От переводчиков

Прикладная линейная алгебра – это наука о вычислениях с матрицами, выполняемых на ЭВМ, так что ее история укладывается в последние 50 лет. Несмотря на такой небольшой период времени, здесь были достигнуты очень большие успехи, которым способствовали как обилие интересных и важных практических задач, так и быстро расширяющиеся технические возможности для их решения. Предмет этой книги является фундаментом всей вычислительной математики и в то же время он пока видится весьма сложным как в теоретическом, так и в прикладном плане. Теоретически он труден потому, что в нем теперь используются весьма тонкие математические методы, а практически – потому, что быстро усложняются алгоритмы, особенно в связи с необходимостью распараллеливать вычисления на все большее число процессоров.

Хорошие учебники всегда появляются позже тех основных результатов, которые в них излагаются, поскольку создание подобных учебников бывает не менее трудным, чем развитие самой науки. Тем важнее появление такой книги, если она относится к быстро развивающейся отрасли знания, нужной многим специалистам. Сегодня прикладная линейная алгебра – одна из таких отраслей. Большинство сколько-нибудь заметных зарубежных руководств по ней всегда переводилось на русский язык (см. список литературы). Особенность данной книги состоит в ее высоком педагогическом уровне, которого автор сумел достичь при изложении этого трудного раздела без снижения уровня научного. Не только оригинальная последовательность изложения основных идей, но и тщательно подобранная система упражнений помогают освоить как теоретический, так и практический аспекты предмета. Более подробно о книге можно узнать из предисловия автора, а также из оглавления. Надеемся, что она надолго привлечет внимание тех студентов, преподавателей и специалистов, которым приходится так или иначе сталкиваться с задачами прикладной линейной алгебры.

При переводе были исправлены замеченные нами (и неизбежные в таких изданиях) погрешности, а также те ошибки, которые указал сам автор на своем сайте уже после выхода второго издания.

Мы весьма признательны директору ИСС Минатома РФ академику В.Н. Михайлову за поддержку этой работы, а также И.А. Маховой (изд-во БИНОМ) и А.С. Попову (изд-во МИР), немало содействовавшим опубликованию данного перевода.

В.Е. Кондрашов
С.Б. Королев

Предисловие

Эта книга была написана для студентов старших курсов, аспирантов и опытных специалистов по математике, информатике, инженерным дисциплинам и вообще по всем тем отраслям знания, в которых используются различные численные методы. В основе большинства научных применений компьютеров лежат матричные вычисления, так что важно понимать, как правильно и эффективно выполнять такие вычисления. Книга идет навстречу такой потребности, давая детальное введение в основные идеи вычислительной линейной алгебры.

Необходимый предварительный объем знаний – это первоначальный курс по линейной алгебре и некоторый опыт компьютерного программирования. Для понимания студентом некоторых примеров, особенно во второй половине книги, полезно пройти начальный курс дифференциальных уравнений.

Есть несколько других превосходных книг по данному предмету, таких, как книги Деммеля [15], Голуба и Ван Лоуна [33] и Трефтена и Бау [71]. Студенты, не знакомые с этим предметом, часто находят их весьма трудными для чтения. Цель настоящего руководства состоит в том, чтобы дать простое, более постепенное введение в предмет, не теряя при этом математической строгости. Весьма положительная реакция студентов на первое издание убедила меня в успехе моей первой попытки и побудила к созданию обновленного и расширенного варианта.

Первое издание было предназначено главным образом для студентов старших курсов. Однако случилось так, что специалисты оценили книгу гораздо выше, чем просто учебник. Поэтому я добавил новый материал, чтобы сделать книгу более привлекательной и для студентов младших курсов. Эти дополнения детализированы ниже. Однако текст остается подходящим и для старшекурсников, поскольку элементарный материал остался в значительной степени нетронутым, а число простых упражнений было увеличено. Преподаватель может регулировать уровень трудности, решая, какие разделы и как глубоко должны быть проработаны. Многочисленные более трудные темы вынесены в упражнения в конце каждого раздела.

Книга содержит много упражнений от легких до умеренно трудных. Первые вкраплены в текст, вторые собраны в конце каждого раздела. Предполагается, что расположенные в тексте будут сразу прорабатываться читателем. Это

мой путь активного вовлечения студентов в процесс изучения. Чтобы что-то получить, вы должны что-то вложить. Многие из упражнений в конце разделов своими размерами могут поначалу насторожить читателя. Однако настойчивый студент найдет, что он может решить их с помощью обстоятельных подсказок и указаний. Я убедительно рекомендую каждому студенту проработать как можно больше упражнений.

Нумерация

Почти для всех нумерованных объектов в этой книге, включая теоремы, леммы, пронумерованные формулы, примеры и упражнения, использована единая схема нумерации. Так, например, первым пронумерованным объектом разд. 1.3 является теорема 1.3.1. Далее идут два уравнения (1.3.2) и (1.3.3) соответственно. За ними следует первое упражнение раздела с номером 1.3.4. Таким образом однозначно определяется номер каждого объекта: например, единственный объект в книге, который имеет номер 1.3.4 – это упражнение 1.3.4. Хотя эта схема необычна, я полагаю, что большинство читателей найдет ее совершенно естественной, как только они начнут пользоваться ею. Ее большое преимущество состоит в том, что она облегчает процесс поиска: читатель, нашедший упражнения 1.4.15 и 1.4.25, но ищущий пример 1.4.20, знает наверняка, что искомый пример находится где-то между этими двумя упражнениями. Есть два исключения из этой схемы. По техническим причинам, связанным с типографским набором текста, таблицы и рисунки (так называемые *плавающие объекты*) нумеруются отдельно по главам. Например, третий рисунок главы 1 обозначается как рис. 1.3.

Новое во втором издании использование MATLAB'A

К настоящему времени MATLAB[1] по общему признанию является наиболее широко используемым средством обучения матричным вычислениям. MATLAB – удобный язык очень высокого уровня, который позволяет студенту выполнять гораздо более сложные вычислительные эксперименты, чем прежде. MATLAB также широко применяется в промышленности. Поэтому я добавил много примеров и упражнений, которые используют MATLAB. Однако эта книга – не введение в MATLAB и не руководство по MATLAB'у. Для этих целей есть другие доступные издания, например, «Руководство по MATLAB'у» Хигэмов (D.I. Higham, N.I. Higham) [40]. Но справочная система MATLAB'а настолько хороша, что в действительности читателю не нужны никакие дополнительные пособия. Чтобы облегчить студенту использование MATLAB'а при

[1] MATLAB – зарегистрированная торговая марка MathWorks Inc. (http://www.mathworks.com).

чтении этой книги, я включил указатель терминов MATLAB'а отдельно от общего указателя.

Раньше я просил моих студентов писать и отлаживать их собственные программы на Фортране. Упражнения на Фортране из первого издания я оставил в значительной степени без изменений. Надеюсь, что часть студентов захочет проработать некоторые из этих заслуживающих внимания упражнений.

Другие приложения

Чтобы помочь студенту лучше понять важность предмета этой книги, я включил большое количество примеров и упражнений прикладного характера (решаемых с помощью MATLAB'а) главным образом в началах глав. Я выбрал очень простые приложения: электрические цепи, маятниковые системы, простые дифференциальные уравнения в частных производных. По моему мнению, именно из самых простых примеров мы можем узнать больше всего.

Более раннее введение сингулярного разложения (Singular Value Decomposition – SVD)

SVD – один из наиболее важных инструментов вычислительной линейной алгебры. В первом издании оно было помещено в заключительную главу книги, поскольку невозможно обсуждать методы вычисления SVD до обсуждения проблемы собственных значений. Но с тех пор я решил, что SVD должно быть введено раньше, чтобы и студент мог раньше узнать о его свойствах и использовании. С помощью MATLAB'а он может экспериментировать с SVD, не задумываясь о том, как оно вычисляется. Поэтому я добавил краткую главу об SVD в середине книги.

Новый материал по итерационным методам

Самое большое добавление к книге – глава по итерационным методам решения больших разреженных систем линейных уравнений. Главное здесь – это эффективный метод сопряженных градиентов для решения симметричных положительно определенных систем. Но там обсуждаются как классические итерации, так и предобуславливатели. Кратко рассмотрены методы крыловского подпространства для решения незнакоопределенных и несимметричных задач.

Есть также два новых раздела о методах решения задачи на собственные значения больших разреженных матриц. Обсуждение включает получившие

широкое признание неявно перезапускаемый метод Арнольди и метод Якоби–Дэвидсона.

Надеюсь, что именно эти дополнения сделают книгу более привлекательной для специалистов.

Другие новинки

Чтобы сделать книгу более полной, добавлено несколько других тем, а именно:

- обратный анализ ошибок гауссова исключения, а также обсуждение современного покомпонентного анализа ошибок;
- обсуждение перенормировки при ортогонализации – практического способа получения численно ортогональных векторов;
- обсуждение того, как обновить QR-разложение, когда строка или столбец добавляются или удаляются из матрицы данных, как это происходит при обработке сигналов и анализе данных;
- раздел, где представляются новые методы решения симметричной задачи на собственные значения, разработанные после выхода первого издания.

Было опущено несколько тем на том основании, что они стали устаревшими или слишком специализированными. Я использовал возможность исправить несколько досадных ошибок первого издания. Надеюсь, что при этом не допустил слишком много новых.

Благодарности

Я в великом долгу перед авторами некоторых ранних работ в данной области знания. В их числе А. С. Хаусхолдер [43], Дж. Уилкинсон [81], Дж. Э. Форсайт и К. Б. Моулер [24], Дж. У. Стьюарт [67], Ч. Л. Лоусон и Р. Дж. Хенсон [48], Б. Н. Парлетт [54], А. Джордж и Д. Лю [30], а также авторы Справочника [83], Руководства по EISPACK [64] и Руководства для пользователей по LINPACK [18]. Все они оказали сильное влияние на меня. Кстати, каждую из этих книг стоит прочитать и сегодня. Особая благодарность Кливу Моулеру за изобретение MATLAB'а, изменившего сам характер численных экспериментов.

Большая часть первого издания была написана, когда я находился в отпуске в Университете Билфилда, Германия. Мне доставляет удовольствие еще раз поблагодарить пригласившего меня туда давнего друга Людвига Элснера. В течение моего пребывания там я получал финансовую поддержку от комиссии Фулбрайта. Большой фрагмент второго издания также был написан в Германии, в Техническом Университете Хемнитца. Я благодарен за это пригласившему меня туда другому давнему другу Волкеру Мерманну. Во время этого визита я получал финансовую поддержку Отдела специальных исследований 393 ТУ Хемнитца. Я также в долгу перед моим родным институтом – Университетом штата Вашингтон – за его поддержку моей работы над обоими изданиями.

Еще раз благодарю профессоров Дейла Олески, Кэмбла Йетса и Тжаллинга Ипма, проверивших на семинарских занятиях предварительную версию первого издания. После первого издания множество читателей прислали мне поправки, отзывы и комментарии. В их числе А. Клайн, Л. Диеси, Е. Джессуп, Д. Койа, Д. Д. Олески, Б. Н. Парлетт, А. К. Рейнес, А. Витт и К. Райт. И, наконец, я благодарен многим студентам, помогавшим мне освоить методику изложения этого материала на протяжении многих лет.

Дэвид С. Уоткинс

Пулман, Вашингтон, январь 2002

1
Гауссово исключение и его варианты

Одна из наиболее часто встречающихся задач во всех областях научных исследований состоит в решении системы n линейных уравнений с n неизвестными. Например, в разд. 1.2 мы увидим, как посредством решения систем линейных уравнений рассчитываются напряжения и токи в электрических цепях, анализируются простые системы с упругими деформациями и численно решаются дифференциальные уравнения. Главная цель этой главы заключается в исследовании применения гауссова исключения при решении таких систем. Мы увидим, что есть много способов реализовать этот фундаментальный алгоритм.

1.1. Умножение матриц

Прежде чем начать изучение способов решения линейных систем, рассмотрим несколько простейших вычислений с матрицами. По ходу дела мы рассмотрим часть основных вопросов, которые будут возникать на протяжении всей книги. Среди них способы подсчета числа операций (числа флопов) при оценке сложности алгоритма, использование блочных матриц и операции над ними, а также демонстрация большого разнообразия способов организации простых матричных вычислений.

Умножение матрицы на вектор

Из множества матричных операций наиболее фундаментальной, которую нельзя назвать полностью тривиальной, является умножение матрицы на вектор. Рассмотрим $n \times m$-матрицу, т. е. прямоугольную таблицу с n строками и m столбцами

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{12} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}.$$

Элементы матрицы A могут быть вещественными или комплексными числами. Предположим пока, что они вещественные. Для заданного набора x из m вещественных чисел

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

можно ввести операцию умножения A на x, чтобы получить произведение $b = Ax$, где b — набор из n чисел. Его i-я компонента дается выражением

$$b_i = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{im}x_m = \sum_{j=1}^{m} a_{ij}x_j \ . \qquad (1.1.1)$$

Другими словами, i-я компонента ветктора b получается как внутреннее (скалярное) произведение i-й строки матрицы A на x.

Пример 1.1.2. Умножим матрицу на вектор при $n = 2$ и $m = 3$:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 50 \\ 122 \end{bmatrix},$$

поскольку $1 \times 7 + 2 \times 8 + 3 \times 9 = 50$ и $4 \times 7 + 5 \times 8 + 6 \times 9 = 122$. □

Программный код, выполняющий матрично-векторное умножение, мог бы выглядеть подобно следующему:

$$\begin{array}{l} b \leftarrow 0 \\ \text{for } i = 1, \ldots, n \\ \left[\begin{array}{l} \text{for } j = 1, \ldots, m \\ \quad [b_i \leftarrow b_i + a_{ij} x_j \end{array} \right. \end{array} \qquad (1.1.3)$$

Здесь цикл по j накапливает внутреннее произведение b_i.

Возможна и другая интерпретация матрично-векторного умножения, которая оказывается весьма полезной. Взгляните еще раз на (1.1.1), но теперь рассматривайте это выражение как формулу для всего вектора b, а не его отдельных компонент. Другими словами, возьмите выражение (1.1.1), которое в действительности есть n равенств для b_1, b_2, ..., b_n, и объедеините их в одно векторное равенство

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{bmatrix} x_2 + \ldots + \begin{bmatrix} a_{1m} \\ a_{2m} \\ \vdots \\ a_{nm} \end{bmatrix} x_m. \qquad (1.1.4)$$

Это показывает, что b есть линейная комбинация столбцов матрицы A.

Пример 1.1.5. Для условий примера 1.1.2 будем иметь

$$\begin{bmatrix} 50 \\ 122 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} 7 + \begin{bmatrix} 2 \\ 5 \end{bmatrix} 8 + \begin{bmatrix} 3 \\ 6 \end{bmatrix} 9.$$

□

Утверждение 1.1.6. *Если $b = Ax$, то b является линейной комбинацией столбцов матрицы A.*

Если через A_j обозначить j-й столбец матрицы A, то

$$b = \sum_{j=1}^{m} A_j x_j .$$

Записывая это в виде машинного псевдокода, получим

$$\begin{aligned} &b \leftarrow 0 \\ &\text{for } j = 1, \ldots, m \\ &\quad [b \leftarrow b + A_j x_j \end{aligned}$$

Если каждую векторную операцию представить в виде цикла, то псевдокод примет вид

$$\begin{aligned} &b \leftarrow 0 \\ &\text{for } j = 1, \ldots, m \\ &\left[\begin{array}{l} \text{for } i = 1, \ldots, n \\ \quad [b_i \leftarrow b_i + a_{ij} x_j \end{array} \right. \end{aligned} \qquad (1.1.7)$$

Заметьте, что (1.1.7) идентично (1.1.3), за исключением перестановки циклов. Эти алгоритмы выполняют точно одни и те же действия, но в разной последовательности. Назовем (1.1.3) матрично-векторным умножением *по строкам*, потому что при этом A выбирается по строкам. В противоположность этому (1.1.7) есть матрично-векторное умножение *по столбцам*.

Подсчет числа флопов

Вещественные числа в компьютере обычно записываются в формате с плавающей точкой. Арифметические операции, выполняемые над этими числами компьютером, называются операциями с плавающей точкой или коротко *флопами* (flops – floating-point operations). Присвоение $b_i \leftarrow b_i + a_{ij}x_j$ требует выполнения двух флопов – одного умножения и одного сложения с плавающей точкой.[1]

Запуская программу, мы каждый раз задаемся вопросом: сколько времени потребуется для ее выполнения? Если приходится работать с большими мат-

[1] Арифметика с плавающей точкой будет обсуждаться в разд. 2.5.

рицами, скажем, размеров 1000×1000, то времени может потребоваться много. Традиционный способ оценки времени работы программы состоит в подсчете числа флопов, которые должен выполнить при этом компьютер. Подсчитаем число флопов, выполняемых при матрично-векторном умножении. Из (1.1.7) мы видим, что если A есть $n \times m$-матрица, то внешний цикл будет выполняться m раз. При каждом его проходе внутренний цикл выполняется n раз. Каждое выполнение внутреннего цикла требует двух флопов. Поэтому полное число флопов, выполняемых алгоритмом, равно $2nm$. Здесь очень легко подсчитать число флопов. Для более сложных алгоритмов может оказаться полезным следующий прием. Заменим каждый цикл символом суммирования Σ. Так как внутренний цикл выполняется для $i = 1, ...,n$ и за каждый его проход выполняется два флопа, то общее количество флопов, выполненных при каждом проходе внутреннего цикла, равно $\sum_{i=1}^{n} 2$. Так как внешний цикл выполняется для $j = 1, ..., m$, то общее число флопов равно

$$\sum_{j=1}^{m} \sum_{i=1}^{n} 2 = \sum_{j=1}^{m} 2n = 2nm.$$

Подсчет флопов дает примерное представление о том, как долго будет выполняться алгоритм. Предположим, что мы выполнили алгоритм, например, для матрицы размеров 300×400 и отметили, сколько потребовалось времени. Если мы теперь захотим выполнить его для матрицы размеров 600×400, то должны ожидать, что на это потребуется вдвое больше времени, так как мы удвоили n, не меняя при этом m, и таким образом удвоили число флопов, ставшее равным $2nm$.

Квадратные матрицы часто появляются в приложениях. Если A имеет размеры $n \times n$, то число флопов при матрично-векторном умножении равно $2n^2$. Если мы выполним матрично-векторное умножение, скажем, на 500×500-матрицу, то можем ожидать, что то же самое действие для матрицы 1000×1000 будет выполняться приблизительно в четыре раза дольше, поскольку в этом случае удвоение n учетверит число флопов.

Умножение $n \times n$-матрицы на вектор – пример того, что называют $O(n^2)$-процессом или процессом *порядка* n^2. Это означает только то, что необходимый объем работы пропорционален n^2. Такое обозначение используется, чтобы подчеркнуть зависимость от n и ослабить роль постоянного множителя в пропорциональной зависимости, который в этом случае равен 2. Любой $O(n^2)$-процесс обладает тем свойством, что при удвоении размерности задачи объем работы учетверяется.

Важно понимать, что число флопов дает лишь грубую оценку объема работы при выполнении алгоритма. При этом не учитывается много других действий, выполняемых компьютером в процессе работы алгоритма. Наиболее важные из них – выборка из памяти данных, необходимых для выполнения операций, и запись результатов в память после выполнения этих операций. На многих компьютерах операции доступа к памяти производятся медленнее

операций с плавающей точкой, так что это делает более осмысленным подсчет числа обращений к памяти, а не числа флопов. Подсчет флопов полезен, однако, потому что дает нам также грубую оценку использования памяти. Для каждой операции мы должны извлечь операнды из памяти, а после каждой операции записать результат в память. Это очень грубое упрощение того, что в действительности происходит в современных компьютерах. На скорость выполнения алгоритма может сильно влиять то, как организовано использование памяти. Мы еще поговорим об этом в конце раздела. Тем не менее число флопов дает первую полезную оценку времени выполнения алгоритма, и мы обязательно будем его подсчитывать.

Упражнение 1.1.8. Начнем знакомиться с MATLAB'ом. Войдите в систему машины, на которой установлен MATLAB, и запустите его. В командной строке MATLAB'а наберите `A = randn(3,4)`[1], чтобы сгенерировать матрицу размеров 3×4 со случайными элементами, имеющими стандартное нормальное распределение. Чтобы больше узнать о команде `randn`, выполните `help randn`. Теперь выполните `x = randn(4,1)`, чтобы получить вектор (матрицу размеров 4×1) случайных чисел. Для умножения A на x и записи результата в новый вектор b выполните `b = A*x`.

Чтобы MATLAB сохранял стенограмму ваших действий, выполните команду `diary on`. Это позволит сохранить файл, называемый `diary`, в котором содержится запись о вашем MATLAB-сеансе. Позже вы сможете подредактировать этот файл, распечатать его, показать его вашему преподавателю и т.п. Чтобы больше узнать о команде `diary`, выполните `help diary`.

Другими полезными командами являются `help` и `help help`. Чтобы увидеть демонстрацию возможностей MATLAB'а, выполните `demo`. □

Упражнение 1.1.9. Рассмотрим следующую простую MATLAB-программу:

```
n = 200;
for jay = 1:4
   if jay > 1
      oldtime = time;
   end
   A = randn(n);
   x = randn(n,1);
   t = cputime;
   b = A*x;
   matrixsize = n
   time = cputime - t
   if jay > 1
      ratio = time/oldtime
   end
   n = 2*n;
end
```

[1] И нажмите клавишу Enter, что приведет к выполнению набранных в строке инструкций.—*Прим. пер.*

[. . .]