Глава 1

Предварительные сведения

1.1. Машинная арифметика

1.1.1. Числа с плавающей точкой

Для того чтобы использовать вещественные числа в машинных вычислениях, необходимо решить следующую общую проблему: каким образом сохранить произвольное $x \in \mathbb{R}$ в ограниченном количестве ячеек памяти? Существует несколько способов решения этой проблемы, и наиболее распространенным является представление x в виде числа с плавающей точкой.

Определение 1.1. Пусть $\beta \in \mathbb{N}$ — основание системы счисления, $p \in \mathbb{N}$ — число значащих разрядов, d_i — цифры. Вещественное число вида

$$\pm \underbrace{d_0, d_1 d_2 \ldots d_{p-1}}_{m} \cdot \beta^e, \quad 0 \leqslant d_i < \beta, \tag{1.1}$$

называется *числом с плавающей точкой* (ЧПТ). Число $m \in \mathbb{R}$ называют *мантиссой* или *значащей частью*. Число $e \in \mathbb{Z}$ называют *показателем* или *экспонентой* (не путать с числом e).

Представление (1.1) для любого x очевидно не является единственным — оно зависит от «положения точки»:

$$0{,}0001234 = 0{,}0012340 \cdot 10^{-1} = 1{,}2340000 \cdot 10^{-4}.$$

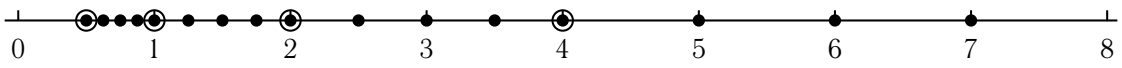
Поэтому по умолчанию используется так называемая нормализованная форма записи ЧПТ, в которой точка ставится после первой значащей цифры.

Определение 1.2. Число с плавающей точкой с ненулевым первым разрядом ($d_0 \neq 0$) называется *нормализованным*. Множество всех нормализованных ЧПТ с основанием β, p-разрядной мантиссой и $e_{\min} \leqslant e \leqslant e_{\max}$ условимся обозначать $\mathbb{F}_1(\beta, p, e_{\min}, e_{\max})$ или просто $\mathbb{F}_1$.

1.1.2. Двоичные числа с плавающей точкой

Поскольку в большинстве современных компьютеров используется двоичная арифметика, рассмотрим подробно случай $\beta = 2$. Возьмем $p = 3$, $e_{\min} = -1$, $e_{\max} = 2$. Все соответствующие положительные нормализованные ЧПТ приведены в таблице, они же изображены на рисунке. Степени двойки, соответствующие единичному значению мантиссы, обведены окружностями.

$m \diagdown e$	-1		0		1		10 (2)	
1,00	0,1	(0,5)	1	(1)	10	(2)	100	(4)
1,01	0,101	(0,625)	1,01	(1,25)	10,1	(2,5)	101	(5)
1,10	0,110	(0,75)	1,1	(1,5)	11	(3)	110	(6)
1,11	0,111	(0,875)	1,11	(1,75)	11,1	(3,5)	111	(7)



Приведенные данные наглядно демонстрируют следующие важные свойства, общие для всех нормализованных ЧПТ:
1) отсутствие нуля: $0 \notin \mathbb{F}_1$;
2) ЧПТ распределены на числовой прямой *неравномерно*;
3) чем больше модуль $\xi \in \mathbb{F}_1$, тем больше и расстояние между ξ и соседними элементами $\mathbb{F}_1$;
4) между нулем и минимальным положительным $\xi_{\min} \in \mathbb{F}_1$ существует «зазор», ширина которого больше расстояния от $\xi_{\min}$ до следующего ЧПТ в 2^{p-1} раз.

Двоичные ЧПТ выгодно отличаются от остальных тем, что в их нормализованной записи первый разряд d_0 всегда равен 1, поэтому его в памяти можно не хранить. *Таким образом, для хранения p-разрядной двоичной мантиссы достаточно $(p-1)$ битов.*

1.1.3. Способы округления

Если представление x в системе счисления по основанию β содержит больше p значащих цифр, мы не можем точно записать его в виде (1.1). В этом случае можно лишь приблизить x каким-то числом с плавающей точкой, которое в дальнейшем будем обозначать $\mathrm{fl}(x)$.

Определение 1.3. *Правилом округления* для данного множества чисел с плавающей точкой $\mathbb{F} \subset \mathbb{R}$ будем называть отображение

$$\mathrm{fl} : \mathbb{R} \to \mathbb{F}$$

такое, что $\mathrm{fl}(x) = x$, если $x \in \mathbb{F}$, и $\mathrm{fl}(x) \approx x$ в противном случае.

Рассмотрим несколько способов задания fl, считая $\beta = 10$, $p = 3$.
1) Отбрасывание «лишних» знаков ($\mathrm{fl} = \mathrm{fl}_d$): $\mathrm{fl}_d(0{,}12345) = 1{,}23 \cdot 10^{-1}$.
2) Округление вверх («школьное округление», $\mathrm{fl} = \mathrm{fl}_u$). $\mathrm{fl}_u(543{,}21) = 5{,}43 \cdot 10^2$, $\mathrm{fl}_u(5678) = 5{,}68 \cdot 10^3$. В случае, когда запись x заканчивается на 5, его округляют до большего ЧПТ (вверх) : $\mathrm{fl}_u(23{,}45) = 2{,}35 \cdot 10^1$.

3) Округление до четного ($\mathrm{fl} = \mathrm{fl}_e$). Этот способ отличается от предыдущего только трактовкой «спорного» случая, когда x находится ровно между двумя ЧПТ $\underline{x}$ и $\overline{x}$. Оба эти приближения на самом деле равноправны, поэтому вместо того, чтобы всегда выбирать $\overline{x}$, с вероятностью 50 % выбирается $\mathrm{fl}_e(x) = \underline{x}$ либо $\mathrm{fl}_e(x) = \overline{x}$. Реализовать это можно, всегда выбирая из $\underline{x}$ и $\overline{x}$ то значение, мантисса которого заканчивается на четную цифру. Таким образом, получим: $\mathrm{fl}_e(23{,}45) = 2{,}34 \cdot 10^1$, но $\mathrm{fl}_e(23{,}55) = 2{,}36 \cdot 10^1$.

Возникает вопрос: какой из описанных способов округления лучше? Ответ на него дает следующая теорема.

Теорема 1.1. *Пусть x и y — два числа с плавающей точкой. Рассмотрим последовательность $\{x_i\}$, определенную по правилу*

$$x_0 = x, \quad x_{i+1} = \mathrm{fl}(\mathrm{fl}(x_i + y) - y).$$

Если $\mathrm{fl} = \mathrm{fl}_e$, то либо $x_i = x \; \forall i \geqslant 0$, либо $x_i = x_1 \; \forall i \geqslant 1$.

Заметим, что по стандарту IEEE 754 в современных ЭВМ используется $\mathrm{fl} = \mathrm{fl}_e$.

1.1.4. Расширение множества чисел с плавающей точкой

Для практического использования машинной арифметики нам недостаточно множества нормализованных ЧПТ $\mathbb{F}_1$. Как минимум к этому множеству нужно добавить нуль (свойство 1). Кроме этого, современные машинные арифметики включают специальные значения для обозначения бесконечностей, результатов некорректных операций и т. п.

Денормализованные числа

Наличие «зазора» между нулем и минимальным положительным нормализованным ЧПТ (свойство 4) может привести к серьезным проблемам на практике. Возьмем, например, ЧПТ $x = 0{,}75$ и $y = 0{,}625$ из рассмотренного выше модельного множества $\mathbb{F}_1(2, 3, -1, 2)$. Поскольку $x - y = 0{,}125$, при любом разумном способе округления мы имеем $\mathrm{fl}(x - y) = 0$, т. е., например, выполнение обычного кода

```
if (x != y) then z = 1/(x - y)
```

в нашем случае приведет к плачевному результату.

Эта проблема в современных машинных арифметиках решается дополнением множества нормализованных ЧПТ так называемыми денормализованными числами.

Определение 1.4. Вещественные числа вида

$$0, d_1 d_2 \ldots d_{p-1} \cdot \beta^{e_{\min}},$$

где d_i — произвольные цифры (по основанию β), называются *денормализованными* числами с плавающей точкой (ДЧПТ). Множество всех ДЧПТ с параметрами $\beta, p, e_{\min}$ будем обозначать $\mathbb{F}_0(\beta, p, e_{\min})$ либо кратко $\mathbb{F}_0$.

Введение денормализации сразу решает две проблемы: теперь выполняется свойство

$$x = y \;\Leftrightarrow\; \mathrm{fl}(x - y) = 0 \quad \text{для любых ЧПТ } x, y,$$

а также добавляется нуль ко множеству машинных чисел.

Заметим, что для хранения денормализованных ЧПТ необходимо одно дополнительное значение для экспоненты (как правило, это $e_{\min} - 1$).

Специальные величины

Стандарт IEEE 754, которому соответствуют практически все современные ЭВМ, предусматривает наличие специальных значений для машинных чисел, которым соответствуют не ЧПТ, а другие объекты. Простейшие объекты такого типа — это $+\infty$ и $-\infty$ (присутствуют также $+0$ и -0). Результаты вычислений с бесконечностями являются вполне определенными: например, если x — положительное число, то по стандарту $x/\pm\infty = \pm 0$, $x/\pm 0 = \pm\infty$ и т. д. Кроме этого, стандартом определяются так называемые «нечисла» (NaN, от «not a number»), которые обозначают результаты некорректных арифметических операций, таких как, например, извлечение корня из отрицательного числа.

В дальнейшем под *машинными числами* будем понимать элементы множества

$$\mathbb{F}_0 \cup \mathbb{F}_1,$$

а под термином *арифметика с плавающей точкой* (АПТ) — множество машинных чисел в совокупности с правилом округления fl.

При вычислениях в АПТ будем считать, что результаты операций сложения, вычитания, умножения и деления являются *точно округляемыми*. Это означает, что результат указанных операций всегда вычисляется точно, после чего округляется до ЧПТ по правилу fl .

1.1.5. Машинный эпсилон

Параметры $\beta, p, e_{\min}, e_{\max}$ и fl полностью определяют свойства АПТ, однако их знание не дает прямой информации о том, насколько хороша или плоха соответствующая арифметика. С практической точки зрения пользователю нужны критерии, по которым можно определить качество арифметики. Основным показателем качества будем считать точность, с которой арифметика приближает вещественные числа.

Определение 1.5. *Абсолютной погрешностью округления* для числа $x \in \mathbb{R}$ в данной арифметике с плавающей точкой называется число

$$\Delta(x) = |x - \mathrm{fl}(x)|, \tag{1.2}$$

а *относительной погрешностью округления* — число

$$\delta(x) = \frac{|x - \mathrm{fl}(x)|}{|x|} = \frac{\Delta(x)}{|x|}. \tag{1.3}$$

Иногда относительную погрешность измеряют в процентах, умножая ее на 100. Важно понимать, что при работе с машинной арифметикой уместнее всего оперировать относительными погрешностями, так как чем больше модуль x, тем большее значение может иметь $\Delta(x)$ (свойство 3), в то время как максимальная относительная погрешность $\delta(x)$ не зависит от величины $|x|$.

Определение 1.6. *Машинным эпсилон* (ε_M) для арифметики с плавающей точкой называется наименьшее положительное число ε, удовлетворяющее условию

$$\mathrm{fl}(1 + \varepsilon) > 1.$$

Заметим, что значение машинного эпсилон зависит от правила округления и от количества бит в мантиссе.

$\triangleright_1$ Выразите ε_M в зависимости от указанных параметров.

Определение, как это часто бывает, не указывает явно на истинный смысл величины ε_M, который выражается в следующей теореме.

Теорема 1.2. *Для всех вещественных x таких, что $\xi_{\min} \leqslant |x| \leqslant \xi_{\max}$, где $\xi_{\min}$ и $\xi_{\max}$ — минимальное и максимальное положительное нормализованное ЧПТ соответственно, справедлива оценка*

$$\delta(x) \leqslant \varepsilon_M.$$

Другими словами, величина машинного эпсилон ограничивает относительную погрешность округления для всех чисел в диапазоне нормализованных ЧПТ. Чем меньше величина ε_M, тем точнее вещественные числа приближаются в машинной арифметике.

$\triangleright_2$ Докажите теорему 1.2. Для этого можно использовать следующий способ: оцените максимальную относительную погрешность $\Delta(x)$ для $x \in (1, 2)$, затем для $x \in (2, 4)$ и обобщите для $x \in (2^k, 2^{k+1})$, $k \in \mathbb{Z}$. После этого останется применить определение относительной погрешности $\delta(x)$.

$\triangleright_3$ Покажите, что утверждение теоремы 1.2 не выполняется в диапазоне денормализованных ЧПТ.

1.1.6. Стандарт IEEE 754

Международный стандарт *IEEE 754 floating point standard* определяет правила организации машинной арифметики с плавающей точкой. В настоящее время ему соответствует большинство вычислительных машин. В частности, наиболее распространенный тип данных, известный как double precision floating point (тип double в C/C++), по стандарту имеет следующие параметры:

β	p	$e_{\min}$	$e_{\max}$	fl
2	53	-1022	1023	fl_e

1.1.7. Проблемы машинных вычислений

Потеря значимости. Эта проблема возникает при вычитании двух близких чисел, которые не являются точно представимыми в виде ЧПТ.

Покажем это на примере модельной арифметики с $\mathbb{F}_1(2, 3, -1, 2)$ и $\mathrm{fl} = \mathrm{fl}_u$: пусть $x = 4{,}51$, $y = 4{,}49$. Имеем $\mathrm{fl}(x) = 5$, $\mathrm{fl}(y) = 4$, и $\mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y)) = 1$, тогда как $x - y = 0{,}02$. Таким образом мы получили относительную погрешность вычисления, равную 5000 %, несмотря на то, что относительная погрешность округления для x и y составляет менее 12,5 %. Отметим, что сложение этих двух чисел выполняется в данной арифметике точно.

Неассоциативность арифметических операций. Работая с машинными числами, всегда следует помнить о том, что порядок операций существенно влияет на результат. Простейший случай — нарушение привычного свойства ассоциативности: если a, b и c — машинные числа, а $\circ$ — бинарная операция, то в общем случае $\mathrm{fl}(\mathrm{fl}(a \circ b) \circ c) \neq \mathrm{fl}(a \circ \mathrm{fl}(b \circ c))$.

Итак, при использовании машинной арифметики всегда следует помнить о том, что как только в память ЭВМ записывается число x, оно автоматически превращается в число $\tilde{x} = \mathrm{fl}(x)$, которое *почти всегда* не будет равно x. Кроме того, чем больше модуль x, тем больше может быть разница между x и $\tilde{x}$ (абсолютная погрешность $\Delta(x)$). Относительная же погрешность округления согласно теореме 1.2 почти всегда ограничена величиной ε_M.

1.2. Обусловленность задачи

1.2.1. Некорректные и плохо обусловленные задачи

Постоянное присутствие ошибок округления при работе с машинной арифметикой предъявляет особые требования к вычислительным алгоритмам и требует дополнительного анализа решаемой задачи. Так как практически все числа представляются в ЭВМ с погрешностью, необходимо знать, насколько решение чувствительно к изменениям параметров задачи.

Определение 1.7. Задача называется *корректно поставленной* или просто *корректной*, если ее решение существует, единственно и непрерывно зависит от начальных данных. Если нарушено хотя бы одно из этих условий, задачу называют *некорректной*.

Решение некорректных задач на ЭВМ — весьма серьезная проблема. Если, например, нарушено условие непрерывной зависимости от параметров, то наличие малейшей погрешности в начальных данных (которая практически неминуемо произойдет, как только вы запишете эти данные в память), может кардинальным образом исказить решение.

Существует еще один класс задач, формально являющихся корректными, но решения которых, тем не менее, тоже весьма плохо ведут себя при наличии погрешностей в начальных данных — это так называемые плохо обусловленные задачи. В общих чертах плохо обусловленной называется задача, которая при маленькой *относительной* погрешности в начальных данных дает большую *относительную* погрешность в решении. Упор на относительную погрешность делается потому, что, как мы знаем из п. 1.1, при округлении вещественного числа x до машинного $\mathrm{fl}(x)$ абсолютная погрешность $\Delta(x)$ зависит от величины $|x|$, в то время как относительная погрешность $\delta(x)$ постоянна для данной машинной арифметики.

Пример 1.1 (вычисление значения многочлена). Исследовать обусловленность задачи вычисления значения многочлена относительно погрешности, вносимой в один из его коэффициентов.

Решение. Пусть $P(a_0, \ldots, a_n, x) = \sum_{i=0}^{n} a_i x^i$. Рассмотрим задачу вычисления значения данного многочлена, считая фиксированными x и все коэффициенты a_i кроме какого-то одного — a_k, который и будем считать параметром. Пусть вместо точного значения a_k используется «возмущенное» значение $\tilde{a}_k$, т. е. имеем относительную погрешность в

начальных данных, равную

$$\frac{|a_k - \tilde{a}_k|}{|a_k|}.$$

Вычислим относительную погрешность решения и выразим ее через относительную погрешность начальных данных:

$$\frac{|P(a_0, \ldots, a_n, x) - P(a_0, \ldots, \tilde{a}_k, \ldots, a_n, x)|}{|P(a_0, \ldots, a_n, x)|} =$$
$$= \frac{|(a_k - \tilde{a}_k)x^k|}{|\sum_{i=0}^{n} a_i x^i|} = \frac{|a_k\, x^k|}{|\sum_{i=0}^{n} a_i x^i|} \frac{|a_k - \tilde{a}_k|}{|a_k|}.$$

Коэффициент

$$æ = \frac{|a_k\, x^k|}{|\sum_{i=0}^{n} a_i x^i|}$$

называется *числом обусловленности*. Проблемы при решении рассматриваемой задачи могут возникать, когда это число велико: например, когда x близко к одному из корней многочлена P, или $x \gg 1$ и k достаточно велико.

Численный пример:

$$p(x) = (x-2)^{10} = x^{10} - 20x^9 + 180x^8 - 960x^7 + \ldots - 5120x + 1024.$$

Пусть $x = 3$. Тогда $P(a_0, \ldots, a_{10}, x) = 1$. Изменим коэффициент $a_9 = -20$ на 0,01 (что составляет 0,05 %): $\tilde{a}_9 = -19{,}99$. Тогда получим

$$P(a_0, \ldots, \tilde{a}_9, a_{10}, x) = 197{,}83,$$

что на 19683 % больше точного значения. □

Таким образом, *число обусловленности задачи показывает, во сколько раз относительная погрешность возмущенного решения может превосходить относительную погрешность соответствующих начальных данных.*

Задача называется *плохо обусловленной*, если ее число обусловленности велико.

Замечание 1.1. Естественно, понятие «большое число обусловленности» относительно. Судить о величине обусловленности можно лишь в контексте той машинной арифметики, которая используется для вычислений, а еще точнее — от величины машинного эпсилон, так как эта величина ограничивает относительную погрешность округления.

$\triangleright_1$ Пусть известно значение ε_M. Укажите при каких значениях числа обусловленности задачу можно считать плохо обусловленной в такой арифметике.

1.2.2. Векторные нормы

В дальнейшем как параметры, так и решения рассматриваемых задач будут векторами пространства $\mathbb{R}^n$. Для исследования обусловленности задач нужно измерять «величины» этих векторов, для чего используются *векторные нормы*. Мы будем активно пользоваться двумя векторными нормами: *максимум-нормой*

$$\|x\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i| \tag{1.4}$$

и *евклидовой нормой*

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{1.5}$$

Обе эти нормы являются частными случаями *p-нормы*, определяемой формулой

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

1.2.3. Матричные нормы

Рассмотрим задачу решения системы линейных алгебраических уравнений (СЛАУ) вида

$$Ax = b, \tag{1.6}$$

где A — невырожденная квадратная матрица размерности n; $x, b \in \mathbb{R}^n$. Прежде чем приступить к алгоритмам численного решения этой задачи, исследуем ее обусловленность.

Как и в случае векторных параметров, нам нужно будет как-то измерять «величину» матрицы A. Делать это мы будем с использованием операторных матричных норм. При работе с матрицами (по крайней мере в контексте линейной алгебры) всегда важно помнить, что любая матрица определяет *линейный оператор*, т. е. отображение $A : \mathbb{R}^m \to \mathbb{R}^n$, которое обладает свойством линейности

$$A(\alpha x + \beta y) = \alpha Ax + \beta Ay, \quad \forall\, x, y \in \mathbb{R}^n,\ \alpha, \beta \in \mathbb{R}.$$

Важность такой точки зрения следует хотя бы из того, что так называемое правило умножения матриц, которое обычно вводится без обоснования,

есть не что иное, как алгоритм вычисления композиции линейных операторов. В таком контексте вопрос об определении «величины» матрицы сводится к определению нормы соответствующего линейного оператора.

Определение 1.8. Нормой линейного оператора (матрицы) A называют число

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|.$$

Норма оператора полностью определяется векторной нормой, т. е. каждая векторная норма порождает (*индуцирует*) соответствующую ей операторную матричную форму (в этом случае говорят также, что матричная норма *подчинена* векторной).

В дальнейшем мы без оговорок будем предполагать, что используемая матричная норма подчинена векторной.

Норма оператора равна максимальному «коэффициенту растяжения»: она показывает, во сколько раз под его действием может увеличиться норма вектора. Поэтому *по определению* для любой операторной нормы имеем важное свойство

$$\|Ax\| \leqslant \|A\| \, \|x\|. \tag{1.7}$$

Напомним, как вычисляются матричные нормы, индуцированные векторными нормами $\|\cdot\|_\infty$ и $\|\cdot\|_2$.

- Векторной максимум-нормой $\|\cdot\|_\infty$ индуцируется матричная норма, вычисляемая по правилу

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} |a_{ij}|. \tag{1.8}$$

Данную норму будем называть (строчной) *максимум-нормой*, или *кубической* матричной нормой.

- Евклидовой векторной нормой $\|\cdot\|_2$ индуцируется матричная норма, вычисляемая по правилу

$$\|A\|_2 = \max_{1 \leqslant i \leqslant n} \sqrt{\lambda_i}, \tag{1.9}$$

где $\{\lambda_i\}_{i=1}^n$ — собственные значения матрицы A^*A. Эту норму называют *спектральной* матричной нормой.

1.2.4. Число обусловленности матрицы

Рассмотрим СЛАУ (1.6). Решение этой системы эквивалентно вычислению

$$x = A^{-1}b.$$

Исследуем обусловленность этой задачи, считая параметром вектор правой части b. Действуем по схеме, аналогичной примеру 1.2.1. Относительная погрешность начальных данных имеет вид

$$\delta_b = \frac{\|b - \tilde{b}\|}{\|b\|},$$

а возмущенного решения —

$$\delta_x = \frac{\|x - \tilde{x}\|}{\|x\|},$$

где $x = A^{-1}b$, $\tilde{x} = A^{-1}\tilde{b}$. Наша задача — установить связь между этими двумя погрешностями. Имеем

$$\delta_x = \frac{\|A^{-1}(b - \tilde{b})\|}{\|A^{-1}b\|} \leqslant \frac{\|A^{-1}\|\,\|b - \tilde{b}\|}{\|A^{-1}b\|} \leqslant \|A^{-1}\|\,\|A\|\frac{\|b - \tilde{b}\|}{\|b\|}.$$

Для получения последней оценки мы использовали тот факт, что

$$\|b\| = \|AA^{-1}b\| \leqslant \|A\|\,\|A^{-1}b\| \Rightarrow \|A^{-1}b\| \geqslant \|A\|^{-1}\|b\|.$$

В итоге мы получим

$$\delta_x \leqslant æ(A)\delta_b, \tag{1.10}$$

где $æ(A) = \|A^{-1}\|\,\|A\|$.

Исследование обусловленности относительно погрешностей в матрице A требует более тонкого подхода, но приводит к аналогичному результату. Таким образом, из (1.10) вытекает следующее определение.

Определение 1.9. *Числом обусловленности* невырожденной матрицы A называется число

$$æ(A) = \|A\|\,\|A^{-1}\|. \tag{1.11}$$

Если матрица A вырождена, ее число обусловленности полагается равным бесконечности.

Замечание 1.2. Несмотря на то, что это определение ассоциировано с матрицей A, необходимо четко понимать, что речь идет об обусловленности задачи решения СЛАУ.

Замечание 1.3. Число обусловленности по определению зависит от нормы. В случаях, когда это необходимо, мы будем употреблять говорящие обозначения $æ_2(A)$ и $æ_\infty(A)$.

Число обусловленности матрицы обладает следующими свойствами:
1) $æ(A) \geqslant 1 : \ 1 = \|A^{-1}A\| \leqslant \|A\| \, \|A^{-1}\|$;
2) $æ(AB) \leqslant æ(A)æ(B) : \|AB\| \, \|(AB)^{-1}\| \leqslant \|A\| \, \|A^{-1}\| \, \|B\| \, \|B^{-1}\|$;
3) если $A = A^*$, то $æ_2(A) = \dfrac{|\lambda_{\max}|}{|\lambda_{\min}|}$, где $\lambda_{\min}$ и $\lambda_{\max}$ — минимальное и максимальное по модулю собственные значения матрицы A соответственно.

Замечание 1.4. Отметим, что в общем случае отсутствует прямая связь между величинами собственных значений и числом обусловленности. Например, собственные значения матрицы $A = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix}$ равны 1, в то время как $æ_2(A) \to \infty$ при $|\alpha| \to \infty$.

Замечание 1.5. Укажем на одно часто встречающееся заблуждение. Так как $æ(A)$ является своеобразным индикатором близости матрицы A к вырожденной, может возникнуть впечатление, что чем меньше определитель, тем больше число обусловленности. На самом же деле такой связи нет: достаточно заметить, что у A и A^{-1} взаимно обратные определители, но одинаковые числа обусловленности.

В заключение данного раздела повторим основные тезисы.

- Число обусловленности задачи показывает, во сколько раз *относительная* погрешность решения может превышать *относительную* погрешность в начальных данных.
- Величина числа обусловленности, при которой задачу можно считать плохо обусловленной, зависит от параметров используемой машинной арифметики.
- При решении плохо обусловленной задачи обычными методами *нельзя рассчитывать на получение адекватного решения*.

Глава 2

Методы решения СЛАУ

2.1. Метод Гаусса

2.1.1. Введение

Приступим теперь к изучению методов решения систем линейных алгебраических уравнений (СЛАУ)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n, \end{cases}$$

которые будем записывать в матричном виде

$$Ax = b, \tag{2.1}$$

где $A = \left(a_{ij}\right)$ — квадратная матрица размерности n; $x = (x_1, \ldots, x_n)^T$ — искомый вектор решения; $b = (b_1, \ldots, b_n)^T$ — вектор правой части. Везде в дальнейшем будем предполагать, что задача корректно поставлена, т. е. $\det A \neq 0$.

Начнем с методов, которые позволяют найти точное решение системы за конечное число операций (при условии, что все вычисления выполняются точно). Такие методы называются *прямыми* или *точными*.

Все прямые методы решения СЛАУ базируются на простой идее: исходную систему преобразуют к эквивалентной системе (т. е. к системе с тем же решением)

$$Vx = g, \tag{2.2}$$

решение которой находится легко. Например, в наиболее простом случае $V = I$ сразу получаем $x = g$. Но чаще всего исходную систему (2.1) приводят к системе с треугольной матрицей. Так, в случае верхнетреугольной матрицы V система (2.2) имеет вид

$$\begin{cases} v_{11}x_1 + v_{12}x_2 + \ldots + v_{1,n-1}x_{n-1} + v_{1n}x_n = g_1, \\ v_{22}x_2 + \ldots + v_{2,n-1}x_{n-1} + v_{2n}x_n = g_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ v_{n-1,n-1}x_{n-1} + v_{n-1,n}x_n = g_{n-1}, \\ v_{nn}x_n = g_n. \end{cases}$$

Решается такая система путем последовательного выражения x_i через уже известные значения, начиная с x_n:

$$x_i = \frac{1}{v_{ii}} \left(g_i - \sum_{j=i+1}^{n} v_{ij} x_j \right), \quad i = n, n-1, \ldots, 2, 1. \tag{2.3}$$

Вычислительный процесс (2.3) называется *обратной подстановкой* (*обратным ходом*). Заметим, что при $i = n$ верхний предел суммирования будет меньше нижнего, а в таких случаях значение суммы полагается равным нулю. Это соглашение будет нами использоваться в дальнейшем по умолчанию.

Рассмотрим теперь, каким образом осуществляется переход от исходной системы (2.1) к эквивалентной (2.2). Чаще всего для этого последовательно применяют к обеим частям (2.1) некоторые линейные преобразования T_k (умножают обе части системы $Ax = b$ на матрицы T_k):

$$\underbrace{T_N \ldots T_2 T_1 A}_{V} x = \underbrace{T_N \ldots T_2 T_1 b}_{g}.$$

Такой процесс называют *прямым ходом*.

Если в качестве T_k использовать элементарные преобразования, то получим самый известный прямой метод — *метод Гаусса*.

2.1.2. Базовый метод Гаусса

Обозначим $\underline{a}_i$ i-ю строчку матрицы A. Тогда прямой ход метода Гаусса, заключающийся в приведении матрицы системы к верхнетреугольному виду с помощью элементарных преобразований, можно записать в виде следующего алгоритма.

Базовый алгоритм метода Гаусса

1: **for** $k = \overline{1, n-1}$ **do**
2: **for** $i = \overline{k+1, n}$ **do**
3: $l \leftarrow a_{ik} / a_{kk}$
4: $\underline{a}_i \leftarrow \underline{a}_i - l\, \underline{a}_k$
5: $b_i \leftarrow b_i - l\, b_k$
6: **end for**
7: **end for**

Заметим, что при программной реализации в строке 4 необходимо учитывать, что все элементы i-й строки, находящиеся левее k-й позиции, равны нулю. После выполнения данного алгоритма для решения системы остается применить формулы обратной подстановки (2.3) (где $v_{ij} = a_{ij}$, $g_i = b_i$).

Этап алгоритма, определяемый строками 2–4, будем называть *k-м шагом метода Гаусса*. На этом этапе с помощью элементарных преобразований обнуляются элементы k-го столбца, находящиеся ниже главной диагонали. Матрицу системы перед выполнением k-го шага будем обозначать $A^{(k)}$ ($A^{(1)} = A$). Переход от матрицы $A^{(k)}$ к $A^{(k+1)}$ можно представить в виде $A^{(k+1)} = L_k A^{(k)}$, где матрица преобразования L_k имеет следующую структуру:

$$L_k = \left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & l_{k+1}^{(k)} & 1 & & & \\ & & l_{k+2}^{(k)} & & \ddots & & \\ & & \vdots & & & \ddots & \\ & & l_n^{(k)} & & & & 1 \end{array}\right]. \qquad (2.4)$$

(k-й столбец)

Умножение произвольной матрицы M слева на матрицу L_k равносильно добавлению к i-й строке матрицы M k-й строки, умноженной на $l_i^{(k)}$ для всех i от $k+1$ до n. Согласно алгоритму метода Гаусса (см. строку 4), имеем

$$l_i^{(k)} = -a_{ik}^{(k)} / a_{kk}^{(k)}, \qquad (2.5)$$

где $a_{ij}^{(k)}$ — элементы матрицы $A^{(k)}$.

Очевидно, что если хотя бы один из элементов $a_{kk}^{(k)}$ равен нулю, то прямой ход в базовом алгоритме неосуществим. В дальнейшем элементы $a_{kk}^{(k)}$ будем называть *главными*. Если же все главные элементы отличны от нуля, то приведенный алгоритм выполнится успешно.

Пусть $[A]_k$ — матрица, составленная из k первых строк и k первых столбцов матрицы A.

Теорема 2.1. *Базовый алгоритм метода Гаусса осуществим тогда и только тогда, когда все главные угловые миноры матрицы A не равны нулю:* $|[A]_k| \neq 0 \quad \forall k = \overline{1,n}$.

Доказательство. Воспользуемся методом математической индукции. Преобразование L_1 корректно определено и первый шаг метода Гаусса выполним, если (и только если) $a_{11}^{(1)} = a_{11} = |[A]_1| \neq 0$.

Пусть выполнимо k шагов. Это означает, что существует матрица $\tilde{L}_k$,

$$\tilde{L}_k = L_k L_{k-1} \dots L_1, \quad \text{и} \quad A^{(k+1)} = \tilde{L}_k A.$$

Нетрудно заметить, что матрицы $\tilde{L}_k$ имеют блочный вид

$$\left[\begin{array}{c|c} M_k & 0 \\ \hline \boxtimes & I \end{array}\right],$$

где M_k — нижнетреугольная матрица размерности k с единицами на главной диагонали; I — единичная матрица размерности $n-k$.

Запишем равенство $\tilde{L}_k A = A^{(k+1)}$ в блочном виде:

$$\underbrace{\left[\begin{array}{c|c} M_k & 0 \\ \hline \boxtimes & I \end{array}\right]}_{\tilde{L}_k} A = \underbrace{\left[\begin{array}{c|c} U_k & \boxtimes \\ \hline 0 & \boxtimes \end{array}\right]}_{A^{(k+1)}}, \tag{2.6}$$

где U_k — верхнетреугольная матрица размерности k.

Критерием осуществимости $(k+1)$-го шага является условие

$$\theta_{k+1} = a_{k+1,k+1}^{(k+1)} \neq 0,$$

которое гарантирует существование L_{k+1} (см. (2.4), (2.5)).

Из (2.6) имеем

$$[\tilde{L}_k]_{k+1}[A]_{k+1} = [A^{(k+1)}]_{k+1}.$$

Так как $|[\tilde{L}_k]_{k+1}| \neq 0$ и θ_{k+1} — единственный ненулевой элемент в последней строке матрицы $[A^{(k+1)}]_{k+1}$, то

$$\theta_{k+1} \neq 0 \;\Leftrightarrow\; |[A]_{k+1}| \neq 0. \qquad \square$$

Трудоемкость метода Гаусса. Прямой ход базового алгоритма метода Гаусса требует выполнения

$$\frac{n^3}{3} + O(n^2)$$

мультипликативных операций (умножения или деления) и столько же аддитивных операций (сложения или вычитания).

$\triangleright_1$ Докажите это. Оцените трудоемкость обратного хода.

2.1.3. Связь метода Гаусса и LU-разложения

Определение 2.1. LU-разложением невырожденной матрицы A называется ее представление в виде

$$A = LU,$$

где L — нижнетреугольная матрица с единицами на главной диагонали; U — верхнетреугольная матрица.

Теорема 2.2 (связь метода Гаусса и LU-разложения)**.** *Базовый алгоритм метода Гаусса для СЛАУ* (2.1) *выполним тогда и только тогда, когда существует LU-разложение матрицы A.*

Доказательство. Пусть базовый алгоритм осуществим. Тогда из формулы (2.6) при $k = n - 1$ получаем $\tilde{L}_{n-1}A = A^{(n)}$, причем по построению $\tilde{L}_{n-1}$ — нижнетреугольная с единичной главной диагональю, а $A^{(n)}$ — верхнетреугольная. Отсюда получим $A = LU$, где $L = (\tilde{L}_{n-1})^{-1}$, $U = A^{(n)}$.

Необходимость следует из теоремы 2.1: если существует LU-разложение, то нетрудно заметить, что $|[L]_k| \neq 0$ и $|[U]_k| \neq 0$. Следовательно,

$$|[A]_k| = |[L]_k| \cdot |[U]_k| \neq 0. \qquad \square$$

2.1.4. Метод Гаусса с выбором главного элемента

Для того чтобы выполнение алгоритма метода Гаусса не обрывалось при наличии нулевого главного элемента (и не только поэтому), *перед каждым шагом* метода применяется процедура, называемая *выбором главного элемента*. Суть процедуры: путем перестановки строк или столбцов матрицы $A^{(k)}$ поставить на позицию (k, k) ненулевой элемент. При этом, чтобы не «испортить» структуру матрицы, можно использовать лишь последние $n-k$ строк и $n-k$ столбцов. Существует несколько способов выбора главного элемента.

По столбцу: среди элементов $a_{ik}^{(k)}$ для i от k до n выбирается ненулевой элемент $a_{i^*k}^{(k)}$, после чего переставляются местами строки k и i^*.

По строке: среди элементов $a_{kj}^{(k)}$ для j от k до n выбирается ненулевой элемент $a_{kj^*}^{(k)}$, после чего переставляются местами *столбцы* k и j^*.

По матрице: среди элементов $a_{ij}^{(k)}$ для i, j от k до n выбирается ненулевой элемент $a_{i^*j^*}^{(k)}$, после чего переставляются местами *строки* k и i^* и *столбцы* k и j^*.

Рассмотрим следующие вопросы:
1) из каких соображений следует выбирать главный элемент;
2) какой способ выбора главного элемента лучше использовать?

Для ответа рассмотрим еще раз матрицу L_k (2.4). Имеем

$$æ_\infty(L_k) = (1 + \max_i |l_i^{(k)}|)^2, \tag{2.7}$$

откуда с учетом свойства 2 числа обусловленности

$$æ_\infty(A^{(n)}) = æ_\infty(\tilde{L}_{n-1}A) \leqslant æ_\infty(A) \prod_{k=1}^{n-1} (1 + \max_i |l_i^{(k)}|)^2.$$

$\triangleright_2$ Выведите эти формулы.

Таким образом, даже если $æ(A)$ невелико, матрица $A^{(n)}$ может стать плохо обусловленной в случае больших значений $|l_i^{(k)}|$, т. е. *сам процесс метода Гаусса может «испортить» исходную систему*.

Для исправления ситуации мы должны минимизировать величины (2.7). С учетом (2.5) получим следующие ответы:
1) главный элемент должен быть максимальным по модулю среди всех рассматриваемых;
2) выбор главного элемента по столбцу оптимален по соотношению качество/скорость.

2.1.5. Матричные уравнения

Метод Гаусса естественным образом обобщается на случай матричных уравнений вида

$$AX = B, \tag{2.8}$$

где A, как и ранее, — квадратная матрица порядка n; B — матрица размеров $n \times m$; X — неизвестная матрица тех же размеров, что и B. Возможно два подхода к решению таких уравнений.
1) Система (2.8) эквивалентна набору из m СЛАУ вида

$$Ax^{(j)} = b^{(j)}, \quad j = \overline{1, m},$$

где $x^{(j)}$ и $b^{(j)}$ — столбцы матриц X и B. Решая эти m систем, найдем искомую матрицу X.

2) Матричный метод Гаусса. Для того чтобы адаптировать построенный выше алгоритм к решению матричных уравнений, достаточно строку 5 заменить на

$$\underline{b}_i \leftarrow \underline{b}_i - l\,\underline{b}_k$$

(здесь $\underline{b}_i$ — строки матрицы B), а также выполнить соответственно m алгоритмов обратного хода (для каждого из столбцов $b^{(j)}$).

2.1.6. Обращение матрицы и вычисление определителя

Обращение матрицы эквивалентно решению матричного уравнения

$$AX = I,$$

где I — единичная матрица. Для решения этого уравнения могут использоваться оба описанных выше способа. Если же известно LU-разложение матрицы A, то вычислить обратную можно следующими способами [17, п. 14.3]:
1) сначала вычислить U^{-1}, после чего решить матричное уравнение $XL = U^{-1}$;
2) найти U^{-1} и L^{-1}, затем $X = A^{-1} = U^{-1}L^{-1}$.

$\triangleright_3$ Постройте алгоритм обращения треугольной матрицы.

$\triangleright_4$ Постройте соответствующие алгоритмы для обоих указанных выше способов обращения матрицы.

Определитель матрицы также вычисляется с помощью метода Гаусса:

$$\tilde{L}_{n-1}A = A^{(n)} \;\Rightarrow\; 1 \cdot |A| = |A^{(n)}| = a_{11}^{(n)} a_{22}^{(n)} \dots a_{nn}^{(n)}.$$

Однако при этом нужно помнить про важный нюанс: если в ходе метода переставлялись строки и столбцы, то каждая такая операция *меняла знак определителя на противоположный*. Поэтому окончательная формула такова:

$$|A| = (-1)^p a_{11}^{(n)} a_{22}^{(n)} \dots a_{nn}^{(n)}, \tag{2.9}$$

где p — количество перестановок строк и столбцов в ходе метода.

$\triangleright_5$ Как вычислить определитель, если известно LU-разложение матрицы?

2.1.7. Метод прогонки

Рассмотрим СЛАУ

$$\begin{bmatrix} d_1 & e_1 & & & & \\ c_2 & d_2 & e_2 & & & \\ & c_3 & d_3 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & d_{n-1} & e_{n-1} \\ & & & & c_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}. \tag{2.10}$$

Матрицы такой структуры называются *трехдиагональными*. В приложениях достаточно часто встречаются такие системы. Особая структура этой матрицы позволяет найти решение системы методом Гаусса за $O(n)$ операций. Получаемый метод называется *методом прогонки*.

Алгоритм метода прогонки

1: **for** $k = \overline{2, n}$ **do** `// Прямой ход`
2: $\quad d_k \leftarrow d_k - e_{k-1}c_k/d_{k-1}$
3: $\quad b_k \leftarrow b_k - b_{k-1}c_k/d_{k-1}$
4: **end for**
5: $x_n = b_n/d_n$
6: **for** $k = \overline{n-1, 1}$ **do** `// Обратный ход`
7: $\quad x_k \leftarrow (b_k - e_k x_{k+1})/d_k$
8: **end for**

Замечание 2.1. Данный алгоритм отличается от традиционного метода прогонки, основанного на так называемом алгоритме Томаса и приведенного в большинстве учебников (см., например, [10, с. 45]). Метод Томаса также эквивалентен методу Гаусса с той лишь разницей, что в течение прямого хода метода диагональные элементы d_k становятся единичными.

При выполнении алгоритма прогонки мы лишены возможности выбора главного элемента, так как при этом нарушилась бы трехдиагональная структура матрицы A. Следовательно, метод прогонки осуществим тогда и только тогда, когда все главные миноры матрицы отличны от нуля. Существует также более простое для проверки достаточное условие осуществимости метода прогонки. Для его доказательства нам понадобятся следующие предварительные сведения.

Определение 2.2. Если элементы матрицы A удовлетворяют усло-

виям

$$|a_{ii}| \geqslant \sum_{j \neq i} |a_{ij}| \quad \forall i = \overline{1, n}, \tag{2.11}$$

то говорят, что такая матрица обладает свойством *диагонального преобладания (диагонального доминирования)*. Если неравенство в (2.11) строгое, говорят о *строгом диагональном преобладании*.

Теорема 2.3. *Если матрица обладает свойством строгого диагонального преобладания, то все ее главные миноры отличны от нуля.*

Следствие 2.1. *Если матрица системы* (2.10) *удовлетворяет условиям*

$$|d_1| > |e_1|, \quad |d_i| > |c_i| + |e_i| \quad \forall i = \overline{2, n-1}, \quad \textit{и} \quad |d_n| > |c_n|,$$

то алгоритм прогонки выполним.

$\triangleright_6$ На основе метода прогонки постройте экономичный алгоритм решения трехдиагональных СЛАУ с выбором главного элемента. Такой алгоритм может быть полезен в случаях, когда матрица системы не обладает диагональным преобладанием.

2.2. LU-разложение

Зачем может понадобиться LU-разложение матрицы?

Если известно разложение $A = LU$, то решение СЛАУ $Ax = b$ может быть выполнено на порядок эффективнее, чем методом Гаусса. Для этого нужно решить две системы с треугольными матрицами:

$$Ax = b \quad \Leftrightarrow \quad LUx = b \quad \Leftrightarrow \quad \begin{cases} Ly = b, \\ Ux = y. \end{cases}$$

Сначала находится y (прямой подстановкой), затем решение исходной системы — вектор x (обратной подстановкой).

$\triangleright_1$ Запишите алгоритм вычисления y и x.

Нетрудно убедиться, что трудоемкость решения СЛАУ в этом случае равна $O(n^2)$. Особенно эффективен этот подход в ситуациях, когда решается несколько систем с одной и той же матрицей A: разложение строится один раз (как мы скоро увидим, его построение практически ничем не отличается от прямого хода метода Гаусса и имеет такую же трудоемкость), зато все последующие системы решаются за $O(n^2)$ операций.

2.2.1. LU-разложение без выбора главного элемента

Пусть главные угловые миноры матрицы A отличны от нуля. Тогда осуществим прямой ход метода Гаусса, представляемый в виде

$$L_{n-1} \dots L_2 L_1 A = U. \tag{2.12}$$

Матрица U уже получена, осталось разобраться как найти L. Каждая из нижнетреугольных матриц L_k в (2.12) представляет собой единичную матрицу, к которой применены те же самые элементарные преобразования, которые применялись к матрице A при обнулении элементов под главной диагональю в k-м столбце. Другими словами, L_k это матрица с единицами на главной диагонали, а остальные ненулевые ее элементы могут находиться только в k-м столбце ниже диагонали.

Из последнего тождества имеем

$$A = LU,$$

где

$$L = (L_{n-1} \dots L_2 L_1)^{-1} = L_1^{-1} L_2^{-1} \dots L_{n-1}^{-1}. \tag{2.13}$$

Нетрудно заметить, что

L_k^{-1} может быть получена из исходной матрицы L_k путем инверсии знака у поддиагональных элементов,

или

$$L_k^{-1} = 2I - L_k.$$

Для вычисления матрицы L согласно (2.13) нужно последовательно применить к L_{n-1}^{-1} элементарные преобразования L_{n-2}^{-1}, L_{n-3}^{-1} и так далее до L_1^{-1}. Учитывая структуру матриц L_k^{-1}, легко увидеть, что

k-й столбец матрицы L в точности равен k-му столбцу матрицы L_k^{-1}.

Это правило является основой приведенного ниже алгоритма построения LU-разложения матрицы. Также из него следует, что матрица L может быть записана в виде

$$L = nI - L_1 - L_2 - \dots - L_{n-1}. \tag{2.14}$$

Учитывая все вышеизложенное, мы приходим к тому, что алгоритм построения LU разложения без выбора главного элемента имеет минимальные отличия от базового метода Гаусса:

for $k = \overline{1, n-1}$ **do**
  for $i = \overline{k+1, n}$ **do**
    $a_{ik} \mathrel{/}= a_{kk}$
    $a_{i,(k+1):n} \mathrel{-}= a_{ik}\, a_{k,(k+1):n}$
  end for
end for

Здесь обозначение $j_1 : j_2$ обозначает диапазон элементов с индексами от j_1 до j_2. В частности, $a_{i,(k+1):n}$ — это элементы i-й строки матрицы A, находящиеся в столбцах с $(k+1)$-го по n-й. В результате выполнения алгоритма верхний треугольник матрицы A, включая диагональ, будет содержать элементы матрицы U, а нижний треугольник — внедиагональные элементы матрицы L.

2.2.2. LU-разложение с выбором главного элемента

Рассмотрим теперь прямой ход метода Гаусса с выбором главного элемента по столбцу. В этом случае вместо (2.12) будем иметь

$$L_{n-1}P_{n-1}\dots L_2P_2L_1P_1A = U,$$

где P_k — матрица перестановки, которая применяется к матрице $A^{(k)}$ перед выполнением k-го шага.

Рассмотрим подробно частный случай при $n = 4$. Матрицы L_k в этом случае будут иметь общий вид

$$L_1 = \begin{bmatrix} 1 & & & \\ \alpha_1 & 1 & & \\ \beta_1 & & 1 & \\ \gamma_1 & & & 1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & \beta_2 & 1 & \\ & \gamma_2 & & 1 \end{bmatrix}, \quad L_3 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & \gamma_3 & 1 \end{bmatrix}.$$

Пусть матрицы перестановок равны

$$P_1 = \Pi_{12}, \quad P_2 = \Pi_{24}, \quad P_3 = \Pi_{34},$$

где Π_{km} — единичная матрица с переставленными k-й и m-й строками. В тождестве

$$L_3P_3L_2P_2L_1P_1A = U$$

рассмотрим отдельно произведение P_2L_1, учитывая, что умножение на матрицу Π_{km} справа соответствует перестановке k-го и m-го столбцов:

$$P_2L_1 = \Pi_{24}L_1 = \begin{bmatrix} 1 & & & \\ \gamma_1 & & & 1 \\ \beta_1 & & 1 & \\ \alpha_1 & 1 & & \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ \beta_1 & & 1 & \\ \alpha_1 & & & 1 \end{bmatrix} P_2 = L_1'P_2,$$

откуда получим

$$L_3P_3L_2L_1'P_2P_1A = U.$$

Матрица L_1' представляет собой матрицу L_1, у которой элементы (только) первого столбца переставлены в соответствии с $P_2 = \Pi_{24}$, что также можно записать как

$$L_1' = P_2L_1P_2,$$

так как $P_2^{-1} = P_2$. Продолжая по аналогии далее, будем иметь

$$P_3 L_2 = \Pi_{34} L_2 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & \gamma_2 & & 1 \\ & \beta_2 & 1 & \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & \gamma_2 & 1 & \\ & \beta_2 & & 1 \end{bmatrix} P_3 = L_2' P_3,$$

$$P_3 L_1' = \Pi_{34} L_1' = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ \alpha_1 & & & 1 \\ \beta_1 & & 1 & \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ \alpha_1 & & 1 & \\ \beta_1 & & & 1 \end{bmatrix} P_3 = L_1'' P_3.$$

Собирая все вместе, приходим к тождеству

$$L_3 P_3 L_2 P_2 L_1 P_1 A = L_3 L_2' L_1'' P_3 P_2 P_1 A = U,$$

откуда получаем

$$PA = LU,$$

где

$$P = P_3 P_2 P_1, \quad L = (L_1'')^{-1} (L_2')^{-1} L_3^{-1}.$$

$\triangleright_2$ Выразите матрицы L_1'' и L_2' через L_1, L_2, P_2 и P_3.

Благодаря специальной структуре множителей, по аналогии с (2.13), матрицу L легко вычислить:

$$L = \begin{bmatrix} 1 & & & \\ -\gamma_1 & 1 & & \\ -\alpha_1 & -\gamma_2 & 1 & \\ -\beta_1 & -\beta_2 & -\gamma_3 & 1 \end{bmatrix}.$$

В общем случае матрица L может быть представлена в виде (см. (2.14))

$$L = nI - L_{n-1} - L_{n-2}' - L_{n-3}'' - \ldots - L_1^{(n-2)}. \tag{2.15}$$

Здесь верхний индекс в скобках обозначает количество штрихов:

$$L_k^{(q)} = L_k^{\prime\ldots\prime} = P_{q+1} L_k^{(q-1)} P_{q+1}, \quad 1 \leqslant k \leqslant n-2, \quad q \geqslant 1.$$

Приведенные выше рассуждения можно обобщить на общий случай и по индукции доказать следующую теорему.

Теорема 2.4 (*LU*-разложение с выбором главного элемента)**.** *Пусть* $\det A \neq 0$. *Тогда матрица A представима в виде*

$$PA = LU, \tag{2.16}$$

где P — матрица перестановки, L — нижнетреугольная матрица с единичной главной диагональю, U — верхнетреугольная матрица с ненулевыми элементами на главной диагонали.

$\triangleright_3$ Докажите теорему.

Практическая реализация

Для построения алгоритма вычисления LU-разложения с выбором главного элемента, т. е. разложения (2.16) достаточно внести лишь небольшие изменения в базовый алгоритм на стр. 28. Главное наблюдение, необходимое для этого, уже было сделано выше: матрицы $L_k^{(q)}$, по которым строится матрица L (2.14), представляют собой матрицу L_k, к k-му столбцу которой (и только к нему!) последовательно применены перестановки P_{k+1}, P_{k+2}, …, P_{n-1}. Учитывая, что элементы матрицы L хранятся на месте обнуляемых элементов исходной матрицы A, указанные преобразования P_q будут применяться к L_k автоматически при перестановке строк матрицы A.

Что же касается итоговой матрицы перестановки

$$P = P_{n-1} \dots P_2 P_1,$$

то нет необходимости хранить ее в явном виде. Достаточно вместо этого строить в процессе разложения вектор перестановок p. Он получается применением перестановок, осуществляемых в ходе алгоритма, к вектору $(1, 2, \dots, n)^T$.

Если разложение (2.16) построено, то решение системы $Ax = b$ осуществляется по схеме

$$PAx = LUx = Pb = b',$$

то есть сначала нужно применить к вектору b перестановку P, переставив его компоненты в соответствии с вектором p:

$$b'_i = b_{p_i}, \quad i = \overline{1, n}.$$

После этого, как обычно, из уравнения $Ly = b'$ прямой подстановкой находится вектор y, а затем из $Ux = y$ обратной подстановкой находится искомое решение x.

$\triangleright_4$ Постройте полный алгоритм LU-разложения с выбором главного элемента.

2.3. Разложение Холецкого и его друзья

Системы линейных уравнений с симметричными матрицами достаточно часто возникают на практике. В этой лекции мы изучим базовые алгоритмы решения таких систем.

2.3.1. LDL^T-разложение

Теорема 2.5 (LDL^T-разложение)**.** *Пусть $A = A^T$ — симметричная вещественная матрица. Если все ее главные угловые миноры отличны от нуля, то существует разложение*

$$A = LDL^T, \tag{2.17}$$

где L — нижнетреугольная матрица с единичной главной диагональю, D — диагональная матрица.

Доказательство. По условию существует LU-разложение

$$A = LU = U^T L^T = A^T,$$

откуда $U = L^{-1}U^T L^T$. Обозначим $D = L^{-1}U^T$, тогда $U = DL^T$ и

$$A = LU = LDL^T.$$

Рассмотрим подробнее матрицу $D = L^{-1}U^T$. С одной стороны, D — нижнетреугольная, так как является произведением нижнетреугольных матриц L^{-1} и U^T. С другой стороны, $D = U(L^T)^{-1}$, т. е. D является еще и верхнетреугольной. Следовательно, D — диагональная матрица, что и требовалось доказать. □

Из теоремы 2.5 получаем, что LDL^T-разложение симметричной матрицы можно найти, просто выполнив ее LU-разложение. Так как $U = DL^T$, а у L единичная главная диагональ, матрицу D легко найти:

$$D = \operatorname{diag}\{u_{11}, u_{22}, \ldots, u_{nn}\}.$$

Можно поступить и немного по-другому: просто найти матрицу U прямым ходом метода Гаусса (без построения L). Далее имеем

$$U = DL^T \;\Rightarrow\; L = (D^{-1}U)^T, \tag{2.18}$$

то есть матрица L^T находится путем деления каждой строки U на ее диагональный элемент.

Однако, оба этих способа не имеют большого смысла если для выполнения прямого хода метода Гаусса (для построения LU-разложения) мы будем использовать стандартный алгоритм. Самое главное здесь — использовать свойство симметрии матрицы A, так как *прямой ход метода Гаусса можно выполнить в два раза быстрее, если матрица симметрична*. Чтобы в этом убедиться, рассмотрим подробно первый шаг метода для матрицы A. Элементы матрицы $A^{(2)}$, которая получится после этого шага, обозначим a'_{ij}. Тогда по определению для всех i, j от 2 до n будем иметь

$$a'_{ij} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}.$$

Аналогично, учитывая симметрию матрицы A, получим

$$a'_{ji} = a_{ji} - \frac{a_{j1}}{a_{11}} a_{1i} = a_{ij} - \frac{a_{1j}}{a_{11}} a_{i1} = a'_{ij}. \tag{2.19}$$

То есть, после первого шага метода Гаусса матрица системы, не считая первые столбец и строку, остается симметричной! Понятно, что по индукции это утверждение будет выполняться и на всех последующих шагах метода Гаусса. Следовательно, формула (2.19) дает способ практически вдвое сократить количество арифметических операций при построении LU-разложения симметричной матрицы: на каждом шаге метода можно вычислять только элементы верхнего (или нижнего) треугольника, включая диагональ. Один из возможных вариантов алгоритма приведен ниже. После его выполнения на месте верхнего треугольника матрицы A будет находиться матрица L^T (без диагонали), а на диагонали — элементы матрицы D. Обратите внимание на минимальные отличия от метода Гаусса.

1: **for** $k = \overline{1, n-1}$ **do**
2: **for** $i = \overline{k+1, n}$ **do**
3: $a_{i,i:n} \mathrel{-}= (a_{ki}/a_{kk}) a_{k,i:n}$
4: **end for**
5: $a_{k,k+1:n} \mathrel{/}= a_{kk}$
6: **end for**

$\triangleright_1$ Постройте аналогичный алгоритм, который заменяет нижний треугольник A на элементы матрицы L. Какой из этих двух алгоритмов более эффективен при реализации на компьютере? Почему?

$\triangleright_2$ Запишите схему решения СЛАУ $Ax = b$ если известно LDL^T-разложение матрицы A.

О вычислительной устойчивости метода. Так как при выполнении LDL^T-разложения мы не выполняем выбора главного элемента, рассмотренный алгоритм не может быть применим, если матрица A имеет нулевые главные миноры. Более того, если при $\det[A]_k \approx 0$ метод формально пригоден, то по факту в этом случае обусловленность итоговой матрицы L может быть слишком большой. Поэтому LDL^T-разложение является вычислительно неустойчивым методом решения СЛАУ и применять его в машинной арифметике следует с осторожностью.

2.3.2. Метод квадратного корня

Методом квадратного корня называют метод решения СЛАУ с симметричной и положительно определенной матрицей, основанный на разложении Холецкого. Прежде, чем изучить это разложение, рассмотрим еще раз разложение $A = LDL^T$. Диагональную матрицу D можно представить в виде

$$D = HEH,$$

где H и E — диагональные матрицы, элементы которых равны

$$h_{ii} = \sqrt{|d_{ii}|}, \quad e_{ii} = \operatorname{sign} d_{ii}. \tag{2.20}$$

В силу невырожденности матрицы D такое разложение на множители всегда осуществимо. Тогда

$$A = LDL^T = LHEH^T L^T = GEG^T,$$

где

$$G = LH.$$

Таким образом, следствием теоремы 2.5 является

Теорема 2.6. *Если все главные угловые миноры симметричной вещественной матрицы A отличны от нуля, то существует разложение*

$$A = GEG^T, \tag{2.21}$$

где G — нижнетреугольная матрица, E — диагональная матрица, элементы которой по модулю равны 1.

$\triangleright_3$ Опишите схему решения системы $Ax = b$, если известно разложение $A = GEG^T$.

Так как $G = LH$, элементы матрицы G могут быть вычислены как

$$g_{ij} = l_{ij}\sqrt{|d_{jj}|}.$$

С другой стороны, если известна матрица $U = DL^T$, то из тождеств $G^T = HL^T$ и (2.18) получим $G^T = HD^{-1}U = E^{-1}H^{-1}U$, то есть

$$g_{ji} = e_{ii}^{-1}h_{ii}^{-1}u_{ij} = \frac{\operatorname{sign} u_{ii}}{\sqrt{|u_{ii}|}}u_{ij}. \tag{2.22}$$

$\triangleright_4$ Запишите алгоритм построения GEG^T-разложения на основе прямого хода метода Гаусса в двух вариантах: а) заменяя верхний треугольник матрицы A на матрицу G^T, б) заменяя нижний треугольник A на G.

Симметричные положительно определенные матрицы

Вспомним теперь некоторые факты из линейной алгебры.

Определение 2.3. Квадратная вещественная матрица A называется *положительно определенной* ($A > 0$), если

$$(Ax, x) > 0 \quad \forall x \in \mathbb{R}^n,\ x \neq 0.$$

В дальнейшем нам понадобятся следующие свойства положительно определенных матриц:

1) $A^T = A > 0$ тогда и только тогда, когда все ее главные угловые миноры положительны (критерий Сильвестра);

2) $A^T = A > 0$ тогда и только тогда, когда все собственные значения A вещественны и положительны.

В частности, по критерию Сильвестра все главные угловые миноры симметричной положительно определенной матрицы отличны от нуля, то есть у таких матриц всегда существуют LDL^T- и GEG^T-разложения (метод Гаусса без выбора главного элемента всегда осуществим). Кроме этого, положительная определенность матрицы A благотворно влияет на вычислительную устойчивость, так что метод квадратного корня (разложение Холецкого), в отличие от LDL^T-разложения, не обладает повышенной чувствительностью к погрешностям округления.

Разложение Холецкого

Теорема 2.7 (разложение Холецкого)**.** *Для любой положительно определенной вещественной матрицы A справедливо разложение*

$$A = GG^T, \tag{2.23}$$

где G — некоторая нижнетреугольная матрица.

Доказательство. По критерию Сильвестра имеем $\det[A]_k > 0, \quad \forall k = \overline{1,n}$, то есть существует разложение $A = GEG^T$. Нетрудно заметить, что $[A]_k = [G]_k[E]_k[G^T]_k$, откуда

$$\det[A]_k = (\det[G]_k)^2 \det[E]_k > 0,$$

следовательно, $\det[E]_k = 1$ для всех $k = \overline{1,n}$, то есть $E = I$. □

Из теоремы 2.7 следует, что в случае $A = A^T > 0$ формула (2.22) превращается просто в

$$g_{ji} = \frac{1}{\sqrt{u_{ii}}} u_{ij},$$

то есть

для вычисления матрицы G^T нужно просто разделить каждую строку матрицы U (которая получается в результате прямого хода метода Гаусса) на корень из диагонального элемента в этой строке.

Положительная определенность матрицы A при этом гарантирует, что $u_{ii} = d_{ii} > 0$. Естественно, прямой ход метода Гаусса следует выполнять рационально, с учетом симметрии матрицы, как мы делали это ранее.

$\triangleright_5$ Запишите алгоритм построения разложения Холецкого на основе (оптимизированного) прямого хода метода Гаусса в двух вариантах: а) заменяя верхний треугольник матрицы A на матрицу G^T, б) заменяя нижний треугольник A на G.

$\triangleright_6$ Опишите схему решения системы $Ax = b$, если известно разложение Холецкого $A = GG^T$.

2.4. Метод отражений

2.4.1. Ортогональные преобразования

Определение 2.4. Линейное преобразование $Q : \mathbb{R}^n \to \mathbb{R}^n$ называется *ортогональным*, если оно сохраняет евклидову норму векторов:

$$\|Qx\|_2 = \|x\|_2 \quad \forall x \in \mathbb{R}^n.$$

Матрица ортогонального преобразования называется ортогональной.

Возможны также следующие эквивалентные определения ортогонального преобразования:

- преобразование ортогонально тогда и только тогда, когда оно сохраняет скалярное произведение:

$$(Qx, Qy) = (x, y) \quad \forall x, y \in \mathbb{R}^n;$$

- преобразование ортогонально тогда и только тогда, когда его матрица удовлетворяет условию

$$Q^{-1} = Q^T.$$

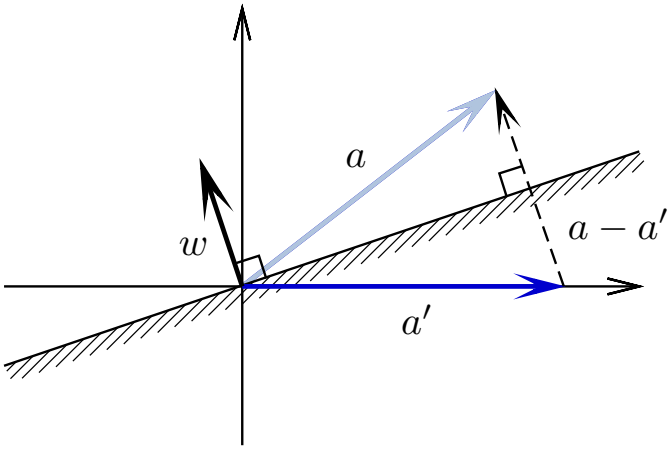
Последняя формулировка наиболее часто приводится в учебниках в качестве определения ортогональных матриц.

Свойства ортогональных матриц

1) Строки и столбцы ортогональной матрицы образуют ортонормированные системы векторов.
2) Определитель ортогональной матрицы по модулю равен 1.
3) Число обусловленности ортогональной матрицы в спектральной норме равно 1.

2.4.2. Преобразование отражения

Важным представителем класса ортогональных преобразований является преобразование отражения, с которым мы сейчас познакомимся. Рассмотрим n-мерном в евклидовом пространстве вектор w единичной длины: $\|w\|_2 = 1$. Этот вектор определяет гиперплоскость W — множество векторов, ортогональных ему. Любой вектор $a \in \mathbb{R}^n$ можно отразить относительно этой гиперплоскости. Используя геометрическую интуицию в двух- или трехмерном случае, можно записать условие, которому должен удовлетворять вектор a':



$$(a' - a) \perp W, \quad \text{или} \quad a' - a = \gamma w, \quad \gamma \in \mathbb{R}. \tag{2.24}$$

Отсюда получим $a' = a + \gamma w$. Из еще одного очевидного свойства

$$(a + a') \in W, \quad \text{или} \quad (a + a', w) = 0$$

легко найти неизвестный коэффициент γ получить правило вычисления отраженного вектора:

$$a' = a - 2(a, w)w. \tag{2.25}$$

Легко видеть, что преобразование отражения линейно, то есть задается некоторой матрицей. Найдем вид этой матрицы:

$$a' = a - 2(a, w)w = Ia - 2w(w^T a) = Ia - 2(ww^T)a = (I - 2ww^T)a,$$

то есть (2.25) эквивалентно $a' = Ha$, где H — матрица отражения, однозначно определяемая вектором нормали w по формуле

$$H = I - 2ww^T. \tag{2.26}$$

Преобразование называют также преобразованием Хаусхолдера (Householder).

Замечание 2.2. Заметим, что вычисление преобразования отражения по формуле (2.25) требует $O(n)$ операций умножения и сложения, в то время как умножение на матрицу H требует $O(n^2)$ операций. Поэтому в явном виде матрицу отражения практически никогда не используют, а хранят только w.

$\triangleright_1$ Докажите, что а) H ортогональна, б) $H^{-1} = H$.

2.4.3. Метод отражений для решения СЛАУ

Рассмотрим систему $Ax = b$. Если для приведения матрицы A к верхнетреугольному виду вместо элементарных преобразований использовать преобразования отражения, получим новый метод решения СЛАУ — метод отражений.

Как и ранее, на k-м шаге прямого хода будут обнуляться все элементы k-го столбца, расположенные ниже главной диагонали. Матрицу системы перед выполнением k-го шага будем обозначать $A^{(k)}$. В итоге мы должны получить

$$Q_{n-1} \dots Q_2 Q_1 A = R,$$

где Q_k — ортогональные матрицы, определяющие преобразования отражения, а R — верхнетреугольная матрица, аналог матрицы U в LU-разложении.

Рассмотрим подробно первый шаг, для краткости обозначая $\|\cdot\| = \|\cdot\|_2$. Пусть a_1 — первый столбец матрицы A. После первого шага этот вектор должен быть преобразован в

$$a'_1 = (a'_{11}, 0, \dots, 0)^T.$$

Так как это преобразование является ортогональным, имеем $\|a_1\| = \|a'_1\|$, откуда

$$a'_{11} = \pm\|a_1\|. \tag{2.27}$$

Теперь необходимо найти преобразование отражения, которое отображает a_1 в a'_1. Соответствующий вектор нормали w_1 можно легко найти из (2.24). Учитывая, что норма этого вектора должна быть единичной, будем иметь

$$w_1 = \|a_1 - a'_1\|^{-1}(a_1 - a'_1). \tag{2.28}$$

Для того, чтобы избежать вычитания близких чисел при использовании этой формулы, следует воспользоваться свободой выбора знака для a'_{11} в формуле (2.27). Чтобы операция вычитания на самом деле представляла собой сложение одинаковых по модулю чисел, следует выбирать

$$a'_{11} = (-\operatorname{sign} a_{11})\|a_1\|.$$

Итак, мы нашли преобразование отражения, которое приводит первый столбец матрицы A к нужному виду: $Q_1 a_1 = a'_1$, где Q_1 — матрица отражения $Q_1 = I - 2w_1 w_1^T$, которую нет смысла вычислять в явном виде.

Чтобы получить $A^{(2)} = Q_1 A$ нам нужно применить Q_1 к остальным столбцам a_j матрицы A, что можно сделать по формуле (2.25):

$$a'_j = a_j - 2(a_j, w_1)w_1, \quad j = \overline{2, n}.$$

Не забудем также и про вектор правой части:

$$b' = b^{(2)} = b - 2(b, w_1)w_1.$$

Первый шаг метода отражений выполнен.

Следующие шаги выполняются аналогично: на k-ом шаге описанные выше преобразования применяются к подматрице, составленной из последних $n - k + 1$ строк и столбцов матрицы $A^{(k)}$. Соответствуюшее преобразование $A^{(k+1)} = Q_k A^{(k)}$ задается блочной матрицей

$$Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & H_k \end{bmatrix},$$

где H_k — матрица отражения, определяемая вектором нормали $w_k \in \mathbb{R}^{n-k+1}$, который вычисляется аналогично (2.28). Как видно, на каждом шаге в столбцах матрицы $A^{(k)}$ и в векторе b под действием преобразований Q_k будут меняться компоненты с индексами от k до n. Еще раз подчеркнем, что матрицы H_k и Q_k в явном виде не строятся, а все преобразования отражения выполняются по правилу (2.25).

$\triangleright_2$ Запишите алгоритм прямого хода метода отражений для решения СЛАУ.

По окончании прямого хода останется обратной подстановкой решить систему $Rx = b^{(n)}$, где $R = A^{(n)}$ — итоговая верхнетреугольная матрица, а $b^{(n)}$ — преобразованный вектор правой части:

$$b^{(n)} = Q_{n-1} \dots Q_1 b.$$

Отметим, что $b^{(n)}$ можно вычислить и после прямого хода, если сохранены все векторы нормали w_k, которые строились во время прямого хода. В этом случае можно назвать описанный подход решением СЛАУ методом QR-разложения (см. далее).

Трудоемкость прямого хода метода отражений в два раза больше, чем у метода Гаусса.

$\triangleright_3$ Докажите это утверждение.

$\triangleright_4$ Как в ходе метода отражений распознать случай вырожденной матрицы A?

2.4.4. QR-разложение

Определение 2.5. Пусть A — матрица размера $m \times n$, причем $m \geqslant n$. *QR-разложением* такой матрицы называется представление ее в виде

$$A = QR,$$

где Q — ортогональная матрица ($Q^{-1} = Q^T$) размера $m \times m$, R — верхнетреугольная матрица ($r_{ij} = 0 \quad \forall i > j$), размер которой совпадает с размером матрицы A.

Приведем пример для случая $m = 5, n = 3$:

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & \\ & & \end{bmatrix}}_{R}$$

Заметим, что при $m > n$ последние $m-n$ столбцов матрицы Q никак не влияют на элементы матрицы A (так как последние $m - n$ строк матрицы R равны нулю), поэтому часто в таких случаях эти лишние столбцы и строки отбрасывают и рассматривают так называемое *сокращенное QR-разложение* $A = \hat{Q}\hat{R}$:

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{\hat{Q}} \underbrace{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \end{bmatrix}}_{\hat{R}}.$$

В общем случае размеры матриц $\hat{Q}$ и $\hat{R}$ равны $m \times n$ и $n \times n$ соответственно. Матрицу $\hat{Q}$ уже нельзя считать ортогональной, но у нее остается важное свойство ортогональных матриц: все ее n столбцов образуют ортонормированную систему, то есть $\hat{Q}^T\hat{Q} = I$.

Построение QR-разложения методом отражений

Легко увидеть, что в случае квадратной матрицы прямой ход метода отражений, описанный в пункте 2.4.3, представляет собой процесс

построения QR-разложения матрицы A:

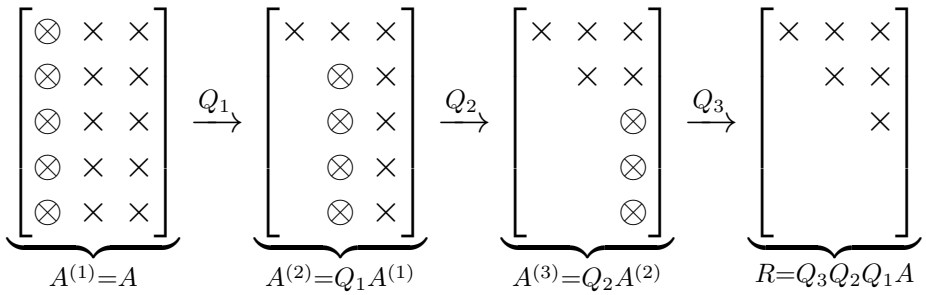
$$Q_{n-1} \dots Q_2 Q_1 A = R,$$

откуда получаем $A = QR$, где R — верхнетреугольная матрица, а ортогональная матрица Q имеет вид

$$Q = (Q_{n-1} \dots Q_1)^{-1} = Q_1 \dots Q_{n-1},$$

так как $Q_k^{-1} = Q_k^T = Q_k$.

При $m > n$ суть метода остается той же, только для приведения матрицы A к верхнетреугольному виду нужно будет n (а не $n-1$) преобразований отражения. Рассмотрим схему построения разложения для $m = 5, n = 3$:



Кружками на этой схеме выделены компоненты вектора, который на соответствующем этапе играет роль вектора a_1, по которому строится нормаль (2.28), определяющая преобразование отражения.

В общем случае получится

$$\underbrace{Q_n \dots Q_2 Q_1}_{Q^{-1}} A = R,$$

откуда

$$Q = (Q_n \dots Q_2 Q_1)^{-1} = Q_1 Q_2 \dots Q_n, \qquad (2.29)$$

$$Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & H_k \end{bmatrix},$$

где I_{k-1} — единичная матрица размера $k-1$, H_k — матрица отражения размера $m-k+1$, R — верхнетреугольная матрица.

Теперь вспомним, что вместо матрицы Q проще хранить векторы нормали $w_1, \dots, w_n$, которые однозначным образом определяют матрицы $H_k = I - 2w_k w_k^T$ и, следовательно, «сборную» матрицу Q по формуле

(2.29). В силу блочной структуры матрицы Q_k при умножении на вектор она изменяет в нем лишь координаты с индексами от k до m.

На практике обычно требуется вычислять преобразование Q^T,

$$Q^T = Q^{-1} = Q_n \dots Q_2 Q_1.$$

Если использовать стандартное обозначение $x_{k:m} = (x_k,\, x_{k+1},\, \dots,\, x_m)^T$, алгоритм вычисления $x \leftarrow Q^T x$ можно записать в следующем виде:

1: **for** $k = \overline{1, n}$ **do**
2: $\quad x_{k:m} \leftarrow x_{k:m} - 2(x_{k:m}, w_k)w_k$
3: **end for**

2.5. Задача наименьших квадратов

2.5.1. Постановка задачи

Рассмотрим переопределенную систему линейных уравнений, то есть систему, в которой количество неизвестных n меньше количества уравнений m:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}. \tag{2.30}$$

Матрица системы A является прямоугольной, строк в ней больше чем столбцов. В дальнейшем будем предполагать, что ранг этой матрицы равен n, то есть все ее столбцы линейно независимы. Решение системы (2.30) существует лишь в случае, если вектор $b \in \mathbb{R}^m$ принадлежит линейной оболочке системы векторов-столбцов матрицы A:

$$b \in V = \operatorname{span}\{a_1, \dots, a_n\}.$$

Чтобы решение существовало при всех b нужно придать уравнениям (2.30) несколько иной смысл, а именно: будем искать вектор $x \in \mathbb{R}^n$ такой, который минимизировал бы норму невязки $Ax - b$:

$$\|Ax - b\|_2 \to \min. \tag{2.31}$$

Задачи такого рода часто возникают в приложениях. Например, пусть задан набор точек на плоскости (ξ_i, η_i), $i = \overline{1, m}$. Требуется найти многочлен φ степени $n-1$, график которого был бы «как можно более близок» к этим точкам. Это условие можно сформулировать как

$$\sum_{i=1}^{m} (\varphi(\xi_i) - \eta_i)^2 \to \min.$$

Подставляя сюда $\varphi(\xi) = \sum_{j=1}^{n} c_j \xi^{j-1}$, получаем линейную задачу наименьших квадратов для нахождения коэффициентов (c_j):

$$\|Ac - b\|_2 \to \min,$$

где $a_{ij} = \xi_i^{j-1}$, $\quad i = \overline{1, m}$, $\quad j = \overline{1, n}$, $\quad c = (c_1, \dots, c_n)^T$, $\quad b = (\eta_1, \dots, \eta_m)^T$.

2.5.2. Решение с помощью нормальных уравнений

Определение 2.6. Ортогональной проекцией вектора $x \in \mathbb{R}^m$ на подпространство $V \subset \mathbb{R}^m$ называется вектор $y \in V$ такой, что $x - y \perp v$ $\forall v \in V$.

Если известен базис $\{a_1, \ldots, a_n\}$ подпространства V, то эквивалентным определением можно считать следующее: вектор $y \in V$ является ортогональной проекцией вектора x, если

$$x - y \perp a_i \quad \forall i = \overline{1, n}. \tag{2.32}$$

$\triangleright_1$ Докажите эквивалентность этих двух определений.

Основное свойство ортогональной проекции вектора x состоит в том, что она является наилучшим среднеквадратичным приближением к x среди всех элементов подпространства V:

$$\|x - y\|_2 = \min_{v \in V} \|x - v\|_2.$$

Этот факт, вообще говоря, следует доказать, но мы этого делать не будем, полагаясь на геометрическую интуицию.

Исходя из сказанного, решение задачи наименьших квадратов (2.31) можно найти из условий (2.32). Действительно, по условию вектор Ax должен быть проекцией вектора b на $V = \operatorname{span}\{a_1, \ldots, a_n\}$, то есть невязка $Ax - b$ должна быть ортогональна каждому базисному вектору подпространства V. Это условие записывается в виде

$$(Ax - b, a_j) = 0, \quad \text{или} \quad (Ax, a_j) = (b, a_j) \quad \forall j = \overline{1, n}.$$

Полученные условия называются *нормальными уравнениями*. Записывая их для всех j нетрудно заметить, что они эквивалентны СЛАУ

$$A^T A x = A^T b. \tag{2.33}$$

Таким образом, решение задачи наименьших квадратов (2.31) совпадает с решением СЛАУ (2.33). Матрица $A^T A$ является симметричной и положительно определенной, поэтому для решения нормальных уравнений лучше использовать метод квадратного корня (разложение Холецкого). Если исходная матрица A имеет ранг n, то матрица $A^T A$ невырождена, и, следовательно, решение задачи наименьших квадратов существует и единственно.

$\triangleright_2$ Подсчитайте количество операций, необходимое для нахождения x из уравнения (2.33).

Решение задачи наименьших квадратов путем сведения к нормальным уравнениям является самым экономичным, но, в то же время, и наименее вычислительно устойчивым способом. Более надежный метод основан на QR-разложении матрицы A.

2.5.3. Решение с помощью QR-разложения

Рассмотрим «нормальную» систему $A^TAx = A^Tb$ и сокращенное QR-разложение $A = \hat{Q}\hat{R}$:

$$A^TAx = A^Tb \quad \Leftrightarrow \quad \hat{R}^T\hat{Q}^T\hat{Q}\hat{R}x = \hat{R}^T\hat{Q}^Tb,$$

откуда

$$\hat{R}x = \hat{Q}^Tb. \tag{2.34}$$

Отсюда получаем следующий алгоритм решения задачи наименьших квадратов:

1) Построить сокращенное QR-разложение $A = \hat{Q}\hat{R}$.
2) Вычислить вектор $\hat{Q}^Tb$.
3) Решить СЛАУ (2.34) обратной подстановкой.

Как было сказано в п. 2.4.4, матрица Q как правило не хранится в явном виде, а определяется набором векторов нормали $\{w_k\}_{k=1}^n$ по формулам (2.29). В связи с этим обсудим схему вычисления вектора $b' = \hat{Q}^Tb$ в пункте 2. Матрица $\hat{Q}^T$ это первые n строк матрицы Q^T, поэтому вектор b' содержит первые n компонент вектора Q^Tb. Значит, сначала нужно найти

$$b^{(n)} = Q_n \dots Q_2Q_1b,$$

см. алгоритм в конце п. 2.4.4, после чего положить

$$b' = b^{(n)}_{1:n}.$$

2.6. Итерационные методы решения СЛАУ

2.6.1. Общая характеристика

Все методы решения СЛАУ, которые мы до сих пор рассматривали, являлись *прямыми*, или *точными*, т. е. позволяли (при отсутствии ошибок округления) найти точное решение системы $Ax = b$ за конечное число операций. Сейчас мы рассмотрим другой класс методов — *итерационные* методы, которые позволяют за конечное число операций найти решение лишь *приближенно*, но с произвольной наперед заданной точностью[1]. Это значит, что в общем случае для нахождения точного решения итерационному методу потребовалось бы *бесконечно много* операций. Однако это не проблема, так как даже точные методы в условиях машинной арифметики всегда дают решение с точностью до ошибок округления.

Если говорить абстрактно, то итерационный метод (процесс) решения СЛАУ может рассматриваться как метод построения последовательности векторов

$$x^0, x^1, \ldots, x^k, \ldots,$$

такой, что если существует ее предел x^*,

$$\|x^k - x^*\| \xrightarrow[k\to\infty]{} 0,$$

то $x^* = A^{-1}b$ — искомое решение.

2.6.2. Стационарные итерационные процессы

Принцип построения

Рассмотрим СЛАУ

$$Ax = b$$

с невырожденной матрицей A, а также эквивалентную ей систему

$$x = Bx + g, \tag{2.35}$$

[1] Естественно, чем выше требуемая точность, тем больше вычислительной работы придется проделать.

где B — матрица; g — вектор соответствующей размерности. Эквивалентность здесь означает, что эти системы имеют одинаковое решение

$$A^{-1}b = (I - B)^{-1}g.$$

Из (2.35) автоматически получается итерационный процесс

$$x^{k+1} = Bx^k + g, \quad k = 0, 1, 2, \ldots . \tag{2.36}$$

Нетрудно заметить, что если такой процесс сходится, т. е. если сходится последовательность (x^k), то предел этой последовательности x^* удовлетворяет условию $x^* = Bx^* + g$, откуда следует, что $x^* = A^{-1}b$ — искомое решение. Процессы типа (2.36) будем называть *стационарными итерационными процессами*. Слово «стационарный» здесь обозначает тот факт, что матрица B не зависит от k. Строго говоря, к этому определению стоило бы добавить «одношаговый», так как следующее приближение x^{k+1} зависит только от одного предыдущего x^k, но для краткости мы этого делать не будем.

Рассмотрим наиболее известные примеры таких процессов.

Классические стационарные итерационные методы

Метод Якоби. Рассмотрим i-е уравнение СЛАУ $Ax = b$:

$$a_{i1}x_1 + \ldots + a_{ii}x_i + \ldots + a_{in}x_n = b_i.$$

Выражая из него x_i, получим

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j \neq i} a_{ij}x_j\Big), \quad i = \overline{1, n}. \tag{2.37}$$

Для того чтобы записать это тождество в векторном виде, рассмотрим разбиение матрицы A на слагаемые согласно рис. 2.4:

$$A = L + D + R. \tag{2.38}$$

Тогда (2.37) примет вид

$$x = D^{-1}(b - (L + R)x) = B_J x + g_J,$$

где

$$B_J = -D^{-1}(L + R), \quad g_J = D^{-1}b. \tag{2.39}$$

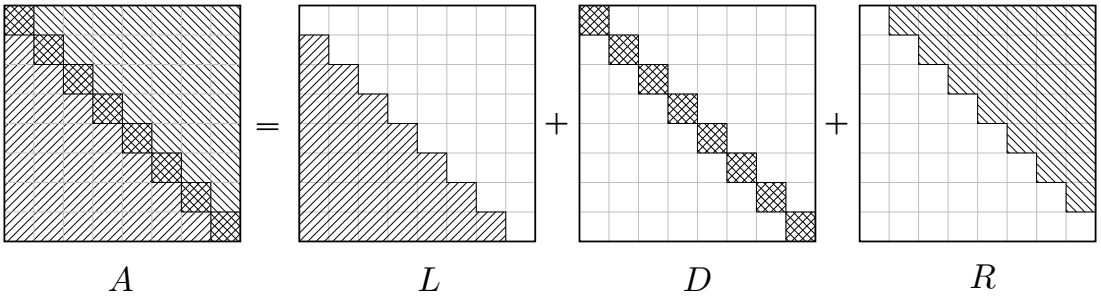


Рис. 2.4

Соответствующий системе (2.37) итерационный метод

$$x_i^{k+1} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j\neq i} a_{ij}x_j^k\Big), \quad i = \overline{1,n}, \quad k = 0,1,2,\ldots \tag{2.40а}$$

называется *методом Якоби*. Его векторная форма имеет вид

$$x^{k+1} = B_J x^k + g_J, \tag{2.40б}$$

где B_J и g_J определяются по формуле (2.39).

Заметим, что $L + R = A - D$, поэтому для матрицы B_J существует альтернативная форма записи

$$B_J = I - D^{-1}A.$$

Метод Гаусса – Зейделя. Рассмотрим i-й шаг k-й итерации метода Якоби:

$$x_i^{k+1} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j\neq i} a_{ij}x_j^k\Big).$$

К этому моменту нам уже известны компоненты вектора x^{k+1} с номерами от 1 до $i-1$. Эти компоненты *могут быть* более точны, чем соответствующие компоненты текущего приближения x^k, поэтому их можно использовать в сумме (2.40а):

$$x_i^{k+1} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\Big). \tag{2.41а}$$

Векторный вариант (2.41а) имеет вид

$$x^{k+1} = D^{-1}(b - Lx^{k+1} - Rx^k),$$

откуда

$$x^{k+1} = B_S x^k + g_S,$$
$$B_S = -(D+L)^{-1}R, \quad g_S = (D+L)^{-1}b. \tag{2.41б}$$

Формулы (2.41а), (2.41б) определяют *метод Гаусса – Зейделя*.

$\triangleright_1$ Постройте модификацию метода Гаусса – Зейделя для случая, когда компоненты вектора x^{k+1} обновляются в обратном порядке (такой метод называется *обратным методом Гаусса – Зейделя*).

Метод релаксации. *Метод релаксации* получается путем взвешенного осреднения текущего приближения и приближения, построенного по методу Гаусса – Зейделя:

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\Big), \tag{2.42а}$$

где ω — весовой коэффициент, обычно $\omega \in (0,2)$. Формула (2.42а) в векторной форме имеет вид

$$x^{k+1} = (1-\omega)x^k + \omega D^{-1}(b - Lx^{k+1} - Rx^k).$$

Умножая обе части на D и группируя слагаемые, получаем

$$x^{k+1} = B_\omega x^k + g_\omega,$$
$$B_\omega = (D+\omega L)^{-1}((1-\omega)D - \omega R), \quad g_\omega = (D+\omega L)^{-1}b. \tag{2.42б}$$

Замечание 2.3. При $\omega = 1$ метод релаксации очевидно превращается в метод Гаусса – Зейделя.

Внимание! *Для программной реализации классических итерационных методов используются исключительно их скалярные формы (2.40а), (2.41а), (2.42а). Соответствующие векторные формы записи (2.40б), (2.41б), (2.42б) используются для анализа сходимости методов (см. далее).*

2.6.3. Сходимость стационарных итерационных процессов

До сих пор мы строили итерационные процессы формально, ничего не говоря об их сходимости. Для их обоснования необходимо изучить

сходимость стационарных итерационных процессов

$$x^{k+1} = Bx^k + g.$$

Для начала вспомним некоторые определения из линейной алгебры.

Определение 2.7. Пусть A — квадратная матрица. Вектор $x \neq 0$, $x \in \mathbb{C}^n$, называется *собственным вектором* матрицы A, если существует $\lambda \in \mathbb{C}$ такое, что

$$Ax = \lambda x.$$

Число λ называется *собственным значением*, соответствующим x. Множество всех собственных значений A называется *спектром* и обозначается $\sigma(A)$.

Определение 2.8. *Спектральным радиусом* $\rho(A)$ называется величина наибольшего по модулю собственного значения матрицы A:

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

Основной результат о сходимости выглядит следующим образом.

Теорема 2.8 (критерий сходимости стационарных итерационных процессов)**.** *Итерационный процесс $x^{k+1} = Bx^k + g$ сходится при любом начальном приближении x^0 тогда и только тогда, когда все собственные значения матрицы B по модулю меньше единицы, т. е. если $\rho(B) < 1$.*

Доказательство этой теоремы основано на следующих двух леммах.

Лемма 2.1. *Стационарный итерационный процесс $x^{k+1} = Bx^k + g$ сходится при любом начальном приближении тогда и только тогда, когда последовательность матриц $I, B, B^2, \ldots, B^k, \ldots$ сходится к нулевой матрице, т. е. $\|B^k\| \to 0$ в некоторой подчиненной матричной норме.*

Доказательство. Рассмотрим погрешность на k-й итерации:

$$x^k - x^* = Bx^{k-1} + g - Bx^* - g = B(x^{k-1} - x^*) = \ldots = B^k(x^0 - x^*).$$

Отсюда видно, что если процесс сходится, то есть $\|x^* - x^k\| \to 0$ при любом $x^0 \in \mathbb{R}^n$, тогда $\|B^k x\| \to 0$ для всех $x \in \mathbb{R}^n$, т. е. $\|B^k\| \to 0$.

Пусть теперь $B^k \to 0$. Тогда

$$\|x^k - x^*\| = \|B^k(x^0 - x^*)\| \leqslant \|B^k\| \, \|x^0 - x^*\| \to 0,$$

при любом $x^0 \in \mathbb{R}^n$. □

Определение 2.9. Матрица A называется *диагонализируемой*, или *матрицей простой структуры*, если она представима в виде

$$A = X\Lambda X^{-1},$$

где матрица X невырождена, а

$$\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

Здесь λ_i являются, очевидно, собственными значениями A, так как преобразование подобия $X\Lambda X^{-1}$ сохраняет спектр.

Лемма 2.2. *Матричная последовательность*

$$I, B, B^2, \ldots, B^k, \ldots$$

сходится к нулевой матрице тогда и только тогда, когда $\rho(B) < 1$.

Доказательство. Мы докажем эту лемму лишь для частного случая, когда матрица B диагонализируема:

$$B = X\Lambda X^{-1}.$$

В этом случае имеем

$$B^2 = X\Lambda X^{-1} X\Lambda X^{-1} = X\Lambda^2 X^{-1}$$

и вообще

$$B^k = X\Lambda^k X^{-1},$$

откуда

$$B^k \to 0 \;\Leftrightarrow\; \Lambda^k \to 0 \;\Leftrightarrow\; |\lambda_i|^k \to 0 \;\Leftrightarrow\; \rho(B) < 1. \qquad \square$$

Собирая вместе две последние леммы, получаем доказательство теоремы 2.8, из которой теперь можно получить и достаточное условие сходимости.

Следствие 2.2 (достаточное условие сходимости)**.** *Если для некоторой подчиненной матричной нормы* $\|B\| < 1$*, то итерационный процесс* $x^{k+1} = Bx^k + g$ *сходится.*

Доказательство. Рассмотрим любое собственное значение λ матрицы B и соответствующий ему собственный вектор ξ с единичной нормой. Тогда в любой подчиненной матричной норме имеем

$$\|B\| \geqslant \|B\xi\| = \|\lambda\xi\| = |\lambda|\,\|\xi\| = |\lambda|,$$

откуда

$$\rho(B) \leqslant \|B\| < 1.$$

Значит, итерационный процесс сходится по теореме 2.8. □

Замечание 2.4. Из доказательства леммы 2.2 можно увидеть, что в случае диагонализируемой матрицы норма погрешности $\|x^k - x^*\|$ стремится к нулю как геометрическая прогрессия со знаменателем $\rho(B)$:

$$\|x^k - x^*\| = \|B^k(x^0 - x^*)\| = \|X\Lambda^k X^{-1}(x^0 - x^*)\| \leqslant C\rho(B)^k,$$

$C = æ(X)\|x^0 - x^*\|$. Это свойство будет справедливым и в общем случае, однако лишь начиная с некоторого достаточно большого номера итерации k.

Сходимость классических итерационных методов

Применим полученные выше общие результаты о сходимости к методу Якоби (2.40).

Лемма 2.3. *Рассмотрим СЛАУ $Ax = b$. Если матрица A имеет строгое диагональное преобладание, то метод Якоби для такой системы сходится при любом начальном приближении.*

Доказательство. Рассмотрим матрицу $B_J = I - D^{-1}A$:

$$B_J = -\begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & \dots & \frac{a_{2n}}{a_{22}} \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 0 & \dots & \frac{a_{3n}}{a_{33}} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \frac{a_{n3}}{a_{nn}} & \dots & 0 \end{bmatrix}.$$

Вычислим максимум-норму этой матрицы:

$$\|B_J\|_\infty = \max_i \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} = \max_i \frac{\sum_{j \neq i} |a_{ij}|}{|a_{ii}|}.$$

По условию матрица A имеет строгое диагональное преобладание, т. е.

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

Отсюда сразу следует, что

$$\|B_J\|_\infty < 1,$$

т. е. метод Якоби сходится по следствию 2.2. □

Замечание 2.5. Аналогичное утверждение можно доказать и для метода Гаусса – Зейделя.

Следующий важный результат мы приведем без доказательства.

Теорема 2.9. *Если матрица A является симметричной и положительно определенной, то метод релаксации* (2.42) *сходится при любом* $\omega \in (0, 2)$.

Следствие 2.3. *Если матрица A является симметричной и положительно определенной, то метод Гаусса – Зейделя* (2.41) *сходится.*

2.7. Чебышевское ускорение стационарных итерационных процессов

Вспомним некоторые предварительные факты из линейной алгебры. Если p — некоторый многочлен степени m общего вида $p(t) = c_m t^m + \\ + c_{m-1}t^{m-1} + \ldots + c_1 t + c_0$, то его значения определены не только для вещественных чисел, но и для квадратных матриц:

$$p(A) = c_m A^m + c_{m-1}A^{m-1} + \ldots + c_1 A + c_0 I.$$

При этом нетрудно доказать следующее важное свойство:

$$\sigma(p(A)) = \{p(\lambda_1), \ldots, p(\lambda_n)\}, \quad \lambda_i \in \sigma(A),$$

то есть собственные значения матрицы $p(A)$ равны значениям многочлена p на спектре матрицы A.

$\triangleright_1$ Докажите это утверждение.

2.7.1. Общая идея метода

Рассмотрим начальное приближение x^0 и выполним m итераций некоторого стационарного метода:

$$x^1 = Bx^0 + g, \ldots, x^m = Bx^{m-1} + g.$$

Будем предполагать что этот процесс сходится и спектр B вещественен, то есть

$$\sigma(B) \subset [-\rho, \rho] \subset (-1, 1).$$

Составим линейную комбинацию из полученных приближений:

$$y^m = \sum_{k=0}^{m} c_{m,k} x^k,$$

и попробуем выбрать коэффициенты $c_{m,k}$ так, чтобы новое приближение y^m было как можно более точным. Для этого рассмотрим погрешность

$$y^m - x^* = \sum_{k=0}^{m} c_{m,k} x^k - x^* = \sum_{k=0}^{m} c_{m,k}(x^k - x^*) - \left(1 - \sum_{k=0}^{m} c_{m,k}\right) x^*. \quad (2.43)$$

Первым условием, которое мы наложим на коэффициенты $c_{m,k}$, будет

$$\sum_{k=0}^{m} c_{m,k} = 1. \tag{2.44}$$

Во-первых, по определению y^m это условие гарантирует, что если все x^k равны x^*, то $y^m = x^*$. Во-вторых, из (2.43) тогда будем иметь

$$\|y^m - x^*\| = \left\| \sum_{k=0}^{m} c_{m,k}(x^k - x^*) \right\| = \left\| \sum_{k=0}^{m} c_{m,k} B^k (x^0 - x^*) \right\| = \\ = \|p_m(B)(x^0 - x^*)\| \leqslant \|p_m(B)\| \, \|x^0 - x^*\|, \tag{2.45}$$

где

$$p_m(t) = \sum_{k=0}^{m} c_{m,k} t^k.$$

Как и ранее, предположим, что матрица B диагонализируема, $B = X\Lambda X^{-1}$, тогда

$$\|p_m(B)\| \leqslant \|X\| \, \|X^{-1}\| \, \|p_m(\Lambda)\| = æ(X) \max_{\lambda \in \sigma(B)} |p_m(\lambda)|.$$

Эта важная оценка наряду с (2.45) говорит о том, что если мы заинтересованы в наименьшей норме погрешности $\|y^m - x^*\|$, то коэффициенты $c_{m,k}$ должны быть такими, *чтобы соответствующий многочлен p_m принимал как можно меньшие по модулю значения на спектре матрицы B*. То есть, идеальным вариантом был бы характеристический многочлен

$$p_m(t) = (t - \lambda_1)(t - \lambda_2) \ldots (t - \lambda_n),$$

но он нам, во-первых, неизвестен, а во-вторых степень его слишком велика, чтобы использовать на практике.

Вместо этого мы вспомним, что $\sigma(B) \subset [-\rho, \rho]$, и попробуем найти «универсальный» многочлен p_m, который бы на всем отрезке $[-\rho, \rho]$ как можно меньше отклонялся от нуля. При этом в силу (2.44) этот многочлен должен удовлетворять условию $p_m(1) = 1$. Говоря более конкретно, необходимо

найти многочлен p_m степени m такой, что $p_m(1) = 1$, а величина

$$\max_{t \in [-\rho, \rho]} |p_m(t)|$$

принимает как можно меньшее значение.

Сразу дадим готовое решение этой задачи:

$$p_m(t) = \frac{T_m(t/\rho)}{T_m(1/\rho)}, \tag{2.46}$$

где T_m — знаменитые *многочлены Чебышева*, о которых стоит рассказать отдельно в небольшом отступлении.

Многочлены Чебышева

Рассмотрим на отрезке $[-1, 1]$ семейство функций $\{T_m\}$, определяемое формулой

$$T_m(t) = \cos(m \arccos t), \quad m = 0, 1, 2, \ldots . \tag{2.47}$$

Как это ни кажется странным на первый взгляд, $T_m(t)$ при $t \in [-1, 1]$ является алгебраическим многочленом степени m:

$$T_0(t) = 1, \quad T_1(t) = t,$$

а для произвольного m справедливо рекуррентное соотношение

$$T_m(t) = 2t\, T_{m-1}(t) - T_{m-2}(t), \tag{2.48}$$

которое позволяет вычислять $T_m(t)$ при любых $t \in \mathbb{R}$.

$\triangleright_2$ Вычислите $T_m(t)$ для $m = 2, 3, 4, 5$.

$\triangleright_3$ Из (2.47) вывести рекуррентное соотношение (2.48), используя тригонометрическую формулу $\cos\alpha \cos\beta = \frac{1}{2}(\cos(\alpha + \beta) + \cos(\alpha - \beta))$.

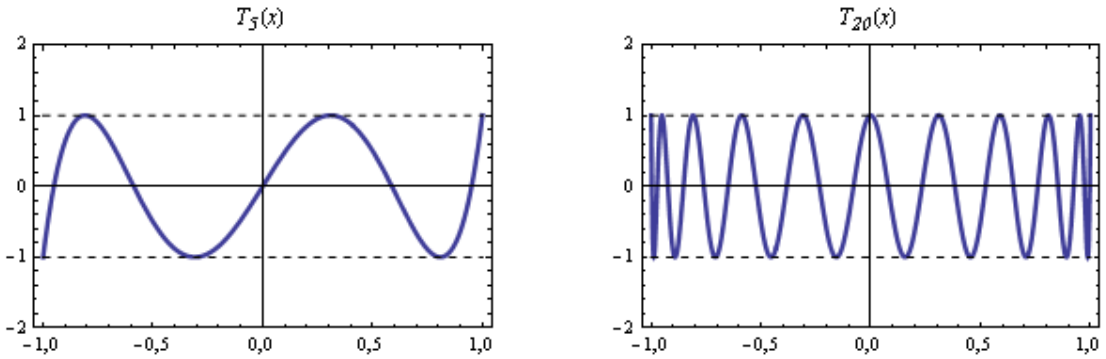


Рис. 2.1. Многочлены Чебышева T_5 и T_{20}

Из рекуррентного соотношения видно, что старший коэффициент многочлена T_m равен 2^{m-1} (при $m > 0$). Основное же свойство многочленов Чебышева состоит в том, что их графики «равномерно колеблются» между -1 и 1 (рис. 2.1). Говоря точнее, многочлен T_m имеет $m+1$ локальных экстремумов, которые поочередно равны ± 1.

Следствием этого свойства является то, что *многочлен Чебышева T_m имеет наименьшее отклонение от нуля на отрезке $[-1, 1]$ среди всех многочленов степени m со старшим коэффициентом 2^{m-1}*. Тогда многочлен $T_m(t/\rho)$ будет наименее отклоняться от нуля на $[-\rho, \rho]$ среди всех многочленов, имеющих такой же как у него старший коэффициент. Добавляя нормировочный множитель, в итоге для решения задачи поиска оптимального многочлена p_m получаем формулу (2.46) (см. также рис. 2.2).

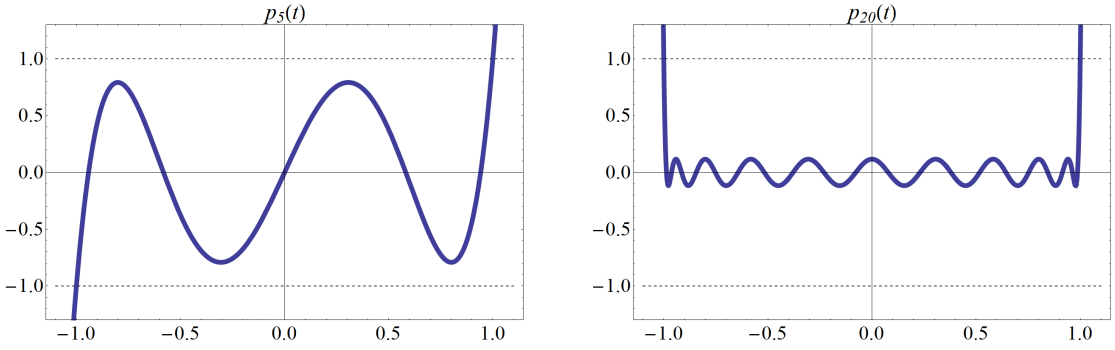


Рис. 2.2. Графики многочленов p_5 и p_{20} для $\rho = 0.99$.

2.7.2. Вывод расчетных формул

Подведем локальный итог вышесказанному: на основе сходящегося стационарного процесса $x^{k+1} = Bx^k + g$ мы строим приближения вида $y^m = \sum_{k=0}^{m} c_{m,k} x^k$, где $c_{m,k}$ определены согласно (2.46) как коэффициенты многочлена

$$p_m(t) = \sum_{k=0}^{m} c_{m,k} t^k = \mu_m T_m(t/\rho),$$

$$\mu_m = \frac{1}{T_m(1/\rho)}.$$

Теоретически этой информации достаточно, чтобы для каждого фиксированного m вычислить коэффициенты $c_{m,k}$ и соответствующие значения

y^m. Однако существует гораздо более элегантный и эффективный способ последовательного вычисления y^m, основанный на рекуррентном соотношении (2.48).

Для начала получим на основе этой формулы аналогичные соотношения для многочленов p_m:

$$p_m(t) = \mu_m T_m(t/\rho) = \frac{2\mu_m}{\rho}\, t\, T_{m-1}(t/\rho) - \mu_m T_{m-2}(t/\rho) = \\ = \frac{2\mu_m}{\rho\mu_{m-1}}\, t\, p_m(t) - \frac{\mu_m}{\mu_{m-2}} p_{m-2}(t). \qquad (2.49)$$

Также нам понадобятся рекуррентные соотношения между коэффициентами μ_m:

$$\mu_m^{-1} = T_m(1/\rho) = (2/\rho) T_{m-1}(1/\rho) - T_{m-2}(1/\rho),$$

или

$$\mu_m^{-1} = 2\mu_{m-1}^{-1}/\rho - \mu_{m-2}^{-1}. \qquad (2.50)$$

Согласно (2.45) имеем

$$y^m - x^* = p_m(B)(x^0 - x^*),$$

откуда, обозначив

$$\alpha_m = \frac{2\mu_m}{\rho\mu_{m-1}}, \quad \beta_m = -\frac{\mu_m}{\mu_{m-2}}, \qquad (2.51)$$

получим

$$y^m - x^* = p_m(B)(x^0 - x^*) = \\ = \big(\alpha_m B p_{m-1}(B) + \beta_m p_{m-2}(B)\big)(x^0 - x^*) = \\ = \alpha_m B\,(y^{m-1} - x^*) + \beta_m (y^{m-2} - x^*),$$

или

$$y^m - \alpha_m B y^{m-1} - \beta_m y^{m-2} = (I - \alpha_m B - \beta_m I)\, x^*. \qquad (2.52)$$

Умножая (2.50) на μ_m, получим $\alpha_m + \beta_m = 1$. С учетом тождества $x^* = Bx^* + g$ тогда будем иметь

$$(I - \alpha_m B - \beta_m I)\, x^* = \alpha_m (I - B) x^* = \alpha_m g.$$

Таким образом (2.52) окончательно принимает чудесный вид

$$y^m = \alpha_m(By^{m-1} + g) + \beta_m y^{m-2}, \quad m = 2, 3, \ldots . \tag{2.53}$$

Это рекуррентное соотношение является основным при программной реализации процесса чебышевского ускорения наряду со стартовым тождеством

$$y^1 = By^0 + g$$

и формулами (2.50), (2.51) для расчета коэффициентов. Если говорить о терминологии, то полученный итерационный процесс можно назвать *двухшаговым нестационарным*.

$\triangleright_4$ Коэффициенты α_m и β_m можно вычислять по прямым рекуррентным соотношениям без использования промежуточных коэффициентов μ_m. Постройте эти соотношения.

$\triangleright_5$ Запишите соответствующий вычислительный алгоритм для чебышевских итераций.

2.7.3. Скорость сходимости

[множитель сходимости равен μ_m]

Для достижения нужного эффекта необходимо иметь хорошую оценку параметра ρ — спектрального радиуса матрицы B.

2.8. Методы подпространства Крылова

2.8.1. Обобщенный метод минимальных невязок (GMRES)

Определение 2.10. Рассмотрим матрицу A и вектор b. *Подпространством Крылова* размерности k называется линейная оболочка векторов b, Ab, A^2b, ..., $A^{k-1}b$:

$$\mathcal{K}_k(A, b) = \operatorname{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}.$$

Подпространства Крылова являются основой многих современных итерационных методов решения СЛАУ и проблемы собственных значений. Сейчас мы познакомимся с одним из наиболее эффективных итерационных методов решения систем с несимметричной матрицей — обобщенным методом минимальных невязок (generalized minimal residual, GMRES).

Суть обобщенного метода минимальных невязок для решения системы $Ax = b$ состоит в следующем: на k-й итерации метода в подпространстве $\mathcal{K}_k(A, b)$ ищется приближенное решение x^k, имеющее минимальную норму невязки $\|Ax^k - b\|_2$. Чтобы сделать формулы более компактными в дальнейшем будем обозначать $x^k = x$.

Так как $x \in \mathcal{K}_k(A, b)$, то его можно представить в виде

$$x = c_1 b + c_2 Ab + \ldots + c_k A^{k-1} b = K_k c,$$

где

$$K_k = \left[\, b \,|\, Ab \,|\, A^2b \,|\, \ldots \,|\, A^{k-1}b \,\right],$$

а $c = (c_1, \ldots, c_k)^T$ — вектор неизвестных коэффициентов, который по построению метода находится путем решения задачи наименьших квадратов

$$\|AK_k c - b\|_2 \to \min. \tag{2.54}$$

На этом можно уже остановиться, так как задачу наименьших квадратов мы решать уже умеем. Но результат получится не очень хорошим, главным образом из-за плохой обусловленности матрицы K_k. Дальнейшие наши усилия будут направлены на получения качественного и эффективного способа решения задачи (2.54).

Основная идея заключается в том, чтобы вместо столбцов матрицы K_k для разложения искомого вектора x использовать *ортонормированный базис* пространства Крылова $\mathcal{K}_k(A, b)$. Векторы этого базиса будем обозначать q_j, а матрицу, составленную из этих столбцов обозначим Q_k. Тогда вместо $x^k = K_k c$ будем иметь $x^k = Q_k y$, где y — вектор коэффициентов разложения по базису $\{q_j\}$, и задачу минимизации

$$\|AQ_k y - b\|_2 \to \min .$$

Для построения нужного нам базиса необходимо сначала изучить несколько новых (и вспомнить несколько старых) понятий.

Алгоритм Арнольди

Определение 2.11. Говорят, что квадратная матрица H имеет *форму Хессенберга* (является хессенберговой), если

$$h_{ij} = 0 \quad \forall\, i \geqslant j + 2.$$

Другими словами, форма Хессенберга отличается от верхнетреугольной матрицы лишь ненулевыми элементами под главной диагональю:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Напомним, что преобразованием подобия называется преобразование $A \mapsto X^{-1}AX$, где X — произвольная невырожденная матрица. Такие преобразования не изменяют спектр матрицы, так как если $X^{-1}AX\xi = \lambda\xi$, то $A\eta = \lambda\eta$, где $\eta = X\xi$

Важный факт: любую квадратную вещественную матрицу A можно ортогональными преобразованиями подобия привести к форме Хессенберга, то есть существует такая матрица Q, что $Q^{-1} = Q^T$ и

$$Q^T AQ = H, \quad \text{или} \quad AQ = QH.$$

Оказывается, что первые k столбцов матрицы Q образуют базис (ортонормированный, естественно) в подпространстве Крылова $\mathcal{K}_k(A, q_1)$.

Лемма 2.4. *Пусть* $h_{j+1,j} \neq 0$ *для* $j = 1, \ldots, k-1$. *Тогда*

$$\operatorname{span}\{q_1, \ldots, q_k\} = \mathcal{K}_k(A, q_1).$$

Доказательство. При $k = 1$ утверждение леммы очевидно. Предположим теперь, что $\operatorname{span}\{q_1, \ldots, q_{k-1}\} = \mathcal{K}_{k-1}(A, q_1)$. В тождестве $AQ = QH$ рассмотрим $(k-1)$-й столбец матриц AQ и QH с учетом, что H хессенбергова:

$$Aq_{k-1} = h_{1,k-1}q_1 + h_{2,k-1}q_2 + \ldots + h_{k,k-1}q_k, \tag{2.55}$$

откуда

$$q_k = \frac{1}{h_{k,k-1}} \left(Aq_{k-1} - \sum_{j=1}^{k-1} h_{j,k-1}q_j \right). \tag{2.56}$$

Применяя рекурсивно эту формулу к q_{k-1} и так далее, легко убедиться, что вектор q_k представим в виде

$$q_k = \gamma_k A^{k-1} q_1 + \delta_k,$$

где $\gamma_k \neq 0$, а $\delta_k \in \operatorname{span}\{q_1, \ldots, q_{k-1}\} = \mathcal{K}_{k-1}(A, q_1)$. Следовательно,

$$\operatorname{span}\{q_1, q_2, \ldots, q_{k-1}, q_k\} = \operatorname{span}\{q_1, q_2, \ldots, q_{k-1}, A^{k-1}q_1\} = \mathcal{K}_k(A, q_1).$$

□

Итак, ортогональный базис подпространства Крылова может быть получен путем нахождения первых k столбцов матрицы Q, которая преобразует матрицу A к форме Хессенберга. Доказательство леммы 2.4 дает и способ нахождения искомых векторов q_j, называемый *алгоритмом Арнольди*.

Опишем один шаг алгоритма. Предположим, что уже найдены векторы $q_1, \ldots, q_j$. Перепишем (2.55), заменив k на $j+1$:

$$Aq_j = h_{1j}q_1 + h_{2j}q_2 + \ldots + h_{j+1,j}q_{j+1}.$$

По построению векторы q_i ортонормированы, поэтому все коэффициенты h_{ij} кроме последнего, $h_{j+1,j}$, можно найти, умножив последнее тождество скалярно на q_i:

$$h_{ij} = (Aq_j, q_i), \quad i = 1, \ldots, j. \tag{2.57}$$

Искомый вектор q_{j+1} находим аналогично (2.56):

$$q_{j+1} = \frac{1}{h_{j+1,j}} \left(Aq_j - \sum_{i=1}^{j} h_{ij} q_i \right), \tag{2.58}$$

где в силу того, что $\|q_{j+1}\|_2 = 1$,

$$h_{j+1,j} = \left\| Aq_j - \sum_{i=1}^{j} h_{ij} q_i \right\|_2. \tag{2.59}$$

Заметим, что коэффициенты h_{ij} представляют собой элементы j-го столбца матрицы H. Таким образом алгоритм Арнольди позволяет последовательно находить столбцы матриц Q и H в разложении $Q^T AQ = H$.

Алгоритм Арнольди

1) *Задать вектор* q_1 такой, что $\|q_1\|_2 = 1$
2) **for** $j = 1$ **to** $k - 1$
3) · $z = Aq_j$
4) · **for** $i = 1$ **to** j
5) · · $h_{ij} = (z, q_i)$
6) · · $z = z - h_{ij} q_i$
7) · $h_{j+1,j} = \|z\|_2$
8) · **if** $h_{j+1,j} = 0$ **then** stop
9) · $q_{j+1} = z / h_{j+1,j}$

Необходимо сделать несколько важных комментариев касательно данного алгоритма.

- С формальной точки зрения строка 5 не точно соответствует формуле (2.57), так как при $i > 1$ переменная z уже не будет равна Aq_j. Тем не менее, алгоритм будет давать нужный результат в силу ортогональности векторов $q_1, \ldots, q_j$.
- Обычно необходимое количество векторов-столбцов матрицы Q, которое мы обозначили k, заранее не известно. Условие выхода из внешнего цикла зависит от приложения. В обобщенном методе минимальных невязок, например, это будет условие малости невязки $Ax^k - b$ на текущем приближении.

• В общем случае вектор q_1 может быть произвольным вектором единичной длины. В обобщенном методе минимальных невязок же по построению $q_1 = b/\|b\|_2$.

• В строке 8 осуществляется проверка на «вырожденный» случай, когда невозможно вычислить q_{j+1} и алгоритм останавливается досрочно. Тогда будет справедливо тождество

$$Aq_j = h_{1j}q_1 + \ldots + h_{jj}q_j,$$

или $Aq_j \in \mathcal{K}_j(A, q_1)$, откуда следует, что

$$Ax \in \mathcal{K}_j(A, q_1) \quad \forall\, x \in \mathcal{K}_j(A, q_1),$$

то есть подпространство Крылова размерности j является *инвариантным* относительно матрицы (отображения) A. Как следует из приведенной ниже леммы это означает, что метод GMRES в этом случае даст точное решение.

Лемма 2.5. *Пусть подпространство Крылова $\mathcal{K}_j = \mathcal{K}_j(A, b)$ инвариантно относительно A. Тогда если*

$$x^j = \operatorname*{argmin}_{x \in \mathcal{K}_j} \|Ax - b\|_2, \tag{2.60}$$

то $Ax^j = b$.

Доказательство. В силу инвариантности имеем $Ax^j \in \mathcal{K}_j$, при этом $b \in \mathcal{K}_j$ по построению. Значит, задача наименьших квадратов (2.60) представляет собой поиск проекции вектора b на подпространство, которому он и так принадлежит. Поэтому эта проекция, равная Ax^j, совпадает с b. □

Таким образом, «вырожденный» случай в алгоритме Арнольди является счастливым для обобщенного метода минимальных невязок.

Вычислительная схема метода GMRES

Теперь нужно собрать вместе все детали. Пусть известна матрица $Q_{k-1} = [q_1|\ldots|q_k]$, столбцы которой образуют ортонормированный базис в подпространстве $\mathcal{K}_k(A, b)$, при этом $q_1 = b/\|b\|_2$. Тогда для вычисления приближения x^k необходимо:

1) *Выполнить один шаг алгоритма Арнольди для вычисления коэффициентов h_{ik} (строки 3-9 при $j = k$). Эти коэффициенты нужны для решения задачи наименьших квадратов в следующем пункте.*
2) *Найти $y^k = \operatorname{argmin}_y \|AQ_k y - b\|_2$ (ниже будет сказано о том, как сделать это эффективно).*
3) *Вычислить $x^k = Q_k y^k$.*
4) *Если норма невязки $\|Ax^k - b\|_2$ превышает требуемое значение, положить $k = k + 1$ и перейти к шагу 1.*

Задача наименьших квадратов в пункте 2 имеет размерность $n \times k$, однако ее можно свести к эквивалентной задаче, размерность которой будет всего $(k + 1) \times k$. Покажем как это делается.

В силу ортогональности матрицы Q имеем

$$\|AQ_k y - b\|_2 = \|Q^T AQ_k y - Q^T b\|_2.$$

Так как q_1 представляет собой нормированный вектор b, и все строки Q^T взаимно ортогональны, то $Q^T b = d$,

$$d = (\|b\|_2, 0, \ldots, 0)^T. \tag{2.61}$$

Далее, по построению имеем $Q^T AQ = H$. Напомним, что ненулевые элементы столбцов матрицы H находятся в процессе выполнения алгоритма Арнольди. Нетрудно заметить, что матрица $Q^T AQ_k$ состоит из первых k столбцов матрицы H, причем строки с номерами от $k + 2$ до n будут нулевыми, так как H хессенбергова. А так как в вектор d только первая компонента отлична от нуля, получаем

$$\|Q^T AQ_k y - Q^T b\|_2 = \|Q^T AQ_k y - d\|_2 = \|\tilde{H}_k y - d_{k+1}\|_2,$$

где $\tilde{H}_k$ — матрица, составленная из первых k столбцов и $(k + 1)$ строк матрицы H, а $d_{k+1} = (\|b\|_2, 0, \ldots, 0)^T \in \mathbb{R}^{k+1}$. Таким образом, на шаге 2) алгоритма GMRES решается задача наименьших квадратов с хессенберговой матрицей:

$$y^k = \operatorname*{argmin}_{y \in \mathbb{R}^k} \|\tilde{H}_k y - d_{k+1}\|_2. \tag{2.62}$$

Для решения этой задачи нужно построить QR-разложение матрицы $\tilde{H}$, что осуществляется за $O(k^2)$ операций с помощью преобразований вращения (см. далее). Кроме этого, если воспользоваться имеющимся с

предыдущей итерации QR-разложением матрицы $\tilde{H}_{k-1}$, то объем работы можно уменьшить на порядок, до $O(k)$ операций.

$\triangleright_1$ Подумайте как это сделать.

В заключение обсуждения этого метода заметим, что по построению при отсутствии ошибок округления метод GMRES сходится не более чем за n итераций (n — размер матрицы A).

2.8.2. Оценка погрешности метода GMRES

...

2.8.3. Метод полной ортогонализации (FOM)

Метод полной ортогонализации (full orthogonalization method, FOM) очень похож на обобщенный метод минимальных невязок. Приближенное решение x^k так же ищется в подпространстве Крылова и раскладывается по ортонормированному базису $\{q_j\}$, $x^k = Q_k y^k$, но вместо условия минимизации невязки в среднеквадратичном смысле накладывается условие ортогональности

$$(Ax^k - b) \perp \mathcal{K}_k(A, b). \tag{2.63}$$

То есть вектор невязки должен быть ортогонален всему подпространству Крылова, что эквивалентно ортогональности всем векторам из базиса этого подпространства. В качестве такого базиса, конечно, следует брать ортонормированный базис $\{q_j\}$, построенный методом Арнольди. В этом случае условие (2.63) примет вид

$$(AQ_k y^k - b, q_j) = 0, \quad j = \overline{1, k},$$

или

$$Q_k^T A Q_k y^k = Q_k^T b.$$

По построению имеем

$$Q_k^T A Q_k = H_k,$$

где $H_k = [H]_k$ — первые k строк и столбцов матрицы H.

Таким образом, алгоритм метода FOM отличается от метода GMRES только пунктом 2: вместо задачи наименьших квадратов здесь нужно решать СЛАУ

$$H_k y^k = d_k, \tag{2.64}$$

где H_k хессенбергова матрица размера k, а вектор $d_k \in \mathbb{R}^k$ определен аналогично (2.61).

$\triangleright_2$ Запишите алгоритм решения СЛАУ с матрицей в форме Хессенберга а) методом Гаусса, б) методом вращений.

Глава 3

Методы решения проблемы собственных значений

3.1. Проблема собственных значений: общая характеристика

3.1.1. Сведения из линейной алгебры

Еще раз вспомним необходимые сведения из курса линейной алгебры.

Пусть A — вещественная квадратная матрица. Ненулевой вектор ξ называется *собственным вектором* матрицы A, если существует $\lambda \in \mathbb{C}$ такое, что

$$A\xi = \lambda\xi.$$

Число λ называется *собственным значением*, соответствующим ξ. Множество всех собственных значений A называется *спектром* и обозначается $\sigma(A)$.

Очевидно, что комплексным собственным значениям соответствуют комплексные собственные векторы.

Замечание 3.1. Собственный вектор определен с точностью до постоянного множителя:

$$A\xi = \lambda\xi \quad \Rightarrow \quad A(\alpha\xi) = \lambda(\alpha\xi).$$

Поэтому все собственные векторы, соответствующие собственному значению λ, образуют так называемое *собственное подпространство*. Размерность собственного подпространства (число линейно независимых собственных векторов с собственным значением λ) называют *геометрической кратностью* собственного значения.

Как известно, все собственные значения являются корнями *характеристического многочлена*

$$P(\lambda) = \det(A - \lambda I).$$

Если λ — корень многочлена P кратности k, то говорят, что *алгебраическая кратность* λ равна k.

Известен следующий факт: *для любой степени $n \geqslant 5$ существует многочлен P_n с рациональнными коэффициентами, вещественный корень которого не может быть представлен в виде выражения, составленного из рациональных чисел и операций сложения, вычитания, умножения, деления и корня целой степени*. Таким образом, проблема собственных значений имеет принципиальное отличие от задачи решения СЛАУ. Решение СЛАУ с рациональными коэффициентами

всегда является вектором рациональных чисел и может быть найдено точно при отсутствии ошибок округления, например, тем же методом Гаусса. Собственные же значения матрицы в общем случае не могут быть найдены точно, поэтому *все методы вычисления собственных значений являются итерационными*.

Определение 3.1. Если квадратная матрица размерности n имеет n линейно независимых собственных векторов, то она называется *диагонализируемой*, а соответствующий ей линейный оператор — *оператором простой структуры*.

Эквивалентным определением можно считать следующее: матрица диагонализируема если и только если она может быть приведена к диагональному виду преобразованием подобия:

$$X^{-1}AX = \Lambda, \qquad \det X \neq 0, \quad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

Нетрудно убедиться, что столбцы матрицы X являются собственными векторами матрицы A.

Задачи на нахождение собственных значений условно делятся на два класса. Если нужно найти одно или несколько собственных значений и соответствующие им собственные векторы, то проблема собственных значений называется *частичной*. Если же необходимо найти все собственные значения и векторы, проблема называется *полной*.

3.2. Степенной метод

Степенной метод позволяет найти максимальное по модулю собственное значение и соответствующий ему собственный вектор вещественной диагонализируемой матрицы. Он использовался, например, в алгоритме PageRank при расчете релевантности веб-страницы поисковому запросу.

Пусть A — диагонализируемая матрица, $\lambda_1, \ldots, \lambda_n$ — упорядоченные по убыванию модуля собственные значения, $\xi_1, \ldots, \xi_n$ — соответствующий базис из собственных векторов. В дальнейшем будем предполагать, что евклидова норма всех ξ_i равна единице. Рассмотрим несколько возможных случаев.

3.2.1. Случай 1

Пусть $\lambda_1 \in \mathbb{R}$, $|\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$.

Разложим по базису $\{\xi_i\}$ произвольный вектор $y^0 \in \mathbb{R}^n$: $y^0 = \sum_1^n \alpha_i \xi_i$. Предположим, что $\alpha_1 \neq 0$ и рассмотрим последовательность (y^k) :

$$y^{k+1} = Ay^k.$$

Тогда

$$y^k = A^k y^0 = \sum_{i=1}^{n} \alpha_i \lambda_i^k \xi_i = \lambda_1^k \left(\alpha_1 \xi_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \xi_i \right). \qquad (3.1)$$

Значит, при $k \to \infty$ имеем

$$y^k = \lambda_1^k (\alpha_1 \xi_1 + O(\lambda_2^k / \lambda_1^k)),$$

т. е. вектор y^k все сильнее приближается к собственному подпространству, соответствующему ξ_1. Для практического применения необходимо на каждом шаге нормировать y^k. Как правило, y^k делят на его евклидову норму, то есть фактически вычисляется последовательность

$$u^k = y^k / \|y^k\|_2 = A^k y^0 / \|A^k y^0\|_2.$$

Такой подход облегчает вычисление собственного значения λ_1 с помощью *отношения Релея*, для знакомства с которым необходимо сделать небольшое отступление.

Пусть y — некоторое приближение к собственному вектору матрицы A. Тогда, скорее всего, не существует такого числа λ, для которого $Ay = \lambda y$. Но можно определить «почти собственное число» как скаляр λ, для которого норма невязки $Ay - \lambda y$ минимальна:

$$r(y) = \operatorname*{argmin}_{\lambda \in \mathbb{C}} \|\lambda y - Ay\|_2.$$

Это обычная одномерная задача наименьших квадратов, решая которую приходим с следующему определению.

Определение 3.2. Для любого вектора $y \neq 0$ отношением Релея называется число

$$r(\lambda) = \frac{(y, Ay)}{(y, y)}. \qquad (3.2)$$

$\triangleright_1$ Выведите эту формулу.

Исходя из вышесказанного легко увидеть, что если y является собственным вектором, то $r(y)$ будет в точности равно его собственному значению.

Можно показать, что если $\|y - \xi\| \leqslant \varepsilon$, то $|r(y) - \lambda| = O(\varepsilon)$, а в случае $A = A^T$ будем иметь квадратичную сходимость: $|r(y) - \lambda| = O(\varepsilon^2)$.

$\triangleright_2$ Докажите эти утверждения.

Таким образом, получаем следующий **базовый алгоритм степенного метода**:

$u = y^0, \quad \|y^0\|_2 = 1, \quad \lambda = 0$
while $\|Au - \lambda u\| > \varepsilon$ **do**
    $y \leftarrow Au$
    $u \leftarrow y/\|y\|_2$
    $\lambda \leftarrow (u, Au)$
end while

На выходе будем иметь $u \approx \xi_1$, $\lambda \approx \lambda_1$. Здесь ε — требуемая точность.

Из (3.1) видно, что степенной метод сходится со скоростью геометрической прогрессии со знаменателем $|\lambda_2/\lambda_1|$.

3.2.2. Случай 2

Пусть $\lambda_1 = \lambda_2 = \ldots = \lambda_m \in \mathbb{R}$, $|\lambda_1| > |\lambda_{m+1}| \geqslant \ldots \geqslant |\lambda_n|$.

Так как матрица A по условию диагонализируема, геометрическая кратность λ_1 равна m, т. е. собственные векторы $\xi_1, \ldots, \xi_m$ линейно независимы и образуют собственное подпространство X_m размерности m. Тогда формула (3.1) примет вид

$$y^k = \lambda_1^k \left(\sum_{i=1}^{m} \alpha_i \xi_i + \sum_{i=m+1}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \xi_i \right) \approx \lambda_1^k \sum_{i=1}^{m} \alpha_i \xi_i.$$

В этом случае алгоритм остается без изменений. Последовательность (u^k) сходится к какому-то вектору $u \in X_m$ — он ничем не хуже других собственных векторов $\xi_1, \ldots, \xi_m$.

3.2.3. Случай 3

Пусть $\lambda_1 = -\lambda_2 \in \mathbb{R}$, $|\lambda_1| > |\lambda_3| \geqslant \ldots \geqslant |\lambda_n|$. Не нарушая общности можно считать, что $\lambda_1 > 0$.

В этом случае (3.1) превращается в

$$y^k = \sum_{i=1}^{n} \alpha_i \lambda_i^k \xi_i = \alpha_1 \lambda_1^k \xi_1 + \alpha_2 (-\lambda_1)^k \xi_2 + \sum_{i=3}^{n} \alpha_i \lambda_i^k \xi_i =$$
$$= \lambda_1^k \left(\alpha_1 \xi_1 + (-1)^k \alpha_2 \xi_2 + \sum_{i=3}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \xi_i \right),$$

откуда получим

$$\begin{aligned} y^{2k} &\approx \lambda_1^{2k} (\alpha_1 \xi_1 + \alpha_2 \xi_2), \\ y^{2k+1} &\approx \lambda_1^{2k+1} (\alpha_1 \xi_1 - \alpha_2 \xi_2), \end{aligned} \tag{3.3}$$

т. е. последовательность (y^k), не сходится, а распадается на две сходящиеся подпоследовательности (y^{2k}) и (y^{2k+1}). Эти последовательности сходятся к двум различным векторам из $\operatorname{span}(\xi_1, \xi_2)$, причем в отличие от предыдущего случая, ни один из них не является, вообще говоря, собственным.

Но выход есть. Рассмотрим подпоследовательность четных элементов (u^{2k}),

$$u^{2k} = y^{2k} / \|y^{2k}\|_2, \quad k = 1, 2, \ldots.$$

После достаточно большого числа итераций будем иметь

$$u^{2k} \approx \beta_1 \xi_1 + \beta_2 \xi_2, \quad \|u^{2k}\|_2 = 1.$$

Тогда

$$A^2 u^{2k} \approx \lambda_1^2 \beta_1 \xi_1 + \lambda_1^2 \beta_2 \xi_2 = \lambda_1^2 u^{2k},$$

то есть

$$(u^{2k}, A^2 u^{2k}) \to \lambda_1^2. \tag{3.4}$$

Таким образом находим λ_1.

Теперь рассмотрим, как можно найти соответствующий собственный вектор. Имеем

$$\begin{aligned} u^{2k} &\approx \beta_1 \xi_1 + \beta_2 \xi_2, \\ A u^{2k} &\approx \lambda_1 \beta_1 \xi_1 - \lambda_1 \beta_2 \xi_2, \\ A^2 u^{2k} &\approx \lambda_1^2 \beta_1 \xi_1 + \lambda_1^2 \beta_2 \xi_2 \end{aligned}$$

Так как приближенное значение λ_1 уже известно, можно исключить компоненту ξ_2, складывая последнее равенство с предыдущим, умноженным на λ_1:

$$\lambda_1 A u^{2k} + A^2 u^{2k} \approx 2 \lambda_1^2 \beta_1 \xi_1.$$

При необходимости пронормировав это выражение, получим приближение к ξ_1. Аналогично будем иметь

$$\lambda_1 A u^{2k} - A^2 u^{2k} \approx -2\lambda_1^2 \beta_1 \xi_2.$$

$\triangleright_3$ Запишите соответствующий алгоритм приближенного вычисления λ_1, ξ_1, ξ_2.

3.2.4. Комплексные собственные значения

Покажем теперь как можно определить сразу два максимальных по модулю собственных значения, в том числе и комплексных. Пусть $|\lambda_1| \geqslant |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$ и в разложении начального приближения y^0 по собственным векторам коэффициенты при ξ_1 и ξ_2 отличны от 0. Тогда после достаточно большого числа итераций будем иметь

$$\begin{aligned} u^k &\approx \beta_1 \xi_1 + \beta_2 \xi_2 = \eta_0, \\ Au^k &\approx \lambda_1 \beta_1 \xi_1 + \lambda_2 \beta_2 \xi_2 = \eta_1, \\ A^2 u^k &\approx \lambda_1^2 \beta_1 \xi_1 + \lambda_2^2 \beta_2 \xi_2 = \eta_2. \end{aligned}$$

Возьмем по вектору с каждой стороны этих приближенных равенств и составим из них матрицы:

$$K = \left[u^k | Au^k | A^2 u^k\right] \approx \left[\eta_0 | \eta_1 | \eta_2\right]. \tag{3.5}$$

Векторы η_i принадлежат двумерному подпространству $\operatorname{span}\{\xi_1, \xi_2\}$ и, следовательно, линейно зависимы. Это означает, что существует вектор $c = (c_0, c_1, c_2)^T \neq 0$ такой, что

$$c_0\eta_0 + c_1\eta_1 + c_2\eta_2 = \left[\eta_0 | \eta_1 | \eta_2\right] c = 0.$$

Учитывая определение η_i, запишем это тождество в виде

$$\begin{aligned} \left[\eta_0 | \eta_1 | \eta_2\right] c = \underbrace{\left[\beta_1 \xi_1 \mid \beta_2 \xi_2\right]}_{B} \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 \\ 1 & \lambda_2 & \lambda_2^2 \end{bmatrix} c = \\ = B \begin{bmatrix} c_0 + c_1\lambda_1 + c_2\lambda_1^2 \\ c_0 + c_1\lambda_2 + c_2\lambda_2^2 \end{bmatrix} = B \begin{bmatrix} P(\lambda_1) \\ P(\lambda_2) \end{bmatrix} = 0, \end{aligned}$$

где

$$P(t) = c_2 t^2 + c_1 t + c_0.$$

Так как столбцы матрицы B линейно независимы, отсюда следует, что

$$P(\lambda_1) = P(\lambda_2) = 0,$$

то есть искомые собственные значения можно найти, решив квадратное уравнение $P(\lambda) = 0$. Осталось найти коэффициенты многочлена P. Они определены с точностью до постоянного множителя, поэтому можно положить $c_2 = 1$. Остальные коэффициенты найдем, используя исходное тождество (3.5):

$$Kc = c_0 u^k + c_1 A u^k + A^2 u^k \approx \left[\eta_0 | \eta_1 | \eta_2\right] c = 0,$$

Отсюда можно приближенно найти коэффициенты многочлена P путем решения задачи наименьших квадратов

$$(c_0, c_1)^T = \underset{c \in \mathbb{R}^2}{\operatorname{argmin}} \left\| \left[u^k \mid A u^k\right] c + A^2 u^k \right\|_2. \tag{3.6}$$

После этого, решая квадратное уравнение $P(t) = 0$, находятся приближения к искомым собственным значениям λ_1 и λ_2.

$\triangleright_4$ Как теперь можно найти приближения к собственным векторам ξ_1 и ξ_2?

3.2.5. Обратная итерация со сдвигом

Идею степенного метода можно использовать для определения любого, а не только максимального по модулю собственного значения. Рассмотрим скаляр $\mu \notin \sigma(A)$. Тогда матрица $(A - \mu I)^{-1}$ имеет такие же собственные векторы, как и A, а соответствующие собственные значения ее равны

$$\mu_i = (\lambda_i - \mu)^{-1}.$$

$\triangleright_5$ Докажите это.

Отсюда видно, что, применив к матрице $(A - \mu I)^{-1}$ степенной метод, мы можем найти ближайшее к μ собственное значение λ_i, а также соответствующий ему собственный вектор. При этом нет необходимости обращать матрицу $A - \mu I$: для вычисления $y = (A - \mu I)^{-1} u^k$ на каждой итерации нужно лишь решать СЛАУ

$$(A - \mu I) y = u^k.$$

Для снижения вычислительных затрат можно предварительно построить LU- или QR- разложение матрицы $A - \mu I$.

Обратная итерация с отношением Релея

...

3.3. *QR*-алгоритм

3.3.1. Общая схема *QR*-алгоритма

QR-алгоритм является одним из наиболее известных методов для вычисления всех (в том числе и комплексных) собственных значений произвольной квадратной матрицы. Суть алгоритма проста. Построим последовательность матриц

$$A_0 = A,\ A_1,\ A_2,\ \ldots,$$

по следующему правилу:

1) строится QR-разложение для матрицы A_k: $A_k = Q_k R_k$;
2) вычисляется $A_{k+1} = R_k Q_k$.

Описанное преобразование $A_k \mapsto A_{k+1}$ является преобразованием подобия:

$$A_{k+1} = R_k Q_k = Q_k^{-1} A_k Q_k,$$

поэтому все матрицы A_k подобны исходной матрице A.

Теорема 3.1 (QR-алгоритм, упрощенная формулировка)**.** *Последовательность матриц $\{A_k\}_{k=0}^{\infty}$, построенная по описанному выше правилу, при выполнении определенных условий*[2] *сходится к почти верхнетреугольной матрице $\tilde{R}$, у которой на диагонали стоят либо λ_i — собственные значения A, либо блоки вида $\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}$, собственные значения которых равны комплексно-сопряженным собственным значениям матрицы A.*

Замечание 3.2. Коэффициенты блоков $\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}$ не обязаны сходиться к какому-то пределу, сходятся их собственные значения.

Замечание 3.3. QR-алгоритм *никогда* не применяется к «неподготовленной» матрице A, потому что это слишком трудоемко. Сначала матрицу необходимо преобразованием подобия (желательно, ортогональным) привести к форме Хессенберга: $H = Q^T A Q$ и лишь потом начинать итерации QR-алгоритма. Это снизит трудоемкость одной итерации с $O(n^3)$ до $O(n^2)$.

[2] Более подробно результаты о сходимости QR-алгоритма обсуждаются в [5, с. 138].

3.3.2. Приведение к форме Хессенберга методом отражений

Нам уже известен один способ ортогонального приведения матрицы к форме Хессенберга — алгоритм Арнольди. Сейчас мы изучим еще один способ, более трудоемкий, но при этом обладающий лучшей вычислительной устойчивостью.

Преобразование матрицы A будем осуществлять по столбцам слева направо. Имеет место почти полная аналогия с процедурой построения QR-разложения из раздела 2.4.3. Различия будет всего два: во-первых, вместо верхнетреугольной теперь нам нужна хессенбергова матрица; во-вторых, нужно использовать преобразования подобия, то есть вместо отображений вида $A \mapsto Q_k A$ будут использоваться отображения $A \mapsto Q_k A Q_k^T$.

Замечание 3.4. Здесь мы используем стандартное обозначение для ортогональных матриц — Q_k. Необходимо понимать, что эти матрицы не имеют никакого отношения к ортогональным матрицам из QR-алгоритма, который был описан в п. 3.3.1.

Итак, на k-м шаге алгоритма в матрице A обнуляются элементы k-го столбца в строках от $k+2$ до n путем применения преобразования отражения, что соответствует применению отображения

$$A \mapsto Q_k A, \quad Q_k = \begin{bmatrix} I_k & 0 \\ 0 & H_k \end{bmatrix}. \tag{3.7}$$

Здесь I_k — единичная матрица размерности k, H_k — матрица отражения размерности $n-k$, $H_k = I - 2w_k w_k^T$. Преобразование Q_k изменяет в матрице A строки с номерами от $k+1$ до n. После применения этого преобразования необходимо домножить полученную матрицу справа на Q_k^T, чтобы сохранить спектр:

$$Q_k A \mapsto Q_k A Q_k^T.$$

Это преобразование изменит в матрице $Q_k A$ *столбцы* от $(k+1)$-го до n-го, гарантируя сохранение сделанных на предыдущих этапах нулевых элементов.

Рассмотрим для наглядности случай $n = 5$.

$$
\begin{aligned}
k=1:\quad & \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ * & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ * & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ * & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ * & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix} \xrightarrow{Q_1(\cdot)} \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix} \xrightarrow{(\cdot)Q_1^T} \begin{bmatrix} \mathrm{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathrm{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}; \\
k=2:\quad & \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & * & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & * & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & * & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix} \xrightarrow{Q_2(\cdot)} \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix} \xrightarrow{(\cdot)Q_2^T} \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathrm{x} & \mathrm{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathrm{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}; \\
k=3:\quad & \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & * & \mathrm{x} & \mathrm{x} \\ 0 & 0 & * & \mathrm{x} & \mathrm{x} \end{bmatrix} \xrightarrow{Q_3(\cdot)} \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & 0 & \mathbf{x} & \mathbf{x} \end{bmatrix} \xrightarrow{(\cdot)Q_3^T} \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathbf{x} & \mathbf{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathrm{x} & \mathbf{x} & \mathbf{x} \\ 0 & 0 & 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}.
\end{aligned}
$$

На этой схеме жирными крестиками обозначены элементы, которые были изменены в матрице в ходе последнего преобразования. Звездочки на k-м шаге обозначают элементы вектора a_k, который служит для определения преобразования отражения H_k из (3.7) по уже знакомой формуле (2.28):

$$w_k = (a_k - a'_k)/\|a_k - a'_k\|_2,$$

где $a'_k = (\pm\|a_k\|_2, 0, \ldots, 0)^T$, а знак первой компоненты вектора a'_k для повышения вычислительной устойчивости должен быть противоположен знаку первой компоненты вектора a_k. Напомним, что матрицы Q_k в явном виде не строятся, а определяются векторами нормали w_k.

$\triangleright_1$ Запишите полностью алгоритм приведения матрицы к форме Хессенберга методом отражений.

Замечание 3.5. Если исходная матрица A симметрична, то нетрудно заметить, что ее форма Хессенберга является симметричной трехдиагональной матрицей. Это обстоятельство позволяет существенно снизить вычислительные затраты.

$\triangleright_2$ Постройте соответствующий алгоритм.

3.3.3. Преобразование вращения

Для эффективного применения QR-алгоритма к матрице в форме Хессенберга необходимо познакомиться еще с одним видом ортогональных преобразований.

Определение 3.3. *Матрицей элементарного вращения* называется матрица вида

$$V_{pq}(\theta) = \begin{array}{c} \begin{matrix} & & & p & & q & & \end{matrix} \\ \left[\begin{array}{ccc|c|c|c|ccc} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ \hline & & & c & & s & & & \\ \hline & & & & \ddots & & & & \\ \hline & & & -s & & c & & & \\ \hline & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{array}\right] \begin{matrix} \\ \\ \\ p \\ \\ q \\ \\ \\ \end{matrix} \end{array}, \tag{3.8}$$

где $c = \cos\theta$, $s = \sin\theta$. Умножение такой матрицы на вектор $x \in \mathbb{R}^n$ эквивалентно повороту этого вектора на угол θ в плоскости, соответствующей координатам с номерами p и q.

$\triangleright_3$ Докажите, что матрица вращения ортогональна.

Заметим, что умножение на матрицу V_{pq} изменяет в произвольном $x \in \mathbb{R}^n$ только p-й и q-й элементы:

$$V_{pq} \begin{bmatrix} x_1 \\ \vdots \\ x_p \\ \vdots \\ x_q \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ c\,x_p + s\,x_q \\ \vdots \\ -s\,x_p + c\,x_q \\ \vdots \\ x_n \end{bmatrix}.$$

Сейчас мы рассмотрим как такие преобразования используются для построения QR-разложения хессенберговоой матрицы.

3.3.4. *QR*-разложение для формы Хессенберга

Пусть A имеет форму Хессенберга. Для приведения ее к верхнетреугольному виду достаточно обнулить $n-1$ элементов, находящихся под главной диагональю. Причем, как уже говорилось, соответствующие линейные преобразования должны быть ортогональными. Преобразования

вращения как нельзя лучше подойдут для этого:

$$\underbrace{V_{n-1,n}\dots V_{23}V_{12}}_{Q^{-1}}A = R, \tag{3.9}$$

где матрицы $V_{i,i+1} = V_{i,i+1}(\theta_i)$ имеют вид (3.8) и определяются лишь параметрами $s_i = \sin\theta_i$ и $c_i = \cos\theta_i$. Проиллюстрируем это схемой для случая $n = 4$:

$$\begin{bmatrix} c_1 & s_1 & & \\ -s_1 & c_1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\begin{bmatrix} \times & \times & \times & \times \\ * & \times & \times & \times \\ & * & \times & \times \\ & & * & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & * & \times & \times \\ & & * & \times \end{bmatrix},$$

$$\begin{bmatrix} 1 & & & \\ & c_2 & s_2 & \\ & -s_2 & c_2 & \\ & & & 1 \end{bmatrix}\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & * & \times & \times \\ & & * & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & * & \times \end{bmatrix},$$

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & c_3 & s_3 \\ & & -s_3 & c_3 \end{bmatrix}\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & * & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}.$$

Таким образом, QR-разложение матрицы Хессенберга определяется матрицей R и набором $\{c_i, s_i\}$, $i = \overline{1, n-1}$.

Согласно (3.9), имеем

$$Q = (V_{n-1,n}\dots V_{23}V_{12})^{-1} = V_{12}^T V_{23}^T \dots V_{n-1,n}^T,$$

поэтому для завершения итерации QR-алгоритма, т. е. для вычисления $A' = RQ$ нужно последовательно умножить полученную матрицу R справа на $V_{i,i+1}^T$ для i от 1 до $n-1$. Каждое такое преобразование будет изменять в матрице R столбцы с номерами i и $i+1$. Важно, что A' также будет иметь форму Хессенберга (иначе изначальное приведение A к данной форме было бы бессмысленным).

$\triangleright_4$ Докажите это утверждение.

Чтобы полностью определить алгоритм, осталось вывести формулы для параметров $\{c_i, s_i\}$, обеспечивающие обнуление «поддиагональных» элементов в ходе (3.9). Для этого достаточно рассмотреть случай 2×2:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha^2 + \beta^2} \\ 0 \end{bmatrix}.$$

Первая компонента правой части равна длине исходного вектора $[\alpha, \beta]^T$ в силу ортогональности преобразования вращения. Второе уравнение этой СЛАУ имеет вид

$$-s\alpha + c\beta = 0,$$

откуда сразу получаем $t = s/c = \beta/\alpha$, или

$$c = \frac{\alpha}{\sqrt{\alpha^2 + \beta^2}}, \quad s = \frac{\beta}{\sqrt{\alpha^2 + \beta^2}}. \tag{3.10}$$

$\triangleright_5$ Запишите псевдокод одной итерации QR-алгоритма для матрицы в форме Хессенберга. Оцените его сложность.

$\triangleright_6$ Выполните предыдущее задание для случая $A = A^T$ (трехдиагональная форма Хессенберга).

3.3.5. Детали реализации алгоритма

Итак, с учетом всего вышеизложенного мы можем записать следующее.

Примитивный *QR*-алгоритм

1) Матрица A приводится к форме Хессенберга A_0.
2) Выполняются итерации QR-алгоритма с начальной матрицей A_0 до тех пор, пока не достигается нужной точности.
3) Из полученной матрицы $A_N \approx \tilde{R}$ «извлекаются» собственные значения.

Обсудим данный алгоритм. Первый пункт нами рассмотрен (пп. 3.3.2). То, как выполнять QR-итерации с использованием преобразований вращения (пункт 2), мы обсудили в пп. 3.3.4.

Для полной ясности приведем пример «извлечения» собственных значений в пункте 3 (см. основную теорему 3.1). Пусть $n = 4$ и после выполнения необходимого числа итераций QR-алгоритма получена матрица

$$A_N = \begin{bmatrix} \alpha_1 & \times & \times & \times \\ 0 & \alpha_2 & \beta & \times \\ 0 & \gamma & \alpha_3 & \times \\ 0 & 0 & 0 & \alpha_4 \end{bmatrix}, \quad \gamma \neq 0.$$

Видно, что у матрицы два вещественных собственных значения: α_1 и α_4. Оставшаяся комплексно-сопряженная пара собственных значений

находится как решение характеристического уравнения

$$\begin{vmatrix} \alpha_2 - \lambda & \beta \\ \gamma & \alpha_3 - \lambda \end{vmatrix} = 0.$$

Замечание 3.6. В заключение необходимо отметить, что открытым остался вопрос о критерии остановке итераций в пункте 2. К сожалению, эффективное решение данного вопроса требует внесения в алгоритм дополнительных деталей, обоснование которых выходит за рамки нашего курса. Необходимо понимать также, что приведенная схема алгоритма является весьма упрощенной и недостаточно эффективной. Рациональная реализация QR-алгоритма является намного более сложной.

Библиографический список

[1] *Бахвалов, Н. С.* Численные методы в задачах и упражнениях : учеб. пособие / Н. С. Бахвалов, А. В. Лапин, Е. В. Чижонков. – М., 2000. – 190 с.

[2] *Богачев, К. Ю.* Практикум на ЭВМ. Методы приближения функций : учеб. пособие / К. Ю. Богачев, 3-е изд. – М., 2002. – 190 с.

[3] *Богачев, К. Ю.* Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений : учеб. пособие / К. Ю. Богачев, 2-е изд. – М., 1999. – 200 с.

[4] *Голуб, Дж.* Матричные вычисления : пер. с англ / Дж. Голуб, Ч. Ван Лоун. – М., 1999. – 548 с.

[5] *Икрамов, Х. Д.* Несимметричная проблема собственных значений. Численные методы / Х. Д. Икрамов. – М., 1991. — 240 с.

[6] Алгоритмы: Построение и анализ / Т. Кормен [и др.]; 2-е изд. М., 2012. – 1293 с.

[7] *Крылов, В. И.* Приближенное вычисление интегралов / В. И. Крылов. – М., 1967. – 500 с.

[8] *Крылов, В. И.* Вычислительные методы высшей математики : в 2 т./ В. И. Крылов, В. В. Бобков, П. И. Монастырный; Т. 1. Под ред. И. П. Мысовских. – Минск, 1972. – 584 с.

[9] *Ланцош, К.* Практические методы прикладного анализа : пер. с англ. / К. Ланцош — М., 1961. – 524 с.

[10] *Самарский, А. А.* Численные методы : учеб. пособие / А. А. Самарский, А. В. Гулин. – М., 1989. – 432 с.

[11] *Тыртышников, Е. Е.* Матричный анализ и линейная алгебра : учеб. пособие / Е. Е. Тыртышников. – М., 2007. – 480 с.

[12] *Уилкинсон, Дж.* Алгебраическая проблема собственных значений : пер. с англ. / Дж. Уилкинсон. – М., 1970. – 564 с.

[13] *Фаддеев, Д. К.* Вычислительные методы линейной алгебры / Д. К. Фаддеев, В. К. Фаддеева. – М., 1963. – 658 с.

[14] *Цехан, О. Б.* Матричный анализ : учеб. пособие / О. Б. Цехан. – Гродно, 2010. – 371 с.

[15] *Berrut, J.-P.* Barycentric lagrange interpolation / J.-P. Berrut, L. N. Trefethen // SIAM Rev. 2004. №46. P. 501–517.

[16] *Goldberg, D.* "What every computer scientist should know about floating-point arithmetic" / D. Goldberg // ACM Computing Surveys. 1991. Vol. 23 №1. P. 5-48.

[17] *Higham, N. J.* Accuracy and stability of numerical algorithms / N. J. Higham, 2nd ed. Philadelphia, 2002. – 680 с.

[18] Numerical Recipes in C: the art of scientific computing / W. H. Press [et al.]. — 2nd ed. Cambridge, 1997. – 994 p.

[19] *Saad, Y.* Iterative methods for sparse linear systems / Y. Saad- 2nd ed. Philadelphia, 2003. – 528 p.

[20] *Sewell, G.* Computational Methods of Linear Algebra / G. Sewell. – 2nd ed. New Jersey, 2005. – 268 p.

[21] *Trefethen, L. N.* Numerical Linear Algebra / L. N. Trefethen, D. Bau. Philadelphia, 1997. – 373 p.