, H + F)$ равна наименьшему из модулей ненулевых собственных значений матрицы $H + F$. Поскольку собственными значениями матрицы $X^T(H+F)X = X^TAX - \hat{\alpha}_i I$ являются числа $\hat{\alpha}_j - \hat{\alpha}_i$, из теоремы 5.6 вытекает, что собственные значения матрицы $H + F$ заключены в интервалах с концами в точках $(1-\epsilon_1)(\hat{\alpha}_j - \hat{\alpha}_i)$ и $(1+\epsilon_1)(\hat{\alpha}_j - \hat{\alpha}_i)$. Поэтому $\mathrm{gap}(i, H + F) \ge (1 - \epsilon_1)\mathrm{gap}(i, X^TAX)$. Используя это неравенство для оценки (5.10), получаем

$$\frac{1}{2}\sin 2\theta_1 \le \frac{\epsilon_1|\hat{\alpha}_i|}{(1-\epsilon_1)\mathrm{gap}(i, X^TAX)} = \frac{\epsilon_1}{(1-\epsilon_1)\mathrm{rel_gap}(i, X^TAX)}. \tag{5.11}$$

Пусть θ_2 — острый угол между векторами $X\hat{q}_i$ и $\hat{q}_i$, так что $\theta \leq \theta_1 + \theta_2$. Из тригонометрических соображений, $\sin\theta_2 \leq \|(X-I)\hat{q}_i\|_2 \leq \|X-I\|_2 = \epsilon_2$. Теперь, применяя неравенство треугольника (см. вопрос 5.11), имеем

$$\begin{aligned}\frac{1}{2}\sin 2\theta &\leq \frac{1}{2}\sin 2\theta_1 + \frac{1}{2}\sin 2\theta_2 \\ &\leq \frac{1}{2}\sin 2\theta_1 + \sin\theta_2 \\ &\leq \frac{\epsilon_1}{(1-\epsilon_1)\text{rel_gap}(i, X^TAX)} + \epsilon_2,\end{aligned}$$

что и требовалось. □

Аналогичное утверждение можно доказать для сингулярных векторов [101].

Пример 5.6. Используем связанную систему материальных точек из примера 5.1, чтобы показать, что оценки для собственных значений, предоставляемые теоремой Вейля (т. е. теоремой 5.1), могут быть много хуже (менее точны) оценок из «относительного» варианта этой теоремы (т. е. теоремы 5.6). Мы также увидим, что оценка для собственного вектора из теоремы 5.7 может быть гораздо лучше (точнее) оценки из теоремы 5.4.

Предположим, что $M = \text{diag}(1, 100, 10000)$, а $K_D = \text{diag}(10000, 100, 1)$. Вслед за примером 5.1, определим матрицы $K = BK_DB^T$ и $\widehat{K} = M^{-1/2}KM^{-1/2}$, где

$$B = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix},$$

а потому

$$\widehat{K} = M^{-1/2}KM^{-1/2} = \begin{bmatrix} 10100 & -10 & \\ -10 & 1.01 & -.001 \\ & -.001 & .0001 \end{bmatrix}.$$

С точностью до пяти десятичных разрядов, собственными значениями матрицы $\widehat{K}$ являются числа 10100, 1.0001 и 0.00099. Предположим теперь, что каждая из масс (m_{ii}) и каждый из коэффициентов жесткости $(k_{D,ii})$ возмущены не более чем на 1%. Насколько при этом могут измениться собственные значения? Если заменить m_{11} на .99, а $k_{D,11}$ на 10100, то наибольший элемент матрицы, а именно $\widehat{K}_{11}$, примет значение 10305, т. е. его изменение составит 205. Но тогда, согласно теореме Вейля, каждое собственное значение может измениться на ± 205, что совершенно исказит два младших собственных значения. Оценка для собственного вектора из теоремы 5.4 также показывает, что соответствующие собственные векторы могут полностью измениться.

Применим теперь теорему 5.6 к матрице $\widehat{K}$ или, фактически, следствие 5.2 к матрице $G = M^{-1/2}BK_D^{1/2}$; в соответствии с примером 5.1, $\widehat{K} = GG^T$. Изменение каждой массы не более чем на 1% равносильно замене матрицы G на XG, где X — диагональная матрица с диагональными элементами, заключенными между $1/\sqrt{.99} \approx 1.005$ и $1/\sqrt{1.01} \approx 0.995$. Согласно следствию 5.2, сингулярные

числа матрицы G могут, самое большое, приобрести множители, находящиеся в интервале $[0.995, 1.005]$, поэтому изменения в собственных значениях матрицы M составят не более 1%. Другими словами, подобно наибольшему собственному значению, младшее собственное значение может измениться разве что в своем втором десятичном разряде. Точно так же, изменение коэффициентов жесткости, не превышающее 1%, равносильно замене G на XG, и снова собственные значения не могут измениться более чем на 1%. Одновременное возмущение матриц M и K_D может повлечь за собой возмущения собственных значений примерно на 2%. Поскольку собственные значения сильно различаются по величине, все относительные отделенности довольно велики, а потому и возмущения в собственных векторах, измеряемые углами, не превысят приблизительно 0.03. ◇

Описание другого подхода к анализу относительных ошибок, более приспособленного для матриц, порождаемых дифференциальными («неограниченными») операторами, дано в [161].

5.3. Алгоритмы для симметричной проблемы собственных значений

Мы рассмотрим ряд алгоритмов для симметричной проблемы собственных значений, при этом, как было указано во введении, только *прямые методы*. Рассмотрение *итерационных методов* откладывается до гл. 7.

В главе 4, посвященной несимметричной проблеме собственных значений, единственным обсуждавшимся алгоритмом была QR-итерация. Она предназначена для вычисления всех собственных значений и, при необходимости, собственных векторов. Для симметричной проблемы собственных значений имеется гораздо больше хороших алгоритмов, что повышает гибкость и эффективность вычислений. Например, описываемый ниже *алгоритм бисекции* способен находить только собственные значения, принадлежащие отрезку $[a, b]$, указанному пользователем. Это требует куда меньше времени, чем вычисление всех собственных значений.

Все рассматриваемые алгоритмы, за исключением RQ-итерации и метода Якоби, предполагают, что матрица была вначале приведена к трехдиагональной форме посредством модификации алгоритма 4.6, описанной в разд. 4.4.7. Стоимость этого начального этапа составляет $\frac{4}{3}n^3$ флопов или $\frac{8}{3}n^3$ флопов, если нужны и собственные векторы.

1. *Трехдиагональная* QR*-итерация*. Этот алгоритм находит все собственные значения и, если необходимо, собственные векторы симметричной трехдиагональной матрицы. Среди практических методов вычисления всех собственных значений симметричной трехдиагональной матрицы эффективно реализованная QR-итерация является наибыстрейшей. Она обходится в $O(n^2)$ флопов, что при достаточно больших n составляет пренебрежимо малую часть от $\frac{4}{3}n^3$ флопов, необходимых для приведения исходной плотной матрицы к трехдиагональной форме. Однако если нужны и все собственные векторы, то QR-итерация в среднем требует несколько больше, чем $6n^3$ флопов, и остается самым быстрым алгоритмом лишь для малых

матриц порядка, не превосходящего $n = 25$. Именно QR-итерация стоит за Matlab-командой `eig`[1] и LAPACK-программами `ssyev` (для плотных матриц) и `sstev` (для трехдиагональных матриц).

2. *RQ-итерация.* Этот алгоритм лежит в основе QR-итерации, но мы рассматриваем его отдельно от нее, чтобы упростить анализ его чрезвычайно быстрой сходимости, а также потому, что RQ-итерация может применяться и как самостоятельный алгоритм. В общем случае, алгоритм сходится кубически (как и QR-итерация), что означает: число верных знаков в приближенных результатах асимптотически *утраивается* на каждом шаге.
3. *«Разделяй-и-властвуй».* В настоящее время это самый быстрый метод вычисления всех собственных значений и собственных векторов симметричной трехдиагональной матрицы порядка n, большего 25. (LAPACK-реализация метода, называемая `sstevd`, переключается на QR-итерацию, если $n \leq 25$.) В наихудшем случае алгоритм «разделяй-и-властвуй» требует $O(n^3)$ флопов, но константа, упрятанная в этом символе, на практике оказывается весьма малой. На большой выборке случайных тестовых матриц в среднем затрачивается лишь $O(n^{2.3})$ флопов, а для некоторых специальных распределений собственных значений даже $O(n^2)$ флопов.
   Теоретически алгоритм может быть реализован за $O(n \cdot \log^p n)$ флопов, где p — небольшое целое число [131]. Эта сверхбыстрая реализация использует быстрый многополюсный метод (FMM) [124], первоначально придуманный для совершенно другой задачи (а именно вычисления электрических сил взаимодействия в ансамбле из n заряженных частиц). Вследствие алгоритмической сложности этого сверхбыстрого метода, в настоящее время следует выбирать QR-итерацию, если вычисляются все собственные значения, и алгоритм «разделяй-и-властвуй» (без использования FMM), если вычисляются все собственные значения и собственные векторы.
4. *Бисекция и обратная итерация.* Бисекцию можно использовать для вычисления только некоторого подмножества собственных значений симметричной трехдиагональной матрицы, скажем, тех, что расположены на отрезке $[a, b]$ или $[\alpha_i, \alpha_{i-j}]$. Она обходится лишь в $O(nk)$ флопов, где k — число требуемых собственных значений. Поскольку для QR-итерации нужны $O(n^2)$ флопов, бисекция может быть гораздо быстрее при $k \ll n$. Соответствующие собственные векторы могут быть найдены с помощью обратной итерации (алгоритм 4.2). В наилучшем случае, когда собственные значения «хорошо разделены» (смысл этого термина мы подробно объясним позже), обратная итерация также стоит лишь $O(nk)$ флопов. Это значительно меньше, чем стоимость QR-итерации или алгоритма «разделяй-и-властвуй» (без FMM), даже при необходимости вычислять все собственные значения и собственные векторы (т. е. при $k = n$). Однако в наихудшем случае, когда имеются обширные кластеры собственных значений, трудоемкость обратной итерации возрастает до $O(nk^2)$ флопов; при этом не гарантируется качество вычисленных собственных векторов (хотя на практике они почти всегда вычисляются с хорошей точностью). Поэтому при необходимости вычисления всех (или большей части) собственных значений и собственных векторов

[1] Matlab проверяет, является ли аргумент команды `eig` симметричной матрицей и, если да, использует симметричный алгоритм.

следует выбирать алгоритм «разделяй-и-властвуй» или QR-алгоритм, особенно при наличии кластеров собственных значений. Бисекция и обратная итерация реализованы как опции в LAPACK-программе `ssyevx`.
В настоящее время ведутся активные исследования по проблеме близких собственных значений в обратной итерации. Они могут превратить ее в самый быстрый метод вычисления всех собственных векторов (если опустить теоретическое сравнение с алгоритмом «разделяй-и-властвуй» на основе FMM) [105, 203, 83, 201, 176, 173, 175, 269]. Однако программ, реализующих этот улучшенный вариант обратной итерации, пока нет.

5. *Метод Якоби.* Этот метод, восходящий к 1846 г., исторически является старейшим для проблемы собственных значений. Обычно он много медленнее любого из перечисленных выше методов, имея трудоемкость $O(n^3)$ флопов с большой константой внутри символа O. В то же время интерес к методу сохраняется, потому что подчас он дает гораздо более точные результаты, чем другие методы. Причина состоит в том, что метод способен иногда достигать уровня относительной точности, описываемого в разд. 5.2.1; при этом малые собственные значения вычисляются много точнее, чем в конкурирующих алгоритмах [82]. Мы обсудим это замечательное свойство метода Якоби в разд. 5.4.3, где показано, как следует вычислять сингулярное разложение.

В последующих разделах мы более подробно рассмотрим названные алгоритмы. В разд. 5.3.6 представлены результаты экспериментального сравнения их производительности.

5.3.1. Трехдиагональная QR-итерация

Вспомним, что QR-алгоритм для несимметричных матриц состоит из следующих двух этапов:

1. Применить к заданной матрице A алгоритм 4.6, вычисляющий ортогональную матрицу Q, такую, что $QAQ^T = H$ имеет верхнюю форму Хессенберга.
2. Применить QR-итерацию к матрице H (как это описано в разд. 4.4.8). В результате будет получена последовательность верхних хессенберговых матриц $H = H_0, H_1, H_2, \ldots$, сходящаяся к вещественной форме Шура.

Наш первый алгоритм для симметричной проблемы собственных значений имеет точно такую же структуру:

1. Пользуясь модификацией алгоритма 4.6, описанной в разд. 4.4.7, вычислить для заданной матрицы $A = A^T$ ортогональную матрицу Q, такую, что $QAQ^T = T$ имеет трехдиагональную форму.
2. Применить QR-итерацию к матрице T. В результате будет получена последовательность трехдиагональных матриц $T = T_0, T_1, T_2, \ldots$, сходящаяся к диагональному виду.

То обстоятельство, что матрицы T_i в QR-итерации остаются трехдиагональными, можно обосновать так: поскольку матрица QAQ^T одновременно симметричная и верхняя хессенбергова, она должна быть и нижней хессенберговой, т. е. трехдиагональной. Именно это обстоятельство делает каждый шаг QR-итерации очень дешевым. Приведем подсчет числа операций:

1. Приведение A к симметричной трехдиагональной матрице T стоит $\frac{4}{3}n^3 + O(n^2)$ флопов или $\frac{8}{3}n^3 + O(n^2)$ флопов, если нужны и собственные векторы.
2. Один шаг трехдиагональной QR-итерации с одинарным сдвигом («вытеснение выступа») стоит $6n$ флопов.
3. При вычислении всех собственных значений матрицы T требуется в среднем только 2 QR-шага на одно собственное значение, что в совокупности дает $6n^2$ флопов.
4. Вычисление всех собственных значений и собственных векторов матрицы T стоит $6n^3 + O(n^2)$ флопов.
5. Общая стоимость вычисления всех собственных значений матрицы A (без вычисления собственных векторов) составляет $\frac{4}{3}n^3 + O(n^2)$ флопов.
6. Общая стоимость вычисления всех собственных значений и собственных векторов матрицы A составляет $8\frac{2}{3}n^3 + O(n^2)$ флопов.

Нужно еще описать, как выбираются сдвиги при реализации QR-алгоритма. Пусть на i-м шаге имеем матрицу

$$T_i = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}.$$

Простейшим выбором было бы правило $\sigma_i = a_n$. Это QR-итерация с одинарным сдвигом, обсуждавшаяся в разд. 4.4.8. Оказывается, что почти для всех матриц она сходится кубически (это показано в следующем разделе). К сожалению, существуют примеры несходимости этого метода [197, с. 76]. Поэтому, чтобы обеспечить глобальную сходимость, используется несколько более сложная стратегия: в качестве сдвига σ_i выбирается то из собственных значений подматрицы $\begin{bmatrix} a_{n-1} & b_{n-1} \\ b_{n-1} & a_n \end{bmatrix}$, которое ближе к a_n. Это правило выбора сдвигов называется *стратегией Уилкинсона*.

Теорема 5.8 (Уилкинсон). *QR-итерация со стратегией Уилкинсона сходится глобально, причем, по меньшей мере, с линейной скоростью. Для почти всех матриц метод асимптотически сходится с кубической скоростью.*

Доказательство этой теоремы можно найти в [197]. Метод реализован LAPACK-программой `ssyev`. Если нужны только собственные значения, то внутренний цикл алгоритма можно организовать более эффективно (`ssterf`; см. также [104, 200]) по сравнению со случаем, когда вычисляются и собственные векторы (`ssteqr`).

Пример 5.7. Приведем иллюстрацию сходимости трехдиагональной QR-итерации, примененной к следующей трехдиагональной матрице (ее диагонали

схематически изображены столбцами):

$$T_0 = \text{tridiag}\begin{bmatrix} & .24929 & \\ 1.263 & & 1.263 \\ & .96880 & \\ -.82812 & & -.82812 \\ & .48539 & \\ -3.1883 & & -3.1883 \\ & -.91563 & \end{bmatrix}.$$

В таблице показано поведение последнего внедиагонального элемента матриц T_i, последнего диагонального элемента и разности между последним диагональным элементом и его предельным значением (т.е. собственным значением $\alpha \approx -3.54627$). Из последнего столбца таблицы очевидна кубическая сходимость ошибки к нулю.

i	$T_i(4,3)$	$T_i(4,4)$	$T_i(4,4)-\alpha$
0	-3.1883	$-.91563$	2.6306
1	$-5.7\cdot 10^{-2}$	-3.5457	$5.4\cdot 10^{-4}$
2	$-2.5\cdot 10^{-7}$	-3.5463	$1.2\cdot 10^{-14}$
3	$-6.1\cdot 10^{-23}$	-3.5463	0

В этот момент ($i = 3$) мы имеем матрицу

$$T_3 = \text{tridiag}\begin{bmatrix} & 1.9871 & \\ .77513 & & .77513 \\ & 1.7049 & \\ -1.7207 & & -1.7207 \\ & .64214 & \\ -6.1\cdot 10^{-23} & & -6.1\cdot 10^{-23} \\ & -3.5463 & \end{bmatrix}.$$

Ее очень малые элементы $(4,3)$ и $(3,4)$ заменяются нулями. Этот прием, называемый *дефляцией* (понижением порядка), численно устойчив, так как T_3 претерпевает возмущение с нормой всего лишь $6.1\cdot 10^{-23}$. Теперь QR-итерация применяется снова, но уже только к ведущей главной 3×3-подматрице матрицы T_3. Этот процесс повторяется, пока не будут найдены все собственные значения (см. HOMEPAGE/Matlab/tridiQR.m).

5.3.2. RQ-итерация

Вспомним, что, согласно нашему анализу в разд. 4.4, на каждом шаге QR-алгоритма неявно выполняется обратная итерация. Исследуем эту связь между двумя алгоритмами более подробно в том случае, когда в качестве сдвига в обратной итерации выбирается отношение Рэлея. Такой метод будем называть RQ-итерацией (от Rayleigh Quotient Iteration).

Алгоритм 5.1. RQ-*итерация: для заданного вектора* x_0 ($\|x_0\|_2 = 1$) *и задаваемого пользователем критерия останова* tol *выполнять итерации*

$\rho_0 = \rho(x_0, A) = \frac{x_0^T A x_0}{x_0^T x_0}$
$i = 0$

repeat
$y_i = (A - \rho_{i-1} I)^{-1} x_{i-1}$
$x_i = y_i / \|y_i\|_2$
$\rho_i = \rho(x_i, A)$
$i = i + 1$
пока метод не сойдется (т. е. пока не будет выполнено условие $\|Ax_i - \rho_i x_i\|_2 < \mathrm{tol}$*)*

Если условие останова удовлетворено, то по теореме 5.5 число ρ_i удалено от некоторого собственного значения матрицы A не более чем на tol.

Пусть в QR-итерации используются сдвиги $\sigma_i = a_{nn}$, а RQ-итерация применяется к начальному вектору $x_0 = [0, \ldots, 0, 1]^T$. Обсуждавшуюся в разд. 4.4 связь между QR-алгоритмом и обратной итерацией можно использовать, чтобы показать, что последовательности чисел σ_i и ρ_i, генерируемые этими двумя алгоритмами, совпадают (см. вопрос 5.13). Мы докажем для этого случая, что сходимость почти всегда является кубической.

Теорема 5.9. RQ-*итерация локально сходится кубически, т. е. если собственное значение простое, а ошибка достаточно мала, то число верных разрядов приближенного собственного значения утраивается на каждом шаге.*

Доказательство. Мы утверждаем, что достаточно проанализировать случай диагональной матрицы A. Действительно, пусть $Q^T A Q = \Lambda$, где Q — ортогональная матрица, составленная из собственных векторов матрицы A, а Λ — диагональная матрица собственных значений. Заменим переменные в RQ-итерации по формулам $\hat{x}_i \equiv Q^T x_i$ и $\hat{y}_i \equiv Q^T y_i$. Тогда

$$\rho_i = \rho(x_i, A) = \frac{x_i^T A x_i}{x_i^T x_i} = \frac{\hat{x}_i^T Q^T A Q \hat{x}_i}{\hat{x}_i^T Q^T Q \hat{x}_i} = \frac{\hat{x}_i^T \Lambda \hat{x}_i}{\hat{x}_i^T \hat{x}_i} = \rho(\hat{x}_i, \Lambda)$$

и $Q\hat{y}_{i+1} = (A - \rho_i I)^{-1} Q \hat{x}_i$, поэтому

$$\hat{y}_{i+1} = Q^T (A - \rho_i I)^{-1} Q \hat{x}_i = (Q^T A Q - \rho_i I)^{-1} \hat{x}_i = (\Lambda - \rho_i I)^{-1} \hat{x}_i.$$

Следовательно, проведение RQ-итерации с A и x_0 равносильно ее проведению с Λ и $\hat{x}_0$. Поэтому, без ограничения общности, предположим, что $A = \Lambda$ — уже диагональная матрица; таким образом, собственными векторами являются столбцы e_i единичной матрицы.

Не ограничивая общности, будем считать, что векторы x_i сходятся к e_1. Тогда можно написать $x_i = e_1 + d_i$, где $\|d_i\|_2 \equiv \epsilon \ll 1$. Для доказательства кубической сходимости нужно показать, что $x_{i+1} = e_1 + d_{i+1}$, где $\|d_{i+1}\|_2 = O(\epsilon^3)$.

Заметим прежде всего, что

$$1 = x_i^T x_i = (e_1 + d_i)^T (e_1 + d_i) = e_1^T e_1 + 2 e_1^T d_i + d_i^T d_i = 1 + 2 d_{i1} + \epsilon^2,$$

поэтому $d_{i1} = -\epsilon^2/2$. Следовательно,

$$\rho_i = x_i^T \Lambda x_i = (e_1 + d_i)^T \Lambda (e_1 + d_i) = e_1^T \Lambda e_1 + 2 e_1^T \Lambda d_i + d_i^T \Lambda d_i = \alpha_1 - \eta,$$

где $\eta \equiv -2 e_1^T \Lambda d_i - d_i^T \Lambda d_i = \alpha_1 \epsilon^2 - d_i^T \Lambda d_i$. Отсюда

$$|\eta| \le |\alpha_1| \epsilon^2 + \|\Lambda\|_2 \|d_i\|_2^2 \le 2 \|\Lambda\|_2 \epsilon^2. \qquad (5.12)$$

Таким образом, число $\rho_i = \alpha_1 - \eta = \alpha_1 + O(\epsilon^2)$ является очень хорошим приближением к собственному значению α_1.

Теперь можно написать

$$
\begin{aligned}
y_{i+1} &= (\Lambda - \rho_i I)^{-1} x_i \\
&= \left[\frac{x_{i1}}{\alpha_1 - \rho_i}, \frac{x_{i2}}{\alpha_2 - \rho_i}, \ldots, \frac{x_{in}}{\alpha_n - \rho_i} \right]^T \\
&\qquad \text{поскольку } (\Lambda - \rho_i I)^{-1} = \operatorname{diag}\left(\frac{1}{\alpha_j - \rho_i} \right) \\
&= \left[\frac{1 + d_{i1}}{\alpha_1 - \rho_i}, \frac{d_{i2}}{\alpha_2 - \rho_i}, \ldots, \frac{d_{in}}{\alpha_n - \rho_i} \right]^T \\
&\qquad \text{поскольку } x_i = e_1 + d_i \\
&= \left[\frac{1 - \epsilon^2/2}{\eta}, \frac{d_{i2}}{\alpha_2 - \alpha_1 + \eta}, \ldots, \frac{d_{in}}{\alpha_n - \alpha_1 + \eta} \right]^T \\
&\qquad \text{поскольку } \rho_i = \alpha_1 - \eta \text{ и } d_{i1} = -\epsilon^2/2 \\
&= \frac{1 - \epsilon^2/2}{\eta} \cdot \left[1, \frac{d_{i2}\eta}{(1 - \epsilon^2/2)(\alpha_2 - \alpha_1 + \eta)}, \ldots, \frac{d_{in}\eta}{(1 - \epsilon^2/2)(\alpha_n - \alpha_1 + \eta)} \right]^T \\
&\equiv \frac{1 - \epsilon^2/2}{\eta} \cdot (e_1 + \hat{d}_{i+1}).
\end{aligned}
$$

Чтобы оценить $\|\hat{d}_{i+1}\|_2$, используем оценку для знаменателей $|\alpha_j - \alpha_1 + \eta| \geq \operatorname{gap}(1, \Lambda) - |\eta|$ и неравенство (5.12). В результате получим

$$
\|\hat{d}_{i+1}\|_2 \leq \frac{\|d_i\|_2 |\eta|}{(1 - \epsilon^2/2)(\operatorname{gap}(1, \Lambda) - |\eta|)} \leq \frac{2\|\Lambda\|_2 \epsilon^3}{(1 - \epsilon^2/2)(\operatorname{gap}(1, \Lambda) - 2\|\Lambda\|\epsilon^2)},
$$

или $\|\hat{d}_{i+1}\|_2 = O(\epsilon^3)$. Наконец, поскольку $x_{i+1} = e_1 + d_{i+1} = (e_1 + \hat{d}_{i+1})/\|e_1 + \hat{d}_{i+1}\|_2$, то и $\|d_{i+1}\|_2 = O(\epsilon^3)$. □

5.3.3. «Разделяй-и-властвуй»

Это наиболее быстрый из существующих методов, если нужны все собственные значения и собственные векторы трехдиагональной матрицы, начиная с порядка n, примерно равного 26. (Точное значение этого порогового порядка зависит от компьютера.) Его численно устойчивая реализация весьма не тривиальна. В самом деле, хотя впервые метод был предложен еще в 1981 г. [59], «правильный» способ его реализации был найден лишь в 1992 г. [127, 131]. Этот способ воплощен LAPACK-программами `ssyevd` (для плотных матриц) и `sstevd` (для трехдиагональных матриц). В них стратегия «разделяй-и-властвуй» используется для матриц порядка, большего чем 25. Для матриц меньшего порядка (или если нужны только собственные значения) происходит автоматический переход к QR-итерации.

Обсудим вначале структуру алгоритма в целом, а уже затем численные детали. Пусть

$$
T = \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m & b_m & & & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & & & \\
 & & & & b_{m+1} & \ddots & & & & \\
 & & & & & & \ddots & & b_{n-1} & \\
 & & & & & & & b_{n-1} & a_n &
\end{array}\right]
$$

$$
= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m - b_m & & & & & & \\
\hline
 & & & & a_{m+1} - b_m & b_{m+1} & & & & \\
 & & & & b_{m+1} & \ddots & & & & \\
 & & & & & & \ddots & & b_{n-1} & \\
 & & & & & & & b_{n-1} & a_n &
\end{array}\right]
$$

$$
+ \left[\begin{array}{ccccc|ccccc}
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & b_m & & b_m & & & & \\
\hline
 & & & b_m & & b_m & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & &
\end{array}\right]
$$

$$
= \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_m \cdot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} [0, \ldots, 0, 1, 1, 0 \ldots, 0] \equiv \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_m v v^T.
$$

Предположим, что нам известны спектральные разложения матриц T_1 и T_2: $T_i = Q_i \Lambda_i Q_i^T$. В действительности, они будут рекурсивно вычисляться тем же самым алгоритмом. Установим связь между собственными значениями матри-

цы T и собственными значениями матриц T_1 и T_2. Имеем

$$\begin{aligned} T &= \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T \\ &= \begin{bmatrix} Q_1 \Lambda_1 Q_1^T & 0 \\ 0 & Q_2 \Lambda_2 Q_2^T \end{bmatrix} + b_m v v^T \\ &= \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix}, \end{aligned}$$

где

$$u = \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix} v = \begin{bmatrix} \text{последний столбец матрицы } Q_1^T \\ \text{первый столбец матрицы } Q_2^T \end{bmatrix},$$

так как $v = [0, \ldots, 0, 1, 1, 0, \ldots, 0]^T$. Следовательно, T имеет те же собственные значения, что и подобная ей матрица $D + \rho u u^T$, где $D = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}$ — диагональная матрица, $\rho = b_m$ — число, а u — вектор. Будем предполагать, не ограничивая общности, что диагональные элементы $d_1, \ldots, d_n$ матрицы D упорядочены по убыванию: $d_n \leq \cdots \leq d_1$.

Чтобы найти собственные значения матрицы $D + \rho u u^T$, вычислим ее характеристический многочлен, считая пока матрицу $D - \lambda I$ невырожденной. Тогда

$$\det(D + \rho u u^T - \lambda I) = \det((D - \lambda I)(I + \rho (D - \lambda I)^{-1} u u^T)). \tag{5.13}$$

Поскольку $D - \lambda I$ невырожденна, $\det(I + \rho (D - \lambda I)^{-1} u u^T) = 0$ тогда и только тогда, когда λ — собственное значение. Заметим, что матрица $I + \rho (D - \lambda I)^{-1} u u^T$ получается из единичной добавлением матрицы ранга 1. Определитель такой матрицы легко вычислить.

Лемма 5.1. *Справедливо равенство* $\det(I + xy^T) = 1 + y^T x$, *где* x *и* y — *векторы.*

Доказательство леммы составляет содержание вопроса 5.14.

Таким образом,

$$\begin{aligned} \det(I + \rho (D - \lambda I)^{-1} u u^T) &= 1 + \rho u^T (D - \lambda I)^{-1} u \\ &= 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} \equiv f(\lambda), \end{aligned} \tag{5.14}$$

т. е. собственные значения матрицы T суть корни так называемого *векового уравнения* $f(\lambda) = 0$. Если все числа d_i различны и все $u_i \neq 0$ (случай общего положения), то $f(\lambda)$ имеет график типа показанного на рис. 5.2 (где $n = 4$ и $\rho > 0$).

Можно видеть, что прямая $y = 1$ является горизонтальной асимптотой для этого графика, а прямые $\lambda = d_i$ суть вертикальные асимптоты. Поскольку $f'(\lambda) = \rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2} > 0$, функция возрастает всюду, кроме точек $\lambda = d_i$. Поэтому корни функции разделяются числами d_i и еще один корень находится справа от точки d_1 (на рис. 5.2 $d_1 = 4$). (При $\rho < 0$ функция $f(\lambda)$ всюду убывает

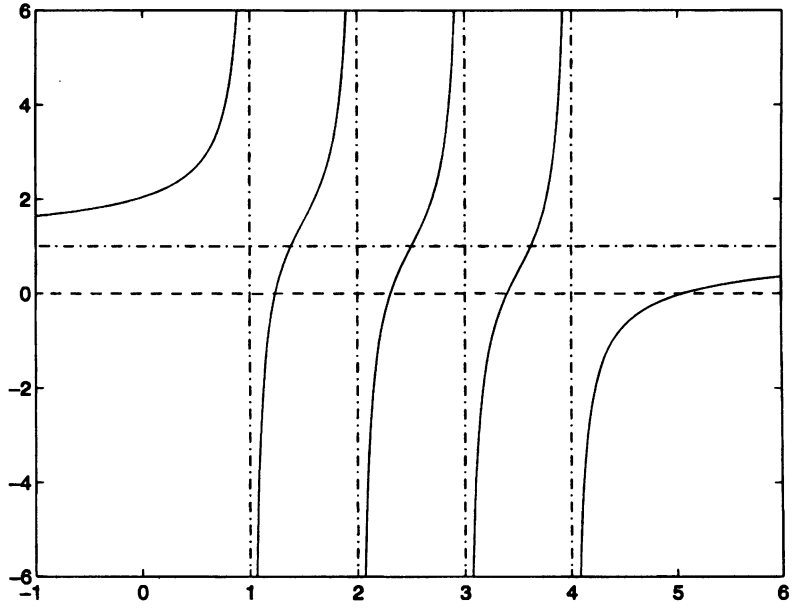


Рис. 5.2. График функции $f(\lambda) = 1 + \frac{0.5}{1-\lambda} + \frac{0.5}{2-\lambda} + \frac{0.5}{3-\lambda} + \frac{0.5}{4-\lambda}$.

и соответствующий корень находится слева от точки d_n.) Для функции $f(\lambda)$, монотонной и гладкой на каждом из интервалов (d_{i+1}, d_i), можно найти вариант метода Ньютона, который быстро и монотонно сходится к каждому из корней, если начальная точка взята в (d_{i+1}, d_i). Мы обсудим детали позднее в данном разделе. Пока нам достаточно знать, что на практике метод сходится к каждому собственному значению за ограниченное число шагов. Поскольку вычисление $f(\lambda)$ и $f'(\lambda)$ стоит $O(n)$ флопов, для вычисления одного собственного значения достаточно $O(n)$ флопов, а для вычисления всех n собственных значений матрицы $D + \rho uu^T$ требуется $O(n^2)$ флопов.

Для собственных векторов матрицы $D + \rho uu^T$ мы легко можем получить явные выражения.

Лемма 5.2. *Если α — собственное значение матрицы $D+\rho uu^T$, то соответствующий собственный вектор равен $(D-\alpha I)^{-1}u$. Поскольку матрица $D-\alpha I$ диагональная, для вычисления такого вектора достаточно $O(n)$ флопов.*

Доказательство.

$$\begin{aligned}(D + \rho uu^T)[(D - \alpha I)^{-1}u] &= (D - \alpha I + \alpha I + \rho uu^T)(D - \alpha I)^{-1}u \\ &= u + \alpha(D - \alpha I)^{-1}u + u[\rho u^T(D - \alpha I)^{-1}u] \\ &= u + \alpha(D - \alpha I)^{-1}u - u \\ &\qquad \text{поскольку } \rho u^T(D - \alpha I)^{-1}u + 1 = f(\alpha) = 0 \\ &= \alpha[(D - \alpha I)^{-1}u], \quad \text{что и требовалось.} \qquad \square\end{aligned}$$

Для вычисления по этой простой формуле всех n собственных векторов требуется $O(n^2)$ флопов. К сожалению, формула не обеспечивает численной устойчивости, так как для двух очень близких значений α_i может давать неортогональные приближенные собственные векторы u_i. Потребовалось целое десятилетие для того, чтобы найти устойчивую альтернативу исходному описанию алгоритма. Снова детали будут обсуждаться позднее в данном разделе.

Алгоритм в целом является рекурсивным.

Алгоритм 5.2. *Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы посредством стратегии «разделяй-и-властвуй»:*

proc $dc_eig(T, Q, \Lambda)$ ... *по входной матрице* T *вычисляются выходные матрицы* Q *и* Λ, *такие, что* $T = Q\Lambda Q^T$

Если T — *матрица размера* 1×1
присвоить выходным параметрам значения $Q = 1$, $\Lambda = T$
иначе
представить T *в виде* $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T$
call $dc_eig(T_1, Q_1, \Lambda_1)$
call $dc_eig(T_2, Q_2, \Lambda_2)$
построить $D + \rho u u^T$ *по* Λ_1, Λ_2, Q_1, Q_2
найти матрицу собственных значений Λ
и матрицу собственных векторов Q' *для матрицы* $D + \rho u u^T$
построить матрицу собственных векторов Q *для матрицы* T:
$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot Q'$
присвоить выходным параметрам значения Q *и* Λ
endif

Проанализируем сложность алгоритма 5.2. Пусть $t(n)$ — число флопов при обработке матрицы размера $n \times n$ процедурой dc_eig. Тогда

$t(n)$	$=$	$2t(n/2)$	два рекурсивных обращения к $\mathrm{dc_eig}(T_i, Q_i, \Lambda_i)$
		$+O(n^2)$	вычисление собственных значений матрицы $D + \rho u u^T$
		$+O(n^2)$	вычисление собственных векторов матрицы $D + \rho u u^T$
		$+c \cdot n^3$	вычисление матрицы $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot Q'$

Если Q_1, Q_2 и Q' рассматриваются как плотные матрицы и используется стандартный алгоритм матричного умножения, то константа c в последней строке равна 1. Таким образом, именно это умножение составляет наиболее трудоемкую часть алгоритма в целом. Игнорируя члены порядка n^2, получаем $t(n) = 2t(n/2) + cn^3$. Решая это разностное уравнение, находим $t(n) \approx c\frac{4}{3}n^3$ (см. вопрос 5.15). На практике константа c обычно гораздо меньше 1, потому что матрица Q' весьма разрежена вследствие явления, называемого *дефляцией*.

Мы обсудим дефляцию в следующем разделе, а затем перейдем к деталям, связанным с решением векового уравнения и устойчивым вычислением

собственных векторов. Наконец, мы поговорим о том, как ускорить метод с помощью техники FMM, используемой при моделировании электростатики заряженных частиц [124]. При первом чтении книги эти разделы можно опустить.

Дефляция

До сих пор предполагалось, что все d_i различны и все u_i отличны от нуля. Если это не так, вековое уравнение $f(\lambda) = 0$ имеет k вертикальных асимптот, где $k < n$, а потому k корней. Однако оказывается, что остальные $n - k$ собственных значений могут быть определены без каких-либо усилий: если $d_i = d_{i+1}$ или $u_i = 0$, то легко показать, что d_i является собственным значением и для матрицы $D + \rho uu^T$ (см. вопрос 5.16). В такой ситуации мы говорим о *дефляции*. На практике выбирается некоторое пороговое значение и дефляция для числа d_i регистрируется, если в смысле этого порога d_i достаточно близко́ к d_{i+1} либо u_i достаточно мало.

Оказывается, что в практических вычислениях дефляция происходит весьма часто. В экспериментах со случайными плотными матрицами, собственные значения которых равномерно распределены, дефляция имела место для более чем 15% собственных значений наибольшей матрицы $D + \rho uu^T$. Когда же собственные значения случайных плотных матриц сходились к нулю со скоростью геометрической прогрессии, дефляция наблюдалась для более чем 85% собственных значений! Важно научиться извлекать выгоду из этого обстоятельства, чтобы сделать алгоритм по-настоящему быстрым [59, 210].

Основной выигрыш от использования дефляции состоит не в том, что убыстряется решение векового уравнения — этот этап в любом случае стоит лишь $O(n^2)$ операций. Выигрыш заключается в ускорении матричного умножения на последнем шаге алгоритма. Действительно, если $u_i = 0$, то соответствующий собственный вектор есть i-й столбец e_i единичной матрицы (см. вопрос 5.16). Это означает, что e_i является i-м столбцом в матрице Q', поэтому при формировании матрицы Q посредством левого умножения на Q_1 и Q_2 вычисление i-го столбца не требует никаких затрат. Аналогичное упрощение имеет место в случае $d_i = d_{i+1}$. При дефляции многих собственных значений устраняется большая часть работы, связанной с матричным умножением. Это отчетливо проявляется в численных экспериментах, обсуждаемых в разд. 5.3.6.

Решение векового уравнения

Предположим, что некоторое число u_i, хотя и мало, все же недостаточно мало для того, чтобы была зарегистрирована дефляция. В этом случае применение метода Ньютона к решению векового уравнения встречается с затруднениями. Вспомним, что пересчет приближенного решения λ_j уравнения $f(\lambda) = 0$ в методе Ньютона основан на следующих положениях:

1. Вблизи точки $\lambda = \lambda_j$ функция $f(\lambda)$ аппроксимируется линейной функцией $l(\lambda)$; график последней есть прямая линия, касающаяся графика функции $f(\lambda)$ при $\lambda = \lambda_j$.
2. В качестве λ_{j+1} берется нуль этого линейного приближения, т. е. $l(\lambda_{j+1})$=0.

Функция, показанная на рис. 5.2, не доставляет видимых трудностей методу Ньютона, поскольку вблизи каждого своего нуля $f(\lambda)$ достаточно хорошо ап-

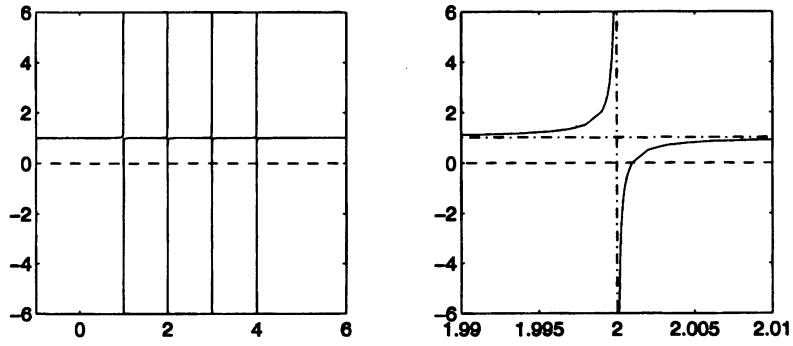


Рис. 5.3. График функции $f(\lambda) = 1 + \frac{10^{-3}}{1-\lambda} + \frac{10^{-3}}{2-\lambda} + \frac{10^{-3}}{3-\lambda} + \frac{10^{-3}}{4-\lambda}$.

проксимируется линейными функциями. Однако рассмотрим график функции на рис. 5.3. Она получена из функции на рис. 5.2 заменой значения .5 для u_i^2 на .001. Это новое значение недостаточно мало для того, чтобы вызвать дефляцию. График функции в левой части рис. 5.3 визуально не отличим от ее вертикальных и горизонтальных асимптот, поэтому в правой части укрупненно воспроизведен фрагмент графика, прилегающий к вертикальной асимптоте $\lambda = 2$. Видно, что график слишком быстро «выполняет поворот» и для большей части значений λ почти горизонтален. Поэтому, применяя метод Ньютона почти к любому начальному приближению λ_0, мы получаем линейное приближение $l(\lambda)$ с почти горизонтальным графиком и малым положительным угловым коэффициентом. В результате λ_1 является отрицательным числом, огромным по абсолютной величине, которое совершенно бесполезно в качестве приближения к истинному корню.

Чтобы найти выход из этого положения, можно модифицировать метод Ньютона следующим образом: раз $f(\lambda)$ нельзя хорошо приблизить линейной функцией $l(\lambda)$, попробуем взять в качестве приближения какую-нибудь другую простую функцию $h(\lambda)$. Нет ничего особого именно в прямых линиях: для метода Ньютона вместо $l(\lambda)$ можно взять любое приближение $h(\lambda)$, значения и нули которого легко вычисляются. Функция $f(\lambda)$ имеет полюсы в точках d_i и d_{i+1}, которые определяют ее поведение в соответствующих окрестностях. Поэтому при поиске корня в интервале (d_{i+1}, d_i) естественно выбрать функцию $h(\lambda)$, также имеющую эти полюсы, т. е. функцию вида

$$h(\lambda) = \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} + c_3.$$

Константы c_1, c_2 и c_3, обеспечивающие, что $h(\lambda)$ есть приближение к $f(\lambda)$, можно выбрать несколькими способами. Мы изложим несколько упрощенный вариант способа, используемого LAPACK-программой `slaed4` [172, 45], но вначале отметим, что если c_1, c_2 и c_3 уже известны, то уравнение $h(\lambda) = 0$ легко решается относительно λ, поскольку сводится к эквивалентному квадратному уравнению

$$c_1(d_{i+1} - \lambda) + c_2(d_i - \lambda) + c_3(d_i - \lambda)(d_{i+1} - \lambda) = 0.$$

Пусть λ_j — приближенное значение корня. Определим c_1, c_2 и c_3 так, чтобы

$$\frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} + c_3 = h(\lambda) \approx f(\lambda) = 1 + \rho \sum_{k=1}^{n} \frac{u_k^2}{d_k - \lambda}$$

для λ в окрестности λ_j. Заметим, что

$$f(\lambda) = 1 + \rho \sum_{k=1}^{i} \frac{u_k^2}{d_k - \lambda} + \rho \sum_{k=i+1}^{n} \frac{u_k^2}{d_k - \lambda} \equiv 1 + \psi_1(\lambda) + \psi_2(\lambda).$$

Если $\lambda \in (d_{i+1}, d_i)$, то $\psi_1(\lambda)$ есть сумма положительных слагаемых, а $\psi_2(\lambda)$ — сумма отрицательных. Поэтому и $\psi_1(\lambda)$, и $\psi_2(\lambda)$ могут быть вычислены с высокой точностью; однако при их сложении вполне вероятно взаимное уничтожение верных разрядов и потеря относительной точности в сумме. Возьмем числа c_1 и $\hat{c}_1$, такие, что функция

$$h_1(\lambda) \equiv \hat{c}_1 + \frac{c_1}{d_i - \lambda} \quad \text{удовлетворяет условиям}$$
$$h_1(\lambda_j) = \psi_1(\lambda_j) \quad \text{и} \quad h_1'(\lambda_j) = \psi_1'(\lambda_j). \tag{5.15}$$

Это означает, что гипербола, являющаяся графиком функции $h_1(\lambda)$, касается графика функции $\psi_1(\lambda)$ при $\lambda = \lambda_j$. Два условия в (5.15) — это обычные условия метода Ньютона, за исключением того, что вместо прямой в качестве приближения используется гипербола. Легко проверить, что $c_1 = \psi_1'(\lambda_j)(d_i - \lambda_j)^2$ и $\hat{c}_1 = \psi_1(\lambda_j) - \psi_1'(\lambda_j)(d_i - \lambda_j)$ (см. вопрос 5.17).

Подобным же образом выбираем c_2 и $\hat{c}_2$ так, чтобы функция

$$h_2(\lambda) \equiv \hat{c}_2 + \frac{c_2}{d_{i+1} - \lambda} \quad \text{удовлетворяла условиям}$$
$$h_2(\lambda_j) = \psi_2(\lambda_j) \quad \text{и} \quad h_2'(\lambda_j) = \psi_2'(\lambda_j). \tag{5.16}$$

Наконец, полагаем

$$\begin{aligned} h(\lambda) &= 1 + h_1(\lambda) + h_2(\lambda) \\ &= (1 + \hat{c}_1 + \hat{c}_2) + \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} \\ &\equiv c_3 + \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda}. \end{aligned}$$

Пример 5.8. Если для функции на рис. 5.3 начать с $\lambda_0 = 2.5$, то

$$h(\lambda) = \frac{1.1111 \cdot 10^{-3}}{2 - \lambda} + \frac{1.1111 \cdot 10^{-3}}{3 - \lambda} + 1.$$

График этой функции визуально не отличим от графика функции $f(\lambda)$, показанного в правой части рис. 5.3. Решая уравнение $h(\lambda_1) = 0$, получаем $\lambda_1 = 2.0011$. Это значение имеет 4 верных десятичных разряда. Продолжая таким же образом, находим λ_2 с 11 верными разрядами и λ_3, все 16 разрядов которого верны. ◇

Алгоритм, реализованный LAPACK-программой `slaed4`, является небольшой модификацией только что описанного метода (который назван в [172]

алгоритмом *среднего пути*). В очень обширных численных тестах LAPACK-программе требовалось в среднем две-три итерации для того, чтобы найти собственное значение с полной машинной точностью, и ни разу не понадобилось более семи итераций.

Устойчивое вычисление собственных векторов

Как только из векового уравнения найдены собственные значения α_i матрицы $D+\rho uu^T$, можно вычислить собственные векторы, пользуясь простой формулой $(D-\alpha_i I)^{-1}u$ из леммы 5.2. К сожалению, вычисления по этой формуле могут быть неустойчивы [59, 90, 234]; так будет, в частности, когда собственные значения α_i и α_{i+1} очень близки. Интуитивно проблема состоит в том, что выражения $(D-\alpha_i I)^{-1}u$ и $(D-\alpha_{i+1}I)^{-1}u$ «очень похожи», в то время как требуется получить ортогональные собственные векторы. Более точно, если α_i и α_{i+1} очень близки, оба близки к находящемуся между ними числу d_i. Поэтому имеет место заметная потеря верных знаков при вычислении $d_i-\alpha_i$ или $d_i-\alpha_{i+1}$, либо при решении векового уравнения методом Ньютона. В любом из этих случаев числа $d_i-\alpha_i$ и $d_i-\alpha_{i+1}$ могут содержать большие относительные ошибки, вследствие чего вычисленные собственные векторы $(D-\alpha_i I)^{-1}u$ и $(D-\alpha_{i+1}I)^{-1}u$ очень неточны и далеки от ортогональности.

Первоначальные попытки разрешить эту проблему [90, 234] предусматривали использование арифметики двойной точности при решении векового уравнения (в предположении, что входные данные имеют обычную точность), чтобы разности $d_i-\alpha_i$ и $d_i-\alpha_{i+1}$ могли быть вычислены с хорошей точностью. Однако, если уже входные данные имеют двойную точность, это означает, что требуется переход к учетверенной точности, недоступной для многих компьютеров и языков или, по меньшей мере, доступной лишь дорогой ценой. Как указано в разд. 1.5, учетверенную точность можно моделировать посредством двойной точности [234, 204]. Это можно сделать переносимым и относительно эффективным образом, если округления в основной арифметике с плавающей точкой выполняются достаточно точно. В частности, требуется, чтобы, исключая переполнения и машинные нули, выполнялось соотношение $\mathrm{fl}(a\pm b)=(a\pm b)(1+\delta)$, где $|\delta|=O(\varepsilon)$; см. по этому поводу разд. 1.5 и вопрос 1.18. К сожалению, эти эффективные алгоритмы моделирования не пригодны для компьютеров Cray 2, YMP и C90, где округления производятся недостаточно точно.

В конце концов, была найдена другая формула, устраняющая необходимость в моделировании арифметики повышенной разрядности. Она основана на следующей теореме Лёвнера [129, 179]:

Теорема 5.10 (Лёвнер). *Пусть для диагональной матрицы* $D=\mathrm{diag}(d_1,\ldots,d_n)$ *справедливы неравенства* $d_n<\cdots<d_1$, *и пусть заданы числа* $\alpha_n<\cdots<\alpha_1$, *удовлетворяющие* условиям перемежаемости

$$d_n<\alpha_n<\cdots<d_{i+1}<\alpha_{i+1}<d_i<\alpha_i<\cdots<d_1<\alpha_1.$$

Тогда найдется вектор $\hat{u}$, *такой, что* α_i *суть точные собственные значения матрицы* $\widehat{D}\equiv D+\hat{u}\hat{u}^T$. *Компоненты вектора* $\hat{u}$ *выражаются формулами*

$$|\hat{u}_i|=\left[\frac{\prod_{j=1}^{n}(\alpha_j-d_i)}{\prod_{j=1,\ j\neq i}^{n}(d_j-d_i)}\right]^{1/2}.$$

Доказательство. Характеристический многочлен матрицы $\widehat{D}$ можно записать, с одной стороны, как $\det(\widehat{D} - \lambda I) = \prod_{j=1}^{n}(\alpha_j - \lambda)$, а с другой стороны, как

$$\det(\widehat{D} - \lambda I) = \left[\prod_{j=1}^{n}(d_j - \lambda)\right] \cdot \left(1 + \sum_{j=1}^{n} \frac{\hat{u}_j^2}{d_j - \lambda}\right)$$

$$= \left[\prod_{j=1}^{n}(d_j - \lambda)\right] \cdot \left(1 + \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\hat{u}_j^2}{d_j - \lambda}\right) + \left[\prod_{\substack{j=1 \\ j\neq i}}^{n}(d_j - \lambda)\right] \cdot \hat{u}_i^2.$$

Здесь использованы формулы (5.13) и (5.14). Полагая $\lambda = d_i$ и приравнивая оба выражения для $\det(\widehat{D} - \lambda I)$, имеем

$$\prod_{j=1}^{n}(\alpha_j - d_i) = \hat{u}_i^2 \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n}(d_j - d_i),$$

или

$$\hat{u}_i^2 = \frac{\prod_{j=1}^{n}(\alpha_j - d_i)}{\prod_{j=1,\ j\neq i}^{n}(d_j - d_i)}.$$

Используя свойство перемежаемости, можно показать, что дробь в правой части положительна, поэтому извлечение квадратного корня возможно и дает требуемое выражение для $\hat{u}_i$. □

Сформулируем теперь устойчивый алгоритм вычисления собственных значений и собственных векторов (чтобы упростить изложение, положим $\rho = 1$).

Алгоритм 5.3. *Вычисление собственных значений и собственных векторов матрицы* $D + uu^T$.

Решая вековое уравнение $1 + \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} = 0$, *найти (приближенные) собственные значения* α_i *матрицы* $D + uu^T$.

Используя теорему Лёвнера, вычислить вектор $\hat{u}$, *такой, что* α_i *являются «точными» собственными значениями матрицы* $D + uu^T$.

С помощью леммы 5.2 вычислить собственные векторы матрицы $D + uu^T$.

Приведем краткое обоснование устойчивости этого алгоритма. Анализируя критерий останова в программе, решающей вековое уравнение, можно показать, что $\|uu^T - \hat{u}\hat{u}^T\|_2 \leq O(\varepsilon)(\|D\|_2 + \|uu^T\|_2)$[1]. Это означает, что $D + uu^T$ и $D + \hat{u}\hat{u}^T$ настолько близки, что собственные значения и собственные векторы второй матрицы суть устойчивые приближения собственных значений и собственных векторов первой. Заметим теперь, что в формулу для $\hat{u}_i$ в теореме Лёвнера входят только разности $d_j - d_i$ и $\alpha_j - d_i$ чисел с плавающей точкой, произведения и частные этих разностей и квадратный корень. Предположим, что *машинная арифметика с плавающей точкой достаточно точна* в том смысле, что $\mathrm{fl}(a \odot b) = (a \odot b)(1 + \delta)$ для всех операций $\odot \in \{+, -, \times, /\}$ и

[1] Говоря более подробно, для справедливости этой оценки необходимо, чтобы программа определяла *не* α_i, а наименьшую из разностей $\alpha_i - d_i$ и $d_{i+1} - \alpha_i$.

$\mathrm{sqrt}(a) = \sqrt{a} \cdot (1+\delta)$, где $|\delta| = O(\varepsilon)$. Тогда вычисления по формуле Лёвнера можно провести с высокой относительной точностью. В частности, легко показать, что, в отсутствие переполнений и машинных нулей,

$$\mathrm{fl}\left(\left[\frac{\prod_{j=1}^{n}(\alpha_j - d_i)}{\prod_{j=1,\ j\neq i}^{n}(d_j - d_i)}\right]^{1/2}\right) = (1+(4n-2)\delta)\cdot\left[\frac{\prod_{j=1}^{n}(\alpha_j - d_i)}{\prod_{j=1,\ j\neq i}^{n}(d_j - d_i)}\right]^{1/2},$$

где $|\delta| = O(\varepsilon)$. С высокой относительной точностью можно реализовать и формулу из леммы 5.2, поэтому собственные векторы матрицы $D + \hat{u}\hat{u}^T$ могут быть вычислены с высокой относительной точностью. В частности, с большой точностью выполняются соотношения ортогональности.

Резюмируем сказанное таким образом: если машинная арифметика с плавающей точкой достаточно точна, то алгоритм 5.3 с очень высокой точностью определяет собственные значения и собственные векторы матрицы $D + \hat{u}\hat{u}^T$, слабо отличающейся от исходной матрицы $D + uu^T$. Это свойство означает численную устойчивость алгоритма.

Читатель должен отметить, что именно нарушение этого требования достаточной точности машинной арифметики привело к тому, что на некоторых компьютерах Cray не работает моделирование учетверенной точности, предложенное в [234, 204]. Пока нам не удалось разработать алгоритма, который бы надежно работал на этих компьютерах. Необходим еще один трюк. Дело в том, что недостаточно точными операциями этих компьютеров являются только сложение и вычитание, причем это связано с отсутствием так называемого *вспомогательного разряда* в аппаратной реализации арифметики с плавающей точкой. Это означает, что при сложении или вычитании самый правый бит операнда может рассматриваться как 0, даже если, в действительности, он равен 1. Если большинство старших разрядов взаимно уничтожаются, то этот «потерянный бит» становится значимым. Например, при вычитании 1 из ближайшего к ней, но меньшего числа с плавающей точкой (это вычитание приводит к взаимному уничтожению всех разрядов, кроме последнего) на компьютере Cray C90 получается результат, вдвое больший правильного, а на Cray 2 получается 0. Однако если самый правый бит действительно равен 0, то никакой ошибки не происходит. Это подсказывает такой трюк: перед применением теоремы Лёвнера или леммы 5.2 в алгоритме 5.3 намеренно заменить самые правые биты во всех числах d_i на 0. Алгоритм при этом сохраняет устойчивость, так как числа d_i и α_i претерпели лишь небольшие относительные изменения[1].

Более подробное описание данного алгоритма можно найти в [129, 131]. LAPACK-программа `slaed3` является его реализацией.

[1] Чтобы заменить самый правый бит числа с плавающей точкой β на 0 на машине Cray, необходимо лишь выполнить присваивание $\beta := (\beta + \beta) - \beta$. Это несложное вычисление вовсе не меняет β на машине с «правильной» двоичной арифметикой (исключая переполнения, которых легко избежать). Но на машине Cray оно действительно заменяет самый правый бит нулем. Приглашаем читателя, знакомого с арифметикой этих машин, доказать данное утверждение. Единственная остающаяся трудность — помешать оптимизирующему транслятору удалить это присваивание из программы; некоторые чересчур усердные оптимизаторы, действительно, могут это сделать. Для нынешнего поколения трансляторов указанную трудность можно преодолеть, вычисляя $\beta + \beta$ посредством обращения к функции, хранящейся в файле, отличном от основной программы. Мы надеемся, что к тому времени, когда трансляторы будут усовершенствованы до такой степени, что смогут оптимизировать и эту ситуацию, арифметика машин Cray отомрет окончательно.

Ускорение алгоритма «разделяй-и-властвуй» посредством метода FMM

Метод FMM был первоначально предложен в [124] для совершенно другой задачи вычисления электрических сил взаимодействия в ансамбле из n заряженных частиц или гравитационных сил в системе из n масс. Мы ограничимся указанием связи этих задач с вычислением собственных значений и собственных векторов, адресуя читателя за деталями к [131].

Пусть $d_1, \ldots, d_n$ — трехмерные радиус-векторы n частиц с зарядами $z_i \cdot u_i$, а $\alpha_1, \ldots, \alpha_n$ — радиус-векторы других n частиц с единичными положительными зарядами. Согласно закону обратных квадратов, сила, действующая на частицу в позиции α_j со стороны частиц в позициях $d_1, \ldots, d_n$, пропорциональна величине

$$f_j = \sum_{i=1}^{n} \frac{z_i u_i (d_i - \alpha_j)}{\|d_i - \alpha_j\|_2^3}.$$

Если электростатика моделируется в двумерном пространстве вместо трехмерного, то силы подчиняются закону обратной пропорциональности[1]

$$f_j = \sum_{i=1}^{n} \frac{z_i u_i (d_i - \alpha_j)}{\|d_i - \alpha_j\|_2^2}.$$

Поскольку d_i и α_j — это векторы в $\mathbf{R^2}$, их можно интерпретировать и как комплексные числа. В таком случае

$$f_j = \sum_{i=1}^{n} \frac{z_i u_i}{\overline{d}_i - \overline{\alpha}_j},$$

где $\overline{d}_i$ и $\overline{\alpha}_j$ — числа, сопряженные соответственно с d_i и α_j. Если все d_i и α_j окажутся вещественными числами, то произойдет еще большее упрощение:

$$f_j = \sum_{i=1}^{n} \frac{z_i u_i}{d_i - \alpha_j}.$$

Рассмотрим теперь матрично-векторное умножение $f^T = z^T Q'$, где Q' есть матрица собственных векторов для $D + uu^T$. Согласно лемме 5.2, $Q'_{ij} = u_i s_j / (d_i - \alpha_j)$, где масштабирующий множитель s_j выбран так, чтобы j-й столбец был нормирован. Поэтому j-й элемент вектора $f^T = z^T Q'$ равен

$$f_j = \sum_{i=1}^{n} z_i Q'_{ij} = s_j \sum_{i=1}^{n} \frac{z_i u_i}{d_i - \alpha_j},$$

т. е. (с точностью до множителя s_j) той же сумме, что и электростатическая сила. Итак, самая трудоемкая часть алгоритма «разделяй-и-властвуй», а именно матричное умножение в последней строке алгоритма 5.2, эквивалентна вычислению электростатических сил.

Вычисление этой суммы для $j = 1, \ldots, n$, по видимости, требует $O(n^2)$ флопов. С помощью FMM и других родственных методов [124, 23] ее можно вычислить (приближенно, но с очень хорошей точностью) за время $O(n \cdot \log n)$

[1] Технически это означает, что потенциал удовлетворяет уравнению Пуассона в двумерном пространстве, а не трехмерном.

или даже $O(n)$. (См. лекции «Быстрые иерархические методы в задаче N тел» на PARALLEL_HOMEPAGE. В них даны все необходимые подробности.)

Однако этой идеи самой по себе недостаточно для того, чтобы снизить цену алгоритма «разделяй-и-властвуй» до $O(n \cdot \log^p n)$. В конце концов, матрица Q на выходе алгоритма состоит из n^2 элементов, что, по-видимому, должно означать, что цена алгоритма никак не может быть меньше, чем n^2. Поэтому мы должны научиться представлять Q, используя менее чем n^2 независимых чисел. Это оказывается возможным, так как трехдиагональная $n \times n$-матрица имеет лишь $2n-1$ «степеней свободы» (например, диагональные и наддиагональные элементы), в качестве которых можно принять n собственных значений, оставив еще $n-1$ для ортогональной матрицы Q. Иначе говоря, не всякая ортогональная матрица может быть матрицей собственных векторов для симметричной трехдиагональной матрицы T. На эту роль годится лишь $(n-1)$-мерное подмногообразие всего многообразия ортогональных матриц, размерность которого равна $n(n-1)/2$.

Матрица Q представляется с помощью дерева, вычисляемого алгоритмом 5.2. Вместо того чтобы производить накопление по формуле $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot Q'$, мы сохраняем все матрицы Q', по одной на каждый узел дерева. Кроме того, вместо хранения Q' в явном виде, мы храним соответствующие величины D, ρ, u и собственные значения α_i матрицы $D + \rho u u^T$. Мы можем действовать таким образом, потому что в методе FMM требуется лишь уметь умножать Q' на векторы. В результате память, необходимая для хранения Q, снижается с n^2 машинных слов до $O(n \cdot \log n)$. Итак, на выходе алгоритма получаем матрицу Q в «факторизованном» виде, представленном множителями Q' для всех узлов дерева. Такое представление Q вполне достаточно для наших целей, поскольку позволяет умножить ее на любой вектор за время $O(n \log^p n)$, используя метод FMM.

5.3.4. Бисекция и обратная итерация

В алгоритме бисекции теорема Сильвестра об инерции (теорема 5.3) используется для вычисления только k желаемых собственных значений (вместо всех n) с затратой $O(nk)$ операций. Напомним, что $\mathrm{Inertia}(A) = (\nu, \zeta, \pi)$, где ν, ζ и π дают, соответственно, число отрицательных, нулевых и положительных собственных значений матрицы A. Если X — невырожденная матрица, то, согласно теореме Сильвестра, $\mathrm{Inertia}(A) = \mathrm{Inertia}(X^T A X)$.

Предположим теперь, что посредством гауссова исключения найдена факторизация $A - zI = LDL^T$, где L — невырожденная матрица, а D диагональная. Тогда $\mathrm{Inertia}(A - zI) = \mathrm{Inertia}(D)$, а инерция диагональной матрицы D находится тривиально. (Ниже мы используем обозначение «$\#\ d_{ii} < 0$» для количества чисел d_{ii}, меньших нуля, и другие обозначения того же типа.)

$$\begin{aligned}\mathrm{Inertia}(A-zI) &= (\#\ d_{ii}<0, \#\ d_{ii}=0, \#\ d_{ii}>0) \\ &= (\text{количество отрицательных собственных значений} \\ &\qquad \text{матрицы } A-zI, \\ &\quad \text{количество нулевых собственных значений матрицы } A-zI, \\ &\quad \text{количество положительных собственных}\end{aligned}$$

значений матрицы $A - zI$)
= (количество собственных значений матрицы A,
меньших, чем z;
количество собственных значений, равных z;
количество собственных значений, бóльших, чем z).

Предположим, что $z_1 < z_2$, и пусть найдены Inertia$(A - z_1 I)$ и Inertia$(A - z_2 I)$. Тогда число собственных значений матрицы A, попадающих в полуинтервал $[z_1, z_2)$, определяется формулой: (число собственных значений, меньших, чем z_2) – (число собственных значений, меньших, чем z_1).

Чтобы превратить это наблюдение в алгоритм, определим величину

$$\text{Negcount}\,(A, z) = \text{количество собственных значений матрицы } A, \text{ меньших, чем } z.$$

Алгоритм 5.4. *Бисекция: найти все собственные значения матрицы A, принадлежащие полуинтервалу $[a, b)$ с заданным допуском на ошибку* tol*:*

$n_a = Negcount(A, a)$
$n_b = Negcount(A, b)$
если $n_a = n_b$, *то выход из алгоритма, так как в* $[a, b)$
нет собственных значений матрицы A
поместить $[a, n_a, b, n_b]$ *в Worklist*
  /* *Worklist — это список полуинтервалов* $[a, b)$, *содержащих собственные значения с номерами от* $n - n_a$ *до* $n - n_b + 1$. *Эти полуинтервалы в алгоритме делятся пополам до тех пор, пока их длина не станет меньше допуска* tol. */
while Worklist непуст do
  удалить $[\text{low}, n_{\text{low}}, \text{up}, n_{\text{up}}]$ *из Worklist*
  if(up − low < tol) *then*
    вывести сообщение «в полуинтервале [low, up) *находятся* $n_{\text{up}} - n_{\text{low}}$ *собственных значений»*
  else
    mid = (low + up)/2
    $n_{\text{mid}} = Negcount(A, \text{mid})$
    if $n_{\text{mid}} > n_{\text{low}}$ *then ... в* [low, mid) *есть собственные значения*
      поместить $[\text{low}, n_{\text{low}}, \text{mid}, n_{\text{mid}}]$ *в Worklist*
    end if
    if $n_{\text{up}} > n_{\text{mid}}$ *then ... в* [mid, up) *есть собственные значения*
      поместить $[\text{mid}, n_{\text{mid}}, \text{up}, n_{\text{up}}]$ *в Worklist*
    end if
  end if
end while

Пусть $\alpha_1 \geq \cdots \geq \alpha_n$ — собственные значения матрицы A. Изложенную идею можно использовать и для определения чисел α_j с номерами $j = j_0, j_0 + 1, \ldots j_1$. Действительно, мы знаем, что в полуинтервале [low, up) находятся собственные значения с номерами от $\alpha_{n-n_{\text{low}}}$ до $\alpha_{n-n_{\text{up}}+1}$.

Если A — плотная матрица, то число $\mathrm{Negcount}(A, z)$ можно определить с помощью симметричного гауссова исключения с выбором главного элемента, как это описано в разд. 2.7.2. Однако такой метод не эффективен, поскольку требует $O(n^3)$ операций для каждого значения z. В то же время, для трехдиагональной A число $\mathrm{Negcount}(A, z)$ найти весьма просто, выполняя разложение без выбора главного элемента:

$$A - zI \equiv \begin{bmatrix} a_1 - z & b_1 & & \\ b_1 & a_2 - z & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n - z \end{bmatrix} = LDL^T$$

$$\equiv \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \cdot \begin{bmatrix} 1 & l_1 & & \\ & \ddots & \ddots & \\ & & \ddots & l_{n-1} \\ & & & 1 \end{bmatrix}.$$

Таким образом, $a_1 - z = d_1$, $d_1 l_1 = b_1$ и далее $l_{i-1}^2 d_{i-1} + d_i = a_i - z$, $d_i l_i = b_i$. Подстановка выражения $l_i = b_i/d_i$ в предпоследнее равенство дает простую рекурсию

$$d_i = (a_i - z) - \frac{b_{i-1}^2}{d_{i-1}}. \tag{5.17}$$

Поскольку выбора главного элемента не производится, этот процесс можно заподозрить в опасной неустойчивости, особенно если встретится малое число d_{i-1}. Однако, в действительности, можно показать, что вычисления по формуле (5.17) очень устойчивы благодаря тому, что матрица $A - zI$ трехдиагональная [73, 74, 156].

Лемма 5.3. *Величины d_i, вычисленные в арифметике с плавающей точкой по формуле* (5.17), *имеют те же знаки (и, следовательно, определяют ту же инерцию), что и величины $\hat{d}_i$, точно вычисленные по матрице $\widehat{A}$, очень близкой к A:*

$$(\widehat{A})_{ii} \equiv \hat{a}_i = a_i \quad \textit{и} \quad (\widehat{A})_{i,i+1} \equiv \hat{b}_i = b_i(1 + \epsilon_i), \quad \textit{где } |\epsilon_i| \le 2.5\varepsilon + O(\varepsilon^2).$$

Доказательство. Пусть $\tilde{d}_i$ — величины, вычисленные с учетом ошибок округления по формуле (5.17):

$$\tilde{d}_i = \left[(a_i - z)(1 + \epsilon_{-,1,i}) - \frac{b_{i-1}^2(1 + \epsilon_{*,i})}{\tilde{d}_{i-1}} \cdot (1 + \epsilon_{/,i})\right](1 + \epsilon_{-,2,i}), \tag{5.18}$$

где все «эпсилоны» ограничены по абсолютной величине машинной точностью ε, а их индексы указывают, какая операция их породила (например, $\epsilon_{-,2,i}$ происходит от второго вычитания при вычислении $\tilde{d}_i$). Определим новые

переменные

$$\hat{d}_i = \frac{\tilde{d}_i}{(1+\epsilon_{-,1,i})(1+\epsilon_{-,2,i})}, \tag{5.19}$$

$$\hat{b}_{i-1} = b_{i-1}\left[\frac{(1+\epsilon_{*,i})(1+\epsilon_{/,i})}{(1+\epsilon_{-,1,i})(1+\epsilon_{-,1,i-1})(1+\epsilon_{-,2,i-1})}\right]^{1/2} \equiv b_{i-1}(1+\epsilon_i).$$

Заметим, что $\hat{d}_i$ и $\tilde{d}_i$ имеют один и тот же знак, а $|\epsilon_i| \leq 2.5\varepsilon + O(\varepsilon^2)$. Подставляя (5.19) в (5.18), получаем

$$\hat{d}_i = a_i - z - \frac{\hat{b}_{i-1}^2}{\hat{d}_{i-1}},$$

что завершает доказательство. □

Полный анализ должен был бы принимать во внимание возможность переполнений и машинных нулей. В действительности, опираясь на правила IEEE-арифметики в отношении обработки особых ситуаций, вычисления можно вести без опаски даже в том случае, когда некоторое число d_{i-1} является точным нулем. В самом деле, тогда имеем $d_i = -\infty$, $d_{i+1} = a_{i+1} - z$, и далее вычисления идут как обычно [73, 81].

Цена одного обращения к процедуре Negcount составляет для трехдиагональной матрицы A, самое большое, $4n$ флопов. Поэтому общая стоимость вычисления k собственных значений равна $O(kn)$. Этот подход реализован LAPACK-программой `sstebz`.

Отметим, что метод бисекции сходится линейно: каждое деление интервала пополам прибавляет один верный разряд. Существует много способов ускорения сходимости, опирающихся на использование метода Ньютона или родственных методов для вычисления нулей характеристического многочлена (значение которого в точке z равно произведению всех чисел d_i) [173, 174, 175, 176, 178, 269].

После того как вычислены (некоторые) собственные значения, вычислить соответствующие собственные векторы можно посредством обратной итерации (алгоритм 4.2); она реализована LAPACK-программой `sstein`. Поскольку в качестве сдвигов можно брать очень хорошие приближения к собственным значениям, для сходимости обычно хватает одной или двух итераций. Так как шаг обратной итерации состоит в решении трехдиагональной системы уравнений (см. разд. 2.7.3), то для вычисления одного собственного вектора достаточно $O(n)$ флопов. Если вычисленные собственные значения $\hat{\alpha}_i, \ldots, \hat{\alpha}_j$ близки друг к другу, то вычисленные для них собственные векторы $\hat{q}_i, \ldots, \hat{q}_j$ могут не быть ортогональны. В этом случае алгоритм предусматривает *переортогонализацию* собственных векторов: вычисляется QR-разложение $[\hat{q}_i, \ldots, \hat{q}_j] = QR$ и каждый вектор $\hat{q}_k$ заменяется k-м столбцом матрицы Q. Это гарантирует, что новые векторы $\hat{q}_k$ будут ортонормированными. QR-разложение обычно вычисляется посредством модифицированного процесса Грама-Шмидта (алгоритм 3.1), т. е. из каждого вектора явным образом вычитаются компоненты в направлениях ранее вычисленных векторов. Если размер кластера k мал, то невелика и стоимость переортогонализации $O(k^2 n)$, поэтому, в принципе, все собственные значения и собственные векторы можно вычислить всего лишь за $O(n^2)$ флопов, пользуясь вначале бисекцией, а потом обратной итерацией.

Это гораздо меньше, чем $O(n^3)$ флопов, необходимых для QR-итерации или (в наихудшем случае) алгоритма «разделяй-и-властвуй». Однако такое ускорение удается получить не всегда: если размер кластера велик, т. е. составляет ощутимую долю от n, то общая стоимость процесса возрастает до тех же $O(n^3)$ флопов. Еще хуже то, что нет никакой гарантии, что вычисленные собственные векторы достаточно точны и приблизительно ортогональны. (Проблема заключается в том, что взаимное уничтожение разрядов, происходящее при переортогонализации почти линейно зависимой системы векторов $\hat{q}_k$, может привести к новым векторам, почти целиком состоящим из ошибок округления.)

В преодолении этих проблем в последние годы наметился известный прогресс [105, 83, 201, 203], и теперь кажется возможным, что обратную итерацию удастся «отремонтировать» таким образом, чтобы всегда получать достаточно точные и ортогональные приближенные собственные векторы, не тратя более чем $O(n)$ флопов на каждый. В этом случае комбинация бисекции с «исправленной» обратной итерацией была бы наиболее предпочтительным алгоритмом во всех случаях, независимо от того, сколько собственных значений и собственных векторов требуется вычислить. Мы надеемся описать такой алгоритм в новом издании книги.

Обратим внимание на удивительную внутреннюю параллельность, присущую бисекции и обратной итерации — ведь каждое собственное значение, а затем и собственный вектор можно вычислить независимо от других. (Такое утверждение предполагает, что поправленная обратная итерация устраняет необходимость в переортогонализации больших групп приближенных собственных векторов.) Это обстоятельство делает названные алгоритмы очень привлекательными для параллельных компьютеров [76].

5.3.5. Метод Якоби

В отличие от уже рассмотренных алгоритмов, метод Якоби не предполагает первоначального приведения к трехдиагональной форме, а работает с исходной плотной матрицей. Обычно он работает много медленнее, чем конкурирующие алгоритмы, и продолжает представлять интерес только по той причине, что иногда способен вычислять малые собственные значения и отвечающие им собственные векторы с гораздо большей точностью, чем ранее описанные методы. Кроме того, он легко распараллеливается. Здесь мы изложим лишь основы теории метода, а обсуждение вопросов, связанных с высокой точностью, отложим до разд. 5.4.3.

По заданной симметричной матрице $A = A_0$ в методе Якоби строится последовательность ортогонально подобных матриц $A_1, A_2, \ldots$. В конечном счете, она сходится к диагональной матрице, на диагонали которой стоят собственные значения. Матрица A_{i+1} получается из A_i по формуле $A_{i+1} = J_i^T A_i J_i$, где J_i — ортогональная матрица, называемая *вращением Якоби*. Таким образом,

$$\begin{aligned} A_m &= J_{m-1}^T A_{m-1} J_{m-1} \\ &= J_{m-1}^T J_{m-2}^T A_{m-2} J_{m-2} J_{m-1} = \cdots \\ &= J_{m-1}^T \cdots J_0^T A_0 J_0 \cdots J_{m-1} \\ &\equiv J^T A J. \end{aligned}$$

При подходящем выборе вращений J_i матрица A_m для больших m будет близка к диагональной матрице Λ. Следовательно, можно написать $\Lambda \approx J^T A J$ или $J \Lambda J^T \approx A$. Поэтому столбцы матрицы J являются приближенными собственными векторами для A.

Матрица J_i выбирается так, чтобы сделать нулями пару внедиагональных элементов матрицы $A_{i+1} = J_i^T A_i J_i$. Именно, угол θ вращения

$$J_i = R(j,k,\theta) \equiv \begin{array}{c} \\ \\ j \\ \\ k \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & j & & k & & & \end{array} \\ \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & \cos\theta & & -\sin\theta & & & \\ & & & & \ddots & & & & \\ & & & \sin\theta & & \cos\theta & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{bmatrix} \end{array}$$

выбирается из условия, чтобы элементы (j,k) и (k,j) в A_{i+1} стали нулями. Чтобы найти θ (фактически определяются $\cos\theta$ и $\sin\theta$), запишем

$$\begin{bmatrix} a_{jj}^{(i+1)} & a_{jk}^{(i+1)} \\ a_{kj}^{(i+1)} & a_{kk}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T \begin{bmatrix} a_{jj}^{(i)} & a_{jk}^{(i)} \\ a_{kj}^{(i)} & a_{kk}^{(i)} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix},$$

где λ_1 и λ_2 — собственные значения подматрицы

$$\begin{bmatrix} a_{jj}^{(i)} & a_{jk}^{(i)} \\ a_{kj}^{(i)} & a_{kk}^{(i)} \end{bmatrix}.$$

Вычислить $\cos\theta$ и $\sin\theta$, для которых мы введем сокращенные обозначения $c \equiv \cos\theta$ и $s \equiv \sin\theta$, несложно. Опуская, для простоты, индекс (i) в приведенном выше матричном равенстве и выполняя перемножение матриц, получаем, с учетом симметрии,

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} a_{jj}c^2 + a_{kk}s^2 + 2sca_{jk} & sc(a_{kk} - a_{jj}) + a_{jk}(c^2 - s^2) \\ sc(a_{kk} - a_{jj}) + a_{jk}(c^2 - s^2) & a_{jj}s^2 + a_{kk}c^2 - 2sca_{jk} \end{bmatrix}.$$

Приравнивая внедиагональный элемент нулю, имеем

$$\frac{a_{jj} - a_{kk}}{2a_{jk}} = \frac{c^2 - s^2}{2sc} = \frac{\cos 2\theta}{\sin 2\theta} = \operatorname{ctg} 2\theta \equiv \tau.$$

Положим $t = \frac{s}{c} = \operatorname{tg}\theta$ и заметим, что $t^2 + 2\tau t - 1 = 0$. Решая квадратное уравнение, находим $t = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}$, $c = \frac{1}{\sqrt{1+t^2}}$ и $s = t \cdot c$. Суммируем этот вывод в виде следующего алгоритма:

Алгоритм 5.5. *Вычислить вращение Якоби в координатной плоскости j,k и применить его к матрице A:*

proc Jacobi-Rotation (A,j,k)
  if $|a_{jk}|$ *не слишком мал*
    $\tau = (a_{jj} - a_{kk})/(2 \cdot a_{jk})$
    $t = \mathrm{sign}(\tau)/(|\tau| + \sqrt{1+\tau^2})$
    $c = 1/\sqrt{1+t^2}$
    $s = c \cdot t$
    $A = R^T(j,k,\theta) \cdot A \cdot R(j,k,\theta)$ ... *где* $c = \cos\theta$ *и* $s = \sin\theta$
    if нужны собственные векторы
      $J = J \cdot R(j,k,\theta)$
    end if
  end if

Стоимость применения вращения $R(j,k,\theta)$ к матрице A (или J) составляет лишь $O(n)$ флопов, так как в A изменяются только строки и столбцы с номерами j и k (а в J только столбцы j и k). Алгоритм Якоби в целом можно описать так:

Алгоритм 5.6. *Метод Якоби для вычисления собственных значений симметричной матрицы:*

repeat
  выбрать пару индексов j,k
  обратиться к процедуре Jacobi-Rotation(A,j,k)
пока A не станет достаточно близка к диагональной матрице

Остается решить, каким образом выбирать пары j,k. Имеется несколько возможностей. Для их описания и в качестве меры продвижения к пределу определим величину

$$\mathrm{off}(A) \equiv \sqrt{\sum_{1 \le j < k \le n} a_{jk}^2}.$$

Таким образом, $\mathrm{off}(A)$ есть норма наддиагональной части матрицы, поэтому A тогда и только тогда будет диагональной матрицей, когда $\mathrm{off}(A) = 0$. Мы хотели бы, чтобы величина $\mathrm{off}(A)$ уменьшалась быстро. Следующая лемма показывает, что $\mathrm{off}(A)$ монотонно убывает с каждым вращением Якоби.

Лемма 5.4. *Пусть* $j \neq k$ *и* A' *обозначает матрицу, полученную в результате обращения к процедуре Jacobi-Rotation(A,j,k). Тогда* $\mathrm{off}^2(A') = \mathrm{off}^2(A) - a_{jk}^2$.

Доказательство. Заметим, что A и A' отличаются только строками и столбцами с номерами j и k. Имеем

$$\mathrm{off}^2(A) = \left(\sum_{\substack{1 \le j' < k' \le n \\ j' \neq j \text{ или } k' \neq k}} a_{j'k'}^2 \right) + a_{jk}^2 \equiv S^2 + a_{jk}^2.$$

Точно так же, $\mathrm{off}^2(A') = S'^2 + a'^2_{jk} = S'^2$, поскольку $a'_{jk} = 0$ в результате применения процедуры Jacobi-Rotation*(A,j,k)*. Используя соотношения $\|X\|_F = \|QX\|_F$ и $\|X\|_F = \|XQ\|_F$, верные для произвольной матрицы X и всякой ортогональной матрицы Q, можно показать, что $S^2 = S'^2$. Следовательно, $\mathrm{off}^2(A') = \mathrm{off}^2(A) - a_{jk}^2$, что и требовалось. □

Опишем исходную версию алгоритма (указанную самим Якоби в 1846 г.). Она сопровождается привлекательным анализом, но, к сожалению, является слишком медленной для практического применения.

Алгоритм 5.7. *Классический алгоритм Якоби:*

while $\mathrm{off}(A) > \mathrm{tol}$ *(здесь* tol *— параметр останова, задаваемый пользователем)*
  выбрать j *и* k *так, чтобы* a_{jk} *был наибольшим по абсолютной величине внедиагональным элементом*
  обратиться к процедуре Jacobi-Rotation (A,j,k)
end while

Теорема 5.11. *Пусть* $N = \frac{n(n-1)}{2}$ *— число наддиагональных элементов матрицы* A. *Тогда для вращения Якоби в классическом алгоритме Якоби выполняется соотношение* $\mathrm{off}(A') \le \sqrt{1 - \frac{1}{N}}\,\mathrm{off}(A)$. *После* k *вращений величина* $\mathrm{off}(\cdot)$ *не превышает* $(1 - \frac{1}{N})^{k/2}\mathrm{off}(A)$.

Доказательство. По лемме 5.4, $\mathrm{off}^2(A') = \mathrm{off}^2(A) - a_{jk}^2$. Поскольку a_{jk} — внедиагональный элемент с наибольшим модулем, то $\mathrm{off}^2(A) \le \frac{n(n-1)}{2} a_{jk}^2$, или иначе $a_{jk}^2 \ge \frac{1}{n(n-1)/2}\mathrm{off}^2(A)$. Поэтому $\mathrm{off}^2(A) - a_{jk}^2 \le (1 - \frac{1}{N})\mathrm{off}^2(A)$, что и требовалось. □

Таким образом, если говорить о величине $\mathrm{off}(A)$, то классический алгоритм Якоби сходится, по меньшей мере, со скоростью геометрической прогрессии, знаменатель которой не превосходит $\sqrt{1 - \frac{1}{N}}$. На самом деле, асимптотически метод сходится квадратично.

Теорема 5.12. *Метод Якоби локально сходится с квадратичной скоростью в том смысле, что для достаточно больших* i *справедливо соотношение*

$$\mathrm{off}(A_{i+N}) = O(\mathrm{off}^2(A_i)).$$

Тем самым, сравниваются состояния метода через N *шагов (что достаточно для того, чтобы выбрать по одному разу каждый элемент* a_{jk}*).*

Классический алгоритм Якоби не используется в практических вычислениях, потому что поиск наибольшего элемента слишком замедляет процесс: нужно провести поиск среди $\frac{n^2-n}{2}$ элементов прежде, чем можно будет выполнить вращение, которое обойдется всего лишь в $O(n)$ флопов. Это означает, что для больших n время поиска преобладает в общем времени алгоритма. Поэтому вместо классического правила выбора пар j, k используется следующий простой метод:

Алгоритм 5.8. *Строчный циклический метод Якоби: построчный циклический обход внедиагональных элементов матрицы* A.

repeat
  for $j = 1$ *to* $n - 1$
    for $k = j + 1$ *to* n
      обратиться к процедуре Jacobi-Rotation(A,j,k)
    end for
  end for
пока A *не станет достаточно близка к диагональной матрице*

Если в течение всего внутреннего цикла процедура Jacobi-Rotation дает лишь значения $c = 1$ и $s = 0$, то матрица A уже более не изменяется. Как и классический метод Якоби, циклический алгоритм асимптотически сходится квадратично [262, с. 270].

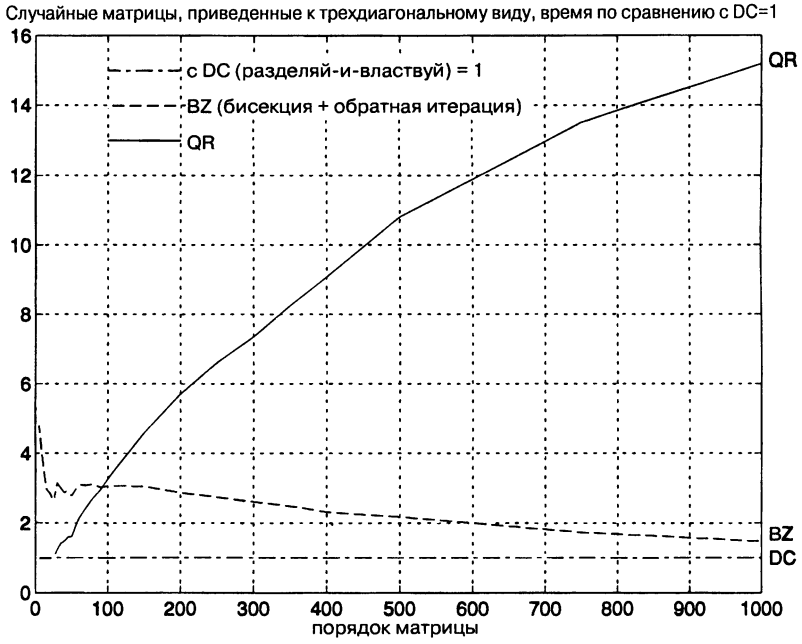
Стоимость одного «цикла» метода Якоби (в котором по одному разу выбирается каждая пара j, k) составляет примерно половину стоимости приведения к трехдиагональной форме и вычисления собственных значений и собственных векторов посредством QR-итерации. Она превышает стоимость алгоритма «разделяй-и-властвуй». Поскольку для сходимости метода Якоби нередко требуется от 5 до 10 циклов, ясно, что он сильно проигрывает конкурирующим алгоритмам.

5.3.6. Сравнение производительности

В этом разделе анализируется производительность трех самых быстрых алгоритмов для симметричной проблемы собственных значений: QR-итерации, бисекции с обратной итерацией и алгоритма «разделяй-и-властвуй». Более подробные сведения можно найти в [10, гл. 3] или NETLIB/lapack/lug/lapack_lug.html.

Начнем обсуждение с наискорейшего метода, а потом сравним с ним другие алгоритмы. Мы пользовались LAPACK-программой `ssyevd`. Если нужны только собственные значения, то программа реализует приведение к трехдиагональному виду, сопровождаемое QR-итерацией; соответствующее число операций равно $\frac{4}{3}n^3 + O(n^2)$. Если же требуются и собственные значения, и собственные векторы, то вслед за приведением к трехдиагональному виду применяется алгоритм «разделяй-и-властвуй». Измерялось время работы программы `ssyevd` на компьютере IBM RS6000/590, представляющем собой рабочую станцию с пиковой производительностью 266 мегафлопов. Оптимизированное матричное умножение выполняется на этой машине со скоростью 233 мегафлопа для матриц порядка 100 и 256 мегафлопов для матриц порядка 1000. В таблице приведены данные для программы `ssyevd`. Графа «скорость в мегафлопах» регистрирует абсолютную скорость программы; в графе «время/время(матричного умножения)» показано отношение времени решения спектральной задачи порядка n к времени перемножения двух квадратных матриц того же порядка. Мы видим, что для достаточно больших матриц вычисление (только) собственных значений симметричной матрицы примерно эквивалентно по стоимости перемножению матриц. (Напротив, для несимметричной матрицы вычисление собственных значений стоило бы, по меньшей мере, в 16 раз дороже [10].) Вычисление и собственных значений, и собственных векторов обходится в три раза дороже, чем умножение матриц.

Порядок	Только собственные значения		Собственные значения и собственные векторы	
	Скорость в мегафлопах	Время/время (матричного умножения)	Скорость в мегафлопах	Время/время (матричного умножения)
100	72	3.1	72	9.3
1000	160	1.1	174	2.8



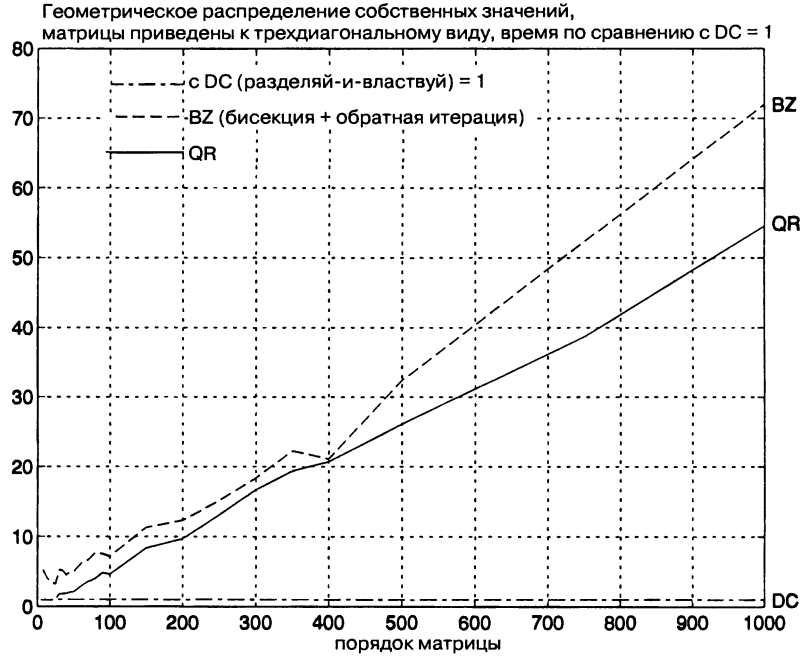
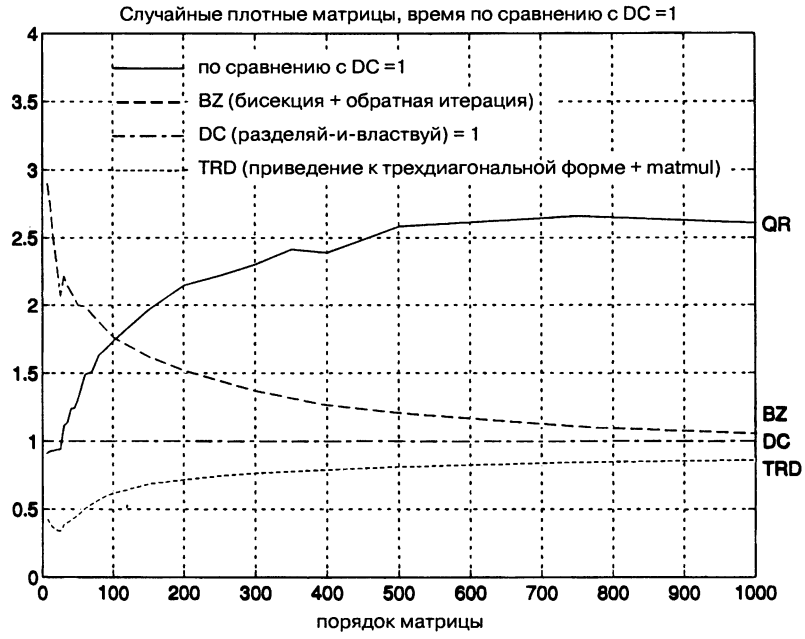


Рис. 5.4. Скорость вычисления собственных значений и собственных векторов симметричной трехдиагональной матрицы по сравнению со скоростью алгоритма «разделяй-и-властвуй».



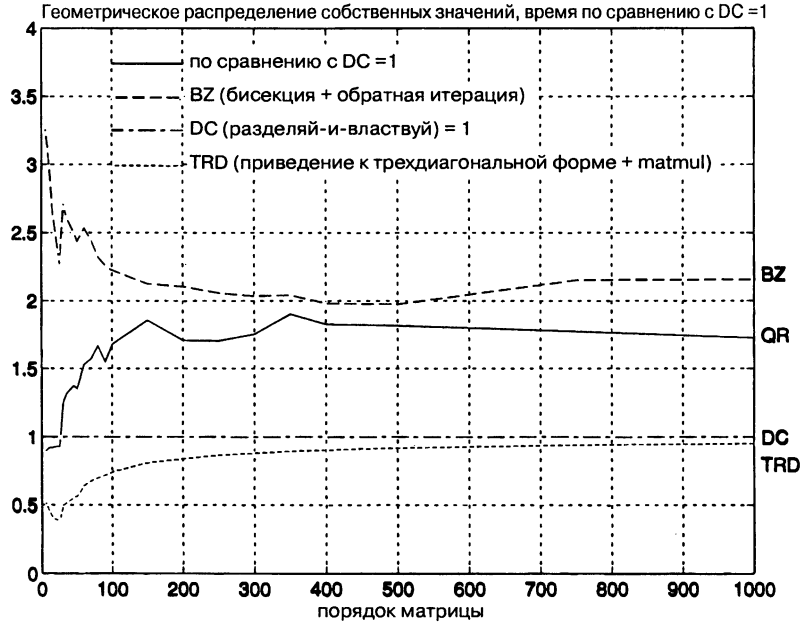


Рис. 5.5. Скорость вычисления собственных значений и собственных векторов симметричной плотной матрицы по сравнению со скоростью алгоритма «разделяй-и-властвуй».

Сравним теперь относительную производительность QR-итерации, бисекции в комбинации с обратной итерацией и алгоритма «разделяй-и-властвуй». На рис. 5.4 и 5.5 эти методы обозначаются соответственно символами QR, BZ (от имени LAPACK-программы `sstebz`, реализующей бисекцию) и DC. На графиках этих рисунков по горизонтальной оси откладывается порядок матрицы, а по вертикальной — отношение времени решения задачи соответствующим методом к времени алгоритма DC. Поэтому самому этому алгоритму соответствует горизонтальная прямая уровня 1, а кривые, помеченные символами BZ и QR, показывают, насколько медленнее эти методы, чем DC. Рис. 5.4 относится к решению трехдиагональной проблемы собственных значений, тогда как рис. 5.5 дает полное время решения для исходной плотной матрицы.

Верхний график на рис. 5.5 соответствует случайным симметричным матрицам. Трехдиагональные матрицы для рис. 5.4 были получены приведением плотных матриц рис. 5.5 к трехдиагональной форме. Случайные матрицы этого типа в среднем имеют хорошо разделенные собственные значения, поэтому обратная итерация (почти) не требует дорогостоящей переортогонализации. В результате, BZ оказывается сравнимым по производительности с DC, тогда как QR значительно медленнее; для трехдиагональных матриц высокого порядка QR проигрывал DC по времени подчас почти в 15 раз.

Для нижних графиков использовались плотные симметричные матрицы с собственными значениями $1, .5, .25, \ldots, 5^{n-1}$. Иными словами, возле нуля находится кластер из многих собственных значений, поэтому в обратной итерации приходится производить большую работу по переортогонализации. Как следствие, для трехдиагональных матриц BZ уступал DC по времени более, чем в 70 раз. QR также был медленнее, чем DC, подчас в 54 раза, что объясняется тем, что, благодаря дефляции, DC в действительности *ускоряется* при наличии большого кластера собственных значений.

Разница скоростей методов QR, BZ и DC на рис. 5.5 менее заметна, чем на рис. 5.4, поскольку здесь в стоимость каждого метода включены расходы на приведение матрицы к трехдиагональной форме и преобразование собственных векторов трехдиагональной матрицы в собственные векторы исходной плотной матрицы. Эти общие накладные расходы составляют $O(n^3)$ операций и помечены на рисунке символом TRD. Близость кривых, соответствующих DC и TRD на рис. 5.5, означает, что любое дальнейшее ускорение DC мало изменит общую скорость алгоритма для плотных матриц.

5.4. Алгоритмы вычисления сингулярного разложения

В теореме 3.3 показано, что сингулярное разложение (SVD) матрицы общего вида G тесно связано со спектральными разложениями симметричных матриц G^TG, GG^T и $\begin{bmatrix} 0 & G^T \\ G & 0 \end{bmatrix}$. Опираясь на эту связь, можно преобразовать алгоритмы предыдущего раздела в алгоритмы вычисления SVD. Это преобразование, однако, выполняется не прямолинейно, поскольку дополнительная структура, которой обладает SVD, часто может быть использована для повышения эффективности и точности алгоритмов [120, 80, 67].

Все алгоритмы вычисления спектрального разложения симметричной матрицы A, за исключением метода Якоби, состоят из следующих этапов:

1. Матрица A приводится к трехдиагональной форме T посредством ортогональной матрицы Q_1, т. е. $A = Q_1 T Q_1^T$.
2. Вычисляется спектральное разложение $T = Q_2 \Lambda Q_2^T$, где Λ — диагональная матрица собственных значений, а Q_2 — ортогональная матрица, столбцами которой являются собственные векторы матрицы T.
3. Разложения этапов 1 и 2 комбинируются с тем, чтобы получить разложение $A = (Q_1 Q_2)\Lambda(Q_1 Q_2)^T$. Столбцы матрицы $Q = Q_1 Q_2$ суть собственные векторы матрицы A.

Из аналогичных этапов состоят алгоритмы вычисления SVD матрицы общего вида G (исключением снова является метод Якоби):

1. Матрица G приводится к двухдиагональной форме B с помощью ортогональных матриц U_1 и V_1, т. е. $G = U_1 B_1 V_1^T$. Двухдиагональность матрицы B означает, что ненулевые элементы в ней могут располагаться только на главной диагонали и первой наддиагонали.
2. Вычисляется сингулярное разложение $B = U_2 \Sigma V_2^T$ матрицы B. Здесь Σ — диагональная матрица сингулярных чисел, а столбцами ортогональных матриц U_2 и V_2 являются соответственно левые и правые сингулярные векторы.
3. Разложения этапов 1 и 2 комбинируются с тем, чтобы получить разложение $G = (U_1 U_2)\Sigma(V_1 V_2)^T$. Столбцы матриц $U = U_1 U_2$ и $V = V_1 V_2$ суть соответственно левые и правые сингулярные векторы матрицы G.

Приведение к двухдиагональной форме выполняется алгоритмом, описанным в разд. 4.4.7. Из обсуждения алгоритма в указанном разделе следует, что вычисление матрицы B стоит $\frac{8}{3}n^3 + O(n^2)$ флопов; только эта матрица и нужна, если требуется найти лишь сингулярные числа. При необходимости вычисления и сингулярных векторов еще $4n^3 + O(n^2)$ флопов затрачиваются для вычисления матриц U_1 и V_1.

Сформулируем простую лемму, показывающую, как свести вычисление SVD двухдиагональной матрицы B к вычислению спектрального разложения симметричной трехдиагональной матрицы T.

Лемма 5.5. *Пусть B — двухдиагональная матрица порядка n с диагональными элементами $a_1, \ldots, a_n$ и наддиагональными элементами $b_1, \ldots, b_{n-1}$. Вычисление SVD матрицы B можно свести к вычислению собственных значений и собственных векторов симметричной трехдиагональной матрицы следующими тремя способами:*

1. *Пусть $A = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$. Обозначим через P матрицу-перестановку $[e_1, e_{n+1}, e_2, e_{n+2}, \ldots, e_n, e_{2n}]$, где e_i есть i-й столбец единичной матрицы порядка $2n$. Тогда $T_{ps} \equiv P^T A P$ является симметричной трехдиагональной матрицей. Индекс ps есть сокращение от perfect shuffle (идеальная тасовка); действительно, умножение вектора x на P «тасует» его компоненты подобно картам в колоде. Можно показать, что в T_{ps} все диагональные элементы равны нулю, а наддиагональными и поддиагональными элементами являются числа $a_1, b_1, a_2, b_2, \ldots, b_{n-1}, a_n$. Пусть (α_i, x_i) —*

собственная пара для T_{ps}, т. е. $T_{ps}x_i = \alpha_i x_i$, причем x_i — нормированный вектор. Тогда $\alpha_i = \pm\sigma_i$, где σ_i — сингулярное число матрицы B, а $Px_i = \frac{1}{\sqrt{2}}\begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$, где u_i и v_i суть соответственно левый и правый (нормированные) сингулярные векторы для B.

2. *Положим $T_{BB^T} \equiv BB^T$. Тогда T_{BB^T} — симметричная трехдиагональная матрица с диагональными элементами $a_1^2 + b_1^2$, $a_2^2 + b_2^2$, ..., $a_{n-1}^2 + b_{n-1}^2$, a_n^2; наддиагональные и поддиагональные элементы равны $a_2 b_1$, $a_3 b_2$, ..., $a_n b_{n-1}$. Сингулярные числа матрицы B суть квадратные корни из собственных значений матрицы T_{BB^T}, а левые сингулярные векторы для B суть собственные векторы для T_{BB^T}.*
3. *Положим $T_{B^TB} \equiv B^TB$. Тогда T_{B^TB} — симметричная трехдиагональная матрица с диагональными элементами a_1^2, $a_2^2 + b_1^2$, $a_3^2 + b_2^2$, ..., $a_n^2 + b_{n-1}^2$; наддиагональные и поддиагональные элементы равны $a_1 b_1$, $a_2 b_2$, ..., $a_{n-1} b_{n-1}$. Сингулярные числа матрицы B суть квадратные корни из собственных значений матрицы T_{B^TB}, а правые сингулярные векторы для B суть собственные векторы для T_{B^TB}. Матрица T_{B^TB} не дает никакой информации о левых сингулярных векторах матрицы B.*

Доказательство леммы вынесено в вопрос 5.19.

Итак, к любой из трехдиагональных матриц леммы 5.5 теоретически можно применить QR-итерацию, алгоритм «разделяй-и-властвуй» или бисекцию в комбинации с обратной итерацией, а затем извлечь из полученного спектрального разложения сингулярные числа и сингулярные векторы (возможно, только левые или только правые). Однако этот простой подход игнорирует особенности сингулярного разложения, что приводит к потерям в скорости и точности. Приведем два аргумента в поддержку этого утверждения.

Во-первых, применение к T_{ps} симметричной трехдиагональной QR-итерации или алгоритма «разделяй-и-властвуй» было бы неэффективным, так как оба алгоритма вычисляют все собственные значения (и, если требуется, собственные векторы) матрицы T_{ps}, тогда как, согласно лемме 5.5, нам нужны лишь неотрицательные собственные значения (и, возможно, соответствующие собственные векторы). Имеются также трудности, связанные с точностью вычисления сингулярных векторов для малых сингулярных чисел.

Во-вторых, явное формирование любой из матриц T_{BB^T} или T_{B^TB} может привести к численной неустойчивости. В малых сингулярных числах матрицы B можно, в действительности, потерять половину верных разрядов. Положим, например, $\eta = \varepsilon/2$, тогда в арифметике с плавающей точкой число $1 + \eta$ будет округляться до 1. Пусть $B = \begin{bmatrix} 1 & 1 \\ 0 & \sqrt{\eta} \end{bmatrix}$; сингулярные числа этой матрицы приближенно равны $\sqrt{2}$ и $\sqrt{\eta/2}$. Матрица $B^TB = \begin{bmatrix} 1 & 1 \\ 1 & 1+\eta \end{bmatrix}$ округляется до точно вырожденной матрицы $T_{B^TB} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. Таким образом, округление числа $1 + \eta$ до 1 изменяет найденное младшее сингулярное число с первоначального значения вблизи $\sqrt{\eta/2} = \sqrt{\varepsilon}/2$ до нуля. Напротив, обратно устойчивый алгоритм не должен изменять сингулярные числа более

чем на $O(\varepsilon)\|B\|_2 = O(\varepsilon)$. Для IEEE-арифметики двойной точности $\varepsilon \approx 10^{-16}$ и $\sqrt{\varepsilon}/2 \approx 10^{-8}$, поэтому ошибка, вызванная формированием матрицы B^TB, в 10^8 раз превышает ошибку единичного округления, что намного больше уровня ошибки в обратно устойчивом алгоритме. Подобная же потеря точности может происходить при явном формировании матрицы T_{BB^T}.

Вследствие неустойчивости, вызываемой вычислением T_{BB^T} или T_{B^TB}, хорошие SVD-алгоритмы работают непосредственно с B или, может быть, с T_{ps}.

Резюмируя, опишем алгоритмы, используемые для практического вычисления SVD.

1. QR-*итерация и ее варианты*. При правильной реализации (см. [104]) — это наиболее быстрый метод отыскания всех сингулярных чисел двухдиагональной матрицы. Более того, все сингулярные числа определяются с высокой относительной точностью в смысле разд. 5.2.1. Это означает, что у всех найденных сингулярных чисел, даже у самых малых, все разряды будут верны. Сопоставим этот факт с тем, что у малых собственных значений, вычисленных трехдиагональной QR-итерацией, верных разрядов может не быть вообще. Если нужны и сингулярные векторы, то применяется другой вариант QR-итерации [80]: для вычисления сингулярных векторов, отвечающих младшим сингулярным числам, QR-итерация используется с нулевым сдвигом; в результате сингулярные векторы вычисляются так точно (а соответствующие сингулярные числа почти столь же точно), как это описано в разд. 5.2.1. Однако такой метод остается скорейшим лишь для малых матриц, порядок n которых не превосходит примерно 25. Он реализован LAPACK-подпрограммой `sbdsqr`.
2. *«Разделяй-и-властвуй»*. В настоящее время это самый быстрый метод отыскания всех сингулярных чисел и сингулярных векторов матриц порядка большего, чем 25. (Реализующая его LAPACK-подпрограмма `sbdsdc` для малых матриц автоматически переходит к выполнению алгоритма `sbdsqr`.) Однако алгоритм «разделяй-и-властвуй» не гарантирует высокой относительной точности для малых сингулярных чисел. Гарантируется лишь граница погрешности того же типа, что и в симметричной проблеме собственных значений: ошибка в сингулярном числе σ_j ограничена величиной $O(\varepsilon)\sigma_1$, а не $O(\varepsilon)\sigma_j$. Для большинства приложений такой точности достаточно.
3. *Бисекция и обратная итерация*. Применяя бисекцию в комбинации с обратной итерацией к матрице T_{ps} из первой части леммы 5.5, можно находить только сингулярные числа из заданного интервала. Этот алгоритм гарантирует высокую относительную точность сингулярных чисел, хотя, как указано в разд. 5.3.4, вычисленные сингулярные векторы могут терять ортогональность.
4. *Метод Якоби*. Сингулярное разложение плотной матрицы G можно найти, применяя метод Якоби из разд. 5.3.5 к матрице GG^T (или G^TG) *неявно*, т. е. так, что ни одна из названных матриц в явном виде не формируется, чем избегается потеря численной устойчивости, связанная с этим формированием. Для некоторых классов матриц G, а именно тех, к которым применима теория относительных возмущений из разд. 5.2.1, можно показать, что сингулярные числа и сингулярные векторы находятся методом Якоби с высокой относительной точностью (в смысле указанного раздела).

В последующих разделах мы даем более подробное описание некоторых из перечисленных алгоритмов. Прежде всего, мы обсуждаем QR-итерацию и ее вариант, называемый dqds (разд. 5.4.1), затем показываем высокую точность как dqds, так и бисекции (разд. 5.4.2), наконец, рассматриваем метод Якоби (разд. 5.4.3). Изложение алгоритма «разделяй-и-властвуй» опущено, поскольку в целом он аналогичен алгоритму, обсуждавшемуся в разд. 5.3.3; относящиеся к нему детали можно найти в [130].

5.4.1. QR-итерация и ее модификации для двухдиагональной SVD-задачи

QR-итерация в применении к вычислению SVD имеет долгую историю модификаций, направленных на достижение возможно большей эффективности и точности. Хороший обзор этого вопроса дан в [200]. Алгоритм, реализованный LAPACK-программой **sbdsqr**, первоначально основывался на [80]. Позднее он был изменен в части, касающейся вычисления только сингулярных чисел. Для этого случая стал использоваться алгоритм из [104], называемый (по причинам, связанным с его историей) dqds[1]. Мы начинаем с описания этого элегантного, быстрого и точного алгоритма.

Выводу формул dqds предпошлем изложение алгоритма, который хронологически предшествовал QR-итерации. Этот алгоритм, называемый LR-итерацией, будет описан для случая симметричных положительно определенных матриц.

Алгоритм 5.9. LR-*итерация: пусть* T_0 — *произвольная симметричная положительно определенная матрица. Строится последовательность симметричных положительно определенных матриц* T_i, *подобных матрице* T_0:

$i = 0$
repeat
  Выбрать сдвиг τ_i^2, *меньший младшего собственного значения матрицы* T_i.
  Вычислить разложение Холесского $T_i - \tau_i^2 I = B_i^T B_i$
    (B_i — *верхняя треугольная матрица с положительной диагональю.*)
  $T_{i+1} = B_i B_i^T + \tau_i^2 I$
  $i = i + 1$
пока не будет достигнута сходимость

По своей структуре, LR-итерация весьма схожа с QR-итерацией: сначала вычисляется разложение текущей матрицы, а затем сомножители перемножаются в обратном порядке, что дает следующую матрицу T_{i+1}. Легко видеть, что T_{i+1} и T_i подобны: $T_{i+1} = B_i B_i^T + \tau_i^2 I = B_i^{-T} B_i^T B_i B_i^T + \tau_i^2 B_i^{-T} B_i^T = B_i^{-T} T_i B_i^T$.

В действительности, можно показать, что при $\tau_i^2 = 0$ два шага LR-итерации порождают ту же самую матрицу T_2, что и один шаг QR-итерации.

[1] dqds — это сокращение от «differential quotient-difference algorithm with shifts» [209] (дифференциальный алгоритм частных и разностей со сдвигами).

Лемма 5.6. *Пусть T_2 — матрица, порождаемая двумя шагами алгоритма 5.9 при $\tau_i^2 = 0$, а T' — матрица, получаемая из T_0 одним шагом QR-итерации (т. е. $QR = T_0$, $T' = RQ$). Тогда $T_2 = T'$.*

Доказательство. Используя симметрию матрицы T_0, разложим T_0^2 в произведение следующими двумя способами. Во-первых, $T_0^2 = T_0^T T_0 = (QR)^T QR = R^T R$. Без ограничения общности, можно считать, что все элементы R_{ii} положительны. Итак, T_0^2 представлена как произведение нижнетреугольной матрицы R^T и (транспонированной) матрицы R; это представление должно совпадать с представлением Холесского в силу единственности последнего. Матрицу T_0 можно факторизовать иначе, а именно $T_0^2 = B_0^T B_0 B_0^T B_0$. Согласно алгоритму 5.9, $T_1 = B_0 B_0^T = B_1^T B_1$, поэтому $T_0^2 = B_0^T B_0 B_0^T B_0 = B_0^T (B_1^T B_1) B_0 = (B_1 B_0)^T (B_1 B_0)$. Снова получено представление T_0^2 произведением нижнетреугольной матрицы $(B_1 B_0)^T$ и транспонированной матрицы; следовательно, оно также должно совпадать с разложением Холесского. Таким образом, единственность разложения Холесского позволяет нам заключить, что $R = B_1 B_0$. Тем самым найдена связь между двумя шагами LR-итерации и одним шагом QR-итерации. Мы используем ее следующим образом: из $T_0 = QR$ следует, что

$$
\begin{aligned}
T' &= RQ = RQ(RR^{-1}) = R(QR)R^{-1} = RT_0R^{-1} \quad \text{поскольку } T_0 = QR \\
&= (B_1B_0)(B_0^TB_0)(B_1B_0)^{-1} \quad \text{поскольку } R = B_1B_0 \text{ и } T_0 = B_0^TB_0 \\
&= B_1B_0B_0^TB_0B_0^{-1}B_1^{-1} = B_1(B_0B_0^T)B_1^{-1} \\
&= B_1(B_1^TB_1)B_1^{-1} \quad \text{поскольку } B_0B_0^T = T_1 = B_1^TB_1 \\
&= B_1B_1^T \\
&= T_2, \quad \text{что и требовалось.}
\end{aligned}
$$

□

Ни алгоритм 5.9, ни лемма 5.6 не требуют, чтобы матрица T_0 была трехдиагональной; достаточно, чтобы она была симметричной и положительно определенной. Используя связь между LR- и QR-итерациями, установленную в лемме 5.6, можно показать, что значительная часть анализа сходимости, проведенного для QR-итерации, переносится на LR-итерацию; однако здесь мы не будем этого делать.

Алгоритм dqds, являющийся нашей целью, математически эквивалентен LR-итерации. Однако он реализуется *не* так, как описано в алгоритме 5.9, поскольку последний предполагает формирование матрицы $T_{i+1} = B_i B_i^T + \tau_i^2 I$ в явном виде. В разд. 5.4 было показано, что такая операция может быть численно неустойчивой. Поэтому матрица B_{i+1} будет вычисляться *непосредственно* из B_i, а промежуточная матрица T_{i+1} не будет формироваться вовсе.

Чтобы упростить обозначения, положим, что B_i имеет диагональные элементы $a_1, \ldots, a_n$ и наддиагональные элементы $b_1, \ldots, b_{n-1}$, а B_{i+1} — диагональные элементы $\hat{a}_1, \ldots, \hat{a}_n$ и наддиагональные элементы $\hat{b}_1, \ldots, \hat{b}_{n-1}$. Условимся, что $b_0 = \hat{b}_0 = b_n = \hat{b}_n = 0$. Запишем соотношение, связывающее B_i и B_{i+1}:

$$B_{i+1}^T B_{i+1} + \tau_{i+1}^2 I = T_{i+1} = B_i B_i^T + \tau_i^2 I. \tag{5.20}$$

Приравнивая элементы (j, j), $j < n$, в левой и правой частях (5.20), имеем

$$\hat{a}_j^2 + \hat{b}_{j-1}^2 + \tau_{i+1}^2 = a_j^2 + b_j^2 + \tau_i^2 \quad \text{или} \quad \hat{a}_j^2 = a_j^2 + b_j^2 - \hat{b}_{j-1}^2 - \delta, \tag{5.21}$$

где $\delta = \tau_{i+1}^2 - \tau_i^2$. Будем считать, что последовательность чисел τ_i^2 сходится к младшему собственному значению матрицы T, монотонно возрастая (тогда все матрицы T_i будут положительно определены и алгоритм является корректным); следовательно, $\delta \geq 0$. Приравнивая теперь квадраты элементов $(j, j+1)$ в левой и правой частях (5.20), находим

$$\hat{a}_j^2 \hat{b}_j^2 = a_{j+1}^2 b_j^2 \quad \text{или} \quad \hat{b}_j^2 = a_{j+1}^2 b_j^2 / \hat{a}_j^2. \tag{5.22}$$

Объединяя равенства (5.21) и (5.22), получаем следующий пока еще не окончательный алгоритм:

for $j = 1$ to $n-1$
  $\hat{a}_j^2 = a_j^2 + b_j^2 - \hat{b}_{j-1}^2 - \delta$
  $\hat{b}_j^2 = b_j^2 \cdot (a_{j+1}^2 / \hat{a}_j^2)$
end for
$\hat{a}_n^2 = a_n^2 - \hat{b}_{n-1}^2 - \delta$

Эта весьма экономичная версия алгоритма содержит в своем внутреннем цикле всего пять операций с плавающей точкой. *Квадраты* элементов матрицы B_i отображаются непосредственно в *квадраты* элементов матрицы B_{i+1}. Извлекать квадратные корни не имеет смысла до самого конца алгоритма. В самом деле, на современных компьютерах извлечение квадратного корня, как и деление, может потребовать в 10–30 раз больше времени, чем сложение, вычитание или умножение, поэтому этих дорогостоящих операций нужно по возможности избегать. Чтобы подчеркнуть, что вычисляются именно квадраты элементов, заменим переменные по формулам $q_j \equiv a_j^2$ и $e_j \equiv b_j^2$. Это приводит к нашему предпоследнему алгоритму qds (название снова связано с историей алгоритма, на которую мы не будем отвлекаться [209]).

Алгоритм 5.10. *Шаг алгоритма qds:*

for $j = 1$ *to* $n-1$
  $\hat{q}_j = q_j + e_j - \hat{e}_{j-1} - \delta$
  $\hat{e}_j = e_j \cdot (q_{j+1} / \hat{q}_j)$
end for
$\hat{q}_n = q_n - \hat{e}_{n-1} - \delta$

Наш окончательный алгоритм dqds выполняет примерно такое же количество работы, что и qds, однако значительно более точен, что будет показано в разд. 5.4.2. Выделим из первой строки алгоритма 5.10 выражение $q_j - \hat{e}_{j-1} - \delta$ и перепишем его следующим образом:

$$\begin{aligned}
d_j &\equiv q_j - \hat{e}_{j-1} - \delta \\
&= q_j - \frac{q_j e_{j-1}}{\hat{q}_{j-1}} - \delta \quad \text{согласно (5.22)} \\
&= q_j \cdot \left[\frac{\hat{q}_{j-1} - e_{j-1}}{\hat{q}_{j-1}} \right] - \delta \\
&= q_j \cdot \left[\frac{q_{j-1} - \hat{e}_{j-2} - \delta}{\hat{q}_{j-1}} \right] - \delta \quad \text{согласно (5.21)} \\
&= \frac{q_j}{\hat{q}_{j-1}} \cdot d_{j-1} - \delta.
\end{aligned}$$

Это позволяет нам записать внутренний цикл алгоритма 5.10 так:

$$\begin{aligned}\hat{q}_j &= d_j + e_j \\ \hat{e}_j &= e_j \cdot (q_{j+1}/\hat{q}_j) \\ d_{j+1} &= d_j \cdot (q_{j+1}/\hat{q}_j) - \delta\end{aligned}$$

Заметим, наконец, что d_{j+1} может быть записано на место d_j и отношение $t = q_{j+1}/\hat{q}_j$ достаточно вычислить один раз. Это приводит нас к окончательному алгоритму dqds.

Алгоритм 5.11. *Шаг алгоритма dqds:*

$d = q_1 - \delta$
for $j = 1$ *to* $n - 1$
  $\hat{q}_j = d + e_j$
  $t = (q_{j+1}/\hat{q}_j)$
  $\hat{e}_j = e_j \cdot t$
  $d = d \cdot t - \delta$
end for
$\hat{q}_n = d$

Во внутреннем цикле алгоритма dqds то же самое число операций с плавающей точкой, что и в алгоритме qds, только одно вычитание заменено умножением. В следующем разделе мы увидим, что это изменение полностью окупает себя, гарантируя высокую относительную точность результатов.

Мы не коснулись двух важных вопросов: выбора сдвига $\delta = \tau_{i+1}^2 - \tau_i^2$ и распознавания сходимости. Их подробное обсуждение можно найти в [104].

5.4.2. Вычисление SVD двухдиагональной матрицы с высокой относительной точностью

Этот раздел, опирающийся на содержание разд. 5.2.1, может быть опущен при первом чтении книги.

Возможность вычисления SVD двухдиагональной матрицы B с высокой относительной точностью (в смысле разд. 5.2.1) основывается на приводимой ниже теореме 5.13. Согласно этой теореме, малые относительные возмущения элементов матрицы B влекут за собой лишь малые относительные возмущения ее сингулярных чисел.

Лемма 5.7. *Пусть B — двухдиагональная матрица с диагональными элементами $a_1, \ldots, a_n$ и наддиагональными элементами $b_1, \ldots, b_{n-1}$. Пусть дана еще одна двухдиагональная матрица $\widehat{B}$ с диагональными элементами $\hat{a}_i = a_i\chi_i$ и наддиагональными элементами $\hat{b}_i = b_i\zeta_i$. Тогда $\widehat{B} = D_1 B D_2$, где*

$$D_1 = \operatorname{diag}\left(\chi_1, \frac{\chi_2\chi_1}{\zeta_1}, \frac{\chi_3\chi_2\chi_1}{\zeta_2\zeta_1}, \ldots, \frac{\chi_n \cdots \chi_1}{\zeta_{n-1}\cdots\zeta_1}\right),$$

$$D_2 = \operatorname{diag}\left(1, \frac{\zeta_1}{\chi_1}, \frac{\zeta_2\zeta_1}{\chi_2\chi_1}, \ldots, \frac{\zeta_{n-1}\cdots\zeta_1}{\chi_{n-1}\cdots\chi_1}\right).$$

Лемма проверяется несложным вычислением (см. вопрос 5.20). Применяя теперь следствие 5.2, приходим к следующему утверждению.

Теорема 5.13. *Пусть B и $\widehat{B}$ — матрицы, определенные в лемме 5.7. Предположим, что найдется число $\tau \geq 1$, такое, что $\tau^{-1} \leq \chi_i \leq \tau$ и $\tau^{-1} \leq \zeta_i \leq \tau$. Иначе говоря, число $\epsilon \equiv \tau - 1$ есть граница для относительного различия произвольного элемента матрицы B и соответствующего элемента матрицы $\widehat{B}$. Пусть $\sigma_n \leq \cdots \leq \sigma_1$ — сингулярные числа матрицы B, а $\hat{\sigma}_n \leq \cdots \leq \hat{\sigma}_1$ — сингулярные числа матрицы $\widehat{B}$. Тогда $|\hat{\sigma}_i - \sigma_i| \leq \sigma_i(\tau^{4n-2} - 1)$. Если $\sigma_i \neq 0$ и $\tau - 1 = \epsilon \ll 1$, то можно написать*

$$\frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i} \leq \tau^{4n-2} - 1 = (4n-2)\epsilon + O(\epsilon^2).$$

Таким образом, относительные возмущения сингулярных чисел, т. е. величины $|\hat{\sigma}_i - \sigma_i|/\sigma_i$, ограничены произведением относительного возмущения ϵ матричных элементов и числа $4n-2$. При небольших дополнительных усилиях множитель $4n-2$ может быть уменьшен до $2n-1$ (см. вопрос 5.21). Можно показать, что и сингулярные векторы определяются весьма точно; погрешность в них будет обратно пропорциональна относительной отделенности (характеристике, введенной в разд. 5.2.1).

Покажем, что сингулярные числа двухдиагональной матрицы можно найти с высокой относительной точностью, пользуясь как бисекцией (алгоритм 5.4 в применении к матрице T_{ps} из леммы 5.5), так и методом dqds (алгоритм 5.11). Начнем с бисекции. Вспомним, что собственными значениями симметричной трехдиагональной матрицы T_{ps} являются сингулярные числа матрицы B и числа, им противоположные. Из леммы 5.3 следует, что инерция матрицы $T_{ps} - \lambda I$, вычисленная с помощью формулы (5.17), есть точная инерция матрицы того же вида, построенной по некоторой матрице $\widehat{B}$, такой, что относительные разности одноименных элементов в $\widehat{B}$ и B не превосходят 2.5ε. Поэтому, согласно теореме 5.13, относительные разности точных и вычисленных сингулярных чисел (т. е. сингулярных чисел матрицы $\widehat{B}$) не превысят величины, примерно равной $(10n-5)\varepsilon$.

Обратимся теперь к алгоритму 5.11. Пользуясь теоремой 5.13, покажем, что сингулярные числа матрицы B (т. е. матрицы на входе алгоритма 5.11) и сингулярные числа матрицы $\widehat{B}$ (матрицы на выходе этого алгоритма) согласуются с высокой относительной точностью. Отсюда будет следовать, что, когда после многих шагов алгоритма dqds матрица $\widehat{B}$ станет почти диагональной и за ее (приближенные) сингулярные числа можно будет принять диагональные элементы, эти элементы будут приближениями высокой относительной точности к сингулярным числам первоначальной матрицы на входе алгоритма.

Проще всего разобраться в ситуации с нулевым сдвигом δ. В этом случае в dqds выполняются только сложения положительных чисел, умножения и деления; взаимного уничтожения верных разрядов нигде не происходит. Упрощая, можно сказать, что всякая последовательность выражений, построенных из указанных основных операций, гарантированно дает высокую относительную точность в каждом вычисленном результате. Итак, $\widehat{B}$ будет вычислена с высокой относительной точностью, а тогда, согласно теореме 5.13, с высокой относительной точностью будут согласованы и сингулярные числа матриц B и $\widehat{B}$.

Общий случай, когда $\delta > 0$, труднее для анализа [104].

Теорема 5.14. *В арифметике с плавающей точкой шаг алгоритма 5.11, применяемого к матрице B и вычисляющего матрицу $\widehat{B}$, эквивалентен следующей последовательности операций:*

1. *Произвести малые относительные возмущения (не превышающие по абсолютной величине 1.5ε) в каждом элементе матрицы B. Пусть $\widetilde{B}$ — полученная матрица.*
2. *Применить к $\widetilde{B}$ шаг алгоритма 5.11 в точной арифметике. Пусть $\check{B}$ — полученная матрица.*
3. *Произвести малые относительные возмущения (не превышающие по абсолютной величине ε) в каждом элементе матрицы $\check{B}$, получив в результате матрицу $\widehat{B}$.*

По теореме 5.13, шаги 1 и 3 влекут за собой лишь малые относительные изменения сингулярных чисел двухдиагональной матрицы. Поэтому сингулярные числа матриц B и $\widehat{B}$ должны быть согласованы с высокой относительной точностью.

Доказательство. Перепишем внутренний цикл алгоритма 5.11, снабдив переменные d и t индексами с тем, чтобы можно было различать их значения на разных итерациях, а также учтя округления посредством членов вида $(1+\epsilon)$, тоже снабженных индексами:

$$\begin{aligned}
\hat{q}_j &= (d_j + e_j)(1+\epsilon_{j,+}) \\
t_j &= (q_{j+1}/\hat{q}_j)(1+\epsilon_{j,/}) \\
\hat{e}_j &= e_j \cdot t_j(1+\epsilon_{j,*1}) \\
d_{j+1} &= (d_j \cdot t_j(1+\epsilon_{j,*2}) - \delta)(1+\epsilon_{j,-}).
\end{aligned}$$

Подставляя выражение для $\hat{q}_j$ из первой строки во вторую, находим

$$t_j = \frac{q_{j+1}}{d_j + e_j} \cdot \frac{1+\epsilon_{j,/}}{1+\epsilon_{j,+}}.$$

Подставляя это выражение для t_j в последнюю строку алгоритма и деля все члены на $1+\epsilon_{j,-}$, имеем

$$\frac{d_{j+1}}{1+\epsilon_{j,-}} = \frac{d_j q_{j+1}}{d_j + e_j} \cdot \frac{(1+\epsilon_{j,/})(1+\epsilon_{j,*2})}{1+\epsilon_{j,+}} - \delta. \tag{5.23}$$

Отсюда видно, как следует определить матрицу $\widetilde{B}$. Положим

$$\begin{aligned}
\tilde{d}_{j+1} &= \frac{d_{j+1}}{1+\epsilon_{j,-}}, \\
\tilde{e}_{j+1} &= \frac{e_j}{1+\epsilon_{j-1,-}}, \\
\tilde{q}_{j+1} &= q_{j+1}\frac{(1+\epsilon_{j,/})(1+\epsilon_{j,*2})}{1+\epsilon_{j,+}}.
\end{aligned} \tag{5.24}$$

Тогда (5.23) принимает вид

$$\tilde{d}_{j+1} = \frac{\tilde{d}_j \tilde{q}_{j+1}}{\tilde{d}_j + \tilde{e}_j} - \delta.$$

Формулы (5.24) показывают, что наибольшее относительное изменение в элементах $\widetilde{B}$ по сравнению с B не превышает 1.5ε (поскольку в выражении для $\tilde{q}_{j+1} = \tilde{a}^2_{j+1,j+1}$ присутствуют три множителя вида $1+\epsilon$).

Определим теперь величины $\check{q}_j$ и $\check{e}_j$ для матрицы $\check{B}$ формулами

$$\begin{aligned}
\check{q}_j &= \tilde{d}_j + \tilde{e}_j \\
\tilde{t}_j &= (\tilde{q}_{j+1}/\check{q}_j) \\
\check{e}_j &= \tilde{e}_j \cdot \tilde{t}_j \\
\tilde{d}_{j+1} &= \tilde{d}_j \cdot \tilde{t}_j - \delta.
\end{aligned}$$

Они описывают шаг алгоритма dqds, применяемого к матрице $\widetilde{B}$ в точной арифметике; результатом является матрица $\check{B}$. Остается показать, что $\check{B}$ отличается от $\widehat{B}$ относительным изменением каждого элемента, не превосходящим по абсолютной величине ε. Это следует из представлений

$$\begin{aligned}
\check{q}_j &= \tilde{d}_j + \tilde{e}_j \\
&= \frac{d_j}{1+\epsilon_{j-1,-}} + \frac{e_j}{1+\epsilon_{j-1,-}} \\
&= (d_j + e_j)(1+\epsilon_{j,+}) \cdot \frac{1}{(1+\epsilon_{j,+})(1+\epsilon_{j-1,-})} \\
&= \hat{q}_j \cdot \left[\frac{1}{(1+\epsilon_{j,+})(1+\epsilon_{j-1,-})}\right], \\
\check{e}_j &= \tilde{e}_j \cdot \tilde{t}_j \\
&= \frac{e_j}{1+\epsilon_{j-1,-}} \cdot \frac{\tilde{q}_{j+1}}{\check{q}_j} \\
&= \frac{e_j}{1+\epsilon_{j-1,-}} \cdot t_j(1+\epsilon_{j,*2})(1+\epsilon_{j-1,-}) \\
&= e_j t_j (1+\epsilon_{j,*1}) \frac{1+\epsilon_{j,*2}}{1+\epsilon_{j,*1}} \\
&= \hat{e}_j \cdot \left[\frac{1+\epsilon_{j,*2}}{1+\epsilon_{j,*1}}\right].
\end{aligned}$$

□

5.4.3. Метод Якоби для вычисления SVD

При обсуждении метода Якоби в разд. 5.3.5 было отмечено, что этот метод является самым медленным из имеющихся алгоритмов вычисления собственных значений и собственных векторов плотной симметричной матрицы A. В этом разделе будет показано, как с помощью метода Якоби найти сингулярное разложение плотной матрицы G; именно, алгоритм 5.8 из разд. 5.3.5 нужно *неявно* применить к симметричной матрице $A = G^T G$. Отсюда следует, что сходимость этого метода вычисления SVD примерно такая же, как алгоритма 5.8; в частности, и для задачи вычисления SVD метод Якоби является самым медленным из имеющихся.

Тем не менее, интерес к методу Якоби сохраняется, потому что для некоторых типов матриц G он способен вычислять сингулярные числа и сингуляр-

ные векторы намного точнее, чем другие обсуждавшиеся нами методы. Для этих матриц G метод Якоби обеспечивает высокую относительную точность (в смысле разд. 5.2.1) вычисленных приближений к сингулярным числам и сингулярным векторам.

После описания неявного метода Якоби мы покажем, что он вычисляет SVD матрицы G с высокой относительной точностью, если G может быть представлена в виде $G = DX$, где D — диагональная матрица, а X хорошо обусловлена. (Наличие такого представления означает, что G плохо обусловлена в том и только в том случае, когда матрица D имеет и большие, и малые диагональные элементы.) Более общо, мы получаем выигрыш всякий раз, когда X обусловлена значительно лучше, чем G. Мы проиллюстрируем это положение с помощью матрицы, для которой всякий алгоритм, включающий в себя приведение к двухдиагональной форме, необходимо теряет все верные разряды во всех сингулярных числах, кроме старшего; в то же время, метод Якоби определяет все сингулярные числа этой матрицы с полной машинной точностью. Затем мы дадим обзор других классов матриц G с тем же свойством, т. е. метод Якоби для них значительно точнее, чем методы, опирающиеся на двухдиагонализацию.

Заметим, что в разд. 5.4.2 было показано: SVD двухдиагональной матрицы G можно вычислить с высокой относительной точностью, применяя бисекцию или алгоритм dqds (см. разд. 5.4.1). Проблема состоит в том, что преобразование плотной матрицы в двухдиагональную форму может сопровождаться ошибками, достаточными для потери высокой относительной точности; именно это покажет наш пример. Поскольку в методе Якоби заданная матрица обрабатывается без предварительного приведения к двухдиагональному виду, метод имеет гораздо больше шансов достичь высокой относительной точности.

Неявный метод Якоби математически эквивалентен применению алгоритма 5.8 к матрице $A = G^TG$. Иначе говоря, на каждом шаге вычисляется вращение Якоби J, с помощью которого матрица G^TG неявно пересчитывается в J^TG^TGJ; вращение выбрано так, чтобы пара внедиагональных элементов из G^TG обратилась в нули в матрице J^TG^TGJ. Однако ни G^TG, ни J^TG^TGJ не вычисляются в явном виде; вместо них вычисляется матрица GJ. По этой причине мы называем алгоритм *методом односторонних вращений*.

Алгоритм 5.12. *Вычислить вращение Якоби в координатной плоскости j, k и применить его (односторонне) к матрице G:*

proc One-Sided-Jacobi-Rotation (G, j, k)
  Вычислить $a_{jj} = (G^TG)_{jj}$, $a_{jk} = (G^TG)_{jk}$ *и* $a_{kk} = (G^TG)_{kk}$
  if $|a_{jk}|$ *не слишком мал*
    $\tau = (a_{jj} - a_{kk})/(2 \cdot a_{jk})$
    $t = \mathrm{sign}(\tau)/(|\tau| + \sqrt{1+\tau^2})$
    $c = 1/\sqrt{1+t^2}$
    $s = c \cdot t$
    $G = G \cdot R(j,k,\theta)$ *... где* $c = \cos\theta$ *и* $s = \sin\theta$
    if нужны правые сингулярные векторы
      $J = J \cdot R(j,k,\theta)$
    end if
  end if

Заметим, что в данной процедуре вначале определяются элементы (j,j), (j,k) и (k,k) матрицы A, после чего вращение Якоби вычисляется таким же образом, как в процедуре Jacobi-Rotation (алгоритм 5.5).

Алгоритм 5.13. *Метод односторонних вращений: пусть G — матрица размера $n \times n$. Алгоритм вычисляет сингулярные числа σ_i, матрицу U левых сингулярных векторов и матрицу V правых сингулярных векторов. Если положить $\Sigma = diag(\sigma_i)$, то $G = U\Sigma V^T$.*

repeat
  for $j = 1$ *to* $n-1$
    for $k = j+1$ *to* n
      обратиться к процедуре One-Sided-Jacobi-Rotation(G,j,k)
    end for
  end for
пока G^TG не станет достаточно близка к диагональной матрице
Положить $\sigma_i = \|G(:,i)\|_2$ *(2-норма i-го столбца в G)*
Положить $U = [u_1,\ldots,u_n]$, *где* $u_i = G(:,i)/\sigma_i$
Положить $V = J$, где J — накопленное произведение вращений Якоби

В вопросе 5.22 предлагается доказать, что матрицы Σ, U и V, вычисляемые методом односторонних вращений, действительно дают SVD матрицы G.

Мы покажем сейчас, что, несмотря на округления, метод односторонних вращений способен вычислять SVD с высокой относительной точностью, если матрица G может быть представлена в виде $G = DX$, где D — диагональная матрица, а X хорошо обусловлена.

Теорема 5.15. *Пусть $G = DX$ — матрица порядка n, причем D и X невырожденны и D — диагональная матрица. Пусть $\widehat{G}$ — матрица, полученная из G в результате m-кратного обращения к процедуре One-Sided-Jacobi-Rotation(G,j,k) в арифметике с плавающей точкой. Обозначим через $\sigma_1 \geq \cdots \geq \sigma_n$ и $\hat{\sigma}_1 \geq \cdots \geq \hat{\sigma}_n$ сингулярные числа матриц G и $\widehat{G}$. Тогда*

$$\frac{|\sigma_i - \hat{\sigma}_i|}{\sigma_i} \leq O(m\varepsilon)\kappa(X), \tag{5.25}$$

где $\kappa(X) = \|X\| \cdot \|X^{-1}\|$ есть число обусловленности матрицы X. Другими словами, относительная погрешность вычисленных сингулярных чисел мала, если мало число обусловленности матрицы X.

Доказательство. Рассмотрим вначале случай $m = 1$, т. е. результат применения одного вращения Якоби, а затем обобщим наш анализ на случай произвольного m.

Изучение процедуры One-Sided-Jacobi-Rotation(G,j,k) показывает, что $\widehat{G} = \mathrm{fl}(G \cdot \widetilde{R})$, где $\widetilde{R}$ — вращение Гивенса, вычисленное в арифметике с плавающей точкой. По построению, $\widetilde{R}$ отличается от некоторого точного вращения R матрицей с нормой $O(\varepsilon)$. (Неважно (и даже не обязательно верно), чтобы $\widetilde{R}$ отличалась на $O(\varepsilon)$ от «подлинного» вращения Якоби, которое процедура One-Sided-Jacobi-Rotation(G,j,k) вычислила бы в точной арифметике. Необходимо лишь, чтобы $\widetilde{R}$ отличалась на $O(\varepsilon)$ от *некоторого* вращения. Это требование сводится к соотношению $c^2 + s^2 = 1 + O(\varepsilon)$, которое легко проверить.)

Ниже мы покажем, что $\widehat{G} = GR(I+E)$ для некоторой матрицы E с малой нормой: $\|E\|_2 = O(\varepsilon)\kappa(X)$. Если бы E была нулевой матрицей, то $\widehat{G}$ и GR имели бы одни и те же сингулярные числа, поскольку R — (точно) ортогональная матрица. Если норма (ненулевой) матрицы E меньше единицы, то относительные разности сингулярных чисел можно оценить с помощью следствия 5.2:

$$\frac{|\sigma_i - \hat{\sigma}_i|}{\sigma_i} \le \|(I+E)(I+E)^T - I\|_2 = \|E + E^T + EE^T\|_2 \le 3\|E\|_2 = O(\varepsilon)\kappa(X), \tag{5.26}$$

что и требовалось.

Построим требуемую матрицу E. Поскольку $\widetilde{R}$ является для G правым множителем, каждая строка в $\widehat{G}$ зависит лишь от соответствующей строки в G; в обозначениях Matlab'a, это можно записать так: $\widehat{G}(i,:) = \mathrm{fl}(G(i,:)\cdot\widetilde{R})$. Положим $F = \widehat{G} - GR$. Используя лемму 3.1 и соотношение $G = DX$, имеем

$$\|F(i,:)\|_2 = \|\widehat{G}(i,:) - G(i,:)R\|_2 = O(\varepsilon)\|G(i,:)\|_2 = O(\varepsilon)\|d_{ii}X(i,:)\|_2,$$

откуда $\|d_{ii}^{-1}F(i,:)\|_2 = O(\varepsilon)\|X(i,:)\|_2$ или $\|D^{-1}F\|_2 = O(\varepsilon)\|X\|_2$. Учитывая соотношения $R^{-1} = R^T$ и $G^{-1} = (DX)^{-1} = X^{-1}D^{-1}$, получаем

$$\widehat{G} = GR + F = GR(I + R^TG^{-1}F) = GR(I + R^TX^{-1}D^{-1}F) \equiv GR(I+E),$$

где

$$\|E\|_2 \le \|R^T\|_2\|X^{-1}\|_2\|D^{-1}F\|_2 = O(\varepsilon)\|X\|_2\|X^{-1}\|_2 = O(\varepsilon)\kappa(X).$$

Именно это и требовалось.

Остается распространить этот результат на m вращений, где $m > 1$. Заметим, что в точной арифметике мы имели бы $\widehat{G} = GR = DXR = D\widehat{X}$, где $\kappa(\widehat{X}) = \kappa(X)$. В этом случае к каждому из m шагов была бы применима оценка (5.26), что влекло бы за собой (5.25). Вследствие округлений, $\kappa(\widehat{X})$ может приобретать на каждом шаге множитель, не превосходящий $\kappa(I+E) \le 1 + O(\varepsilon)\kappa(X)$. Эти множители, очень близкие к 1, учитываются символом $O(m\varepsilon)$ в оценке (5.25). □

Чтобы завершить описание алгоритма, нужно указать критерий останова, т. е. прояснить смысл условия «если $|a_{jk}|$ не слишком мал» в алгоритме 5.12 (т. е. в процедуре One-Sided-Jacobi-Rotation). Подходящим критерием является условие

$$|a_{jk}| \ge \varepsilon\sqrt{a_{jj}a_{kk}}.$$

Дальнейшее его обсуждение дается в вопросе 5.24.

Пример 5.9. Рассмотрим, в известном смысле, экстремальный пример матрицы $G = DX$, все сингулярные числа которой вычисляются методом Якоби с полной машинной точностью. В то же время, любой метод, опирающийся на приведение к двухдиагональному виду, вычисляет с полной машинной точностью лишь наибольшее сингулярное число $\sqrt{3}$; во всех остальных приближенных сингулярных числах верных разрядов нет вовсе (хотя их абсолютные погрешности суть величины вида $\pm O(\varepsilon)\cdot\sqrt{3}$, как и следует ожидать от обратно устойчивого алгоритма). Для нашего примера $\varepsilon = 2^{-53} \approx 10^{-16}$ (двойная

точность IEEE-арифметики), а $\eta = 10^{-20}$ (в действительности, годится почти любое значение $\eta < \varepsilon$). Положим

$$G \equiv \begin{bmatrix} \eta & 1 & 1 & 1 \\ \eta & \eta & 0 & 0 \\ \eta & 0 & \eta & 0 \\ \eta & 0 & 0 & \eta \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & \eta & & \\ & & \eta & \\ & & & \eta \end{bmatrix} \cdot \begin{bmatrix} \eta & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \equiv D \cdot X.$$

С точностью до, по меньшей мере, 16 разрядов, сингулярными числами матрицы G являются 3, $\sqrt{3} \cdot \eta$, η и η. Чтобы убедиться в том, что при приведении G к двухдиагональной форме происходит потеря точности, достаточно рассмотреть первый шаг алгоритма из разд. 4.4.7. Этот шаг состоит в левом умножении на отражение с целью аннулировать элементы 2, 3 и 4 столбца 1. В точной арифметике G имела бы вид

$$\begin{bmatrix} -2\eta & -.5 - \frac{\eta}{2} & -.5 - \frac{\eta}{2} & -.5 - \frac{\eta}{2} \\ 0 & -.5 + \frac{5\eta}{6} & -.5 - \frac{\eta}{6} & -.5 - \frac{\eta}{6} \\ 0 & -.5 - \frac{\eta}{6} & -.5 + \frac{5\eta}{6} & -.5 - \frac{\eta}{6} \\ 0 & -.5 - \frac{\eta}{6} & -.5 - \frac{\eta}{6} & -.5 + \frac{5\eta}{6} \end{bmatrix}.$$

Однако, вследствие малости числа η, округления приводят к матрице

$$G_1 = \begin{bmatrix} -2\eta & -.5 & -.5 & -.5 \\ 0 & -.5 & -.5 & -.5 \\ 0 & -.5 & -.5 & -.5 \\ 0 & -.5 & -.5 & -.5 \end{bmatrix}.$$

Обратим внимание на то, что в трех последних столбцах матрицы G_1 «потеряна» вся информация об η. Поскольку эти столбцы совпадают, G_1 (точно) вырождена; в действительности, ее ранг равен 2. Таким образом, два младших сингулярных числа изменили свои значения с η на 0, т. е., в относительном смысле, точность в них полностью потеряна. При отсутствии новых округлений мы привели бы G_1 к двухдиагональной матрице

$$B = \begin{bmatrix} -2\eta & \sqrt{.75} & & \\ & 1.5 & 0 & \\ & & 0 & 0 \\ & & & 0 \end{bmatrix},$$

имеющей сингулярные числа $\sqrt{3}$, $\sqrt{3} \cdot \eta$, 0 и 0; первые два из них суть точные сингулярные числа матрицы G. Так как, однако, при приведении G_1 к двухдиагональной форме округления происходят, в нулевые элементы матрицы B будут внесены ненулевые возмущения порядка $O(\varepsilon)$; в результате неточными станут все три младших сингулярных числа. Два наименьших ненулевых сингулярных числа, найденных алгоритмом, суть величины порядка ε и полностью состоят из ошибок округлений.

Метод односторонних вращений не испытывает никаких трудностей с этой матрицей. За три цикла вычисляется разложение $G = U\Sigma V^T$, где, с точностью до машинного эпсилон,

$$U = \begin{bmatrix} 0 & -\frac{\eta^2}{\sqrt{2}} & 1 & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{\eta}{3} & \frac{-1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{\eta}{3} & \frac{-1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{\eta}{\sqrt{2}} & \frac{\eta}{3} & \frac{2}{\sqrt{6}} \end{bmatrix}, \quad V = \begin{bmatrix} 1 & \frac{\eta}{\sqrt{2}} & \frac{\eta}{\sqrt{3}} & \frac{-\eta}{\sqrt{6}} \\ -\eta & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} \\ 0 & \frac{-\eta^3}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} \end{bmatrix}$$

и $\Sigma = \mathrm{diag}(\sqrt{3}\eta, \eta, \sqrt{3}, \eta)$. (Метод Якоби сам по себе не упорядочивает сингулярные числа по величине; такое упорядочение может быть пристроено к нему в качестве заключительного этапа.) ◇

Приведем другие примеры ситуаций, где может быть показано: метод Якоби или его варианты гарантируют высокую относительную точность вычисленного SVD (или спектрального разложения симметричной матрицы), тогда как методы, использующие приведение к двухдиагональной (или трехдиагональной) форме, могут терять все значащие разряды наименьших сингулярных чисел (или собственных значений). Еще ряд примеров можно найти в [75].

1. Пусть $A = LL^T$ — разложение Холесского симметричной положительно определенной матрицы A. Тогда SVD матрицы L, т. е. $L = U\Sigma V^T$, дает спектральное разложение для A: $A = U\Sigma^2 U^T$. Если $L = DX$, где X хорошо обусловлена, а D — диагональная матрица, то, согласно теореме 5.15, сингулярные числа σ_i матрицы L могут быть вычислены методом Якоби с высокой относительной точностью, причем относительные погрешности ограничены величинами $O(\varepsilon)\kappa(X)$. Нужно еще учесть округления, произведенные при вычислении множителя Холесского L. Используя оценку (2.16) и теорему 5.6, можно дать оценку $O(\varepsilon)\kappa^2(X)$ для относительных погрешностей в сингулярных числах, появившихся вследствие округлений в алгоритме Холесского. Таким образом, если X хорошо обусловлена, то все собственные значения матрицы A будут вычислены с высокой относительной точностью (см. вопрос 5.23, а также [82, 92, 183]).

Пример 5.10. Этот пример, подобно примеру 5.9, иллюстрирует экстремальную ситуацию, где всякий алгоритм, опирающийся на предварительное приведение матрицы A к трехдиагональной форме, полностью теряет относительную точность в наименьшем собственном значении; в то же время, применяя алгоритм Холесского, а затем метод односторонних вращений к полученному множителю Холесского, можно вычислить все собственные значения с почти полной машинной точностью. Как и в предыдущем примере, положим $\eta = 10^{-20}$ (в действительности, годится любое $\eta < \varepsilon/120$) и пусть

$$A = \begin{bmatrix} 1 & \sqrt{\eta} & \sqrt{\eta} \\ \sqrt{\eta} & 1 & 10\eta \\ \sqrt{\eta} & 10\eta & 100\eta \end{bmatrix} = \begin{bmatrix} 1 & 10^{-10} & 10^{-10} \\ 10^{-10} & 1 & 10^{-19} \\ 10^{-10} & 10^{-19} & 10^{-18} \end{bmatrix}.$$

Если приведение A к трехдиагональной форме выполняется точно, то будет получена матрица

$$T = \begin{bmatrix} 1 & \sqrt{2\eta} & \\ \sqrt{2\eta} & .5 + 60\eta & .5 - 50\eta \\ & .5 - 50\eta & .5 + 40\eta \end{bmatrix}.$$

Однако, вследствие малости числа η, округления приводят к матрице

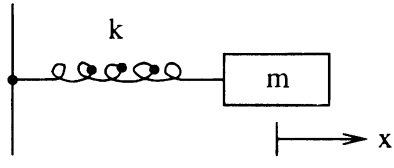
$$\widehat{T} = \begin{bmatrix} 1 & \sqrt{2\eta} & \\ \sqrt{2\eta} & .5 & .5 \\ & .5 & .5 \end{bmatrix},$$

которая не будет даже положительно определенной, так как 2×2-подматрица в ее правом нижнем углу (точно) вырожденна. Таким образом, наименьшее собственное значение матрицы $\widehat{T}$ неположительно, т. е. при приведении к трехдиагональной форме относительная точность в наименьшем собственном значении была полностью утеряна. Напротив, метод односторонних вращений не встречает затруднений при вычислении квадратных корней из собственных значений матрицы A, а именно, чисел $1 + \sqrt{\eta} = 1 + 10^{-10}$, $1 - \sqrt{\eta} = 1 - 10^{-10}$ и $0.99\eta = 0.99 \cdot 10^{-20}$. Найденные приближения имеют почти полную машинную точность. ◇

2. Опишем наиболее общую ситуацию, в которой мы понимаем, как следует вычислять SVD матрицы A с высокой относительной точностью: требуется, чтобы мы могли с хорошей точностью найти *произвольное* разложение вида $A = YDX$, где матрицы X и Y хорошо обусловлены, но в остальном произвольны, а D — диагональная матрица. В предыдущем примере мы имели $L = DX$, т. е. Y была единичной матрицей. Другим источником подходящих разложений является гауссово исключение с полным выбором главного элемента (где Y — нижняя, а X — верхняя треугольные матрицы). Подробнее об этом см. в [74]. Приложения этой техники к симметричным незнакоопределенным задачам на собственные значения можно найти в [228, 250], а к обобщенной симметричной проблеме собственных значений — в [66, 92].

5.5. Дифференциальные уравнения и задачи на собственные значения

Мотивацией для данного раздела являются физические законы сохранения. Снова обратимся к связанной системе материальных точек, введенной в примере 4.1 и затем рассмотренной в примере 5.1. Начнем с простейшего случая одной пружины и одной массы, предполагая, что трение отсутствует.



Пусть x — горизонтальное смещение от положения равновесия. Тогда закон Ньютона $F = ma$ может быть записан уравнением $m\ddot{x}(t) +$

$kx(t) = 0$. Положим $E(t) = \frac{1}{2}m\dot{x}^2(t) + \frac{1}{2}kx^2(t)$ = «кинетическая энергия» + «потенциальная энергия». Закон сохранения энергии говорит нам, что производная $\frac{d}{dt}E(t)$ должна быть равна нулю. Мы можем проверить, что это действительно так: $\frac{d}{dt}E(t) = m\dot{x}(t)\ddot{x}(t) + kx(t)\dot{x}(t) = \dot{x}(t)(m\ddot{x}(t) + kx(t)) = 0$, как и утверждалось.

В более общем случае мы имеем уравнение $M\ddot{x}(t) + Kx(t) = 0$, где M — *матрица масс*, а K — *матрица жесткости*. Энергия определяется формулой $E(t) = \frac{1}{2}\dot{x}^T(t)M\dot{x}(t) + \frac{1}{2}x^T(t)Kx(t)$. То, что такое определение корректно, подтверждается проверкой сохранения этой величины:

$$\begin{aligned}\frac{d}{dt}E(t) &= \frac{d}{dt}\left(\frac{1}{2}\dot{x}^T(t)M\dot{x}(t) + \frac{1}{2}x^T(t)Kx(t)\right)\\ &= \frac{1}{2}(\ddot{x}^T(t)M\dot{x}(t) + \dot{x}^T(t)M\ddot{x}(t) + \dot{x}^T(t)Kx(t) + x^T(t)K\dot{x}(t))\\ &= \dot{x}^T(t)M\ddot{x}(t) + \dot{x}^T(t)Kx(t)\\ &= \dot{x}^T(t)(M\ddot{x}(t) + Kx(t)) = 0.\end{aligned}$$

В этих выкладках была использована симметрия матриц M и K.

Дифференциальное уравнение $M\ddot{x}(t) + Kx(t) = 0$ линейно. Замечательно, что и некоторые нелинейные дифференциальные уравнения сохраняют величины, имеющие смысл «энергии».

5.5.1. Решетка Тода

Чтобы упростить обозначения, будем писать $\dot{x}$ вместо $\dot{x}(t)$, если аргумент известен из контекста.

Решетка Тода — это тоже связанная система материальных точек, только воздействие, оказываемое каждой пружиной, вместо того, чтобы линейно зависеть от растяжения, является экспоненциально убывающей функцией от него:

$$\ddot{x}_i = e^{-(x_i - x_{i-1})} - e^{-(x_{i+1} - x_i)}.$$

В качестве граничных условий возьмем $e^{-(x_1 - x_0)} = 0$ (т. е. $x_0 = -\infty$) и $e^{-(x_{n+1} - x_n)} = 0$ (т. е. $x_{n+1} = +\infty$). Проще говоря, эти условия означают, что ни слева, ни справа от системы нет стенок (см. рис. 4.1).

Сделаем теперь замену переменных $b_k = \frac{1}{2}e^{(x_k - x_{k+1})/2}$ и $a_k = -\frac{1}{2}\dot{x}_k$. Это приводит к дифференциальным уравнениям

$$\begin{aligned}\dot{b}_k &= \frac{1}{2}e^{(x_k - x_{k+1})/2} \cdot \frac{1}{2}(\dot{x}_k - \dot{x}_{k+1}) = b_k(a_{k+1} - a_k),\\ \dot{a}_k &= -\frac{1}{2}\ddot{x}_k = 2(b_k^2 - b_{k-1}^2),\end{aligned} \tag{5.27}$$

где следует считать $b_0 \equiv 0$ и $b_n \equiv 0$. Введем две трехдиагональные матрицы

$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} 0 & b_1 & & \\ -b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & -b_{n-1} & 0 \end{bmatrix}.$$

Заметим, что $B = -B^T$. Легко проверить, что система уравнений (5.27) может быть записана в виде $\frac{dT}{dt} = BT - TB$. Это уравнение называется *потоком Тода*.

Теорема 5.16. *Матрица $T(t)$ при всех t имеет те же собственные значения, что и матрица $T(0)$. Иначе говоря, подобно «энергии», собственные значения сохраняются данным дифференциальным уравнением.*

Доказательство. Рассмотрим задачу $\frac{d}{dt}U = BU$, $U(0) = I$. Мы утверждаем, что матрица $U(t)$ ортогональна для всех t. Поскольку $U^TU(0) = I$, достаточно показать, что $\frac{d}{dt}U^TU = 0$. Имеем

$$\frac{d}{dt}U^TU = \dot{U}^TU + U^T\dot{U} = U^TB^TU + U^TBU = -U^TBU + U^TBU = 0.$$

Здесь была использована косая симметрия матрицы B.

Теперь мы покажем, что матрица $T(t) = U(t)T(0)U^T(t)$ удовлетворяет уравнению Тода $\frac{dT}{dt} = BT - TB$. Отсюда будет следовать, что при любом t матрица $T(t)$ потока Тода ортогонально подобна матрице $T(0)$, а потому имеет те же собственные значения. Имеем

$$\begin{aligned}\frac{d}{dt}T(t) &= \dot{U}(t)T(0)U^T(t) + U(t)T(0)\dot{U}^T(t)\\ &= B(t)U(t)T(0)U^T(t) + U(t)T(0)U^T(t)B^T(t)\\ &= B(t)T(t) - T(t)B(t),\end{aligned}$$

что и требовалось. □

Единственным свойством матрицы B, использовавшимся в этих выкладках, была косая симметрия. Поэтому если $\frac{dT}{dt} = BT - TB$ и $B^T = -B$, то матрица $T(t)$ имеет одни и те же собственные значения для всех t.

Теорема 5.17. *При $t \to +\infty$ или $t \to -\infty$ матрица $T(t)$ сходится к диагональной матрице, на диагонали которой находятся ее собственные значения.*

Доказательство. Мы хотим показать, что $b_i \to 0$ при $t \to \pm\infty$. Для этого вначале покажем, что $\int_{-\infty}^{\infty} \sum_{i=1}^{n-1} b_i^2(t)\,dt < \infty$. Используя индукцию, докажем, что $\int_{-\infty}^{\infty}(b_j^2(t) + b_{n-j}^2(t))\,dt < \infty$, и затем сложим такие неравенства для всех j. При $j = 0$ имеем $\int_{-\infty}^{\infty}(b_0^2(t) + b_n^2(t))\,dt$. Эта величина равна нулю по предположению.

Положим теперь $\varphi(t) = a_j(t) - a_{n-j+1}(t)$. При всех t функция $\varphi(t)$ ограничена величиной $2\|T(t)\|_2 = 2\|T(0)\|_2$. Имеем

$$\begin{aligned}\dot{\varphi}(t) &= \dot{a}_j(t) - \dot{a}_{n-j+1}(t)\\ &= 2(b_j^2(t) - b_{j-1}^2(t)) - 2(b_{n-j+1}^2(t) - b_{n-j}^2(t))\\ &= 2(b_j^2(t) + b_{n-j}^2(t)) - 2(b_{j-1}^2(t) + b_{n-j+1}^2(t)),\end{aligned}$$

откуда

$$\begin{aligned}\varphi(\tau) - \varphi(-\tau) &= \int_{-\tau}^{\tau}\dot{\varphi}(t)\,dt\\ &= 2\int_{-\tau}^{\tau}(b_j^2(t) + b_{n-j}^2(t))\,dt - 2\int_{-\tau}^{\tau}(b_{j-1}^2(t) + b_{n-j+1}^2(t))\,dt.\end{aligned}$$

Последний интеграл ограничен при всех t, согласно предположению индукции. Так как функция $\varphi(\tau) - \varphi(-\tau)$ ограничена при всех τ, то интеграл $\int_{-\infty}^{\infty}(b_j^2(t) + b_{n-j}^2(t))\,dt$ тоже должен быть ограничен.

Положим $p(t) = \sum_{i=1}^{n-1} b_i^2(t)$. Мы уже знаем, что $\int_{-\infty}^{\infty} p(t)\,dt < \infty$. Поскольку $p(t) \geq 0$, мы хотели бы вывести отсюда, что $\lim_{t\to\pm\infty} p(t) = 0$. Однако нужно сначала исключить возможность для $p(t)$ иметь узкие пики при $t \to \pm\infty$, иначе это позволило бы интегралу $\int_{-\infty}^{\infty} p(t)\,dt$ быть конечным даже при том, что $p(t)$ не стремится к нулю. Мы докажем отсутствие пиков, проверив, что производная функции $p(t)$ ограничена:

$$|\dot{p}(t)| = \left|\sum_{i=1}^{n-1} 2\dot{b}_i(t)b_i(t)\right| = \left|\sum_{i=1}^{n-1} 2b_i^2(t)(a_{i+1}(t) - a_i(t))\right| \leq 4(n-1)\|T\|_2^3. \qquad \square$$

Таким образом, чтобы решить проблему собственных значений, в принципе, можно воспользоваться программой, решающей системы обыкновенных дифференциальных уравнений, применяя ее к потоку Тода. Однако этот способ не будет быстрее других существующих методов. Самое интересное в потоке Тода — это его тесная связь с QR-алгоритмом.

Определение 5.5. *Пусть X_- обозначает нижнюю строго треугольную часть матрицы X. Положим $\pi_0(X) = X_- - X_-^T$.*

Заметим, что $\pi_0(X)$ — кососимметричная матрица и если сама матрица X кососимметрична, то $\pi_0(X) = X$. Таким образом, π_0 есть *проектор* на множество кососимметричных матриц.

Рассмотрим дифференциальное уравнение

$$\frac{d}{dt}T = BT - TB, \tag{5.28}$$

где $B = -\pi_0(F(T))$ и F — произвольная гладкая функция вещественного переменного, принимающая вещественные значения. Поскольку $B = -B^T$, из теоремы 5.16 следует, что матрица $T(t)$ имеет одни и те же собственные значения для всех t. Выбор $F(x) = x$ соответствует только что рассмотренному потоку Тода, так как в этом случае

$$-\pi_0(F(T)) = -\pi_0(T) = \begin{bmatrix} 0 & b_1 & & \\ -b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & -b_{n-1} & 0 \end{bmatrix} = B.$$

Оказывается, что QR-разложение связано с решением дифференциального уравнения (5.28).

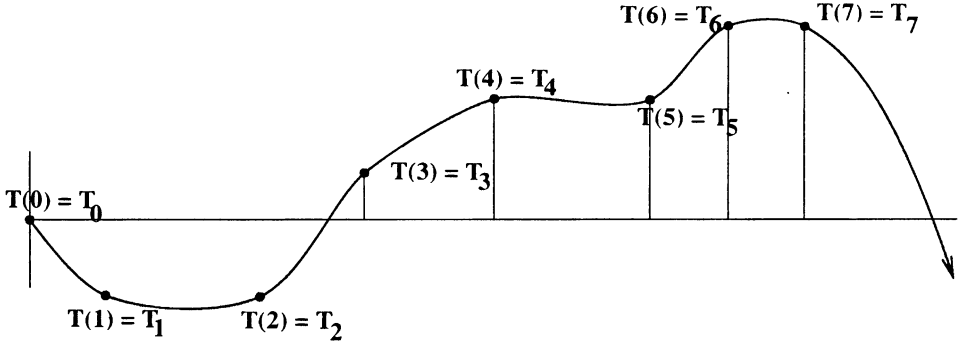
Теорема 5.18. *Пусть $F(T(0)) = F_0$, и пусть $e^{tF_0} = Q(t)R(t)$ есть QR-разложение матрицы e^{tF_0}. Тогда матрица $T(t) = Q^T(t)T(0)Q(t)$ является решением уравнения (5.28).*

Мы докажем эту теорему несколько позже. Оказывается, что при подходящем выборе функции F получается решение дифференциального уравнения,

значения которого *совпадают* с матрицами, вычисляемыми QR-итерацией (алгоритм 4.4).

Определение 5.6. *Выбор* $F(x) = \log x$ *в (5.28) дает дифференциальное уравнение, называемое* QR-потоком.

Следствие 5.3. *Пусть* $F(x) = \log x$. *Предположим, что матрица* $T(0)$ *положительно определена, тогда матрица* $\log T(0)$ *вещественна. Пусть* $T_0 \equiv T(0) = QR$, $T_1 = RQ$, *и т. д. есть последовательность матриц, генерируемая QR-итерацией без сдвигов. Тогда* $T(i) = T_i$. *Таким образом, QR-алгоритм дает значения решения QR-потока в целочисленные моменты времени* t[1].



Доказательство следствия. При $t = 1$ имеем $e^{t \log T_0} = T_0 = Q(1)R(1)$, т. е. QR-разложение матрицы T_0. Отсюда $T(1) = Q^T(1)T_0Q(1) = R(1)Q(1) = T_1$, как и требуется. Поскольку решение задачи Коши единственно, это рассуждение модно распространить на бóльшие значения i, что дает $T(i) = T_i$. □

На рисунке следствие проиллюстрировано графически. Кривая представляет решение дифференциального уравнения, а точки — значения решения $T(i)$ в целочисленные моменты времени $t = 0, 1, 2 \ldots$. Указано, что $T(i)$ равны QR-итерациям T_i.

Доказательство теоремы 5.18. Дифференцируя равенство $e^{tF_0} = QR$, получаем

$$
\begin{aligned}
F_0 e^{tF_0} &= \dot{Q}R + Q\dot{R} \\
\text{или} \quad \dot{Q} &= F_0 e^{tF_0} R^{-1} - Q\dot{R}R^{-1} \\
\text{или} \quad Q^T\dot{Q} &= Q^T F_0 e^{tF_0} R^{-1} - \dot{R}R^{-1} \\
&= Q^T F_0 (QR) R^{-1} - \dot{R}R^{-1} \quad \text{потому что } e^{tF_0} = QR \\
&= Q^T F(T(0)) Q - \dot{R}R^{-1} \quad \text{потому что } F_0 = F(T(0)) \\
&= F(Q^T T(0) Q) - \dot{R}R^{-1} \\
&= F(T) - \dot{R}R^{-1}.
\end{aligned}
$$

[1] Отметим, что поскольку QR-разложение не вполне единственно (Q можно заменить матрицей QS, а R — матрицей SR, где S — диагональная матрица с диагональными элементами ± 1), то T_i и $T(i)$ могут в действительности различаться подобием вида $T_i = ST(i)S^{-1}$. Для простоты, мы предположим, что и здесь, и в следствии 5.4, S выбрана так, чтобы $T_i = T(i)$.

Из соотношения $I = Q^TQ$ выводим, что $0 = \frac{d}{dt}Q^TQ = \dot{Q}^TQ + Q^T\dot{Q} = (Q^T\dot{Q})^T + (Q^T\dot{Q})$. Это означает, что матрица $Q^T\dot{Q}$ кососимметрична, поэтому $\pi_0(Q^T\dot{Q}) = Q^T\dot{Q} = \pi_0(F(T) - \dot{R}R^{-1})$. Присутствие в аргументе верхней треугольной матрицы $\dot{R}R^{-1}$ не изменяет значения функции π_0, поэтому окончательно находим $Q^T\dot{Q} = \pi_0(F(T))$. Теперь имеем

$$\begin{aligned}\frac{d}{dt}T(t) &= \frac{d}{dt}Q^T(t)T(0)Q(t)\\ &= \dot{Q}^TT(0)Q + Q^TT(0)\dot{Q}\\ &= \dot{Q}^T(QQ^T)T(0)Q + Q^TT(0)(QQ^T)\dot{Q}\\ &= \dot{Q}^TQT(t) + T(t)Q^T\dot{Q}\\ &= -Q^T\dot{Q}T(t) + T(t)Q^T\dot{Q}\\ &= -\pi_0(F(T(t)))T(t) + T(t)\pi_0(F(T(t))),\end{aligned}$$

что и требовалось. □

Следующее следствие объясняет наблюдение, сделанное в вопросе 4.15, где QR-итерацию удалось «обратить вспять» и вернуться к исходной матрице. См. по этому поводу также вопрос 5.25.

Следствие 5.4. *Предположим, что матрица T_4 получена из положительно определенной матрицы T_0 посредством следующих операций:*

1. *С T_0 выполняются m шагов QR-алгоритма без сдвигов, что дает матрицу T_1.*
2. *Полагаем T_2 = «отраженная матрица T_1» $= JT_1J$, где J получается из единичной матрицы обращением порядка столбцов.*
3. *С T_2 выполняются m шагов QR-алгоритма без сдвигов, что дает матрицу T_3.*
4. *Полагаем $T_4 = JT_3J$.*

Тогда $T_4 = T_0$.

Доказательство. Легко проверить, что если $X = X^T$, то $\pi_0(JXJ) = -J\pi_0(X)J$. Отсюда следует, что матрица $T_J(t) \equiv JT(t)J$ удовлетворяет уравнению

$$\begin{aligned}\frac{d}{dt}T_J(t) &= J\frac{d}{dt}T(t)J\\ &= J[-\pi_0(F(T))T + T\pi_0(F(T))]J\\ &= -J\pi_0(F(T))J(JTJ) + (JTJ)J\pi_0(F(T))J, \quad \text{так как } J^2 = I,\\ &= \pi_0(JF(T)J)T_J - T_J\pi_0(JF(T)J)\\ &= \pi_0(F(JTJ))T_J - T_J\pi_0(F(JTJ))\\ &= \pi_0(F(T_J))T_J - T_J\pi_0(F(T_J)).\end{aligned}$$

Это почти такое же уравнение, как для $T(t)$. В действительности, это *в точности* то уравнение, которому удовлетворяет функция $T(-t)$:

$$\frac{d}{dt}T(-t) = -\frac{d}{dt}T|_{-t} = -[-\pi_0(F(T))T + T\pi_0(F(T))]_{-t}.$$

Поэтому при одном и том же начальном условии T_2 матрицы $T_J(t)$ и $T(-t)$ должны совпадать всюду. При интегрировании по промежутку длины m реше-

ние $T(-t)$ возвращается из состояния $T_2 = JT_1J$ в начальное состояние JT_0J. Следовательно, $T_3 = JT_0J$ и $T_4 = JT_3J = T_0$, что и требовалось. □

5.5.2. Связь с дифференциальными уравнениями в частных производных

Этот раздел можно опустить при первом чтении книги.

Пусть $T(t) = -\frac{\partial^2}{\partial x^2} + q(x,t)$ и $B(t) = -4\frac{\partial^3}{\partial x^3} + 3(q(x,t)\frac{\partial}{\partial x} + \frac{\partial}{\partial x}q(x,t))$. И $T(t)$, и $B(t)$ суть линейные операторы (т. е. обобщения матриц), действующие в функциональных пространствах.

Подставляя $T(t)$ в уравнение $\frac{dT}{dt} = BT - TB$, получаем

$$q_t = 6qq_x - q_{xxx} \tag{5.29}$$

при условии, что для q выбраны подходящие граничные условия. (B должен быть кососимметричным оператором, а T — симметричным.) Уравнение (5.29) называется уравнением *Кортевега—де Фриза* и описывает течение воды в мелком канале. Можно строго показать, что это уравнение сохраняет при всех t собственные значения оператора $T(t)$, т. е. обыкновенное дифференциальное уравнение

$$\left(-\frac{\partial^2}{\partial x^2} + q(x,t)\right) h(x) = \lambda h(x)$$

имеет при всех t одно и то же бесконечное множество собственных значений $\lambda_1, \lambda_2, \ldots$. Иначе говоря, существует бесконечная последовательность величин, родственных энергии, которые сохраняются уравнением Кортевега—де Фриза. Это важно и с теоретической точки зрения, и с вычислительной.

Более подробно о потоке Тода можно прочитать в [144, 170, 67, 68, 239], а также в статьях Крускала [166], Флашки [106] и Мозера [187] в сборнике [188].

5.6. Литература и смежные вопросы к главе 5

Прекрасным справочником по симметричной проблеме собственных значений является книга [197]. Изложение теории относительных возмущений можно найти в [75, 82, 101]; раздел 5.2.1 был основан на последней из этих публикаций. Родственный материал содержится в [66, 92, 228, 250]. Книга [161] — это классический учебник теории возмущений для линейных операторов общего вида. Обзор параллельных алгоритмов для симметричной проблемы собственных значений дан в [76]. QR-алгоритм в приложении к вычислению SVD двухдиагональных матриц обсуждается в [80, 67, 120], а алгоритм dqds — в [104, 200, 209]. Анализ ошибок метода бисекции проведен в [73, 74, 156]; отметим, что в последнее время метод пытаются ускорить (см. [105, 203, 201, 176, 173, 175, 269]). Современные исследования по усовершенствованию обратной итерации представлены работами [105, 83, 201, 203]. Алгоритм «разделяй-и-властвуй» для задач на собственные значения был впервые предложен в [59], а затем развивался в работах [13, 90, 127, 131, 153, 172, 210, 234]. Возможность вычисления собственных значений с высокой точностью с помощью метода Якоби разрабатывается в [66, 75, 82, 92, 183, 228]. Поток Тода и сходные феномены обсуждаются в [67, 68, 106, 144, 166, 170, 187, 188, 239].

5.7. Вопросы к главе 5

Вопрос 5.1 (*легкий, Z. Bai*). Показать, что матрица $A = B+iC$ тогда и только тогда является эрмитовой, когда матрица

$$M = \begin{bmatrix} B & -C \\ C & B \end{bmatrix}$$

симметрична. Выразить собственные значения и собственные векторы матрицы M через собственные значения и собственные векторы матрицы A.

Вопрос 5.2 (*средней трудности*). Доказать следствие 5.1, используя теорему Вейля (т. е. теорему 5.1) и утверждение 4 теоремы 3.3.

Вопрос 5.3 (*средней трудности*). Исследовать картину линий уровня для произвольной 3×3-матрицы A с собственными значениями $\alpha_3 \le \alpha_2 \le \alpha_1$, аналогичную рис. 5.1. Пусть C_1 и C_2 — две большие окружности, вдоль которых $\rho(u, A) = \alpha_2$. Под каким углом они пересекаются?

Вопрос 5.4 (*трудный*). Опираясь на минимаксную теорему Куранта—Фишера (теорему 5.2), доказать *теорему Коши о перемежаемости*:

- Пусть $A = \begin{bmatrix} H & b \\ b^T & u \end{bmatrix}$ — симметричная $n \times n$-матрица и H — подматрица порядка $n-1$. Пусть $\alpha_n \le \cdots \le \alpha_1$ — собственные значения матрицы A, а $\theta_{n-1} \le \cdots \le \theta_1$ — собственные значения матрицы H. Доказать, что два этих набора чисел *перемежаются*, т. е.
$$\alpha_n \le \theta_{n-1} \le \cdots \le \theta_i \le \alpha_i \le \theta_{i-1} \le \alpha_{i-1} \le \cdots \le \theta_1 \le \alpha_1.$$
- Пусть $A = \begin{bmatrix} H & B \\ B^T & U \end{bmatrix}$ по-прежнему порядка n, а $m \times m$-подматрица H имеет собственные значения $\theta_m \le \cdots \le \theta_1$. Доказать, что собственные значения матриц A и H перемежаются в смысле справедливости неравенств $\alpha_{j+(n-m)} \le \theta_j \le \alpha_j$ (или, что эквивалентно, $\alpha_j \le \theta_{j-(n-m)} \le \alpha_{j-(n-m)}$).

Вопрос 5.5 (*средней трудности*). Пусть $\alpha_1 \ge \cdots \ge \alpha_n$ — собственные значения матрицы $A = A^T$, а $\theta_1 \ge \cdots \ge \theta_n$ — собственные значения матрицы $H = H^T$. Обозначим через $\lambda_1 \ge \cdots \ge \lambda_n$ собственные значения матрицы $A + H$. Используя минимаксную теорему Куранта—Фишера (теорему 5.2), доказать неравенства $\alpha_j + \theta_n \le \lambda_j \le \alpha_j + \theta_1$. Для положительно определенной матрицы H вывести отсюда неравенство $\lambda_j > \alpha_j$. Другими словами, добавление симметричной положительно определенной матрицы H к симметричной матрице A может лишь увеличить собственные значения последней.

Этот результат будет использован в доказательстве теоремы 7.1.

Вопрос 5.6 (*средней трудности*). Пусть $A = [A_1, A_2]$ — матрица порядка n, причем A_1 и A_2 имеют соответственно размеры $n \times m$ и $n \times (n-m)$. Пусть $\sigma_1 \ge \cdots \ge \sigma_n$ — сингулярные числа матрицы A, а $\tau_1 \ge \cdots \ge \tau_m$ — сингулярные числа матрицы A_1. Используя теорему Коши о перемежаемости (вопрос 5.4) и утверждение 4 теоремы 3.3, доказать неравенства $\sigma_j \ge \tau_j \ge \sigma_{j+n-m}$.

Вопрос 5.7 (*средней трудности*). Пусть q — нормированный вектор, а d — произвольный вектор, ортогональный к q. Показать, что $\|(q+d)q^T - I\|_2 = \|q+d\|_2$. (Этот результат используется в доказательстве теоремы 5.4.)

Вопрос 5.8 (*трудный*). Сформулировать и доказать теорему для сингулярных векторов, аналогичную теореме 5.4.

Вопрос 5.9 (*трудный*). Доказать оценку (5.6) в теореме 5.5.

Вопрос 5.10 (*еще более трудный*). Доказать оценку (5.7) в теореме 5.5.

Вопрос 5.11 (*легкий*). Пусть $\theta = \theta_1 + \theta_2$, причем все три угла заключены между 0 и $\pi/2$. Доказать, что $\frac{1}{2}\sin 2\theta \leq \frac{1}{2}\sin 2\theta_1 + \frac{1}{2}\sin 2\theta_2$. Этот результат используется в доказательстве теоремы 5.7.

Вопрос 5.12 (*трудный*). Доказать следствие 5.2. Указание: использовать утверждение 4 теоремы 3.3.

Вопрос 5.13 (*средней трудности*). Пусть A — симметричная матрица. Предположим, что к A применяется QR-итерация со сдвигами (алгоритм 4.5). На каждом шаге в качестве сдвига берется отношение Рэлея ($\sigma_i = a_{nn}$), что порождает последовательность сдвигов $\sigma_1, \sigma_2, \ldots$. Пусть, кроме того, к A применяется RQ-итерация (алгоритм 5.1) с начальным вектором $x_0 = [0, \ldots, 0, 1]^T$, порождающая последовательность частных Рэлея $\rho_1, \rho_2, \ldots$. Показать, что эти две последовательности одинаковы, т. е. $\sigma_i = \rho_i$ для всех i. Этим обосновано утверждение, сделанное в разд. 5.3.2, что QR-итерация со сдвигами обладает свойством локальной кубической сходимости.

Вопрос 5.14 (*легкий*). Доказать лемму 5.1.

Вопрос 5.15 (*легкий*). Доказать, что если $t(n) = 2t(n/2) + cn^3 + O(n^2)$, то $t(n) \approx c\frac{4}{3}n^3$. На этом факте основывается анализ сложности алгоритма «разделяй-и-властвуй» (алгоритма 5.2).

Вопрос 5.16 (*легкий*). Пусть $A = D + \rho uu^T$, где $D = \mathrm{diag}(d_1, \ldots, d_n)$ и $u = [u_1, \ldots, u_n]^T$. Показать, что если $d_i = d_{i+1}$ или $u_i = 0$, то d_i есть собственное значение матрицы A. Для случая $u_i = 0$ показать, что собственному значению d_i в качестве собственного вектора отвечает i-й столбец e_i единичной матрицы. Вывести аналогичное простое выражение для собственного вектора в случае $d_i = d_{i+1}$. Это показывает, как обрабатывается ситуация дефляции в алгоритме «разделяй-и-властвуй» (алгоритм 5.2).

Вопрос 5.17 (*легкий*). Пусть ψ и ψ' — заданные числа. Показать, как вычисляются коэффициенты c и $\hat{c}$, определяющие функцию $h(\lambda) = \hat{c} + \frac{c}{d-\lambda}$, такую, что при $\lambda = \xi$ имеем $h(\xi) = \psi$ и $h'(\xi) = \psi'$. Этот результат нужен для алгоритма решения векового уравнения (см. разд. 5.3.3).

Вопрос 5.18 (*легкий; Z. Bai*). Используя SVD, показать, что для вещественной $m \times n$-матрицы A, где $m \geq n$, найдутся $m \times n$-матрица Q с ортонормированными столбцами (т. е. $Q^TQ = I$) и положительно полуопределенная $n \times n$-матрица P, такие, что $A = QP$. Такое представление матрицы A называется ее *полярным разложением*, потому что оно аналогично полярной форме

комплексного числа: $z = e^{i \arg(z)} \cdot |z|$. Доказать, что полярное разложение квадратной невырожденной матрицы A единственно.

Вопрос 5.19 (*легкий*). Доказать лемму 5.5.

Вопрос 5.20 (*легкий*). Доказать лемму 5.7.

Вопрос 5.21 (*трудный*). Доказать теорему 5.13. Кроме того, уменьшить показатель $4n - 2$ в оценке из этой теоремы до $2n - 1$. Указание: в лемме 5.7 умножить D_1 и разделить D_2 на подходящим образом выбранную константу.

Вопрос 5.22 (*средней трудности*). Доказать, что алгоритм 5.13 вычисляет SVD матрицы G, предполагая, что $G^T G$ сходится к диагональной матрице.

Вопрос 5.23 (*более трудный*). Пусть A — симметричная положительно определенная $n \times n$-матрица с разложением Холесского $A = LL^T$ и пусть $\widehat{L}$ — ее множитель Холесского, вычисленный в арифметике с плавающей точкой. В этой задаче мы оценим относительную погрешность (возведенных в квадрат) сингулярных чисел матрицы $\widehat{L}$ как приближений к собственным значениям матрицы A. Показать, что A может быть представлена в виде $A = D\overline{A}D$, где $D = \mathrm{diag}(a_{11}^{1/2}, \ldots, a_{nn}^{1/2})$ и $\overline{a_{ii}} = 1$ для всех i. Положим $L = DX$. Показать, что $\kappa^2(X) = \kappa(\overline{A})$. Используя оценку (2.16) для обратной ошибки δA метода Холесского (т. е. $A + \delta A = \widehat{L}\widehat{L}^T$), показать возможность представления $\widehat{L}^T\widehat{L} = Y^T L^T L Y$, где $\|Y^T Y - I\|_2 \leq O(\varepsilon)\kappa(\overline{A})$. Опираясь на теорему 5.6, вывести отсюда, что относительные разности собственных значений матриц $\widehat{L}^T\widehat{L}$ и $L^T L$ не превосходят $O(\varepsilon)\kappa(\overline{A})$. Затем показать, что такое же утверждение верно в отношении собственных значений матриц $\widehat{L}\widehat{L}^T$ и LL^T. Это доказывает, что относительные разности квадратов сингулярных чисел матрицы $\widehat{L}$ и собственных значений матрицы A ограничены величиной $O(\varepsilon)\kappa(\overline{A}) = O(\varepsilon)\kappa^2(X)$.

Вопрос 5.24 (*более трудный*). В этой задаче обосновывается критерий останова метода односторонних вращений для вычисления SVD (см. алгоритм 5.13). Пусть G и A — матрицы порядка n и $A = G^T G$. Предположим, что $|a_{jk}| \leq \epsilon\sqrt{a_{jj}a_{kk}}$ для всех $j \neq k$. Пусть $\sigma_n \leq \cdots \leq \sigma_1$ — сингулярные числа матрицы G, а $\alpha_n^2 \leq \cdots \leq \alpha_1^2$ — упорядоченные диагональные элементы матрицы A. Доказать неравенства $|\sigma_i - |\alpha_i|| \leq n\epsilon|\alpha_i|$. Таким образом, $|\alpha_i|$ суть приближения высокой относительной точности к соответствующим сингулярным числам. Указание: использовать следствие 5.2.

Вопрос 5.25 (*более трудный*). В вопросе 4.15 вы «заметили», что если к симметричной матрице применить m шагов QR-итерации, «зеркально отразить» строки и столбцы полученной матрицы, провести еще m шагов QR-итерации и снова выполнить зеркальное отражение, то результатом будет исходная матрица. (Зеркальное отражение матрицы X заменяет ее матрицей JXJ, где J получается из единичной матрицы обращением порядка строк.) В этом упражнении мы обосновываем данный феномен для симметричных положительно определенных матриц T, используя подход, отличающийся от следствия 5.4.

Рассмотрим LR-итерацию (алгоритм 5.9) с нулевым сдвигом в применении к симметричной положительно определенной матрице T (не обязательно трехдиагональной): сначала вычисляется разложение Холесского $T = T_0 = B_0^T B_0$,

затем матрица $T_1 = B_0 B_0^T = B_1^T B_1$ и, более общо, матрица $T_i = B_{i-1} B_{i-1}^T = B_i^T B_i$. Пусть $\widehat{T}_i$ — матрица, полученная из T_0 посредством i шагов QR-итерации без сдвигов (т. е. если $\widehat{T}_i = Q_i R_i$ есть QR-разложение матрицы $\widehat{T}_i$, то $\widehat{T}_{i+1} = R_i Q_i$). В лемме 5.6 было показано, что $\widehat{T}_i = T_{2i}$, т. е. один шаг QR-итерации эквивалентен двум шагам LR-итерации.

1. Показать, что $T_i = (B_{i-1} B_{i-2} \cdots B_0)^{-T} T_0 (B_{i-1} B_{i-2} \cdots B_0)^T$.
2. Показать, что $T_i = (B_{i-1} B_{i-2} \cdots B_0) T_0 (B_{i-1} B_{i-2} \cdots B_0)^{-1}$.
3. Показать, что разложение Холесского матрицы T_0^i имеет вид $T_0^i = (B_i B_{i-1} \cdots B_0)^T (B_i B_{i-1} \cdots B_0)$.
4. Показать, что QR-разложение матрицы T_0^i имеет вид $T_0^i = (Q_0 \cdots Q_{i-2} Q_{i-1}) \cdot (R_{i-1} R_{i-2} \cdots R_0)$.
5. Показать, что разложение Холесского матрицы T_0^{2i} имеет вид $T_0^{2i} = (R_{2i-1} R_{2i-2} \cdots R_0)^T (R_{2i-1} R_{2i-2} \cdots R_0)$.
6. Показать, что матрица, полученная в результате m шагов QR-итерации, зеркального отражения, еще m шагов QR-итерации и еще одного отражения совпадает с исходной матрицей. Указание: использовать единственность разложения Холесского.

Вопрос 5.26 (*трудный; Z. Bai*). Пусть x — вектор размерности n. Определим матрицу C с элементами $c_{ij} = |x_i| + |x_j| - |x_i - x_j|$. Показать, что эта матрица положительно полуопределена.

Вопрос 5.27 (*легкий; Z. Bai*). Пусть

$$A = \begin{pmatrix} I & B \\ B^H & I \end{pmatrix},$$

причем $\|B\|_2 < 1$. Показать, что

$$\|A\|_2 \|A^{-1}\|_2 = \frac{1 + \|B\|_2}{1 - \|B\|_2}.$$

Вопрос 5.28 (*средней трудности; Z. Bai*). Квадратная матрица A называется *косоэрмитовой*, если $A^* = -A$. Доказать, что

1. собственные значения косоэрмитовой матрицы суть чисто мнимые числа;
2. матрица $I - A$ невырожденна;
3. матрица $C = (I - A)^{-1}(I + A)$ унитарная. Эта матрица называется *преобразованием Кэли* матрицы A.

Глава 6

Итерационные методы для линейных систем

6.1. Введение

Итерационные алгоритмы используются для решения системы $Ax = b$ тогда, когда методы типа гауссова исключения требуют слишком много времени или памяти. Методы, вычисляющие точное решение (в отсутствие ошибок округлений!) за конечное число шагов, называются *прямыми*. В отличие от этого, итерационные методы обычно не дают точного ответа за конечное число шагов, однако на каждом шаге уменьшают ошибку на какую-то долю. Итерации прекращают, когда ошибка становится меньше допуска, заданного пользователем. Величина финальной ошибки зависит от количества итераций, а также от свойств метода и линейной системы. Наша общая цель состоит в разработке методов, которые бы существенно уменьшали ошибку на каждой итерации и при этом выполняли бы как можно меньшую работу.

Существенную часть работы по конструированию более эффективных итерационных методов составляет использование связи решаемой линейной системы с математической или физической задачей, которая ее породила. Последняя часто является конечно-разностной или конечно-элементной моделью некоторой физической системы, обычно описываемой дифференциальным уравнением. Существует множество физических систем, дифференциальных уравнений, конечно-разностных и конечно-элементных моделей, а потому и множество итерационных методов. Невозможно охватить все ситуации или хотя бы самые интересные из них, поэтому мы ограничимся исследованием *модельной задачи*, представляющей собой стандартную конечно-разностную аппроксимацию уравнения Пуассона в квадрате. Уравнение Пуассона и тесно с ним связанное уравнение Лапласа возникают во многих приложениях; укажем, например, задачи электромагнитной теории, гидродинамики, теплопередачи, диффузии и квантовой механики. Мы изучим, как каждый из описываемых методов работает для уравнения Пуассона, обрисуем область его применимости в целом и укажем самые распространенные варианты метода.

Остальная часть этой главы организована следующим образом. В разд. 6.2 описаны Интернет-ресурсы и программное обеспечение для обсуждаемых здесь итерационных методов. В разд. 6.3 приводится подробная формулировка модельной задачи. В разд. 6.4 даны сводка итерационных методов из этой главы и сравнение производительности (почти) всех этих методов при решении модельной задачи.

Последовательность изложения методов в следующих пяти разделах примерно соответствует возрастанию их эффективности в применении к модельной задаче. В разд. 6.5 описаны основные итерационные методы, а именно методы Якоби, Гаусса—Зейделя, последовательной верхней релаксации и их варианты. В разд. 6.6 рассмотрены методы крыловского подпространства, причем наибольшее внимание уделено методу сопряженных градиентов. Быстрое преобразование Фурье и его использование для решения модельной задачи составляют содержание разд. 6.7. Блочная циклическая редукция описана в разд. 6.8. Наконец, в разд. 6.9 обсуждается многосеточный метод, являющийся нашим самым эффективным алгоритмом для модельной задачи. В многосеточном методе на одно неизвестное приходится оптимальное количество работы $O(1)$.

В разд. 6.10 изучается техника декомпозиции области, представляющая собой семейство приемов комбинирования простых методов, описанных в предыдущих разделах, с целью решения более сложных задач, чем наша модельная задача.

6.2. Интернет-ресурсы для итерационных методов

Для уравнения Пуассона мы составим короткий список численных методов, которые явно превосходят все остальные из обсуждаемых нами. Однако для других линейных систем не всегда ясно, какой метод является наилучшим (вот почему мы говорим о столь многих!). Чтобы помочь пользователям выбрать среди множества существующих наилучший метод для их линейных систем, в NETLIB/templates помещена справочная система. В этой директории содержатся небольшая книга [24] и программы большинства итерационных методов, обсуждаемых в данной главе. Имеются версии книги на языках PostScript (см. NETLIB/templates/templates.ps) и Hypertext Markup Language (NETLIB/templates/Templates.html). Программы написаны на языках Matlab, фортран, C++.

Слово *templates* (шаблоны) указывает, что в описываемых книгах и программах детали, связанные с представлением матриц, отделены от алгоритмов самих по себе. Например, в *методах крыловского подпространства* (см. разд. 6.6) все, что требуется от матрицы A, это возможность умножить ее на произвольный вектор z. Способ представления A определяет наилучший способ вычисления таких произведений, но никак иначе не влияет на организацию алгоритма. Иными словами, в шаблоне операция матрично-векторного умножения рассматривается как «черный ящик». Конкретная реализация такого черного ящика предоставляется пользователю.

Аналогичный набор шаблонов разрабатывается в настоящее время для задач на собственные значения. Среди других недавно изданных учебников по итерационным методам назовем [15, 136, 214].

В самых трудоемких практических задачах, проистекающих из более сложных дифференциальных уравнений, чем наше модельное уравнение, линейная система $Ax = b$ должна быть «предобусловлена», т. е. заменена эквивалентной системой $M^{-1}Ax = M^{-1}b$, которую в том или ином отношении легче решить. Эта тема подробно обсуждается в разд. 6.6.5 и 6.10. Реализации (включая и параллельные) многих приемов предобусловливания помещены в Интернете в

виде пакета PETSc (от Portable Extensible Toolkit for Scientific computing) по адресу http://www.mcs.anl.gov/petsc/petsc.html [232].

6.3. Уравнение Пуассона

6.3.1. Одномерное уравнение Пуассона

Мы начинаем с одномерной версии уравнения Пуассона

$$-\frac{d^2v(x)}{dx^2} = f(x), \quad 0 < x < 1, \tag{6.1}$$

где $f(x)$ — заданная функция, а $v(x)$ — неизвестная функция, которую мы хотим вычислить. Эта функция должна еще удовлетворять граничным условиям[1] $v(0) = v(1) = 0$. Мы заменим эту задачу *разностной*, стремясь к тому, чтобы найти приближенные значения решения в $N+2$ равноотстоящих точках x_i, находящихся между 0 и 1: $x_i = ih$, где $h = \frac{1}{N+1}$ и $0 \le i \le N+1$. Будем пользоваться сокращенными обозначениями $v_i = v(x_i)$ и $f_i = f(x_i)$. Чтобы превратить дифференциальное уравнение (6.1) в систему линейных уравнений относительно неизвестных $v_1, \ldots, v_N$, аппроксимируем производные *конечными разностями*:

$$\left.\frac{dv(x)}{dx}\right|_{x=(i-.5)h} \approx \frac{v_i - v_{i-1}}{h},$$
$$\left.\frac{dv(x)}{dx}\right|_{x=(i+.5)h} \approx \frac{v_{i+1} - v_i}{h}.$$

Вычитая эти приближения и деля разность на h, приходим к *центрированному разностному приближению*

$$-\left.\frac{d^2v(x)}{dx^2}\right|_{x=x_i} = \frac{2v_i - v_{i-1} - v_{i+1}}{h^2} - \tau_i. \tag{6.2}$$

Можно показать, что *погрешность аппроксимации* τ_i равна $O(h^2 \cdot \|\frac{d^4v}{dx^4}\|_\infty)$. Теперь уравнение (6.1) в точке $x = x_i$ можно записать как

$$-v_{i-1} + 2v_i - v_{i+1} = h^2 f_i + h^2 \tau_i,$$

где $0 < i < N+1$. Поскольку из граничных условий следует, что $v_0 = v_{N+1} = 0$, мы имеем N уравнений с N неизвестными $v_1, \ldots, v_N$:

$$T_N \cdot \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ v_N \end{bmatrix} \equiv \begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ v_N \end{bmatrix}$$
$$= h^2 \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_N \end{bmatrix} + h^2 \begin{bmatrix} \tau_1 \\ \vdots \\ \vdots \\ \tau_N \end{bmatrix}, \tag{6.3}$$

[1] Они называются *условиями Дирихле*. Возможны и другие типы граничных условий.

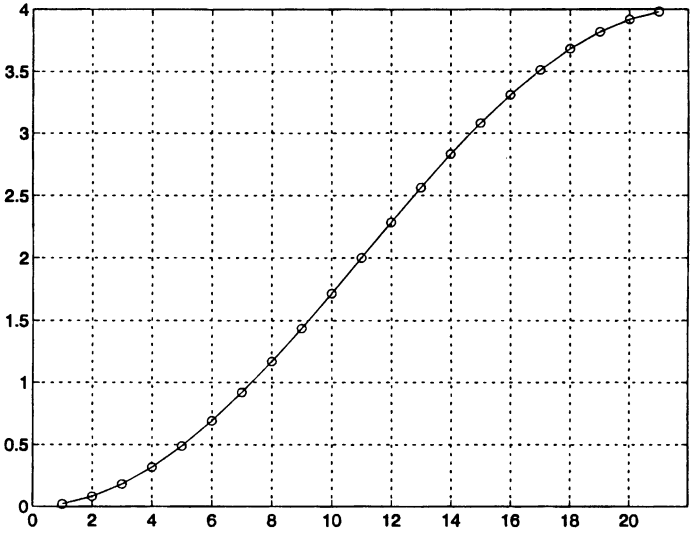


Рис. 6.1. Собственные значения матрицы T_{21}.

или

$$T_N v = h^2 f + h^2 \overline{\tau}. \tag{6.4}$$

Чтобы решить эту систему, пренебрежем величиной $\overline{\tau}$, которая мала по сравнению с f. Тогда получим

$$T_N \hat{v} = h^2 f. \tag{6.5}$$

(Ошибка $v - \hat{v}$ будет оценена позднее.)

Матрица коэффициентов T_N играет центральную роль во всем дальнейшем, поэтому мы исследуем ее несколько подробней. Сначала мы вычислим ее собственные значения и собственные векторы. С помощью тригонометрических тождеств легко проверяется следующая лемма (см. вопрос 6.1).

Лемма 6.1. *Собственными значениями матрицы* T_N *являются числа* $\lambda_j = 2(1 - \cos\frac{\pi j}{N+1})$. *Собственные векторы* z_j *имеют компоненты* $z_j(k) = \sqrt{\frac{2}{N+1}}\sin(jk\pi/(N+1))$; *при этом 2-норма вектора* z_j *равна единице. Пусть* $Z = [z_1, \ldots, z_N]$ *— ортогональная матрица, составленная по столбцам из собственных векторов, и пусть* $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. *Тогда справедливо соотношение* $T_N = Z\Lambda Z^T$.

На рис. 6.1 изображены собственные значения матрицы T_N для $N = 21$.

Наибольшим собственным значением является число $\lambda_N = 2(1 - \cos\pi\frac{N}{N+1}) \approx 4$. Наименьшее собственное значение[1] есть λ_1, где для малых i

$$\lambda_i = 2\left(1 - \cos\frac{i\pi}{N+1}\right) \approx 2\left(1 - \left(1 - \frac{i^2\pi^2}{2(N+1)^2}\right)\right) = \left(\frac{i\pi}{N+1}\right)^2.$$

[1] Обратим внимание, что, в отличие от гл. 5, мы обозначаем через λ_N наибольшее, а через λ_1 — наименьшее собственное значение.

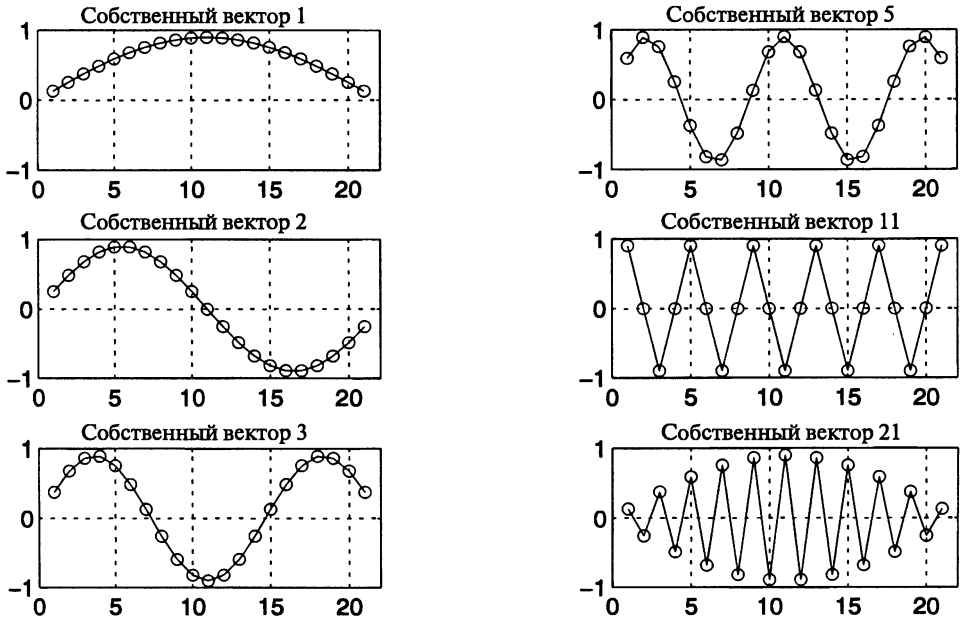


Рис. 6.2. Собственные векторы матрицы T_{21}.

Таким образом, матрица T_N положительно определена и ее число обусловленности λ_N/λ_1 составляет для больших N $\approx 4(N+1)^2/\pi^2$. Собственными векторами являются синусоиды, изображенные на рис. 6.2; наименьшая частота соответствует $j=1$, а наибольшая — $j=N$.

Теперь мы знаем достаточно для того, чтобы оценить ошибку, т. е. разность между решением системы $T_N\hat{v}=h^2f$ и точным решением v дифференциального уравнения. Вычитая (6.5) из (6.4), получаем $v-\hat{v}=h^2T_N^{-1}\overline{\tau}$. Переходя к нормам, находим

$$\|v-\hat{v}\|_2 \le h^2\|T_N^{-1}\|_2\|\overline{\tau}\|_2 \approx h^2\frac{(N+1)^2}{\pi^2}\|\overline{\tau}\|_2 = O(h^2\|\overline{\tau}\|_2) = O\left(h^2\left\|\frac{d^4v}{dx^4}\right\|_\infty\right),$$

поэтому ошибка $v-\hat{v}$ стремится к нулю пропорционально h^2 при условии достаточной гладкости решения v (т. е. конечности величины $\|\frac{d^4v}{dx^4}\|_\infty$).

Начиная с этого момента, мы не будем делать различия между v и его приближением $\hat{v}$, что упростит запись системы до $T_Nv=h^2f$.

Оказывается, что не только решение линейной системы $h^{-2}T_Nv=f$ аппроксимирует решение дифференциального уравнения (6.1), но и собственные значения и собственные векторы матрицы $h^{-2}T_N$ аппроксимируют собственные значения и *собственные функции* дифференциального уравнения. Говорят, что $\hat{\lambda}_i$ есть собственное значение, а $\hat{z}_i(x)$ — собственная функция уравнения (6.1), если

$$-\frac{d^2\hat{z}_i(x)}{dx^2}=\hat{\lambda}_i\hat{z}_i(x) \quad \text{и} \quad \hat{z}_i(0)=\hat{z}_i(1)=0.$$

Определим $\hat{\lambda}_i$ и $\hat{z}_i(x)$ из этих условий. Легко видеть, что $\hat{z}_i(x)$ должна быть функцией вида $\alpha \sin(\sqrt{\hat{\lambda}_i}x) + \beta \cos(\sqrt{\hat{\lambda}_i}x)$ для некоторых констант α и β. Граничное условие $\hat{z}_i(0) = 0$ дает $\beta = 0$, а из граничного условия $\hat{z}_i(1) = 0$ следует, что $\sqrt{\hat{\lambda}_i}$ есть целое кратное числа π; можно положить его равным $i\pi$. Тогда $\hat{\lambda}_i = i^2\pi^2$ и $\hat{z}_i(x) = \alpha \sin(i\pi x)$, где α — произвольная ненулевая константа (можно положить $\alpha = 1$). Таким образом, собственный вектор z_i *точно* равен собственной функции $\hat{z}_i(x)$, вычисляемой на множестве точек $x_j = jh$ (после масштабирования функции множителем $\sqrt{\frac{2}{N+1}}$). Кроме того, для малых i число $\hat{\lambda}_i = i^2\pi^2$ хорошо аппроксимируется величиной $h^{-2} \cdot \lambda_i = (N+1)^2 \cdot 2(1 - \cos\frac{i\pi}{N+1}) = i^2\pi^2 + O((N+1)^{-2})$.

Итак, мы видим, что между матрицей T_N (или $h^{-2}T_N$) и оператором $-\frac{d^2}{dx^2}$ имеется тесная связь. Эта связь будет мотивировать построение и анализ последующих алгоритмов.

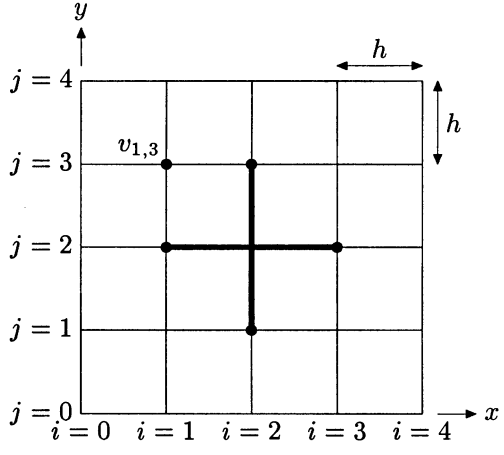
Для сомножителей LU-разложения и разложения Холесского матрицы T_N также можно вывести простые формулы; по поводу деталей см. вопрос 6.2.

6.3.2. Двумерное уравнение Пуассона

Теперь мы обратимся к двумерному уравнению Пуассона

$$-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2} = f(x,y) \tag{6.6}$$

на единичном квадрате $\{(x,y) : 0 < x,y < 1\}$ с условием $v = 0$ на границе квадрата. Для дискретизации уравнения берется сетка с узлами (x_i, y_j), где $x_i = ih$, $y_j = jh$ и $h = \frac{1}{N+1}$. Используются сокращенные обозначения $v_{ij} = v(ih, jh)$ и $f_{ij} = f(ih, jh)$; это иллюстрируется рисунком для $N = 3$.



Согласно (6.2), можно использовать аппроксимации

$$-\frac{\partial^2 v(x,y)}{\partial x^2}\bigg|_{x=x_i, y=y_j} \approx \frac{2v_{i,j} - v_{i-1,j} - v_{i+1,j}}{h^2} \tag{6.7}$$

и

$$-\frac{\partial^2 v(x,y)}{\partial y^2}\bigg|_{x=x_i,y=y_j} \approx \frac{2v_{i,j} - v_{i,j-1} - v_{i,j+1}}{h^2}. \tag{6.8}$$

Складывая их, можем написать

$$\begin{aligned} &-\frac{\partial^2 v(x,y)}{\partial x^2} - \frac{\partial^2 v(x,y)}{\partial y^2}\bigg|_{x=x_i,y=y_j} \\ &= \frac{4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1}}{h^2} - \tau_{ij}, \end{aligned} \tag{6.9}$$

где погрешность аппроксимации τ_{ij} снова ограничена величиной $O(h^2)$. Жирный крест посреди упомянутого выше рисунка называется (5-*точечным*) *шаблоном* данного уравнения, поскольку он связывает все пять значений функции v, присутствующих в уравнении (6.9). Согласно граничному условию, имеем $v_{0j} = v_{N+1,j} = v_{i,0} = v_{i,N+1} = 0$, поэтому (6.9) определяет систему из $n = N^2$ линейных уравнений относительно n неизвестных v_{ij}, где $1 \le i, j \le N$:

$$4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1} = h^2 f_{ij}. \tag{6.10}$$

Имеются два способа записать n уравнений, представляемых формулой (6.10), единым матричным уравнением; оба этих способа будут использованы в дальнейшем.

Первый способ состоит в том, чтобы рассматривать неизвестные v_{ij} как элементы $N \times N$-матрицы V, а правые части $h^2 f_{ij}$ аналогично как элементы $N \times N$-матрицы $h^2 F$. Затем используется трюк, позволяющий матрицу, элемент (i,j) которой равен $4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1}$, несложным образом выразить через V и T_N; именно, заметим, что

$$\begin{aligned} 2v_{ij} - v_{i-1,j} - v_{i+1,j} &= (T_N \cdot V)_{ij}, \\ 2v_{ij} - v_{i,j-1} - v_{i,j+1} &= (V \cdot T_N)_{ij}. \end{aligned}$$

Сложение этих уравнений дает

$$(T_N \cdot V + V \cdot T_N)_{ij} = 4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1} = h^2 f_{ij} = (h^2 F)_{ij},$$

или

$$T_N \cdot V + V \cdot T_N = h^2 F. \tag{6.11}$$

Это линейная система уравнений относительно неизвестных элементов матрицы V, хотя она и не записана в обычном формате «$Ax = b$», где неизвестные образуют вектор x. (Ниже мы представим эту систему и в формате «$Ax = b$».) Все же и такой записи вполне достаточно, чтобы определить собственные значения и собственные векторы соответствующей матрицы A, поскольку равенство $Ax = \lambda x$ эквивалентно равенству $T_N V + V T_N = \lambda V$. Предположим теперь, что взяты две собственные пары матрицы T_N, так что $T_N z_i = \lambda_i z_i$ и $T_N z_j = \lambda_j z_j$; положим $V = z_i z_j^T$. Тогда

$$\begin{aligned} T_N V + V T_N &= (T_N z_i) z_j^T + z_i (z_j^T T_N) \\ &= (\lambda_i z_i) z_j^T + z_i (z_j^T \lambda_j) \\ &= (\lambda_i + \lambda_j) z_i z_j^T \\ &= (\lambda_i + \lambda_j) V. \end{aligned} \tag{6.12}$$

Таким образом, матрица $V = z_i z_j^T$ является «собственным вектором», а число $\lambda_i + \lambda_j$ — собственным значением. Поскольку в V имеется N^2 элементов, следует ожидать, что задача имеет N^2 собственных значений и собственных векторов, по одному для каждой пары собственных значений λ_i и λ_j матрицы T_N. В частности, наименьшее и наибольшее собственные значения равны соответственно $2\lambda_1$ и $2\lambda_N$, поэтому число обусловленности остается тем же, что и в одномерном случае. Ниже мы выведем этот результат заново, используя формат «$Ax = b$». На рис. 6.3 приведены изображения некоторых собственных векторов, представленных как поверхности, определяемые элементами матрицы $z_i z_j^T$.

Собственные значения и собственные векторы матрицы $h^{-2}T_N$ были хорошими приближениями к собственным значениям и собственным функциям одномерного уравнения Пуассона. То же самое верно для двумерного уравнения, чьи собственные значения и собственные функции видны из соотношения (см. вопрос 6.3)

$$\left(-\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}\right) \sin(i\pi x)\sin(j\pi y)$$
$$= (i^2\pi^2 + j^2\pi^2)\sin(i\pi x)\sin(j\pi y). \tag{6.13}$$

Второй способ записать n уравнений, представленных формулой (6.10), одним матричным уравнением предполагает объединение неизвестных v_{ij} в длинном векторе размерности N^2. Это требует задания упорядочения для неизвестных; мы выберем (достаточно произвольно) упорядочение, показанное на рис. 6.4: нумерация неизвестных производится по столбцам при движении от левого верхнего узла к нижнему правому.

Например, при $N = 3$ получаем вектор-столбец $v \equiv [v_1, \ldots, v_9]^T$. При аналогичной нумерации правых частей f_{ij} уравнения (6.10) преобразуются к виду

$$T_{3\times 3} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ \vdots \\ v_9 \end{bmatrix} \equiv \left[\begin{array}{ccc|ccc|ccc} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ \hline -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ \hline & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{array}\right] \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ \vdots \\ v_9 \end{bmatrix}$$
$$= h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ \vdots \\ f_9 \end{bmatrix}. \tag{6.14}$$

Минус единицы рядом с главной диагональю соответствуют вычитанию верхнего и нижнего соседей $-v_{i,j-1} - v_{i,j+1}$. Минус единицы, удаленные от главной

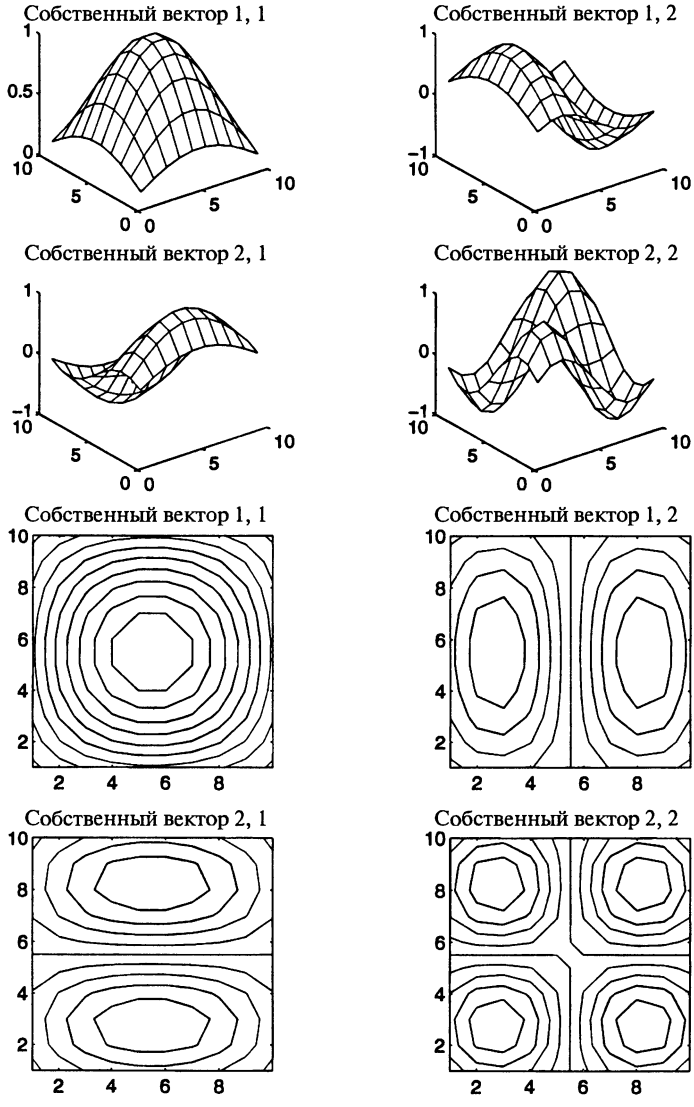


Рис. 6.3. Трехмерные и контурные изображения первых четырех собственных векторов уравнения Пуассона на сетке 10×10.

диагонали, соответствуют вычитанию левого и правого соседей $-v_{i-1,j} - v_{i+1,j}$. В следующем разделе мы убедимся, что для произвольного N получается линейная $N^2 \times N^2$-система

$$T_{N\times N} \cdot v = h^2 f, \tag{6.15}$$

где матрица $T_{N\times N}$ имеет на диагонали N блоков размера $N \times N$ и вида $T_N +$

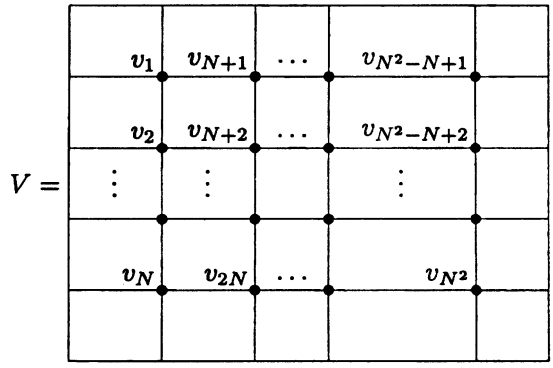


Рис. 6.4. Нумерация неизвестных в уравнении Пуассона.

$2I_N$, а на диагоналях, соседних с главной, блоки $-I_N$:

$$T_{N\times N} = \begin{bmatrix} T_N + 2I_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & \ddots & -I_N \\ & & -I_N & T_N + 2I_N \end{bmatrix}. \tag{6.16}$$

6.3.3. Запись уравнения Пуассона посредством кронекеровых произведений

Познакомимся с систематическим способом вывода уравнений (6.15) и (6.16), а также вычисления собственных значений и собственных векторов матрицы $T_{N\times N}$. Этот способ столь же хорошо работает для уравнения Пуассона в случае трех и более измерений.

Определение 6.1. *Пусть X — матрица размера $m \times n$. Тогда* $\mathrm{vec}(X)$ *определяется как вектор-столбец размерности mn, получаемый последовательной записью столбцов матрицы X при их упорядочении слева направо.*

Заметим, что вектор v, определяемый рис. 6.4, может быть записан как $v = \mathrm{vec}(V)$.

Чтобы найти выражение для матрицы $T_{N\times N}$, а также вычислить ее собственные значения и собственные векторы, мы должны ввести понятие *кронекерова произведения*.

Определение 6.2. *Пусть A — матрица размера $m\times n$, а B — матрица размера $p\times q$. Тогда* кронекерово произведение $A\otimes B$ *матриц A и B — это матрица*

$$\begin{bmatrix} a_{1,1}\cdot B & \dots & a_{1,n}\cdot B \\ \vdots & & \vdots \\ a_{m,1}\cdot B & \dots & a_{m,n}\cdot B \end{bmatrix}$$

размера $mp \times nq$.

Следующая лемма показывает, как записать уравнение Пуассона в терминах кронекеровых произведений и оператора $\mathrm{vec}(\cdot)$.

Лемма 6.2. *Пусть A — матрица размера $m \times m$, B — матрица размера $n \times n$, X и C — матрицы размера $m \times n$. Тогда выполняются следующие свойства:*

1. $\mathrm{vec}(AX) = (I_n \otimes A) \cdot \mathrm{vec}(X)$.
2. $\mathrm{vec}(XB) = (B^T \otimes I_m) \cdot \mathrm{vec}(X)$.
3. *Уравнение Пуассона $T_N V + V T_N = h^2 F$ может быть эквивалентным образом записано как*

$$T_{N\times N} \cdot \mathrm{vec}(V) \equiv (I_N \otimes T_N + T_N \otimes I_N) \cdot \mathrm{vec}(V) = h^2 \,\mathrm{vec}(F). \qquad (6.17)$$

Доказательство. Мы докажем только третье утверждение, отнеся два других к вопросу 6.4. Согласно (6.11), уравнение Пуассона может быть записано в форме $T_N V + V T_N = h^2 F$, которая, очевидно, эквивалентна равенству

$$\mathrm{vec}(T_N V + V T_N) = \mathrm{vec}(T_N V) + \mathrm{vec}(V T_N) = \mathrm{vec}(h^2 F).$$

Согласно первому утверждению леммы,

$$\mathrm{vec}(T_N V) = (I_N \otimes T_N)\,\mathrm{vec}(V).$$

Используя второе утверждение леммы и симметрию матрицы T_N, имеем

$$\mathrm{vec}(V T_N) = (T_N^T \otimes I_N)\,\mathrm{vec}(V) = (T_N \otimes I_N)\,\mathrm{vec}(V).$$

Сложение двух последних равенств завершает доказательство утверждения 3. □

Читатель может проверить, что выражение

$$T_{N\times N} = I_N \otimes T_N + T_N \otimes I_N = \begin{bmatrix} T_N & & & \\ & \ddots & & \\ & & \ddots & \\ & & & T_N \end{bmatrix} + \begin{bmatrix} 2I_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & \ddots & -I_N \\ & & -I_N & 2I_N \end{bmatrix}$$

из уравнения (6.17) находится в согласии с формулой (6.17)[1].

Для вычисления собственных значений матриц, определяемых кронекеровыми произведениями, таких, как матрица $T_{N\times N}$, нам понадобится следующая лемма (ее доказательство также составляет часть вопроса 6.4):

Лемма 6.3. *Справедливы следующие утверждения относительно кронекеровых произведений:*

1. *Пусть определены произведения $A \cdot C$ и $B \cdot D$. Тогда $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$.*
2. *Если матрицы A и B обратимы, то $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.*
3. $(A \otimes B)^T = A^T \otimes B^T$.

[1] С помощью этого равенства матрицу $T_{N\times N}$ можно вычислить в две строки текста на Matlab'e:

```
TN = 2*eye(N) - diag(ones(N-1,1),1) - diag(ones(N-1,1),-1);
TNxN = kron(eye(N),TN) + kron(eye(TN),N));
```

Предложение 6.1. *Пусть $T_N = Z\Lambda Z^T$ есть спектральное разложение матрицы T_N, где $Z = [z_1, \ldots, z_N]$ — ортогональная матрица, составленная по столбцам из собственных векторов матрицы T_N, а $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. Тогда спектральное разложение матрицы $T_{N\times N} = I \otimes T_N + T_N \otimes I$ имеет вид*

$$I \otimes T_N + T_N \otimes I = (Z \otimes Z) \cdot (I \otimes \Lambda + \Lambda \otimes I) \cdot (Z \otimes Z)^T. \qquad (6.18)$$

В диагональной матрице $I\otimes\Lambda+\Lambda\otimes I$ диагональный элемент с номером $iN+j$ есть собственное значение матрицы $T_{N\times N}$, соответствующее паре индексов (i,j) и равное $\lambda_{i,j} = \lambda_i + \lambda_j$. Отвечающий этому собственному значению собственный вектор $z_i \otimes z_j$ является столбцом с номером $iN + j$ в ортогональной матрице $Z \otimes Z$.

Доказательство. Из утверждений 1 и 3 леммы 6.3 легко выводится ортогональность матрицы $Z\otimes Z$. Действительно, $(Z\otimes Z)(Z\otimes Z)^T = (Z\otimes Z)(Z^T\otimes Z^T) = (Z \cdot Z^T) \otimes (Z \cdot Z^T) = I \otimes I = I$. Мы можем теперь проверить равенство (6.18):

$$\begin{aligned}
&(Z \otimes Z) \cdot (I \otimes \Lambda + \Lambda \otimes I) \cdot (Z \otimes Z)^T \\
&\qquad = (Z \otimes Z) \cdot (I \otimes \Lambda + \Lambda \otimes I) \cdot (Z^T \otimes Z^T) \\
&\qquad\qquad \text{согласно утверждению 3 леммы 6.3} \\
&\qquad = (Z \cdot I \cdot Z^T) \otimes (Z \cdot \Lambda \cdot Z^T) + (Z \cdot \Lambda \cdot Z^T) \otimes (Z \cdot I \cdot Z^T) \\
&\qquad\qquad \text{согласно утверждению 1 леммы 6.3} \\
&\qquad = (I) \otimes (T_N) + (T_N) \otimes (I) \\
&\qquad = T_{N\times N}.
\end{aligned}$$

Кроме того, легко видеть, что матрица $I \otimes \Lambda + \Lambda \otimes I$ диагональная, а ее диагональный элемент с номером $iN + j$ равен $\lambda_i + \lambda_j$. Таким образом, равенство (6.18) действительно есть спектральное разложение матрицы $T_{N\times N}$. Наконец, из определения кронекерова произведения следует, что столбцом с номером $iN + j$ в матрице $Z \otimes Z$ является $z_i \otimes z_j$. □

Читатель может проверить, что $z_i\otimes z_j = \mathrm{vec}(z_j z_i^T)$, что находится в согласии с выражением для собственного вектора в уравнении (6.12).

По поводу обобщения предложения 6.1 на случай матрицы $A \otimes I + B^T \otimes I$, возникающей при решении уравнения Сильвестра $AX+XB = C$, см. вопрос 6.5 (а также вопрос 4.6).

Сходным образом, трехмерное уравнение Пуассона приводит к матрице

$$T_{N\times N\times N} \equiv T_N \otimes I_N \otimes I_N + I_N \otimes T_N \otimes I_N + I_N \otimes I_N \otimes T_N.$$

Ее собственными значениями являются всевозможные тройные суммы собственных значений матрицы T_N, а $Z \otimes Z \otimes Z$ есть матрица собственных векторов. Уравнения Пуассона для еще более высокой размерности могут быть представлены в аналогичной форме.

6.4. Краткая сводка методов для решения уравнения Пуассона

В таблице 6.1 указаны затраты при решении модельной задачи на сетке $N \times N$ различными прямыми и итерационными методами. Число неизвестных n в за-

даче равно N^2. При сравнении затрат нужно проявлять известную осмотрительность, поскольку (в отсутствие ошибок округления) прямые методы дают точный ответ, тогда как итерационные методы находят лишь приближенное решение; в то же время приближение невысокой точности может быть вычислено посредством итерационного метода дешевле, чем очень хорошее приближение. Поэтому мы сравниваем затраты в предположении, что число операций итерационного метода достаточно велико для того, чтобы снизить погрешность до некоторой фиксированной малой величины[1] (скажем, 10^{-6}).

Метод	Время	Память	Прямой или итерационный	Раздел
Холесского для плотных матриц	n^3	n^2	П	2.7.1
Явного обращения	n^2	n^2	П	
Холесского для ленточных матриц	n^2	$n^{3/2}$	П	2.7.3
Якоби	n^2	n	И	6.5
Гаусса—Зейделя	n^2	n	И	6.5
Холесского для разреженных матриц	$n^{3/2}$	$n \cdot \log n$	П	2.7.4
Сопряженных градиентов	$n^{3/2}$	n	И	6.6
Последовательной верхней релаксации	$n^{3/2}$	n	И	6.5
Чебышевское ускорение с SSOR	$n^{5/4}$	n	И	6.5
Быстрое преобразование Фурье	$n \cdot \log n$	n	П	6.7
Блочная циклическая редукция	$n \cdot \log n$	n	П	6.8
Многосеточный	n	n	И	6.9
Нижняя оценка	n	n		

Таблица 6.1. Сравнительная сложность решения уравнения Пуассона на сетке $N \times N (n = N^2)$.

Второй и третий столбцы табл. 6.1 показывают число арифметических операций (или время) и память, необходимые для метода на последовательной машине. В столбце 4 отмечено, является ли метод прямым (П) или итерационным (И). Для всех элементов таблицы указан лишь их порядок; константы, входящие в символ O, зависят от деталей реализации и критерия останова, принятого для итерационных методов (например, погрешность не превосходит 10^{-6}). Поэтому, скажем, данные для метода Холесского верны и для гауссова исключения, так как (вдвое) меняются лишь константы в оценках. Последний столбец указывает раздел книги, в котором обсуждается данный алгоритм.

[1] Другая возможность состоит в том, чтобы продолжать итерации до тех пор, пока погрешность не станет величиной порядка $O(h^2) = O((N+1)^{-2})$, т. е. того же порядка, что и погрешность аппроксимации. Можно показать, что в этом случае стоимости итерационных методов, указанные в табл. 6.1, приобрели бы множитель $O(\log n)$.

Методы перечислены в порядке возрастания их скорости от самого медленного (алгоритма Холесского для плотных матриц) до самого быстрого (многосеточный метод); таблица заканчивается нижней оценкой, справедливой для любого метода. Нижняя оценка n объясняется тем, что по крайней мере одна операция нужна для вычисления каждой компоненты решения; в противном случае, не все компоненты различны, что возможно лишь при специальных входных данных. Упорядочение методов также примерно соответствует убыванию их общности: плотная версия метода Холесского применима к любой симметричной положительно определенной матрице, а прочие алгоритмы приложимы (или, по крайней мере, имеют гарантированную сходимость) лишь к специальным классам матриц. В последующих разделах область применимости различных методов описана более подробно.

Под алгоритмом «явного обращения» понимается предварительное вычисление матрицы $T_{N\times N}^{-1}$ в явном виде, после чего вектор $v = T_{N\times N}^{-1} f$ находится посредством матрично-векторного умножения (операции по вычислению $T_{N\times N}^{-1}$ не включаются в общую стоимость алгоритма). Подобно методу Холесского для плотных матриц, в данном алгоритме требуется n^2 слов памяти, что намного больше, чем во всех остальных методах; поэтому алгоритм не удовлетворителен. Метод Холесского для ленточных матриц обсуждался в разд. 2.7.3; это попросту вариант алгоритма Холесского, использующий то обстоятельство, что элементы вне ленты шириной в $2N + 1$ диагоналей не требуют хранения и обработки.

Методы Якоби и Гаусса—Зейделя — это классические итерационные методы; они не особенно быстры, но на них основаны другие, более быстрые методы, такие, как последовательная верхняя релаксация, симметричная последовательная верхняя релаксация и многосеточный метод, быстрейший из рассматриваемых нами алгоритмов. Методы Якоби и Гаусса—Зейделя довольно подробно исследуются в разд. 6.5.

Метод Холесского для разреженных матриц — это алгоритм, обсуждавшийся в разд. 2.7.4. Он представляет собой реализацию алгоритма Холесского, в которой нулевые элементы матрицы $T_{N\times N}$ или ее множителя Холесского не хранятся и не обрабатываются. Кроме того, предполагается, что строки и столбцы в $T_{N\times N}$ были «оптимально упорядочены» с тем, чтобы минимизировать работу и память (для чего используется метод вложенных сечений [112, 113]). Хотя разреженный метод Холесского достаточно быстр для двумерного уравнения Пуассона, он значительно хуже работает для трехмерной задачи (требуя времени $O(N^6) = O(n^2)$ и памяти $O(N^4) = O(n^{4/3})$); причина в том, что происходит большее «заполнение» нулевых элементов.

Алгоритм сопряженных градиентов представляет обширный класс методов, называемых методами *крыловского подпространства* и широко используемых как при решении линейных систем, так и при нахождении собственных значений разреженных матриц. Эти методы более подробно обсуждаются в разд. 6.6.

Наиболее быстрые методы — это блочная циклическая редукция, быстрое преобразование Фурье (FFT) и многосеточный метод. В частности, в многосеточном методе выполняется лишь $O(1)$ операций на компоненту решения, что асимптотически является оптимальным результатом.

В заключение этого обзора предупредим читателя, что таблица не дает полной картины, поскольку в ней отсутствуют константы. Для задачи конкретного

порядка и конкретной машины из таблицы нельзя усмотреть непосредственно, какой метод работает быстрее всего. Все же ясно, что для достаточно больших n такие итерационные методы, как алгоритмы Якоби, Гаусса—Зейделя, сопряженных градиентов и последовательной верхней релаксации уступают методам FFT и блочной циклической редукции, а также многосеточному методу. Тем не менее интерес к этим алгоритмам сохраняется, поскольку из них строятся некоторые более быстрые методы, а также потому, что они применимы к большему кругу задач по сравнению с быстрыми методами.

Все указанные алгоритмы могут быть реализованы как параллельные; см. детали в лекциях, помещенных на PARALLEL_HOMEPAGE. Интересно, что для некоторых параллельных машин многосеточный метод может оказаться не самым быстрым. Это объясняется тем, что на параллельном компьютере время, требуемое для обмена данными между отдельными процессорами, может быть сравнимо с временем, затрачиваемым на операции с плавающей точкой, и для некоторых алгоритмов объем коммуникаций может быть меньше, чем в многосеточном методе.

6.5. Основные итерационные методы

В этом разделе мы обсудим следующие основные итерационные методы: метод Якоби, метод Гаусса—Зейделя, метод последовательной верхней релаксации (SOR(ω)), чебышевское ускорение с симметричной последовательной верхней релаксацией (SSOR(ω)). Обсуждение этих методов можно найти и в NETLIB/templates; там же даны их реализации.

По заданному x_0 каждый из названных методов генерирует последовательность $\{x_m\}$, сходящуюся к решению $A^{-1}b$ системы $Ax = b$; при этом вектор x_{m+1} несложно вычисляется по вектору x_m.

Определение 6.3. Расщеплением *матрицы A называется всякое представление вида $A = M - K$, где матрица M невырожденна.*

Всякое расщепление порождает итерационный метод следующим образом: из равенства $Ax = Mx - Kx = b$ находим $Mx = Kx + b$, или $x = M^{-1}Kx + M^{-1}b \equiv Rx + c$. Полагая $x_{m+1} = Rx_m + c$, приходим к искомому итерационному методу. Определим условия его сходимости.

Лемма 6.4. *Пусть $\|\cdot\|$ — произвольная операторная норма (т. е. $\|R\| \equiv \max_{x\neq 0} \frac{\|Rx\|}{\|x\|}$). Если $\|R\| < 1$, то метод $x_{m+1} = Rx_m + c$ сходится при любом x_0.*

Доказательство. Вычитая равенство $x = Rx + c$ из $x_{m+1} = Rx_m + c$, получаем $x_{m+1} - x = R(x_m - x)$. Отсюда $\|x_{m+1} - x\| \le \|R\|\|x_m - x\| \le \|R\|^{m+1}\|x_0 - x\|$. Правая часть сходится к нулю, так как $\|R\| < 1$. □

Наш окончательный критерий сходимости опирается на приводимое ниже свойство матрицы R.

Определение 6.4. Спектральным радиусом *матрицы R называется число $\rho(R) \equiv \max|\lambda|$, где максимум берется по всем собственным значениям λ этой матрицы.*

Лемма 6.5. *Для любой операторной нормы справедливо неравенство* $\rho(R) \le \|R\|$. *Для любой матрицы* R *и любого числа* $\epsilon > 0$ *найдется операторная норма* $\|\cdot\|_\star$, *такая, что* $\|R\|_\star \le \rho(R) + \epsilon$. *Норма* $\|\cdot\|_\star$ *зависит от* R *и* ϵ.

Доказательство. Покажем, что $\rho(R) \le \|R\|$, какова бы ни была операторная норма. Пусть x — собственный вектор для числа λ, такого, что $\rho(R) = \lambda$; тогда $\|R\| = \max_{y\neq 0} \frac{\|Ry\|}{\|y\|} \ge \frac{\|Rx\|}{\|x\|} = \frac{\|\lambda x\|}{\|x\|} = |\lambda|$.

Построим теперь операторную норму $\|\cdot\|_\star$, такую, что $\|R\|_\star \le \rho(R) + \epsilon$. Пусть матрица $S^{-1}RS = J$ имеет жорданову форму. Положим $D_\epsilon = \operatorname{diag}(1, \epsilon, \epsilon^2, \ldots, \epsilon^{n-1})$. Тогда

$$(SD_\epsilon)^{-1}R(SD_\epsilon) = D_\epsilon^{-1}JD_\epsilon$$

$$= \left[\begin{array}{cccc|cccc|c} \lambda_1 & \epsilon & & & & & & & \\ & \ddots & \ddots & & & & & & \\ & & \ddots & \epsilon & & & & & \\ & & & \lambda_1 & & & & & \\ \hline & & & & \lambda_2 & \epsilon & & & \\ & & & & & \ddots & \ddots & & \\ & & & & & & \ddots & \epsilon & \\ & & & & & & & \lambda_2 & \\ \hline & & & & & & & & \ddots \end{array}\right].$$

Это «жорданова форма» с числом ϵ над главной диагональю. Теперь с помощью векторной нормы $\|x\|_\star \equiv \|(SD_\epsilon)^{-1}x\|_\infty$ построим операторную норму

$$\begin{aligned} \|R\|_\star &\equiv \max_{x\neq 0} \frac{\|Rx\|_\star}{\|x\|_\star} \\ &= \max_{x\neq 0} \frac{\|(SD_\epsilon)^{-1}Rx\|_\infty}{\|(SD_\epsilon)^{-1}x\|_\infty} \\ &= \max_{y\neq 0} \frac{\|(SD_\epsilon)^{-1}R(SD_\epsilon)y\|_\infty}{\|y\|_\infty} \\ &= \|(SD_\epsilon)^{-1}R(SD_\epsilon)\|_\infty \\ &\le \max_i |\lambda_i| + \epsilon \\ &= \rho(R) + \epsilon. \end{aligned}$$

□

Теорема 6.1. *Итерационный процесс* $x_{m+1} = Rx_m + c$ *тогда и только тогда сходится к решению системы* $Ax = b$, *каков бы ни был начальный вектор* x_0, *когда* $\rho(R) < 1$.

Доказательство. При $\rho(R) \ge 1$ выберем x_0 так, чтобы $x_0 - x$ был собственным вектором матрицы R для собственного значения λ, где $|\lambda| = \rho(R)$. Тогда

$$(x_{m+1} - x) = R(x_m - x) = \cdots = R^{m+1}(x_0 - x) = \lambda^{m+1}(x_0 - x),$$

и эта последовательность не сходится к нулю. При $\rho(R) < 1$ нужно, опираясь на лемму 6.5, выбрать операторную норму так, чтобы выполнялось неравенство $\|R\|_\star < 1$, а затем применить лемму 6.4, устанавливающую сходимость метода. □

Определение 6.5. Скоростью сходимости *процесса* $x_{m+1} = Rx_m + c$ *называется число* $r(R) \equiv -\log_{10} \rho(R)$.

Поскольку $\log_{10} \|x_m - x\|_\star - \log_{10} \|x_{m+1} - x\|_\star \geq r(R) + O(\epsilon)$, то $r(R)$ имеет смысл приращения числа верных десятичных знаков в приближенном решении за одну итерацию. Чем меньше $\rho(R)$, тем выше скорость сходимости, т. е. тем больше число верных десятичных знаков, вычисляемых за итерацию.

Теперь нашей целью будет выбор расщепления $A = M - K$, удовлетворяющего следующим условиям:

1. произведения $Rx = M^{-1}Kx$ и $c = M^{-1}b$ должны легко вычисляться;
2. число $\rho(R)$ мало.

Эти конфликтующие условия нужно сбалансировать. Например, выбор $M = I$ хорош для установки 1), однако условие $\rho(R) < 1$ может в этом случае не быть выполнено. С другой стороны, выбор $M = A$ и $K = 0$ хорош для установки 2), но, скорей всего, плох для 1).

Разбиения для методов, обсуждаемых в данном разделе, используют следующие общие обозначения: если A не имеет нулевых диагональных элементов, то пишем

$$A = D - \widetilde{L} - \widetilde{U} = D(I - L - U), \tag{6.19}$$

где D — диагональ матрицы A, матрица $-\widetilde{L}$ есть нижняя строго треугольная часть в A, причем $DL = \widetilde{L}$, а $-\widetilde{U}$ есть верхняя строго треугольная часть в A и $DU = \widetilde{U}$.

6.5.1. Метод Якоби

Метод Якоби можно описать как циклический обход уравнений с изменением текущей переменной j так, чтобы j-е уравнение удовлетворялось точно. В обозначениях уравнения (6.19), расщепление для метода Якоби можно представить как $A = D - (\widetilde{L} + \widetilde{U})$. Полагая $R_J \equiv D^{-1}(\widetilde{L} + \widetilde{U}) = L + U$ и $c_J \equiv D^{-1}b$, можем описать шаг метода формулой $x_{m+1} = R_J x_m + c_J$. Чтобы убедиться, что эта формула соответствует нашему первому описанию метода, заметим, что из нее следует $Dx_{m+1} = (\widetilde{L} + \widetilde{U})x_m + b$, или $a_{jj}x_{m+1,j} = -\sum_{k \neq j} a_{jk}x_{m,k} + b_j$, или $a_{jj}x_{m+1,j} + \sum_{k \neq j} a_{jk}x_{m,k} = b_j$.

Алгоритм 6.1. *Шаг метода Якоби:*

for $j = 1$ *to* n
  $x_{m+1,j} = \frac{1}{a_{jj}}(b_j - \sum_{k \neq j} a_{jk}x_{m,k})$
end for

В специальном случае модельной задачи реализация алгоритма Якоби упрощается. Будем работать непосредственно с уравнением (6.10) и обозначать

через $v_{m,i,j}$ значение решения в сеточном узле i, j на шаге m. Тогда описание метода Якоби принимает следующий вид:

Алгоритм 6.2. *Шаг метода Якоби для двумерного уравнения Пуассона:*

for $i = 1$ *to* N
  for $j = 1$ *to* N
    $v_{m+1,i,j} = (v_{m,i-1,j} + v_{m,i+1,j} + v_{m,i,j-1} + v_{m,i,j+1} + h^2 f_{ij})/4$
  end for
end for

Иными словами, на каждом шаге новое значение v_{ij} получается «усреднением» значений соседних компонент и числа $h^2 f_{ij}$. Отметим, что все новые значения $v_{m+1,i,j}$ могут вычисляться независимо друг от друга. Действительно, алгоритм 6.2 можно записать одной строкой Matlab'a, если $v_{m+1,i,j}$ хранятся в квадратном массиве $\widehat{V}$, содержащем дополнительные первую и последнюю строки и первый и последний столбцы, заполненные нулями (см. вопрос 6.6).

6.5.2. Метод Гаусса—Зейделя

Мотивация этого метода такова: поскольку на j-м шаге цикла в методе Якоби имеются уточненные значения первых $j-1$ компонент решения, их можно использовать при суммировании.

Алгоритм 6.3. *Шаг метода Гаусса—Зейделя:*

for $j = 1$ *to* n

$$x_{m+1,j} = \frac{1}{a_{jj}} \left(b_j - \underbrace{\sum_{k=1}^{j-1} a_{jk} x_{m+1,k}}_{\text{перевычисленные } x} - \underbrace{\sum_{k=j+1}^{n} a_{jk} x_{m,k}}_{\text{старые } x} \right)$$

end for

Для целей последующего анализа мы хотим записать этот алгоритм в виде $x_{m+1} = R_{GS} x_m + c_{GS}$. Для этого заметим, что формулу алгоритма можно переписать как

$$\sum_{k=1}^{j} a_{jk} x_{m+1,k} = -\sum_{k=j+1}^{n} a_{jk} x_{m,k} + b_j. \tag{6.20}$$

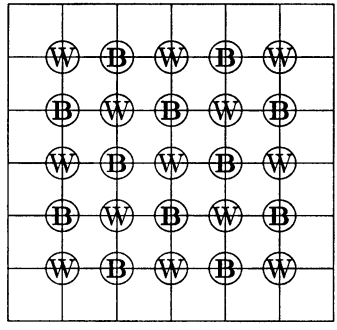
Теперь, используя обозначения уравнения (6.19), перепишем (6.20) как $(D - \widetilde{L}) x_{m+1} = \widetilde{U} x_m + b$, или

$$\begin{aligned} x_{m+1} &= (D - \widetilde{L})^{-1} \widetilde{U} x_m + (D - \widetilde{L})^{-1} b \\ &= (I - L)^{-1} U x_m + (I - L)^{-1} D^{-1} b \\ &\equiv R_{GS} x_m + c_{GS}. \end{aligned}$$

Как и в случае метода Якоби, обсудим, как следует реализовать метод Гаусса—Зейделя для нашей модельной задачи. В принципе, реализация очень похожа с тем исключением, что нужно следить, какие переменные приобрели новые значения (им соответствует индекс $m+1$), а какие сохранили прежние значения (они помечены индексом m). Однако, в зависимости от принятого

порядка обхода сеточных узлов i, j, будут получаться различные (и одинаково законные) реализации метода Гаусса—Зейделя. Не так в методе Якоби, где порядок перевычисления переменных не существенен. Например, если вначале (прежде всех прочих $v_{m,i,j}$) перевычисляется значение $v_{m,1,1}$, то все соседние значения по необходимости старые. Но если переменная $v_{m,1,1}$ перевычисляется последней, то все соседние значения необходимо новые, поэтому для $v_{m,1,1}$ будет получено иное (чем в методе Якоби) значение. В действительности, возможны столько реализаций метода Гаусса—Зейделя, сколько существует способов упорядочить N^2 переменных (а именно, $N^2!$). Два из этих упорядочений представляют особый интерес. Первое из них — это упорядочение, показанное на рис. 6.4; оно называется *естественным упорядочением*.

Второе упорядочение называется *шахматным (или черно-белым)*. Его важность для нас состоит в том, что наиболее сильные результаты о сходимости, устанавливаемые в разд. 6.5.4 и 6.5.5, опираются на это упорядочение. Чтобы описать его, рассмотрим «шахматное» раскрашивание сеточных переменных, показанное на рисунке: узлы, помеченные символом Ⓑ, соответствуют черным квадратам шахматной доски, а узлы, помеченные символом Ⓦ, соответствуют белым квадратам.



При шахматном упорядочении белые узлы нумеруются прежде, чем черные. Заметим, что смежны с каждым белым узлом лишь черные узлы. Поэтому, если вначале перевычисляются компоненты из белых узлов, используются только старые значения из черных узлов. Затем, когда перевычисляются компоненты из черных узлов, которые смежны лишь с белыми узлами, будут использоваться лишь новые значения из этих белых узлов. В результате алгоритм принимает следующий вид.

Алгоритм 6.4. *Шаг метода Гаусса—Зейделя для двумерного уравнения Пуассона при красно-черном упорядочении:*

для всех белых узлов i, j (Ⓦ)

$$v_{m+1,i,j} = (v_{m,i-1,j} + v_{m,i+1,j} + v_{m,i,j-1} + v_{m,i,j+1} + h^2 f_{ij})/4$$

end for

для всех черных узлов i, j (Ⓑ)

$$v_{m+1,i,j} = (v_{m+1,i-1,j} + v_{m+1,i+1,j} + v_{m+1,i,j-1} + v_{m+1,i,j+1} + h^2 f_{ij})/4$$

end for

6.5.3. Последовательная верхняя релаксация

Этот метод будем обозначать через SOR(ω), где ω — *параметр релаксации*. Его идея состоит в том, чтобы улучшить цикл Гаусса—Зейделя подходящим взвешенным усреднением значений $x_{m+1,j}$ и $x_{m,j}$:

$$x_{m+1,j} \text{ для метода SOR } = (1-\omega)x_{m,j} + \omega x_{m+1,j}.$$

Это приводит к следующему алгоритму:

Алгоритм 6.5. *Метод* SOR*:*

for $j = 1$ *to* n

$$x_{m+1,j} = (1-\omega)x_{m,j} + \frac{\omega}{a_{jj}}\left[b_j - \sum_{k=1}^{j-1} a_{jk}x_{m+1,k} - \sum_{k=j+1}^{n} a_{jk}x_{m,k}\right]$$

end for

Эту формулу можно переписать в виде ($j = 1, \ldots, n$)

$$a_{jj}x_{m+1,j} + \omega\sum_{k=1}^{j-1} a_{jk}x_{m+1,k} = (1-\omega)a_{jj}x_{m,j} - \omega\sum_{k=j+1}^{n} a_{jk}x_{m,k} + \omega b_j,$$

или, снова используя обозначения уравнения (6.19),

$$(D - \omega\widetilde{L})x_{m+1} = ((1-\omega)D + \omega\widetilde{U})x_m + \omega b,$$

или

$$\begin{aligned} x_{m+1} &= (D-\omega\widetilde{L})^{-1}((1-\omega)D + \omega\widetilde{U})x_m + \omega(D-\omega\widetilde{L})^{-1}b \\ &= (I-\omega L)^{-1}((1-\omega)I + \omega U)x_m + \omega(I-\omega L)^{-1}D^{-1}b \\ &\equiv R_{\mathrm{SOR}(\omega)}x_m + c_{\mathrm{SOR}(\omega)}. \end{aligned} \tag{6.21}$$

В зависимости от значения ω, будем различать три случая: $\omega = 1$ соответствует методу Гаусса—Зейделя, при $\omega < 1$ говорят о *нижней релаксации*, а при $\omega > 1$ — о *верхней релаксации*. Несколько упрощенный аргумент в пользу верхней релаксации состоит в следующем: если направление от x_m к x_{m+1} удачно в смысле продвижения к точному решению, то имеет смысл пройти по нему в $\omega(>1)$ раз больший путь.

В следующих двух разделах мы покажем, как выбрать оптимальное ω для модельной задачи. Эта оптимальность опирается на использование красно-черного упорядочения.

Алгоритм 6.6. *Шаг метода* SOR(ω) *для двумерного уравнения Пуассона при шахматном упорядочении:*

для всех белых узлов i, j (Ⓦ)

$$v_{m+1,i,j} = (1-\omega)v_{m,i,j} + \omega(v_{m,i-1,j} + v_{m,i+1,j} + v_{m,i,j-1} + v_{m,i,j+1} + h^2 f_{ij})/4$$

end for

для всех черных узлов i, j (Ⓑ)

$$v_{m+1,i,j} = (1-\omega)v_{m,i,j} + \omega(v_{m+1,i-1,j} + v_{m+1,i+1,j} + v_{m+1,i,j-1} + v_{m+1,i,j+1} + h^2 f_{ij})/4$$

end for

6.5.4. Сходимость методов Якоби, Гаусса—Зейделя и SOR(ω) для модельной задачи

Скорость сходимости метода Якоби для модельной задачи определить легко. Соответствующее расщепление имеет вид $T_{N\times N} = 4I - (4I - T_{N\times N})$, поэтому $R_J = (4I)^{-1}(4I - T_{N\times N}) = I - T_{N\times N}/4$. Таким образом, собственными значениями матрицы R_J являются числа $1 - \lambda_{i,j}/4$, где $\lambda_{i,j}$ — собственные значения матрицы $T_{N\times N}$:

$$\lambda_{i,j} = \lambda_i + \lambda_j = 4 - 2\left(\cos\frac{\pi i}{N+1} + \cos\frac{\pi j}{N+1}\right).$$

Наибольшее из чисел $|1 - \lambda_{i,j}/4|$ есть $\rho(R_J)$, а именно,

$$\rho(R_J) = |1 - \lambda_{1,1}/4| = |1 - \lambda_{N,N}/4| = \cos\frac{\pi}{N+1} \approx 1 - \frac{\pi^2}{2(N+1)^2}.$$

Заметим, что по мере роста N и ухудшения обусловленности T спектральный радиус $\rho(R_J)$ приближается к 1. Поскольку на каждом шаге погрешность умножается на спектральный радиус, сходимость метода замедляется. Чтобы более точно оценить скорость сходимости, вычислим число m итераций Якоби, необходимых для уменьшения погрешности в e раз. Число m должно удовлетворять условию $(\rho(R_J))^m = e^{-1}$, или $(1 - \frac{\pi^2}{2(N+1)^2})^m = e^{-1}$, или $m \approx \frac{2(N+1)^2}{\pi^2} = O(N^2) = O(n)$. Таким образом, число итераций пропорционально числу неизвестных. Поскольку на одном шаге метода требуется $O(1)$ операций для перевычисления одной компоненты решения и $O(n)$ операций для перевычисления всех компонент, стоимость снижения погрешности в e раз (или в любое постоянное число раз, большее единицы) составляет $O(n^2)$. Это объясняет элемент табл. 6.1, соответствующий методу Якоби.

Отмеченная зависимость имеет общий характер: чем хуже обусловлена исходная задача, тем медленнее сходится большинство итерационных методов. Важными исключениями из этого правила являются многосеточный метод и метод декомпозиции области; мы обсудим их позже.

В следующем разделе мы покажем, что $\rho(R_{GS}) = \rho(R_J)^2 = \cos^2\frac{\pi}{N+1}$ при условии, что переменные в уравнении Пуассона пронумерованы в соответствии с шахматным упорядочением (см. алгоритм 6.4 и следствие 6.1). Иначе говоря, уменьшение погрешности на одном шаге метода Гаусса—Зейделя равно уменьшению на двух шагах метода Якоби. Это соотношение является общим для матриц, возникающих при аппроксимации дифференциальных уравнений посредством некоторых конечно-разностных схем. Оно также объясняет элемент табл. 6.1, относящийся к методу Гаусса—Зейделя: поскольку этот метод лишь вдвое быстрее метода Якоби, он имеет ту же сложность в смысле символа $O(\cdot)$.

При том же самом шахматном порядке пересчета мы покажем (см. алгоритм 6.6 и теорему 6.7), что при значении параметра релаксации $1 < \omega = 2/(1 + \sin\frac{\pi}{N+1}) < 2$ справедливы соотношения

$$\rho(R_{\mathrm{SOR}(\omega)}) = \frac{\cos^2\frac{\pi}{N+1}}{(1+\sin\frac{\pi}{N+1})^2} \approx 1 - \frac{2\pi}{N+1} \text{ для больших } N.$$

Они контрастируют с равенством $\rho(R) = 1 - O(\frac{1}{N^2})$ для матриц R_J и R_{GS}. Указанное значение ω оптимально, т. е. оно минимизирует $\rho(R_{\mathrm{SOR}(\omega)})$. При таком выборе ω метод SOR(ω) приблизительно в N раз быстрее, чем методы Якоби и Гаусса—Зейделя. Действительно, если j шагов SOR(ω) уменьшают погрешность в такой же мере, как k шагов методов Якоби и Гаусса—Зейделя, то $(1 - \frac{1}{N^2})^k \approx (1 - \frac{1}{N})^j$, откуда следует, что $1 - \frac{k}{N^2} \approx 1 - \frac{j}{N}$, или $k \approx jN$. Это снижает сложность метода SOR(ω) с $O(n^2)$ до $O(n^{3/2})$, как и показано в табл. 6.1.

В следующем разделе для некоторых конечно-разностных матриц будет указан общий метод выбора значения ω, минимизирующего $\rho(R_{\mathrm{SOR}(\omega)})$.

6.5.5. Более подробно о критериях сходимости методов Якоби, Гаусса—Зейделя и SOR(ω)

Мы дадим ряд критериев, гарантирующих сходимость названных методов. Первый критерий прост для проверки, но не всегда применим; в частности, он не применим к модельной задаче. Затем мы укажем несколько более сложных критериев, накладывающих более сильные условия на матрицу A, но зато дающих больше информации о сходимости. Эти более сложные критерии «скроены по мерке» матриц, возникающих при дискретизации некоторых дифференциальных уравнений с частными производными, таких, как уравнение Пуассона.

Приведем сводку результатов данного раздела:

1. Если A — матрица со строгим диагональным преобладанием по строкам (см. определение 6.6), то методы Якоби и Гаусса—Зейделя сходятся, причем второй метод сходится быстрее (теорема 6.2). Строгое диагональное преобладание по строкам означает, что модуль каждого диагонального элемента в A больше суммы модулей прочих элементов той же строки.
2. Указанный результат не приложим к нашей модельной задаче, поскольку в ней строгое диагональное преобладание не имеет места. Поэтому мы предполагаем более слабую форму диагонального преобладания (определение 6.11), но накладываем условие, называемое *неразложимостью*, на характер расположения ненулевых элементов в A (определение 6.7), что позволяет нам доказать сходимость методов Якоби и Гаусса—Зейделя. Второй метод снова сходится быстрее (теорема 6.3). Этот результат применим к модельной задаче.
3. В случае метода SOR(ω) мы показываем, что неравенство $0 < \omega < 2$ необходимо для сходимости (теорема 6.4). Если, к тому же, A — положительно определенная матрица (что имеет место в модельной задаче), то неравенство $0 < \omega < 2$ и достаточно для сходимости (теорема 6.5).
4. Чтобы дать количественное сравнение методов Якоби, Гаусса—Зейделя и SOR(ω), сделаем еще одно предположение о позициях ненулевых элементов в A. Соответствующее свойство называется *свойством* A (определение 6.12) и равносильно требованию, чтобы *граф матрицы* был *двудольным*. По существу, свойство A означает, что переменные можно пересчитывать, пользуясь красно-черным упорядочением. При наличии свойства A справедлива простая алгебраическая формула, связывающая собственные значения матриц R_J, R_{GS} и $R_{\mathrm{SOR}(\omega)}$ (теорема 6.6); это позволяет нам срав-

нить скорости сходимости методов. Эта формула также дает возможность вычислить оптимальное ω, обеспечивающее наиболее быструю сходимость метода SOR(ω) (теорема 6.7).

Определение 6.6. *Матрица A имеет* строгое диагональное преобладание по строкам, *если* $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ *для всех* i.

Теорема 6.2. *Для матрицы A со строгим диагональным преобладанием по строкам методы Якоби и Гаусса—Зейделя сходятся. При этом* $\|R_{GS}\|_\infty \leq \|R_J\|_\infty < 1$.

Неравенство $\|R_{GS}\|_\infty \leq \|R_J\|_\infty$ означает, что в задаче, наихудшей для метода Гаусса—Зейделя, погрешность снижается за один шаг не меньше, чем в задаче, наихудшей для метода Якоби. Оно *не* гарантирует, что для любой конкретной системы $Ax = b$ метод Гаусса—Зейделя сходится быстрее метода Якоби; «случайно» может оказаться, что на каком-то шаге ошибка в методе Якоби меньше.

Доказательство. Снова используя обозначения уравнения (6.19), запишем $R_J = L + U$ и $R_{GS} = (I - L)^{-1}U$. Мы хотим доказать, что

$$\|R_{GS}\|_\infty = \||R_{GS}|e\|_\infty \leq \||R_J|e\|_\infty = \|R_J\|_\infty, \tag{6.22}$$

где $e = [1, \ldots 1]^T$ — вектор, состоящий из единиц. Неравенство (6.22) заведомо будет верно, если мы сможем доказать более сильное покомпонентное неравенство

$$|(I - L)^{-1}U| \cdot e = |R_{GS}| \cdot e \leq |R_J| \cdot e = (|L| + |U|) \cdot e. \tag{6.23}$$

Так как

$$\begin{aligned}
|(I - L)^{-1}U| \cdot e &\leq |(I - L)^{-1}| \cdot |U| \cdot e && \text{по неравенству треугольника} \\
&= \left| \sum_{i=0}^{n-1} L^i \right| \cdot |U| \cdot e && \text{поскольку } L^n = 0 \\
&\leq \sum_{i=0}^{n-1} |L|^i \cdot |U| \cdot e && \text{по неравенству треугольника} \\
&= (I - |L|)^{-1} \cdot |U| \cdot e && \text{поскольку } |L|^n = 0,
\end{aligned}$$

то неравенство (6.23) будет выполняться, если мы сумеем установить еще более сильное покомпонентное неравенство

$$(I - |L|)^{-1} \cdot |U| \cdot e \leq (|L| + |U|) \cdot e. \tag{6.24}$$

Все элементы матрицы $(I - |L|)^{-1} = \sum_{i=0}^{n-1} |L|^i$ неотрицательны, поэтому неравенство (6.24) будет справедливо, если мы докажем, что

$$|U| \cdot e \leq (I - |L|) \cdot (|L| + |U|) \cdot e = (|L| + |U| - |L|^2 - |L| \cdot |U|) \cdot e,$$

или

$$0 \leq (|L| - |L|^2 - |L| \cdot |U|) \cdot e = |L| \cdot (I - |L| - |U|) \cdot e. \tag{6.25}$$

В силу неотрицательности элементов матрицы $|L|$, неравенство (6.25) будет верно, если доказать, что

$$0 \le (I - |L| - |U|) \cdot e, \quad \text{или} \quad |R_J| \cdot e = (|L| + |U|)e \le e. \tag{6.26}$$

Наконец, неравенство (6.26) верно, так как, по предположению, $\||R_J| \cdot e\|_\infty = \|R_J\|_\infty = \rho < 1$. □

Аналогичный результат справедлив, если A имеет строгое диагональное преобладание по столбцам (т. е. A^T имеет строгое диагональное преобладание по строкам).

Читатель легко проверит, что этот простой критерий не приложим к модельной задаче. Поэтому нам придется ослабить предположение о строгом диагональном преобладании. Для этого потребуется учесть *теоретико-графовые свойства* матрицы.

Определение 6.7. *Матрица A называется* неразложимой, *если не существует матрицы-перестановки P, такой, что*

$$PAP^T = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right].$$

Мы вскоре увидим, как это определение связано с *теорией графов*.

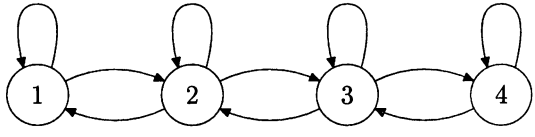
Определение 6.8. Ориентированный граф *представляет собой конечное множество* узлов, *соединенных конечным числом* ориентированных ребер, *т. е. стрелок, ведущих из одного узла в другой.* Путь *в ориентированном графе — это последовательность узлов $n_1, \ldots, n_m$ вместе с ребрами, направленными из каждого n_i в n_{i+1}.* Петлей *называется ребро, ведущее из данного узла к нему самому.*

Определение 6.9. Ориентированный граф $G(A)$ матрицы A *— это граф с узлами $1, 2, \ldots, n$, где из узла i в узел j ведет ребро в том и только том случае, если $a_{ij} \neq 0$.*

Пример 6.1. Матрица

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{bmatrix}$$

имеет направленный граф ◇



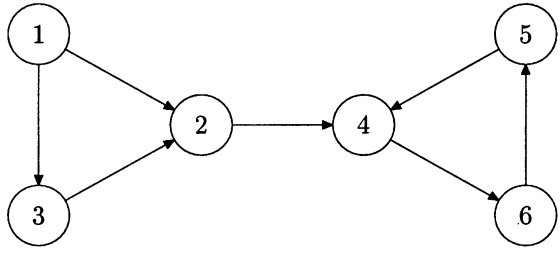
Определение 6.10. *Ориентированный граф* сильно связен, *если в нем существует путь, ведущий из любого узла i в любой узел j.* Сильно связной компонентой *ориентированного графа называется подграф (т. е. подмножество узлов вместе с соединяющими их ребрами), который сильно связен, но не может быть вложен в больший сильно связный подграф.*

Пример 6.2. Граф примера 6.1 сильно связен. ◇

Пример 6.3. Матрица

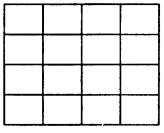
$$A = \left[\begin{array}{cc|cc} 1 & 1 & 1 & \\ 1 & & & \\ \hline & & & 1 \\ & & 1 & \\ & & & 1 \end{array}\right]$$

имеет ориентированный граф



Поскольку в этом графе нет пути из произвольного узла в узел 1, он не является сильно связным. Узлы 4,5 и 6 образуют сильно связную компоненту, так как для любого из них существует путь, ведущий в любой другой узел. ◇

Пример 6.4. Граф модельной задачи сильно связен. По существу, этот граф имеет вид



с той оговоркой, что каждое ребро сетки представляет два ребра графа (по одному в каждом направлении); кроме того, не показаны петли. ◇

Лемма 6.6. *Для того чтобы матрица A была неразложима, необходимо и достаточно, чтобы граф $G(A)$ был сильно связен.*

Доказательство. Если $A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$ — разложимая матрица, то, очевидно, не существует способа достичь узлов, соответствующих блоку A_{11}, из узлов, соответствующих A_{22}, т. е. граф $G(A)$ не будет сильно связен. Точно так же, если $G(A)$ не является сильно связным графом, перенумеруем строки (и столбцы) матрицы так, чтобы первые номера приобрели узлы некоторой сильно связной компоненты. Тогда матрица PAP^T приобретет верхний блочно-треугольный вид. □

Пример 6.5. Матрица A примера 6.3 разложима. ◇

Определение 6.11. *Матрица A имеет* слабое диагональное преобладание по строкам, *если $|a_{ii}| \geq \sum_{k \neq i} |a_{ik}|$ для всех i и хотя бы для одного i имеет место строгое неравенство.*

Теорема 6.3. *Пусть матрица A неразложима и имеет слабое диагональное преобладание по строкам. Тогда методы Якоби и Гаусса—Зейделя сходятся, причем $\rho(R_{GS}) < \rho(R_J) < 1$.*

Доказательство этой теоремы см. в [249].

Пример 6.6. Матрица модельной задачи имеет слабое диагональное преобладание и неразложима, но не имеет сильного диагонального преобладания. (Диагональные элементы равны 4, а суммы модулей внедиагональных элементов равны 2, 3 либо 4.) Таким образом, методы Якоби и Гаусса—Зейделя сходятся для модельной задачи. ◇

Хотя факты, приведенные выше, показывают, что при определенных условиях метод Гаусса—Зейделя быстрее метода Якоби, в общем случае такой результат не верен. Существуют несимметричные матрицы, для которых метод Якоби сходится, а метод Гаусса—Зейделя расходится, а также матрицы, для которых сходится метод Гаусса—Зейделя, но расходится метод Якоби [249].

Исследуем теперь сходимость метода $SOR(\omega)$ [249]. Напомним определение метода:

$$R_{\mathrm{SOR}(\omega)} = (I - \omega L)^{-1}((1-\omega)I + \omega U).$$

Теорема 6.4. *Имеет место неравенство $\rho(R_{\mathrm{SOR}(\omega)}) \geq |\omega - 1|$. Поэтому неравенство $0 < \omega < 2$ необходимо для сходимости метода.*

Доказательство. Запишем характеристический многочлен матрицы $R_{\mathrm{SOR}(\omega)}$ в виде $\varphi(\lambda) = \det(\lambda I - R_{\mathrm{SOR}(\omega)}) = \det((I - \omega L)(\lambda I - R_{\mathrm{SOR}(\omega)})) = \det((\lambda + \omega - 1)I - \omega\lambda L - \omega U)$; тогда

$$\varphi(0) = \pm \prod_{i=1}^{n} \lambda_i(R_{\mathrm{SOR}(\omega)}) = \pm \det((\omega - 1)I) = \pm(\omega - 1)^n,$$

откуда $\max_i |\lambda_i(R_{\mathrm{SOR}(\omega)})| \geq |\omega - 1|$. □

Теорема 6.5. *Если A — симметричная положительно определенная матрица, то $\rho(R_{\mathrm{SOR}(\omega)}) < 1$ для всех $0 < \omega < 2$, поэтому метод $\mathrm{SOR}(\omega)$ сходится для всех $0 < \omega < 2$. В частности, сходится метод Гаусса—Зейделя, для которого $\omega = 1$.*

Доказательство. Примем сокращение $R_{\mathrm{SOR}(\omega)} = R$ и, используя обозначения уравнения (6.19), положим $M = \omega^{-1}(D - \omega\widetilde{L})$. Проведем доказательство в два этапа:

1. определим матрицу $Q = A^{-1}(2M - A)$ и покажем, что $\Re\lambda_i(Q) > 0$ для всех i;
2. покажем, что $R = (Q - I)(Q + I)^{-1}$, откуда будет следовать, что $|\lambda_i(R)| < 1$ для всех i.

Чтобы доказать 1), заметим, что $Qx = \lambda x$ влечет за собой $(2M - A)x = \lambda Ax$, или $x^*(2M - A)x = \lambda x^*Ax$. Складывая последнее равенство с сопряженным к нему, получим $x^*(M + M^* - A)x = \Re\lambda(x^*Ax)$. Отсюда $\Re\lambda = x^*(M + M^* - A)x/x^*Ax = x^*(\frac{2}{\omega} - 1)Dx/x^*Ax > 0$, поскольку матрицы A и $(\frac{2}{\omega} - 1)D$ положительно определены.

Чтобы доказать 2), заметим, что $(Q - I)(Q + I)^{-1} = (2A^{-1}M - 2I)(2A^{-1}M)^{-1} = I - M^{-1}A = R$. По теореме о спектральном отображении (вопрос 4.5)

$$|\lambda(R)| = \left|\frac{\lambda(Q)-1}{\lambda(Q)+1}\right| = \left|\frac{(\Re\lambda(Q)-1)^2 + (\Im\lambda(Q))^2}{(\Re\lambda(Q)+1)^2 + (\Im\lambda(Q))^2}\right|^{1/2} < 1. \qquad \square$$

Вместе теоремы 6.4 и 6.5 означают, что для симметричной положительно определенной матрицы A метод SOR(ω) сходится тогда и только тогда, когда $0 < \omega < 2$.

Пример 6.7. Матрица модельной задачи симметрична и положительно определена, поэтому SOR(ω) сходится для $0 < \omega < 2$. $\diamond$

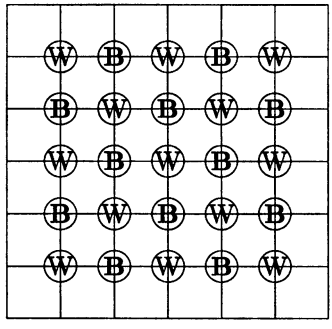
Для окончательного сравнения вычислительных затрат в методах Якоби, Гаусса—Зейделя и SOR(ω) в применении к модельной задаче наложим на A еще одно теоретико-графовое условие, часто возникающее при дискретизации некоторых типов дифференциальных уравнений с частными производными, например уравнения Пуассона. Это условие позволит нам вычислить $\rho(R_{GS})$ и $\rho(R_{\mathrm{SOR}(\omega)})$ как явные функции от $\rho(R_J)$.

Определение 6.12. *Матрица T имеет* свойство A, *если существует матрица-перестановка P, такая, что*

$$PTP^T = \left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline T_{21} & T_{22} \end{array}\right],$$

где T_{11} и T_{22} — диагональные матрицы. Другими словами, множество узлов графа $G(A)$ может быть представлено как $S_1 \cup S_2$, где, исключая петли, нет ребер, соединяющих два узла из S_1 либо два узла из S_2; такой граф $G(A)$ называется двудольным.

Пример 6.8. *Шахматное упорядочение* для модельной задачи. Оно было введено в разд. 6.5.2 с помощью иллюстрируемого рисунком «шахматного» изображения графа модельной задачи: черные узлы, помеченные как Ⓑ, принадлежат подмножеству S_1, а белые, помеченные как Ⓦ, — подмножеству S_2.



Как было отмечено в разд. 6.5.2, каждое уравнение модельной задачи связывает значение в узле сетки со значениями в соседних узлах слева, справа,

сверху и снизу, окрашенных иначе, чем центральный узел. Иначе говоря, узлы типа Ⓦ не связаны между собой непосредственно, как и узлы типа Ⓑ. Поэтому, если вначале пронумеровать белые узлы, а затем черные, то матрица приобретет форму, требуемую определением 6.12. Так, для сетки 3×3 получим следующую матрицу:

$$P\left[\begin{array}{ccc|ccc|ccc}
4 & -1 & & -1 & & & & & \\
-1 & 4 & -1 & & -1 & & & & \\
 & -1 & 4 & & & -1 & & & \\
\hline
-1 & & & 4 & -1 & & -1 & & \\
 & -1 & & -1 & 4 & -1 & & -1 & \\
 & & -1 & & -1 & 4 & & & -1 \\
\hline
 & & & -1 & & & 4 & -1 & \\
 & & & & -1 & & -1 & 4 & -1 \\
 & & & & & -1 & & -1 & 4
\end{array}\right] P^T$$

$$=\left[\begin{array}{cccc|cccc}
4 & & & & -1 & -1 & & \\
 & 4 & & & -1 & & -1 & \\
 & & 4 & & -1 & -1 & -1 & -1 \\
 & & & 4 & & -1 & & -1 \\
\hline
 & & & & 4 & & -1 & -1 \\
\hline
-1 & -1 & -1 & & 4 & & & \\
-1 & & -1 & -1 & & 4 & & \\
 & -1 & -1 & & -1 & & 4 & \\
 & & -1 & -1 & -1 & & & 4
\end{array}\right]. \qquad \diamond$$

Предположим теперь, что матрица T имеет свойство A. Тогда (обозначая диагональные блоки T_{ii} через D_i) можем записать

$$PTP^T = \begin{bmatrix} D_1 & T_{12} \\ T_{21} & D_2 \end{bmatrix} = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ -T_{21} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -T_{12} \\ 0 & 0 \end{bmatrix}$$
$$= D - \widetilde{L} - \widetilde{U}.$$

Определение 6.13. *Положим* $R_J(\alpha) = \alpha L + \frac{1}{\alpha}U$. *Тогда* $R_J(1) = R_J$ *есть итерационная матрица метода Якоби.*

Предложение 6.2. *Собственные значения матрицы* $R_J(\alpha)$ *не зависят от* α.

Доказательство. Матрица

$$R_J(\alpha) = -\begin{bmatrix} 0 & \frac{1}{\alpha}D_1^{-1}T_{12} \\ \alpha D_2^{-1}T_{21} & 0 \end{bmatrix}$$

имеет те же собственные значения, что и подобная ей матрица

$$\begin{bmatrix} I & \\ & \alpha I \end{bmatrix}^{-1} R_J(\alpha) \begin{bmatrix} I & \\ & \alpha I \end{bmatrix} = -\begin{bmatrix} 0 & D_1^{-1}T_{12} \\ D_2^{-1}T_{21} & 0 \end{bmatrix} = R_J(1). \qquad \square$$

Определение 6.14. *Для произвольной матрицы* T *положим* $T = D - \widetilde{L} - \widetilde{U}$ *и* $R_J(\alpha) = \alpha D^{-1}\widetilde{L} + \frac{1}{\alpha}D^{-1}\widetilde{U}$. *Если собственные значения матрицы* $R_J(\alpha)$ *не зависят от* α, *то* T *называют* согласованно упорядоченной *матрицей.*

Легко показать, что если T имеет свойство A (как, например, в модельной задаче), то матрица PTP^T будет согласованно упорядоченной для матрицы-перестановки P, такой, что блоки T_{11} и T_{22} в $PTP^T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}$ суть диагональные матрицы. Однако из свойства согласованной упорядоченности, вообще говоря, не следует, что матрица имеет свойство A.

Пример 6.9. Всякая блочно-трехдиагональная матрица

$$\begin{bmatrix} D_1 & A_1 & & \\ B_1 & \ddots & \ddots & \\ & \ddots & \ddots & A_{n-1} \\ & & B_{n-1} & D_n \end{bmatrix}$$

согласованно упорядочена, если блоки D_i являются диагональными матрицами. ◇

При наличии согласованной упорядоченности справедливы простые формулы, связывающие собственные значения матриц R_J, R_{GS} и $R_{\mathrm{SOR}(\omega)}$ [249].

Теорема 6.6. *Пусть A — согласованно упорядоченная матрица и $\omega \neq 0$. Тогда верны следующие утверждения:*

1. *собственные значения матрицы R_J расположены $\pm$ парами;*
2. *если μ — собственное значение матрицы R_J и*
$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2, \tag{6.27}$$
то λ — собственное значение матрицы $R_{\mathrm{SOR}(\omega)}$;
3. *обратно, если $\lambda \neq 0$ является собственным значением матрицы $R_{\mathrm{SOR}(\omega)}$, то число μ, определяемое уравнением (6.27), есть собственное значение матрицы R_J.*

Доказательство.

1. Из согласованной упорядоченности следует, что собственные значения матрицы $R_J(\alpha)$ не зависят от α, поэтому матрицы $R_J = R_J(1)$ и $R_J(-1) = -R_J(1)$ имеют одни и те же собственные значения, которые, таким образом, составляют $\pm$ пары.
2. Если $\lambda = 0$ и справедливо (6.27), то $\omega = 1$ и число 0 действительно является собственным значением матрицы $R_{\mathrm{SOR}(1)} = R_{GS} = (I - L)^{-1}U$, поскольку R_{GS} — вырожденная матрица. В противном случае,
$$\begin{aligned} 0 &= \det(\lambda I - R_{\mathrm{SOR}(\omega)}) \\ &= \det((I - \omega L)(\lambda I - R_{\mathrm{SOR}(\omega)})) \\ &= \det((\lambda + \omega - 1)I - \omega\lambda L - \omega U) \\ &= \det\left(\sqrt{\lambda}\omega\left(\left(\frac{\lambda + \omega - 1}{\sqrt{\lambda}\omega}\right)I - \sqrt{\lambda}L - \frac{1}{\sqrt{\lambda}}U\right)\right) \\ &= \det\left(\left(\frac{\lambda + \omega - 1}{\sqrt{\lambda}\omega}\right)I - L - U\right)(\sqrt{\lambda}\omega)^n. \end{aligned}$$

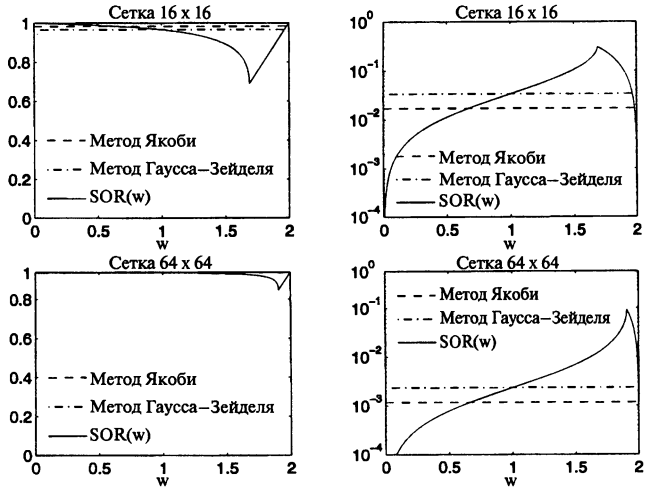


Рис. 6.5. Сходимость методов Якоби, Гаусса—Зейделя и SOR (ω) в зависимости от ω для модельной задачи на сетке 16×16 и сетке 64×64. Спектральный радиус $\rho(R)$ для каждого метода (т. е. $\rho(R_J)$, $\rho(R_{GS})$ и $\rho(R_{SOR(\omega)})$) изображен на левых рисунках, а функция $1 - \rho(R)$ изображена на правых рисунках.

Последнее равенство выполняется в силу предложения 6.2. Итак, число $\frac{\lambda+\omega-1}{\sqrt{\lambda}\omega} = \mu$ есть собственное значение матрицы $L + U = R_J$; при этом $(\lambda + \omega - 1)^2 = \mu^2\omega^2\lambda$.

3. Если $\lambda \neq 0$, то цепочку равенств в 2) можно пройти в обратном направлении. □

Следствие 6.1. *Если A — согласованно упорядоченная матрица, то $\rho(R_{GS}) = (\rho(R_J))^2$. Это означает, что метод Гаусса—Зейделя вдвое быстрее метода Якоби.*

Доказательство. Выбор $\omega = 1$ эквивалентен методу Гаусса—Зейделя. Из (6.27) получаем $\lambda^2 = \lambda\mu^2$, или $\lambda = \mu^2$. □

Чтобы получить наибольшую выгоду от верхней релаксации, хотелось бы найти значение ω_{opt}, минимизирующее $\rho(R_{\mathrm{SOR}(\omega)})$ [249].

Теорема 6.7. *Предположим, что матрица A согласованно упорядочена, матрица R_J имеет вещественные собственные значения и $\mu = \rho(R_J) < 1$. Тогда*

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu^2}},$$

$$\rho(R_{\mathrm{SOR}(\omega_{opt})}) = \omega_{opt} - 1 = \frac{\mu^2}{[1 + \sqrt{1 - \mu^2}]^2},$$

$$\rho(R_{\mathrm{SOR}(\omega)}) = \begin{cases} \omega - 1, & \omega_{opt} \leq \omega \leq 2, \\ 1 - \omega + \frac{1}{2}\omega^2\mu^2 + \omega\mu\sqrt{1 - \omega + \frac{1}{4}\omega^2\mu^2}, & 0 < \omega \leq \omega_{opt}. \end{cases}$$

Доказательство. Достаточно решить относительно λ уравнение $(\lambda+\omega-1)^2 = \lambda\omega^2\mu^2$. □

Пример 6.10. Модельная задача может служить примером: матрица R_J для нее симметрична и, следовательно, имеет вещественные собственные значения. На рис. 6.5 показан график $\rho(R_{\mathrm{SOR}(\omega)})$ как функции от ω, а также графики $\rho(R_{GS})$ и $\rho(R_J)$ для модельной задачи на сетке $N \times N$, где $N = 16$ и $N = 64$. Левые рисунки — это графики функций $\rho(R)$, а правые — полулогарифмические графики функций $1 - \rho(R)$. Главный вывод, который можно сделать из этой иллюстрации, состоит в том, что вблизи точки минимума функция $\rho(R_{\mathrm{SOR}(\omega)})$ изменяется очень быстро, поэтому, если значение ω даже лишь немного отличается от ω_{opt}, сходимость метода существенно замедляется. Еще один вывод: если приближенное значение ω_{opt} приходится угадывать, то лучше переоценить его (указать значение, близкое к 2), чем недооценить. ◇

6.5.6. Чебышевское ускорение и симметричная последовательная верхняя релаксация (SSOR)

Если говорить об уже обсуждавшихся методах, то для использования методов Якоби и Гаусса—Зейделя не требуется никакой информации о матрице системы (хотя для доказательства сходимости этих методов некоторая информация о матрице все же нужна). Метод SOR(ω) зависит от параметра ω, значение которого может быть выбрано в согласии с $\rho(R_J)$ так, чтобы ускорить сходимость. Чебышевское ускорение полезно тогда, когда о спектре матрицы R_J известно большее, чем только значение $\rho(R_J)$; в этом случае сходимость можно ускорить еще сильнее.

Предположим, что система $Ax = b$ с помощью какого-либо метода (Якоби, Гаусса—Зейделя или SOR(ω)) преобразована в итерационный процесс $x_{i+1} = Rx_i + c$. Тогда будет получена последовательность векторов $\{x_i\}$, где $x_i \to x$ при $i \to \infty$, если $\rho(R) < 1$.

Если все эти приближения x_i имеются, то естественно спросить, не существует ли какой-либо их линейной комбинации $y_m = \sum_{i=0}^{m} \gamma_{mi} x_i$, которая еще лучше приближает решение x. Заметим, что коэффициенты γ_{mi} должны удовлетворять условию $\sum_{i=0}^{m} \gamma_{mi} = 1$, поскольку, если $x_0 = x_1 = \cdots = x$, то мы хотели бы, чтобы и y_m было равно x. Таким образом, погрешность приближения y_m можно записать как

$$\begin{aligned} y_m - x &= \sum_{i=0}^{m} \gamma_{mi} x_i - x \\ &= \sum_{i=0}^{m} \gamma_{mi} (x_i - x) \\ &= \sum_{i=0}^{m} \gamma_{mi} R^i (x_0 - x) \\ &= p_m(R)(x_0 - x), \end{aligned} \tag{6.28}$$

где $p_m(R) = \sum_{i=0}^{m} \gamma_{mi} R^i$ есть многочлен степени m, такой, что $p_m(1) = \sum_{i=0}^{m} \gamma_{mi} = 1$.

Пример 6.11. Если бы в качестве p_m мы могли взять характеристический многочлен матрицы R, то по теореме Гамильтона— Кэли мы имели бы $p_m(R) = 0$, что означает сходимость в m шагов. Этот способ, однако, не практичен, так как собственные значения матрицы R редко бывают известны, да и в любом случае мы хотели бы, чтобы метод сошелся быстрее, чем за $m = \dim R$ шагов. ◇

Вместо поисков многочлена p_m, для которого $p_m(R)$ есть нулевая матрица, постараемся насколько возможно уменьшить значение спектрального радиуса матрицы $p_m(R)$. Предположим, что известно следующее:

- все собственные значения матрицы R вещественны и
- эти собственные значения находятся в отрезке $[-\rho, \rho]$, не содержащем 1.

Тогда можно попытаться найти многочлен p_m, такой, что

1. $p_m(1) = 1$ и
2. число $\max_{-\rho \le x \le \rho} |p_m(x)|$ имеет наименьшее возможное значение.

Поскольку собственными значениями матрицы $p_m(R)$ являются числа $p_m(\lambda(R))$ (см. задачу 4.5), эти собственные значения будут малы, а потому будет мал и спектральный радиус (как наибольший из модулей собственных значений).

Построение многочлена p_m, удовлетворяющего сформулированным условиям 1) и 2), — это классическая задача теории приближений, решение которой опирается на *многочлены Чебышева*.

Определение 6.15. *Многочлен Чебышева $T_m(x)$ степени m определяется рекуррентным соотношением $T_m(x) \equiv 2xT_{m-1}(x) - T_{m-2}(x)$, где $T_0(x) = 1$ и $T_1(x) = x$.*

Чебышевские многочлены обладают рядом интересных свойств [240]. Приведем несколько свойств, которые легко вывести из определения (см. вопрос 6.7).

Лемма 6.7. *Многочлены Чебышева имеют следующие свойства:*

- $T_m(1) = 1$.
- $T_m(x) = 2^{m-1}x^m + O(x^{m-1})$.
- $T_m(x) = \begin{cases} \cos(m \cdot \arccos x), & \text{если } |x| \le 1, \\ \operatorname{ch}(m \cdot \operatorname{arccos} x), & \text{если } |x| \ge 1. \end{cases}$
- $|T_m(x)| \le 1$, *если* $|x| \le 1$.
- *Нулями многочлена* $T_m(x)$ *являются числа* $x_i = \cos((2i-1)\pi/(2m))$, $i = 1, \ldots, m$.
- $T_m(x) = \frac{1}{2}[(x + \sqrt{x^2-1})^m + (x + \sqrt{x^2-1})^{-m}]$, *если* $|x| > 1$.
- $T_m(1+\epsilon) \ge .5(1 + m\sqrt{2\epsilon})$, *если* $\epsilon > 0$.

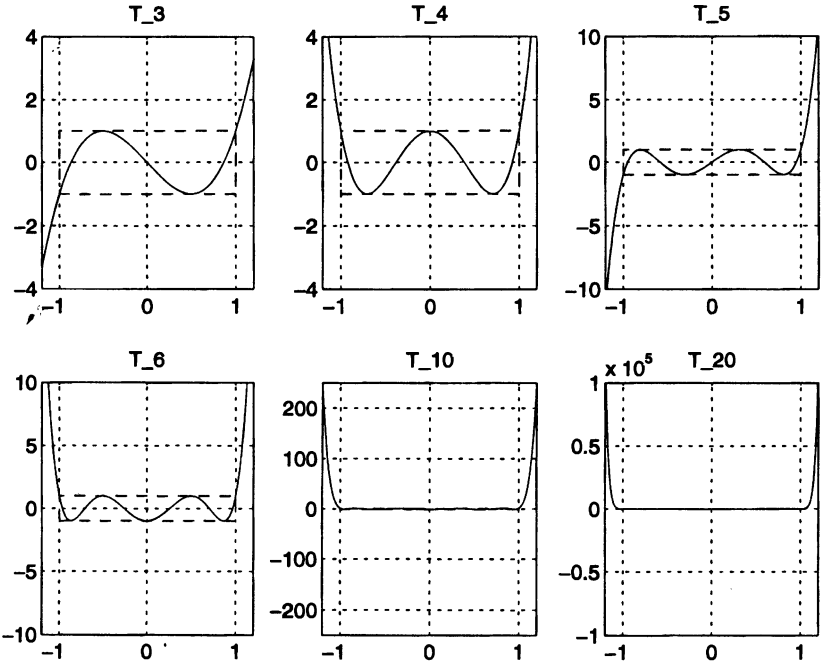


Рис. 6.6. График $T_m(x)$ как функции от x. Пунктирные линии показывают, что $|T_m(x)| \leq 1$ для $|x| \leq 1$.

Приведем таблицу значений для $T_m(1+\epsilon)$. Обратим внимание на то, как быстро растут эти числа с увеличением m, даже если ϵ очень мало (см. рис. 6.6).

m	ϵ		
	10^{-4}	10^{-3}	10^{-2}
10	1.0	1.1	2.2
100	2.2	44	$6.9 \cdot 10^5$
200	8.5	$3.8 \cdot 10^3$	$9.4 \cdot 10^{11}$
1000	$6.9 \cdot 10^5$	$1.3 \cdot 10^{19}$	$1.2 \cdot 10^{61}$

Многочленом с теми свойствами, которые нам нужны, является $p_m(x) = T_m(x/\rho)/T_m(1/\rho)$. Чтобы убедиться в этом, заметим, что $p_m(1) = 1$ и если $x \in [-\rho, \rho]$, то $|p_m(x)| \leq 1/T_m(1/\rho)$. Так, если $\rho = 1/(1+\epsilon)$, то $|p_m(x)| \leq 1/T_m(1+\epsilon)$. Как мы только что видели, эта граница очень мала при малых ϵ и умеренных значениях m.

Для экономной реализации процесса воспользуемся трехчленной рекурсией $T_m(x) \equiv 2xT_{m-1}(x) - T_{m-2}(x)$, использованной в определении многочленов Чебышева. Она позволяет хранить и комбинировать только три вектора y_m, y_{m-1} и y_{m-2}, а не все предыдущие векторы x_m. Посмотрим, как это получается.

Положим $\mu_m \equiv 1/T_m(1/\rho)$, тогда $p_m(R) = \mu_m T_m(R/\rho)$ и, согласно трехчленной рекурсии в определении 6.15, $\frac{1}{\mu_m} = \frac{2}{\rho\mu_{m-1}} - \frac{1}{\mu_{m-2}}$. Далее,

$$\begin{aligned}
y_m - x &= p_m(R)(x_0 - x) \quad \text{согласно равенству (6.28)} \\
&= \mu_m T_m\left(\frac{R}{\rho}\right)(x_0 - x) \\
&= \mu_m \left[2 \cdot \frac{R}{\rho} \cdot T_{m-1}\left(\frac{R}{\rho}\right)(x_0 - x) - T_{m-2}\left(\frac{R}{\rho}\right)(x_0 - x)\right] \\
&\qquad \text{по определению 6.15} \\
&= \mu_m \left[2 \cdot \frac{R}{\rho} \cdot \frac{p_{m-1}(\frac{R}{\rho})(x_0 - x)}{\mu_{m-1}} - \frac{p_{m-2}(\frac{R}{\rho})(x_0 - x)}{\mu_{m-2}}\right] \\
&= \mu_m \left[2 \cdot \frac{R}{\rho} \cdot \frac{y_{m-1} - x}{\mu_{m-1}} - \frac{y_{m-2} - x}{\mu_{m-2}}\right] \quad \text{согласно равенству (6.28),}
\end{aligned}$$

или

$$y_m = \frac{2\mu_m}{\mu_{m-1}} \frac{R}{\rho} y_{m-1} - \frac{\mu_m}{\mu_{m-2}} y_{m-2} + d_m,$$

где

$$\begin{aligned}
d_m &= x - \frac{2\mu_m}{\mu_{m-1}}\left(\frac{R}{\rho}\right)x + \frac{\mu_m}{\mu_{m-2}}x \\
&= x - \frac{2\mu_m}{\mu_{m-1}}\left(\frac{x - c}{\rho}\right) + \frac{\mu_m}{\mu_{m-2}}x \quad \text{так как } x = Rx + c \\
&= \mu_m\left(\frac{1}{\mu_m} - \frac{2}{\rho\mu_{m-1}} + \frac{1}{\mu_{m-2}}\right)x + \frac{2\mu_m}{\rho\mu_{m-1}}c \\
&= \frac{2\mu_m}{\rho\mu_{m-1}}c \quad \text{согласно определению числа } \mu_m.
\end{aligned}$$

В результате приходим к следующему алгоритму:

Алгоритм 6.7. *Чебышевское ускорение процесса* $x_{i+1} = Rx_i + c$*:*

$\mu_0 = 1$; $\mu_1 = \rho$; $y_0 = x_0$; $y_1 = Rx_0 + c$
for $m = 2, 3, \ldots$
$\quad \mu_m = 1/\left(\frac{2}{\rho\mu_{m-1}} - \frac{1}{\mu_{m-2}}\right)$
$\quad y_m = \frac{2\mu_m}{\rho\mu_{m-1}} R y_{m-1} - \frac{\mu_m}{\mu_{m-2}} y_{m-2} + \frac{2\mu_m}{\rho\mu_{m-1}} c$
end for

Отметим, что на каждой итерации выполняется ровно одно матрично-векторное умножение с матрицей R. Если стоимость такого умножения значительно выше, чем прочих скалярных и векторных операций, то шаг алгоритма 6.7 имеет ту же сложность, что и шаг исходного процесса $x_{m+1} = Rx_m + c$.

К сожалению, описанный алгоритм нельзя непосредственно применить к методу SOR(ω) при решении системы $Ax = b$. Дело в том, что матрица $R_{\mathrm{SOR}(\omega)}$ в общем случае имеет комплексные собственные значения, а чебышевское ускорение требует, чтобы собственные значения матрицы R были вещественны и принадлежали сегменту $[-\rho, \rho]$. Однако ситуацию можно поправить с помощью следующего алгоритма.

Алгоритм 6.8. *Симметричный метод* SOR *(*SSOR*):*

1. *Выполнить шаг метода* SOR(ω), *вычисляя компоненты вектора* x *в обычном порядке возрастания* $x_{i,1}, x_{i,2}, \ldots, x_{i,n}$.
2. *Выполнить шаг метода* SOR(ω), *вычисляя компоненты вектора* x *в обратном порядке* $x_{i,n}, x_{i,n-1}, \ldots, x_{i,1}$.

Мы представим этот алгоритм в виде $x_{i+1} = E_\omega x_i + c_\omega$ и покажем, что матрица E_ω имеет вещественные собственные значения, а потому применение чебышевского ускорения становится возможным.

Предположим, что A — симметричная матрица (как это имеет место в модельной задаче), и представим A, как в уравнении (6.19): $A = D - \widetilde{L} - \widetilde{U} = D(I - L - U)$. Поскольку $A = A^T$, имеем $U = L^T$. С помощью равенства (6.21) перепишем два полушага метода SSOR следующим образом:

$$\begin{array}{lrcl} 1. & x_{i+1/2} & = & (I-\omega L)^{-1}((1-\omega)I+\omega U)x_i + c_{1/2} \quad \equiv \quad L_\omega x_i + c_{1/2}, \\ 2. & x_{i+1} & = & (I-\omega U)^{-1}((1-\omega)I+\omega L)x_{i+1/2} + c_1 \quad \equiv \quad U_\omega x_{i+1/2} + c_1. \end{array}$$

Исключая отсюда $x_{i+1/2}$, получим $x_{i+1} = E_\omega x_i + \hat{c}$, где

$$\begin{aligned} E_\omega &= U_\omega L_\omega \\ &= I + (\omega-2)^2(I-\omega U)^{-1}(I-\omega L)^{-1} + (\omega-2)(I-\omega U)^{-1} \\ &\quad + (\omega-2)(I-\omega U)^{-1}(I-\omega L)^{-1}(I-\omega U). \end{aligned}$$

Мы утверждаем, что собственные значения матрицы E_ω вещественны. Действительно, она имеет те же собственные значения, что и подобная ей матрица

$$\begin{aligned} &(I-\omega U)E_\omega(I-\omega U)^{-1} \\ &\quad = I + (2-\omega)^2(I-\omega L)^{-1}(I-\omega U)^{-1} + (\omega-2)(I-\omega U)^{-1} \\ &\qquad + (\omega-2)(I-\omega L)^{-1} \\ &\quad = I + (2-\omega)^2(I-\omega L)^{-1}(I-\omega L^T)^{-1} + (\omega-2)(I-\omega L^T)^{-1} \\ &\qquad + (\omega-2)(I-\omega L)^{-1}. \end{aligned}$$

Очевидно, что эта матрица симметрична и потому должна иметь вещественные собственные значения.

Пример 6.12. Применим метод SSOR(ω) с чебышевским ускорением к модельной задаче. Нужно выбрать значение ω и оценить спектральный радиус $\rho = \rho(E_\omega)$. Оптимальное значение ω, минимизирующее ρ, не известно, однако Янг показал (см. [267, 137]), что вполне удовлетворителен выбор $\omega = \frac{2}{1+[2(1-\rho(R_J))]^{1/2}}$, для которого $\rho(E_\omega) \approx 1 - \frac{\pi}{2N}$. При использовании чебышевского ускорения погрешность на шаге m умножится на число $\mu_m \approx \frac{1}{T_m(1+\frac{\pi}{2N})} \leq 2/(1+m\sqrt{\frac{\pi}{N}})$. Поэтому потребуются $m = O(N^{1/2}) = O(n^{1/4})$ итераций, чтобы уменьшить погрешность в фиксированное число раз, большее единицы. Поскольку каждая итерация имеет ту же стоимость $O(n)$, что и итерация метода SOR(ω), общая стоимость процесса составляет $O(n^{5/4})$. Это объясняет элемент табл. 6.1, относящийся к методу SSOR с чебышевским ускорением.

В отличие от сказанного выше, после m шагов метода SOR(ω_{opt}) погрешность умножилась бы лишь на $(1-\frac{\pi}{N})^m$. К примеру, положим $N = 1000$. Тогда,

чтобы уменьшить погрешность вдвое, методу SOR(ω_{opt}) требуются 220 итераций, тогда как для SSOR(ω_{opt}) с чебышевским ускорением нужны лишь $m = 17$ итераций. ◇

6.6. Методы крыловского подпространства

Эти методы используются и для решения системы $Ax = b$, и для вычисления собственных значений матрицы A. В них предполагается, что A доступна лишь посредством «черного ящика» в виде подпрограммы, вычисляющей по заданному вектору z произведение $y = Az$ (а также, возможно, произведение $y = A^T z$, если матрица A несимметрична). Другими словами, прямой доступ к элементам матрицы и их изменение не используются. Такое предположение разумно по нескольким причинам. Во-первых, умножение на вектор — это наиболее дешевая нетривиальная операция, которую можно проделать с (разреженной) матрицей. Если в A имеется m ненулевых элементов, то для матрично-векторного умножения нужны m скалярных умножений и (самое большее) m сложений. Во-вторых, A может не быть представлена явно в виде матрицы, а доступна лишь через подпрограмму для вычисления произведений Ax.

Пример 6.13. Предположим, что имеется физическое устройство, поведение которого моделируется программой, вычисляющей по вектору x входных параметров вектор y выходных параметров, описывающих поведение устройства. Выходной вектор y может быть как угодно сложной функцией $y = f(x)$, возможно, требующей даже решения нелинейных дифференциальных уравнений. Например, параметры, составляющие вектор x, могут описывать форму крыла, а $f(x)$ может быть лобовым сопротивлением на крыле, вычисляемым путем решения уравнений Навье— Стокса для потока через крыло. Типовой конструкторской задачей является выбор входа x, оптимизирующего поведение устройства $f(x)$; для определенности, предположим, что оптимизация означает наибольшее возможное уменьшение $f(x)$. Наша задача в этом случае сведется к тому, чтобы насколько возможно удовлетворить уравнение $f(x) = 0$. Для целей иллюстрации, предположим, что x и y — векторы одинаковой размерности. Тогда очевидным выбором для приближенного решения уравнения является метод Ньютона, что приводит к итерациям $x^{(m+1)} = x^{(m)} - (\nabla f(x^{(m)}))^{-1} f(x^{(m)})$, где $\nabla f(x^{(m)})$ — матрица Якоби функции f в точке $x^{(m)}$. Это соотношение можно переписать как задачу решения линейной системы $(\nabla f(x^{(m)}))\delta^{(m)} = f(x^{(m)})$ относительно $\delta^{(m)}$ с последующим вычислением вектора $x^{(m+1)} = x^{(m)} - \delta^{(m)}$. Как же, однако, следует решать эту линейную систему с матрицей коэффициентов $\nabla f(x^{(m)})$, если даже вычисление $f(x^{(m)})$ представляет трудность? Оказывается, что для произвольного вектора z можно вычислить матрично-векторное произведение $(\nabla f(x)) \cdot z$, поэтому к решению линейной системы можно применить методы крыловского подпространства. Используя для вычисления $(\nabla f(x)) \cdot z$ *разностные отношения* или формулу Тейлора, можно видеть, что $[f(x+hz) - f(x)]/h \approx (\nabla f(x)) \cdot z$. Таким образом, чтобы вычислить $(\nabla f(x)) \cdot z$, нужно дважды обратиться к подпрограмме, вычисляющей $f(\cdot)$, один раз с аргументом x, а другой — с аргументом $x + hz$. Иногда, однако, бывает трудно выбрать значение h, дающее

хорошую аппроксимацию производной (выбор слишком малого h приводит к потере точности из-за округлений). Другой способ вычисления $(\nabla f(x))\cdot z$ состоит в реальном дифференцировании функции f. Если f — достаточно простая функция, то это можно проделать на бумаге. Для сложных функций f существуют инструментальные средства, позволяющие по (почти) любой подпрограмме, вычисляющей $f(x)$, автоматически породить другую подпрограмму, вычисляющую $(\nabla f(x))\cdot z$ (см. [29]). Можно также, хоть это и менее эффективно, использовать специальные возможности языков C++ и фортран 90. ◇

Существует множество различных методов крыловского подпространства. Некоторые из них пригодны для несимметричных матриц, другие требуют от матрицы симметрии или положительной определенности. Некоторые методы для несимметричных матриц предполагают, что могут вычисляться не только произведения типа Az, но и произведения типа A^Tz; в зависимости от способа представления матрицы A, такие произведения могут быть или не быть доступны (см. пример 6.13). Наиболее эффективным и лучше всего понимаемым методом этого класса является метод сопряженных градиентов (метод CG). Он применим только к симметричным положительно определенным матрицам, в частности он приложим к модельной задаче. В этом разделе мы сосредоточимся на методе CG.

Если матрица не является симметричной положительно определенной, то выбор наилучшего метода из многих существующих может представлять определенные трудности. В разд. 6.6.6 дан краткий обзор методов, имеющихся помимо метода CG, а также рекомендации по поводу того, какой метод в какой ситуации следует использовать. Отсылаем читателя также к более полному сетевому справочнику в NETLIB/templates, включающему в себя книгу [24] и реализации методов на языках Matlab, фортран и C++. Обзоры современных исследований по методам крыловского подпространства можно найти в [15, 107, 136, 214].

В гл. 7 мы обсудим методы крыловского подпространства в применении к задаче вычисления собственных значений.

6.6.1. Извлечение информации о матрице A из матрично-векторных произведений

Пусть даны вектор b и подпрограмма, вычисляющая произведения вида $A\cdot x$; что можно узнать с их помощью о матрице A? Наиболее очевидным решением является вычисление последовательности матрично-векторных произведений $y_1 = b$, $y_2 = Ay_1$, $y_3 = Ay_2 = A^2y_1$, ..., $y_n = Ay_{n-1} = A^{n-1}y_1$, где n — порядок матрицы A. Положим $K = [y_1, y_2, \ldots, y_n]$. Тогда можно написать

$$A\cdot K = [Ay_1, \ldots, Ay_{n-1}, Ay_n] = [y_2, \ldots, y_n, A^ny_1]. \tag{6.29}$$

Заметим, что первые $n-1$ столбцов матрицы $A\cdot K$ совпадают с последними $n-1$ столбцами в K (при нумерации, сдвинутой на единицу). Предположим пока, что матрица K невырожденна; тогда можно вычислить вектор $c = -K^{-1}A^ny_1$. Теперь

$$A\cdot K = K\cdot[e_2, e_3, \ldots, e_n, -c] \equiv K\cdot C,$$

где e_i — столбец i в единичной матрице. Отсюда

$$K^{-1}AK = C = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_1 \\ 1 & 0 & \cdots & 0 & -c_2 \\ 0 & 1 & \cdots & \vdots & \vdots \\ \vdots & 0 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & 0 & \vdots \\ \vdots & \vdots & \cdots & 1 & -c_n \end{bmatrix}.$$

Обратим внимание, что C — верхняя хессенбергова матрица. В действительности, это *сопровождающая матрица* (см. разд. 4.5.3) многочлена $p(x) = x^n + \sum_{i=1}^{n} c_i x^{i-1}$, являющегося ее характеристическим многочленом. Таким образом, используя лишь матрично-векторные произведения, мы привели A к очень простой форме, позволяющей, в принципе, найти собственные значения как нули многочлена $p(x)$.

Однако эта простая форма бесполезна для практики в силу следующих причин:

1. Определение вектора c требует $n-1$ матрично-векторных умножений с участием матрицы A, а затем решения линейной системы с матрицей K. Даже если A — разреженная матрица, K, скорей всего, будет плотной, поэтому нет никаких оснований ожидать, что линейная система с матрицей K решается сколько-нибудь проще, чем исходная система $Ax = b$.
2. С большой вероятностью, матрица K будет очень плохо обусловленной, а потому c будет вычислен очень неточно. Причина в том, что для вычисления столбцов y_i матрицы K наш алгоритм выполняет степенной метод (алгоритм 4.1), поэтому векторы y_i сходятся к собственному вектору, отвечающему наибольшему собственному значению матрицы A. Таким образом, столбцы в K все более и более приближаются к параллельным.

Мы преодолеем эти проблемы следующим образом: заменим K ортогональной матрицей Q, такой, что при любом k линейные оболочки первых k столбцов в K и Q являются одним и тем же подпространством. Оно называется *крыловским подпространством*. В отличие от K, матрица Q хорошо обусловлена и легко обратима. Кроме того, мы будем вычислять лишь столько первых столбцов в Q, сколько необходимо для получения достаточно хорошего приближения к решению (системы $Ax = b$ или задачи $Ax = \lambda x$). На практике, обычно бывает достаточно очень небольшого (в сравнении с порядком n матрицы) числа столбцов.

Используем QR-разложение матрицы K, чтобы записать $K = QR$. Тогда

$$K^{-1}AK = (R^{-1}Q^T)A(QR) = C,$$

откуда

$$Q^TAQ = RCR^{-1} \equiv H.$$

Так как R и R^{-1} — верхние треугольные матрицы, а C — верхняя хессенбергова, то, как легко проверить, матрица $H = RCR^{-1}$ также является верхней

хессенберговой (см. вопрос 6.11). Иначе говоря, мы привели A к верхней хессенберговой форме посредством ортогонального преобразования с матрицей Q. (Это первая часть алгоритма вычисления собственных значений несимметричных матриц, обсуждавшегося в разд. 4.4.6.) Заметим, что при симметричной матрице A матрица $Q^TAQ = H$ также симметрична; симметричная же матрица, имеющая верхнюю хессенбергову форму, должна быть и нижней хессенберговой, т. е. трехдиагональной. В этом случае мы будем писать $Q^TAQ = T$.

Нам еще остается показать, как вычислять столбцы матрицы Q «по одному», а не все сразу. Положим $Q = [q_1, \ldots, q_n]$. Поскольку из $Q^TAQ = H$ следует $AQ = QH$, можно приравнять столбцы j в обеих частях матричного равенства, что дает

$$Aq_j = \sum_{i=1}^{j+1} h_{i,j} q_i.$$

Учитывая, что q_i — ортонормированные векторы, и умножая обе части этого соотношения на q_m^T, получаем

$$q_m^T Aq_j = \sum_{i=1}^{j+1} h_{i,j} q_m^T q_i = h_{m,j} \text{ для } 1 \le m \le j$$

и

$$h_{j+1,j} q_{j+1} = Aq_j - \sum_{i=1}^{j} h_{i,j} q_i.$$

В результате приходим к следующему алгоритму.

Алгоритм 6.9. *Алгоритм Арнольди для (частичного) приведения к форме Хессенберга:*

$q_1 = b/\|b\|_2$
/ k — это число определяемых столбцов в Q и H */*
for $j = 1$ *to* k
  $z = Aq_j$
  for $i = 1$ *to* j
    $h_{i,j} = q_i^T z$
    $z = z - h_{i,j} q_i$
  end for
  $h_{j+1,j} = \|z\|_2$
  если $h_{j+1,j} = 0$ *то выход*
  $q_{j+1} = z/h_{j+1,j}$
end for

Векторы q_i, вычисляемые в алгоритме Арнольди, часто называют *векторами Арнольди*. Цикл по i, в котором пересчитывается вектор z, можно интерпретировать как применение к этому вектору *модифицированного алгоритма Грама—Шмидта* (алгоритм 3.1): из z вычитаются компоненты в направлениях от q_1 до q_j, в результате чего z становится ортогонален этим направлениям. Для вычисления векторов $q_1, \ldots, q_k$ нужны k матрично-векторных умножений с участием матрицы A и другие операции, стоимость которых составляет $O(k^2 n)$. Если алгоритм остановить на этом месте, то что можно узнать о

матрице A? Положим $Q = [Q_k, Q_u]$, где $Q_k = [q_1, \ldots, q_k]$ и $Q_u = [q_{k+1}, \ldots, q_n]$. Заметим, что мы вычислили лишь Q_k и q_{k+1}; прочие столбцы матрицы Q_u не известны. Имеем

$$H = Q^T A Q = [Q_k, Q_u]^T A [Q_k, Q_u] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix}$$

$$\equiv \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} H_k & H_{uk} \\ H_{ku} & H_u \end{pmatrix} \end{array}. \tag{6.30}$$

Отметим, что H_k — верхняя хессенбергова матрица, поскольку вся матрица H обладает этим свойством. По той же причине, матрица H_{ku} имеет единственный (возможно) ненулевой элемент, стоящий в ее правом верхнем углу, а именно элемент $h_{k+1,k}$. Итак, H_u и H_{uk} не известны, мы знаем лишь блоки H_k и H_{ku}.

Если A — симметричная матрица, то $H = T$ симметрична и трехдиагональна. Алгоритм Арнольди значительно упрощается, поскольку большинство чисел h_{ij} являются нулями. Положим

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Приравнивая столбцы j в обеих частях равенства $AQ = QT$, получаем

$$Aq_j = \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1}.$$

Умножая обе части этого соотношения на q_j^T и учитывая, что столбцы в матрице Q ортонормальны, находим $q_j^T A q_j = \alpha_j$. Сказанное обосновывает следующую версию алгоритма Арнольди, называемую *алгоритмом Ланцоша*:

Алгоритм 6.10. *Алгоритм Ланцоша для (частичного) приведения к симметричной трехдиагональной форме.*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
  $\beta_j = \|z\|_2$
  если $\beta_j = 0$ *то выход*
  $q_{j+1} = z/\beta_j$
end for

Векторы q_i, вычисляемые в алгоритме Ланцоша, часто называют *векторами Ланцоша*. Вот что мы знаем о матрице A после k шагов этого алгоритма:

$$\begin{aligned} T = Q^T A Q &= [Q_k, Q_u]^T A [Q_k, Q_u] \\ &= \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix} \\ &\equiv \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} T_k & T_{uk} \\ T_{ku} & T_u \end{pmatrix} \end{array} \\ &= \begin{bmatrix} T_k & T_{ku}^T \\ T_{ku} & T_u \end{bmatrix}. \end{aligned} \tag{6.31}$$

Поскольку матрица A симметрична, нам известны блоки T_k и $T_{ku} = T_{uk}^T$, но не блок T_u. Матрица T_{ku} имеет единственный (возможно) ненулевой элемент, стоящий в ее правом верхнем углу, а именно элемент β_k. Отметим, что этот элемент неотрицателен, поскольку вычисляется как норма вектора z.

Введем некоторые стандартные обозначения, связанные с частичным разложением матрицы A, вычисляемым в алгоритмах Арнольди и Ланцоша.

Определение 6.16. Подпространство Крылова $\mathcal{K}_k(A, b)$ — *это линейная оболочка векторов* b, Ab, A^2b, ..., $A^{k-1}b$.

Мы будем писать $\mathcal{K}_k$ вместо $\mathcal{K}_k(A, b)$, если A и b ясны из контекста. Если не происходит досрочного выхода из алгоритма вследствие $z = 0$, то векторы Q_k, вычисляемые в алгоритмах Арнольди и Ланцоша, образуют ортонормированный базис подпространства $\mathcal{K}_k$. (Можно показать, что $\mathcal{K}_k$ имеет размерность k в том и только том случае, если алгоритм Арнольди или Ланцоша может вычислить вектор q_k без досрочного выхода; см. вопрос 6.12.) Будем называть H_k (или T_k) *проекцией* матрицы A на подпространство Крылова $\mathcal{K}_k$.

Наша цель — построить алгоритмы решения системы $Ax = b$, использующие лишь информацию, вычисляемую на k шагах алгоритма Арнольди или Ланцоша. Мы рассчитываем на то, что k может быть много меньше, чем n, что сделает алгоритмы эффективными.

(В гл. 7 та же информация будет использована для вычисления собственных значений матрицы A. Мы можем уже сейчас вкратце описать, как это будет сделано. Заметим, что если элемент $h_{k+1,k}$ окажется нулем, то матрица H (или T) будет блочно верхнетреугольной, а потому все собственные значения блока H_k будут в то же время собственными значениями для H, следовательно, и для A, поскольку H и A подобны. Правые собственные векторы блока H_k определяют собственные векторы матрицы H; умножив их на Q_k, мы получим собственные векторы матрицы A. Если элемент $h_{k+1,k}$ не равен нулю, но мал, то можно ожидать, что собственные значения и собственные векторы блока H_k будут хорошими приближениями к собственным значениям и собственным векторам матрицы A.)

Закончим это введение указанием на то, что при наличии округлений многие из обсуждаемых алгоритмов ведут себя *совершенно иначе*, чем они делали бы это в точной арифметике. В частности, векторы q_i, вычисляемые в алгоритме Ланцоша, могут быстро потерять ортогональность и, в действительности,

часто становятся линейно зависимыми. Эта кажущаяся катастрофической численная неустойчивость была причиной того, что методы данной группы были на ряд лет преданы забвению после их обнаружения. Однако со временем исследователи научились стабилизировать эти алгоритмы или убедились, что сходимость в них имеет место вопреки неустойчивости! Мы вернемся к этим вопросам в разд. 6.6.4, где анализируется сходимость метода сопряженных градиентов для решения системы $Ax = b$ (который «неустойчив», но тем не менее сходится), и в гл. 7, особенно в разд. 7.4 и 7.5, где показано, как вычислить собственные значения (и основной алгоритм модифицирован так, чтобы обеспечить устойчивость).

6.6.2. Решение системы $Ax = b$ с помощью крыловского подпространства

Как же решать систему $Ax = b$, имея лишь информацию, доступную после k шагов алгоритма Арнольди или Ланцоша?

Поскольку единственными известными нам векторами являются столбцы матрицы Q_k, единственное место, где следует «искать» приближенное решение, — это подпространство Крылова $\mathcal{K}_k$, натянутое на векторы q_i. Иначе говоря, мы будем искать «наилучшее» приближенное решение вида

$$x_k = \sum_{j=1}^{k} z_j q_j = Q_j \cdot z, \quad \text{где } z = [z_1, \ldots, z_k]^T.$$

Теперь нужно определить смысл слова «наилучшее». Имеется несколько естественных, но различных определений, ведущих к разным алгоритмам. Будем обозначать точное решение через $x = A^{-1}b$, а невязку $b - Ax_k$ через r_k.

1. «Наилучшее» x_k минимизирует $\|x_k - x\|_2$. К сожалению, у нас нет достаточной информации в крыловском подпространстве для вычисления такого вектора x_k.
2. «Наилучшее» x_k минимизирует $\|r_k\|_2$. Это определение вполне реализуемо и соответствующий алгоритм называется MINRES (от *minimum residual* — минимальная невязка) в случае симметричной матрицы A [194] и GMRES (от *generalized minimum residual* — обобщенный алгоритм минимальных невязок), если матрица A несимметрична [215].
3. «Наилучшее» x_k обеспечивает условие $r_k \perp \mathcal{K}_k$, или $Q_k^T r_k = 0$. Оно иногда называется условием *ортогональной невязки*, или *условием Галеркина* по аналогии с родственным условием в теории конечных элементов. Если A симметрична, то соответствующий алгоритм называется SYMMLQ [194]. Для несимметричной матрицы A имеется вариант метода GMRES [211].
4. Посредством симметричной положительно определенной матрицы A можно задать норму $\|r\|_{A^{-1}} = (r^T A^{-1} r)^{1/2}$ (см. лемму 1.3). Будем считать, что «наилучшее» x_k минимизирует $\|r_k\|_{A^{-1}}$. Эта норма есть то же самое, что $\|x_k - x\|_A$. Соответствующий метод называется алгоритмом сопряженных градиентов [145].

Для симметричной положительно определенной матрицы A два последних определения слова «наилучшее» оказываются эквивалентными.

Теорема 6.8. *Пусть A — симметричная матрица, $T_k = Q_k^T A Q_k$ и $r_k = b - Ax_k$, где $x \in \mathcal{K}_k$. Если матрица T_k невырожденна и $x_k = Q_k T_k^{-1} e_1 \|b\|_2$, где $e_1^{k\times 1} = [1, 0, \ldots, 0]^T$, то $Q_k^T r_k = 0$. Если матрица A еще и положительно определена, то T_k обязана быть невырожденной матрицей и указанный выбор для x_k минимизирует $\|r_k\|_{A^{-1}}$ по всем $x_k \in \mathcal{K}_k$. Кроме того, $r_k = \pm\|r_k\|_2 q_{k+1}$.*

Доказательство. Чтобы упростить обозначения, опустим индекс k. Предположим, что матрица $T = Q^T A Q$ невырожденна, и пусть $x = QT^{-1}e_1\|b\|_2$ и $r = b - Ax$. Проверим, что $Q^T r = 0$ следующим вычислением:

$$\begin{aligned} Q^T r = Q^T(b - Ax) &= Q^T b - Q^T A x \\ &= e_1\|b\|_2 - Q^T A(QT^{-1}e_1\|b\|_2) \\ &\quad \text{поскольку первым столбцом матрицы } Q \text{ является} \\ &\quad \text{вектор } b/\|b\|_2, \text{ а прочие столбцы ортогональны к } b \\ &= e_1\|b\|_2 - (Q^T A Q)T^{-1}e_1\|b\|_2 \\ &= e_1\|b\|_2 - (T)T^{-1}e_1\|b\|_2 \quad \text{поскольку } Q^T A Q = T \\ &= 0. \end{aligned}$$

Предположим теперь, что A к тому же положительно определена. Тогда и T должна быть положительно определенной, а потому невырожденной (см. вопрос 6.13). Пусть $\hat{x} = x + Qz$ — другое возможное решение из $\mathcal{K}$ и пусть $\hat{r} = b - A\hat{x}$. Нужно показать, что $\|\hat{r}\|_{A^{-1}}$ достигает минимума при $z = 0$. Однако

$$\begin{aligned} \|\hat{r}\|_{A^{-1}}^2 &= \hat{r}^T A^{-1}\hat{r} \quad \text{по определению} \\ &= (r - AQz)^T A^{-1}(r - AQz) \\ &\qquad \text{так как } \hat{r} = b - A\hat{x} = b - A(x + Qz) = r - AQz \\ &= r^T A^{-1} r^T - 2(AQz)^T A^{-1} r + (AQz)^T A^{-1}(AQz) \\ &= \|r\|_{A^{-1}}^2 - 2z^T Q^T r + \|AQz\|_{A^{-1}}^2 \\ &\qquad \text{так как } (AQz)^T A^{-1} r = z^T Q^T A A^{-1} r = z^T Q^T r \\ &= \|r\|_{A^{-1}}^2 + \|AQz\|_{A^{-1}}^2 \quad \text{так как } Q^T r = 0, \end{aligned}$$

поэтому значение $\|\hat{r}\|_{A^{-1}}$ минимально в том и только том случае, если $AQz = 0$. Так как A невырожденна, а Q имеет полный столбцевой ранг, то соотношение $AQz = 0$ равносильно равенству $z = 0$.

Восстанавливая индексы, покажем, что $r_k = \pm\|r_k\|_2 q_{k+1}$. Поскольку $x_k \in \mathcal{K}_k$, имеем $r_k = b - Ax_k \in \mathcal{K}_{k+1}$. Таким образом, r_k есть линейная комбинация столбцов матрицы Q_{k+1}, потому что $\mathcal{K}_{k+1}$ натянуто на эти столбцы. Учитывая, что $Q_k^T r_k = 0$, заключаем, что q_{k+1} — это единственный столбец матрицы Q_{k+1}, к которому вектор r_k не ортогонален. □

6.6.3. Метод сопряженных градиентов

Наиболее предпочтительным методом для симметричных положительно определенных матриц является алгоритм сопряженных градиентов. Теорема 6.8 характеризует решение x_k, вычисляемое этим алгоритмом. Хотя MINRES мо-

жет выглядеть более естественным методом, чем CG, поскольку минимизирует $\|r_k\|_2$, а не $\|r_k\|_{A^{-1}}$, оказывается, что его реализация более трудоемка, он более подвержен численной неустойчивости, а потому часто дает приближенные решения худшего качества, чем CG. Мы увидим, что CG имеет то особенно привлекательное свойство, что в его реализации предусмотрено одновременное хранение в памяти лишь четырех векторов, а не k векторов $q_1, \ldots, q_k$. Кроме того, в его внутреннем цикле, помимо матрично-векторного произведения, выполняются только два скалярных произведения, три операции типа «saxpy» (прибавление кратного одного вектора к другому) и небольшое количество скалярных операций. Таким образом, и память, и работа в методе очень невелики.

А теперь мы выведем формулы метода. Это можно сделать несколькими различными способами. Мы начнем с алгоритма Ланцоша (алгоритм 6.10), вычисляющего столбцы ортогональной матрицы Q_k и элементы трехдиагональной матрицы T_k, а также с формулы $x_k = Q_k T_k^{-1} e_1 \|b\|_2$ из теоремы 6.8. Будет показано, как можно вычислить x_k непосредственно, пользуясь рекуррентными соотношениями для трех групп векторов. Одновременно в памяти хранятся лишь самые последние векторы из каждой группы, записываемые на места своих предшественников. Первая группа векторов — это приближенные решения x_k. Вторая группа — это невязки $r_k = b - Ax_k$; согласно теореме 6.8, они параллельны векторам Ланцоша q_{k+1}. Третья группа векторов — это *сопряженные градиенты* p_k. Эти векторы называются *градиентами* по следующей причине: шаг метода CG может интерпретироваться как выбор числа ν из условия, чтобы новое приближенное решение $x_k = x_{k-1} + \nu p_k$ минимизировало норму невязки $\|r_k\|_{A^{-1}} = (r_k^T A^{-1} r_k)^{1/2}$. Иными словами, p_k используются как *направления градиентного поиска*. Они называются *сопряженными*, или, точнее, *A-сопряженными*, потому что $p_k^T A p_j = 0$ при $j \neq k$. Другими словами, векторы p_k ортогональны по отношению к скалярному произведению, определяемому матрицей A (см. лемму 1.3).

Поскольку A — симметричная положительно определенная матрица, то же самое верно для $T_k = Q_k^T A Q_k$ (см. вопрос 6.13). Это означает, что к T_k можно применить алгоритм Холесского; в результате получим $T_k = L_k D_k L_k^T$, где L_k — нижняя двухдиагональная матрица с единичной диагональю, а D_k — диагональная матрица. Далее, используя формулу для x_k из теоремы 6.8, находим

$$\begin{aligned} x_k &= Q_k T_k^{-1} e_1 \|b\|_2 \\ &= Q_k (L_k^{-T} D_k^{-1} L_k^{-1}) e_1 \|b\|_2 \\ &= (Q_k L_k^{-T})(D_k^{-1} L_k^{-1} e_1 \|b\|_2) \\ &\equiv (\widetilde{P}_k)(y_k), \end{aligned}$$

где $\widetilde{P}_k \equiv Q_k L_k^{-T}$ и $y_k \equiv D_k^{-1} L_k^{-1} e_1 \|b\|_2$. Положим $\widetilde{P}_k = [\tilde{p}_1, \ldots, \tilde{p}_k]$. Сопряженные градиенты p_i окажутся параллельны столбцам $\tilde{p}_i$ матрицы $\widetilde{P}_k$. Теперь мы знаем достаточно для того, чтобы доказать следующую лемму:

Лемма 6.8. *Столбцы $\tilde{p}_i$ матрицы $\widetilde{P}_k$ являются A-сопряженными. Иначе говоря, матрица $\widetilde{P}_k^T A \widetilde{P}_k$ — диагональная.*

Доказательство. Имеем

$$\widetilde{P}_k^T A \widetilde{P}_k = (Q_k L_k^{-T})^T A (Q_k L_k^{-T}) = L_k^{-1}(Q_k^T A Q_k) L_k^{-T} = L_k^{-1}(T_k) L_k^{-T}$$
$$= L_k^{-1}(L_k D_k L_k^T) L_k^{-T} = D_k.$$

□

Выведем теперь простые рекурсии для столбцов матриц $\widetilde{P}_k$ и элементов векторов y_k. Мы покажем, что $y_{k-1} \equiv [\eta_1, \ldots, \eta_{k-1}]^T$ совпадает с первыми $k-1$ компонентами вектора $y_k = [\eta_1, \ldots, \eta_{k-1}, \eta_k]^T$, а $\widetilde{P}_{k-1}$ совпадает с первыми $k-1$ столбцами матрицы $\widetilde{P}_k$. Поэтому соотношение

$$x_k = \widetilde{P}_k \cdot y_k = [\widetilde{P}_{k-1}, \tilde{p}_k] \cdot \begin{bmatrix} y_{k-1} \\ \eta_k \end{bmatrix} = \widetilde{P}_{k-1} y_{k-1} + \tilde{p}_k \eta_k = x_{k-1} + \tilde{p}_k \eta_k \qquad (6.32)$$

может служить искомой рекурсией для векторов x_k.

Рекуррентное соотношение для величин η_k выводится следующим образом. Поскольку T_{k-1} является ведущей главной подматрицей порядка $k-1$ в T_k, матрицы L_{k-1} и D_{k-1} также будут ведущими главными подматрицами порядка $k-1$ соответственно в L_k и D_k:

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}$$
$$= L_k D_k L_k^T$$
$$= \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & d_{k-1} & \\ & & & d_k \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix}^T$$
$$= \begin{bmatrix} L_{k-1} & \\ l_{k-1}\hat{e}_{k-1}^T & 1 \end{bmatrix} \cdot \operatorname{diag}(D_{k-1}, d_k) \cdot \begin{bmatrix} L_{k-1} & \\ l_{k-1}\hat{e}_{k-1}^T & 1 \end{bmatrix}^T,$$

где размерность вектора $\hat{e}_{k-1}^T = [0, \ldots, 0, 1]$ равна $k-1$. Точно так же, D_{k-1}^{-1} и L_{k-1}^{-1} суть ведущие главные подматрицы порядка $k-1$ соответственно в $D_k^{-1} = \operatorname{diag}(D_{k-1}^{-1}, d_k^{-1})$ и

$$L_k^{-1} = \begin{bmatrix} L_{k-1}^{-1} & \\ \star & 1 \end{bmatrix}.$$

Выражения элементов последней строки матрицы L_k^{-1} не существенны для нас. Из сказанного следует, что вектор $y_{k-1} = D_{k-1}^{-1} L_{k-1}^{-1} \hat{e}_1 \|b\|_2$, где $\hat{e}_1$ имеет размерность $k-1$, совпадает с первыми $k-1$ компонентами вектора

$$y_k = D_k^{-1} L_k^{-1} e_1 \|b\|_2 = \begin{bmatrix} D_{k-1}^{-1} & \\ & d_k^{-1} \end{bmatrix} \cdot \begin{bmatrix} L_{k-1}^{-1} & \\ \star & 1 \end{bmatrix} \cdot e_1 \|b\|_2$$
$$= \begin{bmatrix} D_{k-1}^{-1} L_{k-1}^{-1} \hat{e}_1 \|b\|_2 \\ \eta_k \end{bmatrix} = \begin{bmatrix} y_{k-1} \\ \eta_k \end{bmatrix}.$$

Теперь нам нужна рекурсия для столбцов матрицы $\widetilde{P}_k = [\tilde{p}_1, \ldots, \tilde{p}_k]$. Поскольку матрица L_{k-1}^T — верхняя треугольная, то же самое верно в отношении матрицы L_{k-1}^{-T}, являющейся в L_k^{-T} ведущей главной подматрицей порядка $k-1$. Поэтому $\widetilde{P}_{k-1}$ совпадает с первыми $k-1$ столбцами матрицы

$$\widetilde{P}_k = Q_k L_k^{-T} = [Q_{k-1}, q_k] \begin{bmatrix} L_{k-1}^{-T} & \star \\ 0 & 1 \end{bmatrix} = [Q_{k-1} L_{k-1}^{-T}, \tilde{p}_k] = [\widetilde{P}_{k-1}, \tilde{p}_k].$$

Соотношение $\widetilde{P}_k = Q_k L_k^{-T}$ дает $\widetilde{P}_k L_k^T = Q_k$, откуда, приравнивая k-е столбцы обеих частей, получаем рекурсию

$$\tilde{p}_k = q_k - l_{k-1} \tilde{p}_{k-1}. \tag{6.33}$$

Итак, мы имеем рекурсии для векторов q_k (из алгоритма Ланцоша), векторов $\tilde{p}_k$ (из уравнения (6.33)) и приближенных решений x_k (из уравнения (6.32)). Все эти рекурсии являются *короткими*, т. е. для реализации каждой из них нужны лишь один или два предыдущих вектора. Поэтому вместе они дают способ вычисления приближенного решения x_k при хранении в памяти небольшого числа векторов и выполнении во внутреннем цикле малого числа скалярных произведений, операций типа saxpy и скалярных операций.

Чтобы получить алгоритм CG в окончательном виде, нужно еще немного упростить эти рекурсии. Так как, согласно теореме 6.8, векторы r_k и q_{k+1} параллельны, ланцошеву рекурсию для q_{k+1} можно заменить рекурсией $r_k = b - Ax_k$ или, что эквивалентно, рекурсией $r_k = r_{k-1} - \eta_k A\tilde{p}_k$ (получаемой умножением соотношения $x_k = x_{k-1} + \eta_k \tilde{p}_k$ на A и последующим вычитанием из тождества $b = b$). В результате получаем векторные рекурсии

$$r_k = r_{k-1} - \eta_k A\tilde{p}_k, \tag{6.34}$$

$$x_k = x_{k-1} + \eta_k \tilde{p}_k \quad \text{из уравнения (6.32)}, \tag{6.35}$$

$$\tilde{p}_k = q_k - l_{k-1}\tilde{p}_{k-1} \quad \text{из уравнения (6.33)}. \tag{6.36}$$

Чтобы исключить здесь q_k, сделаем подстановки $q_k = r_{k-1}/\|r_{k-1}\|_2$ и $p_k \equiv \|r_{k-1}\|_2 \tilde{p}_k$, что даст

$$\begin{aligned} r_k &= r_{k-1} - \frac{\eta_k}{\|r_{k-1}\|_2} Ap_k \\ &\equiv r_{k-1} - \nu_k Ap_k, \end{aligned} \tag{6.37}$$

$$\begin{aligned} x_k &= x_{k-1} + \frac{\eta_k}{\|r_{k-1}\|_2} p_k \\ &\equiv x_{k-1} + \nu_k p_k, \end{aligned} \tag{6.38}$$

$$\begin{aligned} p_k &= r_{k-1} - \frac{\|r_{k-1}\|_2 l_{k-1}}{\|r_{k-2}\|_2} \cdot p_{k-1} \\ &\equiv r_{k-1} + \mu_k \cdot p_{k-1}. \end{aligned} \tag{6.39}$$

Нам еще нужны формулы для чисел ν_k и μ_k. Как мы увидим, для них имеется несколько математически эквивалентных выражений через скалярные произведения векторов, вычисляемых алгоритмом. Наши окончательные формулы выбраны так, чтобы минимизировать число используемых скалярных произведений, а также потому, что они более устойчивы, чем альтернативные варианты.

Чтобы получить формулу для ν_k, мы вначале умножим обе части уравнения (6.39) слева на $p_k^T A$, а затем используем то обстоятельство, что векторы p_k и p_{k-1} являются A-сопряженными (лемма 6.8). Это дает

$$p_k^T A p_k = p_k^T A r_{k-1} + 0 = r_{k-1}^T A p_k. \tag{6.40}$$

Далее, умножим обе части уравнения (6.37) слева на r_{k-1}^T и используем соотношение $r_{k-1}^T r_k = 0$ (напомним, что векторы r_i параллельны столбцам ортогональной матрицы Q). Получим

$$\begin{aligned} \nu_k &= \frac{r_{k-1}^T r_{k-1}}{r_{k-1}^T A p_k} \\ &= \frac{r_{k-1}^T r_{k-1}}{p_k^T A p_k} \quad \text{согласно уравнению (6.40)} \end{aligned} \tag{6.41}$$

(Уравнение (6.41) можно было бы вывести и из свойства числа ν_k, указанного в теореме 6.8, а именно: ν_k минимизирует норму невязки

$$\begin{aligned} \|r_k\|_{A^{-1}} &= r_k^T A^{-1} r_k \\ &= (r_{k-1} - \nu_k A p_k)^T A^{-1} (r_{k-1} - \nu_k A p_k) \quad \text{согласно уравнению (6.37)} \\ &= r_{k-1}^T A^{-1} r_{k-1} - 2\nu_k p_k^T r_{k-1} + \nu_k^2 p_k^T A p_k. \end{aligned}$$

Это выражение есть квадратичная функция от ν_k, поэтому минимум легко находится приравниванием производной нулю и решением полученного уравнения относительно ν_k. Это дает

$$\begin{aligned} \nu_k &= \frac{p_k^T r_{k-1}}{p_k^T A p_k} \\ &= \frac{(r_{k-1} + \mu_k \cdot p_{k-1})^T r_{k-1}}{p_k^T A p_k} \quad \text{согласно уравнению (6.39)} \\ &= \frac{r_{k-1}^T r_{k-1}}{p_k^T A p_k}. \end{aligned}$$

Здесь было использовано соотношение $p_{k-1}^T r_{k-1} = 0$, вытекающее из того, что r_{k-1} ортогонален ко всем векторам из $\mathcal{K}_{k-1}$, включая вектор p_{k-1}.)

Чтобы получить формулу для μ_k, умножим обе части уравнения (6.39) слева на $p_{k-1}^T A$ и используем A-сопряженность векторов p_k и p_{k-1} (лемма 6.8). В результате, имеем

$$\mu_k = -\frac{p_{k-1}^T A r_{k-1}}{p_{k-1}^T A p_{k-1}}. \tag{6.42}$$

Недостаток этой формулы для μ_k состоит в том, что требуется еще одно скалярное произведение $p_{k-1}^T A r_{k-1}$ помимо двух, необходимых для вычисления ν_k. Поэтому мы выведем другую формулу, не использующую новых скалярных произведений.

Выведем вначале иную формулу для ν_k. Умножим обе части уравнения (6.37) слева на r_k^T, снова используем соотношение $r_{k-1}^T r_k = 0$ и разрешим полученное равенство относительно ν_k, что дает

$$\nu_k = -\frac{r_k^T r_k}{r_k^T A p_k}. \tag{6.43}$$

Окончательную формулу для μ_k найдем, приравнивая два выражения (6.41) и (6.43) для ν_{k-1} (заметим, что индекс здесь уменьшен на единицу), переупорядочивая и сравнивая с соотношением (6.42):

$$\begin{aligned}\mu_k &= -\frac{p_{k-1}^T A r_{k-1}}{p_{k-1}^T A p_{k-1}} \\ &= \frac{r_{k-1}^T r_{k-1}}{r_{k-2}^T r_{k-2}}.\end{aligned} \tag{6.44}$$

Объединяя рекурсии (6.37), (6.38) и (6.39), а также формулы (6.41) и (6.44), получим окончательную реализацию алгоритма сопряженных градиентов.

Алгоритм 6.11. *Алгоритм сопряженных градиентов:*

$k = 0$; $x_0 = 0$; $r_0 = b$; $p_1 = b$;
repeat
  $k = k + 1$
  $z = A \cdot p_k$
  $\nu_k = (r_{k-1}^T r_{k-1})/(p_k^T z)$
  $x_k = x_{k-1} + \nu_k p_k$
  $r_k = r_{k-1} - \nu_k z$
  $\mu_{k+1} = (r_k^T r_k)/(r_{k-1}^T r_{k-1})$
  $p_{k+1} = r_k + \mu_{k+1} p_k$
пока число $\|r_k\|_2$ *не станет достаточно малым*

Стоимость внутреннего цикла алгоритма составляют одно матрично-векторное произведение $z = A \cdot p_k$, два скалярных произведения (при этом нужно сохранить значение $r_k^T r_k$ для последующей итерации), три операции типа saxpy и несколько скалярных операций. В хранении нуждаются лишь текущие значения векторов r, x, p и $z = Ap$. Более подробно о деталях реализации, включая то, как определить, что «$\|r_k\|_2$ достаточно мало», можно узнать по адресу NETLIB/templates/Templates.html.

6.6.4. Анализ сходимости метода сопряженных градиентов

Мы начнем с анализа сходимости алгоритма CG, основанного лишь на значении числа обусловленности матрицы A. Этот анализ покажет, что число итераций метода, необходимых для снижения ошибки в фиксированное число раз, большее единицы, пропорционально квадратному корню из числа обусловленности. Такой анализ наихудшего случая дает хорошую оценку скорости сходимости метода для нашей модельной задачи, т. е. уравнения Пуассона. Однако он серьезно *недооценивает* скорость сходимости во многих других случаях.

После вывода оценки, основанной на числе обусловленности, мы укажем, когда можно рассчитывать на более быструю сходимость.

В качестве начального приближенного решения возьмем $x_0 = 0$. Вспомним, что x_k минимизирует A^{-1}-норму невязки $r_k = b - Ax_k$ по всем возможным выборам $x_k \in \mathcal{K}_k(A, b)$. Это означает, что x_k минимизирует величину

$$\|b - Az\|_{A^{-1}}^2 \equiv f(z) = (b - Az)^T A^{-1}(b - Az) = (x - z)^T A(x - z)$$

по всем векторам $z \in \mathcal{K}_k = \operatorname{span}[b, Ab, A^2b, \ldots, A^{k-1}b]$. Всякий вектор $z \in \mathcal{K}_k(A, b)$ может быть записан в виде $z = \sum_{j=0}^{k-1} \alpha_j A^j b = p_{k-1}(A)b = p_{k-1}(A)Ax$, где $p_{k-1}(\xi) = \sum_{j=0}^{k-1} \alpha_j \xi^j$ — многочлен степени $k-1$. Поэтому

$$\begin{aligned} f(z) &= [(I - p_{k-1}(A)A)x]^T A[(I - p_{k-1}(A)A)x] \\ &\equiv (q_k(A)x)^T A(q_k(A)x) \\ &= x^T q_k(A) A q_k(A) x, \end{aligned}$$

где $q_k(\xi) \equiv 1 - p_{k-1}(\xi) \cdot \xi$ — многочлен степени k, такой, что $q_k(0) = 1$. Заметим, что $(q_k(A))^T = q_k(A)$, так как $A = A^T$. Обозначим через $\mathcal{Q}_k$ множество всех многочленов степени k, принимающих значение 1 в нуле. Тогда

$$f(x_k) = \min_{z \in \mathcal{K}_k} f(z) = \min_{q_k \in \mathcal{Q}_k} x^T q_k(A) A q_k(A) x. \tag{6.45}$$

Чтобы упростить это выражение, введем спектральное разложение $A = Q\Lambda Q^T$ и положим $Q^T x = y$. Получим

$$\begin{aligned} f(x_k) = \min_{z \in \mathcal{K}_k} f(z) &= \min_{q_k \in \mathcal{Q}_k} x^T (q_k(Q\Lambda Q^T))(Q\Lambda Q^T)(q_k(Q\Lambda Q^T))x \\ &= \min_{q_k \in \mathcal{Q}_k} x^T (Q q_k(\Lambda) Q^T)(Q\Lambda Q^T)(Q q_k(\Lambda) Q^T)x \\ &= \min_{q_k \in \mathcal{Q}_k} y^T q_k(\Lambda)\Lambda q_k(\Lambda) y \\ &= \min_{q_k \in \mathcal{Q}_k} y^T \cdot \operatorname{diag}(q_k(\lambda_i)\lambda_i q_k(\lambda_i)) \cdot y \\ &= \min_{q_k \in \mathcal{Q}_k} \sum_{i=1}^{n} y_i^2 \lambda_i (q_k(\lambda_i))^2 \\ &\le \min_{q_k \in \mathcal{Q}_k} \left(\max_{\lambda_i \in \lambda(A)} (q_k(\lambda_i))^2 \right) \sum_{i=1}^{n} y_i^2 \lambda_i \\ &= \min_{q_k \in \mathcal{Q}_k} \left(\max_{\lambda_i \in \lambda(A)} (q_k(\lambda_i))^2 \right) f(x_0), \end{aligned}$$

так как из $x_0 = 0$ следует, что $f(x_0) = x^T A x = y^T \Lambda y = \sum_{i=1}^n y_i^2 \lambda_i$. Поэтому

$$\frac{\|r_k\|_{A^{-1}}^2}{\|r_0\|_{A^{-1}}^2} = \frac{f(x_k)}{f(x_0)} \le \min_{q_k \in \mathcal{Q}} \max_{\lambda_i \in \lambda(A)} (q_k(\lambda_i))^2,$$

или

$$\frac{\|r_k\|_{A^{-1}}}{\|r_0\|_{A^{-1}}} \le \min_{q_k \in \mathcal{Q}} \max_{\lambda_i \in \lambda(A)} |q_k(\lambda_i)|.$$

Итак, вопрос о скорости сходимости алгоритма CG сведен к вопросу о многочленах: насколько мал может быть многочлен $q_k(\xi)$ степени k, удовлетворяющий условию $q_k(0) = 1$, когда ξ пробегает множество собственных значений матрицы A? Поскольку A положительно определена, ее собственные значения принадлежат отрезку $[\lambda_{\min}, \lambda_{\max}]$, где $0 < \lambda_{\min} \leq \lambda_{\max}$. Чтобы получить простую верхнюю оценку, будем вместо $q_k(\xi)$ искать многочлен $\hat{q}_k(\xi)$, как можно более малый на всем отрезке $[\lambda_{\min}, \lambda_{\max}]$ и принимающий значение 1 в нуле. Многочлен $\hat{q}_k(\xi)$ с этими свойствами легко конструируется из чебышевских многочленов $T_k(\xi)$, обсуждавшихся в разд. 6.5.6. Вспомним, что $|T_k(\xi)| \leq 1$ при $|\xi| \leq 1$ и $T_k(\xi)$ быстро возрастает по модулю при $|\xi| > 1$ (см. рис. 6.6). Положим теперь

$$\hat{q}_k(\xi) = T_k\left(\frac{\lambda_{\max} + \lambda_{\min} - 2\xi}{\lambda_{\max} - \lambda_{\min}}\right) / T_k\left(\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}\right).$$

Легко видеть, что $\hat{q}_k(0) = 1$ и если $\xi \in [\lambda_{\min}, \lambda_{\max}]$, то

$$\left|\frac{\lambda_{\max} + \lambda_{\min} - 2\xi}{\lambda_{\max} - \lambda_{\min}}\right| \leq 1.$$

Поэтому

$$\begin{aligned}\frac{\|r_k\|_{A^{-1}}}{\|r_0\|_{A^{-1}}} &\leq \min_{q_k \in \mathcal{Q}} \max_{\lambda_i \in \lambda(A)} |q_k(\lambda_i)| \\ &\leq \frac{1}{T_k(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}})} = \frac{1}{T_k(\frac{\kappa+1}{\kappa-1})} = \frac{1}{T_k(1+\frac{2}{\kappa-1})}, \end{aligned} \tag{6.46}$$

где $\kappa = \lambda_{\max}/\lambda_{\min}$ есть число обусловленности матрицы A.

Если число обусловленности κ близко́ к 1, то величина $1 + 2/(\kappa - 1)$ велика, а число $1/T_k(1 + \frac{2}{\kappa-1})$ мало; поэтому сходимость будет быстрой. Если κ велико, то сходимость замедляется, причем A^{-1}-норма невязки r_k стремится к нулю со скоростью

$$\frac{1}{T_k(1 + \frac{2}{\kappa-1})} \leq \frac{2}{1 + \frac{2k}{\sqrt{\kappa-1}}}.$$

Пример 6.14. Для модельной задачи на сетке $N \times N$ имеем $\kappa = O(N^2)$, поэтому после k шагов метода CG модуль невязки приобретает множитель, приблизительно равный $(1 - O(N^{-1}))^k$, т.е. такой же, как в методе SOR с оптимальным значением параметра верхней релаксации ω. Другими словами, для сходимости метода требуются $O(N) = O(n^{1/2})$ итераций. Поскольку стоимость каждой итерации равна $O(n)$, общая стоимость алгоритма составляет $O(n^{3/2})$. Это объясняет элемент таблицы 6.1, отвечающий методу *CG*. ◇

Проведенный анализ, опиравшийся на число обусловленности, объясняет не все важные особенности в характере сходимости метода CG. Как показывает следующий пример, имеет значение все распределение собственных чисел, а не только отношение наибольшего из них к наименьшему.

Пример 6.15. Рассмотрим график относительной невязки $\|r_k\|_2/\|r_0\|_2$ на шаге k алгоритма CG для восьми различных линейных систем; этот график показан на рис 6.7. Относительная невязка $\|r_k\|_2/\|r_0\|_2$ измеряет скорость сходимости. Наша реализация метода прекращает работу, когда это отношение

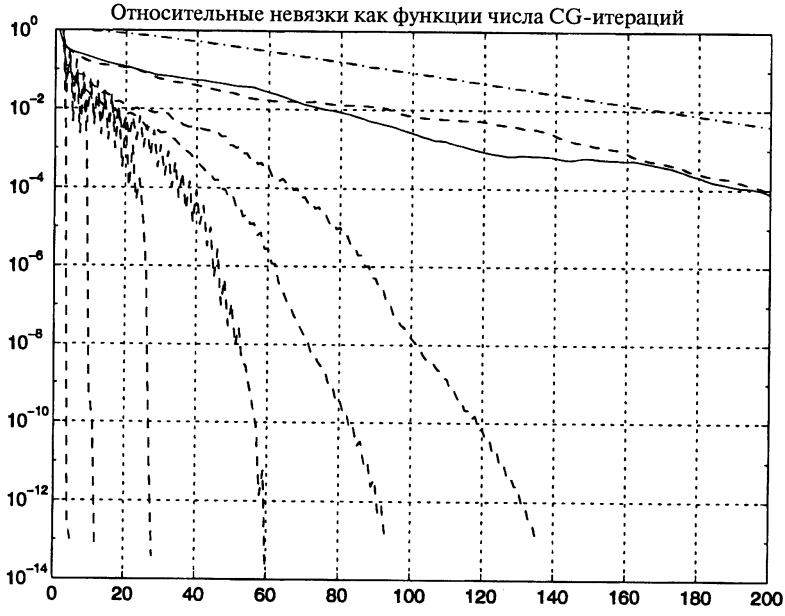


Рис. 6.7. График относительных невязок, вычисляемых в алгоритме CG.

становится меньше, чем 10^{-13}, либо когда уже проделаны $k = 200$ шагов. Выход происходит по первому выполненному из этих условий.

Все восемь линейных систем имеют одну и ту же размерность $n = 10^4$ и одно и то же число обусловленности $\kappa \approx 4134$; тем не менее характер сходимости метода для них радикально различен. Самая верхняя линия (состоящая из точек и тире) соответствует величине $1/T_k(1+\frac{2}{\kappa-1})$; согласно неравенству (6.46), эта величина является верхней границей для $\|r_k\|_{A^{-1}}/\|r_0\|_{A^{-1}}$. Оказывается, что графики отношений $\|r_k\|_2/\|r_0\|_2$ и $\|r_k\|_{A^{-1}}/\|r_0\|_{A^{-1}}$ примерно одинаковы, поэтому на рисунке представлены графики первого отношения, допускающего более простую интерпретацию.

Сплошной линией изображено отношение $\|r_k\|_2/\|r_0\|_2$ для уравнения Пуассона на сетке 100×100 со случайной правой частью b. Мы видим, что верхняя граница отражает общий характер сходимости. Семь штриховых линий — это графики отношения $\|r_k\|_2/\|r_0\|_2$ для семи диагональных линейных систем $D_i x = b$, пронумерованных от D_1 (самый левый график) до D_7 (самый правый график). Все матрицы D_i имеют ту же размерность и то же число обусловленности, что и уравнение Пуассона, поэтому, чтобы понять различия в характере сходимости, нужно присмотреться к этим матрицам внимательнее.

Матрица D_i построена таким образом, чтобы m_i ее наименьших и m_i наибольших собственных значений совпадали с соответствующими собственными значениями уравнения Пуассона, а остальные $n - 2m_i$ собственных значений были равны среднему геометрическому из наибольшего и наименьшего собственных чисел. Другими словами, D_i имеет лишь $d_i = 2m_i + 1$ различных

собственных значений. Обозначим через k_i число CG-итераций, требуемых для того, чтобы приближенное решение системы $D_i x = b$ удовлетворило условию $\|r_k\|_2/\|r_0\|_2 \leq 10^{-13}$. Данные о сходимости представлены в таблице.

Номер примера	i	1	2	3	4	5	6	7
Количество различных собственных значений	d_i	3	11	41	81	201	401	5000
Количество шагов, требуемых для сходимости	k_i	3	11	27	59	94	134	>200

Мы видим, что число k_i шагов, требуемых для сходимости, растет вместе с числом d_i различных собственных значений. Матрица D_7 имеет тот же спектр, что и уравнение Пуассона, и сходимость для нее примерно столь же медленная.

Мы утверждаем, что, в отсутствие округлений, методу CG для сходимости потребовалось бы *ровно* $k_i = d_i$ шагов. Объяснение состоит в том, что можно найти многочлен $q_{d_i}(\xi)$ степени d_i, обращающийся в нуль на собственных значениях α_j матрицы A и равный 1 в нуле, а именно

$$q_{d_i}(\xi) = \frac{\prod_{j=1}^{d_i}(\alpha_j - \xi)}{\prod_{j=1}^{d_i}(\alpha_j)}.$$

Согласно уравнению (6.45), результат d_i шагов метода CG минимизирует величину $\|r_{d_i}\|_{A^{-1}}^2 = f(x_{d_i})$ по всем многочленам степени d_i, принимающим значение 1 в нуле. Поскольку q_{d_i} — один из таких многочленов и $q_{d_i}(A) = 0$, должно быть $\|r_{d_i}\|_{A^{-1}}^2 = 0$, или $r_{d_i} = 0$. ◇

Один из уроков примера 6.15 заключается в том, что если число собственных значений на периферии спектра невелико (или же спектр сконденсирован на краях), то метод CG сходится гораздо быстрее, чем показывает анализ, основанный лишь на числе обусловленности.

Другим уроком является то, что поведение метода в арифметике с плавающей точкой значительно отличается от его поведения в точной арифметике. Мы можем видеть это по тому, что число d_i различных собственных значений в ряде случаев отличается от числа k_i шагов, требуемых для сходимости, хотя, как было показано, теоретически эти числа должны совпадать. Все же d_i и k_i были величинами одного порядка для всех систем.

В действительности, если бы метод CG был проведен в точной арифметике и полученные решения и невязки были сравнены с аналогичными величинами, вычисленными в машинной арифметике, то, скорей всего, результаты с ростом номера k итерации быстро стали бы совершенно различны. Все же, если матрица A не слишком плохо обусловлена, результаты с плавающей точкой в конечном счете сойдутся к искомому решению системы, поэтому метод остается очень полезным. Тот факт, что точные и машинные (промежуточные) результаты могут быть совсем не похожи, интересен, но не препятствует практическому применению метода.

Вслед за открытием метода CG было доказано, что в точной арифметике он дает точное решение системы за n шагов, поскольку невязка r_{n+1} должна быть ортогональна к другим n ортогональным векторам $r_1, \ldots, r_n$, а потому равна 0. Иначе говоря, метод CG воспринимался как *прямой*, а не как *итерационный*.

Поскольку на практике (точная) сходимость за n шагов не наблюдалась, метод стал рассматриваться как неустойчивый, а затем был на много лет заброшен. В конечном счете, было признано, что CG — это очень хороший итерационный метод, часто дающий приближенные решения высокой точности уже после $k \ll n$ шагов.

Не так давно был выполнен тонкий обратный анализ ошибок, объясняющий наблюдаемое поведение метода CG в арифметике с плавающей точкой и его возможное отличие от поведения в точной арифметике [123]. Картина сходимости метода может включать в себя протяженные участки «плоскогорья», где $\|r_k\|_2$ почти не уменьшается в течение многих итераций, перемежаемые периодами быстрой сходимости. Эту картину можно объяснить, показав, что метод CG, применяемый к системе $Ax = b$ в арифметике с плавающей точкой, ведет себя *в точности* так, как тот же метод, применяемый в точной арифметике к системе $\widetilde{A}\tilde{x} = \tilde{b}$. Матрица $\widetilde{A}$ близка к A в следующем смысле: ее порядок намного больше порядка A, однако все собственные значения матрицы $\widetilde{A}$ расположены узкими кластерами возле собственных значений матрицы A. Таким образом, плоскогорья в картине сходимости соответствуют тому, что многочлен q_k, лежащий в основе метода CG, приобретает все больше и больше нулей возле кластеров собственных значений матрицы $\widetilde{A}$.

6.6.5. Предобуславливание

В предыдущем разделе мы видели, что скорость сходимости метода CG зависит от числа обусловленности матрицы A или, более общо, от распределения ее собственных значений. Тем же свойством обладают и другие методы крыловского подпространства. *Предобуславливание* означает, что система $Ax = b$ заменяется системой $M^{-1}Ax = M^{-1}b$, где матрица M есть приближение к A со следующими свойствами:

1. M симметрична и положительно определена.
2. Матрица $M^{-1}A$ хорошо обусловлена или же имеет небольшое число собственных значений на периферии своего спектра.
3. Система $Mx = b$ решается легко.

Продуманный, зависящий от конкретной задачи выбор M может сделать число обусловленности матрицы $M^{-1}A$ много меньшим числа обусловленности матрицы A и тем самым в огромной степени ускорить сходимость. В действительности, хороший предобуславливатель часто необходим для того, чтобы итерационный метод вообще сходился. Поэтому большая часть современных исследований по итерационным методам ориентирована на отыскание эффективных предобуславливателей (см. об этом также разд. 6.10).

Мы не можем применить метод CG непосредственно к системе $M^{-1}Ax = M^{-1}b$, поскольку матрица $M^{-1}A$ в общем случае не является симметричной. Мы построим *предобусловленный метод сопряженных градиентов* следующим образом. Пусть $M = Q\Lambda Q^T$ — спектральное разложение матрицы M; положим $M^{1/2} \equiv Q\Lambda^{1/2}Q^T$. Заметим, что матрица $M^{1/2}$ также симметрична и положительно определена и $(M^{1/2})^2 = M$. Теперь умножим систему $M^{-1}Ax = M^{-1}b$ на матрицу $M^{1/2}$, что дает новую симметричную положительно определен-

ную систему $(M^{-1/2}AM^{-1/2})(M^{1/2}x) = M^{-1/2}b$, или $\widehat{A}\hat{x} = \hat{b}$. Отметим, что $\widehat{A}$ и $M^{-1}A$ имеют одни и те же собственные значения, поскольку эти матрицы подобны ($M^{-1}A = M^{-1/2}\widehat{A}M^{1/2}$). Теперь применим метод CG к системе $\widehat{A}\hat{x} = \hat{b}$ *неявно*, таким образом, чтобы не было необходимости в умножениях на $M^{-1/2}$. Это приводит к следующему алгоритму:

Алгоритм 6.12. *Предобусловленный алгоритм CG:*

$k = 0$; $x_0 = 0$; $r_0 = b$; $p_1 = M^{-1}b$; $y_0 = M^{-1}r_0$
repeat
$k = k + 1$
$z = A \cdot p_k$
$\nu_k = (y_{k-1}^T r_{k-1})/(p_k^T z)$
$x_k = x_{k-1} + \nu_k p_k$
$r_k = r_{k-1} - \nu_k z$
$y_k = M^{-1}r_k$
$\mu_{k+1} = (y_k^T r_k)/(y_{k-1}^T r_{k-1})$
$p_{k+1} = y_k + \mu_{k+1}p_k$
пока число $\|r_k\|_2$ *не станет достаточно малым*

Теорема 6.9. *Пусть A и M — симметричные положительно определенные матрицы. Положим $\widehat{A} = M^{-1/2}AM^{1/2}$ и $\hat{b} = M^{-1/2}b$. Алгоритм CG в применении к системе $\widehat{A}\hat{x} = \hat{b}$, т. е. алгоритм, описываемый формулами*

$k = 0$; $\hat{x}_0 = 0$; $\hat{r}_0 = \hat{b}$; $\hat{p}_1 = \hat{b}$;
repeat
$k = k + 1$
$\hat{z} = \widehat{A} \cdot \hat{p}_k$
$\hat{\nu}_k = (\hat{r}_{k-1}^T \hat{r}_{k-1})/(\hat{p}_k^T \hat{z})$
$\hat{x}_k = \hat{x}_{k-1} + \hat{\nu}_k \hat{p}_k$
$\hat{r}_k = \hat{r}_{k-1} - \hat{\nu}_k \hat{z}$
$\hat{\mu}_{k+1} = (\hat{r}_k^T \hat{r}_k)/(\hat{r}_{k-1}^T \hat{r}_{k-1})$
$\hat{p}_{k+1} = \hat{r}_k + \hat{\mu}_{k+1}\hat{p}_k$
пока число $\|\hat{r}_k\|_2$ *не станет достаточно малым*

и алгоритм 6.12 связаны следующим образом:

$$\begin{aligned}
\hat{\mu}_k &= \mu_k, \\
\hat{\nu}_k &= \nu_k, \\
\hat{z} &= M^{-1/2}z, \\
\hat{x}_k &= M^{1/2}x_k, \\
\hat{r}_k &= M^{-1/2}r_k, \\
\hat{p}_k &= M^{1/2}p_k.
\end{aligned}$$

Следовательно, векторы x_k сходятся к произведению матрицы $M^{-1/2}$ и решения системы $\widehat{A}\hat{x} = \hat{b}$, т. е. к вектору $M^{-1/2}\widehat{A}^{-1}\hat{b} = A^{-1}b$.

По поводу доказательства теоремы см. вопрос 6.14.

Теперь мы опишем некоторые распространенные типы предобуславливателей. Заметим, что преследуемые нами две цели — минимизация числа обусловленности матрицы $M^{-1}A$ и сохранение простоты решения системы $Mx = b$, —

вступают в противоречие друг с другом: выбор $M = A$ минимизирует число обусловленности матрицы $M^{-1}A$, но оставляет систему $Mx = b$ столь же трудной для решения, что и исходная система. Выбор $M = I$ делает решение системы $Mx = b$ тривиальным, но не меняет числа обусловленности матрицы $M^{-1}A$. Поскольку система с матрицей M решается во внутреннем цикле алгоритма, мы ограничимся обсуждением матриц M, для которых подобные системы легко решаются, и укажем те из них, что потенциально могут уменьшить число обусловленности произведения $M^{-1}A$.

- Если диагональные элементы матрицы A сильно различаются по величине, то можно использовать простой *диагональный предобуславливатель* $M = \mathrm{diag}(a_{11}, \ldots, a_{nn})$. Можно показать, что такой выбор дает число обусловленности произведения $M^{-1}A$, не более чем в n раз превышающее минимальное значение по всем возможным диагональным предобуславливателям [244]. Этот выбор M называют еще *предобуславливанием Якоби*.
- Обобщая первый тип предобуславливания, станем рассматривать A как блочную матрицу с квадратными диагональными блоками A_{ii}:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1k} \\ \vdots & \ddots & \vdots \\ A_{k1} & \cdots & A_{kk} \end{bmatrix}.$$

  Тогда среди всех блочно-диагональных предобуславливателей

$$M = \begin{bmatrix} M_{11} & & \\ & \ddots & \\ & & M_{kk} \end{bmatrix}$$

  с блоками M_{ii} тех же размерностей, что и A_{ii}, выбор $M_{ii} = A_{ii}$ с точностью до множителя k минимизирует число обусловленности матрицы $M^{-1/2}AM^{-1/2}$ [69]. Этот выбор называют также *блочным предобуславливанием Якоби*.
- Подобно методу Якоби, метод SSOR может быть использован в качестве (блочного) предобуславливателя.
- Под *неполным разложением Холесского* LL^T матрицы A понимают приближение $A \approx LL^T$, в котором L ограничена требованием иметь конкретный тип разреженности, например тот, что имеет исходная матрица A. Другими словами, не допускается заполнения в ходе алгоритма Холесского. Произведение LL^T используется в качестве матрицы M. (Для несимметричных систем существует соответствующий *неполный LU предобуславливатель*.)
- Если матрица A представляет уравнение (например, уравнение Пуассона) на физической области Ω, то используется *декомпозиция области*. До сих пор в случае уравнения Пуассона Ω была для нас единичным квадратом. Более общо, область Ω может быть разбита на непересекающиеся (или немного перекрывающиеся) подобласти $\Omega = \cup_j \Omega_j$, на каждой из которых уравнение может быть решено независимо от других подобластей. Пусть, например, решается уравнение Пуассона и подобластями являются квадраты или прямоугольники. Тогда подзадачи могут быть решены очень быстро

с помощью алгоритма FFT (см. разд. 6.7). Эти подзадачи соответствуют использованию блочно-диагональной матрицы M (если подобласти не пересекаются) или произведения таких матриц (если подобласти перекрываются). Более подробно эти вопросы обсуждаются в разд. 6.10.

Многие предобуславливатели описанных типов реализованы в программных пакетах PETSc [232] и PARPRE (NETLIB/scalapack/parpre.tar.gz).

6.6.6. Другие алгоритмы крыловского подпространства для решения системы $Ax = b$

До сих пор мы были сосредоточены на решении симметричных положительно определенных линейных систем и минимизации A^{-1}-нормы невязки. В этом разделе будут описаны методы для других типов линейных систем и даны советы о том, какой метод следует использовать, основанные на простых свойствах матрицы. Наши выводы суммируются на рис. 6.8 [15, 107, 136, 214]. Детали, более полные рекомендации по поводу выбора метода и программы можно найти в NETLIB/templates.

Всякую систему $Ax = b$ можно преобразовать в симметричную положительно определенную систему $A^TAx = A^Tb$ (или систему $AA^Ty = b$, $x = A^Ty$), называемую нормальными уравнениями. Это относится и к задаче наименьших квадратов $\min_x ||Ax - b||_2$. Указанное преобразование позволяет применить метод CG, если есть возможность умножать векторы и на A, и на A^T. Число обусловленности каждой из матриц A^TA и AA^T равно квадрату числа обусловленности матрицы A, поэтому CG будет сходиться медленно, если A плохо обусловлена, и быстро, если A обусловлена хорошо (или же A^TA имеет «хорошее» распределение собственных значений в смысле, обсуждавшемся в разд. 6.6.4).

Если A — симметричная положительно определенная матрица, то вместо A^{-1}-нормы невязки можно минимизировать ее 2-норму. Эта установка определяет алгоритм минимальных невязок, или MINRES [194]. Поскольку MINRES более трудоемок, чем CG, и часто менее точен вследствие численной неустойчивости, он не используется для положительно определенных систем. Однако, в отличие от CG, MINRES может быть применен к симметричной незнакоопределенной матрице. Для этого случая может быть также использован алгоритм SYMMLQ, предложенный Пэйджем и Сондерсом [194]; в нем на каждом шаге для невязки обеспечивается условие $r_k \perp \mathcal{K}_k(A, b)$.

К сожалению, существует немного матриц, отличающихся от симметричных и допускающих алгоритмы типа CG, которые одновременно обладают такими свойствами:

1. минимизируется норма невязки $||r_k||_2$ либо поддерживается условие $r_k \perp \mathcal{K}_k(A, b)$;
2. во внутреннем цикле используется фиксированное, не зависящее от k число скалярных произведений и операций типа saxpy.

По существу, алгоритмы с этими двумя свойствами возможны лишь для матриц вида $e^{i\theta}(T + \sigma I)$, где $T = T^T$ (либо $TH = (HT)^T$ для некоторой симметричной положительно определенной матрицы H), θ — вещественное, а σ —

комплексное число [102, 251]. Для таких симметричных или специальных несимметричных матриц A, как оказывается, можно найти короткие рекурсии типа рекурсий алгоритма Ланцоша, позволяющие вычислить ортогональный базис $[q_1, \ldots, q_n]$ подпространства $\mathcal{K}_k(A, b)$. То обстоятельство, что рекурсия, модифицирующая векторы q_k, содержит лишь несколько членов, означает высокую эффективность метода.

Для несимметричных матриц A общего вида подобных коротких рекурсий не существует. Для таких матриц можно применить алгоритм Арнольди. Тогда вместо трехдиагональной матрицы $T_k = Q_k^T A Q_k$ получим полноценную верхнюю хессенбергову матрицу $H_k = Q_k^T A Q_k$. Это разложение используется в алгоритме GMRES (generalized minimum residual) для выбора вектора $x_k = Q_k y_k \in \mathcal{K}_k(A, b)$, минимизирующего невязку

$$
\begin{aligned}
\|r_k\|_2 &= \|b - Ax_k\|_2 \\
&= \|b - AQ_k y_k\|_2 \\
&= \|b - (QHQ^T)Q_k y_k\|_2 \quad \text{согласно уравнению (6.30)} \\
&= \|Q^T b - HQ^T Q_k y_k\|_2, \quad \text{так как матрица } Q \text{ ортогональна,} \\
&= \left\| e_1 \|b\|_2 - \begin{bmatrix} H_k & H_{uk} \\ H_{ku} & H_u \end{bmatrix} \cdot \begin{bmatrix} y_k \\ 0 \end{bmatrix} \right\|_2 \\
&\qquad \text{согласно уравнению (6.30), а также потому, что первым} \\
&\qquad \text{столбцом матрицы } Q = [Q_k, Q_u] \text{ является вектор } b/\|b\|_2 \\
&= \left\| e_1 \|b\|_2 - \begin{bmatrix} H_k \\ H_{ku} \end{bmatrix} y_k \right\|_2 .
\end{aligned}
$$

В блоке H_{ku} отлична от нуля только первая строка, поэтому для элементов вектора y_k получена $(k+1) \times k$-задача наименьших квадратов с верхней хессенберговой матрицей. В силу этого свойства, QR-разложение, необходимое для решения задачи, может быть найдено посредством k плоских вращений за $O(k^2)$ операций вместо $O(k^3)$ в общем случае. Кроме того, для хранения Q_k требуется $O(kn)$ слов памяти. Один из способов ограничения трудоемкости и памяти состоит в том, чтобы *обновлять* метод. Это значит, что берется приближенное решение x_k, вычисленное после k шагов, GMRES применяется к решению линейной системы $Ad = r_k = b - Ax_k$, а затем находится новое приближенное решение $x_k + d$. Описанный алгоритм называется GMRES(k). Однако даже этот алгоритм более трудоемок, чем CG, где стоимость внутреннего цикла вообще не зависит от k.

Другой подход к решению несимметричных линейных систем заключается в том, чтобы отказаться от задачи вычисления ортонормированного базиса в подпространстве $\mathcal{K}_k(A, b)$ и строить вместо этого неортогональный базис, в котором бы A снова приводилась к (несимметричной) трехдиагональной форме. Этот подход называется *несимметричным методом Ланцоша*; он требует возможности формировать матрично-векторные произведения с участием как матрицы A, так и матрицы A^T. Это существенное требование, поскольку вычисление произведений вида $A^T z$ подчас затруднительно или вообще невозможно (см. пример 6.13). Преимущество трехдиагональной формы — в том, что системы с трехдиагональными матрицами легче решать, чем хессенберговы.

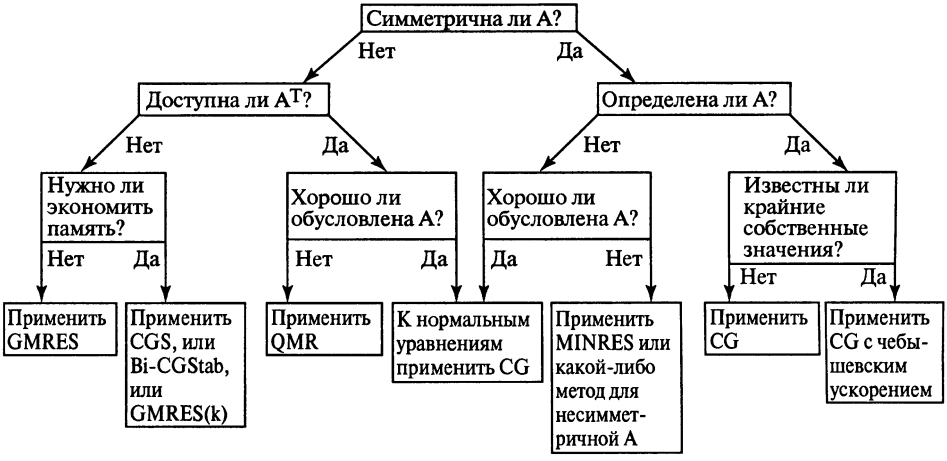


Рис. 6.8. Дерево решений при выборе итерационного алгоритма для системы $Ax = b$. Приняты аббревиатуры: Bi-CGStab = стабилизированный метод бисопряженных градиентов, QMR = метод квазиминимальных невязок; CGS = CG squared.

Недостаток же — в том, что базисные векторы могут быть очень плохо обусловленными. Иногда базисный вектор вообще не удается определить; это явление называется *обрывом*. Потенциальная эффективность данного подхода вызвала к жизни большое число исследований, посвященных тому, чтобы избежать этого типа неустойчивости или ослабить ее (*методы Ланцоша с заглядыванием вперед*), а также конкурирующие методы, среди которых отметим методы *бисопряженных градиентов* и *квазиминимальных невязок*. Существуют и версии этих методов, не требующие произведений с участием матрицы A^T, в том числе методы CGS (conjugate gradients squared) и *стабилизированный метод бисопряженных градиентов*. Ни один метод не является наилучшим для всех случаев.

На рис. 6.8 показано дерево решений, дающее простые указания, какой метод следует попробовать первым, если более подробной информации о матрице A (например, что A возникает из уравнения Пуассона) не имеется.

6.7. Быстрое преобразование Фурье

В данном разделе символ i всегда обозначает $\sqrt{-1}$.

Мы покажем вначале, как можно решить двумерное уравнение Пуассона, используя умножения на матрицу собственных векторов для T_N. Прямолинейная реализация этого подхода стоила бы $O(N^3) = O(n^{3/2})$ операций, что чересчур дорого. Затем мы объясним, как этот подход может быть реализован с помощью алгоритма FFT за $O(N^2 \log N) = O(n \log n)$ операций, что лишь множителем $\log n$ отличается от оптимального результата.

Данный подход является дискретным аналогом метода рядов Фурье, применяемого к исходным дифференциальным уравнениям (6.1) или (6.6). Позднее мы уточним характер этой аналогии.

Пусть $T_N = Z\Lambda Z^T$ — спектральное разложение матрицы T_N, указанное в лемме 6.1. Будем работать с записью двумерного уравнения Пуассона в виде уравнения (6.11):

$$T_N V + V T_N = h^2 F.$$

Подставляя сюда $T_N = Z\Lambda Z^T$ и умножая слева на Z^T, а справа на Z, получим

$$Z^T(Z\Lambda Z^T)VZ + Z^T V(Z\Lambda Z^T)Z = Z^T(h^2F)Z,$$

или

$$\Lambda V' + V'\Lambda = h^2 F',$$

где $V' = Z^T V Z$ и $F' = Z^T F Z$. Элемент (j,k) в последнем равенстве равен

$$(\Lambda V' + V'\Lambda)_{jk} = \lambda_j v'_{jk} + v'_{jk}\lambda_k = h^2 f'_{jk}.$$

Определяя отсюда v'_{jk}, находим

$$v'_{jk} = \frac{h^2 f'_{jk}}{\lambda_j + \lambda_k}.$$

Это дает первый вариант нашего алгоритма.

Алгоритм 6.13. *Решение двумерного уравнения Пуассона с помощью спектрального разложения* $T_N = Z\Lambda Z^T$*:*

1) $F' = Z^T F Z$;
2) *для всех* j *и* k *вычислить* $v'_{jk} = \frac{h^2 f'_{jk}}{\lambda_j + \lambda_k}$;
3) $V = ZV'Z^T$

Стоимость шага 2 составляет $3N^2 = 3n$ операций; шаги 1 и 3 состоят из 4 матрично-матричных умножений с участием матриц Z и $Z^T = Z$; их стоимость при обычном алгоритме умножения равна $8N^3 = 8n^{3/2}$ операций. В следующем разделе будет показано, что умножение на матрицу Z, по существу, есть то же самое, что вычисление *дискретного преобразования Фурье*. С помощью алгоритма FFT это преобразование может быть проделано за $O(N^2 \log N) = O(n \log n)$ операций.

(Используя введенный в разд. 6.3.3 язык кронекеровых произведений и, в частности, спектральное разложение матрицы $T_{N\times N}$, указанное в предложении 6.1, т. е.

$$T_{N\times N} = I \otimes T_N + T_N \otimes I = (Z \otimes Z)\cdot(I \otimes \Lambda + \Lambda \otimes I)\cdot(Z \otimes Z)^T,$$

можно следующим образом переписать формулу, обосновывающую алгоритм 6.13:

$$\begin{aligned}\operatorname{vec}(V) &= (T_{N\times N})^{-1}\operatorname{vec}(h^2F)\\ &= ((Z\otimes Z)\cdot(I\otimes\Lambda+\Lambda\otimes I)\cdot(Z\otimes Z)^T)^{-1}\cdot\operatorname{vec}(h^2F)\\ &= (Z\otimes Z)^{-T}\cdot(I\otimes\Lambda+\Lambda\otimes I)^{-1}\cdot(Z\otimes Z)^{-1}\cdot\operatorname{vec}(h^2F)\\ &= (Z\otimes Z)\cdot(I\otimes\Lambda+\Lambda\otimes I)^{-1}\cdot(Z^T\otimes Z^T)\cdot\operatorname{vec}(h^2F). \quad (6.47)\end{aligned}$$

Мы утверждаем, что выполнение указанных матрично-векторных умножений справа налево математически эквивалентно алгоритму 6.13 (см. вопрос 6.9). Это заодно показывает, как распространить алгоритм на уравнения Пуассона более высокой размерности.)

6.7.1. Дискретное преобразование Фурье

В этом подразделе строки и столбцы матриц будут нумероваться числами от 0 до $N-1$, а не от 1 до N, как обычно.

Определение 6.17. Дискретное преобразование Фурье (DFT) *N-вектора x есть вектор $y = \Phi x$, где Φ — матрица порядка N, определяемая следующим образом. Пусть $\omega = e^{-\frac{2\pi i}{N}} = \cos\frac{2\pi}{N} - i\sin\frac{2\pi}{N}$ есть примитивный корень N-й степени из единицы. Тогда $\varphi_{jk} = \omega^{jk}$.* Обратное дискретное преобразование Фурье (IDFT) *вектора y есть вектор $x = \Phi^{-1}y$.*

Лемма 6.9. *Матрица $\frac{1}{\sqrt{N}}\Phi$ симметрична и унитарна, поэтому* $\Phi^{-1} = \frac{1}{N}\Phi^* = \frac{1}{N}\overline{\Phi}$.

Доказательство. Ясно, что $\Phi = \Phi^T$, поэтому $\overline{\Phi} = \Phi^*$, и нужно лишь показать, что $\Phi\cdot\overline{\Phi} = N\cdot I$. Имеем $(\Phi\overline{\Phi})_{lj} = \sum_{k=0}^{N-1}\phi_{lk}\overline{\phi_{kj}} = \sum_{k=0}^{N-1}\omega^{lk}\overline{\omega^{kj}} = \sum_{k=0}^{N-1}\omega^{k(l-j)}$, поскольку $\overline{\omega} = \omega^{-1}$. Если $l = j$, то эта сумма, очевидно, равна n. Если же $l \neq j$, то, суммируя геометрическую прогрессию и учитывая, что $\omega^N = 1$, находим $\frac{1-\omega^{N(l-j)}}{1-\omega^{(l-j)}} = 0$. □

Итак, и DFT, и IDFT являются всего лишь матрично-векторными умножениями и могут быть бесхитростно реализованы за $2N^2$ флопов. Рассматриваемая операция называется дискретным преобразованием Фурье ввиду своей тесной математической связи с двумя другими типами анализа Фурье (см. таблицу).

преобразование Фурье	$F(\zeta) = \int_{-\infty}^{\infty} e^{-2\pi i\zeta x} f(x)\,dx$
и его обращение	$f(x) = \int_{-\infty}^{\infty} e^{+2\pi i\zeta x} F(\zeta)\,d\zeta$
ряд Фурье, где f периодична с периодом 1	$c_j = \int_0^1 e^{-2\pi ijx} f(x)\,dx$
и его обращение	$f(x) = \sum_{j=-\infty}^{\infty} e^{+2\pi ijx} c_j$
DFT	$y_j = (\Phi x)_j = \sum_{k=0}^{N-1} e^{-2\pi ijk/N} x_k$
и его обращение	$x_k = (\Phi^{-1}y)_k = \frac{1}{N}\sum_{j=0}^{N-1} e^{+2\pi ijk/N} y_j$

Мы конкретизируем эту тесную связь двумя способами. Во-первых, мы покажем, как решить модельную задачу с помощью DFT, а затем как решить уравнение Пуассона (6.1) посредством рядов Фурье. Этот пример послужит стимулом для отыскания быстрого метода умножения на матрицу Φ, потому что такой метод приведет к быстрому способу решения модельной задачи. Этот быстрый метод называется *быстрым преобразованием Фурье*, или FFT. Вместо $2N^2$ флопов в нем требуется гораздо меньшее число операций, приблизительно равное $\frac{3}{2}N\log_2 N$. Алгоритм FFT будет выведен с использованием второго математического свойства, общего для различных видов анализа Фурье, а именно преобразования свертки в умножение.

В алгоритме 6.13 показано, что для решения дискретного уравнения Пуассона $T_N V + V T_N = h^2 F$ относительно V требуется возможность вычислять произведения с участием $N \times N$-матрицы Z, где

$$z_{jk} = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(j+1)(k+1)}{N+1}.$$

(Напомним, что в этом разделе строки и столбцы нумеруются от 0 до $N-1$.) Рассмотрим теперь DFT-матрицу Φ порядка $2N+2$. Ее элемент (j,k) равен

$$\exp\left(\frac{-2\pi ijk}{2N+2}\right) = \exp\left(\frac{-\pi ijk}{N+1}\right) = \cos\frac{\pi jk}{N+1} - i \cdot \sin\frac{\pi jk}{N+1}.$$

Таким образом, $N \times N$-матрица Z с точностью до множителя $-\sqrt{\frac{2}{N+1}}$ совпадает с мнимой частью подматрицы, образованной строками и столбцами матрицы Φ, имеющими номера с 2 до $N+1$. Поэтому, если мы найдем способ эффективного умножения на Φ с помощью FFT, то сможем эффективно умножать и на Z. (Для достижения наибольшей эффективности описываемый ниже алгоритм FFT модифицируют таким образом, чтобы можно было умножать на Z непосредственно. Эта модификация называется быстрым *синус-преобразованием*. Но можно пользоваться и стандартным алгоритмом FFT.) Итак, для быстрого вычисления произведения ZF нужно применить операцию типа FFT к каждому столбцу матрицы F, а для произведения FZ аналогичная операция должна быть применена к каждой строке F. (В трехмерном случае символ V обозначал бы массив неизвестных размера $N \times N \times N$ и одна и та же операция применялась бы к каждому из $3N^2$ сечений, параллельных координатным осям.)

6.7.2. Решение непрерывной модельной задачи посредством рядов Фурье

Мы возвращаемся к нумерации строк и столбцов матриц числами от 1 до N.

В этом разделе будет показано, что алгоритм решения дискретной модельной задачи является естественным аналогом метода рядов Фурье для решения исходного дифференциального уравнения (6.1). Это будет сделано для одномерной модельной задачи.

Напомним, что уравнение Пуассона $-\frac{d^2v}{dx^2} = f(x)$ рассматривается на отрезке $[0,1]$ и имеет краевые условия $v(0) = v(1) = 0$. Чтобы решить эту задачу, разложим $v(x)$ в ряд Фурье: $v(x) = \sum_{j=1}^{\infty} \alpha_j \sin(j\pi x)$. (Отсутствие косинусов в этом разложении объясняется краевым условием $v(1) = 0$.) Подставляя $v(x)$ в уравнение Пуассона, находим

$$\sum_{j=1}^{\infty} \alpha_j (j^2\pi^2) \sin(j\pi x) = f(x).$$

Умножим обе части на $\sin(k\pi x)$, проинтегрируем в пределах от 0 до 1 и используем тот факт, что $\int_0^1 \sin(j\pi x)\sin(k\pi x)\,dx$ равен нулю при $j \neq k$ и $1/2$,

если $j = k$. В результате получим

$$\alpha_k = \frac{2}{k^2\pi^2}\int_0^1 \sin(k\pi x) f(x)\,dx$$

и, наконец,

$$v(x) = \sum_{j=1}^{\infty}\left(\frac{2}{j^2\pi^2}\int_0^1 \sin(j\pi y) f(y)\,dy\right)\sin(j\pi x). \tag{6.48}$$

Рассмотрим теперь дискретную модельную задачу $T_N v = h^2 f$. Поскольку $T_N = Z\Lambda Z^T$, то можем написать $v = T_N^{-1}h^2 f = Z\Lambda^{-1}Z^T h^2 f$, откуда

$$v_k = \sum_{j=1}^{N} z_{kj}\frac{h^2}{\lambda_j}(Z^T f)_j = \sum_{j=1}^{N}\sin\frac{\pi jk}{N+1}\left(\frac{h^2}{\lambda_j}\sqrt{\frac{2}{N+1}}(Z^T f)_j\right), \tag{6.49}$$

где

$$\begin{aligned}\sqrt{\frac{2}{N+1}}(Z^T f)_j &= \sqrt{\frac{2}{N+1}}\sum_{l=1}^{N}\sqrt{\frac{2}{N+1}}\sin\frac{\pi jl}{N+1} f_l \\ &= 2\sum_{l=1}^{N}\frac{1}{N+1}\sin\left(\frac{\pi jl}{N+1}\right) f_l \\ &\approx 2\int_0^1 \sin(\pi jy) f(y)\,dy\end{aligned}$$

В этих выкладках последняя сумма есть не что иное, как риманова сумма для интеграла. Вспомним еще, что $\frac{h^2}{\lambda_j} \approx \frac{1}{j^2\pi^2}$ для малых j. Мы видим, таким образом, что решение (6.49) дискретной задачи аппроксимирует решение (6.48) непрерывной задачи. При этом умножение на Z^T соответствует умножению на $\sin(j\pi x)$ с последующим интегрированием, а умножение на Z соответствует суммированию различных компонент Фурье.

6.7.3. Свертки

Свертка является важной операцией анализа Фурье, точное определение которой зависит от того, рассматриваем ли мы преобразование Фурье, ряд Фурье или DFT (см. таблицу).

Преобразование Фурье	$(f * g)(x) \equiv \int_{-\infty}^{\infty} f(x-y)g(y)\,dy$
Ряд Фурье	$(f * g)(x) \equiv \int_0^1 f(x-y)g(y)\,dy$
DFT	Если $a = [a_0, \ldots, a_{N-1}, 0, \ldots, 0]^T$ и $b = [b_0, \ldots, b_{N-1}, 0, \ldots, 0]^T$ — $2N$-векторы, то $a * b \equiv c = [c_0, \ldots, c_{2N-1}]^T$, где $c_k = \sum_{j=0}^{k} a_j b_{k-j}$

Чтобы проиллюстрировать использование дискретной свертки, рассмотрим операцию умножения многочленов. Пусть заданы многочлены $a(x) =$

$\sum_{k=0}^{N-1} a_k x^k$ и $\mathrm{b}(x) = \sum_{k=0}^{N-1} b_k x^k$ степени $N-1$. Тогда их произведением является многочлен $\mathrm{c}(x) \equiv \mathrm{a}(x) \cdot \mathrm{b}(x) = \sum_{k=0}^{2N-1} c_k x^k$, где коэффициенты $c_0, \ldots, c_{2N-1}$ определяются с помощью дискретной свертки.

Одной из задач, решаемых посредством преобразований Фурье, рядов Фурье или DFT, является превращение свертки в произведение. В случае преобразования Фурье имеем $\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$, т. е. образ Фурье свертки есть произведение образов Фурье. Для рядов Фурье справедливо соотношение $c_j(f * g) = c_j(f) \cdot c_j(g)$, т. е. коэффициенты Фурье свертки суть произведения коэффициентов Фурье. То же самое верно для дискретной свертки.

Теорема 6.10. *Пусть* $a=[a_0, \ldots, a_{N-1}, 0, \ldots, 0]^T$ *и* $b=[b_0, \ldots, b_{N-1}, 0, \ldots, 0]^T$ — *векторы размерности* $2N$ *и пусть* $c = a * b = [c_0, \ldots, c_{2N-1}]^T$. *Тогда* $(\Phi c)_k = (\Phi a)_k (\Phi b)_k$.

Доказательство. Если $a' = \Phi a$, то $a'_k = \sum_{j=0}^{N-1} a_j \omega^{kj}$, т. е. a'_k равно значению многочлена $\mathrm{a}(x) \equiv \sum_{j=0}^{N-1} a_j x^j$ в точке $x = \omega^k$. Точно так же, из $b' = \Phi b$ следует, что $b'_k = \sum_{j=0}^{N-1} b_j \omega^{kj} = \mathrm{b}(\omega^k)$, а из $c' = \Phi c$ следует, что $c'_k = \sum_{j=0}^{2N-1} c_j \omega^{kj} = \mathrm{c}(\omega^k)$. Поэтому

$$a'_k \cdot b'_k = \mathrm{a}(\omega^k) \cdot \mathrm{b}(\omega^k) = \mathrm{c}(\omega^k) = c'_k,$$

что и требовалось. □

Иными словами, DFT эквивалентно вычислению значений многочлена в точках $\omega^0, \ldots, \omega^{N-1}$. Обратно, IDFT эквивалентно интерполированию многочлена, т. е. вычислению его коэффициентов по значениям многочлена в точках $\omega^0, \ldots, \omega^{N-1}$.

6.7.4. Вычисление быстрого преобразования Фурье

Мы выведем алгоритм FFT, пользуясь только что полученной интерпретацией DFT как операции вычисления значений многочлена. Наша цель — вычислить значения многочлена $\mathrm{a}(x) = \sum_{k=0}^{N-1} a_k x^k$ в точках $x = \omega^j$ для $0 \le j \le N-1$. Для простоты, предположим, что $N = 2^m$. Теперь имеем

$$\begin{aligned} \mathrm{a}(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1} \\ &= (a_0 + a_2 x^2 + a_4 x^4 + \cdots) + x(a_1 + a_3 x^2 + a_5 x^4 + \cdots) \\ &= \mathrm{a}_{even}(x^2) + x \cdot \mathrm{a}_{odd}(x^2). \end{aligned}$$

Таким образом, нужно вычислить значения двух многочленов a_{even} и a_{odd} степени $\frac{N}{2} - 1$ в точках $(\omega^j)^2$, $0 \le j \le N-1$. В действительности, можно ограничиться $\frac{N}{2}$ точками ω^{2j}, отвечающими значениям $0 \le j \le \frac{N}{2} - 1$, так как $\omega^{2j} = \omega^{2(j+\frac{N}{2})}$.

Итак, вычисление многочлена степени $N-1 = 2^m - 1$ во всех N корнях N-й степени из единицы равносильно вычислению двух многочленов степени $\frac{N}{2} - 1$ во всех $\frac{N}{2}$ корнях степени $\frac{N}{2}$ из единицы с последующим комбинированием результатов, требующим N умножений и сложений. Эту связь можно использовать рекурсивно.

Алгоритм 6.14. FFT *(рекурсивная версия):*

function FFT*(a,N)*
  if $N = 1$
    возвратить a
  else
    $a'_{even} = \mathrm{FFT}(a_{even}, N/2)$
    $a'_{odd} = \mathrm{FFT}(a_{odd}, N/2)$
    $\omega = e^{-2\pi i/N}$
    $w = [\omega^0, \ldots, \omega^{N/2-1}]$
    возвратить $a' = [a'_{even} + w. * a'_{odd}, a'_{even} - w. * a'_{odd}]$
  endif

Здесь символ . обозначает покомпонентное умножение массивов (как в Matlab'e) и используется соотношение* $\omega^{j+N/2} = -\omega^j$.

Обозначим сложность этого алгоритма через $C(N)$. Мы видим, что функция $C(N)$ удовлетворяет рекуррентному соотношению $C(N) = 2C(N/2)+3N/2$ (при этом предполагается, что степени ω вычислены и записаны в таблицы заранее). Решение этой рекурсии дает выкладка

$$\begin{aligned} C(N) &= 2C\left(\frac{N}{2}\right) + \frac{3N}{2} = 4C\left(\frac{N}{4}\right) + 2 \cdot \frac{3N}{2} = 8C\left(\frac{N}{8}\right) + 3 \cdot \frac{3N}{2} \\ &= \cdots \\ &= \log_2 N \cdot \frac{3N}{2}. \end{aligned}$$

Применение FFT к каждому столбцу (или каждой строке) матрицы порядка N обходится, таким образом, в $\log_2 N \cdot \frac{3N^2}{2}$ операций. Этот анализ сложности объясняет элемент табл. 6.1, относящийся к FFT.

В практических реализациях FFT для достижения наибольшей эффективности используются простые вложенные циклы, а не рекурсии. Кроме того, в этих реализациях компоненты результата иногда выдаются в *двоично-инверсном* порядке. Это означает, что в выходном векторе $y = \Phi x$ вместо обычного порядка компонент y_0, y_1, ..., y_{N-1} принята последовательность, отвечающая обращению порядка битов в двоичном представлении индексов. Например, при $N = 8$ индексы изменяются от $0 = 000_2$ до $7 = 111_2$. В таблице показаны обычный порядок индексов и двоично-инверсный порядок.

обычный порядок индексов	двоично-инверсный порядок
$0 = 000_2$	$0 = 000_2$
$1 = 001_2$	$4 = 100_2$
$2 = 010_2$	$2 = 010_2$
$3 = 011_2$	$6 = 110_2$
$4 = 100_2$	$1 = 001_2$
$5 = 101_2$	$5 = 101_2$
$6 = 110_2$	$3 = 011_2$
$7 = 111_2$	$7 = 111_2$

В обратном FFT происходит возврат от модифицированного порядка к исходному. Следовательно, эти алгоритмы могут быть применены к решению

модельной задачи; нужно лишь, чтобы деление на собственные значения производилось с учетом двоично-инверсного порядка индексов. (Заметим, что в Matlab'е результаты всегда выдаются в обычном порядке возрастания индексов.)

6.8. Блочная циклическая редукция

Блочная циклическая редукция — это еще один быстрый метод (т. е. метод со сложностью $O(N^2 \log_2 N)$) для решения модельной задачи, однако область его применимости несколько шире, чем метода, основанного на FFT. Наиболее быстрое решение модельной задачи на векторных компьютерах часто достигается комбинированием блочной циклической редукции и FFT.

Вначале будет описана простая, но численно неустойчивая версия алгоритма; затем мы кратко обсудим, как стабилизировать ее. Запишем модельную задачу в виде

$$\begin{bmatrix} A & -I & & \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & A \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

и предположим, что порядок N матрицы $A = T_N + 2I_N$ нечетен. Отметим еще, что x_i и b_i суть N-векторы.

Рассматривая три подряд идущие блочные строки, применим блочный метод Гаусса:

$$\begin{array}{rl} + & [\ -x_{j-2} + Ax_{j-1} - x_j = b_{j-1}\ ], \\ +A* & [\ -x_{j-1} + Ax_j - x_{j+1} = b_j\ ], \\ + & [\ -x_j + Ax_{j+1} - x_{j+2} = b_{j+1}\ ]. \end{array}$$

Это приведет к исключению векторов x_{j-1} и x_{j+1}:

$$-x_{j-2} + (A^2 - 2I)x_j - x_{j+2} = b_{j-1} + Ab_j + b_{j+1}.$$

Если проделать это для каждой группы из трех подряд идущих блочных строк, то получим две системы уравнений. Одна из них соответствует векторам x_j с четными индексами j:

$$\begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & -I & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & B \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 + Ab_2 + b_3 \\ b_3 + Ab_4 + b_5 \\ \vdots \\ b_{N-2} + Ab_{N-1} + b_N \end{bmatrix}. \qquad (6.50)$$

Здесь $B = A^2 - 2I$. Другая система соответствует векторам x_j с нечетными индексами:

$$\begin{bmatrix} A & & & \\ & A & & \\ & & \ddots & \\ & & & A \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 + x_2 \\ b_3 + x_2 + x_4 \\ \vdots \\ b_N + x_{N-1} \end{bmatrix}.$$

Она может быть решена после того, как из уравнения (6.50) определены x_j с четными индексами.

Заметим, что уравнение (6.50) имеет ту же форму, что и исходная задача, поэтому процесс может быть повторен рекурсивно. Так, на следующем шаге мы получили бы системы

$$\begin{bmatrix} C & -I & & & \\ -I & C & -I & & \\ & -I & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & C \end{bmatrix} \begin{bmatrix} x_4 \\ x_8 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}, \quad \text{где } C = B^2 - 2I,$$

и

$$\begin{bmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix} \begin{bmatrix} x_2 \\ x_6 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}.$$

Будем продолжать процесс до тех пор, пока не останется только одно (блочное) уравнение. Оно будет решено иным методом.

Мы формализуем этот алгоритм следующим образом: предположим, что $N = N_0 = 2^{k+1} - 1$, $N_r = 2^{k+1-r} - 1$. Положим $A^{(0)} = A$ и $b_j^{(0)} = b_j$ для $j = 1, \ldots, N$.

Алгоритм 6.15. *Блочная циклическая редукция:*

1) *Приведение:*

for $r = 0$ *to* $k - 1$
  $A^{(r+1)} = (A^{(r)})^2 - 2I$
  for $j = 1$ *to* N_{r+1}
    $b_j^{(r+1)} = b_{2j-1}^{(r)} + A^{(r)} b_{2j}^{(r)} + b_{2j+1}^{(r)}$
  end for
end for

Комментарий: после r шагов задача приводится к виду:

$$\begin{bmatrix} A^{(r)} & -I & & \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & A^{(r)} \end{bmatrix} \begin{bmatrix} x_1^{(r)} \\ \vdots \\ x_{N_r}^{(r)} \end{bmatrix} = \begin{bmatrix} b_1^{(r)} \\ \vdots \\ b_{N_r}^{(r)} \end{bmatrix}.$$

2) *Система $A^{(k)} x^{(k)} = b^{(k)}$ решается другим методом.*

3) *Обратная подстановка:*

for $r = k - 1, \ldots, 0$

for $j = 1$ *to* N_{r+1}
$x_{2j}^{(r)} = x_j^{(r+1)}$
end for
for $j = 1$ *to* N_r *step* 2
решить систему $A^{(r)} x_j^{(r)} = b_j^{(r)} + x_{j-1}^{(r)} + x_{j+1}^{(r)}$
относительно $x_j^{(r)}$
(*полагаем* $x_0^{(r)} = x_{N_r+1}^{(r)} \equiv 0$)
end for
end for

Вектор $x = x^{(0)}$ *есть искомый результат.*

У этого простого метода есть два недостатка:

1. Он численно неустойчив, так как матрицы $A^{(r)}$ быстро растут: $\|A^{(r)}\| \sim \|A^{(r-1)}\|^2 \approx 4^{2^r}$, поэтому при вычислении вектора $b_j^{(r+1)}$ значения векторов $b_{2j-1}^{(r)}$ и $b_{2j+1}^{(r)}$ не будут из-за округлений отражены в результате.
2. Если A — трехдиагональная матрица, то $A^{(r)}$ имеет ленту ширины $2^r + 1$, следовательно, быстро заполняется. Поэтому умножение на $A^{(r)}$ и решение систем с этой матрицей становятся дорогостоящими операциями.

Вот как можно исправить второй недостаток. Заметим, что $A^{(r)}$ представляет собой многочлен $p_r(A)$ степени 2^r от матрицы A, где

$$p_0(A) = A \quad \text{и} \quad p_{r+1}(A) = (p_r(A))^2 - 2I.$$

Лемма 6.10. *Пусть* $t = 2\cos\theta$. *Тогда* $p_r(t) = p_r(2\cos\theta) = 2\cos(2^r\theta)$.

Доказательство. Утверждение леммы вытекает из простого тригонометрического тождества. □

Заметим, что $p_r(t) = 2\cos(2^r \arccos(\frac{t}{2})) = 2T_{2^r}(\frac{t}{2})$, где $T_{2^r}(\cdot)$ есть многочлен Чебышева (см. разд. 6.5.6).

Лемма 6.11. *Имеет место представление* $p_r(t) = \prod_{j=1}^{2^r}(t - t_j)$, *где* $t_j = 2\cos(\pi\frac{2j-1}{2^r})$.

Доказательство. Нули многочленов Чебышева указаны в лемме 6.7. □

Таким образом, $A^{(r)} = \prod_{j=1}^{2^r}(A - 2\cos(\pi\frac{2j-1}{2^r})I)$, поэтому решение системы $A^{(r)}z = c$ эквивалентно решению 2^r систем с трехдиагональными матрицами коэффициентов $A - 2\cos(\pi\frac{2j-1}{2^r})I$. Решение каждой из них гауссовым исключением или алгоритмом Холесского обходится в $O(N)$ операций.

Чтобы добиться численной устойчивости, в алгоритм должны быть внесены дополнительные изменения. Окончательная версия алгоритма, принадлежащая Бунеману (Buneman), описана в [47, 46].

Проанализируем сложность простого алгоритма; такова же будет сложность устойчивой версии. Умножение на трехдиагональную матрицу или решение трехдиагональной системы порядка N стоит $O(N)$ флопов. Поскольку

$A^{(r)}$ есть произведение 2^r трехдиагональных матриц, умножение на $A^{(r)}$ или решение системы с этой матрицей стоит $O(2^r N)$ флопов. Поэтому стоимость внутреннего цикла на шаге 1) алгоритма составляет $\frac{N}{2^{r+1}} \cdot O(2^r N) = O(N^2)$ флопов, так как на нем пересчитывается $N_{r+1} \approx \frac{N}{2^{r+1}}$ векторов $b_j^{(r+1)}$. Матрица $A^{(r+1)}$ не вычисляется в явном виде. Поскольку цикл на шаге 1) выполняется $k \approx \log_2 N$ раз, полная стоимость шага 1) равна $O(N^2 \log_2 N)$. Аналогично подсчитывается, что шаг 2) стоит $O(2^k N) = O(N^2)$ флопов, а шаг 3) — $O(N^2 \log_2 N)$ флопов, что дает для алгоритма в целом стоимость $O(N^2 \log_2 N)$ флопов. Это объясняет элемент табл. 6.1, относящийся к блочной циклической редукции.

Алгоритм обобщается на блочно-трехдиагональные матрицы, в которых все диагональные блоки равны симметричной матрице A, а все блоки на диагоналях, соседствующих с главной, равны симметричной матрице F, перестановочной с A (т. е. $FA = AF$). См. по этому поводу вопрос 6.10. Такого рода структура часто встречается при решении линейных систем, возникающих при дискретизации дифференциальных уравнений, таких, как уравнение Пуассона.

6.9. Многосеточные методы

Многосеточные методы были предложены в контексте решения дифференциальных уравнений с частными производными типа уравнения Пуассона, однако они пригодны для более широкого класса задач. В отличие от обсуждавшихся до сих пор итерационных схем, скорость сходимости многосеточного метода, будучи *независимой* от размера N сетки, не замедляется для больших систем. Как следствие, задачи с n неизвестными оказывается возможным решать за время $O(n)$, т. е. затрачивая фиксированную работу на каждое неизвестное. С точностью до (умеренной) константы, спрятанной в символе $O(\cdot)$, этот результат является оптимальным.

Поясним, почему обсуждавшиеся ранее итерационные алгоритмы *не* могут быть оптимальны для модельной задачи. Это утверждение, в действительности, верно в отношении *любого* итерационного алгоритма, вычисляющего приближение x_{m+1} путем усреднения значений x_m и правой части b в соседних узлах сетки. Под это описание подходят методы Якоби, Гаусса—Зейделя, SOR(ω), SSOR с чебышевским ускорением (для трех последних методов имеется в виду шахматное упорядочение) и всякий метод крыловского подпространства, основанный на матрично-векторных умножениях с участием матрицы $T_{N\times N}$ (последнее верно потому, что умножение текущего приближения на матрицу $T_{N\times N}$ равносильно усреднению по соседним узлам сетки). Предположим, что на сетке 31×31 задана правая часть b с единственным ненулевым элементом; она показана на левой верхней картинке рис. 6.9. Точное решение x показано на правой верхней картинке того же рисунка; отметим, что оно нигде не обращается в нуль и убывает по мере удаления от центра. Левая нижняя картинка рис. 6.9 показывает приближенное решение $x_{J,5}$, вычисленное посредством 5 шагов метода Якоби, исходя из нулевого начального вектора. Обратим внимание на то, что $x_{J,5}$ равно нулю на расстоянии от центра, превышающем 5 узлов сетки. Причина в том, что усреднение по соседним узлам способно «передавать информацию» лишь на один узел за итерацию, а у начального

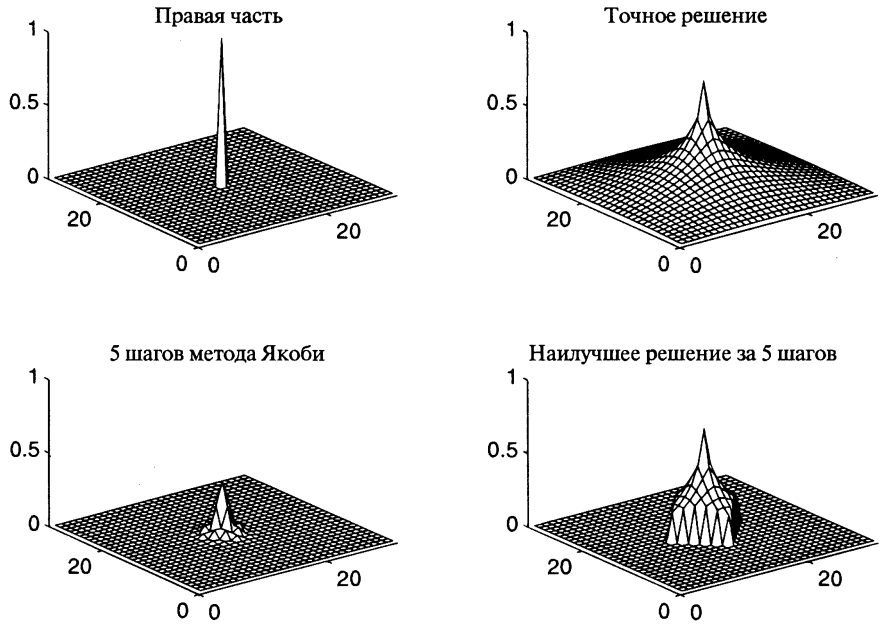


Рис. 6.9. Ограничения, накладываемые усреднением по ближайшим узлам сетки.

приближения была лишь одна ненулевая компонента в центре сетки. Более общо, после k итераций отличны от нуля могут быть лишь значения в узлах сетки, удаленных от центра не более чем на k. На правой нижней картинке показано наилучшее решение $x_{Best,5}$, которое можно получить за 5 шагов любым методом «ближайшего соседа». Это решение совпадает с x на расстоянии от центра, не превышающем 5 узлов сетки, и равно нулю на бóльшем расстоянии. Из рисунка видно, что погрешность $x - x_{Best,5}$ равна значению x на расстоянии шести узлов сетки от центра. Это все еще большая погрешность. Формализуя это рассуждение, можно показать, что какой бы алгоритм «ближайшего соседа» ни использовался, на сетке $n \times n$ потребуется по меньшей мере $O(\log n)$ шагов, чтобы уменьшить погрешность в фиксированное число раз, большее единицы. Чтобы получить лучший результат, чем $O(\log n)$ шагов (и, соответственно, сложность $O(n \log n)$), нужно научиться «распространять информацию» более чем на один узел сетки за итерацию. В многосеточных методах эта цель достигается путем обмена информацией с ближайшими узлами более грубых сеток; такие узлы могут оказаться гораздо дальше от данного, чем ближайшие узлы мелкой сетки.

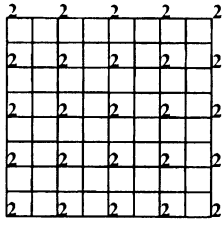
В многосеточных методах грубые сетки используются для реализации стратегии *«разделяй-и-властвуй»* в двух родственных отношениях. Во-первых, начальное решение для сетки $N \times N$ строится посредством аппроксимирующей сетки $(N/2) \times (N/2)$, получаемой удержанием каждого второго узла исходной сетки $N \times N$. Более грубая сетка $(N/2) \times (N/2)$, в свою очередь, аппроксимируется сеткой $(N/4) \times (N/4)$, и так далее. Второй способ использования стратегии

«разделяй-и-властвуй» относится к *частотной области*. Здесь погрешность нужно представлять себе как сумму собственных векторов, или синусоид с различными частотами. Тогда интуитивное описание будет таким: работа, выполняемая на конкретной сетке, уменьшает половину частотных компонент погрешности, для которых уменьшение не было достигнуто на более грубых сетках. В частности, работа, производимая для данной сетки (т. е. усреднение приближенного решения в каждом узле сетки с использованием соседних узлов — некоторый вариант метода Якоби), делает приближенное решение глаже, что эквивалентно подавлению высокочастотных компонент погрешности. Ниже это описание будет развернуто в более подробное.

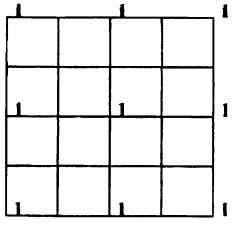
6.9.1. Очерк многосеточного метода на примере двумерного уравнения Пуассона

Мы дадим вначале общую формулировку нашего алгоритма, а затем объясним детали. Как и в случае блочной циклической редукции (разд. 6.8), более удобно рассматривать сетку неизвестных размера $(2^k-1)\times(2^k-1)$, а не сетку $2^k\times 2^k$, предпочтительную для FFT (разд. 6.7). Для лучшего понимания и более удобной реализации разумно добавить еще граничные узлы, где решение принимает известное значение 0, тогда получится сетка размера $(2^k+1)\times(2^k+1)$, показанная на рис. 6.10 и 6.13. Положим еще $N_k = 2^k - 1$.

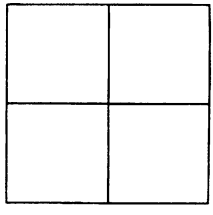
Обозначим через $P^{(i)}$ задачу решения дискретного уравнения Пуассона на сетке размера $(2^i+1)\times(2^i+1)$ с $(2^i-1)^2$ неизвестными, иначе говоря, на сетке размера $(N_i+2)\times(N_i+2)$ с N_i^2 неизвестными. Задача $P^{(i)}$ характеризуется правой частью $b^{(i)}$ и (неявно) размером сетки 2^i-1 и матрицей коэффициентов $T^{(i)} \equiv T_{N_i\times N_i}$. Приближенное решение задачи $P^{(i)}$ обозначим через $x^{(i)}$. Таким образом, $b^{(i)}$ и $x^{(i)}$ суть массивы размера $(2^i-1)\times(2^i-1)$, составленные из значений в узлах сетки. (Нулевые граничные значения присутствуют неявно.) Будет построена последовательность связанных задач $P^{(i)}$, $P^{(i-1)}$, $P^{(i-2)}$, ..., $P^{(1)}$ на



$P^{(3)}$: сетка узлов 9 x 9
сетка неизвестных 7 x 7
узлы, помеченные цифрой 2, составляют следующую более грубую сетку



$P^{(2)}$: сетка узлов 5 x 5
сетка неизвестных 3 x 3
узлы, помеченные цифрой 1, составляют следующую более грубую сетку



$P^{(1)}$: сетка узлов 3 x 3
сетка неизвестных 1 x 1

Рис. 6.10. Последовательность сеток, используемых в двумерном многосеточном методе.

все более грубых сетках; при этом решение задачи $P^{(i-1)}$ будет хорошим приближением погрешности решения задачи $P^{(i)}$.

Для того чтобы объяснить, как работает многосеточный метод, нам потребуется ввести операторы, которые либо улучшают приближенное решение для данной сетки, либо преобразуют данную задачу в родственную задачу для другой сетки:

- *Оператор сглаживания* S, будучи применен к задаче $P^{(i)}$ с приближенным решением $x^{(i)}$, вычисляет уточненное приближение $x^{(i)}$:

$$\text{улучшенное } x^{(i)} = S(b^{(i)}, x^{(i)}). \tag{6.51}$$

  Улучшение состоит в подавлении «высокочастотных компонент» погрешности. Ниже будет объяснено, что это значит. Уточнение решения достигается усреднением значения в каждой точке сетки по его ближайшим соседям; оно представляет собой некоторый вариант метода Якоби.
- *Оператор сужения* R отображает правую часть $b^{(i)}$ задачи $P^{(i)}$ в ее приближение $b^{(i-1)}$ на более грубой сетке:

$$b^{(i-1)} = R(b^{(i)}). \tag{6.52}$$

  Реализация этого оператора также сводится к взвешенному усреднению по ближайшим соседним узлам сетки.
- *Оператор интерполяции* In преобразует приближенное решение $x^{(i-1)}$ задачи $P^{(i-1)}$ в приближенное решение $x^{(i)}$ задачи $P^{(i)}$, поставленной на соседней, более мелкой сетке:

$$x^{(i)} = In(x^{(i-1)}). \tag{6.53}$$

  И здесь реализация оператора требует лишь взвешенного усреднения по ближайшим соседним узлам сетки.

Итак, все три оператора выполняются путем замены текущих значений в каждом узле сетки некоторыми взвешенными средними по ближайшим соседним узлам. Поэтому каждый оператор производит работу $O(1)$ на одно неизвестное или работу $O(n)$ для всех n неизвестных. Это объясняет низкую стоимость алгоритма в целом.

Многосеточный V-цикл

Сказанного выше достаточно для того, чтобы сформулировать основной алгоритм, так называемый *многосеточный* V-*цикл* (MGV).

Алгоритм 6.16. MGV *(строки пронумерованы для последующих ссылок):*

function MGV($b^{(i)}, x^{(i)}$) ... *заменить приближенное*
... *решение* $x^{(i)}$ *задачи* $P^{(i)}$
... *уточненным приближением*
if $i = 1$ ... *случай одного неизвестного*
вычислить точное решение $x^{(1)}$ *задачи* $P^{(1)}$
возвратить $x^{(1)}$
else
1) $x^{(i)} = S(b^{(i)}, x^{(i)})$ ... *найти улучшенное*

приближение

2) $r^{(i)} = T^{(i)} \cdot x^{(i)} - b^{(i)}$ ... *вычислить невязку*

3) $d^{(i)} = In(MGV(4 \cdot R(r^{(i)}), 0))$ ... *использовать рекурсию по ... более грубым сеткам*

4) $x^{(i)} = x^{(i)} - d^{(i)}$ ... *уточнить решение на мелкой сетке*

5) $x^{(i)} = S(b^{(i)}, x^{(i)})$ ... *снова уточнить приближенное решение*

возвратить $x^{(i)}$

endif

Если прибегнуть к словесному описанию, то алгоритм делает следующее:

1. Он начинает работу с задачей на мелкой сетке, характеризуемой векторами $b^{(i)}$ и $x^{(i)}$.
2. Он улучшает текущее приближение, подавляя высокочастотную погрешность: $x^{(i)} = S(b^{(i)}, x^{(i)})$.
3. Он вычисляет невязку $r^{(i)}$ приближенного решения $x^{(i)}$.
4. Он аппроксимирует невязку $r^{(i)}$ для мелкой сетки с помощью соседней, более грубой сетки: $R(r^{(i)})$.
5. Задача на более грубой сетке решается рекурсивно при нулевом начальном приближении: $\mathrm{MGV}(4 \cdot R(r^{(i)}), 0)$. Коэффициент 4 обязан своим происхождением множителю h^2 в правой части уравнения Пуассона; этот множитель изменяется ровно в 4 раза при переходе от мелкой сетки к грубой.
6. Алгоритм интерполирует решение, полученное на грубой сетке, к более мелкой сетке: $d^{(i)} = In(MGV(4 \cdot R(r^{(i)}), 0))$.
7. Вычитает поправку, вычисленную с помощью грубой сетки, из приближенного решения на мелкой сетке: $x^{(i)} = x^{(i)} - d^{(i)}$.
8. Снова уточняет текущее приближение: $x^{(i)} = S(b^{(i)}, x^{(i)})$.

Дадим краткое обоснование алгоритма (детали будут восполнены позже). Сделаем (индуктивное) предположение, что $d^{(i)}$ есть *точное* решение уравнения

$$T^{(i)} \cdot d^{(i)} = r^{(i)} = T^{(i)} \cdot x^{(i)} - b^{(i)}.$$

Перегруппировывая его члены, находим

$$T^{(i)} \cdot (x^{(i)} - d^{(i)}) = b^{(i)}.$$

Таким образом, $x^{(i)} - d^{(i)}$ есть искомое решение.

Название «V-цикл» имеет следующее объяснение. Изобразим алгоритм схематически на плоскости переменных (номер сетки i, время), помещая на график точку при каждом рекурсивном обращении к MGV. Тогда получится нечто вроде рис. 6.11, где работа алгоритма начинается с обращения к $\mathrm{MGV}(b^{(5)}, x^{(5)})$ (см. точку в левом верхнем углу). Затем происходят обращения к MGV для сетки 4, сетки 3, и так далее, пока не дойдем до самой грубой сетки 1, после чего сетки используются в обратном направлении с возвращением к сетке 5.

Исходя только из того, что операторы S, R и In алгоритма MGV заменяют текущие сеточные значения некоторыми взвешенными средними по со-

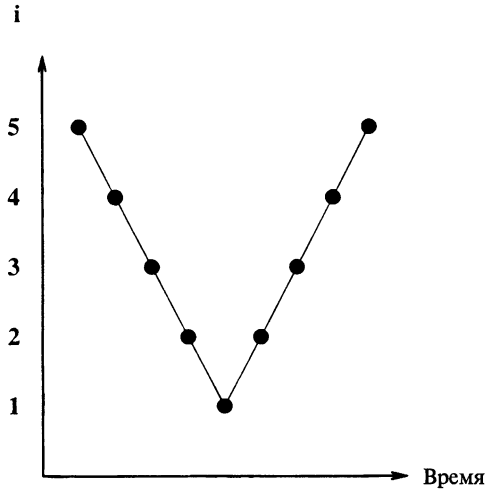


Рис. 6.11. MGV.

седним узлам, мы уже можем провести анализ сложности алгоритма в смысле символа $O(\cdot)$. Поскольку каждый из операторов производит фиксированную работу в каждом узле сетки, общее количество выполняемой им работы пропорционально числу узлов. Таким образом, если точка на букве V в нашем V-цикле соответствует сетке с номером i, то с этой точкой связана работа $O((2^i-1)^2) = O(4^i)$ операций. Если самой мелкой сетке соответствуют уровень k и $n = O(4^k)$ неизвестных, то полная стоимость цикла выразится геометрической прогрессией

$$\sum_{i=1}^{k} O(4^i) = O(4^k) = O(n).$$

Полный многосеточный метод

Наш окончательный алгоритм, называемый *полным многосеточным методом* (FMG), использует описанный выше алгоритм MGV как строительный блок.

Алгоритм 6.17. *FMG:*

function FMG($b^{(k)}, x^{(k)}$) ... *вычисляет приближенное решение* $x^{(k)}$
... *хорошей точности для задачи* $P^{(k)}$
решить задачу $P^{(1)}$ *точно, определив* $x^{(1)}$
for $i = 2$ *to* k
$x^{(i)} = \mathrm{MGV}(b^{(i)}, In(x^{(i-1)}))$
end for

В словесном описании алгоритм делает следующее:

1. Решает самую простую задачу $P^{(1)}$ точно.

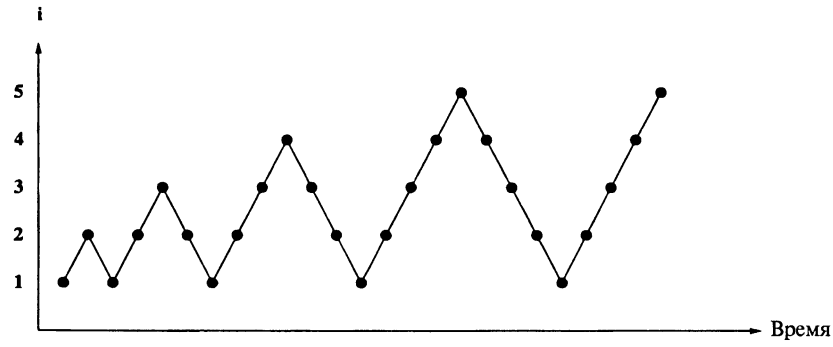


Рис. 6.12. FMG-цикл.

2. При имеющемся решении $x^{(i-1)}$ более грубой задачи $P^{(i-1)}$ отображает $x^{(i-1)}$ в начальное приближение $x^{(i)}$ задачи $P^{(i)}$ на более мелкой сетке: $In(x^{(i-1)})$.
3. Решает задачу на более мелкой сетке с указанным начальным приближением, пользуясь алгоритмом MGV: $\mathrm{MGV}(b^{(i)}, In(x^{(i-1)}))$.

Теперь можно провести анализ сложности в смысле символа $O(\cdot)$ для алгоритма FMG в целом. Рис. 6.12 дает изображение алгоритма на плоскости переменных (номер сетки i, время). Каждому обращению к MGV во внутреннем цикле алгоритма FMG соответствует один символ «V» на рисунке. Как и прежде, стоимость такого символа, начинающегося с уровня i, составляет $O(4^i)$. Поэтому общая стоимость алгоритма снова выражается геометрической прогрессией

$$\sum_{i=1}^{k} O(4^i) = O(4^k) = O(n).$$

Этот результат оптимален, поскольку на каждое из n переменных приходится одна и та же, не зависящая от n работа. Он объясняет элемент табл. 6.1, соответствующий многосеточному методу.

Matlab-реализацию многосеточного метода (как для одномерной, так и для двумерной задачи) можно найти в HOMEPAGE/Matlab/MG_README.html.

6.9.2. Подробное описание многосеточного метода для одномерного уравнения Пуассона

Теперь мы детально разберем конструкцию различных операторов S, R и In, составляющих многосеточный алгоритм, и дадим доказательство его сходимости. Мы сделаем это для одномерного уравнения Пуассона, поскольку здесь будут схвачены все особенности поведения алгоритма, но записи будут проще. В частности, вместо последовательности вложенных двумерных задач теперь можно рассматривать показанные на рис. 6.13 аналогичные одномерные задачи.

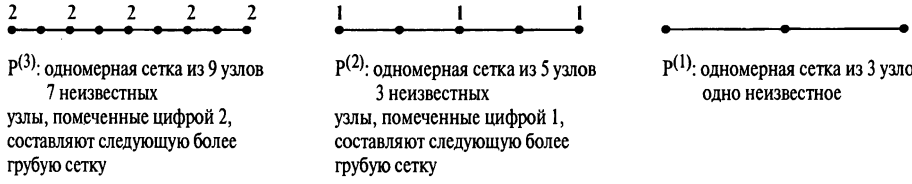


Рис. 6.13. Последовательность сеток, используемых в одномерном многосеточном методе.

Как и выше, будем обозначать через $P^{(i)}$ задачу, решаемую на сетке i, а именно $T^{(i)} \cdot x^{(i)} = b^{(i)}$, где $T^{(i)} \equiv T_{N_i}$, а $N_i = 2^i - 1$, как и раньше. Начнем с описания оператора сглаживания S, который представляет собой вариант *взвешенного метода Якоби*.

Оператор сглаживания в одномерном случае

В этом разделе для упрощения записей верхние индексы в символах $T^{(i)}$, $x^{(i)}$ и $b^{(i)}$ будут опущены. Пусть $T = Z\Lambda Z^T$ — спектральное разложение матрицы T, описанное в лемме 6.1. Стандартный метод Якоби для решения системы $Tx = b$ задается формулой $x_{m+1} = Rx_m + c$, где $R = I - T/2$ и $c = b/2$. Рассмотрим *взвешенный метод Якоби* $x_{m+1} = R_w x_m + c_w$, где $R_w = I - wT/2$ и $c_w = wb/2$; значение $w = 1$ соответствует обычному методу Якоби. Заметим, что спектральное разложение матрицы R_w имеет вид $R_w = Z(I - w\Lambda/2)Z^T$. Как обычно, собственные значения этой матрицы определяют скорость сходимости взвешенного метода Якоби. Пусть $e_m = x_m - x$ есть погрешность на m-й итерации метода, тогда

$$\begin{aligned} e_m &= R_w e_{m-1} \\ &= R_w^m e_0 \\ &= (Z(I - w\Lambda/2)Z^T)^m e_0 \\ &= Z(I - w\Lambda/2)^m Z^T e_0, \end{aligned}$$

откуда

$$Z^T e_m = (I - w\Lambda/2)^m Z^T e_0, \quad \text{или} \quad (Z^T e_m)_j = (I - w\Lambda/2)^m_{jj} (Z^T e_0)_j.$$

Поскольку $e_m = Z(Z^T e_m)$ есть сумма столбцов матрицы Z, где числа $(Z^T e_m)_j$ выступают как веса, иначе говоря, e_m есть сумма синусоид различной частоты (см. рис. 6.2), то число $(Z^T e_m)_j$ называется j-й *частотной компонентой* погрешности e_m. Собственное значение $\lambda_j(R_w) = 1 - w\lambda_j/2$ определяет, с какой скоростью стремится к нулю эта компонента. На рис. 6.14 показан график $\lambda_j(R_w)$ для $N = 99$ и различных значений веса w.

При $w = \frac{2}{3}$ и $j > \frac{N}{2}$, т. е. для правой половины частот λ_j, имеем $|\lambda_j(R_w)| \le \frac{1}{3}$. Это означает, что, независимо от N, соответствующие компоненты погрешности $(Z^T e_m)_j$ умножаются на каждой итерации на $\frac{1}{3}$ или меньшее число. Как показывает рис. 6.15, низкочастотные компоненты погрешности не

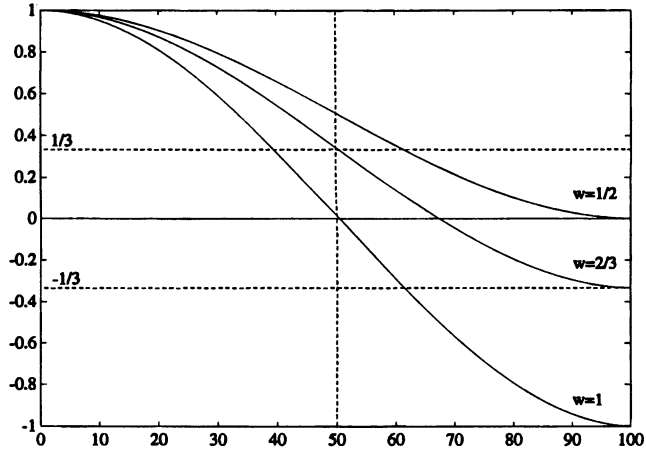


Рис. 6.14. График спектра матрицы R_w для $N = 99$ и $w = 1$ (метод Якоби), $w = 1/2$ и $w = 2/3$.

слишком убывают. Итак, взвешенный метод Якоби для $w = \frac{2}{3}$ эффективен в уменьшении высокочастотной погрешности.

Поэтому наш оператор сглаживания для уравнения (6.51) соответствует одному шагу взвешенного метода Якоби для $w = \frac{2}{3}$:

$$S(b, x) = R_{2/3} \cdot x + b/3. \tag{6.54}$$

Будем писать $R_{2/3}^{(i)}$ вместо $R_{2/3}$, если нужно указать сетку i, на которой действует $R_{2/3}$.

На рис. 6.15 показан результат выполнения двух шагов типа S для $i = 6$; эта задача имеет $2^i - 1 = 63$ неизвестных. Рисунок состоит из трех групп графиков. Первая группа показывает начальное приближение x_m и погрешность e_m после соответствующего применения оператора S. Точное решение есть синусоида, образованная точками на левом графике каждой группы. На том же графике приближенное решение показано сплошной линией. На средних графиках изображена погрешность, причем под графиками приведено значение ее 2-нормы. На правых графиках показаны частотные компоненты погрешности, составляющие вектор $Z^T e_m$. Можно видеть на этих графиках, как применение оператора S подавляет правую половину частотных компонент. Это же явление обнаруживается в средних и левых графиках тем, что приближенное решение становится глаже. Причина в том, что высокочастотная погрешность проявляет себя как «биения», а низкочастотная погрешность выглядит как «гладкая» функция. Поначалу норма погрешности убывает быстро — с 1.65 до 1.055. Однако затем убывание замедляется, поскольку высокочастотные компоненты погрешности уже в значительной мере подавлены. Поэтому имеет смысл выполнять подряд лишь несколько итераций типа S.

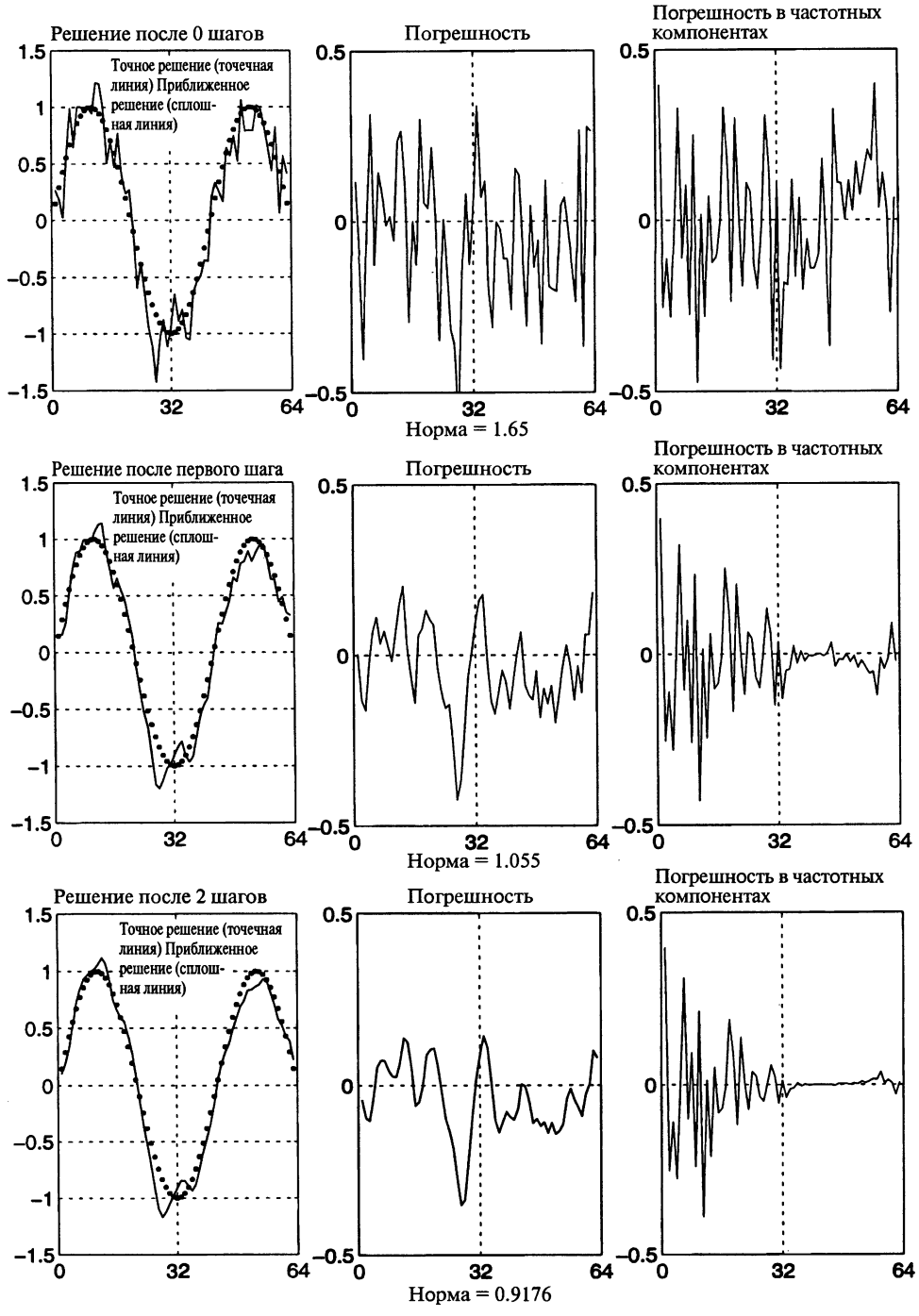


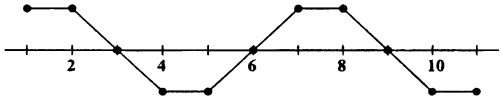
Рис. 6.15. Иллюстрация сходимости взвешенного метода Якоби.

Рекурсивная структура многосеточного метода

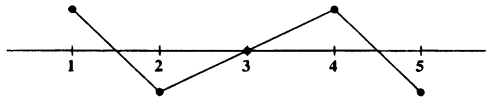
Используя введенную терминологию, можно дать следующее описание рекурсивной структуры многосеточного метода. На самой мелкой сетке $P^{(k)}$ метод подавляет правую половину частотных компонент погрешности приближенного решения. Это достигается применением только что описанного оператора сглаживания S. На следующей сетке с вдвое меньшим числом узлов подавляется правая половина остальных частотных компонент погрешности. Так происходит потому, что двойное уменьшение числа узлов на более грубой сетке имеет видимый эффект удвоения частот. Это иллюстрируется приводимым ниже примером.

Пример 6.16

$N = 12$, $k = 4$
низкая частота,
поскольку $k < \frac{N}{2}$
$\sin \frac{\pi \cdot 4 \cdot j}{12}$
для $1 \le j \le 11$



$N = 6$, $k = 4$
высокая частота,
поскольку $k > \frac{N}{2}$
$\sin \frac{\pi \cdot 4 \cdot j}{6}$
для $1 \le j \le 5$



На следующей, еще более грубой сетке подавляется правая половина оставшихся частот, и так далее, пока не дойдем до решаемой точно задачи $P^{(1)}$ (с одним неизвестным). Это показано схематически на рис. 6.16. Назначение операторов сужения и интерполяции — в том, чтобы преобразовать приближенное решение с данной сетки на соседнюю, более грубую или более мелкую.

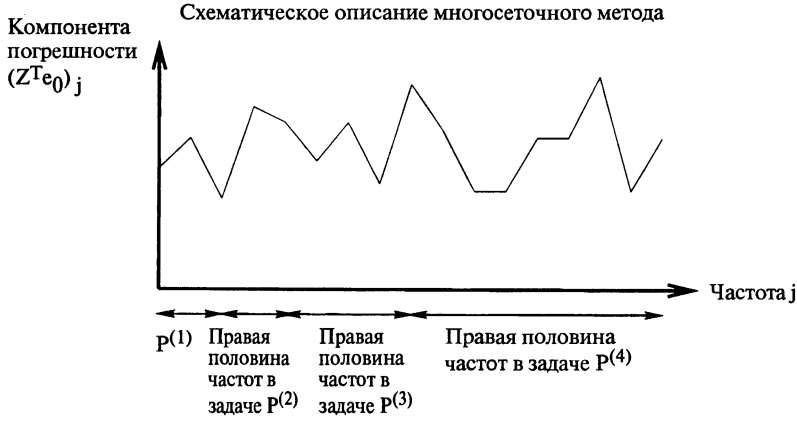


Рис. 6.16. Схематическое описание подавления компонент погрешности в многосеточном методе.

Оператор сужения в одномерном случае

Обратимся теперь к оператору сужения R. Он аппроксимирует правую часть $r^{(i)}$ задачи $P^{(i)}$ на соседней, более грубой сетке, в результате чего получаем $r^{(i-1)}$.

Простейшим способом вычисления вектора $r^{(i-1)}$ был бы простой *отбор* значений $r^{(i)}$ в общих узлах грубой и мелкой сеток. Однако к лучшим результатам приводит вычисление $r^{(i-1)}$ путем усреднения значений $r^{(i)}$ в соседних узлах мелкой сетки: значение в узле грубой сетки есть линейная комбинация значения в том же узле для мелкой сетки (с коэффициентом $\frac{1}{2}$) и значений в соседних узлах мелкой сетки (с коэффициентом $\frac{1}{4}$). Этот способ вычисления будем называть *усреднением*. Оба метода иллюстрируются на рис. 6.17.

Суммируя, мы можем дать следующее описание оператора сужения:

$$
\begin{aligned}
r^{(i-1)} &= R(r^{(i)}) \\
&\equiv P_i^{i-1} \cdot r^{(i)} \\
&= \begin{bmatrix}
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & & & & \\
 & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & & \\
 & & & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & \\
 & & & & & \ddots & \ddots & \ddots & & & \\
 & & & & & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4}
\end{bmatrix} \cdot r^{(i)}.
\end{aligned}
\tag{6.55}
$$

Верхний индекс $i-1$ и нижний индекс i матрицы P_i^{i-1} указывают, что она действует с сетки, состоящей из $2^i - 1$ узлов, в сетку с $2^{i-1} - 1$ узлами.

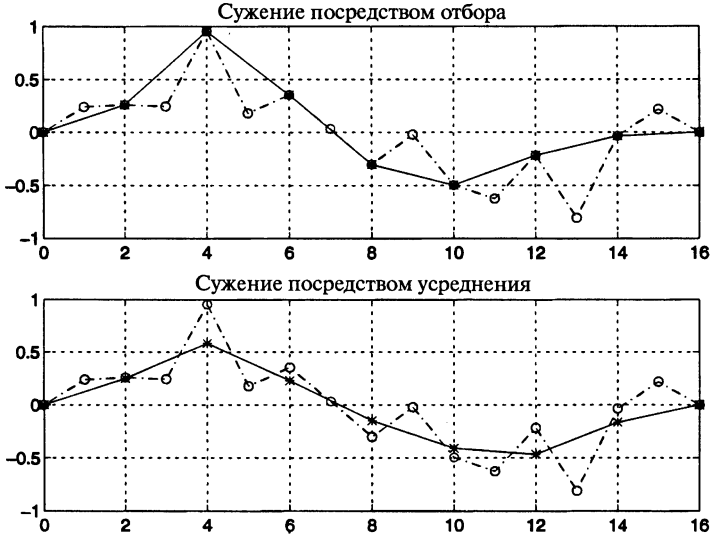


Рис. 6.17. Сужение с сетки с $2^4 - 1 = 15$ узлами на сетку с $2^3 - 1 = 7$ узлами. (Показаны также граничные узлы с нулевыми значениями.)

В двумерном случае сужение означало бы усреднение с участием восьми ближайших соседей данного узла сетки: значение в самом этом узле берется с коэффициентом $\frac{1}{4}$, значения в соседних узлах слева, справа, сверху и снизу берутся с коэффициентом $\frac{1}{8}$ и значения в четырех оставшихся соседних узлах (сверху слева, снизу слева, сверху справа, снизу справа) — с коэффициентом $\frac{1}{16}$.

Оператор интерполяции в одномерном случае

Оператор интерполяции In отображает приближенное решение $d^{(i-1)}$ на грубой сетке в функцию $d^{(i)}$ на соседней, более мелкой сетке. Решение $d^{(i-1)}$ интерполируется на мелкую сетку, как показано на рис. 6.18: значения на мелкой

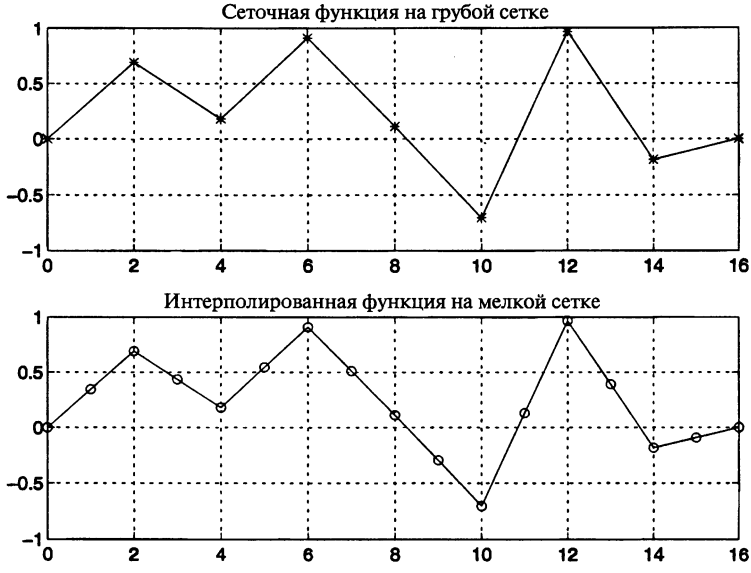


Рис. 6.18. Интерполяция с сетки с $2^3 - 1 = 7$ узлами на сетку с $2^4 - 1 = 15$ узлами. (Показаны также граничные узлы с нулевыми значениями.)

сетке восполняются посредством простой линейной интерполяции (при этом используются граничные узлы, в которых значения функции равны нулю). Математически это можно описать соотношением

$$d^{(i)} = In(d^{(i-1)}) \equiv P_{i-1}^{i} \cdot d^{(i-1)} = \begin{bmatrix} \frac{1}{2} & & & \\ 1 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ & 1 & \ddots & \\ & \frac{1}{2} & \ddots & \frac{1}{2} \\ & & \ddots & 1 \\ & & & \frac{1}{2} \end{bmatrix} \cdot d^{(i-1)}. \qquad (6.56)$$

Верхний индекс i и нижний индекс $i-1$ матрицы P_{i-1}^{i} указывают, что она действует с сетки, состоящей из $2^{i-1}-1$ узлов, в сетку с $2^{i}-1$ узлами.

Заметим, что $P_{i-1}^{i} = 2(P_{i}^{i-1})^{T}$. Другими словами, интерполирование и сужение по существу являются транспонированными друг к другу операциями. Этот факт будет важен для проводимого ниже анализа сходимости.

В двумерном случае интерполяция также связана с усреднением по узлам грубой сетки, соседствующим с данным узлом мелкой сетки (интерполяция тривиальна, если узел мелкой сетки принадлежит и грубой; в интерполяцию вовлечены два узла, если у узла мелкой сетки лишь два соседа на грубой (слева и справа или сверху и снизу), и четыре узла, в противном случае).

Соберем все вместе

Проведем восемь итераций только что описанного алгоритма для задачи, представленной на двух графиках рис. 6.19a. На них показаны точное решение x (левый график) и правая часть b (правый график). Число неизвестных равно $2^{7}-1 = 127$. Сходимость многосеточного метода иллюстрируют три графика на рис. 6.19b. На левом графике нижнего ряда рис. 6.19a показано отношение последовательных невязок $\|r_{m+1}\|/\|r_{m}\|$; индекс m есть номер итерации многосеточного метода (т. е. номер обращения к FMG; см. алгоритм 6.17). Эти

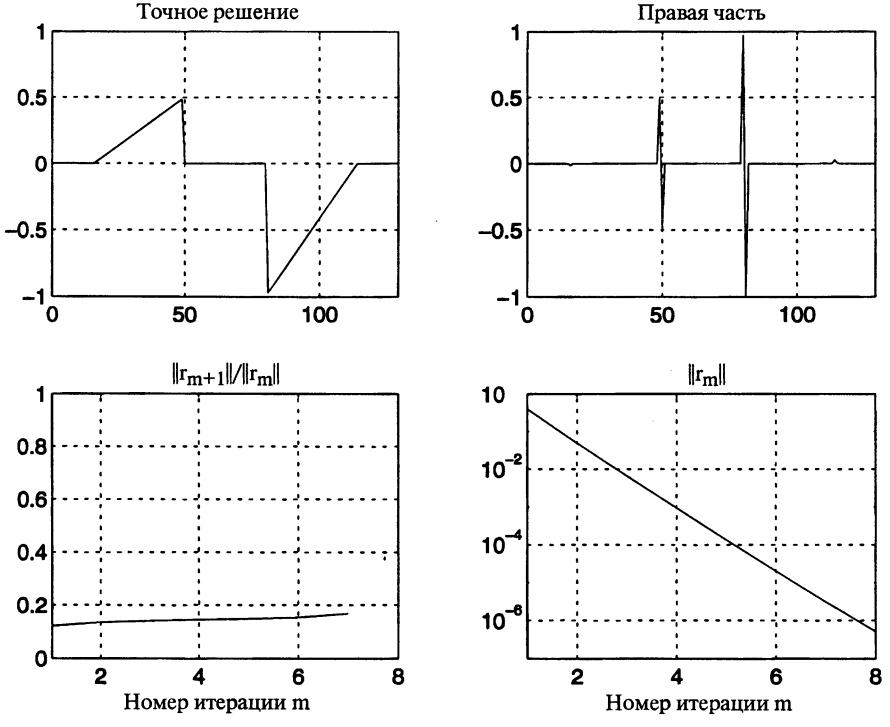


Рис. 6.19a. Решение одномерной модельной задачи многосеточным методом.

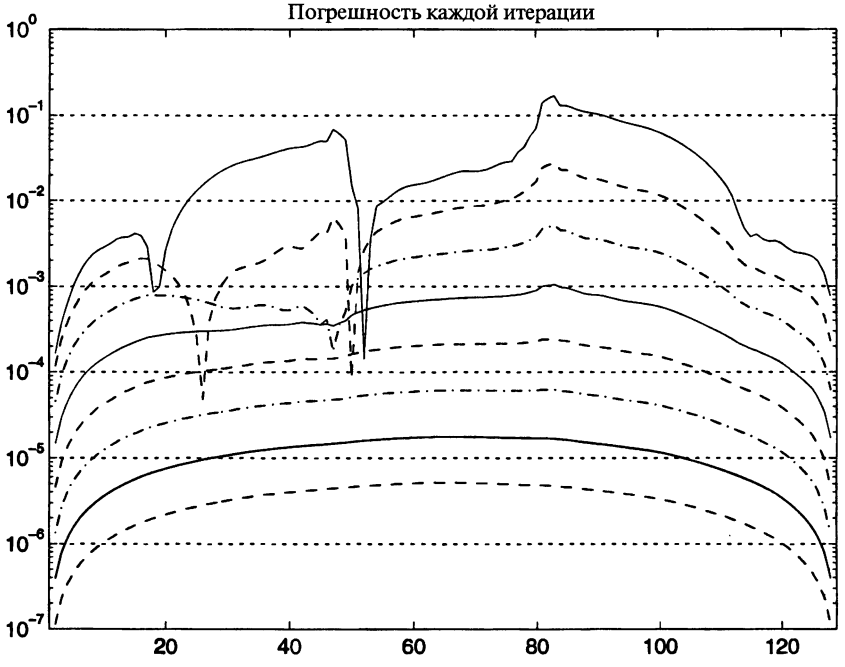


Рис. 6.19b. *Продолжение.*

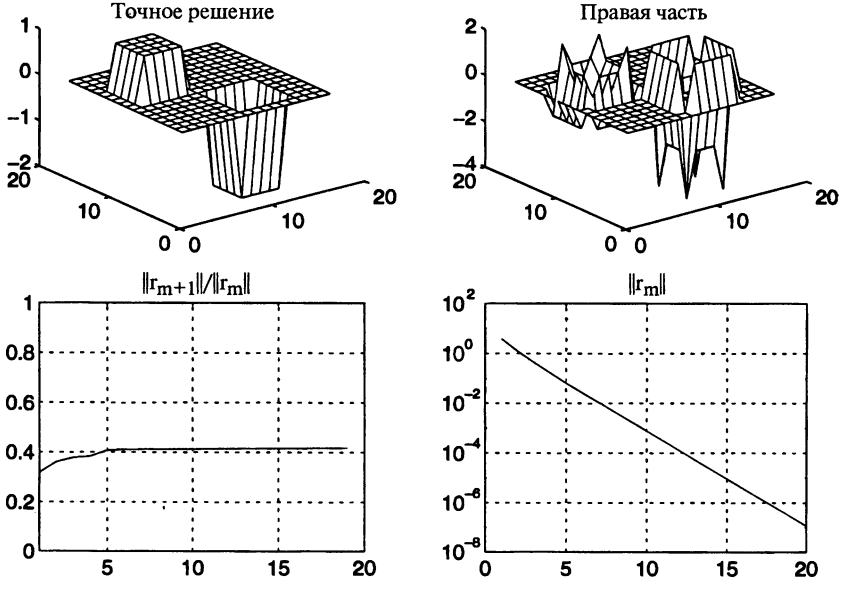


Рис. 6.20. Решение двумерной модельной задачи многосеточным методом.

отношения равны приблизительно 0.15, что означает: на каждой итерации многосеточного метода невязка убывает более чем в 6 раз. Быстрая сходимость видна и из правого графика того же ряда, представляющего собой полулогарифмический график величины $\|r_m\|$ как функции от m. Как и следовало ожидать, это прямая линия с угловым коэффициентом $\log_{10}(0.15)$. Наконец, на графике рис. 6.19b изображены все восемь векторов ошибки $x_n - x$. Из этого полулогарифмического графика видно, что векторы постепенно сглаживаются и становятся параллельными, получаясь один из другого умножением на постоянный коэффициент $|\log_{10}(0.15)|$.

На рис. 6.20 показан аналогичный пример для двумерной модельной задачи.

Доказательство сходимости

В заключение, мы дадим набросок доказательства сходимости, показывающего, что общая погрешность за один FMG V-цикл умножается на константу, меньшую, чем 1, и не зависящую от размера сетки $N_k = 2^k - 1$. Это означает, что число FMG V-циклов, необходимых для того, чтобы снизить погрешность в любое число раз, большее 1, не зависит от k. Поэтому суммарная работа пропорциональна стоимости одного FMG V-цикла, т. е. пропорциональна числу неизвестных n.

Мы упростим доказательство, анализируя лишь один V-цикл и предполагая по индукции, что задача на грубой сетке решается *точно* [43]. В действительности, она не решается вполне точно, однако нашего грубого анализа достаточно, чтобы понять смысл доказательства: низкочастотная погрешность ослабляется на более грубой сетке, а высокочастотная погрешность исключается на мелкой.

Выпишем теперь все формулы, определяющие V-цикл, и объединим их в одну формулу типа «новый $e^{(i)} = M \cdot e^{(i)}$», где $e^{(i)} = x^{(i)} - x$ есть вектор ошибки, а M — матрица, собственные значения которой определяют скорость сходимости. Наша задача — показать, что собственные значения отделены от 1 расстоянием, не зависящим от i. Номера, упоминаемые в приводимой ниже записи, — это номера строк в алгоритме 6.16.

$$
\begin{aligned}
(a)\quad & x^{(i)} = S(b(i), x(i)) = R_{2/3}^{(i)} x^{(i)} + b^{(i)}/3 \\
& \qquad \text{согласно строке 1) и уравнению (6.54),} \\
(b)\quad & r^{(i)} = T^{(i)} \cdot x^{(i)} - b^{(i)} \\
& \qquad \text{согласно строке 2),} \\
& d^{(i)} = In(MGV(4 \cdot R(r^{(i)}), 0)) \\
& \qquad \text{согласно строке 3)} \\
& \quad = In(\left[T^{(i-1)}\right]^{-1}(4 \cdot R(r^{(i)}))) \\
& \qquad \text{по нашему предположению о том, что задача} \\
& \qquad \text{на грубой сетке решается точно} \\
& \quad = In(\left[T^{(i-1)}\right]^{-1}(4 \cdot P_i^{i-1} r^{(i)})) \\
& \qquad \text{согласно уравнению (6.55)}
\end{aligned}
$$

$$
\begin{aligned}
(c) \qquad & = P_{i-1}^{i}\left(\left[T^{(i-1)}\right]^{-1}\left(4 \cdot P_{i}^{i-1} r^{(i)}\right)\right) \\
& \qquad \text{согласно уравнению (6.56)} \\
(d) \quad x^{(i)} & = x^{(i)} - d^{(i)} \\
& \qquad \text{согласно строке 4)} \\
(e) \quad x^{(i)} & = S(b(i), x(i)) = R_{2/3}^{(i)} x^{(i)} + b^{(i)}/3 \\
& \qquad \text{согласно строке 5).}
\end{aligned}
$$

Чтобы получить уравнение для пересчета вектора ошибки $e^{(i)}$, вычтем из строк (a) и (e) тождество $x = R_{2/3}^{(i)} x + b^{(i)}/3$, из строки (b) — тождество $0 = T^{(i)} \cdot x - b^{(i)}$ и, наконец, из строки (d) тождество $x = x$. В результате имеем:

$$
\begin{aligned}
(a) \quad & e^{(i)} = R_{2/3}^{(i)} e^{(i)}, \\
(b) \quad & r^{(i)} = T^{(i)} \cdot e^{(i)}, \\
(c) \quad & d^{(i)} = P_{i-1}^{i}\left(\left[T^{(i-1)}\right]^{-1}\left(4 \cdot P_{i}^{i-1} r^{(i)}\right)\right), \\
(d) \quad & e^{(i)} = e^{(i)} - d^{(i)}, \\
(e) \quad & e^{(i)} = R_{2/3}^{(i)} e^{(i)}.
\end{aligned}
$$

Подставляя каждое из этих уравнений в следующее, приходим к формуле, показывающей, как изменяется вектор ошибки в V-цикле:

$$
\begin{aligned}
\text{новый } e^{(i)} &= R_{2/3}^{(i)} \left\{ I - P_{i-1}^{i} \cdot \left[T^{(i-1)}\right]^{-1} \cdot \left(4 \cdot P_{i}^{i-1} T^{(i)}\right) \right\} R_{2/3}^{(i)} \cdot e^{(i)} \\
&\equiv M \cdot e^{(i)}. \qquad (6.57)
\end{aligned}
$$

Теперь нужно вычислить собственные значения матрицы M. Упростим вначале уравнение (6.57), используя соотношения $P_{i-1}^{i} = 2 \cdot (P_{i}^{i-1})^{T}$ и

$$
T^{(i-1)} = 4 \cdot P_{i}^{i-1} T^{(i)} P_{i-1}^{i} = 8 \cdot P_{i}^{i-1} T^{(i)} (P_{i}^{i-1})^{T} \qquad (6.58)
$$

(см. вопрос 6.15). Подставляя их в выражение для M в формуле (6.57), находим

$$
M = R_{2/3}^{(i)} \left\{ I - (P_{i}^{i-1})^{T} \cdot \left[P_{i}^{i-1} T^{(i)} (P_{i}^{i-1})^{T} \right]^{-1} \cdot \left(P_{i}^{i-1} T^{(i)}\right) \right\} R_{2/3}^{(i)},
$$

или, опуская индексы, чтобы упростить запись,

$$
M = R_{2/3} \left\{ I - P^{T} \cdot [P T P^{T}]^{-1} \cdot P T \right\} R_{2/3}. \qquad (6.59)
$$

Используем теперь то обстоятельство, что все матрицы, составляющие M (т. е. T, $R_{2/3}$ и P), могут быть (почти) диагонализованы с помощью матриц $Z = Z^{(i)}$ и $Z^{(i-1)}$, состоящих из собственных векторов соответственно задач $T = T^{(i)}$ и $T^{(i-1)}$. Вспомним, что $Z = Z^{T} = Z^{-1}$, $T = Z \Lambda Z$ и $R_{2/3} = Z(I - \Lambda/3)Z \equiv Z \Lambda_{R} Z$. Предоставляем читателю проверку соотношения $Z^{(i-1)} P Z^{(i)} = \Lambda_{P}$, где Λ_{P} — почти диагональная матрица (см. вопрос 6.15), т. е.

$$
\lambda_{P,jk} = \begin{cases} (+1 + \cos \frac{\pi j}{2^{i}})/\sqrt{8}, & \text{если } k = j, \\ (-1 + \cos \frac{\pi j}{2^{i}})/\sqrt{8}, & \text{если } k = 2^{i} - j, \\ 0, & \text{в противном случае.} \end{cases} \qquad (6.60)
$$

Это позволяет написать

$$\begin{aligned} ZMZ &= (ZR_{2/3}Z) \\ &\quad \times \Big\{ I - (ZP^T Z^{(i-1)}) \cdot \Big[(Z^{(i-1)}PZ)(ZTZ)(ZP^T Z^{(i-1)}) \Big]^{-1} \\ &\quad \times (Z^{(i-1)}PZ)(ZTZ) \Big\} \cdot (ZR_{2/3}Z) \\ &= \Lambda_R \cdot \{ I - \Lambda_P^T [\Lambda_P \Lambda \Lambda_P^T]^{-1} \Lambda_P \Lambda \} \cdot \Lambda_R. \end{aligned}$$

Поскольку $Z = Z^{-1}$, матрица ZMZ подобна M, следовательно, имеет те же собственные значения. Кроме того, матрица ZMZ почти диагональна, что означает: ненулевые элементы в ней могут находиться только на главной диагонали и «пердиагонали» (т. е. диагонали, идущей из левого нижнего угла матрицы в ее правый верхний угол). Это позволяет вычислить собственные значения матрицы M в явном виде.

Теорема 6.11. *Независимо от i, матрица M имеет собственные значения $1/9$ и 0. Поэтому многосеточный метод сходится с фиксированной скоростью, не зависящей от числа неизвестных.*

По поводу доказательства теоремы см. вопрос 6.15. Более общий анализ сходимости можно найти в [268].

Реализация данного алгоритма обсуждается в вопросе 6.16. Интернет-страница [91] содержит ссылки на обширную литературу по этой теме, программы и прочее.

6.10. Декомпозиция области

Декомпозиция области при решении разреженных систем линейных уравнений является предметом текущих исследований. Обзоры последних результатов в этом направлении можно найти в [49, 116, 205] и, особенно, в [232]. Здесь мы приведем лишь простые примеры.

Потребность в методах, отличных от описанных выше, проистекает из нерегулярности и размера реальных задач, а также из необходимости приспособить алгоритмы к параллельным компьютерам. Самые быстрые методы из тех, что обсуждались до сих пор, а именно методы, основанные на блочной циклической редукции, алгоритме FFT и многосеточном подходе, работают только или быстрее всего для очень регулярных задач типа нашей модельной задачи, т. е. уравнения Пуассона в прямоугольнике, дискретизованного посредством равномерной сетки. Однако с решением реальной задачи может быть связан не прямоугольник, а менее регулярная область, представляющая физический объект вроде крыла на рис. 2.12. Этот рисунок показывает также, что в тех частях области, где, как ожидается, решение является менее гладким, сетка может сгущаться по сравнению с частями, где решение гладкое. Кроме того, нам могут встретиться более сложные уравнения, чем уравнение Пуассона, или даже различные уравнения в разных подобластях. Регулярна задача или нет, она может из-за своего размера не поместиться в памяти компьютера и тогда ее, возможно, придется решать «по частям». Или же мы можем захотеть разбить задачу на части, которые допускают одновременное решение на параллельной машине.

Теория декомпозиции области рассматривает все эти вопросы с тем, чтобы указать систематический способ конструирования «гибридных» алгоритмов из более простых методов, обсуждавшихся в предыдущих разделах. Эти более простые методы применяются к меньшим и более регулярным подзадачам исходной задачи, после чего из полученных частичных решений «монтируется» решение задачи в целом. Если вся задача не помещается в памяти компьютера, то подзадачи могут решаться по очереди, а на параллельном компьютере — одновременно. Ниже будут приведены соответствующие примеры. В общем случае, имеется много способов разбиения большой задачи на части, много способов решения индивидуальных подзадач и много способов монтажа их решений. Теория декомпозиции области не имеет магического рецепта для выбора в каждом случае наилучших способов, однако она указывает набор вариантов, среди которых разумно сделать выбор. В некоторых случаях (например, для задач, достаточно похожих на уравнение Пуассона) эта теория приводит к «оптимальным методам» (где выполняется работа $O(1)$ на одно неизвестное).

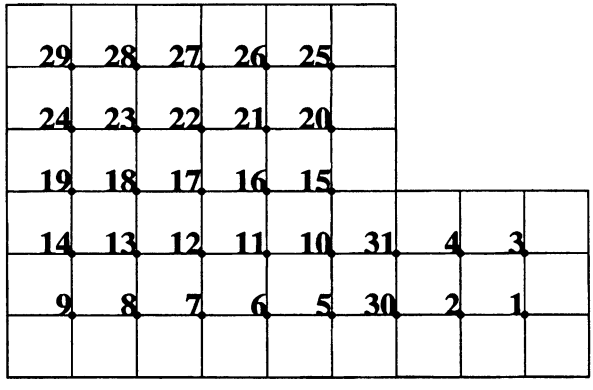
Наше обсуждение будет разделено на две части, соответствующие методам *без перекрытия* и методам *с перекрытием*.

6.10.1. Методы без перекрытия

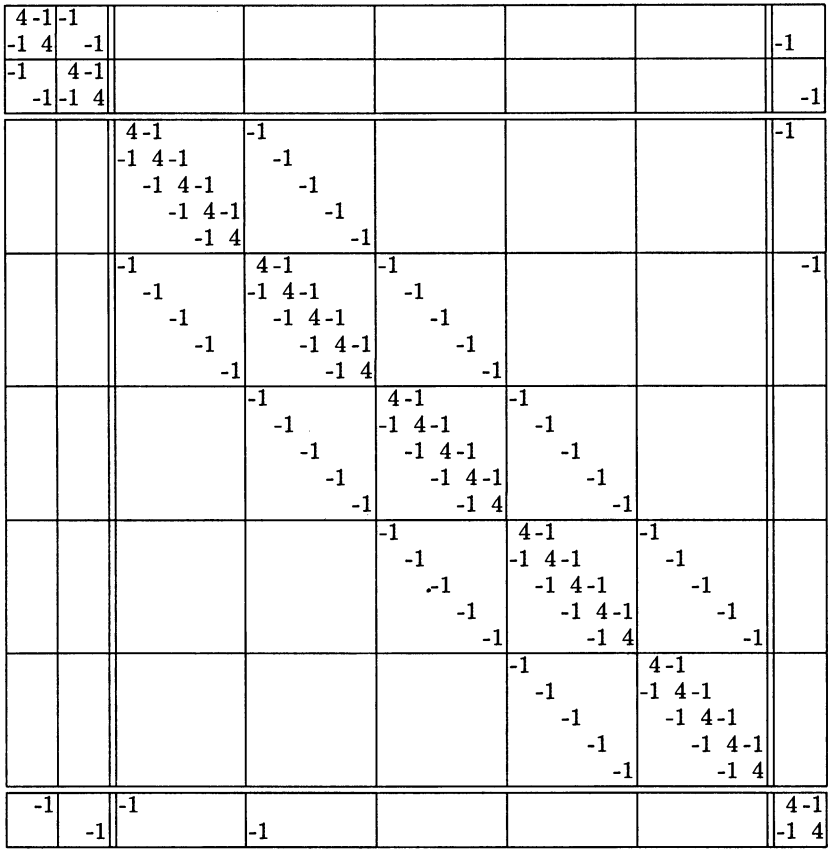
В литературе методы этого типа называют еще методами *подструктур*, или методами *дополнений Шура*. Такие методы используются уже несколько десятилетий, особенно специалистами по строительной механике, для разбиения больших задач на части, могущие быть размещенными в памяти компьютера.

Для простоты, проиллюстрируем данный тип методов с помощью обычного уравнения Пуассона с граничными условиями Дирихле, дискретизованного посредством 5-точечного шаблона. Однако уравнение будем рассматривать не на квадрате, а на *L-образной области*. Эту область можно разбить на две: малый квадрат и большой с вдвое большей стороной, причем малый квадрат примыкает к нижней части правой стороны большого квадрата. Построим метод, использующий нашу способность быстро решать задачи на квадратах.

Для грубой сетки, показанной на рисунке, каждый внутренний узел L-образной области помечен номером (стоящим слева сверху от узла).



Заметим, что вначале пронумерованы внутренние узлы каждой из подобластей (номера с 1 по 4 и с 5 по 29) и только потом узлы на их общей границе (номера 30 и 31). В результате получается такая матрица:



$$\equiv A \equiv \left[\begin{array}{c|c|c} A_{11} & 0 & A_{13} \\ \hline 0 & A_{22} & A_{23} \\ \hline A_{13}^T & A_{23}^T & A_{33} \end{array}\right].$$

Здесь $A_{11} = T_{2\times 2}$, $A_{22} = T_{5\times 5}$ и $A_{33} = T_{2\times 1} \equiv T_2 + 2I_2$, где матрица T_N определена в (6.3), а матрица $T_{N\times N}$ — в (6.14). Одним из важнейших свойств этой матрицы является то, что $A_{12} = 0$, поскольку между внутренними узлами двух подобластей нет прямого сцепления. Сцепление осуществляется лишь через занумерованные последними узлы общей границы подобластей (узлы 30 и 31). Таким образом, блок A_{13} соответствует сцеплению между границей и малым квадратом, а блок A_{23} — сцеплению между границей и большим квадратом.

Чтобы понять, как извлечь выгоду из специальной структуры матрицы A при решении системы $Ax = b$, запишем блочное LDU-разложение этой матрицы

в виде

$$A = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ A_{13}^T A_{11}^{-1} & A_{23}^T A_{22}^{-1} & I \end{bmatrix} \cdot \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & S \end{bmatrix} \cdot \begin{bmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & I \end{bmatrix},$$

где матрица

$$S = A_{33} - A_{13}^T A_{11}^{-1} A_{13} - A_{23}^T A_{22}^{-1} A_{23} \tag{6.61}$$

называется *дополнением Шура* ведущей главной подматрицы, содержащей блоки A_{11} и A_{22}. Следовательно, можно написать

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 & -A_{11}^{-1} A_{13} \\ 0 & A_{22}^{-1} & -A_{22}^{-1} A_{23} \\ 0 & 0 & I \end{bmatrix} \cdot \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & S^{-1} \end{bmatrix} \cdot \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -A_{13}^T A_{11}^{-1} & -A_{23}^T A_{22}^{-1} & I \end{bmatrix}.$$

Таким образом, чтобы умножить вектор на A^{-1}, нам нужно умножать на блочные элементы этой факторизованной формы, а именно на блоки A_{13} и A_{23} (и транспонированные к ним), A_{11}^{-1}, A_{22}^{-1} и S^{-1}. Умножение на A_{13} и A_{23} стоит дешево, так как эти блоки очень разрежены. Умножение на A_{11}^{-1} и A_{22}^{-1} тоже дешево: ведь подобласти были выбраны так, чтобы были применимы алгоритм FFT, блочная циклическая редукция, многосеточный метод и некоторые другие быстрые методы из обсуждавшихся выше. Остается объяснить, как происходит умножение на матрицу S^{-1}.

Поскольку на границе гораздо меньше узлов, чем в подобластях, порядок блоков A_{33} и S много меньше порядка блоков A_{11} и A_{22}; этот эффект еще более выражен при сгущении сетки. Как и A, блок S симметричен и положительно определен, но является (в данном случае) плотным. Чтобы вычислить его в явном виде, потребовалось бы решить по одной задаче для каждой подобласти на каждый из общих граничных узлов (что соответствует произведениям $A_{11}^{-1} A_{13}$ и $A_{22}^{-1} A_{23}$ в формуле (6.61)). Это, разумеется, можно сделать, после чего можно было бы факторизовать S посредством плотного алгоритма Холесского, а затем решить систему с этой матрицей. Но такой способ действий был бы чересчур дорогостоящим, куда дороже, чем простое умножение вектора на матрицу S, для чего, согласно (6.61), понадобится лишь одно решение каждой задачи на подобласти. Это обстоятельство делает привлекательным применение итерационных методов крыловского подпространства, таких, как метод CG (разд. 6.6), поскольку в них матрица S используется лишь в произведениях с векторами. Число матрично-векторных умножений в методе CG зависит от числа обусловленности матрицы S. Оказывается (и в этом состоит особая привлекательность метода декомпозиции области!), что S гораздо лучше обусловлена, чем исходная матрица A (ее число обусловленности растет как $O(N)$, а не как $O(N^2)$), а потому метод CG сходится быстро [116, 205].

В более общем случае имеется $k > 2$ подобластей, разделенных границами (см. рис. 6.21, где подобласти разделены жирными линиями). Если пронумеровать последовательными числами узлы каждой подобласти и лишь затем

граничные узлы, то получим матрицу вида

$$A = \left[\begin{array}{ccc|c} A_{1,1} & & 0 & A_{1,k+1} \\ & \ddots & & \vdots \\ 0 & & A_{k,k} & A_{k,k+1} \\ \hline A_{1,k+1}^T & \cdots & A_{k,k+1}^T & A_{k+1,k+1} \end{array}\right]. \tag{6.62}$$

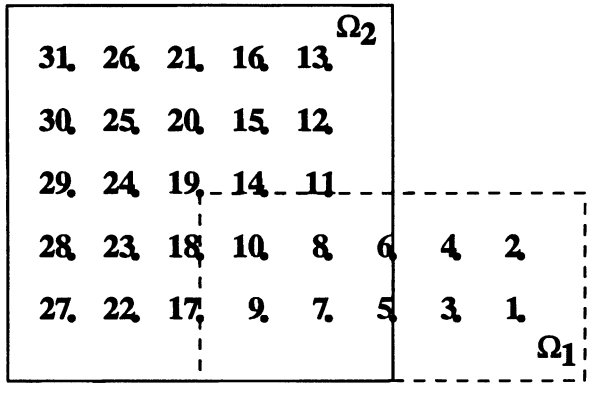
Снова эту матрицу можно факторизовать, факторизуя независимо друг от друга диагональные блоки A_{ii}, а затем формируя дополнение Шура $S = A_{k+1,k+1} - \sum_{i=1}^{k} A_{i,k+1}^T A_{i,i}^{-1} A_{i,k+1}$.

В том случае, если имеется более чем один граничный сегмент, S обладает структурой, которую можно использовать для предобуславливания. Так, если внутренние узлы каждого граничного сегмента занумерованы прежде, чем узлы на пересечении таких сегментов, то возникает блочная структура, схожая со структурой матрицы A. Диагональные блоки в S устроены сложно, но их можно аппроксимировать матрицей $T_N^{1/2}$, допускающей эффективное обращение посредством метода FFT [36, 37, 38, 39, 40]. Мы можем резюмировать достижения в теории метода следующим образом: при подходящем выборе предобуславливателя для матрицы S можно добиться, чтобы число шагов алгоритма CG не зависело от числа N граничных узлов [231].

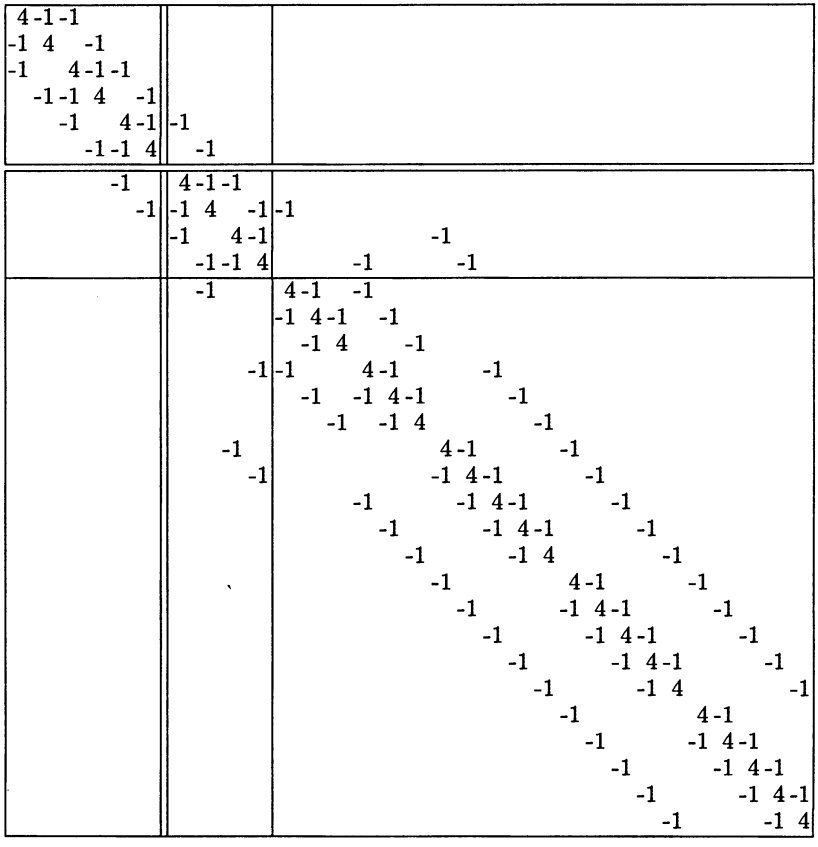
6.10.2. Методы с перекрытием

Мы говорили в предыдущем разделе о методах *без перекрытия*, потому что области, соответствующие узлам диагональных блоков A_{ii}, были непересекающимися, что приводило к блочно-диагональной структуре, показанной в уравнении (6.62). В данном разделе мы допускаем пересечение областей, иллюстрируемое рисунком. Как увидим, это позволит нам построить алгоритм, сравнимый по скорости с многосеточным методом, но применимый к более широкому кругу задач.

Прямоугольник на рисунке, граница которого указана пунктиром, обозначим через Ω_1, а квадрат с границей, проведенной сплошной линией, через Ω_2. Узлы пронумерованы таким образом, чтобы первые номера получили узлы из Ω_1, а последние — из Ω_2; при этом средние номера будут присвоены узлам из пересечения $\Omega_1 \cap \Omega_2$.



Области Ω_1 и Ω_2 отражены и в приводимой ниже матрице A, которая, с точностью до порядка строк и столбцов, отвечающего теперь нумерации на рисунке, совпадает с матрицей разд. 6.10.1:



Разбиение матрицы на блоки определяется границами между областями. Одинарные линии делят матрицу на части, ассоциированные с узлами из области Ω_1 (их номера — с 1 до 10) и ее дополнения $\Omega \setminus \Omega_1$ (номера — с 11 по 31). Двойные линии делят матрицу на части, соответствующие узлам из Ω_2 (номера — с 7 по 31) и $\Omega \setminus \Omega_2$ (номера — с 1 до 6). С этими разбиениями согласованы индексы в следующих блочных представлениях:

$$A = \left[\begin{array}{c|c} A_{\Omega_1,\Omega_1} & A_{\Omega_1,\Omega\setminus\Omega_1} \\ \hline A_{\Omega\setminus\Omega_1,\Omega_1} & A_{\Omega\setminus\Omega_1,\Omega\setminus\Omega_1} \end{array}\right] = \left[\begin{array}{c\|c} A_{\Omega\setminus\Omega_2,\Omega\setminus\Omega_2} & A_{\Omega\setminus\Omega_2,\Omega_2} \\ \hline\hline A_{\Omega_2,\Omega\setminus\Omega_2} & A_{\Omega_2,\Omega_2} \end{array}\right].$$

Соответствующим образом разбиваются на части и векторы:

$$x = \left[\begin{array}{c} x_{\Omega_1} \\ \hline x_{\Omega\setminus\Omega_1} \end{array}\right] = \left[\begin{array}{c} x(1:10) \\ \hline x(11:31) \end{array}\right]$$

$$= \left[\begin{array}{c} x_{\Omega\setminus\Omega_2} \\ \hline\hline x_{\Omega_2} \end{array}\right] = \left[\begin{array}{c} x(1:6) \\ \hline\hline x(7:31) \end{array}\right].$$

Теперь у нас достаточно обозначений для того, чтобы описать два основных алгоритма декомпозиции области с перекрытием. Простейший из них называется (по причинам, связанным с его историей) *аддитивным методом Шварца*, но с равным основанием может именоваться *блочными итерациями Якоби с перекрытием* вследствие своей схожести с (блочным) методом Якоби из разд. 6.5 и 6.6.5.

Алгоритм 6.18. *Аддитивный метод Шварца для пересчета приближенного решения x_i системы $Ax = b$ в более точное приближение x_{i+1}:*

$r_i = b - Ax_i$ /* *вычислить невязку* */
$x_{i+1} = 0$
$x_{i+1,\Omega_1} = x_{i,\Omega_1} + A^{-1}_{\Omega_1,\Omega_1} \cdot r_{\Omega_1}$ /* *модифицировать решение на* Ω_1 */
$x_{i+1,\Omega_2} = x_{i+1,\Omega_2} + A^{-1}_{\Omega_2,\Omega_2} \cdot r_{\Omega_2}$ /* *модифицировать решение на* Ω_2 */

Этот алгоритм можно записать одной строкой следующим образом:

$$x_{i+1} = x_i + \begin{bmatrix} A^{-1}_{\Omega_1,\Omega_1} \cdot r_{\Omega_1} \\ \hline 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \hline\hline A^{-1}_{\Omega_2,\Omega_2} \cdot r_{\Omega_2} \end{bmatrix}.$$

В словесном описании алгоритм работает так: поправка $A^{-1}_{\Omega_1,\Omega_1} r_{\Omega_1}$ соответствует решению уравнения Пуассона только на области Ω_1; при этом используются граничные условия, зависящие от предыдущего приближенного решения x_i, в узлах 11, 14, 17, 18 и 19. Аналогично интерпретируется поправка $A^{-1}_{\Omega_2,\Omega_2} r_{\Omega_2}$, использующая граничные условия, зависящие от x_i, в узлах 5 и 6.

В данном случае, Ω_i суть прямоугольники, поэтому для определения поправок $A^{-1}_{\Omega_i,\Omega_i} r_{\Omega_i}$ можно применить любой из ранее описанных быстрых алгоритмов, например многосеточный метод. Поскольку аддитивный метод Шварца является итерационным, нет необходимости решать задачи на областях Ω_i точно.

Действительно, аддитивный метод Шварца обычно используется как предобуславливатель для методов крыловского подпространства типа алгоритма сопряженных градиентов (см. разд. 6.6.5). В обозначениях разд. 6.6.5 предобуславливатель M описывается формулой

$$M^{-1} = \left[\begin{array}{c|c} A^{-1}_{\Omega_1,\Omega_1} & 0 \\ \hline 0 & 0 \end{array}\right] + \left[\begin{array}{c||c} 0 & 0 \\ \hline\hline 0 & A^{-1}_{\Omega_2,\Omega_2} \end{array}\right].$$

Если бы Ω_1 и Ω_2 не пересекались, матрица M^{-1} упростилась бы к блочно-диагональному виду

$$\begin{bmatrix} A^{-1}_{\Omega_1,\Omega_1} & 0 \\ 0 & A^{-1}_{\Omega_2,\Omega_2} \end{bmatrix},$$

что соответствует блочным итерациям Якоби. Однако, как мы знаем, метод Якоби сходится не слишком быстро, потому что «информация» о решении передается из одной области в другую через границу между ними довольно медленно (см. обсуждение в начале разд. 6.9). Но если пересечение двух областей составляет достаточно большую часть каждой из них, информация будет распространяться со скоростью, достаточной для того, чтобы обеспечить быструю сходимость. Разумеется, пересечение не должно быть слишком велико,

поскольку это сильно увеличило бы работу. При построении хорошего метода декомпозиции области целью является выбор подобластей и их пересечений, гарантирующий быструю сходимость при выполнении по возможности меньшей работы. Ниже мы вернемся к вопросу о том, каким образом сходимость зависит от размера перекрытия.

Из обсуждения в разд. 6.5 мы знаем, что метод Гаусса—Зейделя обычно эффективнее метода Якоби. Это же справедливо для рассматриваемой ситуации: *блочный метод Гаусса—Зейделя с перекрытием* (чаще называемый *мультипликативным методом Шварца*) нередко оказывается вдвое более быстрым, чем аддитивные блочные итерации Якоби (т. е. аддитивный метод Шварца).

Алгоритм 6.19. *Мультипликативный метод Шварца для пересчета приближенного решения x_i системы $Ax = b$:*

(1) $r_{\Omega_1} = (b - Ax_i)_{\Omega_1}$
/ вычислить невязку для x_i на области Ω_1 */*

(2) $x_{i+\frac{1}{2},\Omega_1} = x_{i,\Omega_1} + A^{-1}_{\Omega_1,\Omega_1} \cdot r_{\Omega_1}$
/ перевычислить решение на Ω_1 */*

(2′) $x_{i+\frac{1}{2},\Omega\setminus\Omega_1} = x_{i,\Omega\setminus\Omega_1}$

(3) $r_{\Omega_2} = (b - Ax_{i+\frac{1}{2}})_{\Omega_2}$
/ вычислить невязку для $x_{i+\frac{1}{2}}$ на области Ω_2 */*

(4) $x_{i+1,\Omega_2} = x_{i+\frac{1}{2},\Omega_2} + A^{-1}_{\Omega_2,\Omega_2} \cdot r_{\Omega_2}$
/ перевычислить решение на Ω_2 */*

(4′) $x_{i+1,\Omega\setminus\Omega_2} = x_{i+\frac{1}{2},\Omega\setminus\Omega_2}$

Заметим, что строки (2′) и (4′) не требуют перемещения данных, если $x_{i+\frac{1}{2}}$ и x_{i+1} записываются на место прежнего вектора x_i.

В этом алгоритме, как и в алгоритме 6.18, вначале решается уравнение Пуассона на Ω_1, использующее граничные данные, зависящие от x_i. Затем решается уравнение Пуассона на Ω_2, но теперь используются только что пересчитанные граничные данные. И этот алгоритм может быть применен в качестве предобуславливателя для методов крыловского подпространства.

На практике используется большее, чем 2, число подобластей Ω_i. Причинами могут быть более сложная геометрия основной области, наличие многих независимых параллельных процессоров, позволяющее независимо решать подзадачи $A^{-1}_{\Omega_i,\Omega_i} r_{\Omega_i}$, или просто желание сделать подзадачи $A^{-1}_{\Omega_i,\Omega_i} r_{\Omega_i}$ малыми и дешево решаемыми.

Приведем краткое изложение теоретического анализа сходимости этих методов для модельной задачи и подобных ей дифференциальных уравнений с частными производными. Пусть h — шаг сетки. Теория предсказывает число итераций, необходимых для сходимости, как функцию от h при h, стремящемся к нулю. В случае двух областей Ω_1 и Ω_2 указанное число итераций не зависит от h, если пересечение $\Omega_1 \cap \Omega_2$ составляет ненулевую долю всей области $\Omega_1 \cup \Omega_2$. Это привлекательное свойство, напоминающее о многосеточном методе, скорость сходимости которого также не зависела от шага сетки h. Однако в стоимость одной итерации входит *точное* решение подзадач на Ω_1 и Ω_2, что может быть сравнимо со стоимостью решения исходной задачи. Следовательно, задачи на Ω_1 и Ω_2 должны решаться очень дешево (как это было в

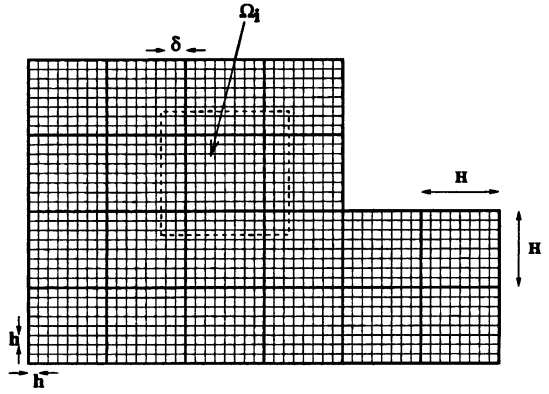


Рис. 6.21. Грубая и более точная дискретизации L-образной области.

разобранном выше примере с L-образной областью), чтобы общая стоимость не была слишком высока.

Предположим теперь, что имеется много областей Ω_i, каждая размера $H \gg h$. Иначе говоря, можно представлять себе Ω_i как элементы грубой сетки с шагом H, включающие в себя дополнительно некоторые ячейки мелкой сетки, находящиеся за границей крупной ячейки (это показано пунктирной линией на рис. 6.21).

Пусть число $\delta < H$ измеряет степень перекрытия подобластей, и пусть все три величины H, δ и h стремятся к нулю таким образом, чтобы доля перекрытия δ/H оставалась постоянной, причем $H \gg h$. Тогда число итераций, необходимых для сходимости, растет как $1/H$, т. е. *не зависит* от шага h мелкой сетки. Это близко к соответствующему свойству многосеточного метода, но хуже его, так как число итераций многосеточного метода постоянно и работа на одно неизвестное, выполняемая в нем, равна $O(1)$.

Для достижения такой же производительности нужна еще одна идея, которая (что, возможно, не является удивительным) похожа на идею многосеточного метода. Используем приближение A_H задачи на сетке с шагом H как *предобуславливатель грубой сетки* в дополнение к предобуславливателям мелкой сетки $A^{-1}_{\Omega_i,\Omega_i}$. Для описания алгоритма нам потребуются три матрицы. Пусть, во-первых, A_H — матрица модельной задачи, дискретизованной с крупным шагом H. Во-вторых, нам нужен *оператор сужения* R, который бы преобразовывал невязку с мелкой сетки в функцию на грубой сетке. По существу, это тот же оператор, что и в многосеточном методе (см. разд. 6.9.2). Наконец, нужен *оператор интерполяции* для интерполирования значений с грубой сетки на мелкую. Как и в многосеточном методе, таким оператором оказывается R^T.

Алгоритм 6.20. *Двухуровневый аддитивный метод Шварца для пересчета приближенного решения x_i системы $Ax = b$ в более точное приближение x_{i+1}:*

$x_{i+1} = x_i$
for $j = 1$ *to число областей* Ω_k

$r_{\Omega_j} = (b - Ax_j)_{\Omega_j}$
$x_{i+1,\Omega_j} = x_{i+1,\Omega_j} + A^{-1}_{\Omega_j,\Omega_j} \cdot r_{\Omega_j}$
end for
$x_{i+1} = x_{i+1} + R^T A_H^{-1} Rr$

Как и алгоритм 6.18, этот алгоритм обычно используется в качестве предобуславливателя для методов крыловского подпространства.

Согласно теории сходимости этого алгоритма, применимой к более общим задачам, чем уравнение Пуассона, число итераций, необходимых для сходимости, не зависит от H, h и δ, когда все три величины стремятся к нулю так, что отношение δ/H остается неизменным. Это означает, что сложность алгоритма будет столь же низкой, как и у многосеточного метода, если работа при решении подзадач с матрицами $A^{-1}_{\Omega_i,\Omega_i}$ и A_H^{-1} пропорциональна числу неизвестных.

Читателю, вероятно, ясно, что реализация подобных алгоритмов для реальных задач является непростым делом. Имеется комплекс программ, доступный в Интернете, реализующий многие из описанных здесь «строительных блоков»; программы способны работать и на параллельных компьютерах. Комплекс называется PETSc (сокращение от Portable Extensible Toolkit for Scientific computing). Он может быть найден по адресу http://www.mcs.anl.gov/petsc/petsc.html, а краткое его описание дано в [232].

6.11. Литература и смежные вопросы к главе 6

Обзоры современных итерационных методов, доведенные до новейших достижений, даны в [15, 107, 136, 214], а обзор их параллельных реализаций — в [76]. Классические методы типа методов Якоби, Гаусса—Зейделя и SOR подробно описаны в [249, 137]. По поводу многосеточных методов см. [43, 185, 186, 260, 268], а также литературу, упоминаемую в этих публикациях; на Интернет-сайте [91] можно найти ссылки на обширную библиографию, программы и. т. п. Методы декомпозиции области обсуждаются в [49, 116, 205, 232]. Описание чебышевских и родственных многочленов дано в [240]. Изложение алгоритма FFT содержится в любом хорошем учебнике по теории вычислений; см., например, [3] и [248]. Стабилизированный вариант блочной циклической редукции можно найти в [47, 46].

6.12. Вопросы к главе 6

Вопрос 6.1 (*легкий*). Доказать лемму 6.1.

Вопрос 6.2 (*легкий*). Доказать приводимые ниже формулы для треугольных разложений матрицы T_N:

1. Разложение Холесского $T_N = B_N^T B_N$ имеет множитель Холесского B_N верхнего двухдиагонального вида с элементами

$$B_N(i,i) = \sqrt{\frac{i+1}{i}} \quad \text{и} \quad B_N(i,i+1) = \sqrt{\frac{i}{i+1}}.$$

2. Гауссово исключение с частичным выбором ведущего элемента дает для T_N разложение $T_N = L_N U_N$, где оба треугольных множителя — двухдиагональные:

$$L_N(i,i) = 1 \quad \text{и} \quad L_N(i+1,i) = -\frac{i}{i+1},$$
$$U_N(i,i) = \frac{i+1}{i} \quad \text{и} \quad U_N(i,i+1) = -1.$$

3. $T_N = D_N D_N^T$, где D_N — верхняя двухдиагональная матрица размера $N \times (N+1)$ с 1 на главной диагонали и -1 на первой наддиагонали.

Вопрос 6.3 (*легкий*). Проверить равенство (6.13).

Вопрос 6.4 (*легкий*).

1. Доказать лемму 6.2.
2. Доказать лемму 6.3.
3. Доказать, что уравнение Сильвестра $AX - XB = C$ эквивалентно системе линейных уравнений $(I_n \otimes A - B^T \otimes I_m)\,\mathrm{vec}(X) = \mathrm{vec}(C)$.
4. Доказать, что $\mathrm{vec}(AXB) = (B^T \otimes A) \cdot \mathrm{vec}(X)$.

Вопрос 6.5 (*средней трудности*). Пусть $A^{n\times n}$ — диагонализуемая матрица, т. е. A имеет n линейно независимых собственных векторов: $Ax_i = \alpha_i x_i$, или $AX = X\Lambda_A$, где $X = [x_1, \ldots, x_n]$ и $\Lambda_A = \mathrm{diag}(\alpha_i)$. Точно так же, пусть $B^{m\times m}$ — диагонализуемая матрица, так что B имеет m линейно независимых собственных векторов: $By_i = \beta_i y_i$, или $BY = Y\Lambda_B$, где $Y = [y_1, \ldots, y_m]$ и $\Lambda_B = \mathrm{diag}(\beta_j)$. Доказать следующие утверждения:

1. Собственными значениями матрицы $(I_m \otimes A + B \otimes I_n)$ являются mn чисел $\lambda_{ij} = \alpha_i + \beta_j$, т. е. все возможные парные суммы собственных значений матриц A и B. Соответствующие собственные векторы имеют вид $z_{ij} = x_i \otimes y_j$, где $(km+l)$-я компонента равна $x_i(k)y_j(l)$. Это же можно записать матричным соотношением
$$(I_m \otimes A + B \otimes I_n)(Y \otimes X) = (Y \otimes X) \cdot (I_m \otimes \Lambda_A + \Lambda_B \otimes I_n). \qquad (6.63)$$
2. Уравнение Сильвестра $AX + XB^T = C$ невырожденно (т. е. разрешимо относительно X при любой матрице C) тогда и только тогда, когда $\alpha_i + \beta_j \neq 0$ для всех собственных значений α_i матрицы A и β_j матрицы B. То же самое справедливо для несколько иного уравнения Сильвестра $AX + XB = C$ (см. также вопрос 4.6).
3. Собственными значениями матрицы $A \otimes B$ являются mn чисел $\lambda_{ij} = \alpha_i\beta_j$, т. е. все возможные парные произведения собственных значений матриц A и B. Соответствующие собственные векторы имеют вид $z_{ij} = x_i \otimes y_j$, где $(km+l)$-я компонента равна $x_i(k)y_j(l)$. Это же можно записать матричным соотношением
$$(B \otimes A)(Y \otimes X) = (Y \otimes X) \cdot (\Lambda_B \otimes \Lambda_A). \qquad (6.64)$$

Вопрос 6.6 (*легкий; программирование*). Написать однострочную Matlab-программу, реализующую алгоритм 6.2, т. е. шаг алгоритма Якоби для уравнения Пуассона. Проверьте ее, убедившись, что скорость сходимости алгоритма соответствует предсказанию в разд. 6.5.4.

Вопрос 6.7 (*трудный*). Доказать лемму 6.7.

Вопрос 6.8 (*средней трудности; программирование*). Написать Matlab-программу, решающую дискретную модельную задачу на квадрате посредством алгоритма FFT. Входными параметрами программы должны быть порядок N и $N \times N$-матрица значений f_{ij}. Выходными параметрами должны быть $N \times N$-матрица решения v_{ij} и невязка $\|T_{N\times N}v - h^2 f\|_2/(\|T_{N\times N}\|_2 \cdot \|v\|_2)$. Нужно также получить трехмерные графики функций f и v. Используйте процедуру FFT, встроенную в Matlab. Если использовать все возможности, предоставляемые Matlab'ом, то длина вашей программы не должна превышать нескольких строк. Решите с ее помощью несколько задач как с известным, так и неизвестным вам решением:

1. $f_{jk} = \sin(j\pi/(N+1)) \cdot \sin(k\pi/(N+1))$.
2. $f_{jk} = \sin(j\pi/(N+1)) \cdot \sin(k\pi/(N+1)) + \sin(3j\pi/(N+1)) \cdot \sin(5k\pi/(N+1))$.
3. f имеет несколько острых пиков (как в положительную, так и отрицательную сторону), а в остальном равна нулю. Такую функцию можно рассматривать как аппроксимацию электростатического потенциала системы заряженных частиц, помещенных у оснований пиков, заряды которых пропорциональны (положительным или отрицательным) высотам пиков. Если все пики направлены в положительную сторону, то функцию можно интерпретировать и как аппроксимацию гравитационного потенциала.

Вопрос 6.9 (*средней трудности*). Проверить, что последовательное выполнение справа налево матрично-векторных произведений в формуле (6.47) математически эквивалентно алгоритму 6.13.

Вопрос 6.10 (*трудный; средней трудности*).

1 (*трудный*). Пусть вещественные симметричные матрицы A и H *перестановочны*, т. е. $AH = HA$. Доказать, что найдется ортогональная матрица Q, такая, что обе матрицы QAQ^T и QHQ^T — диагональные: $QAQ^T = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$, $QHQ^T = \mathrm{diag}(\theta_1, \ldots, \theta_n)$. Иными словами, A и H имеют одни и те же собственные векторы. Указание: предположите вначале, что все собственные значения матрицы A различны, а затем устраните это предположение.

2 (*средней трудности*). Рассмотрим симметричную трехдиагональную теплицеву матрицу

$$\widehat{T} = \begin{bmatrix} \alpha & \theta & & \\ \theta & \ddots & \ddots & \\ & \ddots & \ddots & \theta \\ & & \theta & \alpha \end{bmatrix},$$

т. е. симметричную трехдиагональную матрицу с константой α на главной диагонали и константой θ на двух соседних диагоналях. Выписать *простые* формулы для собственных значений и собственных векторов матрицы $\widehat{T}$. Указание: воспользоваться леммой 6.1.

3 (*трудный*). Рассмотрим $n^2 \times n^2$-матрицу

$$T = \begin{bmatrix} A & H & & \\ H & \ddots & \ddots & \\ & \ddots & \ddots & H \\ & & H & A \end{bmatrix}$$

блочно-трехдиагонального вида с n копиями матрицы A вдоль главной диагонали. Пусть $QAQ^T = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ и $QHQ^T = \mathrm{diag}(\theta_1, \ldots, \theta_n)$ — указанные выше спектральные разложения матриц A и H. Выписать *простые* формулы для n^2 собственных значений и собственных векторов матрицы T как функций от чисел α_i и θ_i и матрицы Q.

4 (*средней трудности*). Указать способ решения системы $Tx = b$ за время $O(n^3)$. Насколько больше было бы время при решении системы посредством плотного или ленточного LU-разложения?

5 (*средней трудности*). Пусть A и H — (возможно, различные) симметричные трехдиагональные ттплицевы матрицы (см. определение выше). Используя алгоритм FFT, найти способ решения системы $Tx = b$ за время $O(n^2 \log n)$.

Вопрос 6.11 (*легкий*). Пусть R — невырожденная верхняя треугольная матрица. Проверить, что для верхней хессенберговой матрицы C матрица RCR^{-1} также верхняя хессенбергова.

Вопрос 6.12 (*средней трудности*). Проверить, что крыловское подпространство $\mathcal{K}_k(A, y_1)$ тогда и только тогда имеет размерность k, когда при вычислении вектора q_k не происходит досрочного выхода из алгоритма Арнольди (алгоритм 6.9) или алгоритма Ланцоша (алгоритм 6.10).

Вопрос 6.13 (*средней трудности*). Пусть $A^{n\times n}$ — симметричная положительно определенная матрица и пусть матрица $Q^{n\times k}$ имеет полный столбцевой ранг. Проверить, что матрица $T = Q^T AQ$ также симметричная и положительно определенная. (Здесь матрица Q не обязана иметь ортонормированные столбцы.)

Вопрос 6.14 (*средней трудности*). Доказать теорему 6.9.

Вопрос 6.15 [(*средней трудности; трудный*)].

1 (*средней трудности*). Проверить равенство (6.58).
2 (*средней трудности*). Проверить равенство (6.60).
3 (*трудный*). Доказать теорему 6.11.

Вопрос 6.16 (*средней трудности; программирование*). На Интернет-странице HOMEPAGE/Matlab/MG_README.html имеется Matlab-программа, реализующая решение дискретной модельной задачи на квадрате многосеточным методом. Начните работу с ней запуском демонстрационной программы (для чего введите команду «makemgdemo», а затем «testfmgv»). После этого проверьте, как работает testfmgv для различных правых частей (входной массив b), различного числа взвешенных итераций Якоби до и после каждого рекурсивного обращения к многосеточному решателю (входные параметры jac1 и jac2) и различного числа итераций (входной параметр iter). Программа выдаст график

скорости сходимости (т. е. отношения последовательных невязок). Зависит ли эта скорость от величины b? от частоты осцилляций в b? значений параметров jac1 и jac2? Для каких значений jac1 и jac2 задача решается наиболее эффективно?

Вопрос 6.17 (*средней трудности; программирование*). Используя быстрый решатель модельной задачи из вопроса 6.8 или вопроса 6.16, применить метод декомпозиции области к построению быстрого решателя для уравнения Пуассона на L-образной области, как это описано в разд. 6.10. Большой квадрат должен иметь размер 1×1, а малый — размер 0.5×0.5; при этом малый квадрат примыкает к нижней половине правой стороны большого квадрата. Вычислить невязку, чтобы убедиться в правильности ответа.

Вопрос 6.18 (*трудный*). Составить таблицу типа табл. 6.1, но не для двумерного, а для трехмерного уравнения Пуассона. Считать, что сетка неизвестных имеет размер $N \times N \times N$ и $n = N^3$. Заполнить как можно больше позиций во втором и третьем столбцах таблицы.

Глава 7

Итерационные методы для задач на собственные значения

7.1. Введение

В этой главе обсуждаются итерационные методы вычисления собственных значений для матриц столь больших, что к ним нельзя применить прямые методы из гл. 4 и 5. Иными словами, нужны алгоритмы, которые требуют памяти, меньшей чем $O(n^2)$ машинных слов, и работы, меньшей чем $O(n^3)$ флопов. Так как для хранения всех собственных векторов почти любой $n \times n$-матрицы необходимо иметь n^2 машинных слов, то наши требования к алгоритму означают, что он будет вычислять лишь несколько собственных значений, каким-либо образом выделенных пользователем, и соответствующих собственных векторов.

Нам понадобятся сведения о крыловских подпространствах (см. разд. 6.6), симметричной проблеме собственных значений (разд. 5.2), а также о степенном методе и обратной итерации (разд. 5.3). Рекомендуем читателю вначале просмотреть названные разделы.

Простейшая спектральная задача состоит в вычислении собственного значения с наибольшим модулем и соответствующего ему собственного вектора. Среди методов, решающих эту задачу, простейшим является степенной метод (алгоритм 4.1). Напомним, что внутренний цикл метода имеет вид

$$\begin{aligned} y_{i+1} &= Ax_i, \\ x_{i+1} &= y_{i+1}/\|y_{i+1}\|_2. \end{aligned}$$

Последовательность $\{x_i\}$ сходится к собственному вектору, отвечающему искомому собственному значению (при условии, что имеется только одно собственное значение с наибольшим модулем и x_1 не лежит в инвариантном подпространстве, где требуемый собственный вектор не представлен). Заметим, что матрица A используется в алгоритме только для выполнения матрично-векторных умножений. Поэтому все, что нужно для алгоритма, — это «черный ящик», на вход которого подается x_i, а на выходе появляется Ax_i (см. пример 6.13).

С рассмотренной задачей тесно связана задача вычисления собственного значения, ближайшего к задаваемому пользователем числу σ, и соответствующего собственного вектора. Именно для такой ситуации предназначена обрат-

ная итерация (алгоритм 4.2). Напомним, что ее внутренний цикл имеет вид

$$y_{i+1} = (A - \sigma I)^{-1} x_i,$$
$$x_{i+1} = y_{i+1} / \|y_{i+1}\|_2,$$

т. е. предусматривает решение системы линейных уравнений с матрицей коэффициентов $A - \sigma I$. Снова последовательность $\{x_i\}$ сходится к нужному собственному вектору при условии, что имеется лишь одно собственное значение, ближайшее к σ (и x_1 удовлетворяет сформулированному выше условию). Для вычисления вектора y_{i+1} можно использовать любой из методов для разреженных матриц, описанных в гл. 6 или разд. 2.7.4, хотя обычно это стоит много дороже, чем простое умножение матрицы A на вектор. Если A — симметричная матрица, то для ускорения сходимости можно применить RQ-итерацию (алгоритм 5.1); правда, не всегда можно гарантировать, что она сойдется именно к собственному значению, ближайшему к числу σ.

В результате $k-1$ шагов степенного метода или обратной итерации, начатых с заданного вектора x_1, будет получена последовательность векторов $x_1, x_2, \ldots, x_k$. На эти векторы натянуто *крыловское подпространство*, определенное в разд. 6.6.1. Для степенного метода крыловское подпространство имеет вид $\mathcal{K}_k(A, x_1) = \mathrm{span}[x_1, Ax_1, A^2x_1, \ldots, A^{k-1}x_1]$, а в случае обратной итерации это будет подпространство $\mathcal{K}_k((A-\sigma I)^{-1}, x_1)$. Естественно спросить, не правильней ли было бы, вместо того чтобы брать x_k в качестве приближенного собственного вектора, поискать «наилучшее» приближение к собственному вектору во всем подпространстве $\mathcal{K}_k$, т. е. искать наилучшую линейную комбинацию $\sum_{i=1}^{k} \alpha_i x_i$. Тот же принцип был применен в разд. 6.6.2 к решению системы $Ax = b$, при этом в $\mathcal{K}_k$ разыскивалось наилучшее приближенное решение системы. Мы увидим, что наилучшее приближение к собственному вектору, выбранное из $\mathcal{K}_k$, имеет значительно лучшую точность, чем вектор x_k (и то же самое верно для приближений к собственному значению). Поскольку подпространство $\mathcal{K}_k$ в общем случае k-мерно, мы можем с его помощью вычислить k наилучших приближений к собственным значениям и собственным векторам. Эти наилучшие приближения называются *числами* и *векторами Ритца*.

Мы сосредоточимся на случае симметричной матрицы A. Краткое обсуждение несимметричного случая проводится в последнем разделе.

Остальная часть данной главы организована следующим образом. В разд. 7.2 обсуждается метод Рэлея—Ритца, наше основное средство для извлечения информации о собственных значениях и собственных векторах из крыловского подпространства. В разд. 7.3 описан наш главный метод, а именно алгоритм Ланцоша, для случая точной арифметики. В разд. 7.4 дан анализ существенно иного поведения алгоритма Ланцоша в арифметике с плавающей точкой. В разд. 7.5 и 7.6 описаны практические реализации метода Ланцоша, вычисляющие результаты хорошего качества, несмотря на округления. Наконец, в разд. 7.7 дано краткое обсуждение алгоритмов для несимметричной проблемы собственных значений.

7.2. Метод Рэлея—Ритца

Пусть $Q = [Q_k, Q_u]$ — произвольная ортогональная матрица порядка n, причем Q_k и Q_u имеют соответственно размеры $n \times k$ и $n \times (n-k)$. На практике

столбцы матрицы Q_k вычисляются алгоритмом Ланцоша (см. алгоритм 6.10 или алгоритм 7.1); на них натянуто крыловское подпространство $\mathcal{K}_k$. Индекс u указывает, что подматрица Q_u (большей частью) неизвестна (u — от английского «unknown»). Однако пока нам совершенно не важно происхождение матрицы Q.

Мы будем пользоваться следующими обозначениями (они использовались и в формуле (6.31)):

$$T = Q^T A Q = [Q_k, Q_u]^T A [Q_k, Q_u] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix}$$

$$\equiv \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} T_k & T_{uk} \\ T_{ku} & T_u \end{pmatrix} \end{array}$$

$$= \begin{bmatrix} T_k & T_{ku}^T \\ T_{ku} & T_u \end{bmatrix}. \tag{7.1}$$

При $k = 1$ T_k — это попросту отношение Рэлея: $T_1 = \rho(Q_1, A)$ (см. определение 5.1). Таким образом, T_k есть естественное обобщение отношения Рэлея на случай $k > 1$.

Определение 7.1. Процедура Рэлея—Ритца *заключается в интерпретации собственных значений матрицы $T_k = Q_k^T A Q_k$ как приближений к собственным значениям матрицы A. Эти приближения называются* числами Ритца. *Пусть $T_k = V \Lambda V^T$ есть спектральное разложение матрицы T_k. Столбцы матрицы $Q_k V$ рассматриваются как приближения к соответствующим собственным векторам и называются* векторами Ритца.

Числа и векторы Ритца считаются *оптимальными* приближениями к собственным значениям и собственным векторам матрицы A по нескольким причинам. Во-первых, когда известны лишь Q_k и T_k, а Q_u не известна, а потому не известны T_{ku} и T_u, то числа и векторы Ритца суть естественные приближения, которые можно извлечь из известной части матрицы. Во-вторых, они удовлетворяют приводимому ниже обобщению теоремы 5.5. (Теорема 5.5 показывала, что отношение Рэлея есть «наилучшее приближение» к одному собственному значению.) Вспомним, что столбцы матрицы Q_k тогда и только тогда определяют инвариантное подпространство для A, когда для некоторой матрицы R выполняется соотношение $AQ_k = Q_k R$.

Теорема 7.1. *Минимум величины $\|AQ_k - Q_k R\|_2$ по всем симметричным $k \times k$-матрицам R достигается при $R = T_k$; при этом $\|AQ_k - Q_k R\|_2 = \|T_{ku}\|_2$. Пусть $T_k = V \Lambda V^T$ — спектральное разложение матрицы T_k. Минимум величины $\|AP_k - P_k D\|_2$, когда P_k пробегает множество $n \times k$-матриц с ортонормированными столбцами, таких, что $\operatorname{span}(P_k) = \operatorname{span}(Q_k)$, а D пробегает множество диагональных $k \times k$-матриц, также равен $\|T_{ku}\|_2$ и достигается для $P_k = Q_k V$ и $D = \Lambda$.*

Иными словами, столбцы матрицы $Q_k V$ (т. е. векторы Ритца) суть «наилучшие» приближенные собственные векторы, а диагональные элементы матрицы Λ (числа Ритца) суть «наилучшие» приближенные собственные значения в том смысле, что они минимизируют норму невязки $\|AP_k - P_k D\|_2$.

Доказательство. Чтобы упростить записи, временно опустим индекс k у матриц T_k и Q_k и будем писать $T = Q^T AQ$. Положим $R = T + Z$. Мы хотим показать, что минимум величины $\|AQ - QR\|_2^2$ достигается при $Z = 0$. Это будет достигнуто применением теоремы Пифагора в замаскированной форме:

$$\begin{aligned}
\|AQ - QR\|_2^2 &= \lambda_{\max}[(AQ - QR)^T(AQ - QR)] \\
&\qquad \text{согласно утверждению 7 леммы 1.7} \\
&= \lambda_{\max}[(AQ - Q(T+Z))^T(AQ - Q(T+Z))] \\
&= \lambda_{\max}[(AQ - QT)^T(AQ - QT) - (AQ - QT)^T(QZ) \\
&\quad - (QZ)^T(AQ - QT) + (QZ)^T(QZ)] \\
&= \lambda_{\max}[(AQ - QT)^T(AQ - QT) - (Q^TAQ - T)Z \\
&\quad - Z^T(Q^TAQ - T) + Z^TZ] \\
&= \lambda_{\max}[(AQ - QT)^T(AQ - QT) + Z^TZ] \\
&\qquad \text{поскольку } Q^TAQ = T \\
&\geq \lambda_{\max}[(AQ - QT)^T(AQ - QT)] \\
&\qquad \text{поскольку } Z^TZ \text{ — симметричная положительно} \\
&\qquad \text{полуопределенная матрица (см. вопрос 5.5)} \\
&= \|AQ - QT\|_2^2 \quad \text{согласно утверждению 7 леммы 1.7.}
\end{aligned}$$

Легко определить само значение минимума. Восстанавливая индексы, имеем

$$\|AQ_k - Q_kT_k\|_2 = \|(Q_kT_k + Q_uT_{ku}) - (Q_kT_k)\|_2 = \|Q_uT_{ku}\|_2 = \|T_{ku}\|_2.$$

Если заменить Q_k произведением Q_kU, где U — произвольная ортогональная матрица порядка k, то столбцы матриц Q_k и Q_kU порождают одно и то же подпространство и справедливы равенства

$$\|AQ_k - Q_kR\|_2 = \|AQ_kU - Q_kRU\|_2 = \|A(Q_kU) - (Q_kU)(U^TRU)\|_2.$$

Эти величины по-прежнему достигают минимума при $R = T_k$. Выбирая $U = V$ так, чтобы матрица U^TT_kU стала диагональной, мы решим вторую задачу минимизации в формулировке теоремы. □

Эта теорема обосновывает использование чисел Ритца как приближений к собственным значениям. Предположим, что матрица Q_k вычислена алгоритмом Ланцоша. В этом случае (см. (6.31))

$$T = \left[\begin{array}{c|c} T_k & T_{ku}^T \\ \hline T_{ku} & T_u \end{array}\right] = \left[\begin{array}{ccccc|ccccc}
\alpha_1 & \beta_1 & & & & & & & & \\
\beta_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & \ddots & \beta_{k-1} & & & & & & \\
 & & \beta_{k-1} & \alpha_k & \beta_k & & & & & \\
\hline
 & & & \beta_k & \alpha_{k+1} & \beta_{k+1} & & & & \\
 & & & & \beta_{k+1} & \ddots & \ddots & & & \\
 & & & & & \ddots & \ddots & \beta_{n-1} & & \\
 & & & & & & \beta_{n-1} & \alpha_n & &
\end{array}\right]$$

и все величины, упомянутые в теореме 7.1, легко находятся. Действительно, существуют хорошие алгоритмы для вычисления собственных значений и собственных векторов симметричной трехдиагональной матрицы T_k (см. разд. 5.3), а норма невязки $||T_{ku}||_2$ есть просто число β_k. (Из описания алгоритма Ланцоша известно, что число β_k неотрицательно.) Все это облегчает использование оценок для погрешностей в приближенных собственных значениях и собственных векторах, содержащихся в следующем утверждении.

Теорема 7.2. *Пусть T_k, T_{ku} и Q_k — те же матрицы, что и в формуле (7.1). Пусть $T_k = V\Lambda V^T$ — спектральное разложение матрицы T_k, где $V = [v_1, \ldots, v_k]$ — ортогональная матрица, а $\Lambda = \mathrm{diag}(\theta_1, \ldots, \theta_k)$. Тогда:*

1. *Найдутся k (необязательно наибольших) собственных значений $\alpha_1, \ldots, \alpha_k$ матрицы A, такие, что $|\theta_i - \alpha_i| \le ||T_{ku}||_2$ для $i = 1, \ldots, k$. Если матрица Q_k вычислена алгоритмом Ланцоша, то $|\theta_i - \alpha_i| \le ||T_{ku}||_2 = \beta_k$, где β_k — единственный элемент блока T_{ku}, который (может быть) отличен от нуля; он находится в правом верхнем углу этого блока.*
2. *$||A(Q_k v_i) - (Q_k v_i)\theta_i||_2 = ||T_{ku} v_i||_2$. Таким образом, разность между числом Ритца θ_i и некоторым собственным значением α матрицы A не превышает величины $||T_{ku} v_i||_2$, которая может быть много меньше, чем $||T_{ku}||_2$. Если матрица Q_k вычислена алгоритмом Ланцоша, то $||T_{ku} v_i||_2 = \beta_k |v_i(k)|$, где $v_i(k)$ — последняя (k-я) компонента вектора v_i. Эта формула дает дешевый способ вычисления нормы невязки $||A(Q_k v_i) - (Q_k v_i)\theta_i||_2$, не требующий умножения какого-либо вектора на матрицы Q_k или A.*
3. *Не имея информации о спектре матрицы T_u, нельзя дать какую-либо содержательную оценку для погрешности в векторе Ритца $Q_k v_i$. Если известно, что θ_i отделено расстоянием, не меньшим g, от прочих собственных значений матриц T_k и T_u, то угол θ между $Q_k v_i$ и точным собственным вектором матрицы A можно оценить так:*

$$\frac{1}{2} \sin 2\theta \le \frac{||T_{ku}||_2}{g}. \tag{7.2}$$

Если матрица Q_k вычислена алгоритмом Ланцоша, то эта оценка упрощается к виду

$$\frac{1}{2} \sin 2\theta \le \frac{\beta_k}{g}.$$

Доказательство.

1. Числа $\theta_1, \ldots, \theta_k$ входят в множество собственных значений матрицы $\widehat{T} = \begin{bmatrix} T_k & 0 \\ 0 & T_u \end{bmatrix}$. Так как

$$||\widehat{T} - T||_2 = \left|\left| \begin{bmatrix} 0 & T_{ku}^T \\ T_{ku} & 0 \end{bmatrix} \right|\right|_2 = ||T_{ku}||_2,$$

то из теоремы Вейля (теоремы 5.1) следует, что собственные значения матриц $\widehat{T}$ и T разнятся не более чем на $||T_{ku}||_2$. Нужный результат следует из того, что собственные значения матриц T и A одинаковы.

2. Имеем

$$\begin{aligned}\|A(Q_k v_i) - (Q_k v_i)\theta_i\|_2 &= \|Q^T A(Q_k v_i) - Q^T(Q_k v_i)\theta_i\|_2 \\ &= \left\| \begin{bmatrix} T_k v_i \\ T_{ku} v_i \end{bmatrix} - \begin{bmatrix} v_i\theta_i \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} 0 \\ T_{ku} v_i \end{bmatrix} \right\|_2 \\ &\quad \text{так как } T_k v_i = \theta_i v_i \\ &= \|T_{ku} v_i\|_2.\end{aligned}$$

   По теореме 5.5, матрица A должна иметь собственное значение α, такое, что $|\alpha - \theta_i| \le \|T_{ku} v_i\|_2$. Если матрица Q_k вычислена алгоритмом Ланцоша, то $\|T_{ku} v_i\|_2 = \beta_k |v_i(k)|$, потому что в T_{ku} может быть отличен от нуля только элемент β_k, стоящий в правом верхнем углу.

3. Мы обратимся еще раз к примеру 5.4, чтобы показать, что нельзя получить содержательной оценки для погрешности в векторе Ритца, не имея информации о спектре матрицы T_u. Пусть

$$T = \begin{bmatrix} 1+g & \epsilon \\ \epsilon & 1 \end{bmatrix},$$

   где $0 < \epsilon < g$. Положим $k = 1$ и $Q_1 = [e_1]$, тогда $T_1 = 1+g$, и приближенным собственным вектором является сам вектор e_1. В примере 5.4 было показано, что собственные векторы матрицы T близки к $[1, \epsilon/g]^T$ и $[-\epsilon/g, 1]^T$. Поэтому без нижней границы для g, т. е. для отделенности данного собственного значения матрицы T_k от всех прочих собственных значений, включая и собственные значения матрицы T_u, нельзя оценить погрешность в вычисленном собственном векторе. Если же такая нижняя граница имеется, то к матрицам T и $T + E = \operatorname{diag}(T_k, T_u)$ можно применить вторую оценку теоремы 5.4; в результате получится неравенство (7.2). ◇

7.3. Алгоритм Ланцоша в точной арифметике

Алгоритм Ланцоша для вычисления собственных значений симметричной матрицы A соединяет метод Ланцоша для построения крыловского подпространства (алгоритм 6.10) с процедурой Рэлея—Ритца, описанной в предыдущем разделе. Иными словами, из ортонормированных векторов Ланцоша строится матрица $Q_k = [q_1, \ldots, q_k]$, и в качестве приближенных собственных значений матрицы A принимаются числа Ритца, т. е. (см. (7.1)) собственные значения симметричной трехдиагональной матрицы $T_k = Q_k^T A Q_k$.

Алгоритм 7.1. *Алгоритм Ланцоша для вычисления собственных значений и собственных векторов матрицы $A = A^T$ в точной арифметике:*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
  $\beta_j = \|z\|_2$
  если $\beta_j = 0$, *то прекратить выполнение алгоритма*

$q_{j+1} = z/\beta_j$
Вычислить собственные значения и собственные векторы матрицы T_j *и оценки погрешностей в них*
end for

В этом разделе мы изучим характер сходимости алгоритма Ланцоша с помощью подробного разбора численного примера. Пример был сконструирован таким образом, чтобы проиллюстрировать как типичную картину сходимости, так и некоторые отклонения от нее, которые мы называем *ложной сходимостью*. Ложная сходимость может иметь место вследствие того, что начальный вектор q_1 почти ортогонален собственному вектору для желаемого собственного значения, или потому, что среди собственных значений матрицы есть кратные (или очень близкие).

Название данного раздела указывает, что влияние ошибок округления в нашем примере было (почти полностью) исключено. Разумеется, Matlab-программа, использованная для примера, выполнялась в арифметике с плавающей точкой (эта программа помещена на HOMEPAGE/Matlab/LanczosFullReorthog.m). Однако алгоритм Ланцоша в ней (в частности, внутренний цикл алгоритма 7.1) был реализован особо тщательным (и дорогостоящим) образом, чтобы как можно ближе воспроизвести точные результаты. Эта тщательная реализация называется *алгоритмом Ланцоша с полной переортогонализацией*. Мы используем это название в надписях на рисунках, сопровождающих данный пример.

В следующем разделе мы исследуем этот же самый пример, но для исходной необременительной реализации алгоритма 7.1. Мы называем ее *алгоритмом Ланцоша без переортогонализации* по контрасту с *алгоритмом с полной переортогонализацией*. (Будет также объяснено различие двух реализаций.) Мы увидим, что поведение исходного алгоритма Ланцоша может значительно отличаться от поведения более дорогостоящего «точного» алгоритма. Тем не менее, мы укажем способ надежного вычисления собственных значений с помощью более дешевого алгоритма.

Пример 7.1. Мы проиллюстрируем алгоритм Ланцоша и оценивание погрешностей в нем на примере большой диагональной матрицы A порядка 1000. Большинство ее собственных значений выбрано случайно из нормального гауссовского распределения. На рис. 7.1 дана картина собственных значений. Чтобы упростить понимание последующих графиков, диагональные элементы матрицы A были упорядочены так, чтобы с ростом индекса происходило их убывание. Элементы a_{ii} суть собственные значения $\lambda_i(A)$, а соответствующими собственными векторами являются столбцы e_i единичной матрицы. Имеется небольшое число собственных значений с наибольшими абсолютными величинами; все остальные группируются вблизи центра спектра. В начальном векторе Ланцоша q_1 все компоненты, кроме одной, равны; об этом мы скажем ниже.

Экспериментируя с диагональной матрицей, мы не теряем общности: проведение алгоритма Ланцоша для матрицы A при начальном векторе q_1 эквивалентно проведению алгоритма для $Q^T A Q$ при начальном векторе $Q^T q_1$ (см. вопрос 7.1).

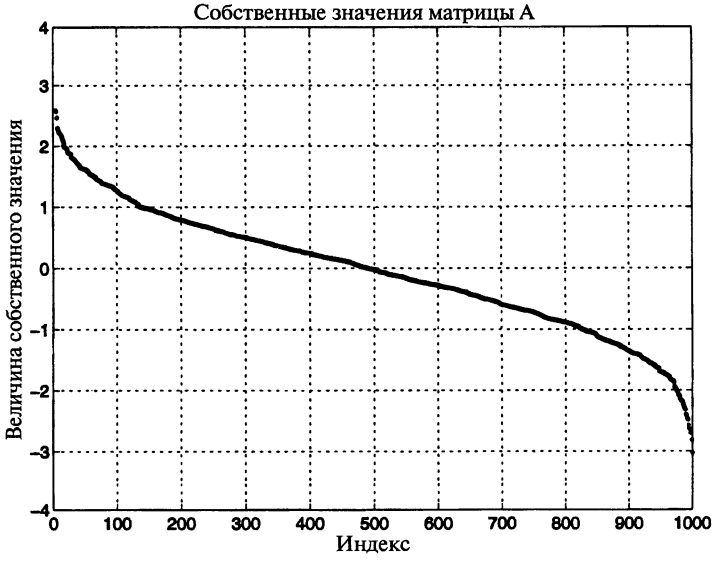


Рис. 7.1. Собственные значения диагональной матрицы A.

Чтобы лучше выявить характер сходимости, мы приводим несколько диаграмм типа тех, что показаны на рис. 7.2. На этом рисунке собственные значения каждой матрицы T_k помещены в соответствующий столбец k. Верхняя диаграмма соответствует значениям k от 1 до 9, а нижняя — значениям k от 1 до 29. Столбец k содержит k плюсов, по одному для каждого собственного значения. В дополнительном столбце в правой части диаграммы помещены собственные значения матрицы A. Мы используем такой символьный код: наибольшее и наименьшее собственные значения каждой матрицы T_k указаны крестиками, вторые по величине собственные значения (с обоих концов спектра) — квадратиками, третьи по величине — звездочками и, наконец, четвертые по величине — кружочками. Далее эти обозначения циклически повторяются по направлению к середине спектра.

Для лучшего понимания сходимости рассмотрим поведение старших собственных значений матриц T_k; они указаны крестиками вверху каждого столбца. Видно, что с ростом k старшие собственные значения возрастают. Это следствие теоремы Коши о перемежаемости: ведь T_k есть подматрица матрицы T_{k+1} (см. вопрос 5.4). В действительности, теорема Коши утверждает большее, а именно, что все собственные значения матрицы T_{k+1} *перемежаются* собственными значениями подматрицы T_k, т. е. $\lambda_i(T_{k+1}) \geq \lambda_i(T_k) \geq \lambda_{i+1}(T_{k+1}) \geq \lambda_{i+1}(T_k)$. Иначе говоря, $\lambda_i(T_k)$ монотонно возрастает с ростом k для любого фиксированного значения i, а не только для $i = 1$ (что соответствует старшему собственному значению). Это утверждение иллюстрируется последовательностями символов одного типа, направленными вправо и вверх.

Совершенно аналогично поведение младших собственных значений: нижний крестик в каждом столбце на рис. 7.2 указывает наименьшее собственное

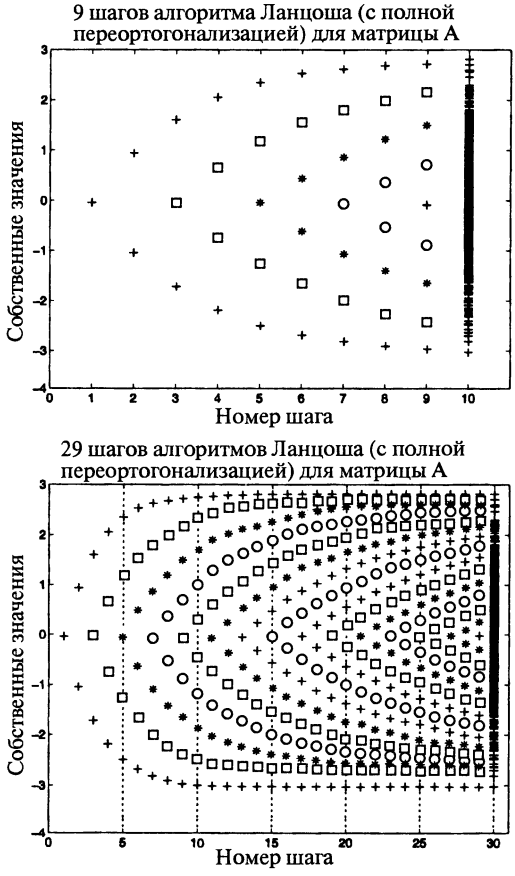


Рис. 7.2. Алгоритм Ланцоша в применении к матрице A. На верхней диаграмме представлены первые 9 шагов алгоритма, а на нижней — первые 29 шагов. В столбце k указаны собственные значения матрицы T_k, а в самом правом столбце (10-м столбце верхней диаграммы и 30-м столбце нижней диаграммы) — собственные значения матрицы A.

значение соответствующей матрицы T_k, и эти собственные значения монотонно убывают с ростом k. Точно так же монотонно убывает i-е собственное значение, считая от левого конца спектра. Это тоже простое следствие теоремы Коши о перемежаемости.

Мы можем задаться вопросом, к какому собственному значению матрицы A сходится собственное значение $\lambda_i(T_k)$ с ростом k. Ясно, что наибольшее собственное значение $\lambda_1(T_k)$ должно сходиться к наибольшему собственному значению $\lambda_1(A)$. Действительно, когда алгоритм Ланцоша дойдет до шага $k = n$ (в предположении, что он не остановится раньше из-за того, что какое-то β_k равно нулю), то будет получена матрица T_n, подобная матрице A, поэтому $\lambda_1(T_n) = \lambda_1(A)$. Аналогично, i-е по старшинству собственное значение $\lambda_i(T_k)$,

монотонно возрастая, должно сойтись к i-му по старшинству собственному значению $\lambda_i(A)$ матрицы A (если не произойдет преждевременной остановки алгоритма Ланцоша), а собственное значение $\lambda_{k+1-i}(T_k)$, монотонно убывая, должно сойтись к собственному значению $\lambda_{n+1-i}(A)$.

Все эти сходящиеся последовательности представлены последовательностями символов одного типа на рис. 7.2 и других рисунках данного раздела. Рассмотрим нижнюю диаграмму рис. 7.2. Для k, начиная примерно с 16, верхние и нижние плюсы образуют горизонтальные линии, находящиеся на уровне крайних собственных значений матрицы A, показанных в самом правом столбце; такое их поведение свидетельствует о сходимости. Аналогично, верхняя последовательность квадратиков образует горизонтальную линию на уровне второго по старшинству собственного значения матрицы A (указанного в самом правом столбце). Эта последовательность сходится несколько позже, чем последовательности крайних собственных значений. Для большего числа шагов алгоритма Ланцоша поведение собственных значений показано на двух верхних диаграммах рис. 7.3.

Подведем итог предыдущего обсуждения: *крайние собственные значения (т. е. наибольшие и наименьшие) сходятся первыми, а собственные значения в середине спектра — последними. Кроме того, сходимость монотонная, причем i-е по счету с правой (левой) стороны спектра собственное значение матрицы T_k возрастает (убывает) к i-му по счету справа (слева) собственному значению матрицы A, при условии, что алгоритм не остановится досрочно из-за того, что некоторое β_k равно нулю.*

Теперь мы более подробно изучим характер сходимости, вычислим действительные ошибки чисел Ритца и сравним эти ошибки с оценками из утверждения 2 теоремы 7.2. Для той же матрицы, которая использовалась для рис. 7.2, были проведены 99 шагов алгоритма Ланцоша; результаты показаны на рис. 7.3. Верхняя левая диаграмма относится к наибольшим собственным значениям, а верхняя правая — к наименьшим.

Две средние диаграммы рис. 7.3 показывают ошибки в вычисленных собственных значениях: левая — для четырех старших собственных значений, а правая — для четырех младших. Цвета на средних диаграммах соответствуют цветам на верхних. Ошибки измеряются (и наносятся на диаграммы) тремя способами:

- *Глобальные ошибки* (сплошные линии) вычисляются по формулам $|\lambda_i(T_k) - \lambda_i(A)|/|\lambda_i(A)|$. Деление на $|\lambda_i(A)|$ производится для нормализации ошибок так, чтобы их значения были заключены между 1 (что соответствует отсутствию какой-либо точности), с одной стороны, и величиной, примерно равной 10^{-16} (машинное эпсилон, или полная возможная точность), с другой стороны. С ростом k глобальная ошибка монотонно уменьшается; мы ожидаем, что в конечном счете она достигнет уровня машинного эпсилон, если только алгоритм Ланцоша не остановится преждевременно.
- *Локальные ошибки* (линии, состоящие из точек) вычисляются по формулам $\min_j |\lambda_i(T_k) - \lambda_j(A)|/|\lambda_i(A)|$. Локальная ошибка измеряет расстояние между $\lambda_i(T_k)$ и *ближайшим* собственным значением $\lambda_j(A)$ матрицы A, которое не обязательно совпадает с предельным собственным значением $\lambda_i(A)$. Мы

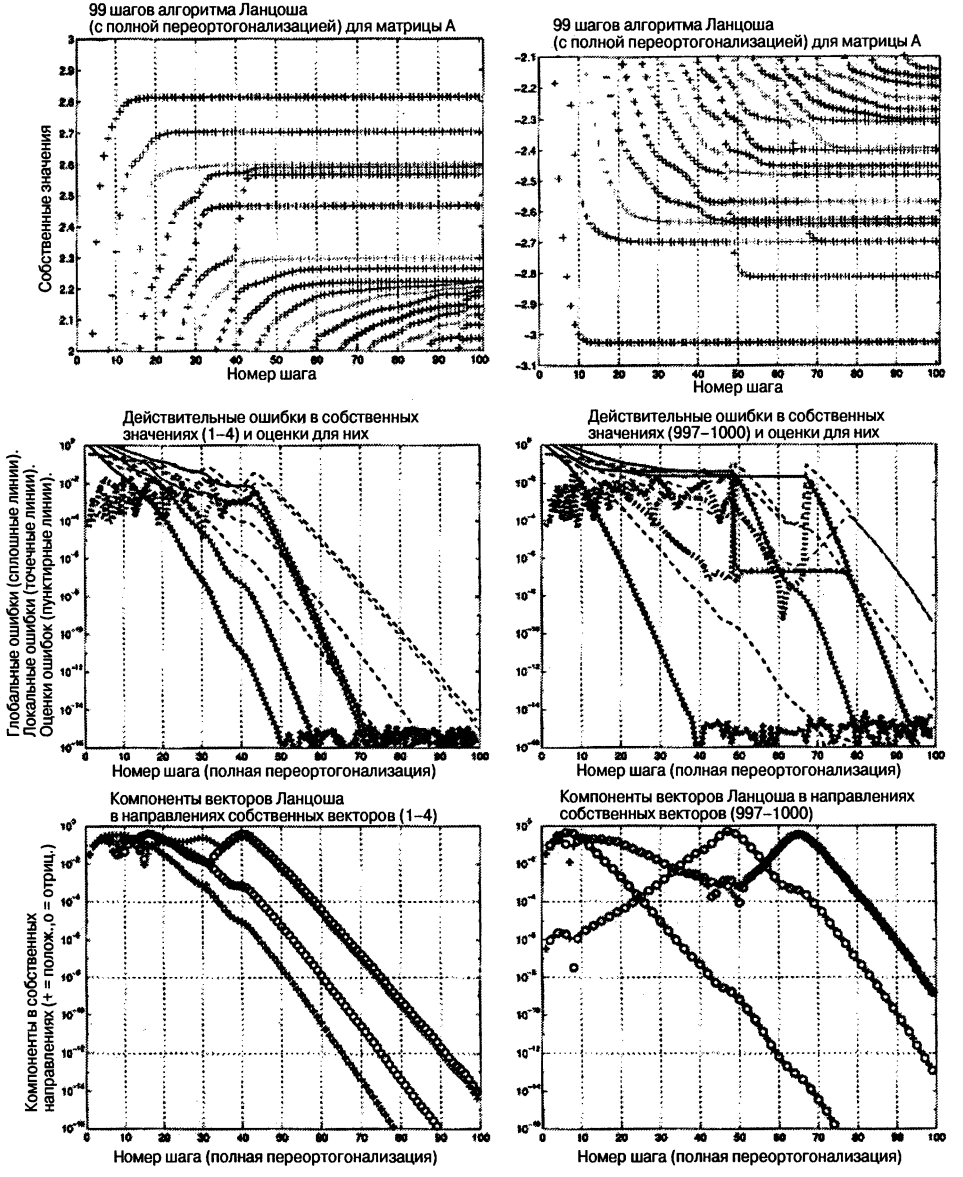


Рис. 7.3. 99 шагов алгоритма Ланцоша для матрицы A. Левые диаграммы относятся к наибольшим собственным значениям, а правые — к наименьшим. На двух верхних диаграммах показаны сами собственные значения, на двух средних — ошибки в них (глобальным ошибкам соответствуют сплошные линии, локальным ошибкам — линии, состоящие из точек, и оценкам ошибок — пунктирные линии). На двух нижних диаграммах показаны компоненты векторов Ланцоша по собственным направлениям. Цвета всех трех диаграмм одного вертикального ряда согласованы. (Цветной вариант рис. 7.3 см на. обложке книги. — *Перев.*)

вводим эту характеристику потому, что локальная ошибка подчас оказывается значительно меньше глобальной.

- *Оценки ошибок* (пунктирные линии) — это величины $|\beta_i v_i(k)|/|\lambda_i(A)|$, вычисляемые алгоритмом (с точностью до нормирующих коэффициентов $|\lambda_i(A)|$, которых алгоритм, разумеется, не знает!).

Две нижние диаграммы рис. 7.3 показывают компоненты векторов Ланцоша q_k в направлениях собственных векторов: левая диаграмма соответствует собственным векторам для четырех старших собственных значений, а правая диаграмма — собственным векторам для четырех младших собственных значений. Иными словами, на диаграммах изображены величины $q_k^T e_j = q_k(j)$, где e_j есть j-й собственный вектор диагональной матрицы A. Значения k изменяются от 1 до 99, а значения j от 1 до 4 (на левой диаграмме) и от 997 до 1000 (на правой диаграмме). Для изображения компонент взята логарифмическая шкала; символы «+» и «o» указывают соответственно положительные и отрицательные значения компонент. Ниже эти диаграммы используются для объяснения сходимости собственных значений.

Теперь с помощью рис. 7.3 мы исследуем сходимость более подробно. Наибольшие собственные значения матриц T_k (им соответствуют верхние черные плюсы на верхней левой диаграмме рис. 7.3) начинают сходиться к своему предельному значению (≈ 2.81) немедленно. После 25 шагов Ланцоша они имеют шесть верных десятичных разрядов, а к 50-му шагу верны с полной машинной точностью. Глобальная ошибка показана сплошной черной линией на левой средней диаграмме. Локальная ошибка (черная линия, состоящая из точек) после небольшого количества шагов совпадает с глобальной, хотя «случайно» может оказаться значительно меньше, если собственное значение $\lambda_i(T_k)$ на своем пути к $\lambda_i(A)$ окажется очень близким к некоторому другому собственному значению $\lambda_j(A)$. Пунктирная черная линия на той же диаграмме — это оценка относительной ошибки, вычисляемая алгоритмом. Примерно до 75-го шага она переоценивает величину действительной ошибки. Все же и оценка относительной ошибки правильно указывает на наличие нескольких верных десятичных разрядов в вычисленном старшем собственном значении.

Собственные значения с номерами 2, 3 и 4 (им соответствуют верхние красные, зеленые и синие плюсы на левой верхней диаграмме рис. 7.3) сходятся аналогичным образом, хотя для всякого i собственное значение i сходится немного быстрее, чем собственное значение $i+1$. Таково типичное поведение алгоритма Ланцоша.

На левой нижней диаграмме рис. 7.3 сходимость измеряется с помощью величин $q_k^T e_j$. Чтобы понять смысл этой диаграммы, посмотрим, что происходит с вектором Ланцоша q_k, когда сходится первое собственное значение. Сходимость означает, что соответствующий собственный вектор e_1 почти принадлежит крыловскому подпространству, натянутому на векторы Ланцоша. В частности, поскольку первое собственное значение сошлось после $k = 50$ шагов Ланцоша, вектор e_1 очень мало отличается от некоторой линейной комбинации векторов $q_1, \ldots, q_{50}$. Поскольку векторы q_i попарно ортогональны, вектор q_k для $k > 50$ должен быть почти ортогонален вектору e_1. Это подтверждается черной кривой на левой нижней диаграмме, которая к шагу 50 опускается до уровня, меньшего чем 10^{-7}. Красная кривая соответствует второй компоненте

вектора q_k, которая к шагу 60 достигает уровня 10^{-8}. Несколькими шагами позже сравнимую малость приобретают зеленая и синяя кривые (т. е. третья и четвертая компоненты векторов q_k).

Обсудим теперь поведение четырех младших собственных значений, которым соответствуют три правых диаграммы рис. 7.3. Матрица A и начальный вектор q_1 были выбраны так, чтобы проиллюстрировать некоторые осложнения в сходимости метода Ланцоша и подчеркнуть, что сходимость не всегда происходит столь очевидным образом, как в только что рассмотренном случае четырех старших собственных значений.

В частности, мы взяли значение, примерно равное 10^{-7}, для $q_1(999)$, т. е. для компоненты вектора q_1 в направлении собственного вектора для второго слева собственного значения (-2.81). Это значение в 10^5 раз меньше, чем все другие компоненты вектора q_1, которые равны между собой. Кроме того, в качестве третьего и четвертого слева собственных значений (т. е. собственных значений с номерами 998 и 997) мы взяли очень близкие числа -2.700001 и -2.7.

Сходимость наименьшего собственного значения матрицы T_k к $\lambda_{1000}(A) \approx -3.03$ происходит столь же «гладко», как и для старших собственных значений. К 40-му шагу оно имеет 16 верных десятичных разрядов.

Второе слева собственное значение матрицы T_k, указываемое красным цветом, демонстрирует вначале *ложную сходимость* к *третьему* слева собственному значению матрицы A, находящемуся вблизи -2.7. В самом деле, точечная красная линия на правой средней диаграмме рис. 7.3. показывает, что для шагов Ланцоша с номерами $40 < k < 50$ собственное значение $\lambda_{999}(T_k)$ согласуется с $\lambda_{998}(A)$ в шести десятичных разрядах. Для тех же значений k соответствующая оценка ошибки (красная пунктирная линия) свидетельствует, что $\lambda_{999}(T_k)$ в трех или четырех десятичных разрядах совпадает с *некоторым* собственным значением матрицы A. Причина ложной сходимости в том, что построение крыловского подпространства начинается с вектора, имеющего очень малую компоненту (а именно, 10^{-7}) в направлении e_{999}. Это подтверждается красной кривой на правой нижней диаграмме, начинающейся с уровня 10^{-7}. Лишь на 45-м шаге появляется большая компонента по направлению e_{999}. Только с этого момента, когда в крыловском подпространстве собственный вектор e_{999} уже достаточно хорошо представлен, собственное значение $\lambda_{999}(T_k)$ снова может начать сходиться, теперь к своему окончательному значению $\lambda_{999}(A) \approx -2.81$. Это видно из верхней и средней диаграмм правого ряда. По мере прогресса этого второго этапа сходимости компонента в направлении e_{999} снова убывает и становится очень малой, когда $\lambda_{999}(T_k)$ достаточно точно аппроксимирует $\lambda_{999}(A)$. (Количественное описание связи между скоростью сходимости и величиной $q_1^T e_{999}$ дает теорема Каниэля—Саада, обсуждаемая ниже.)

Если бы q_1 был *точно* ортогонален вектору e_{999} (т. е. вместо $q_1^T e_{999} = 10^{-7}$ мы имели бы $q_1^T e_{999} = 0$), то все последующие векторы Ланцоша тоже были бы ортогональны к e_{999}. Это означало бы, что $\lambda_{999}(T_k)$ не может сойтись к $\lambda_{999}(A)$. (Доказательство этого утверждения вынесено в вопрос 7.3). На рис. 7.4 показана эта ситуация; вектор q_1 для него был немного изменен так, чтобы выполнялось условие $q_1^T e_{999} = 0$. Мы видим, что здесь вообще не появляется приближения к собственному значению $\lambda_{999}(A) \approx -2.81$.

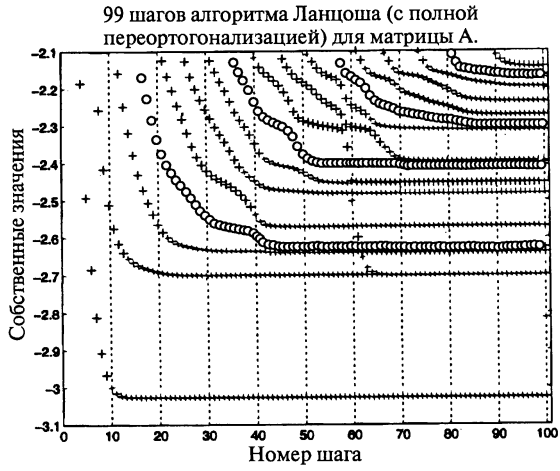


Рис. 7.4. Алгоритм Ланцоша в применении к матрице A. Начальный вектор q_1 ортогонален собственному вектору, соответствующему второму (с левого конца спектра) собственному значению ≈ -2.81. Приближение к этому собственному значению вычислить не удается.

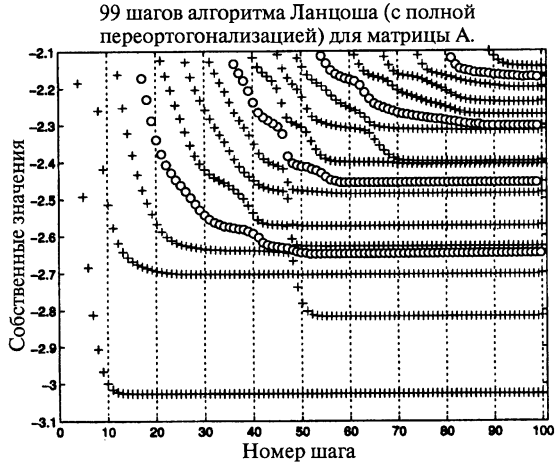


Рис. 7.5. Алгоритм Ланцоша в применении к матрице A. Третье и четвертое (с левого конца спектра) собственные значения равны. Вычисляется только одно приближение к этому двойному собственному значению.

К счастью, если q_1 выбирается случайным образом, то вероятность для него оказаться ортогональным какому-либо собственному вектору чрезвычайно мала. Чтобы получить «статистическое» подтверждение, что никакие собственные значения не были упущены, мы всегда можем повторно применить алгоритм Ланцоша с другим случайным вектором q_1.

Другим источником «ложной сходимости» являются (почти) кратные собственные значения; в нашем случае это третье и четвертое (с левого конца спектра) собственные значения $\lambda_{998}(A) = -2.700001$ и $\lambda_{997}(A) = -2.7$. Исследуя поведение $\lambda_{998}(T_k)$, которому соответствует самая нижняя зеленая кривая на верхней и средней правых диаграммах рис. 7.3, мы видим, что на шагах Ланцоша с номерами $50 < k < 75$ происходит *ложная сходимость* этого собственного значения к числу -2.7000005, находящемуся *посередине* между двумя ближайшими собственными значениями матрицы A. Это не заметно при уровне разрешения, даваемом верхней диаграммой, но очевидно из горизонтального сегмента, который сплошная зеленая кривая на средней диаграмме имеет для шагов Ланцоша с номерами $50 < k < 75$. Начиная с шага 76 устанавливается быстрая сходимость к окончательному значению $\lambda_{998}(A) = -2.700001$.

Между тем, четвертое (с левого конца спектра) собственное значение $\lambda_{997}(T_k)$, показанное синим цветом, демонстрирует ложную сходимость к числу, находящемуся вблизи $\lambda_{996}(A) \approx -2.64$. Синяя точечная линия на средней правой диаграмме показывает, что вблизи $k = 61$ собственные значения $\lambda_{997}(T_k)$ и $\lambda_{996}(A)$ совпадают в девяти десятичных разрядах. Начиная с шага $k = 65$, устанавливается быстрая сходимость к окончательному значению $\lambda_{997}(A) = -2.7$. Это же можно видеть из нижней правой диаграммы, где компоненты в направлениях e_{997} и e_{998} увеличиваются на шагах с номерами $50 < k < 65$; они снова убывают, когда собственные значения вступают в зону быстрой сходимости.

Если бы $\lambda_{997}(A)$ было *в точности* двойным собственным значением, то (в точной арифметике) T_k имела бы лишь одно собственное значение, близкое к $\lambda_{997}(A)$, а не два. (Доказательство этого утверждения вынесено в вопрос 7.3.) На рис. 7.5 показана эта ситуация; матрица A для него была совсем немного изменена так, чтобы число -2.7 стало двойным собственным значением. Заметим, что на любом шаге имеется лишь одно приближение к $\lambda_{998}(A) = \lambda_{997}(A) = -2.7$.

К счастью, существует ряд приложений, в которых достаточно определить одну копию каждого (кратного) собственного значения, а знание кратностей несущественно. Кроме того, кратные собственные значения вместе с правильными кратностями можно находить посредством «блочных методов Ланцоша» (см. алгоритмы, упоминаемые в разд. 7.6).

Исследуя другие собственные значения на верхней правой диаграмме рис. 7.3, мы видим, что ложная сходимость является весьма обычным событием. Об этом свидетельствуют многочисленные короткие горизонтальные сегменты, образованные плюсами одного цвета; они затем сменяются фазами убывания к меньшему собственному значению. Например, седьмое (с левого конца спектра) собственное значение на разных шагах Ланцоша хорошо аппроксимируется пятым (черный цвет), шестым (красный) и седьмым (зеленый) собственными значениями матрицы T_k.

Эти явления ложной сходимости объясняют важность вычислимой оценки погрешности из утверждения 2 теоремы 7.2 для контроля сходимости собственных значений [198]. Если оценка погрешности мала, то вычисленное собственное значение действительно является хорошим приближением к некоторому точному собственному значению, даже если какое-то другое оказалось «пропущенным». ◇

Имеется еще одна оценка погрешности, принадлежащая Каниэлю и Сааду, которая проливает свет на причины ложной сходимости. Эта оценка составлена в терминах углов между начальным вектором q_1 и нужными собственными векторами, а также чисел Ритца и нужных собственных значений. Другими словами, в ней участвуют величины, неизвестные в ходе вычислений, поэтому с практической точки зрения она бесполезна. Однако эта оценка показывает, что если q_1 почти ортогонален к разыскиваемому собственному вектору или же разыскиваемое собственное значение почти кратно, то следует ожидать медленной сходимости. По поводу деталей см. [197, разд. 12-4].

7.4. Алгоритм Ланцоша в арифметике с плавающей точкой

Пример, разобранный в предыдущем разделе, описывал поведение «идеального» алгоритма Ланцоша, когда округления по существу отсутствуют. Мы назвали тщательную, но дорогостоящую реализацию алгоритма 6.10 *методом Ланцоша с полной переортогонализацией*, противопоставляя ее исходной необременительной реализации, называемой методом Ланцоша без переортогонализации (см. HOMEPAGE/Matlab/LanczosNoReorthog.m). Приведем запись, объединяющую оба этих метода.

Алгоритм 7.2. *Алгоритм Ланцоша (соответственно с полной переортогонализацией и без переортогонализации) для вычисления собственных значений и собственных векторов матрицы* $A = A^T$*:*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $\begin{cases} z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i,\ z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i \\ \qquad\qquad \textit{полная переортогонализация} \\ z = z - \alpha_j q_j - \beta_{j-1} q_{j-1} \\ \qquad\qquad \textit{переортогонализация отсутствует} \end{cases}$
  $\beta_j = \|z\|_2$
  если $\beta_j = 0$, *то прекратить выполнение алгоритма*
  $q_{j+1} = z/\beta_j$
  Вычислить собственные значения и собственные векторы матрицы T_j *и оценки погрешностей в них*
end for

Полная переортогонализация соответствует *повторному* проведению процесса ортогонализации Грама—Шмидта «$z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i$» для того, чтобы почти гарантировать, что z будет ортогонален векторам $q_1, \ldots, q_{j-1}$. (См. алгоритм 3.1, а также [197, разд. 6-9] и [171, гл. 7] по поводу обсуждений вопроса о том, когда «двух ортогонализаций достаточно».) В разд. 6.6.1 мы показали, что в точной арифметике z ортогонален $q_1, \ldots, q_{j-1}$ без какой-либо переортогонализации. К сожалению, это свойство ортогональности, на котором до сих пор основывался весь наш анализ, теряется из-за округлений.

Потеря ортогональности не заставляет алгоритм вести себя совершенно непредсказуемым образом. В самом деле, мы увидим, что цена, которую мы платим за потерю, — это приобретение *повторных копий* сошедшихся чисел Ритца. Иными словами, для больших k матрица T_k может иметь не одно собственное значение, очень близкое к $\lambda_i(A)$, но много таких собственных значений. Это не является катастрофой, если нас не интересуют кратности собственных значений и если замедление сходимости внутренних собственных значений, вызванное появлением копий крайних собственных чисел, несущественно для нас. Детальное описание реализации алгоритма Ланцоша, оперирующей именно таким образом, можно найти в [57], а сами программы — в NETLIB/lanczos. (В этих программах используются некоторые эвристики для оценки кратностей собственных значений.)

Однако если знание точных кратностей является важным, то нужно поддерживать (почти) ортогональность векторов Ланцоша. В таком случае можно прибегнуть к алгоритму Ланцоша с полной переортогонализацией, как было сделано в предыдущем разделе. Легко проверить, правда, что k шагов этого алгоритма потребуют $O(k^2n)$ флопов вместо $O(kn)$ и $O(kn)$ слов памяти вместо $O(n)$. Такая цена может оказаться слишком высокой.

К счастью, имеется вариант, промежуточный между полной переортогонализацией и отсутствием всякой переортогонализации; он соединяет в себе почти в полном объеме лучшие качества этих двух крайних возможностей. Оказывается, что векторы q_k теряют ортогональность весьма систематическим образом, а именно, приобретая большие компоненты в направлениях уже сошедшихся векторов Ритца. (Именно это приводит к появлению повторных копий сошедшихся чисел Ритца.) Мы сейчас проиллюстрируем эту систематическую потерю точности примером, а затем объясним ее с помощью приводимой ниже теоремы Пэйджа. Будет видно, что, следя за вычисляемыми оценками ошибок, можно с большой надежностью предсказывать, какие векторы q_k приобретают большие компоненты и в направлении каких векторов Ритца. После этого можно *выборочно ортогонализовать* векторы q_k по отношению к нескольким ранее сошедшимся векторам Ритца вместо того, чтобы на каждом шаге ортогонализовать q_k ко всем более ранним векторам q_i, как это делается при полной переортогонализации. Таким способом мы достигаем (почти) ортогональности векторов Ланцоша, затрачивая очень небольшую дополнительную работу. Более подробно выборочная ортогонализация обсуждается в следующем разделе.

Пример 7.2. Для матрицы примера 7.1 были проведены 146 шагов Ланцоша. Картина сходимости метода представлена на рис. 7.7. Диаграммы правого ряда соответствуют методу с полной переортогонализацией, а диаграммы левого ряда — методу без переортогонализации. Эти диаграммы аналогичны диаграммам рис. 7.3 с тем отличием, что удалены графики глобальной ошибки, загромождающие средние диаграммы.

На рис. 7.6 наименьшее сингулярное число $\sigma_{\min}(Q_k)$ изображено как функция от номера шага k. В точной арифметике матрица Q_k имела бы ортонормированные столбцы, а потому было бы $\sigma_{\min}(Q_k) = 1$. В арифметике с округлениями Q_k теряет ортогональность столбцов, начиная примерно с $k = 70$. К 80-му шагу значение $\sigma_{\min}(Q_k)$ падает до уровня 0.01, и именно с этого момента две верхние диаграммы рис. 7.7 начинают заметно различаться.

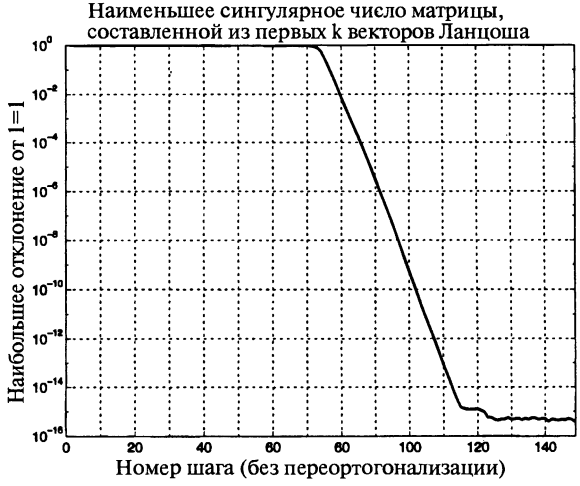


Рис. 7.6. Алгоритм Ланцоша без переортогонализации в применении к матрице A. Для k от 1 до 149 показано наименьшее сингулярное число $\sigma_{\min}(Q_k)$ матрицы (Q_k), составленной из векторов Ланцоша. В отсутствие округлений столбцы матрицы Q_k ортонормированы, и все сингулярные числа равны единице. При наличии округлений Q_k теряет свойство полноты ранга.

В частности, начиная с $k = 80$, второе по величине (красное) собственное значение $\lambda_2(T_k)$, которое уже сошлось к $\lambda_2(A) \approx 2.7$ почти в 16 разрядах, вдруг в несколько шагов поднимается до уровня $\lambda_1(A) \approx 2.81$ (см. верхнюю левую диаграмму рис. 7.7). В результате получается «вторая копия» числа $\lambda_1(A)$; первую копию дает $\lambda_1(T_k)$, которому соответствует черный цвет. (На рисунке эти черные плюсы трудно разглядеть, потому что прямо на них нанесены красные.) Это «превращение» числа $\lambda_2(T_k)$ сопровождается прыжком оценки ошибки (пунктирная красная линия на левой средней диаграмме). Кроме того, этому превращению предшествует возрастание компоненты по направлению e_1 на левой нижней диаграмме: на шаге 50 черная кривая начинает расти вместо того, чтобы, продолжая убывать, снизиться до уровня машинного эпсилон, как это происходит при полной переортогонализации на правой нижней диаграмме. И то, и другое свидетельствует о том, что алгоритм отклоняется от своего точного пути (и что, следовательно, необходима некоторая выборочная ортогонализация). Вслед за завершением сходимости второй копии числа $\lambda_1(A)$ начинает снова убывать компонента векторов Ланцоша в направлении e_1 (этот момент наступает вскоре после $k = 80$).

Аналогично, начиная примерно с $k = 95$ появляется вторая копия числа $\lambda_2(A)$: синяя кривая (соответствующая $\lambda_4(T_k)$) на левой верхней диаграмме перемещается с уровня $\lambda_3(A) \approx 2.6$ на уровень $\lambda_2(A) \approx 2.7$. В этот момент мы имеем две копии для $\lambda_1(A) \approx 2.81$ и две копии для $\lambda_2(A)$. Это не так легко увидеть из диаграмм, поскольку плюсы одного цвета затеняют плюсы другого (красный цвет поверх черного и синий поверх зеленого). Превращение сопровождается резким прыжком оценки ошибки для $\lambda_4(T_k)$ (пунктирная

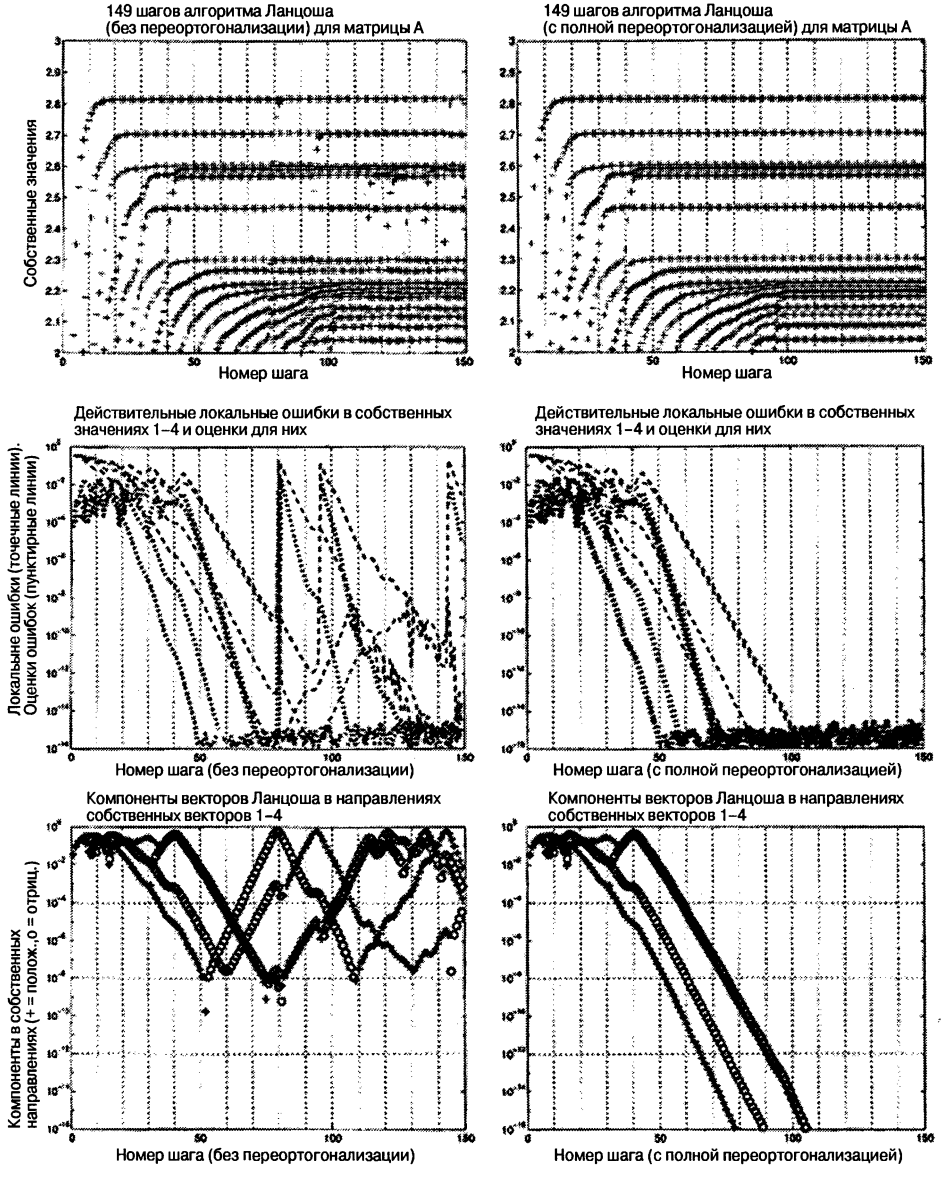


Рис. 7.7. 149 шагов алгоритма Ланцоша для матрицы A. Столбец 150 (правый край верхних диаграмм) показывает собственные значения матрицы A. Диаграммы левого ряда соответствуют алгоритму без переортогонализации, а диаграммы правого ряда — алгоритму с полной переортогонализацией. (Цветной вариант рис. 7.7 см. на обложке книги. — *Перев.*)

синяя линия на левой средней диаграмме) возле $k = 95$ и предсказывается возрастанием красной кривой на левой нижней диаграмме, свидетельствующей о росте компоненты по направлению e_2 в векторах Ланцоша. Эта компонента достигает максимума вблизи $k = 95$, а затем начинает убывать.

Наконец, в районе $k = 145$ появляется *третья* копия числа $\lambda_1(A)$. Снова это событие сопровождается и предсказывается изменениями на двух нижних левых диаграммах. Если бы мы продолжали процесс Ланцоша далее, то периодически получали бы новые копии многих ранее сошедшихся чисел Ритца. ◇

Обсуждаемая ниже теорема объясняет явления, которые мы наблюдали в данном примере, и указывает практичный критерий для выборочной ортогонализации векторов Ланцоша. Чтобы не завязнуть в анализе, берущем в расчет все округления, мы выделим, опираясь на существующий практический опыт, те немногие ошибки округлений, которые действительно важны, а остальные будем просто игнорировать [197, разд. 13-4]. Это позволит нам записать алгоритм Ланцоша без переортогонализации одним уравнением

$$\beta_j q_{j+1} + f_j = Aq_j - \alpha_j q_j - \beta_{j-1} q_{j-1}. \tag{7.3}$$

В этом уравнении переменные обозначают величины, действительно хранимые в машине, за исключением вектора f_j, который представляет ошибки округлений, проистекающие из вычисления правой части, а затем числа β_j и вектора q_{j+1}. Норма $\|f_j\|_2$ ограничена величиной $O(\varepsilon\|A\|)$, где ε — машинный эпсилон, и это все, что нам нужно знать о f_j. Кроме того, мы будем пользоваться *точным* спектральным разложением $T_k = V\Lambda V^T$, поскольку известно, что ошибки округлений, совершенных при его вычислении, не важны для нашего анализа. Таким образом, V — ортогональная матрица, в то время как столбцы матрицы Q_k не обязаны быть ортогональными.

Теорема 7.3 (Пэйдж). *Примем обозначения и предположения предыдущего абзаца. Кроме того, положим* $Q_k = [q_1, \ldots, q_k]$, $V = [v_1, \ldots, v_k]$ *и* $\Lambda = \operatorname{diag}(\theta_1, \ldots, \theta_k)$. *Столбцы* $y_{k,i} = Q_k v_i$ *матрицы* $Q_k V$ *будем по-прежнему называть векторами Ритца, а числа* θ_i *— числами Ритца. Тогда*

$$y_{k,i}^T q_{k+1} = \frac{O(\varepsilon\|A\|)}{\beta_k |v_i(k)|}.$$

Иными словами, компонента $y_{k,i}^T q_{k+1}$ вычисленного вектора Ланцоша q_{k+1} в направлении вектора Ритца $y_{k,i} = Q_k v_i$ обратно пропорциональна величине $\beta_k |v_i(k)|$, являющейся оценкой погрешности для соответствующего числа Ритца θ_i (см. утверждение 2 теоремы 7.2). Поэтому, когда число Ритца сходится, а его оценка погрешности $\beta_k |v_i(k)|$ приближается к нулю, вектор Ланцоша q_{k+1} приобретает большую компоненту в направлении вектора Ритца $y_{k,i}$. В результате векторы Ритца становятся (почти) линейно зависимыми, что мы и видели в примере 7.2. На рис. 7.8 приведены графики оценки погрешности $|\beta_k v_i(k)|/|\lambda_i(A)| \approx |\beta_k v_i(k)|/\|A\|$ и величины $y_{k,i}^T q_{k+1}$ для наибольшего числа Ритца ($i = 1$, верхний график) и второго по величине числа Ритца ($i = 2$, нижний график) в нашем примере с диагональной матрицей порядка 1000. Согласно теореме Пэйджа, произведение этих двух величин должно иметь порядок $O(\varepsilon)$, что подтверждается нашими полулогарифмическими графиками: на каждом из них обе кривые симметричны относительно средней линии $\sqrt{\varepsilon}$.

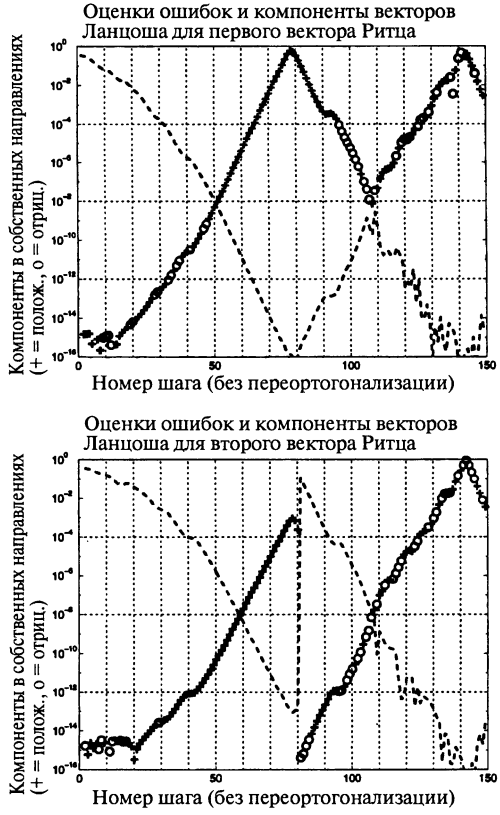


Рис. 7.8. Алгоритм Ланцоша без переортогонализации в применении к матрице A. Показаны первые 149 шагов для наибольшего собственного значения (верхний график) и для второго по величине собственного значения (нижний график). Как и на предыдущих рисунках, пунктирные линии соответствуют оценкам ошибок. Линии, составленные из символов + и o, указывают значение величины $y_{k,i}^T q_{k+1}$, т. е. компоненты вектора Ланцоша q_{k+1} в направлении вектора Ритца для наибольшего числа Ритца ($i = 1$, верхний график) или для второго по величине числа Ритца ($i = 2$, нижний график).

Доказательство (теоремы Пэйджа). Начнем с того, что запишем k уравнений (7.3), отвечающих значениям j от 1 до k, одним соотношением

$$\begin{aligned} AQ_k &= Q_k T_k + [0, \ldots, 0, \beta_k q_{k+1}] + F_k \\ &= Q_k T_k + \beta_k q_{k+1} e_k^T + F_k, \end{aligned}$$

где $e_k^T = [0, \ldots, 0, 1]$ — вектор размерности k, а $F_k = [f_1, \ldots, f_k]$ — матрица, составленная из ошибок округлений. Опуская для простоты индекс k, имеем $AQ = QT + \beta q e^T + F$. Умножая обе части слева на Q^T, получаем $Q^T AQ = Q^T QT + \beta Q^T q e^T + Q^T F$. Поскольку $Q^T AQ$ — симметричная матри-

ца, матрица $Q^TQT + \beta Q^Tqe^T + Q^TF$ совпадает со своей транспонированной, откуда выводим

$$0 = (Q^TQT - TQ^TQ) + \beta(Q^Tqe^T - eq^TQ) + (Q^TF - F^TQ). \tag{7.4}$$

Пусть θ и v суть соответственно число и вектор Ритца, так что $Tv = \theta v$. Заметим, что величина

$$v^T\beta(eq^TQ)v = [\beta v(k)] \cdot [q^T(Qv)] \tag{7.5}$$

есть произведение оценки погрешности $\beta v(k)$ и величины $q^T(Qv) = q^Ty$, т. е. компоненты вектора q в направлении вектора Ритца. Теорема Пэйджа утверждает, что это произведение должно иметь величину $O(\varepsilon\|A\|)$. Чтобы доказать это, получим выражение для eq^TQ, преобразуя равенство (7.4), а затем воспользуемся соотношением (7.5).

С этой целью введем дополнительные упрощающие предположения относительно округлений. Всякий столбец матрицы Q получается делением вектора z на его норму, поэтому в пределах машинной точности все диагональные элементы матрицы Q^TQ равны 1; мы будем считать их точно равными 1. Далее, вектор $z' = z - \alpha_j q_j = z - (q_j^Tz)q_j$ вычисляется в алгоритме Ланцоша так, чтобы быть ортогональным вектору q_j; следовательно, q_{j+1} и q_j ортогональны с почти полной машинной точностью. Поэтому $q_{j+1}^Tq_j = (Q^TQ)_{j+1,j} = O(\varepsilon)$; для простоты будем считать, что $(Q^TQ)_{j+1,j} = 0$. Введем представление $Q^TQ = I + C + C^T$, где C — нижнетреугольная матрица. Наши предположения об округлениях означают, что в матрице C отличные от нуля могут быть только элементы, находящиеся на второй поддиагонали и ниже ее. Имеем

$$Q^TQT - TQ^TQ = (CT - TC) + (C^TT - TC^T).$$

Учитывая расположение нулей в C и T, легко показать, что матрица $CT - TC$ нижняя строго треугольная, а матрица $C^TT - TC^T$ верхняя строго треугольная. Поскольку в векторе e отлична от нуля только последняя компонента, в матрице eq^TQ отлична от нуля только последняя строка. Более того, из наших предположений об округлениях следует, что последний элемент этой строки равен нулю. Это означает, в частности, что матрица eq^TQ нижняя строго треугольная, а матрица Q^Tqe^T верхняя строго треугольная. Используя нижний строго треугольный вид матриц eq^TQ и $CT - TC$, можно вывести из (7.4) равенство

$$0 = (CT - TC) - \beta eq^TQ + L, \tag{7.6}$$

где L — нижняя строго треугольная часть матрицы $Q^TF - F^TQ$. Умножая (7.6) слева на v^T, а справа на v и учитывая соотношение (7.5) и равенство $v^T(CT - TC)v = v^TCv\theta - \theta v^TCv = 0$, получаем

$$v^T\beta(eq^TQ)v = [\beta v(k)] \cdot [q^T(Qv)] = v^TLv.$$

Так как $|v^TLv| \le \|L\| = O(\|Q^TF - F^TQ\|) = O(\|F\|) = O(\varepsilon\|A\|)$, то

$$[\beta v(k)] \cdot [q^T(Qv)] = O(\varepsilon\|A\|),$$

что доказывает теорему Пэйджа. □

7.5. Алгоритм Ланцоша с выборочной ортогонализацией

Обсудим модификацию алгоритма Ланцоша, которая обладает почти столь же высокой точностью, как алгоритм с полной переортогонализацией, и почти столь же низкой стоимостью, как алгоритм без переортогонализации. Эта модификация называется алгоритмом Ланцоша с *выборочной ортогонализацией*. Как уже отмечалось в предыдущем разделе, наша цель состоит в том, чтобы поддерживать в вычисленных векторах Ланцоша q_k как можно бóльшую степень ортогональности (для обеспечения высокой точности), достигая этого путем ортогонализации q_k к возможно меньшему числу векторов (чтобы снизить стоимость). Теорема Пэйджа (теорема 7.3 из предыдущего раздела) показывает, что векторы q_k теряют ортогональность вследствие приобретения больших компонент в направлениях векторов Ритца $y_{i,k} = Q_k v_i$, отвечающих сошедшимся числам Ритца θ_i; на сходимость же указывают малые значения оценки погрешности $\beta_k|v_i(k)|$. Иллюстрация этого явления была дана в примере 7.2.

Итак, простейший вариант выборочной ортогонализации состоит в том, чтобы следить на каждом шаге за оценками погрешностей $\beta_k|v_i(k)|$ и, когда какая-то оценка становится слишком малой, проводить ортогонализацию вектора z во внутреннем цикле алгоритма Ланцоша по отношению к соответствующему вектору $y_{i,k}$: $z = z - (y_{i,k}^T z)y_{i,k}$. Величина $\beta_k|v_i(k)|$ считается малой, если она меньше, чем $\sqrt{\varepsilon}||A||$. Действительно, в этом случае, согласно теореме Пэйджа, компонента $|y_{i,k}^T q_{k+1}| = |y_{i,k}^T z|/||z||_2$, скорей всего, превосходит уровень $\sqrt{\varepsilon}$. (На практике можно использовать $||T_k||$ вместо $||A||$, поскольку первое число известно, а второе, может быть, и нет.) В результате приходим к следующему алгоритму:

Алгоритм 7.3. *Алгоритм Ланцоша с выборочной ортогонализацией для вычисления собственных значений и собственных векторов матрицы* $A = A^T$*:*

$q_1 = b/||b||_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
  / Провести выборочную ортогонализацию по отношению к сошедшимся векторам Ритца */*
  for $i \le k$, *таких, что* $\beta_k|v_i(k)| \le \sqrt{\varepsilon}||T_k||$
    $z = z - (y_{i,k}^T z)y_{i,k}$
  end for
  $\beta_j = ||z||_2$
  если $\beta_j = 0$, *то прекратить выполнение алгоритма*
  $q_{j+1} = z/\beta_j$
  Вычислить собственные значения и собственные векторы матрицы T_j *и оценки погрешностей в них*
end for

В следующем примере описывается, что происходит с нашей диагональной матрицей порядка 1000 при использовании этого алгоритма (HOMEPAGE/Matlab/LanczosSelectOrthog.m).

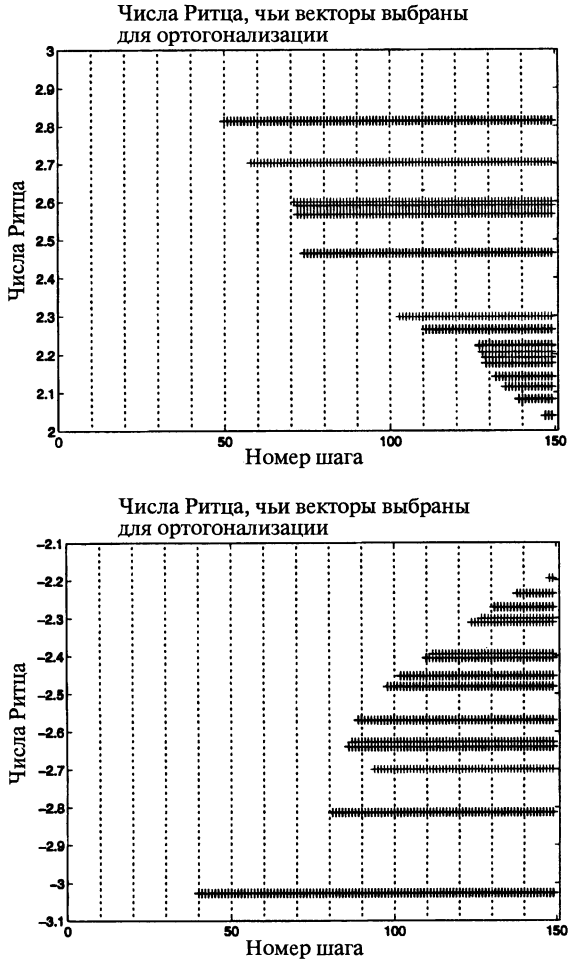


Рис. 7.9. Алгоритм Ланцоша с выборочной ортогонализацией в применении к матрице A. Показаны числа Ритца, соответствующие векторам Ритца, выбранным для ортогонализации.

Пример 7.3. Поведение алгоритма Ланцоша с выборочной ортогонализацией визуально неотличимо от поведения алгоритма с полной переортогонализацией, показанного на трех правых диаграммах рис. 7.7. Иными словами, выборочная ортогонализация обеспечивала такую же точность, как полная переортогонализация.

Для всех матриц Q_k наименьшие сингулярные числа были больше уровня $1 - 10^{-8}$. Это означает, что выборочная ортогонализация действительно поддерживает ортогональность векторов Ланцоша в пределах примерно половины рабочей точности.

На рис. 7.9 показаны числа Ритца, соответствующие векторам Ритца, выбранным для переортогонализации. Так как выбираемые векторы отвечают сошедшимся числам Ритца, а первыми сходятся наибольшие и наименьшие числа, то рисунок состоит из двух графиков: большие сошедшиеся числа Ритца изображены на верхнем графике, а малые — на нижнем. Верхний график соответствует числам Ритца, сошедшимся, по крайней мере, с половиной машинной точности, тем самым числам, что показаны на правой верхней диаграмме рис. 7.7. Всего для ортогонализации было выбрано 1485 векторов Ритца из общего числа их $149*150/2 = 11175$. Таким образом, чтобы поддерживать (почти) тот же уровень ортогональности векторов Ланцоша, что и в полной переортогонализации, в методе выборочной ортогонализации пришлось затратить лишь $1485/11175 \approx 13\%$ работы по сравнению с алгоритмом полной переортогонализации.

На рис. 7.10 показано, как метод Ланцоша с выборочной ортогонализацией осуществляет поддержание ортогональности векторов Ланцоша к векторам Ритца, отвечающим двум старшим числам Ритца. Верхний график есть суперпозиция двух графиков рис. 7.8, изображающих оценки погрешностей и компоненты в направлениях векторов Ритца для метода Ланцоша без переортогонализации. Нижний график показывает те же величины для метода выборочной ортогонализации. Обратим внимание на то, что на шаге $k = 50$ оценка погрешности для наибольшего собственного значения (пунктирная черная линия) достигает порогового значения $\sqrt{\varepsilon}$. Соответствующий вектор Ритца выбирается для ортогонализации (это показано плюсами в верхней части верхнего графика рис. 7.9); в результате компонента в направлении этого вектора исчезает с нижнего графика рис. 7.10. Несколькими шагами позже, а именно при $k = 58$, оценка погрешности для второго по величине числа Ритца достигает уровня $\sqrt{\varepsilon}$, и соответствующий вектор Ритца также выбирается для ортогонализации. Оценки погрешностей на нижнем графике продолжают убывать, достигая в конечном счете уровня машинного эпсилон и оставаясь далее на этом уровне. В то же время на верхнем графике оценки погрешностей с какого-то момента вновь начинают расти. ◇

7.6. Другие возможности

Выборочной ортогонализацией история не заканчивается, так как стоимость симметричного алгоритма Ланцоша можно снизить в еще большей степени. Оказывается, что в течение многих шагов после того, как вектор Ланцоша был ортогонализован к конкретному вектору Ритца y, ортогонализации по отношению к этому вектору y не требуется. Поэтому можно устранить значительную часть работы по переортогонализации из алгоритма 7.3. Существуют простые и дешевые рекурсии, позволяющие определить момент, когда переортогонализация необходима [224,192]. Другое усовершенствование состоит в использовании оценок погрешностей для эффективного различения сошедшихся и «ложно сошедшихся» собственных значений [198]. Наиболее современная реализация алгоритма Ланцоша описана в [125]. Другой вариант реализации выбран в пакете ARPACK (NETLIB/scalapack/readme.arpack [171, 233]).

Если алгоритм Ланцоша применяется к матрице $(A - \sigma I)^{-1}$, полученной из A сдвигом и обращением, то следует ожидать, что в первую очередь сойдут-

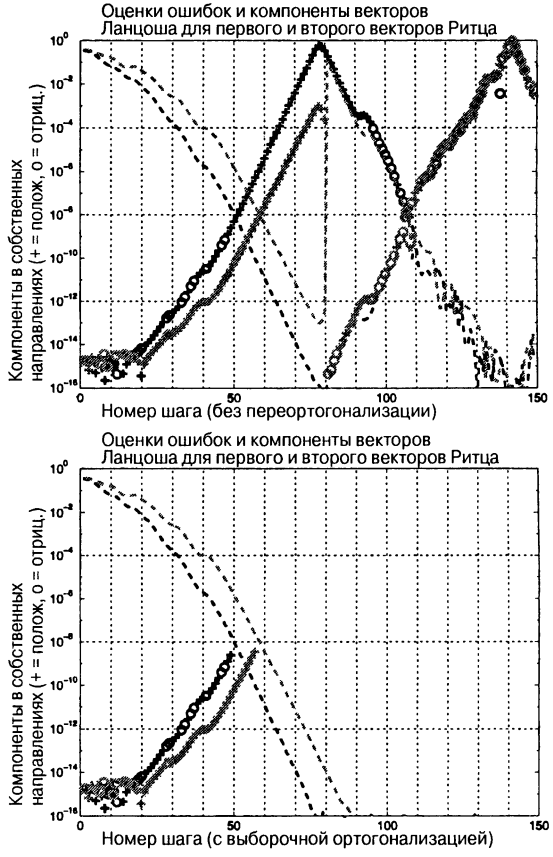


Рис. 7.10. Алгоритм Ланцоша с выборочной ортогонализацией в применении к матрице A. Показаны первые 149 шагов алгоритма без переортогонализации (верхний график) и алгоритма с выборочной ортогонализацией (нижний график). Наибольшее собственное значение указано жирным цветом, а второе по величине собственное значение — бледным. Как и на предыдущих рисунках, пунктирные линии соответствуют оценкам ошибок. Линии, составленные из символов + и ∘, указывают значение величины $y_{k,i}^T q_{k+1}$, т. е. компоненты вектора Ланцоша q_{k+1} в направлении вектора Ритца для наибольшего числа Ритца ($i = 1$, жирный цвет) или для второго по величине числа Ритца ($i = 2$, бледный цвет). Отметим, что в методе выборочной ортогонализации эти компоненты исключаются в результате первых выборочных ортогонализаций на шагах 50 ($i = 1$) и 58 ($i = 2$).

ся собственные значения, ближайшие к σ. Существуют и другие методы «предобусловливания» матрицы A, обеспечивающие более быструю сходимость к некоторым собственным значениям. Например, в задачах квантовой химии, где типичны матрицы A со строгим диагональным преобладанием, широко используется метод Дэвидсона [60]. Этот метод может быть скомбинирован с методом Якоби [229].

7.7. Итерационные алгоритмы для несимметричной проблемы собственных значений

Для несимметричной матрицы A описанный выше алгоритм Ланцоша не пригоден. Имеются две альтернативные возможности.

Первая состоит в том, чтобы использовать *алгоритм Арнольди* (алгоритм 6.9). Вспомним, что в алгоритме Арнольди вычисляется ортогональный базис Q_k крыловского подпространства $\mathcal{K}_k(A, q_1)$, в котором $Q_k^T A Q_k = H_k$ — верхняя хессенбергова матрица (а не симметричная трехдиагональная). Аналог процедуры Рэлея—Ритца для этого случая состоит в том, чтобы аппроксимировать собственные значения матрицы A собственными значениями матрицы H_k. Поскольку матрица A несимметрична, ее собственные значения могут быть комплексными и/или плохо обусловленными. Поэтому многие привлекательные оценки погрешностей и свойства монотонной сходимости, справедливые для описанного в разд. 7.3 алгоритма Ланцоша, теперь не имеют места. Тем не менее, существуют эффективные алгоритмы и их реализации. Полезные сведения по этому поводу можно найти в [154, 171, 212, 216, 217, 233] и книге [213]. Наиболее современные программы описаны в [171, 233]; они помещены в NETLIB/scalapack/readme.arpack. Matlab-команда `eigs` (от «sparse eigenvalues» — собственные значения разреженной матрицы) использует эти программы.

Вторая возможность заключается в том, чтобы использовать *несимметричный алгоритм Ланцоша*. Этот алгоритм пытается привести A к несимметричной трехдиагональной форме посредством неортогонального подобия. Надежда здесь на то, что вычислить собственные значения для (разреженной!) несимметричной трехдиагональной матрицы проще, чем для хессенберговой матрицы, вычисляемой алгоритмом Арнольди. К сожалению, преобразования подобия могут быть весьма плохо обусловлены, вследствие чего собственные значения исходной матрицы и полученной из нее трехдиагональной матрицы могут сильно различаться. В действительности, подходящее подобие даже не всегда можно найти из-за явления, называемого «обрывом» [42, 134, 135, 199]. В [16, 18, 55, 56, 64, 108, 202, 265, 266] предлагаются различные способы исправления ситуации обрыва с помощью процесса, называемого «заглядыванием вперед», дается их анализ и описываются их реализации.

Наконец, к решению разреженной несимметричной проблемы собственных значений можно применить методы итерирования подпространства (алгоритм 4.3) [19], Дэвидсона [216] или Якоби—Дэвидсона [230].

7.8. Литература и смежные вопросы к главе 7

Помимо публикаций, упоминавшихся в разд. 7.6 и 7.7, отметим еще наличие ряда хороших обзоров по алгоритмам для разреженных задач на собственные значения (см. [17, 51, 125, 163, 197, 213, 262]). Параллельные реализации алгоритмов обсуждаются в [76].

В разд. 6.2 говорилось о существовании сетевого помощника для выбора между многочисленными итерационными методами решения системы $Ax = b$. Аналогичный проект разрабатывается в настоящее время для задач на собственные значения. Он будет описан в будущем издании этой книги.

7.9. Вопросы к главе 7

Вопрос 7.1 (*легкий*). Проверить, что алгоритм Арнольди (алгоритм 6.9) или алгоритм Ланцоша (алгоритм 6.10), примененные к матрице A и начальному вектору q, порождают ту же матрицу Хессенберга H_k (или трехдиагональную матрицу T_k), что и те же алгоритмы, примененные к матрице Q^TAQ и начальному вектору Q^Tq.

Вопрос 7.2 (*средней трудности*). Пусть λ_i — простое собственное значение матрицы A, а q_1 ортогонален соответствующему собственному вектору. Доказать, что в точной арифметике собственные значения трехдиагональных матриц T_k, вычисляемых алгоритмом Ланцоша, не могут сходиться к λ_i в том смысле, что наибольшая из вычисленных матриц T_k не может иметь собственного значения λ_i. С помощью примера 3×3 показать, что собственное значение некоторой промежуточной матрицы T_k может «случайно» оказаться равно λ_i.

Вопрос 7.3 (*средней трудности*). Проверить, что симметричная трехдиагональная матрица T_k, вычисляемая алгоритмом Ланцоша, не может иметь (в точности) кратных собственных значений. Показать, что если у матрицы A есть кратное собственное значение, то алгоритм Ланцоша, примененный к A, должен завершиться досрочно.

Список литературы

[1] R. Agarwal, F. Gustavson, and M. Zubair. Exploiting functional parallelism of POWER2 to design high performance numerical algorithms. *IBM J. Res. Development*, 38:563–576, 1994.

[2] L. Ahlfors. *Complex Analysis.* McGraw-Hill, New York, 1966.

[3*][1] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[4*] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[5] P. R. Amestoy and I. S. Duff. Vectorization of a multiprocessor multifrontal code. *International Journal of Supercomputer Applications*, 3:41–59, 1989.

[6] P. R. Amestoy. Factorization of large unsymmetric sparse matrices based on a multifrontal approach in a multiprocessor environment. Technical Report TH/PA/91/2, CERFACS, Toulouse, France, February 1991. Ph.D. thesis.

[7] A. Anda and H. Park. Fast plane rotations with dynamic scaling. *SIAM J. Matrix Anal. Appl.*, 15:162–174, 1994.

[8] A. Anda and H. Park. Self scaling fast rotations for stiff least squares problems. *Linear Algebra Appl.*, 234:137–162, 1996.

[9] A. Anderson, D. Culler, D. Patterson, and the NOW Team. A case for networks of workstations: NOW. *IEEE Micro*, 15(1):54–64, February 1995.

[10] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide (2nd edition).* SIAM, Philadelphia, PA, 1995.

[11] ANSI/IEEE, New York. *IEEE Standard for Binary Floating Point Arithmetic*, Std 754-1985 edition, 1985.

[12] ANSI/IEEE, New York. *IEEE Standard for Radix Independent Floating Point Arithmetic*, Std 854-1987 edition, 1987.

[13] P. Arbenz and G. Golub. On the spectral decomposition of Hermitian matrices modified by row rank perturbations with applications. *SIAM J. Matrix Anal. Appl.*, 9:40–58, 1988.

[14] M. Arioli, J. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10:165–190, 1989.

[1] Работы, помеченные звездочкой имеют русский перевод, см. с. 421. — *Прим. ред.*

[15] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, UK, 1994.

[16] Z. Bai. Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem. *Math. Comp.*, 62:209–226, 1994.

[17] Z. Bai. Progress in the numerical solution of the nonsymmetric eigenvalue problem. *J. Numer. Linear Algebra Appl.*, 2:219–234, 1995.

[18] Z. Bai, D. Day, and Q. Ye. ABLE: An adaptive block Lanczos method for non-Hermitian eigenvalue problems. Mathematics Dept. Report 95-04, University of Kentucky, May 1995. Submitted to *SIAM J. Matrix Anal. Appl.*

[19] Z. Bai and G. W. Stewart. SRRIT: A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. Computer Science Dept. Report TR 2908, University of Maryland, April 1992. Available as pub/reports for reports and pub/srrit for programs via anonymous ftp from thales.cs.umd.edu.

[20] D. H. Bailey. Multiprecision translation and execution of Fortran programs. *ACM Trans. Math. Software*, 19:288–319, 1993.

[21] D. H. Bailey. A Fortran-90 based multiprecision system. *ACM Trans. Math. Software*, 21:379–387, 1995.

[22] D. H. Bailey, K. Lee, and H. D. Simon. Using Strassen's algorithm to accelerate the solution of linear systems. *J. Supercomputing*, 4:97–371, 1991.

[23] J. Barnes and P. Hut. A hierarchical $O(n \log n)$ force calculation algorithm. *Nature*, 324:446–449, 1986.

[24] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, V. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, PA, 1994. Also available electronically at http://www.netlib.org/templates.

[25] S. Batterson. Convergence of the shifted QR algorithm on 3 by 3 normal matrices. *Numer. Math.*, 58:341–352, 1990.

[26] F. L. Bauer. Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *Z. Angew. Math. Mech.*, 46:409–421, 1966.

[27] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Linear Algebra Appl.*, 105:9–65, 1988.

[28] C. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11:312–322, 1990.

[29] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming*, 1:11–29, 1992. Software available at http://www.mcs.anl.gov/adifor/.

[30] C. Bischof and G. Quintana-Orti. Computing rank-revealing QR factorizations of dense matrices. Argonne Preprint ANL-MCS-P559-0196, Argonne National Laboratory, Argonne, IL, 1996.

[31] Å. Björck. *Solution of Equations in* $\mathbb{R}^n$, volume 1 of *Handbook of Numerical Analysis*, chapter Least Squares Methods. Elsevier/North Holland, Amsterdam, 1987.

[32] Å. Björck. Least squares methods. Mathematics Department Report, Linköping University, 1991.

[33] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, PA, 1996.

[34] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide.* Software, Environments, and Tools 4. SIAM, Philadelphia, PA, 1997.

[35] J. Blue. A portable FORTRAN program to find the Euclidean norm of a vector. *ACM Trans. Math. Software*, 4:15–23, 1978.

[36] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, I. *Math. Comp.*, 47:103–134, 1986.

[37] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. An iterative method for elliptic problems on regions partitioned into substructures. *Math. Comp.*, 46:361–369, 1986.

[38] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, II. *Math. Comp.*, 49:1–16, 1987.

[39] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, III. *Math. Comp.*, 51:415–430, 1988.

[40] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, IV. *Math. Comp.*, 53:1–24, 1989.

[41] K. Brenan, S. Campbell, and L. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* North Holland, New York, 1989.

[42] C. Brezinski, M. Redivo Zaglia, and H. Sadok. Avoiding breakdown and near-breakdown in Lanczos type algorithms. *Numer. Algorithms*, 1:261–284, 1991.

[43] W. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, PA, 1987.

[44] J. Bunch and L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, 31:163–179, 1977.

[45] J. Bunch, P. Nielsen, and D. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31:31–48, 1978.

[46] B. Buzbee, F. Dorr, J. George, and G. Golub. The direct solution of the

discrete Poisson equation on irregular regions. *SIAM J. Numer. Anal.*, 8:722–736, 1971.

[47] B. Buzbee, G. Golub, and C. Nielsen. On direct methods for solving Poisson's equation. *SIAM J. Numer. Anal.*, 7:627–656, 1970.

[48] T. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[49] T. Chan and T. Mathew. Domain decomposition algorithms. In A. Iserles, editor, *Acta Numerica, Volume* 3. Cambridge University Press, Cambride, UK, 1994.

[50] S. Chandrasekaran and I. Ipsen. On rank-revealing factorisations. *SIAM J. Matrix Anal. Appl.*, 15:592–622, 1994.

[51] F. Chatelin. *Eigenvalues of Matrices.* Wiley, Chichester, England, 1993. English translation of the original 1988 French edition.

[52] F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations.* SIAM, Philadelphia, PA, 1996.

[53] J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK: A portable linear algebra library for distributed memory computers—Design issues and performance. Computer Science Dept. Technical Report CS-95-283, University of Tennessee, Knoxville, TN, March 1995. (LAPACK Working Note 95.)

[54] J. Coonen. Underflow and the denormalized numbers. *Computer*, 14:75–87, 1981.

[55] J. Cullum, W. Kerner, and R. Willoughby. A generalized nonsymmetric Lanczos procedure. *Comput. Phys. Comm.*, 53:19–48, 1989.

[56] J. Cullum and R. Willoughby. A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices. In J. Cullum and R. Willoughby, editors, *Large Scale Eigenvalue Problems.* North Holland, Amsterdam, 1986. Mathematics Studies Series Vol. 127, Proceedings of the IBM Institute Workshop on Large Scale Eigenvalue Problems, July 8–12, 1985, Oberlech, Austria.

[57] J. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations.* Birkhaüser, Basel, 1985. Vol. 1, Theory, Vol. 2, Program.

[58] J. J. M. Cuppen. The singular value decomposition in product form. *SIAM J. Sci. Statist. Comput.*, 4:216–221, 1983.

[59] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[60] E. Davidson. The iteration calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices. *J. Comp. Phys.*, 17:87–94, 1975.

[61] P. Davis. *Interpolation and Approximation.* Dover, New York, 1975.

[62] T. A. Davis and I. S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. Technical Report RAL 93-036, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, UK, 1994.

[63] T. A. Davis and I. S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. Technical Report TR-95-020, Computer and Information Sciences Department, University of Florida, 1995.

[64] D. Day. *Semi-duality in the two-sided Lanczos algorithm.* Ph.D. thesis, University of California, Berkeley, CA, 1993.

[65] D. Day. How the QR algorithm fails to converge and how to fix it. Technical Report 96-0913J, Sandia National Laboratory, Albuquerque, NM, April 1996.

[66] A. Deichmoller. *Über die Berechnung verallgemeinerter singulärer Werte mittles Jacobi-ähnlicher Verfahren.* Ph.D. thesis, Fernuniversität-Hagen, Hagen, Germany, 1991.

[67] P. Deift, J. Demmel, L.-C. Li, and C. Tomei. The bidiagonal singular values decomposition and Hamiltonian mechanics. *SIAM J. Numer. Anal.*, 28:1463–1516, 1991. (LAPACK Working Note 11.)

[68] P. Deift, T. Nanda, and C. Tomei. ODEs and the symmetric eigenvalue problem. *SIAM J. Numer. Anal.*, 20:1–22, 1983.

[69] J. Demmel. The condition number of equivalence transformations that block diagonalize matrix pencils. *SIAM J. Numer. Anal.*, 20:599–610, 1983.

[70] J. Demmel. Underflow and the reliability of numerical software. *SIAM J. Sci. Statist. Comput.*, 5:887–919, 1984.

[71] J. Demmel. On condition numbers and the distance to the nearest ill-posed problem. *Numer. Math.*, 51:251–289, 1987.

[72] J. Demmel. The componentwise distance to the nearest singular matrix. *SIAM J. Matrix Anal. Appl.*, 13:10–19, 1992.

[73] J. Demmel, I. Dhillon, and H. Ren. On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic. *Electronic Trans. Numer. Anal.*, 3:116–140, December 1995. (LAPACK Working Note 70.)

[74] J. Demmel and W. Gragg. On computing accurate singular values and eigenvalues of acyclic matrices. *Linear Algebra Appl.*, 185:203–218, 1993.

[75] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. Technical Report CSD-97-934, Computer Science Division, University of California, Berkeley, CA, February 1997. LAPACK Working Note 119. Submitted to *Linear Algebra Appl.*

[76] J. Demmel, M. Heath, and H. van der Vorst. Parallel numerical linear algebra. In A. Iserles, editor, *Acta Numerica, Volume* 2. Cambridge University Press, Cambridge, UK, 1993.

[77] J. Demmel and N. J. Higham. Stability of block algorithms with fast Level 3 BLAS. *ACM Trans. Math. Software*, 18:274–291, 1992.

[78] J. Demmel and B. Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Matrix Anal. Appl.*, 9:126–145, 1988.

[79] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications. Parts I and II. *ACM Trans. Math. Software*, 19:160–201, June 1993.

[80] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, 11:873–912, 1990.

[81] J. Demmel and X. Li. Faster numerical algorithms via exception handling. *IEEE Trans. Comput.*, 43:983–992, 1994. (LAPACK Working Note 59.)

[82] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13:1204–1246, 1992. (LAPACK Working Note 15.)

[83] I. S. Dhillon. *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem.* Ph.D. thesis, Computer Science Division, University of California, Berkeley, May 1997.

[84] P. Dierckx. *Curve and Surface Fitting with Splines.* Oxford University Press, Oxford, UK, 1993.

[85] J. Dongarra. Performance of various computers using standard linear equations software. Computer Science Dept. Technical Report, University of Tennessee, Knoxville, April 1996. Up-to-date version available at NETLIB/benchmark.

[86] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:18–28, 1990.

[87] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[88] J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. Algorithm 656: An extended set of FORTRAN Basic Linear Algebra Subroutines. *ACM Trans. Math. Software*, 14:18–32, 1988.

[89] J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subroutines. *ACM Trans. Math. Software*, 14:1–17, 1988.

[90] J. Dongarra and D. Sorensen. A fully parallel algorithm for the symmetric eigenproblem. *SIAM J. Sci. Statist. Comput.*, 8:139–154, 1987.

[91] C. Douglas. MGNET: Multi-Grid net. http://NA.CS.Yale.EDU/mgnet/www/mgnet.html.

[92] Z. Drmač. *Computing the Singular and the Generalized Singular Values.* Ph.D. thesis, Fernuniversität-Hagen, Hagen, Germany, 1994.

[93] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, London, 1986.

[94] I. S. Duff. Sparse numerical linear algebra: Direct methods and preconditioning. Technical Report RAL-TR-96-047, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, UK, 1996.

[95] I. S. Duff and J. K. Reid. MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems. Technical Report RAL-95-001, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, UK, 1995.

[96] I. S. Duff and J. K. Reid. The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Trans. Math. Software*, 22:187–226, 1996.

[97] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. Math. Software*, 9:302–325, 1983.

[98] I. S. Duff and J. A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Trans. Math. Software*, 22:30–45, 1996.

[99] A. Edelman. The complete pivoting conjecture for Gaussian elimination is false. *The Mathematica Journal*, 2:58–61, 1992.

[100] A. Edelman and H. Murakami. Polynomial roots from companion matrices. *Math. Comp.*, 64:763–776, 1995.

[101] S. Eisenstat and I. Ipsen. Relative perturbation techniques for singular value problems. *SIAM J. Numer. Anal.*, 32:1972–1988, 1995.

[102] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:315–339, 1984.

[103] D. M. Fenwick, D. J. Foley, W. B. Gist, S. R. VanDoren, and D. Wissel. The AlphaServer 8000 series: High-end server platform development. *Digital Technical Journal*, 7:43–65, 1995.

[104] K. Fernando and B. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67:191–229, 1994.

[105] V. Fernando, B. Parlett, and I. Dhillon. A way to find the most redundant equation in a tridiagonal system. Berkeley Mathematics Dept. Preprint, 1995.

[106] H. Flaschka. *Dynamical Systems, Theory and Applications*, volume 38 of Lecture Notes in Physics, chapter Discrete and periodic solutions of some aspects of the inverse method. Springer-Verlag, New York, 1975.

[107] R. Freund, G. Golub, and N. Nachtigal. Iterative solution of linear systems. In A. Iserles, editor, *Acta Numerica* 1992, pages 57–100. Cambridge University Press, Cambridge, UK, 1992.

[108] R. Freund, M. Gutknecht, and N. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, 14:137–158, 1993.

[109] X. Sun, G. Quintana-Orti, and C. Bischof. A blas-3 version of the QR factorization with column pivoting. Argonne Preprint MCS-P551-1295, Argonne National Laboratory, Argonne, IL, 1995.

[110*] F. Gantmacher. *The Theory of Matrices, vol.* II *(translation).* Chelsea, New York, 1959.

[111*] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

[112] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[113] A. George, M. Heath, J. Liu, and E. Ng. Solution of sparse positive definite systems on a shared memory multiprocessor. *Internat. J. Parallel Programming*, 15:309–325, 1986.

[114*] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[115] A. George and E. Ng. Parallel sparse Gaussian elimination with partial pivoting. *Ann. Oper. Res.*, 22:219–240, 1990.

[116] R. Glowinski, G. Golub, G. Meurant, and J. Periaux, editors. *Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1988. Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Paris, France, January 1987.

[117] S. Goedecker. Remark on algorithms to find roots of polynomials. *SIAM J. Sci. Statist. Comp.*, 15:1059–1063, 1994.

[118] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials.* Academic Press, New York, 1982.

[119] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23:5–48, 1991.

[120] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal. (Series B)*, 2:205–224, 1965.

[121*] G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[122] N. Gould. On growth in Gaussian elimination with complete pivoting. *SIAM J. Matrix Anal. Appl.*, 12:354–361, 1991. See also editor's note in *SIAM J. Matrix Anal. Appl.*, 12(3), 1991.

[123] A. Greenbaum and Z. Strakos. Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.*, 13:121–137, 1992.

[124] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.

[125] R. Grimes, J. Lewis, and H. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15:228–272, 1994.

[126] M. Gu. *Numerical Linear Algebra Computations.* Ph.D. thesis, Dept. of Computer Science, Yale University, November 1993.

[127] M. Gu and S. Eisenstat. A stable algorithm for the rank-1 modification of the symmetric eigenproblem. Computer Science Dept. Report YALEU/DCS/RR-916, Yale University, September 1992.

[128] M. Gu and S. Eisenstat. An efficient algorithm for computing a rank-revealing QR decomposition. Computer Science Dept. Report YALEU/DCS/RR-967, Yale University, June 1993.

[129] M. Gu and S. C. Eisenstat. A stable and efficient algorithm for the rank-1 modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15:1266–1276, 1994. Yale Technical Report YALEU/DCS/RR-916, September 1992.

[130] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix Anal. Appl.*, 16:79–92, 1995.

[131] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, 16:172–191, 1995.

[132] A. Gupta and V. Kumar. Optimally scalable parallel sparse Cholesky factorization. In *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 442–447. SIAM, Philadelphia, PA, 1995.

[133] A. Gupta, E. Rothberg, E. Ng, and B. W. Peyton. Parallel sparse Cholesky factorization algorithms for shared-memory multiprocessor systems. In R. Vichnevetsky, D. Knight, and G. Richter, editors, *Advances in Computer Methods for Partial Differential Equations*—VII. IMACS, 1992.

[134] M. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, Part I. *SIAM J. Matrix Anal. Appl.*, 13:594–639, 1992.

[135] M. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, Part II. *SIAM J. Matrix Anal. Appl.*, 15:15–58, 1994.

[136] W. Hackbusch. *Iterative Solution of Large Sparse Linear Systems of Equations.* Springer-Verlag, Berlin, 1994.

[137*] L. A. Hageman and D. M. Young. *Applied Iterative Methods.* Academic Press, New York, 1981.

[138] W. W. Hager. Condition estimators. *SIAM J. Sci. Statist. Comput.*, 5:311–316, 1984.

[139] P. Halmos. *Finite Dimensional Vector Spaces.* Van Nostrand, New York, 1958.

[140] E. R. Hansen. *Global Optimization Using Interval Analysis.* Marcel Dekker, New York, 1992.

[141] P. C. Hansen. The truncated SVD as a method for regularization. *BIT*, 27:534–553, 1987.

[142] P. C. Hansen. Truncated singular value decomposition solutions to discrete ill-posed problems ill-determined numerical rank. *SIAM J. Sci. Statist. Comput.*, 11:503–518, 1990.

[143] M. T. Heath and P. Raghavan. Performance of a fully parallel sparse solver. In *Proceedings of the Scalable High-Performance Computing Conference*, pages 334–341, IEEE, Los Alamitos, CA, 1994.

[144] M. Hénon. Integrals of the Toda lattice. *Phys. Rev. B*, 9:1421–1423, 1974.

[145] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49:409–436, 1954.

[146] N. J. Higham. A survey of condition number estimation for triangular matrices. *SIAM Rev.*, 29:575–596, 1987.

[147] N. J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. *ACM Trans. Math. Software*, 14:381–396, 1988.

[148] N. J. Higham. Experience with a matrix norm estimator. *SIAM J. Sci. Statist. Comput.*, 11:804–809, 1990.

[149] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, 1996.

[150] P. Hong and C. T. Pan. The rank revealing QR and SVD. *Math. Comp.*, 58:575–232, 1992.

[151] X. Hong and H. T. Kung. I/O complexity: The red blue pebble game. In *Proceedings of the 13th Symposium on the Theory of Computing*, pages 326–334. ACM, New York, 1981.

[152] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[153] E. Jessup and D. Sorensen. A divide and conquer algorithm for computing the singular value decomposition of a matrix. In *Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing*, pages 61–66, SIAM, Philadelphia, PA, 1989.

[154] Z. Jia. *Some Numerical Methods for Large Unsymmetric Eigenproblems.* Ph.D. thesis, Universität Bielefeld, Bielefeld, Germany, 1994.

[155] W.-D. Webber, J. P. Singh, and A. Gupta. Splash: Stanford parallel applications for shared-memory. *Computer Architecture News*, 20:5–44, 1992.

[156] W. Kahan. Accurate eigenvalues of a symmetric tridiagonal matrix. Computer Science Dept. Technical Report CS41, Stanford University, Stanford, CA, July 1966 (revised June 1968).

[157] W. Kahan. A survey of error analysis. In *Information Processing* 71, pages 1214–1239, North Holland, Amsterdam, 1972.

[158] W. Kahan. The baleful effect of computer benchmarks upon applied mathematics, physics and chemistry. http://HTTP.CS.Berkeley.EDU/~wkahan/ieee754status/baleful.ps, 1995.

[159] W. Kahan. Lecture notes on the status of IEEE standard 754 for binary floating point arithmetic. http://HTTP.CS.Berkeley.EDU/~wkahan/ieee754status/ieee754.ps, 1995.

[160] T. Kailath and A. H. Sayed. Displacement structure: Theory and applications. *SIAM Rev.*, 37:297–386, 1995.

[161*] T. Kato. *Perturbation Theory for Linear Operators.* Springer-Verlag, Berlin, 2nd edition, 1980.

[162] R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer, Dordrecht, the Netherlands, 1996. See also http://interval.usl.edu/euromath.html.

[163] W. Kerner. Large-scale complex eigenvalue problems. *J. Comput. Phys.*, 85:1–85, 1989.

[164] G. Kolata. Geodesy: Dealing with an enormous computer task. *Science*, 200:421–422, 1978.

[165] S. Krishnan, A. Narkhede, and D. Manocha. BOOLE: A system to compute Boolean combinations of sculptured solids. Computer Science Dept. Technical Report TR95-008, University of North Carolina, Chapel Hill, 1995. http://www.cs.unc.edu/~geom/geom.html.

[166] M. Kruskal. *Dynamical Systems, Theory and Applications*, volume 38 of Lecture Notes in Physics, chapter Nonlinear Wave Equations. Springer-Verlag, New York, 1975.

[167] K. Kundert. Sparse matrix techniques. In A. Ruehli, editor, *Circuit Analysis, Simulation and Design.* North Holland, Amsterdam, 1986.

[168*] C. Lawson and R. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[169] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic Linear Algebra Subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:308–323, 1979.

[170] P. Lax. Integrals of nonlinear equations of evolution and solitary waves. *Comm. Pure Appl. Math.*, 21:467–490, 1968.

[171] R. Lehoucq. *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration.* Ph.D. thesis, Rice University, Houston, TX, 1995.

[172] R.-C. Li. Solving secular equations stably and efficiently. Computer Science Dept. Technical Report CS-94-260, University of Tennessee, Knoxville, TN, November 1994. (LAPACK Working Note 89.)

[173] T.-Y. Li and Z. Zeng. Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 59:483–502, 1992.

[174] T.-Y. Li and Z. Zeng. Laguerre's iteration in solving the symmetric tridiagonal eigenproblem—a revisit. Michigan State University Preprint, 1992.

[175] T.-Y. Li, Z. Zeng, and L. Cong. Solving eigenvalue problems of nonsymmetric matrices with real homotopies. *SIAM J. Numer. Anal.*, 29:229–248, 1992.

[176] T.-Y. Li, H. Zhang, and X.-H. Sun. Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problem. *SIAM J. Sci. Statist. Comput.*, 12:469–487, 1991.

[177] X. Li. *Sparse Gaussian Elimination on High Performance Computers.* Ph.D. thesis, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, September 1996.

[178] S.-S. Lo, B. Phillipe, and A. Sameh. A multiprocessor algorithm for the symmetric eigenproblem. *SIAM J. Sci. Statist. Comput.*, 8:155–165, 1987.

[179] K. Löwner. Über monotone matrixfunctionen. *Math. Z.*, 38:177–216, 1934.

[180] R. Lucas, W. Blank, and J. Tieman. A parallel solution method for large sparse systems of equations. *IEEE Trans. Computer Aided Design*, CAD-6:981–991, 1987.

[181] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves i: simple intersections. *ACM Transactions on Graphics*, 13:73–100, 1994.

[182] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves ii: Higher order intersections. *Computer Vision, Graphics and Image Processing: Graphical Models and Image Processing*, 57:80–100, 1995.

[183] R. Mathias. Accurate eigensystem computations by Jacobi methods. *SIAM J. Matrix Anal. Appl.*, 16:977–1003, 1996.

[184] The MathWorks, Inc., Natick, MA. *MATLAB Reference Guide*, 1992.

[185] S. McCormick, editor. *Multigrid Methods*, volume 3 of SIAM Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1987.

[186] S. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations*, volume 6 of SIAM Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1989.

[187] J. Moser. *Dynamical Systems, Theory and Applications*, volume 38 of Lecture Notes in Physics, chapter Finitely many mass points on the line under the influence of an exponential potential—an integrable system. Springer-Verlag, New York, 1975.

[188] J. Moser, editor. *Dynamical Systems, Theory and Applications*, volume 38 of Lecture Notes in Physics. Springer-Verlag, New York, 1975.

[189] A. Netravali and B. Haskell. *Digital Pictures.* Plenum Press, New York, 1988.

[190] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge, UK, 1990.

[191] E. G. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM J. Sci. Statist. Comp.*, 14:1034–1056, 1993.

[192] B. Nour-Omid, B. Parlett, and A. Liu. How to maintain semi-orthogonality among Lanczos vectors. CPAM Technical Report 420, University of California, Berkeley, CA, 1988.

[193] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right hand sides. *Numer. Math.*, 6:405–409, 1964.

[194] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[195] V. Pan. How can we speed up matrix multiplication. *SIAM Rev.*, 26:393–416, 1984.

[196] V. Pan and P. Tang. Bounds on singular values revealed by QR factorization. Technical Report MCS-P332-1092, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992.

[197*] B. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, NJ, 1980.

[198] B. Parlett. Misconvergence in the Lanczos algorithm. In M. G. Cox and S. Hammarling, editors, *Reliable Numerical Computation*, chapter 1. Clarendon Press, Oxford, UK, 1990.

[199] B. Parlett. Reduction to tridiagonal form and minimal realizations. *SIAM J. Matrix Anal. Appl.*, 13:567–593, 1992.

[200] B. Parlett. *Acta Numerica*, The new qd algorithms, pages 459–491. Cambridge University Press, Cambridge, UK, 1995.

[201] B. Parlett. The construction of orthogonal eigenvectors for tight clusters by use of submatrices. Center for Pure and Applied Mathematics PAM-664, University of California, Berkeley, CA, January 1996. Submitted to *SIAM J. Matrix Anal. Appl.*

[202] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[203] B. N. Parlett and I. S. Dhillon. Fernando's solution to Wilkinson's problem: An application of double factorization. *Linear Algebra Appl.*, 1997. To appear.

[204] D. Priest. Algorithms for arbitrary precision floating point arithmetic. In P. Kornerup and D. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–145, Grenoble, France, June 26–28, 1991. IEEE Computer Society Press, Los Alamitos, CA.

[205] A. Quarteroni, editor. *Domain Decomposition Methods*, AMS, Providence, RI, 1993. Proceedings of the Sixth International Symposium on Domain Decomposition Methods, Como, Italy, 1992.

[206] H. Ren. *On Error Analysis and Implementation of Some Eigenvalue and Singular Value Algorithms.* Ph.D. thesis, University of California at Berkeley, 1996.

[207] E. Rothberg and R. Schreiber. Improved load distribution in parallel sparse Cholesky factorization. In *Supercomputing*, pages 783–792, November 1994.

[208] S. Rump. Bounds for the componentwise distance to the nearest singular matrix. *SIAM J. Matrix Anal. Appl.*, 18:83–103, 1997.

[209] H. Rutishauser. *Lectures on Numerical Mathematics.* Birkhäuser, Basel, 1990.

[210] J. Rutter. A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem. Mathematics Dept. Master's Thesis, University of California, 1994. Available by anonymous ftp from tr-ftp.cs.berkeley.edu, directory pub/tech-reports/csd/csd-94-799, file all.ps.

[211] Y. Saad. Krylov subspace methods for solving large unsymmetric linear system. *Math. Comp.*, 37:105–126, 1981.

[212] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm.*, 53:71–90, 1989.

[213] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester, UK, 1992.

[214] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996.

[215] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[216] M. Sadkane. Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems. *Numer. Math.*, 64:195–211, 1993.

[217] M. Sadkane. A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices. *Numer. Math.*, 64:181–193, 1993.

[218] J. R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. Technical Report CMU-CS-96-140, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1996.

[219] M. Shub and S. Smale. Complexity of Bezout's theorem I: Geometric aspects. *J. Amer. Math. Soc.*, 6:459–501, 1993.

[220] M. Shub and S. Smale. Complexity of Bezout's theorem II: Volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Progress in Mathematics, Vol. 109—Computational Algebraic Geometry.* Birkhäuser, Basel, 1993.

[221] M. Shub and S. Smale. Complexity of Bezout's theorem III: Condition number and packing. *J. Complexity*, 9:4–14, 1993.

[222] M. Shub and S. Smale. Complexity of Bezout's theorem IV: Probability of success; extensions. Mathematics Department Preprint, University of California, 1993.

[223] SGI Power Challenge. Technical Report, Silicon Graphics, 1995.

[224] H. Simon. The Lanczos algorithm with partial reorthogonalization. *Math. Comp.*, 42:115–142, 1984.

[225] R. D. Skeel. Scaling for numerical stability in Gaussian elimination. *Journal of the ACM*, 26:494–526, 1979.

[226] R. D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comp.*, 35:817–832, 1980.

[227] R. D. Skeel. Effect of equilibration on residual size for partial pivoting. *SIAM J. Numer. Anal.*, 18:449–454, 1981.

[228] I. Slapničar. *Accurate Symmetric Eigenreduction by a Jacobi Method.* Ph.D. thesis, Fernuniversität-Hagen, Hagen, Germany, 1992.

[229] G. Sleijpen and H. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. Dept. of Mathematics Report 856, University of Utrecht, 1994.

[230] G. Sleijpen, A. Booten, D. Fokkema, and H. van der Vorst. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems, Part I. Dept. of Mathematics Report 923, University of Utrecht, 1995.

[231] B. Smith. Domain decomposition algorithms for partial differential equations of linear elasticity. Technical Report 517, Department of Computer Science, Courant Institute, September 1990. Ph.D. thesis.

[232] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations.* Cambridge

University Press, Cambridge, UK, 1996. Corresponding PETSc software available at http://www.mcs.anl.gov/petsc/petsc.html.

[233] D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[234] D. Sorensen and P. Tang. On the orthogonality of eigenvectors computed by divide-and-conquer techniques. *SIAM J. Numer. Anal.*, 28:1752–1775, 1991.

[235] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[236] G. W. Stewart. Rank degeneracy. *SIAM J. Sci. Statist. Comput.*, 5:403–413, 1984.

[237] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

[238] SPARCcenter 2000 architecture and implementation. Sun Microsystems, Inc., November 1993. Technical White Paper.

[239] W. Symes. The QR algorithm for the finite nonperiodic Toda lattice. *Phys. D*, 4:275–280, 1982.

[240] G. Szegö. *Orthogonal Polynomials.* AMS, Providence, RI, 1967.

[241] K.-C. Toh and L. N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numer. Math.*, 68:403–425, 1994.

[242] L. Trefethen and R. Schreiber. Average case analysis of Gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 11:335–360, 1990.

[243] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[244] A. Van Der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969.

[245] A. F. van der Stappen, R. H. Bisseling, and J. G. G. van der Vorst. Parallel sparse LU decomposition on a mesh network of transputers. *SIAM J. Matrix Anal. Appl.*, 14:853–879, 1993.

[246] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Linear Algebra Appl.*, 27:103–141, 1979.

[247] P. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Automat. Control*, AC-26:111–128, 1981.

[248] C. V. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[249] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[250] K. Veselić and I. Slapničar. Floating point perturbations of Hermitian matrices. *Linear Algebra Appl.*, 195:81–116, 1993.

[251*] V. Voevodin. The problem of non-self-adjoint generalization of the conjugate gradient method is closed. *Comput. Math. Math. Phys.*, 23:143–144, 1983.

[252] D. Watkins. *Fundamentals of Matrix Computations.* Wiley, Chichester, UK, 1991.

[253] The Cray C90 series. http://www.cray.com/PUBLIC/product-info/C90/. Cray Research, Inc.

[254] The Cray J90 series. http://www.cray.com/PUBLIC/product-info/J90/. Cray Research, Inc.

[255] The Cray T3E series. http://www.cray.com/PUBLIC/product-info/T3E/. Cray Research, Inc.

[256] The IBM SP-2. http://www.rs6000.ibm.com/software/sp_products/sp2.html. IBM.

[257] The Intel Paragon. http://www.ssd.intel.com/homepage.html. Intel.

[258] P.-Å. Wedin. Perturbation theory for pseudoinverses. *BIT*, 13:217–232, 1973.

[259] S. Weisberg. *Applied Linear Regression.* Wiley, Chichester, UK, 2nd edition, 1985.

[260] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, Chichester, UK, 1992.

[261] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice Hall, Englewood Cliffs, NJ, 1963.

[262] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford, UK, 1965.

[263] S. Winograd and D. Coppersmith. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing*, pages 1–6. ACM, New York, 1987.

[264] M. Wolfe. *High Performance Compilers for Parallel Computing.* Addison-Wesley, Reading, MA, 1996.

[265] Q. Ye. A convergence analysis for nonsymmetric Lanczos algorithms. *Math. Comp.*, 56:677–691, 1991.

[266] Q. Ye. A breakdown-free variation of the nonsymmetric Lanczos algorithm. *Math. Comp.*, 62:179–207, 1994.

[267] D. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

[268] H. Yserentant. Old and new convergence proofs for multigrid methods. In A. Iserles, editor, *Acta Numerica* 1993, pages 285–326. Cambridge University Press, Cambrigde, UK, 1993.

[269] Z. Zeng. *Homotopy-Determinant Algorithm for Solving Matrix Eigenvalue Problems and Its Parallelizations.* Ph.D. thesis, Michigan State University, East Lansing, MI, 1991.

[270] Z. Zlatev, J. Waśniewski, P. C. Hansen, and Tz. Ostromsky. PARASPAR: a package for the solution of large linear algebraic equations on parallel computers with shared memory. Technical Report 95-10, Technical University of Denmark, Lyngby, September 1995.

[271] Z. Zlatev. *Computational Methods for General Sparse Matrices.* Kluwer Academic, Dordrecht, Boston, 1991.

Работы на русском языке

Здесь приведены работы из списка литературы, имеющиеся на русском языке.

[3] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.

[4] Алефельд Г., Херцбергер Дж. Введение в интервальные вычисления. — М.: Мир, 1987.

[110] Гантмахер Ф. Р. Теория матриц, Изд. второе, дополн. — М.: Наука, 1966.

[111] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982.

[114] Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. — М.: Мир, 1984.

[121] Голуб Дж., Ван Лоун Ч. Матричные вычисления. — М.: Мир, 1999.

[137] Хейгеман Л., Янг Д. Прикладные итерационные методы. — М.: Мир, 1986.

[161] Като Т. Теория возмущений линейных операторов. — М.: Мир, 1972. (Перевод с 1-го английского издания.)

[168] Лоусон К., Хансон Р. Численное решение задач метода наименьших квадратов. — М.: Наука, 1986.

[197] Парлетт Б. Симметричная проблема собственных значений: Численные методы. — М.: Мир, 1983.

[251] Воеводин В. В. Журнал вычислительной математики и математической физики, т. 23, с. 143—144, 1983.

Предметный указатель

2-норма (two-norm) 10
p-норма (p-norm) 29
IEEE-стандарт двоичной арифметики (IEEE standard for binary arithmetic) 19
L-образная область (L-shaped region) 363
max-норма (max-norm) 31
QR-алгоритм с выявлением ранга (rank-revaling QR algorithm) 142
- - неявным сдвигом (implicit shift QR algorithm) 149
QR-итерация (QR iteration) 165, 172
- со сдвигом (with shift) 173
- трехдиагональная (tridiagonal) 225
QR-поток (QR flow) 271
QR-разложение (QR decomposition) 111, 117
- с выбором главного столбца (with column pivoting) 141, 142
QZ-алгоритм (QZ algorithm) 191
RQ-итерация (RQ iteration) 226

Алгоритм Арнольди (Arnoldi's algorithm) 318, 402
- бисекции (bisection) 223
- Грама-Шмидта классический (classical Gram-Schmidt) 117
- - модифицированный (modified Gram-Schmidt) 117
- итерационный (iterative) 278
- Ланцоша (Lanczos) 317
- - без переортогонализации (Lanczos algorithm with no reorthogonalization) 382
- - несимметричный (nonsymmetric Lanczos algorithm) 402
- - с выборочной переортогонализацией (Lanczos algorithm with selective reorthogonalization) 398
- - - полной переортогонализацией (Lanczos algorithm with full reorthogonalization) 382
- матричного умножения Штрассена (Strassen's matrix multiplication) 81
- минимальной степени (minimum degree ordering) 100
- обратный Катхилл-Макки (reverse Cuthill-McKee) 99
- последовательный (serial) 100
- «разделяй и властвуй» (divide-and-conquer) 228
- с разделяемой памятью (shared-memory) 100
- - распределенной памятью (distributed-memory) 100
- сопряженных градиентов (conjugate gradients) 325
- среднего пути (the middle way) 236
- Холесского ленточный (band Cholesky) 92
- Якоби классический (classical Jacobi's) 247
аппроксимация данных (curve fitting) 111
A-сопряженные векторы (A-conjugate vectors) 321

Блочная циклическая редукция (block cyclic reduction) 343
блочное предобуславливание Якоби (block Jacobi preconditioning) 332
- разбиение (blocking) 79
быстрое синус-прелбразование (fast sine transform) 338
быстрые вращения (fast givens rotations) 133

Ведущая главная подматрица (leading principal submatrix) 48
векторы Арнольди (Arnoldi's vectors) 318
- Ланцоша (Lanczos) 318
- Ритца (Ritz) 378
- сингулярные левые (left singular) 119
- - правые (right singular) 119
верхняя форма Хессенберга (upper Hessenberg form) 165, 176
взвешенные наименьшие квадраты (weighted least squares) 145
вращение Гивенса (Givens rotation) 129
- Якоби (Jacobi) 246

вспомогательный разряд (guard digit) 239
входное управление (control input) 194
выступ (горб) (bilge) 181
вычисление многочлена (polynomial evaluation) 15
вычислительная геометрия (computational geometry) 196

Гауссово исключение (Gaussian elimination) 47
- - с полным выбором главного элемента (with complete pivoting) (GECP) 50
- - - частичным выбором главного элемента (with partial pivoting) (GEPP) 50
геодезическое моделирование (geodesic modeling) 111
градиент (gradient) 323
граф двудольный (bipartite graph) 299
- матрицы (graph of the matrix) 299
- ориентированный (directed) 301
- - матрицы (directed graph of the matrix) 301
- - сильно связанный (strongly connected directed) 301

Двоично-инверсный порядок (bit-reversed order) 341
декомпозиция области (domain decomposition) 332
демпфирование критическое (critically damped system) 155
- сильное (overdamped system) 155
- слабо затухающее (underdamped system) 155
дефляция (deflation) 232
диагональный предобуславливатель (diagonal preconditioner) 332
дополнение Шура (Schur complement) 109, 365

Жорданов блок (Jordan block) 151
жорданова каноническая форма (Jordan canonical form) 11

Задача корректная (well-posed problem) 26
- линейная наименьших квадратов (linear least squares) 111, 112
- - - - недоопределенная (underdetermined) 111, 147
- - - - переопределенная (overdetermined) 111
- модельная (model) 278
- наименьших квадратов с ограничениями (constrained least squares) 148
- на сингулярные значения (singular value) 10
- - собственные значения (eigenvalue) 10
- некорректная (ill-posed) 26

Импульсный отклик (impulse response) 187, 190
инвариантное подпространство (invariant subspace) 155
индикатор особого случая (exception flag) 20
инерция (inertia) 213
интервальная арифметика (interval arithmetic) 23
итерация обратная (inverse iteration) 165, 167, 240
- одновременная (simultanious) 168
- ортогональная (orthogonal) 165, 168
итерирование подпространства (subspace iteration) 168

Каноническая форма Вейерштрасса (Weierstrass canonical form) 188
- - Кронекера (Kronecker) 192
- - матрицы (canonical form of matrix) 150
- - - Жордана (Jordan) 150
- - - Шура (Schur) 150
каноническое разложение Жордана (Jordan canonical factorization) 11
конечные разности (finite differences) 280
коэффициент роста (growth factor) 58
крыловское подпространство (Krylov subspace) 318, 377

Линейная модель (linear model) 113
- регрессия (regression) 113

Малое отосительное покомпонентное возмущение матрицы (small componentwise relative perturbation) 45
матрица блочно-треугольная (block-triangular matrix) 150
- Вандермонда (Vandermonde) 102
- верхняя двухдиагональная (upper bidiagonal) 178
- Гильберта (Hilbert) 103

- двухдиагональная (bidiagonal) 208
- демпфирования (damping) 153
- дефектная (defective) 152
- диагонализуемая (diagonalizable) 152
- диагональная (diagonal) 150
- жесткости (stiffness) 153
- итерационная метода Якоби (iteration matrix for Jacobi method) 305
- квазитреугольная (quasi-triangular) 151
- ковариационная (covariance) 113
- комплексная эрмитова (complex Hermitian) 30
- Коши (Cauchy) 102
- ленточная (band) 89
- масс (mass) 153
- Мура-Пенроуза (Moore-Penrose) - см. матрица псевдообратная 127, 137
- неразложимая (irreducible) 301
- неразложимая верхняя Хессенбергова (unreduced upper Hessenberg) 178
- ортогональная (orthogonal) 31
- отраженная (flipped) 272
- перестановок (permutation) 11, 47
- плохо масштабированная (badly scaled) 46
- псевдообратная (pseudoinverse) 127
- разреженная (sparse) 93
- симметричная положительно определенная (symmetric positive definite) 30, 87
- согласованно упорядоченная (consistent ordering) 305
- сопровождающая (companion) 195, 315
- - блочная (block companion) 195
- стреловидная (arrow) 94
- строго верхнетреугольная (strictly upper triangular) 33
- тёплицева (Toeplitz) 103
- треугольная (triangular) 150
- трехдиагональная (tridiagonal) 208
- унитарная (unitary) 31

машинная точность (machine precision) 20
машинный нуль (underflow) 18
машинный эпсилон (machine epsilon) 20
метод FMM (=быстрый многополюсный метод) (fast multipole method) 240
- бисопряженных градиентов (biconjugate gradients) 335
- Гаусса-Зейделя (Gauss-Seidel) 295
- - блочный с перекрытием (overlapping block Gauss-Seidel) 369
- дополнения Шура (Schur complement) 363
- квазиминимальных невязок (quasi-minimum residuals) 335
- Ланцоша несимметричный (nonsymmetric Lanczos) 334
- - с заглядыванием вперед (method with look-ahead) 335, 402
- многосеточный полный (FMG) (full multigrid) 350
- наискорейшего подъема (hill climbing) 62
- односторонних вращений Якоби (one-sided Jacobi rotation) 262
- подструктур (substructuring) 363
- симметричный SOR (SSOR) (symmetric SOR) 312
- сопряженных градиентов (conjugate gradients) 320
- степенной (power) 165
- Шварца аддитивный (additive Schwarz) 368
- - - двухуровневый (two-level additive Schwarz) 370
- - мультипликативный (multiplicative Schwarz) 369
- Якоби (Jacobi's) 244
- - взвешенный (weighted Jacobi's) 352
- - для двумерного уравнения Пуассона (Jacobi method for two-dimensional Poisson's equation) 294
- - строчный циклический (cyclic-by-row-Jacobi) 247

методы без перекрытия (nonoverlapping methods) 363
- вычисления собственных значений итерационные (iterative method for eigenproblem) 149
- - - - прямые (direct method for eigenproblem) 149
- итерационные (iterative) 40, 149
- крыловского подпространства (Krylov subspace) 313
- прямые (direct methods) 40, 149, 278
- с перекрытием (overlapping) 366

механические колебания (mechanical vibrations) 152, 208
многосеточные методы (multigrid) 345

многосеточный V-цикл (multigrid V-cycle) 348
многочлен матричный (matrix polynomial) 194
многочлены Чебышева (Chebyshev polynomials) 309
множитель Холесского (Cholesky factor) 87
модификация ранга 1 (rank-1 update) 82

Направление градиентного поиска (gradient search direction) 321
невязка (residual) 43, 145
нелинейная проблема собственных значений (nonlinear eigenvalue problem) 194
неполный LU предобуславливатель (incomplete LU preconditioner) 332
неразложимость (irreducibility) 299
неявная Q-теорема (implicit Q theorem) 179, 180
неявный QR-алгоритм с двойным сдвигом (implicit double shift QR algorithm) 182
--- одинарным сдвигом (implicit single shift) 181
норма (norm) 30
- операторная (подчиненная, индуцированная) (operator (subordinate, induced)) 31
- Фробениуса (Frobenius) 31
нормальные уравнения (normal equations) 111, 116

Обобщенная форма Шура для регулярных пучков (generalized Schur form for regular pencils) 190
образ матрицы (space of matrix) 121
обратная устойчивость (backward stability) 13
обратный анализ ошибок (backward error analysis) 53
обрыв (поиска) (breakdown) 335
округление до ближайшего четного (rounding to nearest even) 19
- правильное (correctly rounding) 19
оператор интерполяции (interpolation operator) 348, 372
- сглаживания (solution) 348
- сужения (restriction) 348, 370
операция с плавающей точкой (floating point operation) 13
ориентированное ребро (directed edg) 301
ортогонализация выборочная (selective orthogonalization) 392, 398
основные подпрограммы линейной алгебры (BLAS) (Basic Linear Algebra Subroutines) 76
отбор (sample) 356
отделенность собственного значения (gap) 214
--- относительная (relative gap) 220
отложенная модификация подматрицы (delaying the update) 83
отношение Рэлея (Rayleigh quotient) 209
отосительное число обусловленности (relative condition number) 12
--- покомпонентное (componentwise) 46
отражение Хаусхолдера (Householder reflection) 129
оценка наибольшего правдоподобия (maximum-likelihood estimate) 113
оценщик обусловленности (condition estimator) 61
ошибка (погрешность) (error) 13
- абсолютная (absolute) 12
- глобальная (global) 385
- локальная (local) 385
- обратная (backward) 13
- округления (roundoff) 19
- представления отосительная (relative representation) 18
--- максимальная (maximum) 18

Параллельная машина с разделяемой памятью (shared-memory parallel machines) 85
--- распределенной памятью (distributed-memory) 85
параллельный компьютер (parallel computer) 85
параметр релаксации (relaxation parameter) 297
перемежение (interlace) 383
переполнение (overflow) 20
петля (self edg) 301
плохо обусловленная функция в точке x (function ill-conditioned at x) 12
подобие (similarity transformation) 151
подстановка обратная (backward substitution) 11

- прямая (forward) 11
полуширина ленты (semibandwidth) 10
порог машинного нуля (underflow threshold) 19
- переполнения (overflow) 19
поток Тода (Toda flow) 207
правило Фрэнсиса (Francis rule) 184
предобуславливание (preconditioning) 330
- Якоби (Jacobi) 332
предобуславливатель грубой сетки (coarse grid preconditioner) 370
предобусловленный метод сопряженных градиентов (preconditioned conjugate gradient method) 330
преобладание диагональное слабое по строкам (weakly row diagonal dominant) 302
- Кэли (Cayley transform) 277
- строгое диагональное по столбцам (strictly column diagonally dominant) 109
- - - по строкам (strictly) 300
- Фурье быстрое (FFT) (fast Fourier) 337
- - дискретное (DFT) (discrete Fourier) 337
- - - обратное (IDFT) (inverse diskrete Fourier) 337
приближение конечно-разностное (finite difference approximation) 92
приведение к двухдиагональной форме (bidiagonal reduction) 179
- - трехдиагональной (tridiagonal) 178
проектор (projection matrix) 201
- спектральный (spectral projection) 201
проекция матрицы (matrix) 318
произведение кронекерово (Kronecker product) 287
- скалярное (dot) 29
- - стандартное (standard dot) 29
прцедура Рэлея-Ритца (Rayleigh-Ritz procedure) 378
пучок матричный (matrix pencil) 185
- - регулярный (regular) 185
- - сингулярный (singular) 185
- определенный (definite) 191

Разложение Жордана каноническое (Jordan canonical factorization) 11
- матрицы (matrix) 11
- сингулярное (SVD) (singular value decomposition) 111, 119
- - усеченное (truncated SVD) 138
- Холесского (Cholesky) 87
- - неполное (incomplete) 332
- Шура (Schur) 12
размер блока (block size) 83
расщепление матрицы (splitting) 292
ребро (edg) 301
регуляризация (regularization) 135
релаксация верхняя (overrelaxation) 297
- нижняя (underrelaxation) 297
- последовательная верхняя (successive overrelaxation) 297
решетка Тода (Toda lattice) 268

Свертка (convolution) 339
свойство A (property A) 299
сдвиг (shift) 165, 167
- двойной (double) 180
сжатие изображений (image compression) 115, 123
сингулярные числа (singular values) 119
система линейных уравнений (linear system of equations) 10
- недопределенная (underdeterminated) 10
- переопределенная (overdeterminated) 10
- уравнений трехдиагональная (tridiagonal) 92
скорость сходимости процесса (rate of convergence) 294
собственное значение простое (simple eigenvalue) 151
- - пучка (eigenvalue of matrix pencil) 185, 186
собственные функции дифференциального уравнения (eigenfunctions of the differential equation) 282
собственный вектор левый (left eigenvector) 150
- - правый (right) 150
- - пучка левый (left eigenvector of matrix pencil) 186
- - - правый (right eigenvector of matrix pencil) 186
сопряженные градиенты (conjugate gradients) 321
спектральный радиус матрицы (spectral radius of matrix) 292
специальная структура задачи (special structure of problem) 10

статистическое моделирование (statistical modeling) 111
стратегия Уилкинсона (Wilkinson's shift) 225
сходимость квадратичная (quadratic convergence) 174
- ложная (misconvergence) 382

Теорема Вейля (Weyl theorem) 209
- - относительная (relative Weyl) 219
- Гершгорина (Gershgorin's) 93
- Лёвнера (Lowner) 236
- о спектральном отображении (spectral mapping) 200
- Пейджа (Paige's 395
- Сильвестра об инерции (Sylvester's inertia) 213
теоретико-графовые свойства матрицы (grapf properties) 301
теория графов (graph theory) 301
точность дважды двойная (double double precision) 23
- расширенная двойная (double extended) 23
- учетверенная (quadruple) 23

Узел графа (node) 301
упорядочивание естественное (natural ordering) 296
- шахматное (chess) 296
управляемое подпространство (controllable subspace) 194
уравнение вековое (secular equation) 230
- Лапласа (Laplace's equation) 278
- Ляпунова (Lyapunov) 200
- Пуассона (Poisson's) 278, 280
- - двумерное (two-dimensional Poisson's) 283
- - одномерное (one-dimensional) 280
- Сильвестра (Sylvester) 200
уравнения дифференциально-алгебраические (differential-algebreaic equations) 187
- Кортевега-де Фриза (Korteweg-de Vries) 273
уравновешивание (equilibration) 72
условие Галеркина - см. условие ортогональной невязки (Galerkin condition)
- Дирихле (Dirichlet boundary) 280
- ортогональной невязки (orthogonal residual) 319
условия согласования (consistency conditions) 194
усреднение (averaging) 356

Флоп (flop) 13
формула Шермана-Моррисона (Sherman-Morrison formula) 106
- Шермана-Моррисона-Вудбери (Sherman-Morrison-Woodbury) 106

Характеристический многочлен матрицы (characteristic polynomial) 150
- - пучка (characteristic polynomial of pencil) 185

Центрированное разностное приближение (centered difference approximation) 280

Частотная компонента погрешности (frequensy component of the error) 352
- область (domain) 347
чебышевское ускорение (Chebyshev acceleration) 311
числа Ритца (Ritz numbers) 378
численная неустойчивость (numerical instability) 55
число двойной точности (double precision) 19
- нормализованное (normalized number) 18
- обусловленности (condition number) 12
- - абсолютное (absolute) 12
- - матрицы (condition number of matrix) 41
- - относительное (relative) 12
- обычной точности (single precision) 19

Шаблон (template) 15
- уравнения (stencil of the equation) 286
ширина верхняя ленточной матрицы (upper bandwidth) 89
- нижняя ленточной матрицы (lower) 89

Эквивалентные пучки (equivalent pencils) 187

Ядро матрицы (null space) 121

Оглавление

От переводчика ... 5
Предисловие к русскому изданию ... 6
Предисловие ... 7
Глава 1. Введение ... 9
1.1. Основные обозначения ... 9
1.2. Стандартные задачи вычислительной линейной алгебры ... 10
1.3. Общие аспекты ... 11
1.4. Пример: вычисление многочлена ... 15
1.5. Арифметика с плавающей точкой ... 18
1.6. Еще раз о вычислении многочлена ... 24
1.7. Векторные и матричные нормы ... 28
1.8. Литература и смежные вопросы к главе 1 ... 32
1.9. Вопросы к главе 1 ... 33
Глава 2. Решение линейных уравнений ... 40
2.1. Введение ... 40
2.2. Теория возмущений ... 41
2.3. Гауссово исключение ... 47
2.4. Анализ ошибок ... 53
2.5. Улучшение точности приближенного решения ... 70
2.6. Блочные алгоритмы как средство повышения производительности ... 73
2.7. Специальные линейные системы ... 86
2.8. Литература и смежные вопросы к главе 2 ... 103
2.9. Вопросы к главе 2 ... 104
Глава 3. Линейные задачи наименьших квадратов ... 111
3.1. Введение ... 111
3.2. Матричные разложения для решения линейной задачи наименьших квадратов ... 115
3.3. Теория возмущений для задачи наименьших квадратов ... 127
3.4. Ортогональные матрицы ... 129
3.5. Задачи наименьших квадратов неполного ранга ... 135
3.6. Сравнение производительности методов для решения задач наименьших квадратов ... 143
3.7. Литература и смежные вопросы к главе 3 ... 144
3.8. Вопросы к главе 3 ... 144
Глава 4. Несимметричная проблема собственных значений ... 149
4.1. Введение ... 149
4.2. Канонические формы ... 150
4.3. Теория возмущений ... 159
4.4. Алгоритмы для несимметричной проблемы собственных значений ... 165
4.5. Другие типы несимметричных спектральных задач ... 185
4.6. Резюме ... 196
4.7. Литература и смежные вопросы к главе 4 ... 199
4.8. Вопросы к главе 4 ... 199
Глава 5. Симметричная проблема собственных значений и сингулярное разложение ... 206
5.1. Введение ... 206

5.2. Теория возмущений ... 209
5.3. Алгоритмы для симметричной проблемы собственных значений ... 222
5.4. Алгоритмы вычисления сингулярного разложения ... 251
5.5. Дифференциальные уравнения и задачи на собственные значения ... 267
5.6. Литература и смежные вопросы к главе 5 ... 273
5.7. Вопросы к главе 5 ... 274
Глава 6. Итерационные методы для линейных систем ... 278
6.1. Введение ... 278
6.2. Интернет-ресурсы для итерационных методов ... 279
6.3. Уравнение Пуассона ... 280
6.4. Краткая сводка методов для решения уравнения Пуассона ... 289
6.5. Основные итерационные методы ... 292
6.6. Методы крыловского подпространства ... 313
6.7. Быстрое преобразование Фурье ... 335
6.8. Блочная циклическая редукция ... 342
6.9. Многосеточные методы ... 345
6.10. Декомпозиция области ... 362
6.11. Литература и смежные вопросы к главе 6 ... 371
6.12. Вопросы к главе 6 ... 371
Глава 7. Итерационные методы для задач на собственные значения ... 376
7.1. Введение ... 376
7.2. Метод Рэлея—Ритца ... 377
7.3. Алгоритм Ланцоша в точной арифметике ... 381
7.4. Алгоритм Ланцоша в арифметике с плавающей точкой ... 391
7.5. Алгоритм Ланцоша с выборочной ортогонализацией ... 398
7.6. Другие возможности ... 400
7.7. Итерационные алгоритмы для несимметричной проблемы собственных значений ... 402
7.8. Литература и смежные вопросы к главе 7 ... 402
7.9. Вопросы к главе 7 ... 403
Список литературы ... 404
Работы на русском языке ... 421
Предметный указатель ... 422

Учебное издание

Джеймс Деммель

Вычислительная линейная алгебра

Теория и приложения

Заведующий редакцией академик В. И. Арнольд
Зам. зав. редакцией А. С. Попов
Ведущий редактор В. И. Ахмолин
Художник П. Инфанте
Художественный редактор В. П. Григорьев
Технический редактор О. Г. Лапко

Оригинал-макет подготовлен А. А. Пудовым
в пакете LaTeX 2_ε с использованием семейства шрифтов Computer Modern
с кириллическим расширением LH

Лицензия ЛР № 010174 от 20.05.97 г.

Подписано к печати 20.09.2001 г. Формат 70 × 100/16.
Печать офсетная.
Объем 13,50 бум. л. Усл.-печ. л. 35,10. Уч.-изд. л. 33,46.
Изд. № 1/9733. Тираж 3 000 экз. Заказ №271

Издательство «Мир»
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций
107996, ГСП-6, Москва, 1-й Рижский пер., 2.

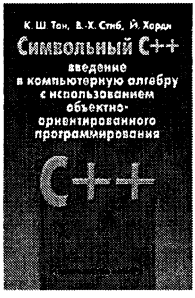
Диапозитивы изготовлены
в издательстве «Мир»

Отпечатано с готовых диапозитивов в
АО «Внешторгиздат»
127576, Москва, Илимская ул., 7.

ЛУЧШИЕ КНИГИ МИРА — В ИЗДАТЕЛЬСТВЕ «МИР»

ИЗДАТЕЛЬСТВО «МИР»

Имеется в продаже:



Тан К. Ш., Стиб В.-Х., Харди Й. **СИМВОЛЬНЫЙ C++: Введение в компьютерную алгебру с использованием объектно-ориентированного программирования:** Пер. со 2-го англ. изд. — 624 с., ил.

В книге представлен подход к разработке новой системы компьютерной алгебры, основанной на объектно-ориентированном программировании. В первых вводных главах излагаются требования пользователя к таким системам, описан необходимый математический аппарат, лежащий в основе разработки системы, дан обзор наиболее популярных из существующих систем (REDUCE, MAPLE, AXIOM, MATEMATICA, MuPAD). Далее обсуждаются основные понятия объектно-ориентированного программирования, языки Java, Eiffel, Smalltalk и Оберон, излагаются средства языка C++, вводится новая система компьютерной алгебры SymbolicC++. Приводятся примеры применения системы к задачам из математики и физики (нумерация Геделя, аппроксимация Паде, техника рядов Ли, метод Пикара, фрактальное множество Мандельброта и др.). В книгу включены листинги всех компонентов системы, что позволяет пользователям развивать и наращивать систему в соответствии со своими потребностями.

Основное содержание книги:

- Математические основы компьютерной алгебры
- Системы компьютерной алгебры
- Объектно-ориентированное программирование
- Основные средства языка C++
- Классы компьютерной алгебры
- Символьный класс
- Приложения
- Язык Лисп и компьютерная алгебра
- Листинги программ
- Параллельная виртуальная машина и абстрактные типы данных
- Техника обработки ошибок
- Программа Gnuplot и язык PostScript

Для студентов и аспирантов вузов в качестве учебного пособия, для специалистов по компьютерной алгебре как справочное пособие, а также для непрофессионалов, желающих расширить свои познания в возможностях вычислительной техники.

Адрес: 107996, ГСП-6, Москва, 1-й Рижский пер., 2
Контактные телефоны: (095)286-83-88, 286-82-33, 286-25-50
Факс: (095)286-84-55 ✧ **E-mail:** victor@mir.msk.su ✧ **http://www.mir-pubs.dol.ru**

ЛУЧШИЕ КНИГИ МИРА — В ИЗДАТЕЛЬСТВЕ «МИР»

Новинка 2001 г.



Чен К., Джиблин П., Ирвинг А. **MATLAB В МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЯХ:** Пер. с англ. — 352 с., ил.

Учебник по использованию популярного программно-математического пакета MATLAB, написанный известными английскими математиками (второй автор известен по книге Брус Дж., Джиблин П. Кривые и особенности. - М.: Мир, 1988). В первой части объясняется, как применять пакет для решения основных математических задач студентами 1-2 курсов вузов. Во второй части представлены задачи с элементами исследовательской работы, а в заключительной, третьей части делается упор на математическое моделирование. Изложение сопровождается примерами и упражнениями для самостоятельной работы. Книга написана ясно и просто. Авторы используют только возможности системы Windows, установленной практически на всех компьютерах России.

Основное содержание книги:

Часть I. Основы

- Введение
- Матрицы и комплексные числа
- Целые числа
- Графики и кривые
- Представление данных
- Вероятность и случайные числа
- Дифференциальные и разностные уравнения

Часть II. Исследования

- Магические квадраты
- НОД, псевдопростые числа
- Графики: кривые и огибающие
- Ломаные и кривые наискорейшего спуска
- Последовательности
- Итерации и фракталы
- Перестановки
- Матрицы и линейные системы
- Интерполяция
- Дифференциальные уравнения

Часть III. Моделирование

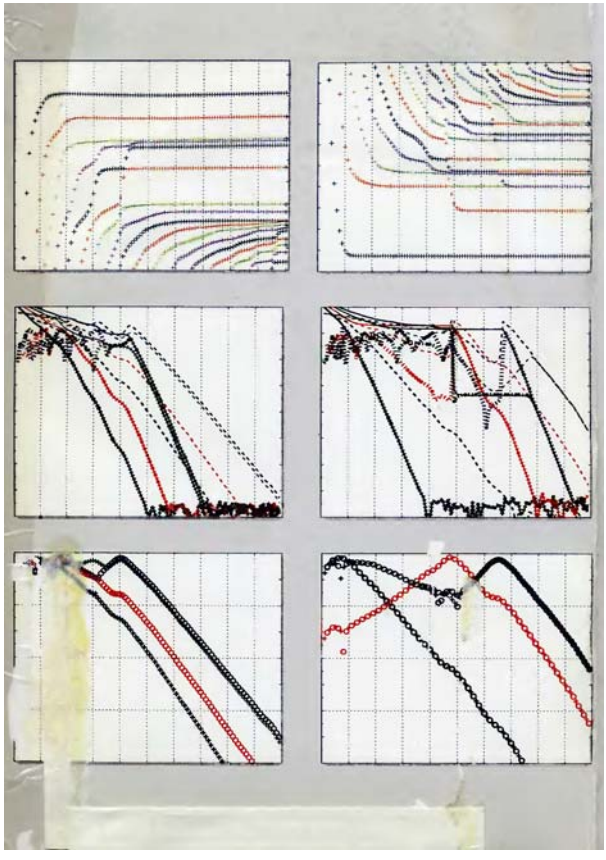
- Кассовые очереди
- Рыбное хозяйство
- Эпидемии
- Динамика сноуботинга
- Приливы

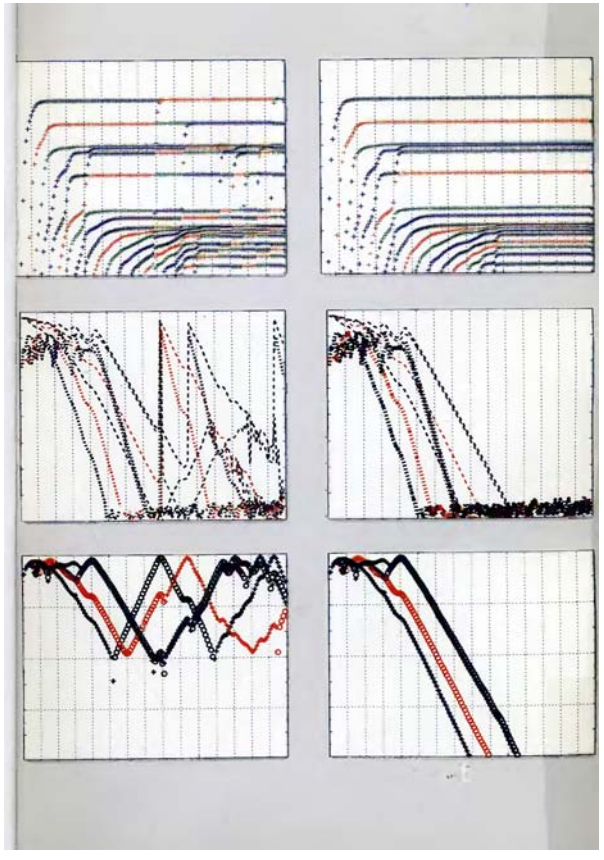
Приложения

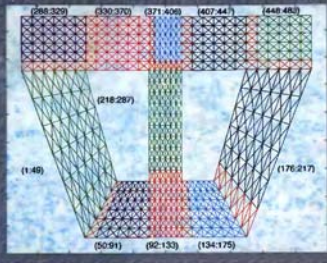
- Краткий перечень команд MATLAB'а
- Символьные вычисления в MATLAB'е

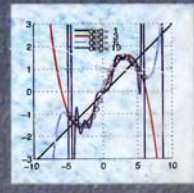
Для всех, кто изучает и применяет пакет MATLAB, для студентов, аспирантов и преподавателей вузов, а также для практиков-вычислителей.

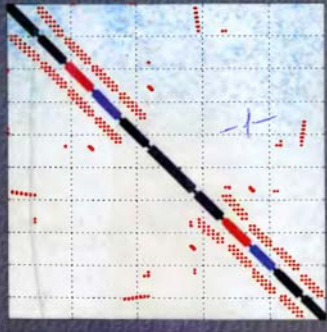
Адрес: 107996, ГСП-6, Москва, 1-й Рижский пер., 2
Контактные телефоны: (095)286-83-88, 286-82-33, 286-25-50
Факс: (095)286-84-55 ✧ **E-mail:** victor@mir.msk.su ✧ **http://www.mir-pubs.dol.ru**











ISBN 5-03-003402-1